

UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

DEEP LEARNING METHODS
APPLIED TO HIGGS PHYSICS AT THE LHC

Giles Chatham Strong

Supervisor: Doctor Michele Gallinaro

Thesis approved in public session to obtain the PhD Degree in Physics

Jury final classification: Pass with Distinction and Honour



TÉCNICO
LISBOA

UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

DEEP LEARNING METHODS
APPLIED TO HIGGS PHYSICS AT THE LHC

Giles Chatham Strong

Supervisor: Doctor Michele Gallinaro

Thesis approved in public session to obtain the PhD Degree in Physics

Jury final classification: Pass with Distinction and Honour

Jury

Chairperson: Doctor Jorge Manuel Rodrigues Crispim Romão, Instituto Superior Técnico, Universidade de Lisboa.

Members of the committee:

Doctor João Manuel Coelho dos Santos Varela, Instituto Superior Técnico, Universidade de Lisboa;

Doctor João Carlos Carvalho de Sá Seixas, Instituto Superior Técnico, Universidade de Lisboa;

Doctor Tommaso Dorigo, Instituto Nazionale di Fisica Nucleare (INFN), Itália;

Doctor Nuno Filipe da Silva Fernandes de Castro, Escola de Ciências, Universidade do Minho;

Doctor Michele Gallinaro, Instituto Superior Técnico, Universidade de Lisboa.

Funding institutions:

Work contained in this thesis received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 675440.

2020

Abstract

The impact that machine learning (ML) has had on research in high-energy physics (HEP) is undeniable; the use of ML-based classifiers in many analyses is now the norm, and they have a long history of being used within reconstruction algorithms (e.g. b -tagging at LEP, 1995). The combined effect of advances in both hardware and algorithms, and the vast datasets now available (both collected and simulated), has facilitated the paradigm shift towards deep learning (DL). The true effect of this is only beginning to be felt in the field of HEP, but it is something that can be expected to continue into the future; overturning several “foundational” principles, and challenging more traditional thinking. Over the course of my PhD I have developed and applied a range of advanced ML and DL algorithms to the study of the fundamental interactions of matter. This thesis documents over four and a half years of work; beginning from preliminary feasibility studies, through algorithm optimisation studies and dedicated software development, and concluding with a full study leveraging the refined methods.

From a pragmatic perspective, no matter how *beautiful* a new method is, it is only applicable if either it provides better performance than alternative methods, or it saves the applicer time and effort. Preferably, the new method should meet both of these criteria and throughout this thesis I test DL methods against them. As a context for studying these new methods, I pick the sub-field of Higgs physics. With the discovery of the Higgs boson in 2012, this area has now moved into the regime of precision measurements and experimental confirmation of theoretically predicted channels of production and decay; any deviation from the theoretical model (the Standard Model (SM)) could be a sign of new physics, and open up further avenues for progressing humanity’s understanding of Nature. The importance of these studies, therefore, goes far beyond demonstrating the applicability of new methods.

One key focus of my research is the search for the simultaneous production of two Higgs bosons. One property of the Higgs boson that has yet to be measured is the strength with which they couple to themselves (the Higgs interacts with particles that have mass, and therefore is expected to interact with itself). Beyond academic interest, the value of

this parameter can allow us to more precisely estimate the stability of the Universe and determine whether we are at risk of suddenly being snuffed from existence (as we know it) by spontaneous vacuum decay. The production of such *di-Higgs* processes, however, is extremely rare, meaning that whilst we are unlikely to discover it at the LHC (barring the effects of new physics), it is a perfect testing ground for DL methods; even moderate improvements in sensitivity, through whatever means, can correspond to months or years of less data-taking required to achieve discovery.

At the time of writing, the Large Hadron Collider (LHC) has finished its Run-II data-taking and three years worth of $\sqrt{s} = 13$ TeV proton-proton collision-data (137.19 fb⁻¹ of integrated luminosity) stands ready to be scrutinised in minute detail for any hints of new phenomena, such as super-SM rates of production of di-Higgs. The concluding chapter of this thesis documents my contributions to the examination of this extremely large dataset.

Particular highlights of my thesis are summarised below.

Advanced methods for deep learning New methods and techniques of DL are constantly being developed, and they are usually presented in terms providing performance improvements benchmark datasets. Some of these approaches, though, have the potential to not only improve performance, but also to decrease the time required to train models to the same level of performance. In the chapter “Advanced methods in deep learning” I use the data from the 2014 Higgs ML Kaggle challenge as a HEP-specific benchmark dataset in order to evaluate the advantages of a range of recent methods. The Higgs ML challenge is representative of a typical HEP-search in terms of metric and dataset size, and the original challenge received the attention of not only physicists but also data-scientists and computer scientists. The winning solution, therefore, forms a strong baseline against which to compare, and was based on a large ensemble of deep neural networks (DNNs).

Using a smaller ensemble of DNNs, I was able to match the performance of the winning solution through a combination of: HEP-specific data augmentation, to artificially increase the training dataset size and improve test-time generalisation; a connection method that improves the efficiency of information flow in the network; a more modern activation function which provides an incremental benefit; and a dynamic adjustment of the optimiser’s hyper-parameters during training, which significantly reduces the training time for the networks. The main advantage, then, is that the solution developed can be trained

in a fraction of the time of the winning solution: 13 times quicker on GPU accounting for hardware improvements; and seven times quicker on a standard laptop CPU. Application times of my solution are also significantly reduced compared to the winning solution. The fact that the solution can be trained and applied using laptop CPU is particularly important as it makes the usage of DNNs much more viable in HEP, particularly for researchers without access to the powerful GPUs which are usually required.

Automated feature selection Often in classification or regression tasks in HEP, or indeed many other problem-domains, one has, or can compute, a large range of possible input features for ML models from the available data; more than is expected to be necessary to complete the task satisfactorily. To avoid the possibility of overfitting to the data, it is often advantageous to whittle down the menu of input features to a reduced set of only those that offer information relevant to the particular task, and contain minimal overlap in the information they each provide. There are several ways to approach this task. The most thorough, but intensive, being a brute-force tests of all combinations of possible input features. Such an approach, though, is often not viable due to the time required, and the risk of inadvertently overfitting to the validation data during feature selection.

Throughout the course of my PhD I gradually refined a series of feature-selection steps into a comparatively quick, and (so far) reliable automated process controlled by a few intuitive hyper-parameters. Over the course of several stages, non-generalising features are removed, as are those with a significant information overlap with other features, and finally the selection of consistently important features is made.

The process relies heavily on Random Forest (RF) models, which can be quickly fitted to the data without requiring preprocessing, and can also be interpreted to compute the importance of each feature. Feature removal involves first ensuring that the removal of the candidate feature does not result in a reduction of the mean performance of the model. RFs are also used as regressors to find which features can be accurately predicted from the other features. The main hyper-parameter for the process is the level of feature importance required to be selected. This process has been used now in a variety of different projects, and has been able to reduce large menus of features down to only the most relevant set, which have then been used to train well-performing models.

LUMIN Over the course of my PhD, I gradually accumulated a library of my own custom methods, which I frequently used. Eventually this grew to the point where I began thinking about sharing it with others and opening it up to community contributions. Before doing so, I first rewrote the library from scratch, using `PYTORCH` to provide the network implementations and tensor manipulations on both CPUs and GPUs (previously it has used `KERAS`). This library, **LUMIN**, was made available under an open-source software licence in early 2019, and is still under development and gradually attracting the interest and contributions from the HEP community.

The library itself provides modular implementations for networks and DL methods, allowing users to adjust and them to their specific needs, as well a range of other useful classes and methods for data manipulation, statistics, and plotting. One main focus of the library is the application of these methods to HEP data.

LUMIN was used to perform my investigation of the Higgs ML Kaggle challenge, and also contains implementations of the automated feature-selection process. My thesis contains a dedicated section discussing the software library and its distinguishing characteristics.

The CMS search for di-Higgs production The culmination of my thesis concerns the CMS search of di-Higgs production. Here, all the methods I've spent the thesis researching and developing are applied to the task of classifying this rare signal in Run-II data from the LHC. Using the predictions of discriminators trained to perform binary classification of the signal and background as a test statistic, 95 % confidence upper-limits are placed on the cross-section of non-resonant di-Higgs production, and the ratio of the Higgs self-coupling to the SM prediction is constrained.

Keywords: Higgs boson, machine learning, data science, LHC, CMS

Título: Métodos de aprendizagem automática aplicados à física de Higgs no LHC

Nome: Giles Chatham Strong

Doutoramento em: física

Orientador: Michele Gallinaro

Resumo alargado

O impacto que a aprendizagem automática (machine learning, ML) tem tido na investigação em física das altas energias (HEP) é inegável; o uso de classificadores baseados em ML em muitas análises é agora a norma, e têm uma longa história de emprego em algoritmos de reconstrução (por exemplo, b -tagging no LEP, 1995). O efeito combinado de avanços em hardware e algoritmos, e os vastos conjuntos de dados agora disponíveis (adquiridos e simulados) facilitaram a mudança de paradigma em direção a deep learning (DL). O verdadeiro efeito disso começa apenas a ser sentido no campo de HEP, mas é algo que se pode esperar que continue no futuro; derrubando vários princípios “fundamentais” e desafiando o pensamento mais tradicional. Ao longo do meu doutoramento, desenvolvi e apliquei uma série de algoritmos avançados de ML e DL ao estudo das interações fundamentais da matéria. Esta tese documenta mais de quatro anos e meio de trabalho; começando com estudos preliminares de viabilidade, através de estudos de otimização de algoritmos e desenvolvimento de software dedicado, e concluindo com um estudo completo baseado na versão refinada dos métodos.

De uma perspectiva pragmática, não importa quão *bonito* seja um novo método, ele só é aplicável se fornecer melhor desempenho do que métodos alternativos ou economizar tempo e esforço ao utilizador. De preferência, o novo método deve atender a ambos os critérios e, ao longo desta tese, testarei métodos de DL nestes aspectos. Como contexto para estudar esses novos métodos, escolho o subcampo da física do bóson de Higgs. Com a descoberta do bóson de Higgs em 2012, esta área agora transitou para um regime de medições de precisão e confirmação experimentais de canais teoricamente previstos de produção e decaimento; qualquer desvio do modelo teórico (o Modelo Padrão, SM) pode ser um sinal de nova física e abrir novos caminhos para o progresso da compreensão da natureza pela humanidade. A importância desses estudos, portanto, vai muito além de demonstrar a aplicabilidade de novos métodos.

Um dos principais focos de minha investigação é a busca pela produção simultânea de dois bósons de Higgs. Uma propriedade do bóson de Higgs que ainda não foi medida é a magnitude do acoplamento consigo próprio (o Higgs interage com partículas que têm massa e, portanto, espera-se que interaja consigo mesmo). Além do interesse acadêmico, o valor desse parâmetro pode-nos permitir estimar com mais precisão a estabilidade do Universo (tal como o conhecemos) e determinar se corremos o risco de ser subitamente extintos devido ao decaimento espontâneo do vácuo. A produção de tais processos *di-Higgs*, entretanto, é extremamente rara, o que significa que embora seja improvável que possamos descobri-los no LHC (excluindo efeitos da nova física), é um campo de teste perfeito para métodos de DL; mesmo melhorias moderadas na sensibilidade, por quaisquer meios, podem corresponder a menos meses ou anos de aquisição de dados necessários para alcançar a descoberta.

No momento em que esta tese é escrita, o Large Hadron Collider (LHC) terminou a sua aquisição de dados do Run II, envolvendo três anos de colisões próton-próton a 13 TeV (137.19 fb^{-1} de luminosidade integrada), que estão prontos para ser examinados em detalhe minucioso tendo em vista quaisquer dicas de novos fenômenos, como taxas de produção de diHiggs para além do SM. O capítulo final desta tese documenta as minhas contribuições para a análise deste conjunto de dados extremamente grande.

Destaques particulares de minha tese são resumidos de seguida.

Métodos avançados de aprendizagem automática Novos métodos e técnicas de ML estão constantemente a ser desenvolvidos, e são geralmente apresentados em termos de melhorias de desempenho alcançadas em conjuntos de dados de referência. Algumas dessas abordagens, entretanto, têm o potencial não apenas de melhorar o desempenho, mas também de diminuir o tempo necessário para treinar modelos para o mesmo nível de desempenho. No capítulo “Métodos avançados de aprendizagem automática”, utilizo os dados do desafio Higgs ML Kaggle de 2014 como um conjunto de dados de referência específicos de HEP, com vista a avaliar as vantagens de uma série de métodos recentes. O desafio Higgs ML é representativo de investigação típica em HEP em termos de métrica e tamanho do conjunto de dados, e o desafio original recebeu a atenção não apenas de físicos, mas também de cientistas de dados e cientistas de computação. A solução vencedora, portanto, forma uma forte linha de base para comparação e foi baseada num grande conjunto de redes neurais profundas (DNNs).

VIII

Usando um conjunto menor de DNNs, consegui igualar o desempenho da solução vencedora por meio de uma combinação de: métodos de aumento de dados (data augmentation) específicos de HEP, para aumentar artificialmente o tamanho do conjunto de dados de treino e melhorar a generalização do tempo de teste; um método de conexão que melhora a eficiência do fluxo de informações na rede; uma função de ativação mais moderna que fornece benefícios incrementais; e um ajuste dinâmico dos hiper-parâmetros do otimizador durante o treino, o que reduz significativamente o tempo de treino das redes. A principal vantagem, então, é que a solução desenvolvida pode ser treinada numa fração do tempo da solução vencedora: 13 vezes mais rápido em GPUs considerando melhorias de hardware; e sete vezes mais rápido em CPUs padrão. Os tempos de aplicação da minha solução também são significativamente reduzidos em comparação com a solução vencedora. O facto de a solução poder ser treinada e aplicada usando CPUs de computadores portáteis é particularmente importante, pois torna o uso de DNNs muito mais viável em HEP, principalmente para investigadores sem acesso às poderosas GPUs que normalmente são necessárias.

Seleção automatizada de parâmetros Frequentemente, em tarefas de classificação ou regressão em HEP, ou mesmo em muitos outros domínios de problemas, alguém tem, ou pode calcular, uma grande variedade de parâmetros de entrada possíveis para modelos de ML a partir dos dados disponíveis; mais do que o que se espera ser necessário para completar a tarefa de maneira satisfatória. Para evitar a possibilidade de ajuste excessivo aos dados, muitas vezes é vantajoso reduzir a lista de parâmetros de entrada a um conjunto reduzido de apenas aqueles que oferecem informação relevante para a tarefa específica e que contêm sobreposição mínima da informação que cada um fornece. Existem várias maneiras de abordar essa tarefa. A mais completa, mas intensiva, é um teste de força bruta de todas as combinações de parâmetros de entrada possíveis. Essa abordagem, no entanto, muitas vezes não é viável devido ao tempo necessário e ao risco de ajuste excessivo inadvertido para os dados de validação durante a seleção de parâmetros.

Ao longo do meu doutoramento, melhorei gradualmente uma série de etapas de seleção de parâmetros num processo automatizado comparativamente rápido, e (até agora) confiável, controlado por alguns hiper-parâmetros intuitivos. Ao longo de vários estágios, os parâmetros não generalizantes são removidos, assim como aqueles com uma sobreposição significativa de informações com outros parâmetros e, finalmente, é feita a seleção de

parâmetros consistentemente importantes.

O processo depende muito de modelos Random Forest (RF), que podem ser ajustados rapidamente aos dados sem exigir pré-processamento e também podem ser interpretados para calcular a importância de cada parâmetro. O processo de remoção de parâmetros envolve primeiro garantir que a remoção do parâmetro candidato não resulte numa redução do desempenho médio do modelo. RFs também são usadas como regressores para descobrir quais parâmetros podem ser previstos com precisão a partir de outros parâmetros. O principal hiper-parâmetro para o processo é o nível de importância do parâmetro necessário para ser selecionado. Esse processo tem sido usado recentemente numa variedade de projetos diferentes, e foi capaz de reduzir grandes listas de parâmetros a apenas o conjunto mais relevante, que têm de seguida sido usados para treinar modelos com bom desempenho.

LUMIN Ao longo do meu doutoramento, gradualmente acumulei uma biblioteca dos meus próprios métodos adaptados, que usei frequentemente. Eventualmente, tal cresceu a ponto de começar a pensar em compartilhá-lo com outras pessoas e abri-lo para contribuições da comunidade. Antes de fazer isso, primeiro reescrevi a biblioteca do zero, usando PYTORCH para fornecer as implementações de rede e manipulações de tensores em CPUs e GPUs (anteriormente, usava KERAS). Esta biblioteca, LUMIN, foi disponibilizada sob uma licença de software de código aberto no início de 2019, e ainda está em desenvolvimento e atrai gradualmente o interesse e contribuições da comunidade de HEP.

A biblioteca em si fornece implementações modulares para redes e métodos de DL, permitindo que os utilizadores os ajustem às suas necessidades específicas, bem como uma variedade de outras classes e métodos úteis para manipulação de dados, análise estatística, e visualização. Um dos principais focos da biblioteca é a aplicação desses métodos aos dados de HEP.

LUMIN foi usado para realizar a minha investigação no contexto do desafio Higgs ML Kaggle e também contém implementações do processo de seleção automatizada de parâmetros. A minha tese contém uma secção dedicada discutindo a biblioteca de software e as suas características distintivas.

A procura de produção de di-Higgs em CMS O ponto culminante da minha tese diz respeito à investigação realizada em CMS da produção de di-Higgs. Aqui, todos os

métodos que investiguei e desenvolvi ao longo da tese são aplicados à tarefa de classificar esse sinal raro em dados de Run II do LHC. Usando as previsões de discriminadores treinados para realizar a classificação binária do sinal e do fundo como uma estatística de teste, limites superiores, com nível de confiança de 95 %, são atribuídos à secção eficaz de produção di-Higgs não ressonante, e a razão do autoacoplamento do Higgs relativamente à previsão do SM é restringida.

Palavras-chave: Bosão de Higgs, aprendizagem automática, ciência de dados, LHC, CMS

Acknowledgements

The research documented in my thesis was carried out between 2016 and 2020, whilst part of the Compact Muon Solenoid (CMS) collaboration at the European Organization for Nuclear Research (CERN). Between March 2016 and March 2019 I was employed by the CMS group at Laboratório de Instrumentação e Física Experimental de Partículas (LIP) in Portugal as an early-stage researcher as part of the AMVA4NewPhysics Marie Skłodowska-Curie innovative training network funded by the Horizon2020 program of the European Commission under grant agreement number 675440. This funding also covered travel expenses and training courses. From March 2019, my PhD related research and expenses were self-funded.

Personally, I am thankful to everyone who has helped me get to this point: to my family for their support; to my teachers, in both academia and in life, for their guidance; and to my friends for their encouragement.

I am grateful for the supervision, direction, and advice of Michele Gallinaro, who oversaw the completion of my PhD, whilst at the same time allowing me the freedom to explore the many possible areas of research in particle physics and machine learning. Thanks also for his invaluable feedback and reviews of the many papers and presentations I've sent his way over the years.

Working as part of the AMVA4NewPhysics ITN has been an amazing experience, and I am thankful for the friendships and connections I have built up through it, and the opportunity to work alongside, and hangout with, some excellent researchers. I aim to put to good use, and pass on, the many skills I have acquired during the schools and workshops I was able to attend due to the network. Particular thanks to Tommaso Dorigo for successfully coordinating the ITN and ensuring that all deliverables were met well within time. During my involvement in AMVA4NewPhysics I attended several secondments and I am grateful for the hosts and teams there: Tommaso and the INFN-Padova CMS group; Giovanna Menardi and Bruno Scarpa at the University of Padova statistics department; Andrey Ustyuzhanin and the Yandex Lambda group in Moscow.

Thanks to the CMS group at LIP, headed by João Varela and Michele, for their useful feedback, comments, and discussions over the years. Thank you to my fellow LIP students for helping the lab grow to be more open and welcoming through the various organised activities and outings. I would like to thank Nuno Castro for providing me with numerous

opportunities to grow and develop as a researcher through the LIP Big Data Competence Centre. Thanks to Nuno Leonardo for leading the organisation of the annual LIP summer students' projects, and to the seven students I helped supervise over the years, who allowed me to grow as a teacher; particularly João Gonçalves for his insightful and challenging questions and discussions. I appreciate the hard work of the LIP admin staff, especially João Pedro for helping with the organisation of the many trips I undertook and for his weekly Portuguese lessons.

This thesis contains contents from one of my publications (the chapter on the Higgs ML challenge). Acknowledgements regarding this sections are as follows: I am very grateful to Michele Gallinaro, Nuno Castro, and João Gonçalves for their valued feedback and comments in reviewing the paper, and to the other members of the AMVA4NewPhysics ITN for all the discussions over the years, which no doubt helped to contribute to this work. Additional thanks also to the fast.ai team whose courses introduced me to many of the ideas tested, as well as inspired me to begin writing my own deep learning framework. I would also like to thank Tommaso Dorigo and David Rousseau for the useful discussions, as well the two referees of the manuscript for their thorough and helpful recommendations.

The final part of this thesis documents CMS work which would not have been possible without the hard work of many others in both CMS and CERN. Thanks to everyone involved in the CMS $hh \rightarrow \tau^+ \tau^- b \bar{b}$ search, and in particular Konstantin Androsov, Francesco Brivio, and Luca Cadamuro for their help in getting me up and running with the software, for processing results, and helping to organise the various analyses. Luca Cadamuro was also a great help with the HL-LHC projection study.

Lastly, am thankful to Fausto Carriço and my fellow students at the Oriental Branch of Shorinji Kempo for providing a familiar and welcoming family when I arrived in Portugal not knowing anybody.

Dedication

To my mother, father, and brother for always supporting, encouraging, and believing in me.

Contents

Abstract	II
Resumo	VI
Acknowledgements	XI
1 Introduction	1
1.1 Thesis overview	1
1.2 Machine learning in high-energy physics	2
1.2.1 The growth of machine learning in high-energy physics	2
1.2.2 Introduction to searches at the LHC	4
1.3 The AMVA4NewPhysics European network	5
2 The Standard Model of particle physics	9
2.1 Electroweak sector	9
2.1.1 Leptons	9
2.1.2 Quarks	10
2.1.3 EW bosons	11
2.2 Quantum Chromodynamics	12
2.2.1 Quarks	12
2.2.2 Gluons	13
2.2.3 QCD interactions	14
2.2.4 Hadrons	15
2.3 The Higgs mechanism	16
2.3.1 Spontaneous symmetry breaking	17
2.3.2 Yukawa terms	18
2.3.3 Higgs interactions	18
3 Particle physics at the LHC	21
3.1 Assorted definitions of relevant physics terminology	21
3.1.1 Natural units	21
3.1.2 Luminosity	21
3.1.3 Coordinate systems	22
3.2 The Large Hadron Collider	23

3.3	The CMS detector	24
3.3.1	Superconducting solenoid	25
3.3.2	Inner tracker	26
3.3.3	Calorimeters	27
3.3.4	Muon system	29
3.3.5	Forward detectors	30
3.3.6	Trigger system	30
3.3.7	The MIP timing detector and other high-luminosity upgrades . . .	31
3.4	Event reconstruction	34
3.4.1	Tracking	34
3.4.2	Particle flow	35
3.4.3	Physics objects	36
3.4.4	Jet tagging	40
3.4.5	Post-reconstruction algorithms	45
3.5	Monte Carlo simulation of particle collisions	53
3.5.1	Overview	53
3.5.2	Cross section computation	53
3.5.3	Parton distribution functions	54
3.5.4	The matrix element	54
3.5.5	The parton shower	56
3.5.6	Matrix element and parton shower combination	57
3.5.7	Hadronisation	58
3.5.8	Multi-parton interaction	58
3.5.9	Detector simulation	58
4	Statistical methods for discoveries	59
4.1	Hypothesis testing	59
4.1.1	Simple-hypothesis testing	59
4.1.2	Composite hypotheses	61
4.1.3	The likelihood ratio as a test statistic	61
4.2	Discovery significance	63
4.3	Confidence limits	63
4.4	Expected performance	64
5	Machine learning and data science	67
5.1	Overview of supervised learning	67
5.1.1	Overview of machine learning	67
5.1.2	Supervised learning	68
5.1.3	Columnar data format	70
5.2	Neural networks	70
5.2.1	Functional approximation	70

5.2.2	Neural networks	71
5.2.3	Optimisation through backpropagation	73
5.2.4	Components of neural networks	74
5.3	Random Forests	82
5.3.1	Decision-tree learning	82
5.3.2	Random Forests	86
5.4	Assorted data-science techniques	87
5.4.1	Pre-processing	88
5.4.2	Stratified splitting	88
5.4.3	Cross validation	88
5.4.4	Ensembling	89
6	The search for di-Higgs production	91
6.1	Motivation	91
6.1.1	Probing the Higgs potential - the κ_λ framework	91
6.1.2	The necessity of di-Higgs measurements	93
6.1.3	Resonant di-Higgs production	95
6.2	Production mechanisms	100
6.3	The $\tau^+ \tau^- b \bar{b}$ decay channel	103
6.3.1	Higgs decay channels	103
6.3.2	Di-Higgs decay channels	103
6.3.3	$h h \rightarrow \tau^+ \tau^- b \bar{b}$ sub-categories and decay channels	104
6.3.4	Major background processes to the $h h \rightarrow \tau^+ \tau^- b \bar{b}$ decay channel	106
6.4	Basic analysis chain	109
6.5	Current state of the art	109
7	Regression in di-Higgs searches	115
7.1	Overview	115
7.1.1	Event reconstruction in $\tau^+ \tau^- b \bar{b}$	115
7.1.2	Regression in $\tau^+ \tau^- b \bar{b}$	116
7.1.3	Feature optimisation via regression	117
7.1.4	Regression approaches	117
7.2	Datasets, event selection, and feature definitions	118
7.2.1	Training samples	118
7.2.2	Object definitions	119
7.2.3	Event selection	120
7.2.4	Feature definition	121
7.3	Regressor architecture	128
7.4	Higgs-boson mass regression	128
7.4.1	Pair-object momenta-regression	128
7.4.2	Single-object regression	133

7.5	Di-Higgs mass regression	136
7.5.1	Features	136
7.5.2	Performance	136
7.6	Summary of regression studies	138
8	Classification in di-Higgs searches	141
8.1	Overview	141
8.1.1	The aim of event classification	141
8.1.2	Classification approaches	142
8.2	Classification of di-Higgs signal with MEM	143
8.2.1	Overview	143
8.2.2	Signal weights	144
8.2.3	Background weights	145
8.2.4	Discriminator construction	146
8.2.5	Discussion	148
8.3	Classification of di-Higgs signal with DNNs	148
8.3.1	Overview	148
8.3.2	The classifier	148
8.3.3	Verification of feature optimisation	149
8.3.4	Feature selection	150
8.3.5	Performance	151
8.3.6	Summary of classification studies	153
8.4	Inference of di-Higgs signal	153
8.4.1	Cut and count	153
8.4.2	Shape analysis	156
8.4.3	Discussion	157
8.5	Conclusion	157
8.6	Retrospective thoughts on the preliminary investigations	157
9	Advanced methods in deep learning	161
9.1	Overview	161
9.2	The HiggsML dataset	162
9.3	Higgs ML challenge	162
9.3.1	Challenge description	163
9.3.2	Current state of the art	165
9.3.3	ATLAS geometry and coordinate system	165
9.3.4	Data and processing	165
9.3.5	Spirit of the investigation	167
9.3.6	Reported metrics	168
9.4	Method testing	169
9.4.1	Machinery	169

9.4.2	Feature selection	170
9.4.3	Baseline model	177
9.4.4	Categorical entity embedding	178
9.4.5	Data symmetry exploitations	181
9.4.6	Stochastic gradient descent with warm restarts	184
9.4.7	Activation function	187
9.4.8	Advanced ensembling	188
9.4.9	Super-convergence	191
9.4.10	Densely connected networks	193
9.4.11	Parallel networks and feature splitting	197
9.4.12	Architecture optimisation	205
9.5	Solution evaluation	211
9.5.1	Test results	211
9.5.2	Solution timing	212
9.5.3	Discussion	212
9.6	Benefit and ablation studies	217
9.6.1	Individual benefits	217
9.6.2	Ensemble weighting	217
9.7	Conclusion	218
9.8	Software package: LUMIN	219
9.9	Case study: HL-LHC projections	228
9.9.1	Motivation	228
9.9.2	Datasets and event selection	229
9.9.3	Neural network discriminator training	229
9.9.4	Results	229
9.9.5	Impact of the MIP timing detector on analysis sensitivity	236
10	The search for di-Higgs production at CMS	239
10.1	Introduction	239
10.2	Datasets	240
10.2.1	Collider data	240
10.2.2	Simulated samples	240
10.3	Event selection	243
10.3.1	High-level trigger requirements	243
10.3.2	Lepton selection	243
10.3.3	Channel assignment	243
10.3.4	Third-lepton veto	245
10.3.5	τ_h selection	245
10.3.6	MC-correction categories	245
10.3.7	Jet selection	246
10.3.8	Jet categorisation	247

10.3.9	Mass cuts	247
10.3.10	Regional categorisation	249
10.3.11	Process contributions	249
10.4	Collider-data-driven background samples and corrections	251
10.4.1	QCD multi-jet	251
10.4.2	Drell-Yan background corrections	252
10.5	Analysis and inference approach	252
10.5.1	Non-resonant search	252
10.5.2	Resonant search	253
10.5.3	Systematic uncertainties	254
10.6	Feature definitions	256
10.6.1	Approach	256
10.6.2	Higgs and di-Higgs construction	256
10.6.3	Feature definitions	257
10.7	Methodologies for automated feature-selection	259
10.7.1	Constant features	260
10.7.2	Subsampling	260
10.7.3	Random Forest hyper-parameter optimisation	260
10.7.4	Monotonic relationships	261
10.7.5	Feature importance	263
10.7.6	Removal by mutual dependence	264
10.8	General approach to discriminator training	271
10.8.1	General approach	271
10.8.2	Architecture and training	271
10.8.3	Parametrised learning	272
10.8.4	Feature pre-processing	273
10.8.5	Weighted training	273
10.9	Non-resonant search	274
10.9.1	Input features	274
10.9.2	Interpretation	275
10.9.3	Binned predictions	275
10.9.4	Inference and interpretation	276
10.10	Resonant search	284
10.10.1	Parameterised model	284
10.10.2	Non-parameterised model	290
10.10.3	Discussion	292
10.11	Conclusion	295

11 Summary and conclusions

297

12 Appendices	325
12.1 A study of the SuperTML result	325
12.1.1 Pretraining and transfer learning	325
12.1.2 The SuperTML method	326
12.1.3 Re-implementation of the method	326
12.1.4 HiggsML tests	327
12.1.5 Closing remarks	328
12.2 Software details	332

Chapter 1

Introduction

1.1 Thesis overview

This thesis documents my research into the development and application of machine learning approaches in high-energy physics. It focusses in particular on the search for di-Higgs production at the CMS detector at the Large Hadron Collider based at CERN, and how the sensitivity of this search can be extended through the use of advanced techniques. Particular applications include: regression of kinematic properties of particles via neural networks; classification of signal and background classes of collision events; state-of-the-art training methodologies and progressive architectures for neural networks; and approaches to automated feature-selection.

I'll begin with a short essay (Sec. 1.2) that aims to introduce the reader to the current status, and possible applications of, machine learning in high-energy physics, as well as motivate parts of my research. Following this I will describe the context of my PhD within AMVA4NewPhysics, a research and training network (Sec. 1.3).

In Chap. 2 I will describe the Standard Model, our current theory of particle interactions at the fundamental level, and in Chap. 3 the experimental and computational systems which allow us to test this theory. Chapter 4 discusses several statistical methods necessary for comparing our theories to experimental data. An introduction to machine learning is given in Chap. 5, which focusses mainly on *supervised learning* with neural networks and random forests, but also explains a few basic concepts for data science.

With the necessary concepts introduced, Chap. 6 motivates the search for di-Higgs production as a means of more confidently formulating aspects of the Standard Model which cannot be completely specified by theory alone. In reviewing the current state of the art for these analyses, the chapter implicitly motivates the use of bleeding edge techniques in the challenging search for this rare process.

Chapters 7 and 8 document the first year and a half of my PhD research, in which I developed and tested the use of neural networks for improving the reconstruction of the

Higgs and di-Higgs masses, and various means of classifying di-Higgs production against a large background.

Having confirmed that neural networks were applicable to the di-Higgs search, in Chap. 9 I work on testing the benefits of using a variety of state-of-the-art training techniques and architectures for neural networks. For testing I use data from the Higgs ML Kaggle challenge, which provides strong baselines for performance and timing.

With a much improved architecture and training scheme in place, the thesis moves into the final study (Chap. 10), in which I document the results of applying the various techniques built up so far to the official CMS search for di-Higgs production, using data collected during 2016, 2017, and 2018.

1.2 Machine learning in high-energy physics

1.2.1 The growth of machine learning in high-energy physics

The rise of machine-learning (ML) applications has had a remarkable impact on many areas of industry and in a few academic disciplines. The increase in data availability, and computational power, along the development of more advanced algorithms, now allows the solving of challenging tasks to be delegated to adaptable models which learn in a data-driven mode of operation, rather than relying on theoretical models and solutions handcrafted by experts. The field of experimental high-energy physics (HEP), however, was comparatively slower to adopt these new approaches, with only the occasional and isolated use of basic *multivariate analyses*, such as b -jet tagging at LEP e.g. [1] in 1995, and D0’s observation of single-top-quark production [2] in 2001; in the latter, the authors explicitly noted the importance of their MVA (a neural network (NN), see Sec. 5.2) in fully utilising all available data to make the observation. Indeed, the importance and potential benefits of these methods was well recognised within the HEP community, as indicated by Ref. [3] (2003), which notes “The best use of data is ensured only with a multivariate treatment” when discussing approaches for data-analysis by the CDF [4] and D0 [5] collaborations during Run-II of the Tevatron (2001-2011).

This process of adoption continued at the Large Hadron Collider (LHC) with analyses such as Ref. [6] (2011), which performed two complimentary measurements: one using physics-derived variables, and the other using an MVA (a boosted decision tree (BDT) [7, 8, 9, 10, 11, 12]). ML continued to become more and more integrated into the general HEP framework, with multiple MVA tools being used for different tasks within analyses. A notable example of this came in 2012 with the concerted use of no less than four MVAs (BDTs) in a single analysis: the search for Higgs boson decays to pairs of photons performed by the CMS collaboration at the LHC [13] - which contributed significantly to the discovery of the Higgs boson by ATLAS [14] and CMS [15, 16]. This process of MVA integration accompanied a shift in the community’s trust of approaches which relied less

on expert knowledge.

This degree of trust had yet to extend fully to neural networks, and although they had been used to some extent at both LEP and the Tevatron, they were still perceived as “black-box” methods; whilst researchers may be able to obtain better results by using them, they might be unable to explain how the results were achieved to sufficiently satisfy internal collaboration reviewers. In more recent years, the advancement of deep-learning approaches has led to improvements in model interpretability and greater knowledge of neural networks in the scientific community; both of which have helped foster a climate more amenable to their use [17, 18].

Whilst the improvements in performance that more advanced algorithms can bring is often the primary driving force in optimisation, it is also necessary to appreciate the run-time environment and time-requirements of the tasks they intend to solve. Algorithms used during data-collection at particle colliders must be extremely quick in order to not cause delays, which could lead to data not being recorded. Algorithms run after data are recorded have slightly relaxed time-requirements, however given the quantity of data that must be processed, which will increase considerably at the High Luminosity LHC, these algorithms must be sufficiently quick so as not to cause delays in the analyses. These two types of algorithms normally require infrequent training, i.e. they are trained once and used for a long period of data taking and processing by a large number of researchers. Therefore the training time is not so much an issue as their application time. Algorithms used at analysis-level, however, are normally developed by a small group of researchers, and tailor-made for a specific analysis. As the analysis evolves, these algorithms will need to be adapted and re-optimised several times, occasionally at short notice. It is therefore necessary that they are both quick to train and reasonably quick to apply. It is also important to consider the hardware available on demand to analysis-level researchers; while some institutes or individual researchers may have their own private GPUs, this cannot be assumed and algorithms must work sufficiently well on CPU as well.

Nowadays, deep neural networks (DNNs) are one of the main drivers of solutions to the domain-specific problems in HEP. The benefit of these approaches is that although we have theories and models to describe particle collisions, these do not automatically translate into optimal treatments of the data that are recorded. Previously researchers would have to handcraft solutions to reconstruct or represent the data in a more useful way. Deep learning instead allows the vast amount of data (both recorded and simulated) available to be used not only to reduce the statistical uncertainties on measurements, but to learn also the optimal treatment of the data starting from comparatively low-level representations. Such tasks faced in HEP are: object reconstruction[19, 20], collision and detector simulation [21, 22, 23], object identification [24, 25, 26], event triggering [27], likelihood-free inference [28, 29, 30], parameterised learning for multiple search hypotheses [31], unsupervised searches for new physics (anomaly detection) [32], and the optimal treatment of systematic uncertainties [33, 34]. Although these tasks are mostly peculiar

to particle physics, their solutions normally rely on applying and adapting techniques developed outside of HEP.

1.2.2 Introduction to searches at the LHC

At particle colliders such as the LHC [35], sub-atomic particles or ions are accelerated to high energies and made to collide with one another. The resulting collisions are attributable to fundamental-physics interactions between the particles, and their study can be used to compare theoretical models to reality in order to better understand the nature of the universe. The energy of these collisions gives rise to particles which are recorded with specialised instruments called *particle detectors*. Since there are many possible processes which could have given rise to a particular final state, and because the detectors only capture information of the products of the collision, there is no way to tell exactly how a given final state was created. However, each of the contributing processes is likely to have some specific signature, which is reflected in its outgoing products, e.g. a process which gives rise to an unstable particle will have decay products which can be used to reconstruct its mass, but due to experimental limits, such as the finite resolution of the detector, other processes can still give rise to collisions which reproduce the signature of desired process. Effectively, the data are unlabelled with respect to the classes of process that gave rise to them.

When looking to test some theory, such as the presence of a new *signal* process ¹, one normally defines some signal-enriched region, in which the process being searched for is expected to contribute appreciably. One can then examine the rate at which events populate this region and compare it to the rate expected if the new process were not occurring and the region were only being populated by known processes (background). This is done via a test of the signal+background hypothesis against the null hypothesis of background only, e.g. using the CL_s method [36, 37]. The catch is that, due to the data being unlabelled, the background and signal rates are unknown, and only their sum is known. In some situations the signal signature (e.g. a particle mass with some determined resolution) is well defined, and the background rate can be extrapolated from, or interpolated between, signal-depleted regions, but this is not always the case.

Instead, what can be done is to simulate particle collisions and the experimental equipment using a combination of analytical methods and stochastic modelling (see e.g. Refs. [38, 39, 40]), a process referred to as Monte Carlo generation and detector simulation. Since one is able to generate collisions for specified processes, one now has a labelled approximation of the collider data and can estimate the signal and background rates in the signal-enriched region, applying correction factors if need be to improve the modelling

¹Signal refers to a process or particle which is being searched for. An example of this is the presence of the Higgs boson as in Ref. [13], where the Higgs boson was treated as a new particle and other processes which had already been discovered formed a background (noise) to its discovery.

and marginalising over any remaining unknown (nuisance) parameters in the hypothesis test.

The exact specification of the signal-enriched region is an important task, which can have a strong influence on the outcome of a search; if the region is too wide the included signal becomes washed out by background, while if it is too narrow it may end up not being sufficiently populated. The main problem is the high dimensionality of the data; due to the high energy of the collisions, each collision can result in hundreds to thousands of particles, and each particle can produce multiple ‘hits’ and energy deposits in the detector. Traditionally, the first step is to reduce the dimensionality by *reconstructing* the hits and deposits back into known physics objects (fundamental particles, jets, missing energy, *et cetera*). The next stage is to select objects from the data which correspond to the expected final states of the signal process (and cut away events which fail to produce such objects). The signal region can then be defined using some theoretically or phenomenologically motivated function of the properties of these final states and other objects in the event.

ML techniques are becoming more and more used in HEP analyses, due to the ease with which they can discover high-dimensional patterns in data and use them to learn to solve some problem, such as learning to separate particle collisions by class (e.g. signal or background). This ability, however, has limits and it is currently beyond our computational capacity to run such event-level classification algorithms directly on the detector recordings. The contemporary compromise is to still perform the particle reconstruction and event selection, but then to use features based on the properties of the selected particles and the event as a whole as inputs to a machine-learning based classifier and use its response to help define the signal-enriched region. The development, training, and inference of such algorithms is still a difficult task and a source of experimentation in its own right.

1.3 The AMVA4NewPhysics European network

Advanced Multi-Variate Analysis for New Physics Searches at the LHC (AMVA4NewPhysics) was a European Commission Horizon 2020 Marie Skłodowska Curie Actions Innovative Training Network, funded under grant number 675440 for 48 months between 2015 and 2019. As its name suggests, it aimed to research and develop advanced solutions for improving searches for new physics at the LHC.

The Network itself consisted of nine recruiting participants: Istituto Nazionale di Fisica Nucleare - Padova (Italy), The University of Oxford (United Kingdom), Université Blaise Pascal Clermont-Ferrand II (France), Université Catholique de Louvain (Belgium), Laboratório de Instrumentação e Física Experimental de Partículas - Lisbon (Portugal), Institute of Accelerating Systems & Applications (Greece), Università Degli Studi di Padova (Italy), European Organization for Nuclear Research (Switzerland), and Technische Universität München (Germany).

In addition to these institutions, the network involved six partner institutions: University of California at Irvine (United States), École polytechnique fédérale de Lausanne (Switzerland), B12-Consulting (Belgium), SDG Consulting SpA (Italy), Yandex (Russian Federation), MathWorks (United States), and Universidad de Oviedo (Spain). In total 11 early-stage researchers (ESRs) were hired (ten concurrent) on contracts lasting two or three years, with the possibility of concurrently pursuing a PhD programme at local universities alongside their research responsibilities.

The research programme was subdivided into work packages (WP):

1. MVA for Higgs studies;
2. MVA for New Physics searches;
3. Advancing the Matrix Element Method;
4. New statistical learning tools for HEP analyses;
5. Career development and interdisciplinarity;
6. Outreach activities;
7. Management activities;

Each of these work packages stipulated a set of *deliverables*, items that had to be completed by set dates, these mainly consisted of public or internal papers, and open-source software packages and algorithms. Each ESRs' research activities were normally focussed on one of the four main WPs (1, 2, 3, 4) but they could also be partly responsible for deliverables in other WPs, allowing for greater cooperation with the network. In addition to pure research: ESRs contributed to scientific outreach through blogging (<https://amva4newphysics.wordpress.com>) and public events; attended an extensive programme training events ranging from single-day masterclasses to fortnight-long schools on topics such as public speaking, statistics, and machine learning; and were seconded to other network institutions, either to work more closely with other ESRs on particular tasks, or to learn new skills and experience working in a different environment at the industry partners.

My particular involvements in with AMVA4NewPhysics are as follows: I was hired March 2016 - March 2019 to work at Laboratório de Instrumentação e Física Experimental de Partículas (LIP) in Lisbon, beginning my PhD course at Instituto Superior Técnico - Universidade de Lisboa in November 2016. My research took place within the CMS group at LIP where I primarily contributed to deliverables in WP 1 with a focus on developing classification and regression tools for Higgs and di-Higgs applications in the $h \rightarrow b\bar{b}$ and $h \rightarrow \tau^+ \tau^-$ decay channels. These deliverables were:

- D1.1 - Multivariate Analysis Methods for Higgs Boson Searches At The Large Hadron Collider: A public report on the use of neural networks for classification and regression in the $h \rightarrow b\bar{b}$ and $h \rightarrow \tau^+ \tau^-$ decay channels of non-resonant di-Higgs searches [41];
- D1.3 - Multivariate Analysis Methods for Higgs Boson Searches At The Compact Muon Solenoid: An internal report on the use of neural networks for classification and regression in the $h \rightarrow b\bar{b}$ and $h \rightarrow \tau^+ \tau^-$ decay channels of non-resonant di-Higgs searches at CMS [42];
- D1.4 - Classification and Regression Tools in Higgs Measurements: A public paper on the use of advanced methods in HEP searches with a focus on Higgs boson measurements [43].

In this thesis, Chapters Chap. 7 and Chap. 8 summarise my contributions to D1.1 and D1.3, and a followup investigation of my contribution to D1.4 is reported in Chap. 9. My secondments consisted of:

- The University of Oxford, Oxford, UK - Two weeks - Worked with fellow ESRs at the physics department to produce simulated samples of Monte Carlo data for use in D1.1 and other investigations.
- Universita Degli Studi di Padova, Padova, Italy - One month - Worked with fellow ESRs at the statistics department on classification techniques for D1.1.
- INFN, Padova, Italy - One month - Working with fellow ESRs on regression techniques for D1.1.
- CERN, Geneva, Switzerland - One month - Worked on advanced methods for neural networks in preparation for D1.4.
- Yandex, Moscow, Russian Federation - Three months - Helped develop ‘Coopetition’ (<https://coopetition.coresearch.club>), a hosting site for data-science challenges with a focus on cooperation between participants to solve challenges together, rather than the traditional approach of participants competing against one another. Gained experience in: software development and coding, Scrum agile development, and different approaches to problem solving.

Following the end of my AMVA4NewPhysics contract I partly funded the final year of my PhD by working half-time for eight months as the principal researcher in a data science project that was a joint collaboration between Nielsen (an international data analytics firm) and The Competence Centre for Big Data and Simulation, which had recently been established at LIP. This allowed me to apply the skills I had learnt during the first three years of my PhD, as well as develop them further in a slightly different environment. One

particular outcome of this work was the fine tuning of the feature-selection methodology I had used and improved over the course of the network deliverables. This eventually fed back into my PhD research and I will describe it in Sec. 10.7.

Chapter 2

Background explanations: The Standard Model of particle physics

The Standard Model (SM) is a theory used to describe the interactions between fundamental particles. It consists of two main sectors: quantum chromodynamics (QCD), an $SU(3)$ gauge theory describing the *strong* force - interactions between particles carrying *colour* charge; and the electroweak sector (EW), an $SU(2) \times U(1)$ gauge theory containing the electromagnetic and *weak* forces, and the Higgs mechanism. These sectors are summarised below, using the following as references: Peskin and Schroeder “An introduction to quantum field theory” [44], Griffiths “Introduction To Elementary Particles” [45], Krane “Introductory Nuclear Physics” [46], Eisberg and Resnick “Quantum physics of atoms, molecules, solids, nuclei and particles” [47], and Maxwell “The Standard Model” [48].

2.1 Electroweak sector

The EW sector according to Glashow-Weinberg-Salam [49, 50, 51] is described by a $SU(2) \times U(1)$ gauge theory. The *weak* force is carried by two types of bosons, W and Z , and the electromagnetic force is transmitted by the photon (γ). Additional particles contained in the EW sector are the families of leptons and quarks, and the Higgs boson.

2.1.1 Leptons

Leptons are fermions (spin- $1/2$) elementary particles and occur in two types: charged leptons, ℓ and neutrinos, ν , which are electrically neutral. Additionally, three *generations* of leptons have so far been observed. In order of increasing mass these are: electron

Lepton	Charge	Mass	Generation number			Mean lifetime
	[e]	[MeV]	L_e	L_μ	L_τ	
e	-1	$0.510\,998\,946\,1(31)$	$+1$	0	0	$> 6.6 \times 10^{28}$ years
μ	-1	$105.658\,374\,5(24)$	0	$+1$	0	$2.196\,981\,1(22) \times 10^{-6}$ s
τ	-1	$1776.86(12)$	0	0	$+1$	$290.3(5) \times 10^{-15}$ s
ν_e	0	$< 2 \times 10^{-6}$	$+1$	0	0	Unknown
ν_μ	0	$< 2 \times 10^{-6}$	0	$+1$	0	Unknown
ν_τ	0	$< 2 \times 10^{-6}$	0	0	$+1$	Unknown

Table 2.1: Properties of leptons. Values as per Ref. [53].

(e), muon (μ), and tau (τ). The lepton field can be represented as a *chiral weak isospin doublet*:

$$\chi_L^f = \begin{pmatrix} \nu_f \\ \ell_f \end{pmatrix}, \quad (2.1)$$

where f indicates the generation number. Each lepton carries a *lepton number* of $+1$, a *generation number* of $+1f$, and an electric charge of $-1e$ for ℓ (where e is the absolute electric charge of the electron) and 0 for ν . Antileptons carry opposite quantum numbers, as usual. In all interactions, lepton number, generation number, and electric charge must be conserved. A summary of the properties of leptons is detailed in Tab. 2.1, note that whilst empirically indicated to be massive [52], neutrinos are considered massless within the SM.

2.1.2 Quarks

Similarly to leptons, quarks are fermions which occur in two types and three generations:

$$\chi_q^f = \begin{pmatrix} U_f \\ D'_f \end{pmatrix}, \quad (2.2)$$

where again f is the generation number, U is an *Up-type* quark, carrying an electric charge of $+2/3e$, and D' is a *Down-type* quark (electric charge of $-1/3e$) which has been rotated in *flavour space*:

$$D'_f = \sum_{f'} V_{ff'} D_{f'}. \quad (2.3)$$

Six quark flavours have so far been discovered: Of the up-type, up (u), charm (c), and top (t); and of the down-type, down (d), strange (s), and bottom (b).

Quark	Charge [$e/3$]	Mass	Generation
Up (u)	+2	$2.16^{+0.49}_{-0.26}$ MeV	1
Down (d)	-1	$4.67^{+0.48}_{-0.17}$ MeV	1
Charm (c)	+2	1.27 ± 0.02 GeV	2
Strange (s)	-1	93^{+11}_{-5} MeV	2
Top (t)	+2	172.9 ± 0.4 GeV	3
Bottom (b)	-1	$4.18^{+0.03}_{-0.02}$ GeV	3

Table 2.2: Properties of quarks. Values as per Ref. [53]. Note that the top-mass quoted is via direct measurements.

V in Eq. 2.3 is a matrix of coefficients known as the Cabibbo-Kobayashi-Masukawa matrix [54, 55]:

$$V_{CKM} = \begin{pmatrix} |V_{ud}| = 1 - \frac{\lambda^2}{2} & |V_{us}| = \lambda & |V_{ub}| = A\lambda^3\phi \\ |V_{cd}| = -\lambda & |V_{cs}| = 1 - \frac{\lambda^2}{2} & |V_{cb}| = A\lambda^2 \\ |V_{td}| = A\lambda^3(1 - \phi) & |V_{ts}| = -A\lambda^2 & |V_{tb}| = 1 \end{pmatrix} + \mathcal{O}(\lambda^4), \quad (2.4)$$

where $\phi = \bar{\rho} - i\bar{\eta}$. The remaining parameters must be determined experimentally, and the latest values at time of writing are approximately: $\lambda = 0.225$, $A = 0.83$, $\bar{\rho} = 0.1$, and $\bar{\eta} = 0.35$, as taken from Ref. [53] and considering the two fitting prescriptions used.

Each quark has a *baryon number* of $+1/3$ ($-1/3$ for antiquarks) and this must be conserved in all interactions. The properties of the quarks are summarised in Tab. 2.2.

2.1.3 EW bosons

The W and Z bosons, of mass $80.379(12)$ GeV and $91.1876(21)$ GeV respectively [53], are responsible for transmitting the *weak* force. The W boson carries an electric charge of $\pm 1e$ and provides *charge-current interactions*; processes in which lepton-antineutrino, or up-quark-anti-down-quark pairs are created or annihilated - as shown in Fig. 2.1. Additionally, the W boson only interacts with left-handed fermions (fermions with their spin in the opposite direction to their momentum).

The Z boson carries no electric charge and instead mediates *neutral-current interactions*; scatterings and creation/annihilation net-neutral pairs of particles. The photon γ (massless and electrically neutral) also mediates neutral-current interactions, but only couples to electric charge. Examples of neutral-current interactions are shown in Fig. 2.2.

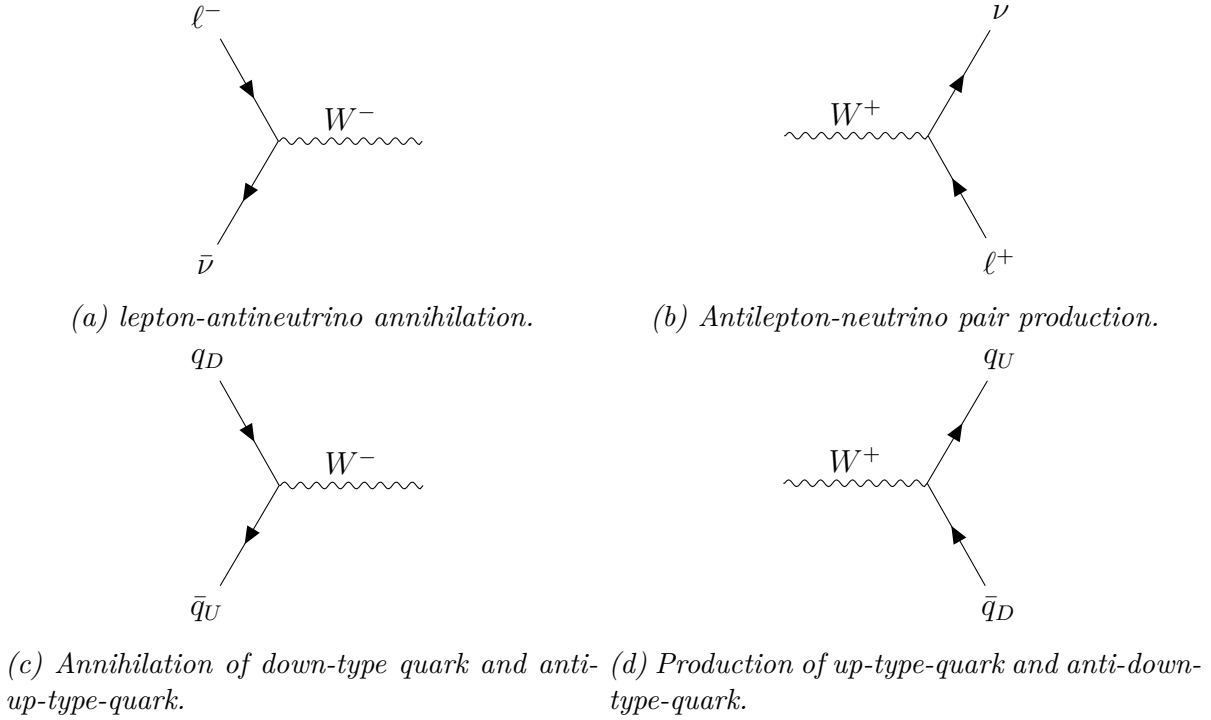


Figure 2.1: Examples of W -boson mediated charge-current interactions.

2.2 Quantum Chromodynamics

QCD is an $SU(3)$ gauge theory, where three is the number of *colours*. Colour charge was a quantum number proposed by Greenberg [56], Han and Nambu [57], and Fritzsche, Gell-Mann, and Leutwyler [58] to explain the observation of the Δ^{++} meson; a spin- $3/2$, doubly charged particle consisting of three *up*-type quarks. Since quarks are expected to obey Fermi-Dirac statistics, which does not allow multiple fermions (particles with spin- $1/2$) to occupy the same state, such a configuration would violate the Pauli exclusion-principle. By introducing a new quantum number, this inconsistency can be reconciled by requiring each of the three quarks to be of unique colour.

2.2.1 Quarks

Considering three colours (red, green, and blue), the quark field in QCD can be represented as a colour-triplet:

$$\psi(\tilde{x}) = \begin{pmatrix} \psi_r(\tilde{x}) \\ \psi_g(\tilde{x}) \\ \psi_b(\tilde{x}) \end{pmatrix}.$$

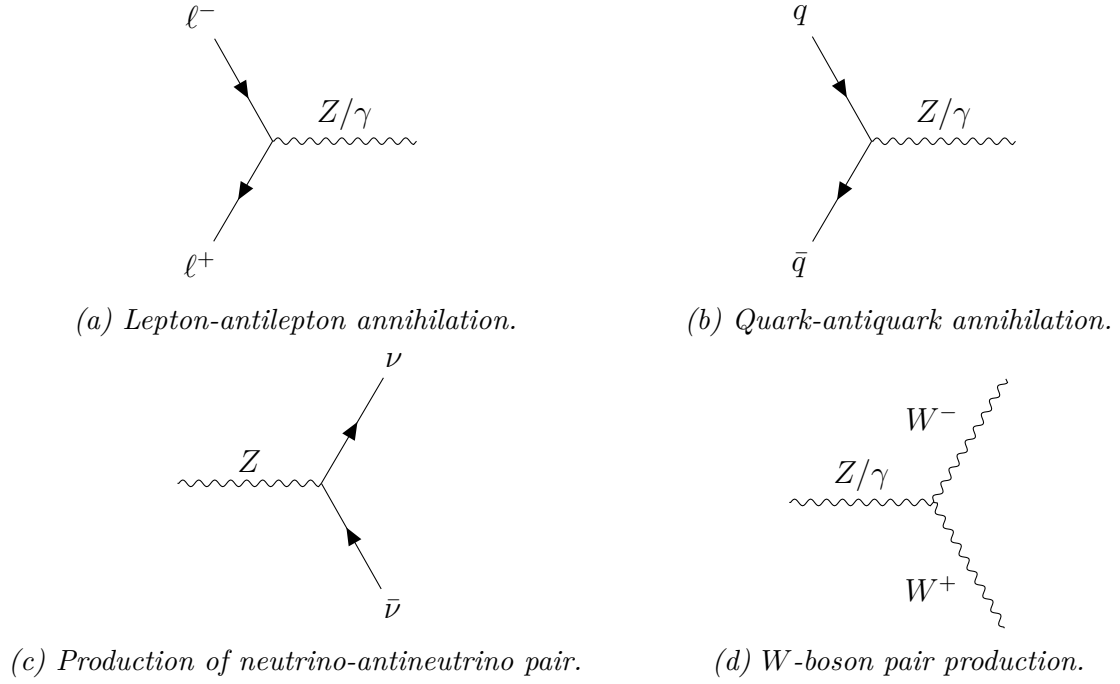


Figure 2.2: Examples of Z/γ -boson mediated neutral-current interactions.

Transformations between colour-states is accomplished by interacting with the gluon field.

2.2.2 Gluons

Gluons are massless bosons which mediate the *strong* force by carrying colour charge over small distances. Each gluon carries both colour and anti-colour charges, forming a colour-octet:

$$\begin{aligned}
 |1\rangle &= (r\bar{b} + b\bar{r}) / \sqrt{2} & |2\rangle &= -i (r\bar{b} + b\bar{r}) / \sqrt{2} \\
 |3\rangle &= (r\bar{r} - b\bar{b}) / \sqrt{2} & |4\rangle &= (r\bar{g} + g\bar{r}) / \sqrt{2} \\
 |5\rangle &= -i (r\bar{g} + g\bar{r}) / \sqrt{2} & |6\rangle &= (b\bar{g} + g\bar{b}) / \sqrt{2} \\
 |7\rangle &= -i (b\bar{g} + g\bar{b}) / \sqrt{2} & |8\rangle &= (r\bar{r} + b\bar{b} - 2g\bar{g}) / \sqrt{6}
 \end{aligned} \tag{2.5}$$

If QCD were to be formed as an abelian $U(3)$ gauge theory, then a ninth state, the colour-singlet, could also be formed: $|9\rangle = (r\bar{r} + b\bar{b} + g\bar{g}) / \sqrt{3}$. Such a gluon would allow for long-range interaction via the strong force. Since no experimental evidence for gluons acting as free particles has been found, QCD is instead formed as a non-abelian $SU(3)$ theory, which does not include the singlet state.

The gluon field, $\mathcal{A}_\mu$, is a linear combination of colour-charged scalar fields, $\mathcal{A}_\mu^a$ and

the generators of $SU(3)$, $T_a = \lambda_a/2$:

$$\mathcal{A}_\mu = \frac{1}{2} \sum_{a=1}^8 \lambda_a \mathcal{A}_\mu^a, \quad (2.6)$$

where λ_a are the eight Gell-Mann matrices; 3×3 traceless, Hermitian matrices:

$$\begin{aligned} \lambda_1 &= \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \lambda_2 &= \begin{pmatrix} 0 & -i & 0 \\ -i & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \lambda_3 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ \lambda_4 &= \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} & \lambda_5 &= \begin{pmatrix} 0 & 0 & -i \\ 0 & 0 & 0 \\ i & 0 & 0 \end{pmatrix} & \lambda_6 &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \\ \lambda_7 &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -i \\ 0 & i & 0 \end{pmatrix} & \lambda_8 &= \frac{1}{\sqrt{3}} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{pmatrix} \end{aligned} \quad (2.7)$$

The generators of QCD satisfy the Lie algebra:

$$[T_a, T_b] = T_a T_b - T_b T_a = i \sum_{c=1}^8 f_{abc} T_c, \quad (2.8)$$

where $[]$ is the *commutator* operator and f_{abc} are referred to as the *antisymmetric structure constants*:

$$f_{abc} = \begin{cases} 1, & \text{if } a, b, c = 1, 2, 4 \\ 0.5, & \text{if } a, b, c = 1, 4, 7 \mid 2, 4, 6 \mid 2, 5, 7 \mid 3, 4, 5 \mid 5, 1, 6 \mid 6, 3, 7 \\ \sqrt{3}/2, & \text{if } a, b, c = 4, 5, 8 \mid 6, 7, 8 \\ 0, & \text{otherwise.} \end{cases} \quad (2.9)$$

2.2.3 QCD interactions

The Lagrangian density of QCD for a particular quark flavour is:

$$\mathcal{L} = (i\bar{\psi}\gamma^\mu\partial_\mu\psi - m\bar{\psi}\psi) - (g\bar{\psi}\gamma^\mu\lambda\psi) \cdot \mathcal{A}_\mu - \frac{1}{16\pi} F^{\mu\nu} \cdot F_{\mu\nu}, \quad (2.10)$$

where ψ is the quark field, m is the quark mass-term, g is referred to as the *strong coupling constant*, γ are the four 4×4 contravariant Dirac matrices, and $F^{\mu\nu}$ is the gluon-field strength-tensor:

$$F^{\mu\nu} \equiv \partial^\mu \mathcal{A}^\nu - \partial^\nu \mathcal{A}^\mu - 2g(\mathcal{A}^\mu \times \mathcal{A}^\nu) \quad (2.11)$$

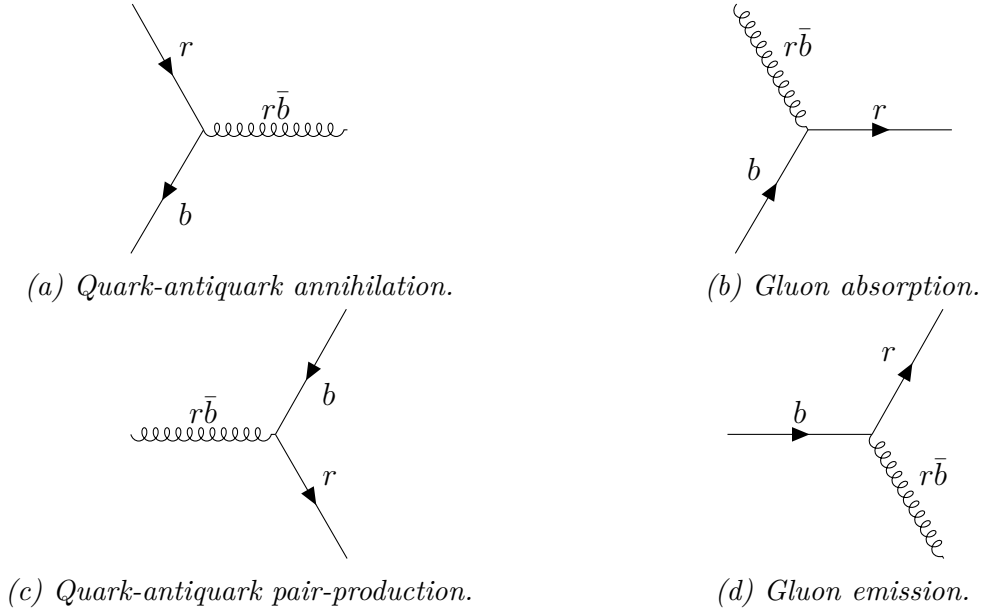


Figure 2.3: Quark-gluon interaction diagrams.

The $-(g\bar{\psi}\gamma^\mu\lambda\psi)\cdot\mathcal{A}_\mu$ term in Eq. 2.10 describes the QCD interactions possible between quarks and gluons. These are shown in Fig. 2.3. Note that interaction with the gluons allow the quarks to change colour charge but not flavour.

Since the gluons themselves also carry colour charge they can self-interact, giving rise to the $-\frac{1}{16\pi}F^{\mu\nu}\cdot F_{\mu\nu}$ term in Eq. 2.10. Such self-interactions are illustrated in Fig. 2.4. This self-interaction means that gluons can quickly lead to *parton showers* in which a multitude of *partons* (collective term for quarks and gluons) are produced. Such phenomena are difficult too model exactly, and will be discussed further in Sec. 3.5.5. Experimentally, we are normally more interested in knowing information about the originating parton, rather than each individual resulting parton. To do so, *jets* are formed by clustering the resulting particles together. These techniques will be discussed further in Sec. 3.4.3.

2.2.4 Hadrons

Quarks exhibit a property called *colour confinement*; isolated, colour-charged particles are not found outside of the high-energy regime. Instead colour-charged particles are connected by a *tube* of gluon field with a set radius and energy density, meaning that the force required to separate the particles increases exponentially and the overall system has no net colour charge. Eventually, if enough energy is put into the system, a pair of new particles may be formed at closer separation.

The term *hadronisation* is used to refer to the process of quarks forming into colour-

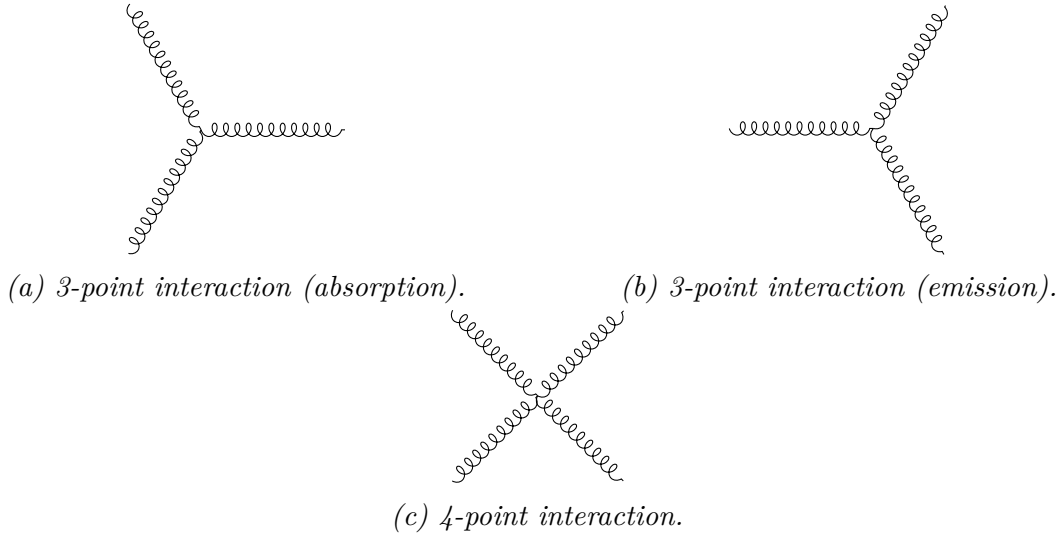


Figure 2.4: Quark-gluon interaction diagrams.

neutral particles, collectively called hadrons. Either these can be mesons (two-quark particles), or baryons (three-quark particles). Well known baryons are the proton (uud) and neutron (udd). Mesons occur in several varieties, but of occasional use in this thesis are the lightest mesons, the pions. These can either be charged ($\pi^+ = u\bar{d}$ and $\pi^- = d\bar{u}$), or neutral ($\pi^0 = (u\bar{u} - d\bar{d})/\sqrt{2}$), with masses 139.570 61(24) MeV and 134.9770(5) MeV, respectively [53].

The previous paragraph is a slightly simplified view of hadrons, however. The quarks used to determine their properties are referred to as their *valence* quarks, though these only carry a fraction of the the total momentum of the hadron. The remaining momentum comes from the continual creation and annihilation of virtual quark/anti-quark pairs, and virtual gluons. These virtual particles are referred to as the quark-gluon *sea*. The momentum composition of hadrons can be parameterised as a *parton distribution function*, and these will be discussed further in Sec. 3.5.3.

It should be noted that the top quark, the heaviest particle in the Standard Model (so far), decays on a timescale shorter than that of the strong force and so does not form hadrons. These decays invariably result in a W boson and a bottom quark [53].

2.3 The Higgs mechanism

Within the Standard Model described thus far, there is no potential for quarks, leptons, and weak bosons to be massive without violating gauge invariance; the dynamics of the system should not depend on the choice of representation, however a mass term of the

form $-m_{\mathcal{A}}^2 \mathcal{A}_\mu \mathcal{A}^\mu$ does not remain unchanged when $\mathcal{A}_\mu$ undergoes a local gauge transformation: $\mathcal{A}_\mu \rightarrow \mathcal{A}'_\mu = \mathcal{A}_\mu + \partial_\mu \Lambda$, but $-m_{\mathcal{A}^2} \mathcal{A}_\mu \mathcal{A}^\mu \neq -m_{\mathcal{A}^2} \mathcal{A}'_\mu \mathcal{A}'^\mu$. Thus the theoretical description of these particles cannot be reconciled with their experimentally massive nature. The Higgs mechanism, or more fully the Anderson-Brout-Englert-Guralnik-Hagen-Higgs-Kibble-'t Hooft mechanism, [59, 60, 61, 62] provides a way for these particles to acquire mass via a process referred to as *spontaneous symmetry breaking*.

2.3.1 Spontaneous symmetry breaking

The Higgs field is specified to be a complex-valued isospin doublet in an $SU(2)$ gauge theory:

$$\Phi = \begin{pmatrix} \phi^+ \\ \phi^0 \end{pmatrix}, \quad (2.12)$$

where

$$\phi^+ = \frac{\phi_1 + i\phi_2}{\sqrt{2}}, \quad \phi^0 = \frac{\phi_3 + i\phi_4}{\sqrt{2}}. \quad (2.13)$$

In EW $SU(2) \times U(1)$, this can form a scalar potential of the form:

$$V(\Phi) = \mu^2 \Phi^\dagger \Phi - \lambda (\Phi^\dagger \Phi)^2. \quad (2.14)$$

For the case in which $\lambda < 0$ and $\mu^2 < 0$ (resulting in a *wrong-sign* mass term $-|\mu^2| \Phi^\dagger \Phi$), the ground-state of $V(\Phi)$ is no longer zero, but rather forms a ring of degenerate minima at:

$$\Phi^\dagger \Phi = \frac{1}{2} (\phi_1^2 + \phi_2^2 + \phi_3^2 + \phi_4^2) = \frac{\mu^2}{2\lambda}, \quad (2.15)$$

as illustrated in Fig. 2.5.

Spontaneous symmetry breaking refers to the act of picking one minima from the set of degenerate minima of V to act as the vacuum state. This is chosen such that the vacuum expectation values (vev) of ϕ_1 , ϕ_2 , and ϕ_4 (the ‘‘Goldstone’’ fields) are zero, and ϕ_3 has a non-zero vev:

$$\langle 0 | \phi_i | 0 \rangle = 0, \quad i \in \{1, 2, 4\}, \quad \text{and} \quad \langle 0 | \phi_3 | 0 \rangle^2 = v^2 = \frac{\mu^2}{\lambda}. \quad (2.16)$$

Moving to the *unitary gauge* and setting $\phi_3 = H + v$, Φ is now:

$$\Phi = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ H + v \end{pmatrix}, \quad (2.17)$$

where H is the scalar Higgs field. The Goldstone fields can be removed via a local gauge transformation and are said to be ‘‘eaten’’ by the $W^\pm$ and Z bosons, allowing them to acquire mass. The photon remains massless.

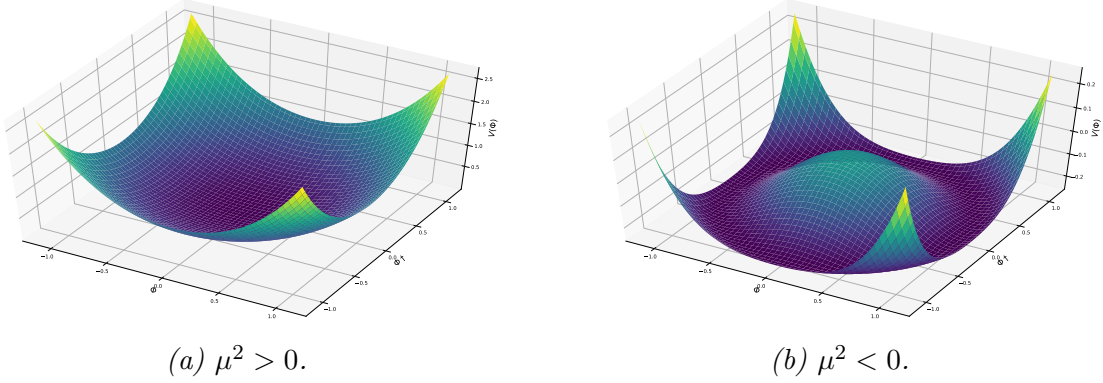


Figure 2.5: Illustrations of an example of $V(\Phi)$. For $\mu^2 > 0$ (a) a single minima exists at $V(0) = 0$, whereas for $\mu^2 < 0$ (b) a ring of degenerate minima exists, forming a shape sometimes called the “wine-bottle” or “Mexican-hat” potential.

2.3.2 Yukawa terms

The charged leptons and quarks may also now acquire masses via the introduction of *Yukawa* terms to their respective Lagrangian densities. For leptons this term is:

$$\begin{aligned}\mathcal{L}_Y(\ell) &= -G_\ell (\bar{\chi}_L \Phi \ell_R + \bar{\ell}_R \Phi^\dagger \chi_L) \\ &= -\frac{G_\ell v}{\sqrt{2}} (\bar{\ell}) - \frac{G_\ell}{\sqrt{2}} (\bar{\ell} \ell H),\end{aligned}\tag{2.18}$$

providing a mass term $G_\ell v/\sqrt{2}$ and a lepton-Higgs coupling $G_\ell/\sqrt{2} (\bar{\ell} \ell H)$, which is proportional to the lepton’s mass.

The Yukawa term for the quarks takes a similar form:

$$\mathcal{L}_Y(q) = \sum_f - \left(\bar{\chi}_L^f G_{ff'}^D \Phi D_{f'R} + \bar{\chi}_L^f G_{ff'}^U \Phi^c U_{f'R} + \text{Higgs couplings} \right),\tag{2.19}$$

where the conjugate scalar field:

$$\Phi^c = \begin{pmatrix} H + v \\ 0 \end{pmatrix}.\tag{2.20}$$

2.3.3 Higgs interactions

As seen in Sec. 2.3.2 the Higgs boson, H , couples to particles with a strength proportional to their mass, meaning that whilst the Higgs boson cannot couple directly to massless particles (photons, neutrinos, and gluons) it can couple to itself, which will be discussed further in Chap. 6. Interactions with massive SM particles leads to a set of three- and four-point interactions, summarised in Fig. 2.6 and Fig. 2.7.

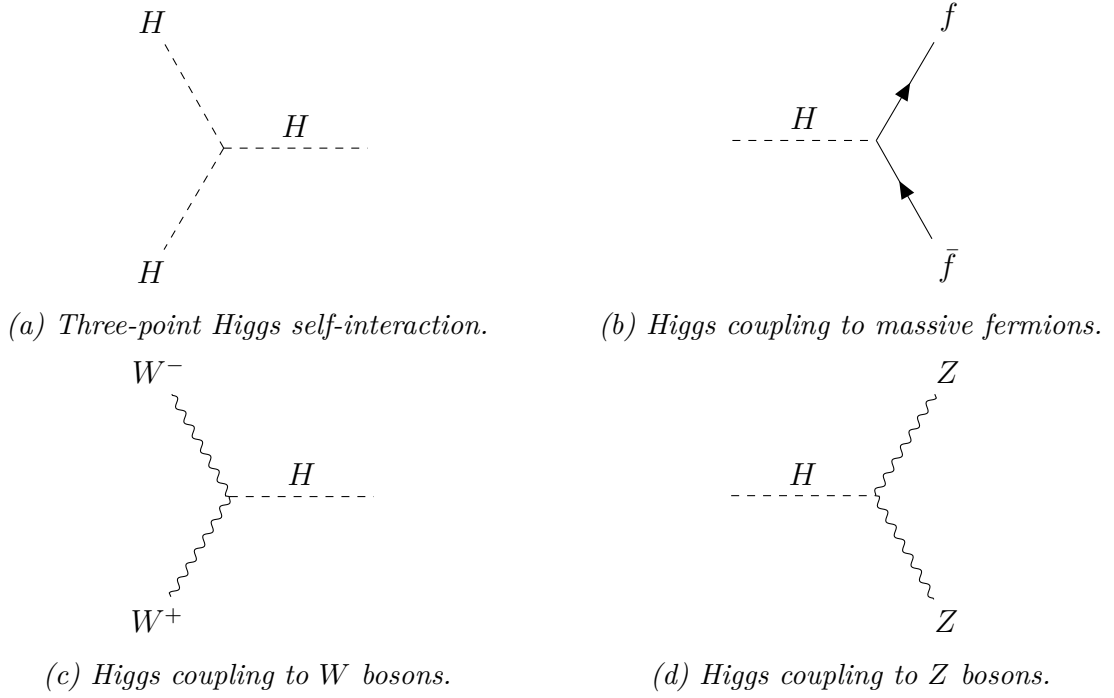


Figure 2.6: Examples of three-point Higgs interactions.

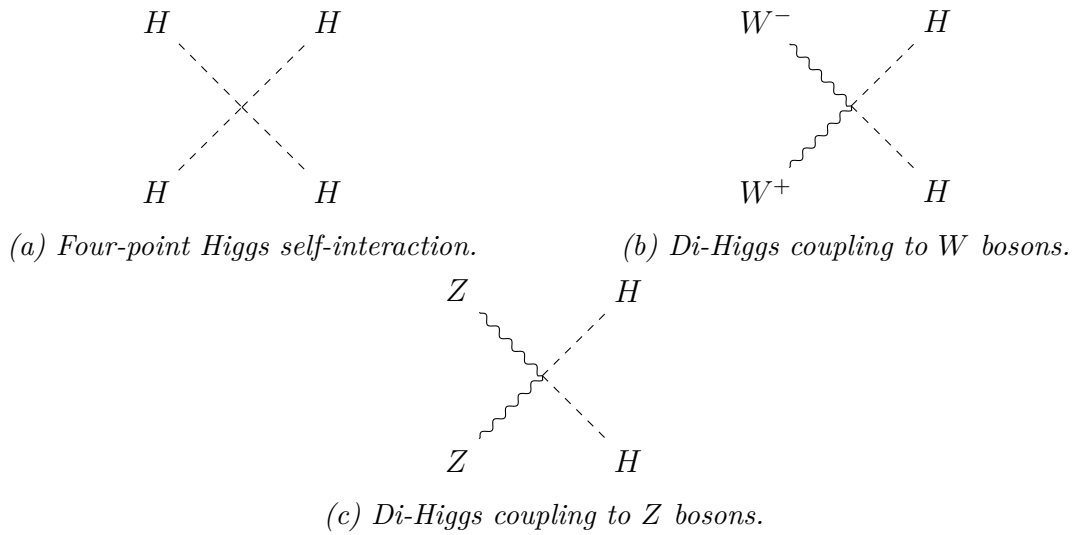


Figure 2.7: Examples of four-point Higgs interactions.

Chapter 3

Background explanations: Particle physics at the LHC

3.1 Assorted definitions of relevant physics terminology

3.1.1 Natural units

Throughout this thesis, measurements of kinematic quantities will usually be given in *natural units*, in which the speed of light *in vacuo* and the reduced Plank constant $\hbar = \hbar/2\pi$ are both equal to one. Under such a scheme, mass and energy may be written in units of electron volts, eV.

3.1.2 Luminosity

Instantaneous luminosity is a useful measure of the rate at which a particle collider, such as the LHC, can produce particle-collision data. For circular colliders which accelerate k_b bunches of N particles at a revolution frequency f , the instantaneous luminosity L is:

$$L = \frac{f\gamma}{4\pi\epsilon_n} N^2 k_b \frac{F(\beta^*)}{\beta^*}, \quad (3.1)$$

where F and β^* adjust the collision area according to the angle of beam crossing, and ϵ_n accounts of the spread of particles in position-momentum space [63]. L is normally expressed in terms of inverse-area per second, $\text{cm}^{-2}\text{s}^{-1}$.

Integrating the instantaneous luminosity of a collider over the runtime of an experiment provides a summary of the amount of data which can be collected, the integrated

luminosity in inverse-barns (b^{-1}):

$$L_{int} = \int_0^T L dt. \quad (3.2)$$

3.1.3 Coordinate systems

As will be discussed further in 3.3, the CMS detector is cylindrical in shape, displaying rotational and forwards-backwards symmetry in the transverse and longitudinal planes, respectively, with respect to the beam-axis.

Two coordinate systems are normally used in particle physics: the Cartesian system of (x, y, z) , which can be used to express geometric position and components of particle momenta; and cylindrical coordinates (r, η, ϕ) , again used for geometric position and components of particle momenta (in which case radius r is the particles' transverse momenta p_T).

At CMS the Cartesian system is defined as shown in Fig. 3.1 such that the origin is the nominal collision-point of the experiment, the z -axis points along the beam-axis, the x axis points towards the centre of the LHC ring, and the y axis points perpendicularly upwards.

The cylindrical coordinates for momenta are defined in terms of transverse momentum p_T , azimuthal angle ϕ , and pseudorapidity η , where for a particle with momentum p :

$$\eta = \tanh^{-1} \left(\frac{p_z}{|\vec{p}|} \right). \quad (3.3)$$

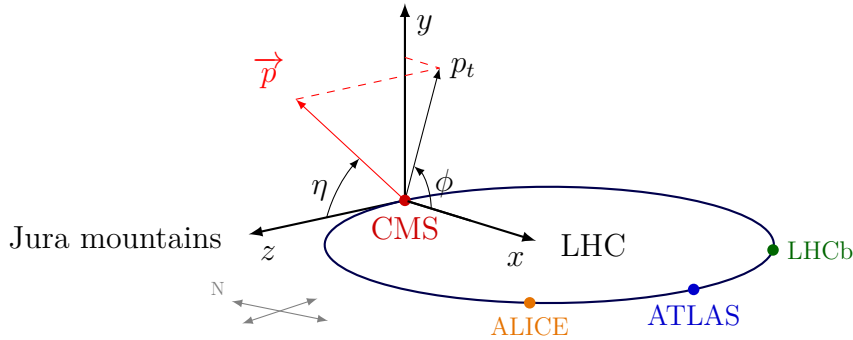


Figure 3.1: Illustration of the coordinate systems used at the CMS experiment in the geographical context of the LHC. Image notice [64].

In this thesis the ΔR separation between two objects refers to the Euclidean distance between two points in (ϕ, η) space:

$$\Delta R = \sqrt{\Delta\phi^2 + \Delta\eta^2}, \quad (3.4)$$

where $\Delta\phi$ and $\Delta\eta$ are the angular separations between the points in terms of azimuthal and pseudorapidity, remembering that ϕ is cyclical within $[-\pi, \pi)$.

3.2 The Large Hadron Collider

The Large Hadron Collider (LHC) is a proton-proton, proton-lead, and lead-lead circular collider with a circumference of 27 km located near Geneva on the French-Swiss boarder (see Fig. 3.2 for a map) as part of the CERN complex. It begun operation in 2008 and has continued to run since then, interspaced with shutdown periods for maintenance and upgrades. It has so far completed two long periods of operation: Run-1 at a centre-of-mass (CoM) energy of 7 and 8 TeV for proton-proton collisions, between 2009-2013; and Run-2 at 13 TeV CoM energy proton-proton collisions, between 2015-2018¹. A third run is expected to begin in 2021. Following Run-3, the LHC is planned to undergo extensive upgrades to reach ten times its current design luminosity. The High Luminosity LHC (HL-LHC) is expected to begin operation in 2027.

This thesis primarily focusses on the data collected during Run-2 in which an integrated luminosity of 158.64 fb^{-1} was recorded, with peak instantaneous-luminosity reaching about $2 \times 10^{34} \text{ cm}^{-2}\text{s}^{-1}$ in 2017 and 2018; double the design luminosity of $1 \times 10^{34} \text{ cm}^{-2}\text{s}^{-1}$. The bunch spacing was 25 ns. Since the LHC collides bunches of hadrons at such high frequencies, data collection at the LHC are affected by a phenomenon called *pile up*: there can be collisions between multiple pairs of hadrons (in-time pile-up), and remnants from earlier collisions arriving in the detector during the recording of a later collision (out-of-time pile-up).

¹Collisions involving lead ions are not reported here

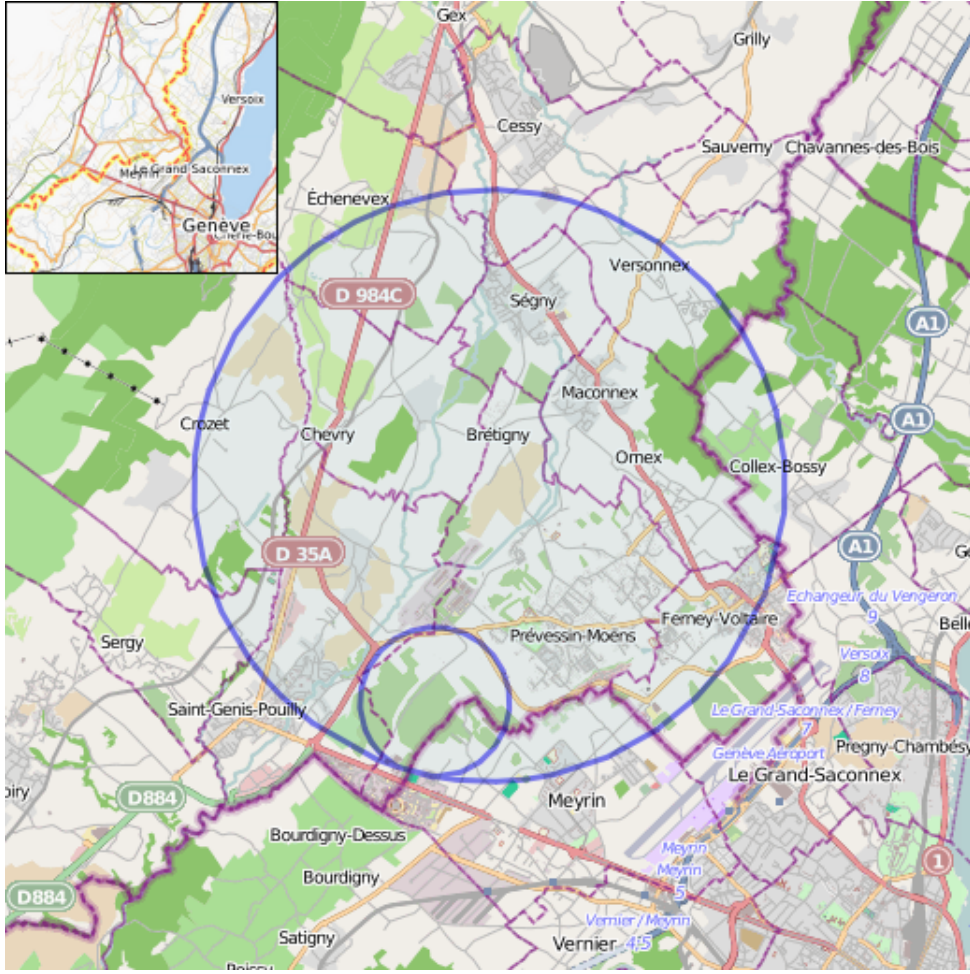
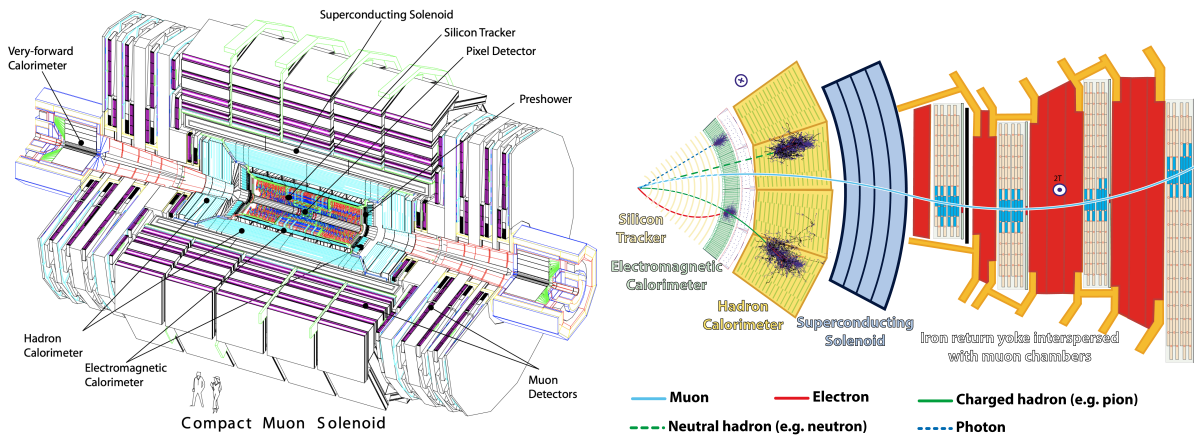


Figure 3.2: Map of the LHC (blue rings) located near Geneva. Image notice [65].

3.3 The CMS detector

The Compact Muon Solenoid (CMS) [66] is one of the two multi-purpose detectors in operation at the LHC. Particle detection takes place in several sub-detector components, as illustrated in Fig. 3.3. In order of proximity to the nominal interaction point, these are: The inner tracker, which provides position-recording of charged particles; the calorimeters, which measure the energy carried by electromagnetic and hadronic particles; the muon system provides detection and measurement of muons; and forward detectors to study high-pseudorapidity interactions, including the precision proton spectrometer (PPS) to detect non-dissociative interactions between protons. A super conducting solenoid provides a magnetic field to bend the flight-path of charged particles, allowing their charge and momenta to be measured. In addition to the detector subcomponents, a series of

triggers are used to select which collisions to record, allowing the very high data-rate to be reduced to a more manageable fraction of scientifically interesting collisions. The detector is approximately cylindrical, displaying azimuthal and longitudinal symmetry about the nominal interaction point, with a diameter of 21.6 m, length of 14.6 m, and mass of 12 500 t.



(a) Subcomponent layout of the CMS detector. (b) Transverse slice of the CMS detector illustrating particle interactions with detector sub-components. Image source [66].

Figure 3.3: Layout of the CMS detector.

3.3.1 Superconducting solenoid

The superconducting niobium-titanium (NbTi) solenoid [68] generates a homogenous 3.8 T magnetic field across a cylindrical volume of diameter 6.3 m and length 12.5 m in a direction parallel to the beam axis. An iron *return yoke* of diameter 14 m and length 13 m surrounds the solenoid, providing a 2 T magnetic field in the opposite direction.

Charged particles propagating through these fields are accelerated in an azimuthal direction, according to their charge. This acceleration leads to a curvature of their flight-paths, the rate of which is proportional to their transverse momentum. The use of such a high magnetic flux density provides greater momentum-resolution for charged particles. This in turn improves invariant-mass resolutions for decaying particles, increases the efficiency of muon triggers, and allows more accurate calibration of the electromagnetic calorimeter.

3.3.2 Inner tracker

The aim of the inner tracker is to provide positional information of charged particles. From this, the momenta and charges of particles can be measured, and secondary vertices resulting from the decay of long-lived particles can also be located. The full tracker is of diameter 2.5 m and length 5.8 m and consists of two main sub-detectors: the pixel detector and the strip detector. The Phase-0 layout is illustrated in Fig. 3.4.

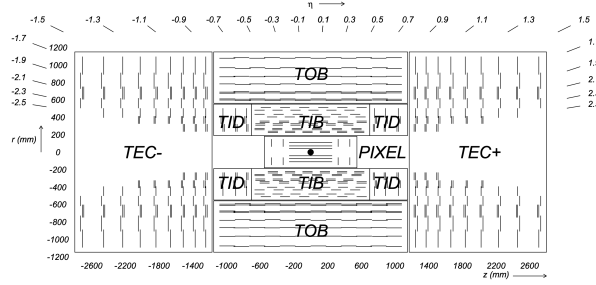


Figure 3.4: Layout of the CMS inner tracker. Image source [66].

Pixel detector

The pixel detector is the sub-detector closest to the interaction point, providing good reconstruction of secondary vertices through high-resolution of impact-parameters. Silicon pixels are arranged in layers about the nominal interaction-point and together they provide 3D positional information in (r, ϕ, z) space over a pseudorapidity range of $|\eta| \leq 2.5$. Through charge-interpolation a spatial-resolution of 15-20 μm can be achieved.

For the 2016 data-taking, the Phase-0 pixel detector was used, which contained 66 million pixels in total, arranged across three barrel layers parallel to the beam axis and two layers of end-cap discs transverse to the beam axis. In the Extended Year-End Technical Stop (EYETS, 2016-2017) Phase-I upgrades [69] were implemented with a new pixel detector being installed for 2017 and 2018 data-taking. The new detector contains four barrel layers and three end-cap discs, with a total of 122 million pixels. This upgrade was necessary to deal with radiation damage to the Phase-0 detector, which was expected, and to maintain efficient tracking in the event that the strip-detector also began to suffer radiation damage. Additionally this upgrade improved the read-out electronics and the cooling system of the components.

Strip detector

The strip detector surrounds the pixel detector, extending tracking of charged particles out to a radius of 113.5 cm. As indicated in Fig. 3.4, the strip detector consists of several

modules: Tracker Inner Barrel (TIB), Tracker Inner Disk (TID), Tracker Outer Barrel (TOB), and Tracker End Cap (TEC). In total 9.3 million silicon strips provide further (r, ϕ, z) hits for charged particle tracking.

3.3.3 Calorimeters

Calorimeters act to measure the energy of particles by passing them through a medium in which they will interact and deposit all of their energy. Since some particles, such as neutrinos or BSM particles, do not, or are not hypothesised to, interact readily with matter, their presence will instead be recorded as missing transverse energy ($\vec{E}_T^{\text{miss}}$ or $\vec{p}_T^{\text{miss}}$), computed by vectorially summing the transverse energy deposits. In order to improve the resolution of *true* $\vec{E}_T^{\text{miss}}$, it is necessary to ensure that $\vec{E}_T^{\text{miss}}$ caused by particles escaping the calorimeters due to gaps or passing through inactive material is minimal. Calorimeters are therefore designed to be *hermetic* in the sense that all angles are covered by active media either by angling volumes such that no openings between volumes point directly towards the nominal interaction point, or by overlapping regions of the sub-detector.

Electromagnetic calorimeter

The electromagnetic calorimeter (ECAL) is designed to measure the energies of electrons and photons, and to allow hadrons and muons to pass through unperturbed. The active volumes are lead-tungstate (PbWO_4) crystals in which the electrons and photons interact, creating sharp pulses of scintillation light, which are recorded by photodetectors at the ends of the crystals.

As shown in Fig. 3.5, the ECAL consists of two sections: the barrel, with an outer radius of 1.77 m covering $|\eta| \leq 1.479$, and the end-caps, covering $1.479 < |\eta| \leq 3.0$. Additionally a preshower calorimeter is located in front of the ECAL end-caps covering $1.653 < |\eta| \leq 2.6$. This consists of lead radiators that initiate electromagnetic showers from incoming particles, and silicon strip detectors to record the energies of the showers. This submodule is useful identifying short-lived neutral pions that decay to di-photons prior to the hadronic calorimeter, and might otherwise be detected as single, high-energy photons by the ECAL.

Hadronic calorimeter

The hadronic calorimeter (HCAL) is designed to measure the energies of hadronic particles, whilst allowing muons to pass through. Due to the presence of solenoid starting at $R = 2.95$ m, the HCAL consists of a submodule within the solenoid and an outer section surrounding the solenoid, in order to ensure that all hadrons are absorbed by the active medium.

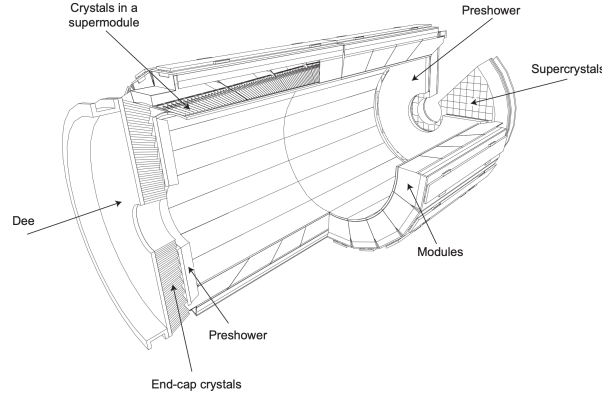


Figure 3.5: Layout of the CMS electromagnetic calorimeter. Image source [66].

Figure 3.6 details the layout of the HCAL: The inner (HB) and outer barrels (HO) both cover $|\eta| \leq 1.3$; The end-caps (HE) cover $1.3 < |\eta| \leq 3.0$; and forward calorimeters (HF) 11.2 m from the nominal interaction-point cover $3 < |\eta| \leq 5.2$.

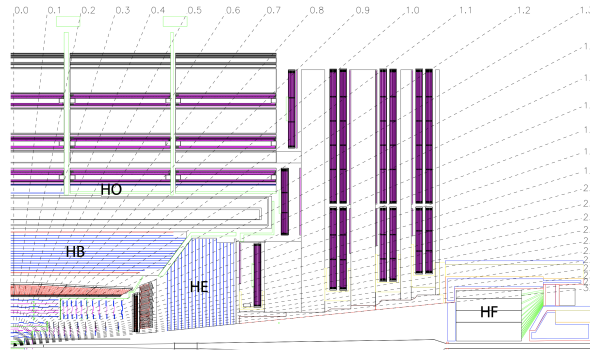


Figure 3.6: Layout of the CMS hadronic calorimeter. Purple regions indicate the muon system. Image source [66].

The absorber material for the HB, HE, and HO consists of layers of brass plates with steel plates forming outer and inner layers. These are interspaced with plastic scintillators to record the showers of particles produced by interactions with the absorber plates. Plates and scintillators are arranged to form *towers* of equal pseudorapidity, as shown in Fig. 3.7. The HO is a slight exception in that it uses the solenoid as an absorber, except for the low $|\eta|$ region where an additional absorber layer is placed due to the flight path of particles there otherwise passing through a lower thickness of absorber material. The HF consists of a large steel absorber into which fibres are inserted in an arrangement that makes it possible to distinguish electromagnetic and hadronic showers (the ECAL does

not cover this pseudorapidity range).

Phase-I upgrades to the HCAL [70, 71] involve improvements to both the front- and back-end electronics, and replacement of the photodetectors with more modern technologies. These will help the HCAL deal with the higher luminosity expected during Run-3 of the LHC. Backend electronics and HF upgrades were completed prior to 2016 data taking, and HB and HE upgrades were completed in 2019.

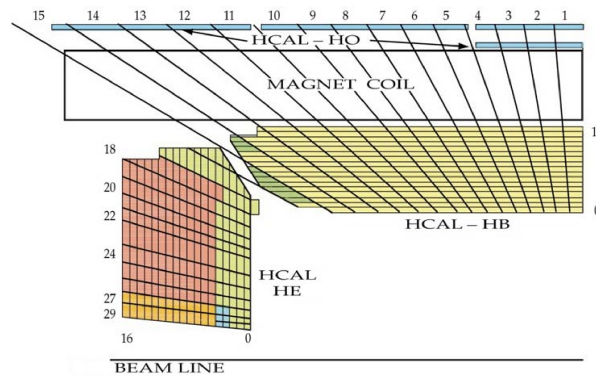


Figure 3.7: Focused layout of the CMS hadronic calorimeter detailing the tower arrangement of the absorber plates and tile scintillators. Image source [66].

3.3.4 Muon system

Muons are able to penetrate through material to a greater degree than other detectable particles. The CMS detector is therefore designed with the muons system as the outermost detector (in transverse distance) such that by the time muons reach this part of the detector, nearly all of the other particles will have been absorbed in the calorimeters. This means that efficient detection of muons can provide very clear signals on which to trigger the recording of an event. Additionally, by placing the muon system within the solenoid's flux-return yoke, the magnetic field can be used to refine momentum measurements, improving the invariant-mass resolution of particles decaying to di-muons.

The muon system consists of a barrel region and end-cap regions, and three types of gas-based muon-detectors are used. Drift-tubes (DTs) in the barrel cover $|\eta| \leq 1.2$ providing up to 32 (r, ϕ) positions and up to 12 z positions. Cathode Strip Chambers (CSC) are used in the end-cap regions ($0.9 < |\eta| \leq 2.4$), where muon rates are higher, and provide up to 24 (r, ϕ, η) positions which can be analysed to reject non-muon hits and to match detected muons to tracks in the inner tracker.

Indeed, both the muon system and the inner tracker can be used to measure the momentum of muons. The Level-1 trigger, however, relies on accurate p_T thresholds using just the muon system. In order to improve triggering, a complementary system

of Resistive Plate Chambers (RPCs) is used to provide sharp p_T thresholds with coarse spatial resolution in the region $|\eta| \leq 1.6$. Six layers of RPCs are used in the barrel and three layers in each end-cap. The layout of the muon system is detailed in Fig. 3.8.

3.3.5 Forward detectors

The Centauro And Strange Object Research (CASTOR) and the Zero-Degree Calorimeter (ZDC) are calorimeters covering $5.2 < |\eta| < 6.6$ and $|\eta| \geq 8.3$, respectively. These are designed for heavy-ion interactions as well as diffractive proton physics. The PPS [72] is located some 200 m from the nominal interaction-point and designed to study protons which interact but do not dissociate. These detectors are not used in this thesis and will not be discussed further.

3.3.6 Trigger system

With a nominal bunch-spacing of 25 ns, the frequency of events is 40 MHz, with multiple collisions per event. At about 1 MB per event this is too high a data-rate to possibly store every single event, however only a subset of the collisions will be actually be of scientific interest. By triggering on events likely to fall into this subset, a large reduction in data-rate can be achieved, reducing it down to a more manageable level. This is done in two stages: the Level-1 and the High-Level Trigger.

Level-1 trigger

Being the first level of triggering, the main priority of the Level-1 (L1) trigger [73] is that it is quick to run per event, which necessitates that it is hardware-based. Field-Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs) are used to run comparatively basic algorithms using primitive reconstruction of particles in the muon system and calorimeters with course granularity, as indicated in Fig. 3.9.

The L1 trigger is designed to have a maximum output rate of 100 kHz, but in practice is lower to allow a sufficient safety margin.

High-Level Trigger

The High-Level Trigger (HLT) [74] is designed to further reduce the acceptance rate to around 1000 times lower, however with the large reduction already applied by the L1 trigger, the HLT has a higher time-budget enabling the use of more complex, software-based algorithms at full granularity. Full detector information is transferred to C++ for processing in CMSSoftWare (CMSSW) [75] on a farm of $\mathcal{O}(1000)$ CPUs, which run a variety of trigger algorithms (the *trigger menu*) in about 50 ms. Additionally they

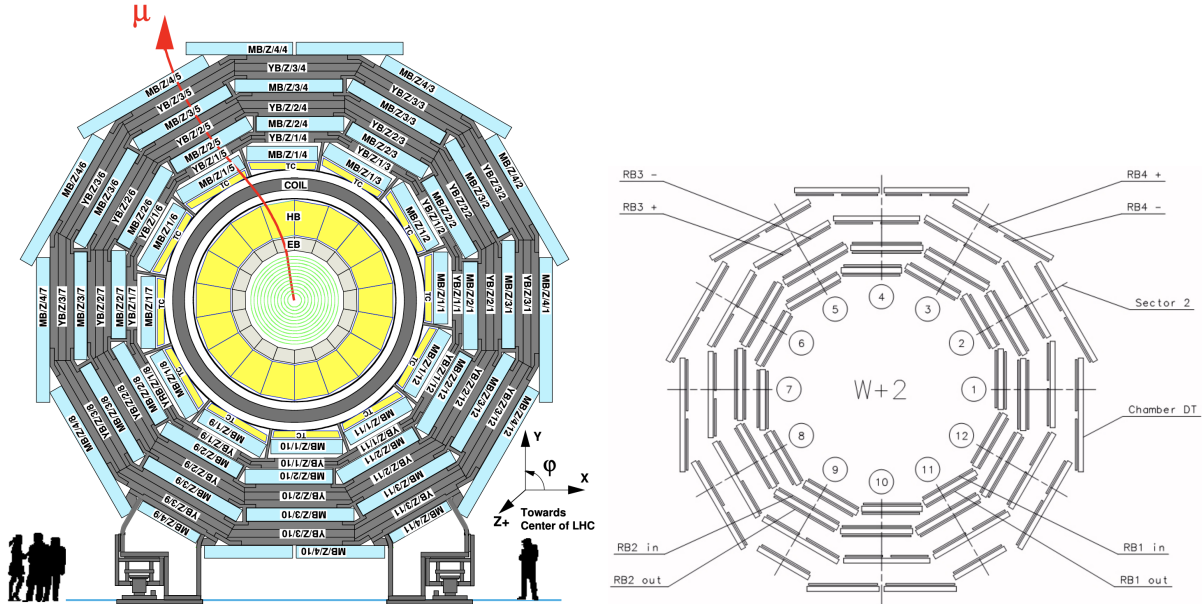
supply logging information for data-quality monitoring. Events passing the HLT are then transferred for long-term storage and eventual analysis.

3.3.7 The MIP timing detector and other high-luminosity upgrades

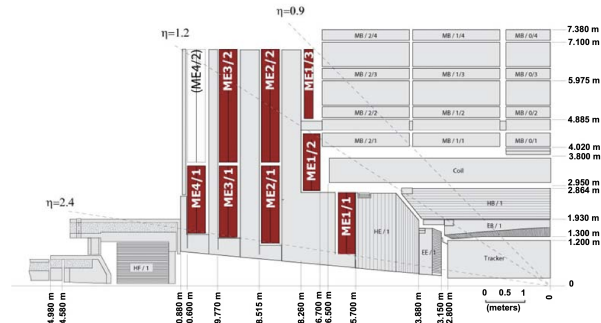
Following the CMS Phase-I upgrades and Run-3 of the LHC, CMS Phase-II upgrades are planned to improve and ready the detector for operation at the HL-LHC. These upgrades include further improvements to the inner tracker [76], high-granularity calorimeters [77, 78], and the minimum ionising particle timing detector (MTD) [79], which is designed specifically to measure time-of-arrival information to help deal with the huge increase in pile-up collisions (about 200 collisions per bunch crossing). A sensitivity projection study for the HL-LHC is summarised in Sec. 9.9 and assumes the presence of the MTD. The study also includes a section dedicated to the impact of different timing precisions for the MTD.

The MTD is designed to provide time-of-arrival information for tracks crossing it. The event can then be analysed in time-slices, which reduces the spatial density of tracks thereby improving vertex resolution by reducing the chance of tracks being associated to the wrong vertices. This in turn helps to mitigate the high-pileup conditions of the HL-LHC. As well as reducing the impact of pileup, the improved vertex precision can lead to increases in b -tagging efficiency (a type of reconstruction algorithm discussed further in Sec. 3.4.4). The removal of pileup tracks leads to improvements in the efficiency of identifying isolated objects, such as leptons and photons, and increased accuracy when calculating the kinematic properties of jets (clusters of particles described further in Sec. 3.4.3). It is also possible that the MTD could contribute information to the level 1 trigger.

Whilst the ECAL can be adapted to provide some timing information, its precision is limited, particularly in the case of low-energy photons and minimum ionising particles (MIPs). To achieve the required sensitivity, a dedicated detector, the MTD, is required. This is proposed to consist of two sections: The barrel timing layer (BTL), inserted between the tracker and the ECAL; and the end-cap timing layer (ETL), placed just prior to the end-cap calorimeters. These additional detectors are expected to allow timing information to be measured with precisions in the range of 30-50 ps.



(a) Transverse slice of the CMS detector detailing muon propagation through the detector. Cyan indicates the muon Drift Tubes. (b) Transverse slice of the CMS detector detailing the positions of the Resistive Plate Chambers, mounted on the sides of the Drift Tubes.



(c) Longitudinal slice of the CMS detector illustrating the positions of the Cathode Strip Chambers (red). Resistive Plate Chambers are mounted on the inner sides of the CSCs

Figure 3.8: Layout of the CMS muon system. Image sources [66].

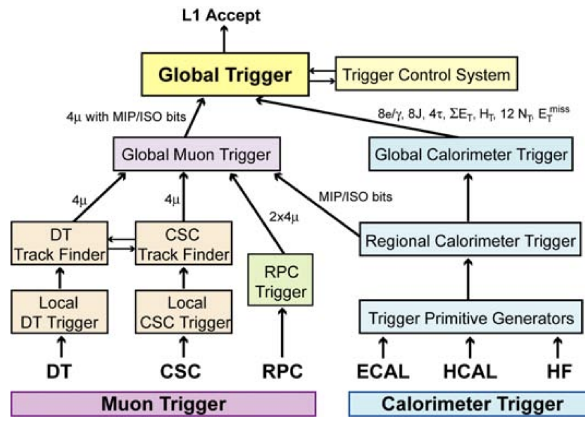


Figure 3.9: Diagram of the Level-1 trigger basing its acceptance decision on information from the muon system and the calorimeters. Image source [66].

3.4 Event reconstruction

The detector subcomponents described in Sec. 3.3 provide measurement capabilities for a variety of particles, and some of these measurements are complementary, such as muon tracking in both the inner tracker and the muon system. In components such as the calorimeters, incoming particles react with the detector creating showers of particles. In the majority of particle physics analyses we are most interested the properties of the incoming particles, rather than the properties of every single particle in the resulting shower. However, in the case of the hadronic calorimeters, these incoming particles will likely be the result of hadronised partons from a shower of QCD interactions. Again we will most interested in the original particles or interactions which gave rise to these *jets* of hadrons than every single hadron contained within them. As well as the particles we can detect, there are neutrinos which will pass through the detector without readily interacting with it. Additionally there may well be certain new, beyond Standard Model particles which also do not interact with our detector. The presence of the neutrinos and such BSM particles can be inferred by computing the vector of energy necessary to balance the vectorial sum of detected energies in the transverse plane.

Ultimately, particle interactions in the detector show up as electrical signals. A series of *reconstruction* algorithms are required to combine these signals into information more easily understandable to physicists. Considering that we are most interested in the original particles which gave rise to the signals in the detector, at CMS a compact set of *physics objects* are reconstructed per event: electrons, muons, photons, jets, and missing transverse momentum. Given the presence of complementary measurements, CMS uses a *particle flow* algorithm to combine different sub-detectors to give a better reconstruction of the physics objects.

3.4.1 Tracking

As described in Sec. 3.3.2, the CMS inner tracker records the positions of charged particles. However, since the tracker is arranged in layers, the coordinates are discrete in either radius (in the barrel region) or z (in the end-caps). Reconstructing the trajectories of charged particles therefore requires connecting detector hits. Given the high multiplicity of particles, though, this process is non-trivial since many possible combinations of hits are possible. Indeed at the HL-LHC tracking could well become a limiting factor due to the vast increase in pile-up collisions and was the subject of a recent data-science competition, the Tracking Machine Learning challenge (TrackML) [80], designed to help improve both the speed and accuracy of track reconstruction.

The tracking process currently used by the CMS detector, as described in Ref. [81], reconstructs both tracks and primary vertices:

Track reconstruction Track reconstruction uses an algorithm called the Combinatorial Track Finder (CTF). This is based on combinatorial Kalman filters [82, 83, 84, 85], a class of algorithms designed to combine multiple noisy measurements in order to, in this case, predict trajectories.

In order to lower the complexity due to the number of possible combinations, iterative track-finding is used. In this process, tracks which are more easily findable (such as the those with a high p_T) are reconstructed first and their hits are “removed” from further consideration. This process continues until the remaining tracks, which would otherwise be more difficult to find, are now more obvious due to the removal of many of the detector hits. Proto-track seeds are generated from combinations of two or three hits, Kalman filters are then used to extrapolate the trajectories and search for hits in the expected regions to add to the candidate tracks. Fitting and quality procedures are then used to reject tracks or refine measurements of the track parameters. Five track parameters extracted: d_0 , z_0 , ϕ , $\cot \theta$, and p_T . d_0 , z_0 are the radial and z positions of the hit closest to the beam line. ϕ and θ are the azimuthal and polar angle of the momentum vector.

Using an Intel Core i7-960 CPU, track reconstruction for a typical $t\bar{t}$ event takes 0.68 s, increasing to 3.58 s [81] with just eight pile-up collisions. Run conditions at the HL-LHC are expected to involve an average of 200 pile-up collisions. For HLT use, however, a version of tracking using only the pixel detector may be used, which is much quicker, or the algorithm can be seeded using the a L1 track-trigger.

Vertex reconstruction Having reconstructed the tracks in an event, it is then possible to accurately resolve the primary vertices of the event; the “signal” vertex and other vertices due to pile-up collisions. This involves selected tracks consistent with being promptly produced (produced during a collision rather than via a secondary decay) in the primary interaction region. Selected tracks are then passed through an annealing process which clusters the tracks into an arbitrary number of candidate vertices whilst also rejecting outlier tracks. The positions of each vertex are then determined using an *Adaptive Vertex Fitter* [86], which also assigns a weight to each track according to the probability of them belonging to their assigned vertex.

3.4.2 Particle flow

As mentioned earlier, the sub-components of the CMS detector can be used to provide complementary measurements of the same types of particles, therefore rather than relying on single sub-detectors to be solely responsible for reconstructing each physics object, CMS employs an approach in which multiple sources of information are used to identify particles and determine their properties. This is called the *particle flow* (PF) algorithm [87, 88, 89]. Generally, it works to combine tracks in the inner detector and muon system with energy deposits in the calorimeters and more in-depth descriptions will be given in Sec. 3.4.3.

3.4.3 Physics objects

Muons

Since muons can be tracked in both the inner tracker and the muon system, several muon collections can be generated. Tracks within the muon system are labeled *standalone muons*. Standalone muons which can be matched to tracks in the inner detector are combined with those tracks and used to form *global muons*. Inner detector tracks meeting momentum requirements are extrapolated to the muon system. If they can be matched to at least one muon segment, they are labeled as *tracker muons*. Global muons and tracker muons with common inner tracks are merged. Global muons typically have a better momentum resolution, but due to having to pass through a lot of detector material and produce multiple hits in the muon system have a higher p_T requirement than tracker muons. Charged hadrons which are not fully absorbed by the HCAL can *fake* being muons by creating hits in the muon system (punch-through), and the trade off between efficiency and purity can be adjusted by placing requirements on the muon tracks.

The PF algorithm then takes the global muons and applies isolation requirements based on surrounding tracks and energy deposits: the sum of p_T and E_T within a ΔR separation of 0.3 must not exceed 10 % of the muon's p_T . This rejects fake muons due to hadrons.

Global muons failing the isolation requirement can potentially be recovered via the *tight muon selection* [90], which places more stringent requirements on the matching between standalone muons and inner tracks, and requires the candidate to have been merged with a tracker muon with hits in at least two muon stations.

Muons failing the tight-selection can still be used provided they either have a high-quality track in the muon system a large number of hits, or they have a high-quality track in the inner detector with a large number of hits and compatible deposits in the calorimeters.

Muon momenta are computed from their tracks. For p_T less than 200 GeV the inner tracks are used, otherwise the tracks are chosen according to their goodness of fit for various possible track combinations.

Building elements associated with identified muons are removed from consideration when building other PF objects. It is still possible, though, that muons failing selection requirements could be selected using looser criteria if their building elements cannot be assigned to other physics objects.

Electrons and isolated photons

Several methods for electron reconstruction are possible. The *ECAL-based* method uses ECAL clusters of 5×5 crystals with $E_T > 4$ GeV to infer the hit positions in the inner tracker. Since electrons will emit bremsstrahlung photons, the ECAL clusters are extended in η and ϕ to form *superclusters* in order to capture this energy and so improve their

resolution [91]. Isolation requirements are normally applied to ECAL electrons since extrapolating to the inner tracker in a busy region (e.g. in a jet) leads to severe misreconstruction. Additionally, low- p_T electrons will have their bremsstrahlung spread over a large area of the ECAL, most of which will be missed by the supercluster, preventing accurate extrapolation back to the inner detector for track identification.

Complementing the ECAL-based approach, the *tracker-based* method takes tracks with $p_T > 2 \text{ GeV}$ as potential seeds for electrons. These are extrapolated to the ECAL and associated with ECAL clusters, requiring the ratios of cluster energy to track momentum to be compatible with one. Since the emission of hard or soft photons by the electrons can spoil the quality of the track fit, selected tracks are re-fitted using Gaussian-Sum Filters (GSF) [92] which are more amenable to sudden changes in energy. Discriminating variables of the track fit under both GSF and CTF along with variables such as the distance between the ECAL extrapolation point and the ECAL cluster are passed to a Boosted Decision Tree (BDT) trained to identify true electrons. Electron seeds from both the tracker and ECAL methods are collected for further selection.

Reconstruction of electrons and isolated photons is performed together: Electrons candidates are seeded by GSF tracks linked to ECAL clusters provided they are not linked to three or more other tracks. Photon candidates are seeded from ECAL superclusters with $E_T > 10 \text{ GeV}$ which are not linked to GSF tracks. The HCAL energy within $\Delta R = 0.15$ of ECAL electrons and photon candidates is computed and required to be less than 10 % of their supercluster energy. Energy corrections are applied to the ECAL clusters based on analytic functions of energy and pseudorapidity, and the final energies of the electron and photon candidates are computed.

Electrons are identified using BDTs acting on kinematic variables from the ECAL and HCAL, and track fitting from both GSF and CTF. Isolated photons are identified by requiring that their ECAL clusters are isolated from other tracks and clusters and that their HCAL clusters are compatible with a photon shower. Tracks and clusters associated with electrons and isolated photons are removed from further consideration when reconstructing remaining particles.

Hadrons and non-isolated photons

Remaining deposits and tracks will be due to charged and neutral hadrons, and non-isolated photons and muons from decays-in-flight of unstable hadrons.

Within the tracker acceptance ($|\eta| < 2.5$) neutral hadrons and photons are reconstructed from HCAL and ECAL clusters, respectively, provided those clusters have no associated tracks. Outside the tracker acceptance, charged and neutral hadrons cannot be distinguished and both will deposit some energy in the ECAL. For this region, ECAL clusters with associated HCAL clusters are assigned to hadrons, where as ECAL clusters without associated HCAL clusters are assigned to photons. Identified hadrons and photons then undergo energy calibration.

Jets

The reconstruction of hadrons and non-isolated photons from hadron decays is likely due to the decay of some heavy particle such as a top-quark or tau-lepton, or a QCD interaction giving rise to a parton shower. As discussed earlier, at analysis level we are not normally concerned with the kinematics of every single particle within these showers, but rather the properties of the originating particle or interaction. Jet clustering is a method of gathering the detector-level particle into groups called jets, such that the properties of their origins can be inferred. Having clustered particles into jets, *tagging* and *identification* algorithms are run on them to predict what interaction or decay gave rise to them.

Jet reconstruction In simple terms jet clustering may be thought of as drawing a cone of radius R in (η, ϕ) space about some point and collecting all the particles within the cone into a single jet. Indeed “cone” algorithms operate on this principle. However, they bring several problems: what criteria should be used to set the centres of the cones?; how should particles be allocated in the case of overlapping cones? Additionally, from a theoretical point of view, the jets should not be changed due to a collinear splitting of partons, or the radiation of a soft gluon (e.g. the multiplicity and location of jets should not change simply because a low- p_T gluon was radiated). These Infrared and Collinear requirements (IRC-safety) allow the experimental reconstruction of an event to be tied back to the partonic structure of the hard-process that gave rise to the collision. Another desirable characteristic is that the jet boundary is resilient to soft particles, i.e. the presence of pile-up particles should not dramatically alter the shape of the jet.

The Seedless Infrared-Safe Cone (SIS Cone) algorithm [93] is IRC-safe, resistant to pile-up, and handles overlapping regions with a split-merge procedure, however it underestimates the true radii of jets leading to underestimations of the jet energies.

The type of jets mostly used now by particle colliders use a *sequential recombination* algorithm. Considering a set of objects (initially particles and eventually proto-jets) with transverse kinematic k_t (normally E_T or p_T), distance measures may be defined thusly:

$$d_{i,j} = \min(k_{t,i}^{2p}, k_{t,j}^{2p}) \frac{\Delta R_{i,j}^2}{R^2},$$

$$d_i = k_{t,i}^{2p}, \tag{3.5}$$

where $\Delta R_{i,j}$ is the separation between object i and j in (η, ϕ) , and R and p are parameters which alter the trade-off between the transverse kinematic and separation. Sequential recombination proceeds iteratively:

1. Compute all $d_{i,j}$ and d_i for all remaining objects
2. Find the minimal value of $d_{i,j}$ and d_i

3. If the minimal value is a member of $d_{i,j}$, then objects i and j are merged together, otherwise the minimal value is a member of d_i and object i is designated a jet and removed from further clustering.
4. Repeat until either all objects have been clustered into jets, or a predetermined stop-condition is met.

Since hard assignments of particle to jets are made and no cone radius is used, the problem of overlapping jets is implicitly avoided.

Three common settings for p in Eq. 3.5 exist:

- $p = 1$ gives the k_t algorithm [94]. This has a tendency to cluster softer particles together first, approximately inverting the QCD branching process.
- $p = 0$ results in the Cambridge/Aachen (C/A) algorithm [95]. This purely considers the distance separation of objects, clustering in order of closest separation. This is useful in studies of jet *substructure*, where jets might contain multiple decays: As an example, a boosted top-quark jet can contain two jets resulting from a hadronically decaying W boson and a b -quark jet (three sub-jets). Clustering via C/A and then undoing the last to clustering step should yield the b -jet and w jets.
- $p = -1$ is the anti- k_t algorithm [96]. This generally clusters harder particles first and implicitly leads to circular jet areas which are unaffected by pile-up. It was the default clustering algorithm at CMS during Runs 1 and 2.

A comparison of SISCone and the three sequential recombination algorithms is shown in Fig. 3.10. From Fig. 3.10a we can see that Both k_t and C/A lead to highly irregular jets shapes, whereas anti- k_t and SISCone both give circular jets. SISCone, however, gives much smaller jet radii, which leads to poorer p_T resolution, as shown in Fig. 3.10b.

Jets in particle flow Jet clustering in CMS normally involves running the anti- k_t algorithm over hadrons and non-isolated photons as computed by PF. This results in jets which better match the “true” jets (as computed from Monte Carlo simulation, for example) in terms of p_T and direction than jets computed using on calorimeter clusters (*Calo jets*), as shown in Fig. 3.11. Still, the jet energies will not match entirely due to detector inefficiencies, therefore jets undergo energy corrections prior to analysis use. CMS typically reconstructs jets with $R = 0.4$ and $R = 0.8$ for “fat jets”, in which the decay products of heavy, boosted particles might be clustered into a single jet.

Pile-up mitigation Whilst the presence of pile-up collisions has a limited effect on jet shapes when using the anti- k_t algorithm, they nonetheless add to jet energy. Pile-up mitigation for jets aims to account for pile-up being included in jets, thereby improving

the accuracy of jet reconstruction. The method currently used by CMS is Pile-Up Per Particle Identification (PUPPI) [97]. This is a global process in which the surrounding regions of every particle are examined to compute weights for each particle according to how *pile-up-like* they are. These weights are then used to rescale jet momenta.

Missing transverse momentum

As discussed earlier, particles which do not interact with the detector, e.g. neutrinos, can still be inferred by computing the amount of missing energy in the event. Whilst the exact momentum carried by the colliding particles is not known, what is known is that the initial transverse momentum of these particles is zero. Therefore the amount of sum of transverse momentum of the particles after the collision should also be zero. Computing the vector of momentum in the transverse plane required to balance the recorded transverse momentum thereby provides a handle for inferring the presence of undetected particles in the collision. This vector is referred to as the missing transverse momentum, $\vec{p}_T^{\text{miss}}$, or (assuming massless particles) the missing transverse energy, $\vec{E}_T^{\text{miss}}$. These are occasionally referenced in text as MPT or MET.

The *raw* MPT is computed as the negative of the vectorial summation over all PF particles:

$$\vec{p}_T^{\text{miss}}(\text{raw}) = - \sum_{i=1}^{N_{\text{particles}}} \vec{p}_{T,i}. \quad (3.6)$$

Since jets undergo energy corrections to account for detector losses, the *jet-energy-corrected* MPT is computed as:

$$\vec{p}_T^{\text{miss}} = \vec{p}_T^{\text{miss}}(\text{raw}) - \sum_{j=1}^{N_{\text{PF jets}}} [\vec{p}_{T,j}^{\text{corr.}} - \vec{p}_{T,j}], \quad (3.7)$$

where $N_{\text{PF jets}}$ only includes jets with $p_T \geq 10 \text{ GeV}$. The MPT used by CMS is computed using PUPPI-corrected jets

3.4.4 Jet tagging

As mentioned earlier, we are mostly interested in working with the particles which give rise to the showers of hadrons that are collected in the calorimeters. Jet clustering provides us with approximations of the kinematic properties of these origin particles, however it does not provide us with explicit details of the type of particle. *Tagging* algorithms further examine jets in order to try to identify the types of origin particles.

A range of tagging can be performed and often this is analysis-dependent. In this thesis, two important sources of jets are those originating from the hadronisation of bottom quarks (*b* jets) and tau leptons which decay hadronically ($\tau \rightarrow \nu_\tau w \rightarrow \nu_\tau q \bar{q}$) (hadronic taus, tau jets, τ_h , or simply ‘taus’).

***b*-jet tagging**

b tagging is normally based on exploiting the displaced secondary vertex arising due to the relatively long lived *b* hadrons formed during *b*-quark hadronisation (as illustrated in Fig. 3.12), along with the higher invariant mass of *b* jets compared to quark/gluon jets ($m_b \approx 4 \text{ GeV}$). Variables capturing this information are then combined to form a single discriminant. In certain decay chains of *b* hadrons a soft lepton may be produced, which can provide additional sources of discriminating features.

Secondary vertices (SVs) during Run-2 of CMS were identified using the Inclusive Vertex Finding algorithms (IVF). This selects seed tracks from the collection of all reconstructed tracks based on p_T and impact-parameter requirements. These tracks are then cluster requiring that the separation at the point of closest approach between pairs of tracks is less than the minimum distance of the tracks to the primary vertex. The Adaptive Vertex Fitter is then used to fit the track clusters to form SVs, discarding those which are not sufficiently displaced from the primary vertex (PV) and removing tracks which are more compatible with the (PV) than they are with their SV(s). Duplicated SVs are also resolved.

CSV The Combine Secondary Vertex tagging algorithm [99] is used for jet tagging by passing various features of the jet, and associated tracks and secondary vertices to a discriminator and producing tagging scores on a per jet basis. The original version of the algorithm used a likelihood ratio for the discriminator, whereas version 2 (CSVv2) [98], that was used for analyses during the beginning of Run-2 by CMS, instead uses a wider range of variables and neural networks for the discriminator.

It is applied on a per jet basis:

1. Jets must have at least two tracks, and track pairs with invariant masses within 30 MeV of the mass of the K_S^0 meson are removed.
2. Jets without tracks and secondary vertices have a CSV score of -1 .
3. Jets are then exclusively categorised:
 - **RecoVertex**: has at least one SV.
 - **PseudoVertex**: has no SV, but does have a pair of tracks with high impact-parameters and invariant mass not within 50 MeV of the mass of the K_S^0 .
 - **NoVertex**: remaining jets no assigned to the other categories.
4. Variables characterising the SV, the tracks, and the jet are computed, according to the vertex category.

5. The variables are passed through a pair of neural networks according to the vertex category; one to classify b jets and light-parton jets ($udsg$), and the other to classify b jets and c jets. One pair of networks is used per vertex category; three pairs of networks exist in total.
6. The CSV score of the jet is computed using a weighted combination of its two b -jet predictions; b/c score weighted by one and $b/udsg$ weighted by three.

DeepCSV DeepCSV [98] builds on CSVv2, using more input features, better preprocessing and training, and a deeper neural network, which performs multi-class classification; the network prediction is a probability of the jet belonging to each of five categories. The categories used are:

- Exactly one b hadron.
- Exactly one c hadron and zero b hadrons.
- At least two b hadrons.
- At least two c hadrons and zero b hadrons.
- $udsg$ light jet.

DeepFlavour The DeepFlavour tagger [100] (also presented as the generic jet-tagging algorithm DeepJet in Ref. [101]) further extends DeepCSV by including more information about charged and neutral PF objects, as well as additional SVs. This lower-level information goes through initial processing in the form of convolutional and recurrent layers, before being combined with the global features of the jet, as shown in Fig. 3.13.

Algorithm performance and usage Figure 3.14 indicates the performance of each of the three algorithms discussed. From this we can see that DeepFlavour currently provides the lowest mistag rates for a given b tagging efficiency. As with any new algorithm in a large collaboration, however, it has taken sometime to propagate through to usage in analyses.

Whilst each jet will be assigned a b -tag score, in actual application, official working points (WPs) in the form of score thresholds are prescribed. These WPs correspond to set to particular tagging efficiencies, to help control the trade-off between acceptance and purity; the Loose WP provides greater efficiency but also lower purity (meaning that more $udsg$ and c jets will be tagged as b jets), where as the Tight WP will provide a smaller, but highly pure collection of b -jets.

Hadronic taus

With its high mass of (1776.86 ± 0.12) GeV, the tau is the only lepton in the Standard Model capable of decaying to hadronic final-states, via $\tau \rightarrow \nu_\tau w \rightarrow \nu_\tau q \bar{q}$. These hadrons will then lead to the formation of a jet. The selection of taus is done in two stages: First, tau candidates are identified using the Hadrons Plus Strips (HPS) algorithm [103, 104, 105]; Next, candidates are required to pass quality checks to remove fake taus due to electrons, photons, and jets.

The HPS algorithm The aim of the HPS algorithm is not just to provide hadronic-tau candidates, but also to determine the *decay mode* (DM) of the tau - the category of hadronic-level final-state of the tau decay in terms of charged- and neutral-hadron multiplicity.

Acting on a PF jet, the algorithm proceeds thusly:

1. Jets are rejected if their charge is not ± 1 , or if they contain charged particles outside of their *signal cone*, defined along the jet axis and radius $3/p_{T,\text{jet}}$ (clipped between 0.05 and 0.10).
2. The hadronic components of the jet are identified by using calorimeter strips in (η, ϕ) space to collect deposits left by electrons and photons resulting from the hadron decays.
3. Strip and charged-particle multiplicity information is used to form the decay mode hypothesis for the tau.
4. Constraints are placed on the visible mass reconstructed from the hadrons, according to the decay hypothesis.

For Run-2, *dynamic* strips are reconstructed, in which electrons or photons iteratively seed, or merge with existing, strips in order of decreasing p_T , with the strip position shifting with each merger and new strips seeded when no e/γ is within a close enough window of the strip centre. The p_T -weighted centre of a given strip is required to be within the jet's signal-cone. The use of dynamic strips build on the fixed-strips used during the Run-1 implementation, and better account for particle interactions and scattering which would otherwise lead to deposits outside of the strips.

Four main decay modes are considered, and a jet compatible with one of these are labelled as a tau candidate:

1. $h^\pm \pi^0$: Requiring one charged particle, one strip, and $1.3 - \Delta m < m_{\tau_h} < \sqrt{p_{T,\tau_h}/100} \times 1.3 \text{ GeV} + \Delta m$, where Δm is the additional mass due to the e/γ candidates and the upper mass-limit is clipped within 1.3-4.2 GeV.

2. $h^\pm\pi^0\pi^0$: Requiring one charged particle, two strips, and $0.4 - \Delta m < m_{\tau_h} < \sqrt{p_{T,\tau_h}/100} \times 1.2 \text{ GeV} + \Delta m$ and the upper mass-limit is clipped within 1.2-4.0 GeV.
3. $h^\pm h^\pm h^\mp$: Requiring three charged particles and $0.8 < m_{\tau_h} < 1.5 \text{ GeV}$.
4. $h^\pm$: Requiring one charged particle and no strips.

Later implementations of HPS also include additional and *recovery* decay modes to further improve acceptance, and BDT-based assignment of candidates to decay modes, rather than solely relying on multiplicity and mass criteria.

τ_h discrimination While the HPS algorithm selects true taus, it is also possible that other particles can also pass its requirements, resulting in fake taus. The three considered sources are electrons, muons, and non-tau jets, and quality cuts are normally made against each source independently.

The rejection of jets and electrons both use BDT discriminators taking relevant features as inputs, whereas the muon discriminant is based on two possible sets of conditions:

- **Loose:** Track segments in at least two muon planes are within a $\Delta R = 0.3$ of the jet axis OR the energy of the hardest charged hadron in the jet is less than 20 % of its track momentum.
- **Tight:** **Loose** OR a hit is located in any muon detector in the two outermost stations within a $\Delta R = 0.3$ of the jet axis.

Tau candidates that match their set of muon conditions or fail cuts on their electron or jet discriminator scores are vetoed.

DeepTau Similar to the impact that DeepCSV and DeepFlavour had on the performance of b -tagging (Sec. 3.4.4), true tau discrimination from faked taus can also benefit from the application of deep learning. DeepTau [106] acts on tau candidates passing HPS to provide multi-class classification into electron, muon, hadronic tau, and light jet classes. It consists of a deep neural-network running convolutional filters over (η, ϕ) cells populated by jet constituents, which are then combined with higher level features. Figure 3.15 details the architecture layout.

Performance comparisons to the previous set of three discriminators detailed in Sec. 3.4.4 are shown in Fig. 3.16. From this we can see that DeepTau consistently provides large reductions in mis-tag rates over the old discriminators for all three sources of fake taus. Given that DeepTau was only released in 2019, it has yet to be applied in many analyses, and further improvements are envisaged such as also providing the decay-mode assignments to HPS candidates.

3.4.5 Post-reconstruction algorithms

Following the reconstruction of the physics objects, depending on the analysis to be performed it is sometimes useful to run further algorithms on the physics objects in order to further improve reconstruction. In the analyses performed in this thesis, two particular algorithms are used to improve the mass resolution of $h_{\tau^+\tau^-}$ and the di-Higgs mass: **SVFit** and **HHKinFit**.

SVFit

As mentioned in Sec. 3.4.4, the decay chain of a hadronic tau involves a neutrino, which cannot be explicitly reconstructed in the CMS detector. In the case of leptonic decays of the tau, two neutrinos are emitted, $\tau \rightarrow \nu_\tau w \rightarrow \nu_\tau \bar{\nu}_\ell \ell$ $\ell \in \{e, \mu\}$, further increasing the missing energy. In the case of a particle, such as the Higgs boson, decaying to di-tau, this loss of energy in the neutrinos leads to a miss-reconstruction of the invariant mass of the particle. Secondary Vertex Fit (**SVFit**) [108, 109] is a fitting algorithm designed to improve the reconstruction of decaying particle's 4-momenta by assuming that the particle decays to di-tau. It was first used in the analysis of CMS Run-1 data [110] and has further been improved for Run-2 [108] and the latest version [109] also includes a more advanced computation method involving the Matrix Element Method [111].

The basic method uses a dynamic likelihood approach to reconstruct the di-tau mass on a per-event basis, using MPT to constrain the kinematics of the neutrinos in the decay. These are combined in a probability model which includes terms for the resolution of MPT and the kinematics of the tau decays. The likelihood function takes the form:

$$p(\vec{x}|\vec{y}, \vec{a}), \quad (3.8)$$

the probability to observe MPT:

$$\vec{x} = (p_x^{\text{miss}}, p_y^{\text{miss}}), \quad (3.9)$$

given the unknown kinematics of the tau decays:

$$\vec{a} = (X_1, \phi_1, m_{1,\nu\nu}, X_2, \phi_2, m_{2,\nu\nu}) \quad (3.10)$$

and the known kinematics of the visible products of the tau decays:

$$\vec{y} = (p_{1,\text{vis.}}, p_{2,\text{vis.}}). \quad (3.11)$$

Marginalising the unknown parameters allows the mass hypothesis to be computed:

$$p(M_{\tau\tau}^i) = \int d\vec{a} \delta(M_{\tau\tau}^i - M_{\tau\tau}(\vec{y}, \vec{a})) M_{\tau\tau}(\vec{y}, \vec{a}). \quad (3.12)$$

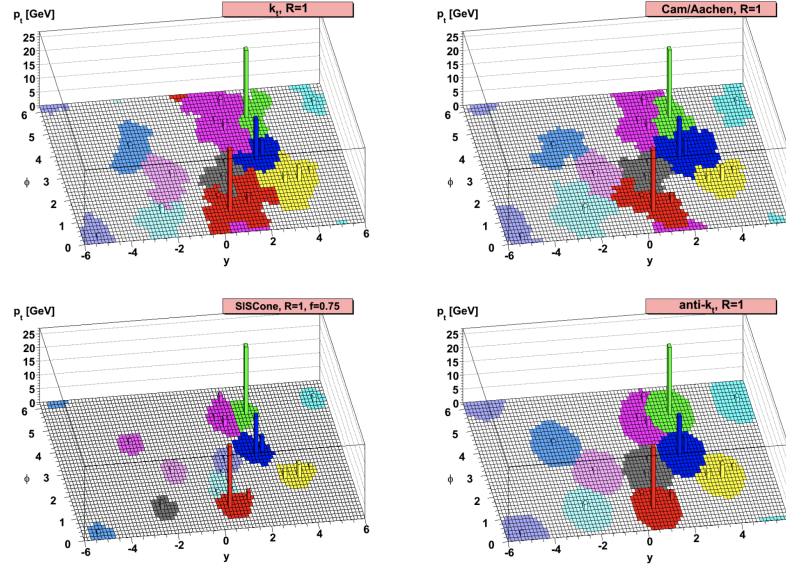
The probability density functions (PDFs) for hadronic and leptonic tau decays are modelled using matrix elements (see Sec. 3.5.4) computed from particle theory, and the PDF for MPT is modelled by assuming that the neutrinos produced in the di-tau decay are the only source of missing energy in the event. These PDFs are used to compute $M_{\tau\tau}(\vec{y}, \vec{a})$ and $M_{\tau\tau}(\vec{y}, \vec{a})$.

Equation 3.12 is used to compute the PDF of $M_{\tau\tau}^i$ for a range of masses ($[5, 2000]$ GeV). The mass with the highest probability is the **SVFit** mass, and the uncertainty can be taken from the 0.16 and 0.84 quantiles of $p(M_{\tau\tau}^i)$. Full computation takes about 1 s on a CPU and the impact of **SVFit** is demonstrated in Fig. 3.17.

HHKinFit

Similar to **SVFit** **HHKinFit** [112] is a kinematic fit designed to improve a mass measurement, in this case that of a pair of Higgs bosons. In the case of $hh \rightarrow \tau^+ \tau^- b \bar{b}$, miss-measurements can arise not only from the miss-reconstruction of $h_{\tau^+ \tau^-}$ due to the neutrinos but also mis-reconstruction of $h_{b \bar{b}}$ due to jet-energy resolution and the potential for neutrinos to be present in the hadronisation of the b quarks.

HHKinFit assumes that the ΔR between the visible final-states of a di-Higgs decay and the direct products of the di-Higgs decay is minimal, i.e. the visible products of each tau decays are collinear with the original taus, and the b -jet axes are collinear with the b quarks. This is reasonable considering that the large difference in mass between a Higgs boson and a tau lepton or b quark will result in a large boost. On top of this collinear assumption is the mass constraint that the invariant mass of the b -quark pair and the τ -lepton pair must both equal the Higgs boson mass. These constraints and assumption reduce the free parameters of the fit down to just two energies, which are fitted via a χ^2 minimisation. As will be seen in this thesis, the χ^2 value of the fit is also a useful variable for rejecting background; whilst the fit can converge, since the assumptions made are valid only for signal (actual di-Higgs events) the fit will be poor, resulting in a larger χ^2 .



(a) Comparison of jet shapes

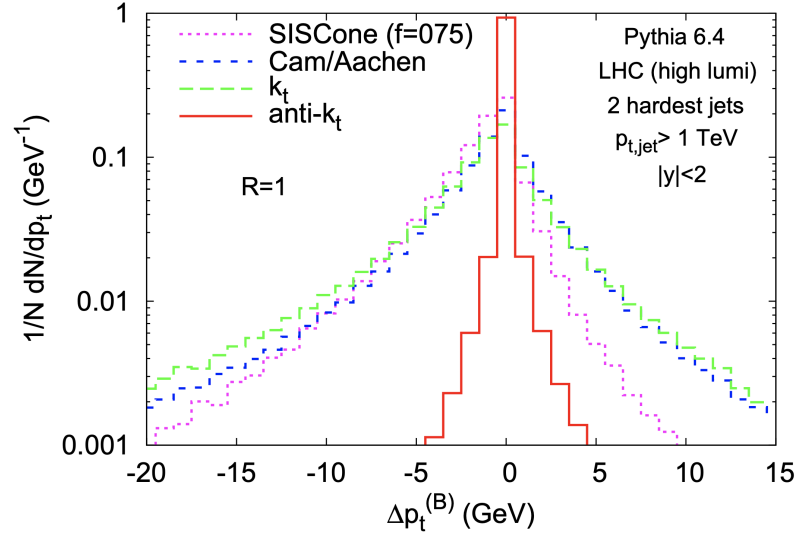
(b) Comparison of p_T resolution for different jet algorithms.

Figure 3.10: Comparison of four different jet-clustering algorithms. k_t and C/A are both affected by pile-up, as shown by their irregular shapes in (a) which leads to poor p_T resolution in (b). $SISCone$ and $anti-k_t$ are less affected by pile-up, however $SISCone$ gives narrower jet radii and so underestimates their p_T , whereas $anti-k_t$ better reconstructs the jet p_T . Image sources [96].

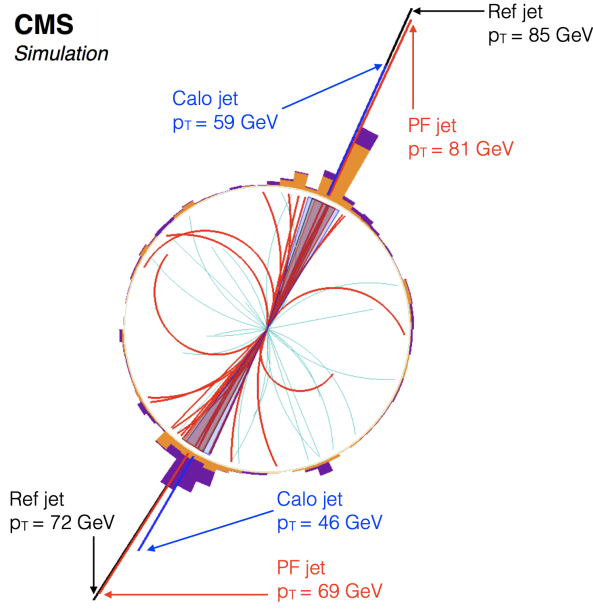


Figure 3.11: Performance comparison of PF and Calo jets in reconstructing reference jets from simulation. PF jets offer improvements in both p_T and direction by more closely matching the reference jets than the Calo jets do. Image source [66].

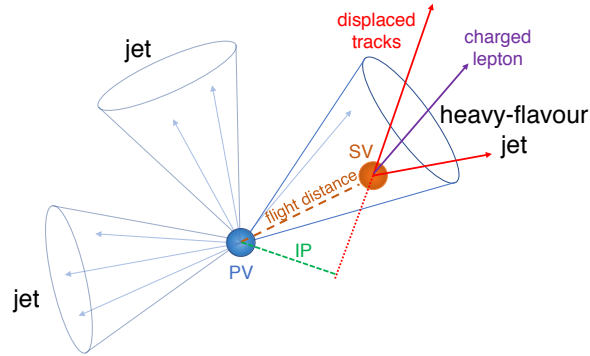


Figure 3.12: Illustration of the secondary vertex arising from the long-lived b hadron in a b -quark hadronisation. Image source [98].

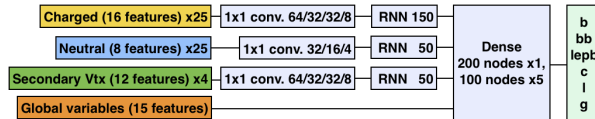
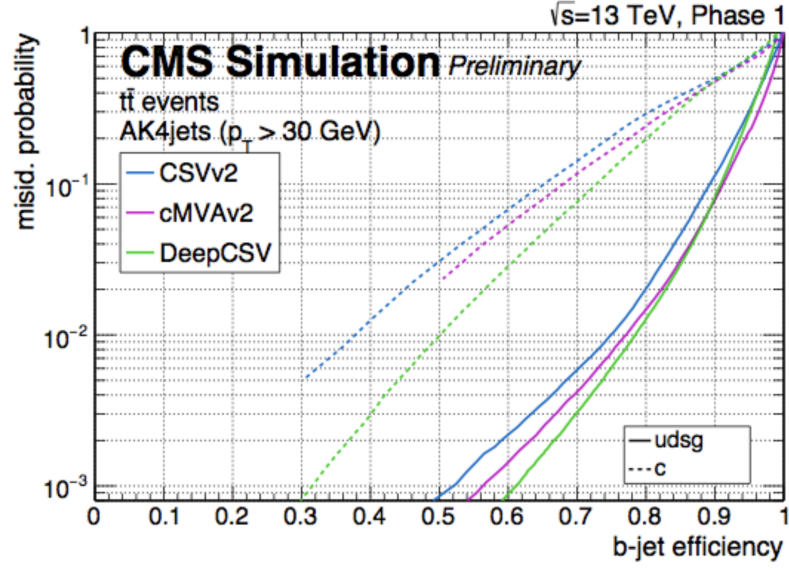
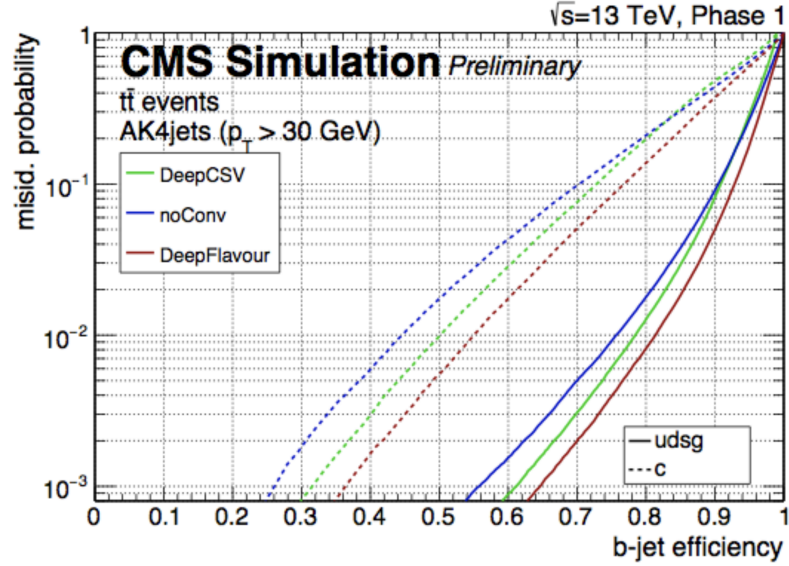


Figure 3.13: Network schematic for the DeepFlavour tagger. Image source [102].



(a) Comparison of DeepCSV to CSVv2



(b) Comparison of DeepFlavour to DeepCSV

Figure 3.14: Comparison of the three taggers discussed examining b -jet identification against $udsg$ and c jets. Plots indicate the mis-tag rate for given b -tag efficiencies; lines closer to the bottom-right provide better performance. “NoConv” is a version of DeepFlavour without the convolution layers, and “cMVA v2” is tagger that computes a score based on inputs from several other CMS algorithms, including CSVv2. Image sources [100].

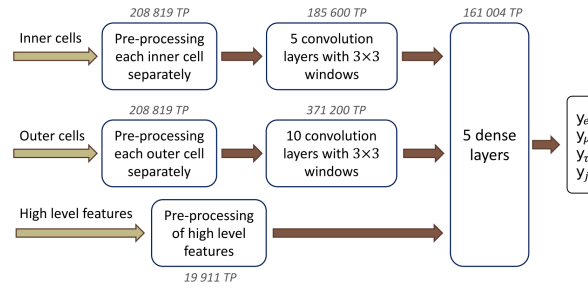


Figure 3.15: Network schematic for the DeepTau discriminator. Image source [107].

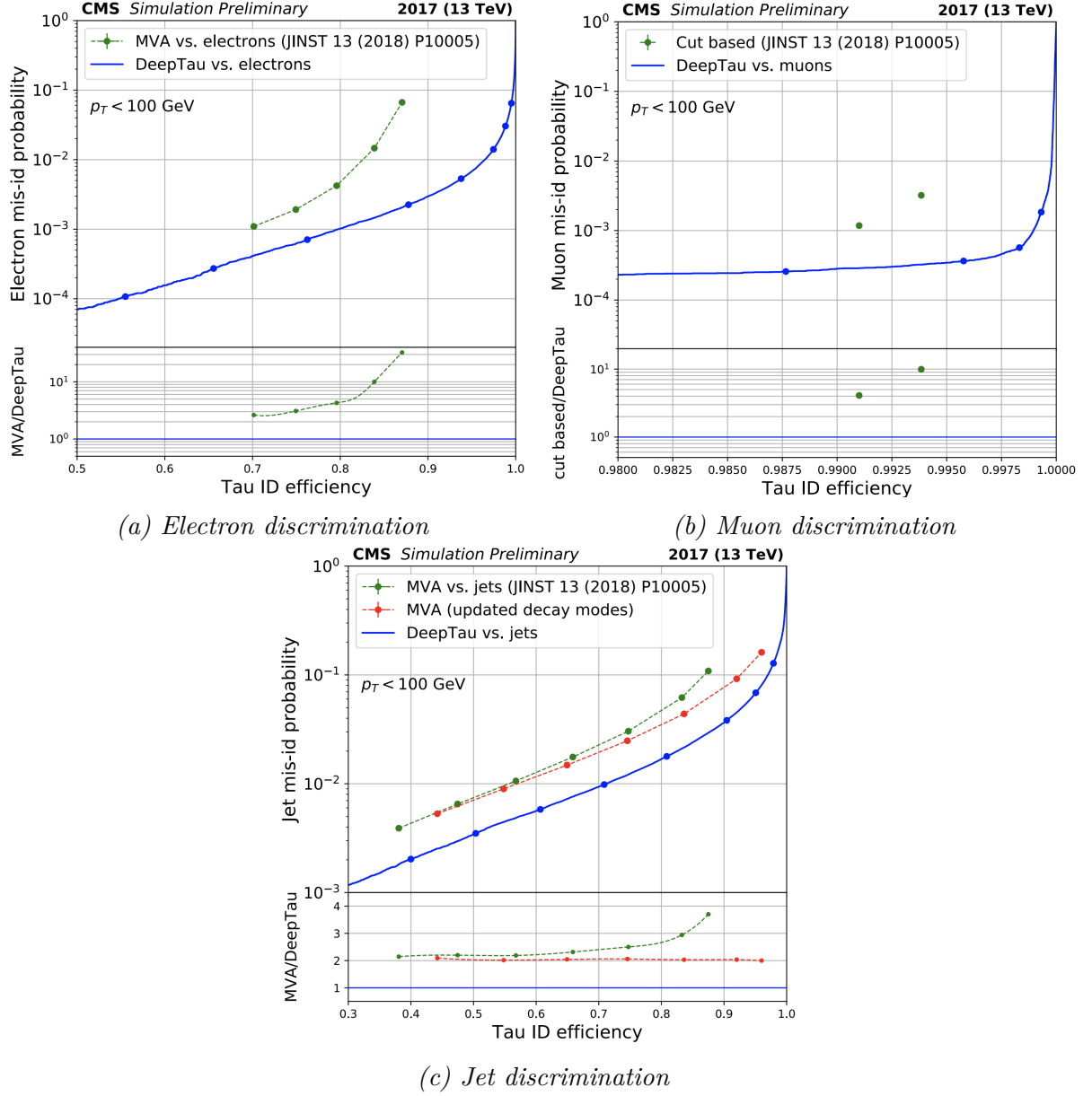
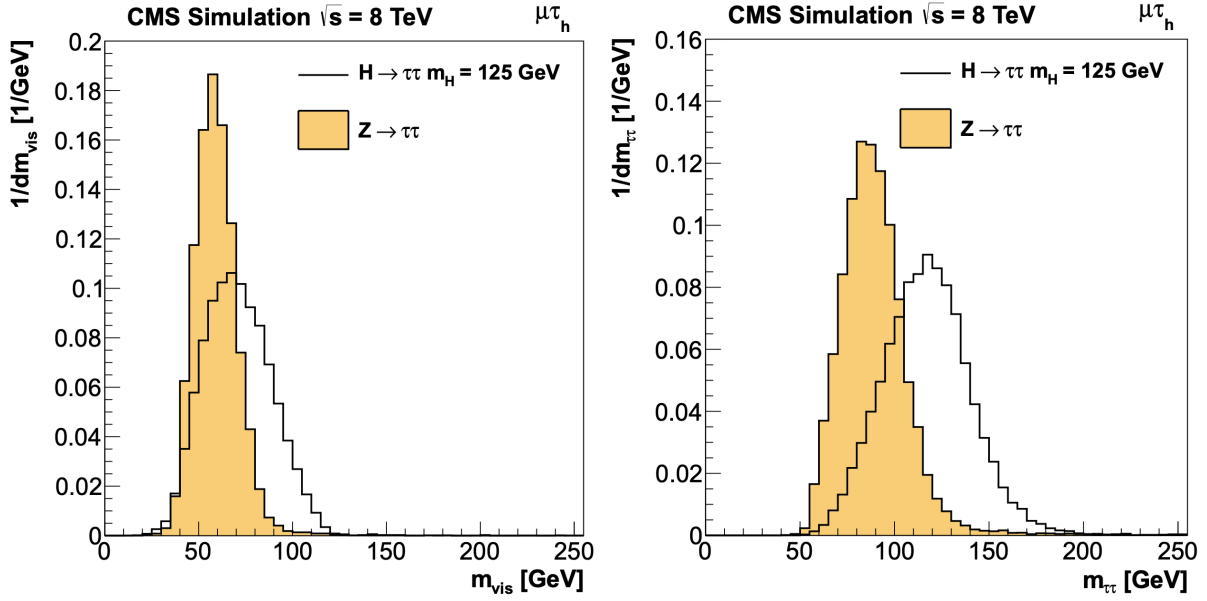


Figure 3.16: Comparison of the three older discriminators discussed in Sec. 3.4.4 against the new DeepTau (v2) discriminator. Plots indicate the mis-tag rate for jets, electrons and muons for given τ_h efficiencies; lines closer to the bottom-right provide better performance (i.e. a higher efficiency for the same mis-tag rate). The points indicate the established working points for each discriminator. Image sources [106].



(a) Di-tau mass reconstruction using visible particles (b) Di-tau mass reconstruction using *SVFit*

Figure 3.17: Impact of *SVFit* on mass reconstruction for two sources: $Z \rightarrow \tau^+ \tau^-$ ($m_Z = 91 \text{ GeV}$) and $h \rightarrow \tau^+ \tau^-$ ($m_h = 125 \text{ GeV}$). The application of *SVFit* in (b) serves to centre the reconstructed distributions about their respective true masses. Image sources [108].

3.5 Monte Carlo simulation of particle collisions

3.5.1 Overview

As we will see throughout this thesis, there are many scenarios in which it is useful to be able to model the realistic evolution of particle processes, whilst knowing exactly what interactions occurred during the evolution, and what initial process gave rise to the collision. Monte Carlo (MC) generators are a class of program that are designed to provide such realistic simulations by using a mixture of analytic and stochastic functions based on our current knowledge of particle theory, empirical observations of particle phenomenology, and optionally any new hypothetical interactions the user may wish to explore.

The fact that parts of the Standard Model can self-interact, means that simulations can easily become very complicated to compute. Most notable is QCD, which can only be easily computed in the *perturbative* regime (pQCD) of high-momenta partons operating over short distances. This computational restriction limits how much of the collision simulation may be calculated analytically in the *matrix element*, or evolved stochastically in the *parton shower*, before empirical models have to be used.

In the case of simulating the collision of two hadrons, the two prominent areas where these empirical models must be used are: the sampling of initial partons from the colliding hadrons, where precomputed parton distribution functions (PDFs) are used to bridge the gap between macro-scale interactions and the micro-scale partonic interactions which can be computed in pQCD; and in the hadronisation of partons resulting from the collision, which allows the simulation to move from the computationally tractable high-momenta partons, to the low momenta bound hadrons, which may subsequently decay to produce the final-state particles of the collision.

The MC simulation therefore consists of four main sections: PDF sampling, analytical matrix element calculations, stochastic evolution in the parton shower, and empirical modelling of hadronisation.

3.5.2 Cross section computation

Each possible particle process has an associated *cross-section* (σ), which is effectively the likelihood for that process to occur in a collision. Full computation of the cross section therefore involves integrating over all possible ways in which the given process can occur, subject to the collider specifications. Considering the scattering process in which two partons (a and b) interact to produce a given final-state (n), the hadronic cross-section

$(\sigma_{ab \rightarrow n})$ may be calculated according to:

$$\sigma_{ab \rightarrow n} = \int_0^1 dx_a \int_0^1 dx_b \int d\Phi_n f_a^{h_1}(x_a, \mu_F) f_b^{h_2}(x_b, \mu_F) \times \frac{1}{2\hat{s}} |M_{ab \rightarrow n}|^2(\Phi_n; \mu_F, \mu_R), \quad (3.13)$$

where: Φ_n is the partonic final-state phase-space (range of possible outgoing momenta), which may be restricted based on generational requirements; $f_i^{h_j}(x_i, \mu_F)$ is the PDF of parton i , which depends on x_i , is the fraction of momentum carried by parton i and μ_F , the factorisation scale; $|M_{ab \rightarrow n}|$ is the matrix-element (ME) for the scattering process, which sums over all possible ways in which the partonic final-state n can be produced within the phase-space Φ_n through the interactions of the incoming partons a and b , within the perturbative regime dictated by the renormalisation scale μ_R and μ_F ; and $\frac{1}{2\hat{s}} = x_a x_b s$ is referred to as the parton flux, which is a function of the momenta fractions of both incoming partons, and the hadronic centre-of-mass energy, $\sqrt{s}$. [38]

3.5.3 Parton distribution functions

As mentioned in Sec. 2.2.4, hadrons don't just consist of their valence quarks, but also virtual quark-anti-quark pairs called *sea quarks*, and gluons. All of these components carry different fractions of the hadron's total momentum. These fractions are summarised by *parton distribution functions* (PDFs), which can be used to lookup the probability density that a given parton has a given fraction of momentum of the hadron at a given energy scale [113]; the $f_i^{h_j}(x_i, \mu_F)$ terms in Eq. 3.13. Examples of two PDFs at different energy scales are shown in Fig. 3.18.

3.5.4 The matrix element

The matrix element (ME) refers to the part of the simulation that is analytically computed using particle theory in the perturbative regime. The process computed is referred to as the *hard process*, and is normally the main interaction for the process which is being generated. As an example, for simulating non-resonant di-Higgs production via gluon fusion, the hard process of $g g \rightarrow h h$, and its hadronic cross-section, would be computed in the ME via pQCD, using gluons sampled from a PDF, and Monte Carlo integration over the phase-space.

ME calculations are said to be performed at a given *order*. At *leading order* (LO) only the simplest possible ways of producing the requested outgoing particles are considered (tree-level Feynman diagrams). However the self-interacting nature of some sections of the Standard Model, and the potential for virtual particles to be created and subsequently reabsorbed, means that more complicated diagrams containing such virtual loops can

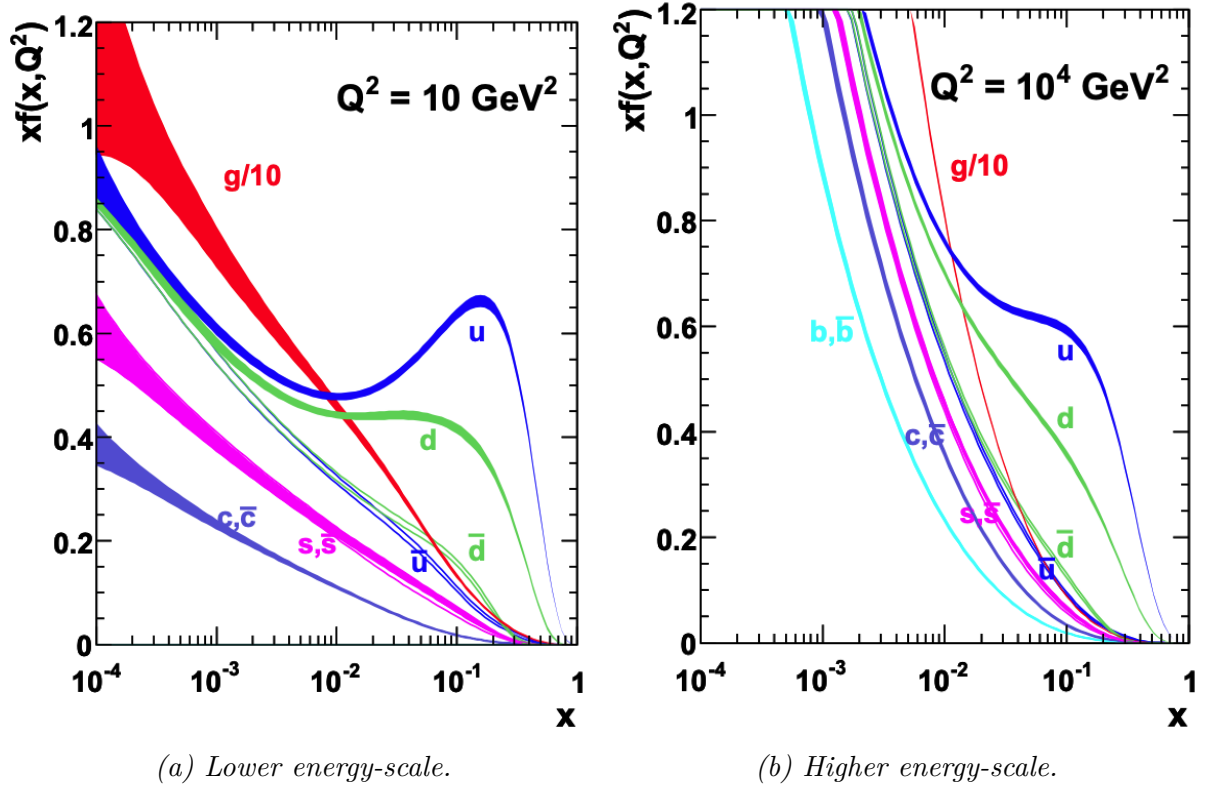


Figure 3.18: The MSTW 2008 NLO PDF at two different energy scales. The curves illustrate the probability-density (normalised to number of partons) of finding partons carrying a fraction x of the total momentum of a proton at an energy scale Q^2 . The width of the bands indicates the 68% confidence. Image source [114].

also produce the same requested outgoing particles. ME calculations which contain these loop corrections take place at *next-to-leading order* (NLO), *next-to-next-to-leading order* (NNLO), *et cetera*. Each extra loop correction, however, adds to the difficulty computing the ME, and most MC generators do not go beyond NLO.

Additional corrections for the hard process come in the form of extra outgoing partons. Taking the example of gluon scattering, $gg \rightarrow gg$, the self-interacting nature of QCD, and high momenta of the outgoing partons means these gluons are likely to radiate other gluons, $gg \rightarrow ggg$, or quark-anti-quark pairs could form via gluon splitting, $gg \rightarrow q\bar{q}g$. These extra emissions can further radiate and split, meaning that a given hard process can have a highly variable number of outgoing particles. Full, analytic computation of all possible extra emissions is far beyond the scope of contemporary ME calculations, hence the need for the parton shower stage to connect the exclusive multiplicity of the ME to the inclusive multiplicity of nature.

Process	Function
$q \rightarrow q g$	$P_{qq}(z) = C_F \frac{1+z^2}{1-z}$
$q \rightarrow g q$	$P_{gq}(z) = C_F \frac{1+(1-z)^2}{z}$
$g \rightarrow g g$	$P_{gg}(z) = C_A \frac{z^4+1+(1-z)^4}{z(1-z)}$
$g \rightarrow q \bar{q}$	$P_{qg}(z) = \frac{1}{2} (z^2 + (1-z)^2)$

Table 3.1: Summary of the spin-averaged parton splitting functions for energy fraction z , in which the quark colour-factor $C_F = \frac{N_c^2-1}{2N_c}$ and the gluon colour-factor $C_A = N_c$, for N_c number of colours (three in the Standard Model)

3.5.5 The parton shower

In the parton shower (PS), the partons exiting the ME are passed through a stochastic evolution in order to allow the MC generation to consider all possible parton multiplicities.

The key term for a parton shower is the Sudakov form-factor $\Delta_i(Q^2, q^2)$; the probability for a parton to not emit a new parton with a scale between q and Q [38]. This can be computed using:

$$\Delta_i(Q^2, q^2) = \exp \left(-\frac{\alpha_s}{2\pi} \int_{q^2}^{Q^2} \frac{dk^2}{k^2} \int_{Q_0^2/k^2}^{1-Q_0^2/k^2} dz P_{ji}(z) \right), \quad (3.14)$$

where: $P_{ji}(z)$ $i, j \in \{g, q, \bar{q}\}$ are parton splitting functions listed in Tab. 3.1, and z is the energy fraction of parton j with respect to parton i .

The choice of variable for the scale Q, q in Eq. 3.14, is not fixed, but must be chosen *a priori*. Whilst Eq. 3.14 was initially derived using θ , the opening angle between the original parton and the new emission, any variable proportional to θ^2 may be used; such as the virtuality $q^2 = z(1-z)\theta^2 E^2$ (as is used in Eq. 3.14), and the transverse momentum $p_T^2 = z^2(1-z)^2\theta^2 E^2$. In the collinear limit, when $\theta = 0$, these choices are equivalent, it is only once the scale is extrapolated from the collinear limit that the choice of evolution variable begins to have an effect [38]. The choice of evolution variable is referred to as the *ordering* of the shower, e.g. using θ results in an *angular-ordered* shower.

Evolution of a parton shower involves attempting to construct parton splittings on each outgoing parton from the ME, recursively. For each splitting, the Sudakov form factor is sampled from a uniform distribution between zero and one and the right-hand-side of Eq. 3.14 is then solved for q^2 . If q^2 is greater than a preset cut-off scale Q_0^2 , then a branching is created with scale q^2 , otherwise evolution of that parton ceases. This continues until evolution of all partons has stopped.

Parton-mass effects

The formulation for parton showers so far has assumed that the partons are massless. In order to account for the masses of quarks, or any coloured massive particle, certain modifications are made. HERWIG[115] used an angular-ordered shower and treated branchings of massive partons with an opening angle above some threshold the same as massless partons, but then vetoed all branchings below the threshold. This *dead-cone approximation*, however, leads to a discontinuity in the splittings.

Current approaches either: make process-dependent matrix-element corrections [116], the approach used by PYTHIA 8 [117]; or generalise the splitting kernels in Tab. 3.1 to *quasi-collinear splitting functions*, which account for quark masses [118], the approach used by SHERPA and HERWIG++ [119, 120].

Dipole showering

The splittings and PS evolution described so far involve $1 \rightarrow 2$ splitting kernels, which locally violate momentum conservation. In implementation, a global *reshuffling* of momentum is then used to restore conservation. More recent PS implementations, such as PYTHIA 8 and SHERPA, instead use $2 \rightarrow 3$ splitting kernels based on colour dipoles between quarks or gluons. These allow momentum to be explicitly conserved locally allowing any parton recoils to effect the subsequent evolution of other dipoles. Additionally, this provides the transverse momentum of dipoles to be a natural choice for the PS evolution variable.

3.5.6 Matrix element and parton shower combination

Most MC generation for particle physics at the LHC uses combination of both ME and PS simulation; using the ME to compute the interactions of hard partons with (semi-)exclusive multiplicity, and the PS to describe softer, near-collinear partons with inclusive multiplicity. Combining both forms of simulation, however, must be done with care in a process of *matching* or *merging*.

For LO MEs with fixed final-state multiplicity, the PS can be matched simply by showering the outgoing partons based on their kinematics. It can be useful, however, to allow the ME to compute some of the partonic evolution, since the initial branchings are often harder, or far from collinear, and therefore better described by the ME. SHERPA [121], for instance, allows one to compute semi-exclusive MEs, which have the potential to include a limited number of extra partonic emissions. Passing such MEs to a PS, however, requires a more advanced merging technique to avoid under counting, or over counting, phase-space regions, such as in Ref. [122]. NLO MEs also require specialised matching schemes, such as MC@NLO [123], POWHEG [124], and MEPS@NLO [125].

3.5.7 Hadronisation

Partonic evolution in the PS ends because the partons can no longer be described accurately using pQCD, however the lack of observed free colour-charge in nature means that the partons must be combined into colourless hadrons. This stage of *hadronisation* uses empirical models to cluster and confine the partons. Two common models are: the Lund string model [126, 127], used by PYTHIA, in which the gluon field between partons collapses to a thin tube (strings); and the cluster model [128], used in SHERPA and HERWIG++, based on *pre-confinement* of partons into colour-neutral clusters.

Hadrons and other particles exiting the hadronisation stage may be unstable, and so are allowed to decay, optionally via external packages such as EVTGEN [129] and TAUOLA [130, 131, 132].

3.5.8 Multi-parton interaction

The scattering process described by the ME involves two incoming partons sampled from PDFs, however in the collisions of hadrons, there can be multiple interactions between the composite particles, not just the hard process; multi-parton interaction (MPI). MPI can be simulated by including extra MEs for $2 \rightarrow 2$ QCD scattering, and showering them during the PS, however the exact settings used are normally computed beforehand and released as *tunes*, e.g. Ref. [133].

3.5.9 Detector simulation

The final stage for producing simulated data that is as close as possible to the data which might actually be collected by a specific detector is to simulate the interactions between the final-states of the hadronisation and decay processes with that detector. This is necessary since: not all particles will be detected by the detector, either due to detection inefficiencies, or from the absorption of low-energy particles in the earlier stages of the detector; and the kinematics of the registered particles are not measured with exact precision. Additionally, pile-up simulation may be added to recreate the expected running conditions of the LHC.

Detector simulation can either be performed by fully accounting for all the volumes and materials within the real-life detector using packages such as Geant [39, 134], or by trading accuracy for speed by using parameterised responses for the sub-detector components in packages like DELPHES [40, 135, 136].

Chapter 4

Background explanations: Statistical methods for discoveries

4.1 Hypothesis testing

The LHC was designed primarily to be a discovery experiment, aiming to provide the flexibility to run a large number of various searches for new physics. This flexibility also allows it to be used to study and to perform measurements of known processes, such as rare decays. In the case of searches, these are performed in terms of testing for the presence of some new particle or production mechanism (signal). The problem is that the LHC lacks the control to produce specific collisions, e.g. one cannot request the production of 5000 $t\bar{t}$ events or to attempt to produce a BSM particle with requested properties, as one could in a MC simulation. Instead the collisions recorded at the LHC will be a mixture of a variety of different collision processes, proportional to their hadronic cross-sections. Additionally, unlike simulated data, there are no *truth* labels for each collision; one cannot tell for certain exactly which process gave rise to a particular collision. This means that when attempting to discover signal, one must also contend with the background of collisions produced by other processes, and must do so by comparing the observed yield of events with what would be expected if the signal process were present or if production were purely through the background processes; a hypothesis test between the null hypothesis of background only (H_b) and the alternative of signal and background (H_{s+b}).

4.1.1 Simple-hypothesis testing

For a given phase-space of the data, signal and background production mechanisms can be expected to contribute to the total number of events populating the phase-space (yield) at random according to fixed rates; the yield therefore has a Poisson distribution, with a

probability to record n events being equal to:

$$\mathcal{P}(n|s, b) = \frac{(s+b)^n}{n!} e^{-(s+b)}, \quad (4.1)$$

where s and b are the expected yields from signal and background, respectively. If the number of observed events in the phase-space region is n_{obs} we can perform a hypothesis test to reject, or not, the background-only hypothesis by computing the probability of the background producing at least n_{obs} events, i.e. integrating the tail-end of the Poisson distribution:

$$p = \mathcal{P}(n \geq n_{\text{obs}} | s = 0, b) = \sum_{n=n_{\text{obs}}}^{\infty} \frac{b^n}{n!} e^{-b}. \quad (4.2)$$

This p -value is a function of the data and so is a random variable and for convenience, it is converted to a Z -score, which corresponds to the number of standard deviations by which a Gaussian variable would have to fluctuate in order to produce such a p -value:

$$p = 1 - \Phi(Z) = \frac{1}{\sqrt{2\pi}} \int_Z^{\infty} dx e^{-\frac{x^2}{2}},$$

$$Z = \Phi^{-1}(1 - p). \quad (4.3)$$

In particle physics, the Z -score is referred to as the significance, and a significance of 3σ or more is regarded as “evidence” of the signal, whilst 5σ or more can be grounds for “discovery” of the signal (probability of being incorrect less than 2.9×10^{-7}). For a discussion on the history and appropriateness of this convention one may read Ref. [137]. Full acceptance of the signal hypothesis also depends on the plausibility of the new process, and the how well it describes the observed data.

The sensitivity of the test may be increased by measuring the expected and observed yields as a function of some feature of the data, x , e.g. a kinematic variable. In HEP, histograms are normally used, providing a set of discrete bin populations $\vec{n} = \{n_1, n_2, \dots, n_N\}$, but unbinned density estimation methods may instead be used. The populations of each bin are:

$$n_i = s_i + b_i, \quad (4.4)$$

where

$$s_i = s_{\text{tot.}} \int_{\text{bin } i} dx f_s(x)$$

$$b_i = b_{\text{tot.}} \int_{\text{bin } i} dx f_b(x), \quad (4.5)$$

with $f_s(x)$ and $f_b(x)$ as the probability-density functions (PDFs) of signal and background for variable x . Equation 4.1 now becomes modified to take the product over every bin:

$$\mathcal{P}(n|s, b) = \prod_{i=1}^N \frac{(s_i + b_i)^{n_i}}{n_i!} e^{-(s_i + b_i)}. \quad (4.6)$$

This can be continued into higher-dimensions by considering the signal and background densities parameterised over many features of the data, $\vec{x} = \{x_1, x_2, \dots, x_o\}$, but for simplicity I will stay with a single feature.

4.1.2 Composite hypotheses

Equations 4.1 and 4.5 (and hence 4.6) assume that the signal and background yields b , and their analytic distribution in x , are known, however in particle physics these factors must be estimated using MC generated data, or extrapolated from signal-depleted phase-space regions. This means that they are subject to not only statistical uncertainty, but also systematic uncertainties in the modelling and overall yield. These *nuisance* parameters mean that the (differential) yields of signal and background are uncertain:

$$\begin{aligned} s_i &= s_{\text{tot.}} \int_{\text{bin } i} dx f_s(x; \vec{\theta}_s) \\ b_i &= b_{\text{tot.}} \int_{\text{bin } i} dx f_b(x; \vec{\theta}_b), \end{aligned} \quad (4.7)$$

in which $s_{\text{tot.}}$ and $b_{\text{tot.}}$ are nuisance parameters on the overall yields of signal and background, and $\vec{\theta}_s$ and $\vec{\theta}_b$ are nuisance parameters affecting the PDFs of signal and background in x . For ease later on, we normally assume that the signal yield $s_{\text{tot.}}$ is known, and instead multiply it by a coefficient known as the *signal strength* parameter, μ . The expected yields in each bin are now:

$$E[n_i] = \mu s_i + b_i. \quad (4.8)$$

Henceforth, $\boldsymbol{\theta} = (\vec{\theta}_s, \vec{\theta}_b, b_{\text{tot.}})$ will be used to represent all nuisance parameters in the model. Hypotheses with nuisance parameters are referred to as *composite hypotheses*.

For some nuisance parameters it may be possible to assign functional forms to them, and for others it may be possible to make subsidiary measurements, such as in signal-depleted regions of the data space (control regions). These help to constrain the values of the nuisance parameters, providing estimated values $\hat{\boldsymbol{\theta}}$. These can be taken in product with the Poisson distributed event counts to form a likelihood function:

$$\mathcal{L}(\mu, \boldsymbol{\theta}) = \mathcal{P}(n | \mu s + b; \boldsymbol{\theta}) \cdot p(\tilde{\boldsymbol{\theta}} | \boldsymbol{\theta}). \quad (4.9)$$

4.1.3 The likelihood ratio as a test statistic

The variable used when performing a hypothesis test is called the *test statistic*, t , which is a function of the observed data and is used to either accept or reject the hypotheses. The

Neyman-Pearson lemma [138] states that the most powerful test statistic is (a monotonic function of) the likelihood ratio of the two hypotheses considered:

$$t(\mathcal{D}; H_b, H_{s+b}) = \frac{\mathcal{L}(H_b)}{\mathcal{L}(H_{s+b})} = \frac{p(\mathcal{D}|H_b)}{p(\mathcal{D}|H_{s+b})}, \quad (4.10)$$

where $p(\mathcal{D}|H)$ is the PDF of the observed data, $\mathcal{D}$, under a given hypothesis, which must be estimated using MC or extrapolation. When dealing with composite hypotheses, however, one must use *profile* likelihood-ratios. For searches at the LHC, we can define such a ratio in terms of our parameter of interest, the signal strength μ :

$$\lambda(\mu) = \frac{\mathcal{L}(\mu, \hat{\boldsymbol{\theta}}(\mu))}{\mathcal{L}(\hat{\mu}, \hat{\boldsymbol{\theta}})}, \quad (4.11)$$

in which $\hat{\boldsymbol{\theta}}(\mu)$ is the value of $\boldsymbol{\theta}$ that maximises $\mathcal{L}(\mu, \boldsymbol{\theta})$ for a specific value of μ , and $\mathcal{L}(\hat{\mu}, \hat{\boldsymbol{\theta}})$ is the maximum value of $\mathcal{L}$ considering all values of μ and $\boldsymbol{\theta}$. This includes negative values of μ , provided that $\mu s_i + b_i$ is non-negative. For convenience, the form of test statistic used is:

$$t_\mu = -2 \ln \lambda(\mu), \quad (4.12)$$

in which higher values correspond to greater incompatibility of μ with the observed data. The p -value for such a statistic for a specific value of μ is:

$$p_\mu = \int_{t_{\mu, obs.}}^{\infty} dt_\mu f(t_\mu | \mu), \quad (4.13)$$

where $t_{\mu, obs.}$ is the observed value of t_μ in the data.

Additive-only signal

Generally, signals at the LHC only add to the mean event count over what is expected from background production alone. i.e. the signal strength is non-negative ($\mu \geq 0$). In this case a slightly modified version of the test statistic, $\tilde{t}_\mu$ may be used:

$$\tilde{t}_\mu = \begin{cases} -2 \ln \frac{\mathcal{L}(\mu, \hat{\boldsymbol{\theta}}(\mu))}{\mathcal{L}(0, \hat{\boldsymbol{\theta}}(0))} & \hat{\mu} < 0, \\ -2 \ln \frac{\mathcal{L}(\mu, \hat{\boldsymbol{\theta}}(\mu))}{\mathcal{L}(\hat{\mu}, \hat{\boldsymbol{\theta}})} & \hat{\mu} \geq 0. \end{cases} \quad (4.14)$$

This new definition accounts for the fact that for $\hat{\mu} < 0$, the best level of compatibility with the observed data occurs when $\mu = 0$.

4.2 Discovery significance

As shown in Eq. 4.2, “discovery” of signal corresponds to rejecting the background-only hypothesis (H_b) in which the signal strength is zero ($\mu = 0$). For this case, the test statistic is indicated by $\tilde{t}_\mu = \tilde{t}_0 = q_0$, which, according to Eq. 4.14, has the form:

$$q_0 = \begin{cases} 0 & \hat{\mu} < 0, \\ -2 \ln \frac{\mathcal{L}(0, \hat{\boldsymbol{\theta}}(0))}{\mathcal{L}(\hat{\mu}, \hat{\boldsymbol{\theta}})} & \hat{\mu} \geq 0. \end{cases} \quad (4.15)$$

$q_0 = 0$ for $\hat{\mu} < 0$ means that if the observed data contain fewer events than expected from background only, which may well indicate that the background hypothesis may be incorrect, it is not an indication of the presence of an additive-only signal. Instead it would point towards a systematic error. q_0 may then be converted to a p -value using Eq. 4.13:

$$p_0 = \int_{q_0, obs.}^{\infty} dq_0 f(q_0|0). \quad (4.16)$$

4.3 Confidence limits

In the absence of signal discovery, it is useful to recast the analysis in terms of finding the lowest value of the signal strength which the analysis can confidently exclude, i.e. to set an upper limit on μ . In this case the signal is assumed to be present in the data with a contribution scaled by μ . One then defines a test statistic, q_μ :

$$q_\mu = \begin{cases} 0 & \hat{\mu} > \mu, \\ -2 \ln \frac{\mathcal{L}(\mu, \hat{\boldsymbol{\theta}}(\mu))}{\mathcal{L}(\hat{\mu}, \hat{\boldsymbol{\theta}})} & \hat{\mu} \leq \mu, \end{cases} \quad (4.17)$$

and for cases with an additive-only signal:

$$\tilde{q}_\mu = \begin{cases} 0 & \hat{\mu} > \mu, \\ -2 \ln \frac{\mathcal{L}(\mu, \hat{\boldsymbol{\theta}}(\mu))}{\mathcal{L}(\hat{\mu}, \hat{\boldsymbol{\theta}})} & 0 \leq \hat{\mu} \leq \mu, \\ -2 \ln \frac{\mathcal{L}(\mu, \hat{\boldsymbol{\theta}}(\mu))}{\mathcal{L}(0, \hat{\boldsymbol{\theta}}(0))} & \hat{\mu} < 0. \end{cases} \quad (4.18)$$

Comparing q_μ and $\tilde{q}_\mu$ to q_0 we can see that q_0 is zero when the observed data fluctuate downwards ($\hat{\mu} < 0$), whereas q_μ and $\tilde{q}_\mu$ are zero when the data fluctuate upwards ($\hat{\mu} > \mu$); the reason being that when setting a limit on μ , an upwards fluctuation of data should not be regarded as evidence against the presence of signal (lower compatibility between μ and the data). As usual, these test statistics are used to compute p -values, e.g.:

$$\tilde{p}_\mu = \int_{\tilde{q}_\mu, obs.}^{\infty} d\tilde{q}_\mu f(\tilde{q}_\mu|\mu). \quad (4.19)$$

In order to set a limit, one first chooses a level of confidence (normally, 95 % confidence for HEP), and then finds the largest value of μ for which the p value is greater than, or equal to, to one minus the confidence level (e.g. $1 - 0.95 = 0.05 = \alpha$):

$$CL_{s+b} = \arg \max_{\mu} \begin{cases} 0 & \tilde{p}_{\mu} < \alpha, \\ \mu & \tilde{p}_{\mu} \geq \alpha. \end{cases} \quad (4.20)$$

This can also be thought of as computing the largest rate of signal production at which the probability that the observed data can be explained by an under-fluctuation of signal production is less than α .

It is possible, though, that the analysis being performed may have limited sensitivity to the signal, which may lead to over exclusion of the signal (perhaps even requiring a negative cross-section for the signal process due to a downwards fluctuation of the background). In order to guard against spurious exclusion, a modified limit can be computed called the CL_s [36, 37] limit, which is the ratio of CL_{s+b} and CL_b , the confidence limit under the background-only hypothesis.

4.4 Expected performance

An important advantage of the profile likelihood ratio is that as more data are collected, the impact of the nuisance parameters decreases. Reference [139] describes how using theorems from Wilks [140] and Wald [141], the discovery significance can be asymptotically approximated as:

$$Z_0 = \sqrt{q_0}. \quad (4.21)$$

This can similarly be extended to compute limits in the asymptotic approximation.

Where this approximation becomes considerably useful is in computing the expected performance of an analysis. Most analyses at the LHC are performed *blind*, meaning that the entire analysis and statistical treatment of the recorded data in the signal region(s) (the phase-space region used for hypothesis testing) is defined and fixed before applying and checking the data in the signal region. Note that this does not exclude checking data outside of the signal region to ensure agreement between MC and collider data, nor checking MC data in the signal region. This process aims to reduce biases entering into the analyses via the experimental decisions, e.g. noticing an excess of observed data and altering the analysis to specifically enhance the excess when in fact the excess was purely due to a statistical fluctuation. When designing the analysis, though, it is useful to have an optimisation metric to help guide decisions with the aim of improving the sensitivity of the analysis.

For LHC searches, the expected significance is used, which is the significance that would be computed if the data were produced under H_{s+b} (called the Asimov dataset, in which $n_i = s_i + b_i$). Since many datasets can be generated, the expected significance

provides a measure of the general performance of the analysis (the median expected significance), and the variance of performance (e.g. 1-sigma and 2-sigma bands, assuming a Gaussian distribution). Similarly, expected upper limits can be computed using the median value of the test statistic under the background-only hypothesis instead of the observed data.

When optimising parts of the analysis, it is useful to be able to compute the expected performance quickly, especially when it has to be computed multiple times, e.g. when computing the effect of some selection threshold on the expected performance. Reference [142] continues the work of Ref. [139] to provide a quick and analytic form, the Approximate Median Significance (AMS):

$$z_A = \sqrt{2 \left((s+b) \ln \left(1 + \frac{s}{b} \right) - s \right)}, \quad (4.22)$$

where s and b are the yields of signal and background respectively. This can be further extended to the case in which the background carries an uncertainty σ_b :

$$z_A = \sqrt{2 (s+b) \ln \left(\frac{(s+b)(b+\sigma_b^2)}{b^2 + (s+b)\sigma_b^2} \right) - \frac{b^2}{\sigma_b^2} \ln \left(1 + \frac{\sigma_b^2 s}{b(b+\sigma_b^2)} \right)}. \quad (4.23)$$

For the cases in which $s \ll b$, these simplify to:

$$z_A = \frac{s}{\sqrt{b + \sigma_b^2}}. \quad (4.24)$$

Chapter 5

Background explanations: Machine learning and data science

5.1 Overview of supervised learning

5.1.1 Overview of machine learning

The field of machine learning studies the problem of how to model the relationship between observational data and some desired outcome, which may or may not be explicitly described, with little to no reliance on domain knowledge. That is to say, a sufficiently flexible model is used to discover patterns and relationships between observations and targets via a data-driven approach rather than an expert in the particular field handcrafting the model. These approaches are particularly useful when either the problem domain does not have sufficiently developed theoretical framework to analytically describe the true relationships between observation and target, or such analytical relationships are expensive to compute.

Example applications could be:

- Image recognition, such as determining what breeds of animal are shown in a large number of images, which would otherwise require an expert to check each image individually;
- Language modelling, such as predicting what an internet search user is going to type next based on what they have already written and previously searched for, which would be impractical if another human were instead used;
- Unsupervised clustering of high dimensionality data into a representation more understandable to humans, which would otherwise require searching through many 1D and 2D density-distribution plots or summarising relationships as numerical coefficients;

- Reinforcement learning, such as making a robot navigate efficiently through several different environments, which would otherwise require teams of people to hard-code each individual movement for all of the environments.

The possible applications of machine learning are indeed broad, but there are three main categories in which they fall: supervised learning, unsupervised learning, and reinforcement learning. The category most applicable to this thesis is supervised learning, which I will describe next.

5.1.2 Supervised learning

Supervised learning is a two step process: a training phase and an application phase. In the training phase the model is provided with the observational data (x) and the corresponding target values (y). In the animal breed recognition example, x would be images of animals (or rather the values in the RGB channels of the pixels of the images), and y would be the animal breeds. The model algorithm then uses this data to adapt to the specific task.

During the application phase, the model is provided with new observational data, but is not given the target values and instead produces predictions for what the target values are ($\hat{y}$). This allows the model to be applied to observational data for which the target values are unknown. The performance of the model is how closely its predictions match the true target values on data which was not previously seen during the training phase.

In general there are two common tasks in supervised learning: classification and regression.

Classification

Classification tasks refer to when the targets are discrete labels. Continuing the example of animal-breed recognition, each breed would be its own individual label. Since standard ML algorithms deal with numerical data, these labels (also referred to as *classes*) are usually transformed into a vector with Boolean elements according to a predefined ordering scheme which is kept consistent for both training and application, e.g. the first element always refers to whether the breed is *Cat: British Longhair*, the second element to *Cat: Highlander, et cetera*, with a 1 corresponding to true and 0 corresponding to false.

Multi-class classification The animal-breed recognition is an example of a specific sub-category of problem called *multi-class classification*, in which each datum falls into one, and only one, class (an animal cannot be two or more breeds). The ML algorithm, however may not output a single class, but rather predicted probabilities of the data belonging to each of the classes, which sum up to one. In this case, the user may wish

to either take the class with the highest probability as being the prediction, or instead consider a few of the highest predictions.

$$y = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \dots \\ 0 \end{pmatrix}, \hat{y} = \begin{pmatrix} 0.01 \\ 0.70 \\ 0.15 \\ \dots \\ 0.05 \end{pmatrix} \quad (5.1)$$

Binary classification Binary classification is a special case of multi-class classification in which the training data only contains two classes; an example could be classifying cats and dogs. In this case, the target can either be a vector of two elements, the same as with multi-classification, or the target can be a single Boolean, with a 0 corresponding to one class, say dog, and a 1 corresponding to the other, cat.

$$y = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \hat{y} = \begin{pmatrix} 0.30 \\ 0.70 \end{pmatrix} \text{ or } y = 1, \hat{y} = 0.70 \quad (5.2)$$

Multi-label classification This type of task refers to when the model should determine whether the data has certain properties; expanding on our earlier example, the model could instead be predicting whether an animal has fur, is carnivorous, is domesticated, is old, *et cetera*. These can be thought of as multiple binary-classification problems. The key point is that for a given datum there can be several labels, whereas for multi-class problems (e.g. *Cat: British Longhair*, *Cat: Norwegian Forest*, or *Cat: Highlander*) there can be only one.

Again these labels can be represented as a vector of Boolean elements, however the model now produces a per-element probability for the datum having each particular label, i.e. there is no requirement for all the predictions to sum to one.

$$y = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}, \hat{y} = \begin{pmatrix} 0.85 \\ 0.55 \\ 0.70 \\ 0.10 \end{pmatrix} \quad (5.3)$$

Regression

In regression problems, the targets have numerical values, e.g. predicting the auction price of a used bulldozer. Since the target is already numerical, the direct output of the model can normally be used with no need to perform array indexing like a classification problem. Additionally, multiple quantities can be regressed simultaneously, e.g. predicting both the auction price and the number of bids.

5.1.3 Columnar data format

The natural format of the observational data often determines what types of ML algorithms are most applicable to the given tasks. In this thesis, the data are in a format referred to as *columnar data*. This means that the data can be represented as a table of values, such as that shown in Fig. 5.1, where each row corresponding to a separate datapoint (bulldozer sale), and each column corresponding to different *features* (observable properties of the datapoints). The **SalePrice** column in Fig. 5.1 is the target feature to be predicted, which would be unavailable during application time.

Each datapoint in columnar format can be thought of as a vector in observation space, the components of which are the column features. The task of supervised learning is then predicting the position of the datapoint in target space.

	SalePrice	YearMade	saledate	UsageBand	ProductSize	ProductGroup	ProductGroupDesc	Drive_System	Enclosure	Forks
0	66000.0	2004	11/16/2006 0:00	Low	NaN	WL	Wheel Loader	NaN	EROPS w AC	None or Unspecified
1	57000.0	1996	3/26/2004 0:00	Low	Medium	WL	Wheel Loader	NaN	EROPS w AC	None or Unspecified
2	10000.0	2001	2/26/2004 0:00	High	NaN	SSL	Skid Steer Loaders	NaN	OROPS	None or Unspecified
3	38500.0	2001	5/19/2011 0:00	High	Small	TEX	Track Excavators	NaN	EROPS w AC	NaN
4	11000.0	2007	7/23/2009 0:00	Medium	NaN	SSL	Skid Steer Loaders	NaN	EROPS	None or Unspecified

Figure 5.1: Example of columnar data. Each row is a separate auction sale for a bulldozer, and each column is a feature of the bulldozer. Data comes from Ref. [143].

5.2 Neural networks

5.2.1 Functional approximation

Considering Sec. 5.1.1 in a more mathematical sense: Given that one has a datapoint x which occupies some *observational space*, one may wish to apply a transformation to this datapoint in order to map it into a desired *target space* y . Examples of such cases include: an image classifier using the pixel values in a picture of animal to determine what breed of animal is pictured; a sentiment analyser parsing the text of a film review to determine whether the reviewer enjoyed the film or not; an autonomous agent observing its environment and deciding on its next action to better accomplish some desired goal; and an event-level classifier using the 4-momenta of a particle collision to determine which process gave rise to the collision. This mapping function can be represented as $y = f(x)$.

It is possible that domain theory offers an exact, analytical form for the mapping function $f(x)$, however if this is not the case, or if the full analytical form is computationally intensive to compute, then one may instead approximate the mapping using a

sufficiently flexible parameterisation of f : $y \approx \hat{y} = f_\theta(x)$, in which θ are the parameters of the functional approximation of f .

Adopting such an approximation, the challenge is then to adjust the parameters θ such that the values of $\hat{y}$ are as close as possible to y for all possible values of x , i.e. $\theta = \arg \min_\theta \mathcal{L}(y, \hat{y}_\theta)$. Here $\mathcal{L}(y, \hat{y}_\theta)$, the *loss function*, is a function which quantifies the difference between the true values and the current approximated values, i.e it provides a numerical representation of the current performance of the approximated form of f . The exact form of the loss function is problem dependent.

The introduction of the loss function now allows the adjustment of θ to be termed as an optimisation problem. Whilst many algorithms, beyond grid search or random searches, exist to attack these kinds of problems, such as simulated annealing [144], genetic algorithms (see e.g. Ref. [145]), and Bayesian optimisation [146, 147], the requirement that f_θ is sufficiently flexible means that the parameter space is extremely high-dimensional. This prevents such advanced algorithms from converging within a reasonable time. Neural networks, however, make use of several constraints in their construction to allow their parameters to be optimised in a time-efficient manner.

5.2.2 Neural networks

Whilst the history of artificial neural-networks (ANN/NN) extends back to 1957 [148], it was only in 2010 that they really started to become competitive in comparison to other possible machine learning (ML) tools, such as boosted decision trees and support-vector machines, particularly in the realms of image [149] and speech recognition [150]. Their recent renaissance being due to several changes in their design which allowed a greater number of free parameters to be optimised.

Following the principle of functional approximation, a NN attempts to learn a mathematical function which maps a selection of *input features* (the coordinates of the data in observational space) to the desired target space. This is accomplished by a series of matrix operations involving learned weights, and non-linear transformations, on the inputs. It can be visualised as layers of *neurons*, each of which receives inputs from all neurons in the previous layer. Such an illustration is shown in Fig. 5.2. Layers between the input and output layers are referred to as *hidden*, and when the number of hidden layers (the network *depth*) goes beyond one, the network is normally referred to as *deep* (DNN). The number of neurons in a given layer is referred to as its *width*.

Linear layers

Each layer of neurons applies a linear transformation to the incoming vector from the previous layer, or the inputs. This transformation involves a multiplication with a matrix, the values of which are referred to as the *weights*, and the addition of a vector of *biases*: $x_{i+1} = w_i \cdot x_i + b_i$, shown more explicitly in Eq. 5.4.

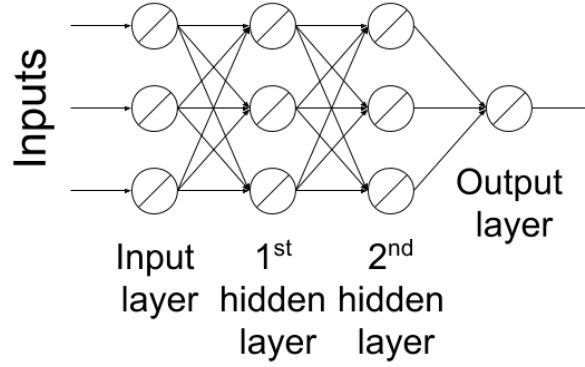


Figure 5.2: Conceptual illustration of a deep artificial neural network.

$$\begin{pmatrix} w_{i,0,0} & w_{i,0,1} & \dots & w_{i,0,N} \\ w_{i,1,0} & & & \\ \dots & & & \\ w_{i,M,0} & & & w_{i,M,N} \end{pmatrix} \times \begin{pmatrix} x_{i,0} \\ x_{i,1} \\ \dots \\ x_{i,N} \end{pmatrix} + \begin{pmatrix} b_{i,0} \\ b_{i,1} \\ \dots \\ b_{i,M} \end{pmatrix} = \begin{pmatrix} x_{i+1,0} \\ x_{i+1,1} \\ \dots \\ x_{i+1,M} \end{pmatrix} \quad (5.4)$$

These layers are referred to interchangeably as *linear layers* and *dense layers*. It should also be mentioned that some applications use more complicated layers, such as: convolution filters [151], which share parameters to apply the same transformations for multiple regions of the input space and are particularly useful to image data; and recurrent layers, e.g. [152, 153, 154], which again use parameter sharing to help deal with serial data. These kinds of layers are not used in this thesis, and so will not be discussed further.

Linear transformations alone, however are insufficient to allow the neural network to approximate functions reasonably well, hence the need for non-linear transformations between the linear layers.

Activation functions

The non-linear transformations placed between linear layers are referred to as *activation functions*. These generally apply the same function to each element of the incoming vector from the previous layer: $x_{i+1,j} = \text{Act}(x_{i,j})$. Whilst the activation functions are normally quite simple, through combining many subsequent applications of them, highly complex functions can be approximated. Additionally, activation functions are constructed to be continuously differentiable.

5.2.3 Optimisation through backpropagation

The choice to use continuously differentiable activation functions now means that the output of a neural network is also fully differentiable with respect to each weight and bias parameter (referred to henceforth as the parameters of the network). By further requiring that the loss function is a continuously differentiable function of the network's output, the influence that each parameter has on the network's overall performance can be analytically evaluated, i.e. we can compute $\nabla_{\theta} \mathcal{L}$, via a process called *backpropagation*.

Computing $\nabla_{\theta} \mathcal{L}$ becomes a two-step process. First, data are passed through the network in order to compute the corresponding values of $\hat{y}$. The mean value of the loss function is then computed based on the network's predictions and the true values y . Next, the gradient of the loss is passed backwards through the network via chain-rule differentiation.

Now, having evaluated the gradient of the loss with respect to all network parameters, *gradient descent* [155] may be used to update each parameter by a small amount in the direction of steepest descent of the loss surface:

$$\theta_{t+1} = \theta_t - \gamma \nabla_{\theta_t} \mathcal{L}(y, \hat{y}_{\theta_t}), \quad (5.5)$$

where, θ_t are the network parameters at iteration t and γ is a small coefficient, referred to as the *learning rate*, which controls the size of the update step taken at each iteration. Backpropagation was first applied to NNs in Ref. [156] (1981), and was independently discovered and popularised by Ref. [157] in 1986, however an earlier form had already been used in Ref. [158] (1970) although not explicitly on neural networks; see Ref. [159] for a full chronology of the history of deep learning, and in particular Sec 5.5 for the development of backpropagation.

It can be expected that the performance of the network can vary according to the input region from which a datapoint is drawn, hence why the loss is averaged over multiple data-points. Conversely, there is the potential for the network to 'memorise' the training data over multiple update iterations, potentially impacting its ability to generalise to new datasets. It is therefore common to sample the full training dataset without replacement into sets of data-points called *mini batches*, the size of which is called the *batch size*. This then provides the practitioner a trade-off between accurate evaluation of the loss, and generalisation of the trained model. Training with mini-batches is referred to as *stochastic gradient descent* (SGD), and the use of all the mini-batches is *one epoch of training*. Invariably, training a network requires multiple epochs, and the resampling mini-batches is re-performed at the start of each epoch.

5.2.4 Components of neural networks

Activation function

The choice of output activation function is often determined by the specific problem. The sigmoid function,

$$\text{Act}(x) = \sigma(x) = \frac{1}{1 + e^{-x}}, \quad (5.6)$$

is used for binary or multi-label classification due to its smooth asymptotic convergence to zero and one, as shown in Fig. 5.3. For regression tasks, the linear function ($y = x$) is a common choice since the targets are expected to be in the range $[-\infty, +\infty]$, although the sigmoid can also be used by rescaling the network output manually to occupy the target region.

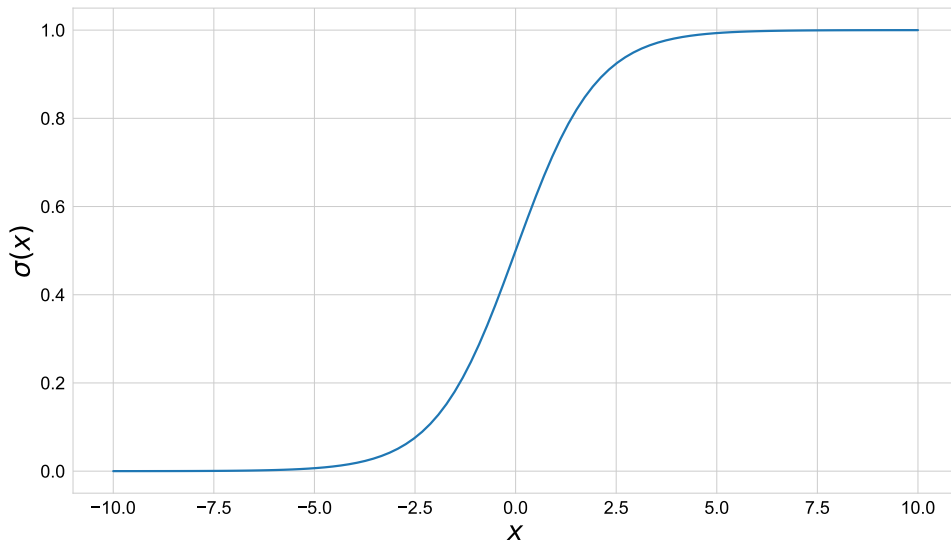


Figure 5.3: Output of the sigmoid activation function for a range of inputs, x . Note that the range of outputs is constrained to the range $(0, 1)$, with asymptotic convergence to each boundary and a smooth interpolation in between.

The choice of internal activation function, however is an area of on going development in the field of deep learning. The sigmoid and hyperbolic tangent ($\tanh$) functions were used for many years, however they possessed several problems which limited the size of networks that could be trained, the most important of which is the *vanishing gradient problem*. This refers to when the scale of the input to the function is a very large, which places the output in the saturation region (asymptotic zero, one, or minus one). The gradient of the function in these regions is extremely small and we can see from Eq. 5.5 that this then prevents the network parameters from updating.

As the depth of networks increased, so did the likelihood that the internal states would stray into these saturation regions. One attempt to ameliorate this was to pre-train layers and then sandwich them together as in e.g. Ref. [160] (1991) and Ref. [161] (2006), however nowadays the problem can be reduced with a suitable *weight initialisation*, and by using different activation functions.

A common default choice for the internal activation-function in contemporary NNs is now the rectified linear-unit (ReLU) [162]. This consists of a unit-gradient linear function for positive activations, and a flat line at zero for negative activation, as shown in Fig. 5.4, or mathematically $\text{Act}(x) = \max[0, x]$. This reduces the impact of the vanishing gradient problem by only allowing the gradient to saturate in half of the input space. Additionally, it is a much quicker function to compute, allowing networks to be trained in a shorter period of time.

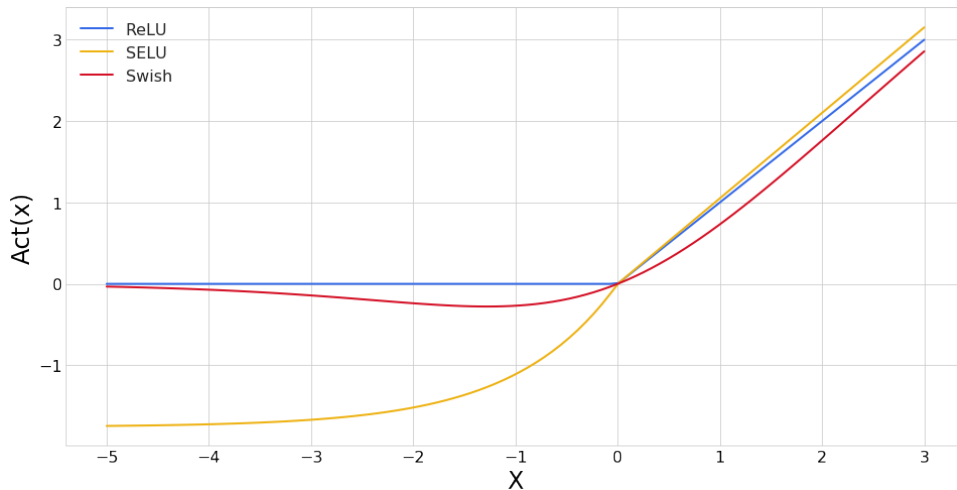


Figure 5.4: Responses ($\text{Act}(x)$) for several activation functions for a range of input values, x . PReLU (not shown) is the same as ReLU in the positive region, $\text{Act}(x) = x$, and with a constant, non-zero gradient in the negative region, $\text{Act}(x) = \alpha x$, where α is a parameter of the model.

The ReLU, however, is affected by two problems: dead ReLUs, where the output is constantly zero due to its input always being negative; and non-zero-centred outputs, which leads to inefficient learning of the correct weights. The exponential linear unit (ELU) [163] attempts to fix these problems by having a negative output for negative activations, which pushes the mean activation closer to zero, potentially allowing quicker convergence.

Scaled exponential linear units (SELU) [164] (used in chapters 7 and 8 and illustrated in Fig. 5.4) improve on ELU by using mathematically derived scaling coefficients to allow

the NN to *self-normalise*. This means that the signals in the network are kept close unit-Gaussian distributions, allowing the weight initialisations of the network to act optimally (see next Sec. 5.2.4). Previously this normalisation required a computationally-costly process called Batch Normalisation [165], which applied a transformation to batches of data to rescale it.

Chapters 9 and 10 use a more recent activation function called the Swish function [166] (illustrated in Fig. 5.4), which was found via a reinforcement-learning based search for optimal activation functions. It features the interesting characteristic of having a region of negative gradient, allowing outputs to decrease as the input increases: $\text{Swish}(x) = x \cdot \text{sigmoid}(\beta x)$, where β can either be kept constant or be learnt during training.

Weight initialisation

The weights in the network must start from some initial values; however, since uniform initialisation would lead to each neuron acting the same way, with no possibility of learning, the weights are instead sampled randomly from a distribution. This distribution, however must be carefully specified, in order to prevent the levels of activation within the network from being too low and or too high; ideally, unit Gaussian inputs should remain unit Gaussian throughout the network, and not explode to large values or shrink to negligible ones. This was one of the limitations on the depths of trainable networks.

The exact specification for the weight distributions is a function of both the widths of the layers, and the activation functions used. Generally the distributions used are either normal or a uniform limited between some values. In my experience so far, I have found that the choice of Normal or Uniform does not have a great influence on the performance of the final model, and so to reduce the hyper-parameter search space when designing a model I will only consider the Normal distribution in this thesis.

The **Glorot** [167] initialisation was derived for use with linear activation functions, and is also applicable to sigmoid and *tanh* functions. Here the normal distribution is defined as: $\mathcal{N}(0, \sigma)$, where σ is computed per neuron as:

$$\sqrt{\frac{2}{n_{\text{in}} + n_{\text{out}}}}, \quad (5.7)$$

where n_{in} and n_{out} are the number of inputs and outputs from the neuron, respectively.

For the ReLU function, the version modified by He [168] is required to account for the fact that half of the activation function is zero:

$$\sqrt{\frac{2}{n_{\text{in}}}}. \quad (5.8)$$

Note that for layers of the same width ($n_{\text{in}} = n_{\text{out}}$), this is equivalent to the **Glorot** initialisation but only considering half the number of inputs or outputs. The Swish function also uses this initialisation.

The function SELU uses another variation called the **LeCun** initialisation scheme [169]:

$$\sqrt{\frac{1}{n_{\text{in}}}}. \quad (5.9)$$

Bias initialisation

Bias initialisation is simpler than weight initialisation, as one can safely set all biases to the same starting value, and still get the network to respond differently. This is because the weights will not be the same, and adding an initially uniform number to this (see Eq. 5.4) will still result in a non-uniform set of values. The standard practice is initialise all biases to zero. The output bias for the network, however, can be potentially be more intelligently initialised, e.g. to the mean target value of the data in a regression problem.

Loss functions

As mentioned before, the loss function quantises the level of performance the NN currently appears to offer. Several common examples exist, however the choice is normally problem-dependent: for binary classification tasks, *binary cross-entropy* (BCE), is the common choice; and for regression tasks I use *mean squared-error* (MSE).

The BCE is calculated according to:

$$\text{BCE} = -\frac{1}{N} \sum_{n=1}^N [y_n \ln \hat{y}_n + (1 - y_n) \ln (1 - \hat{y}_n)], \quad (5.10)$$

and the MSE is calculated according to:

$$\text{MSE} = \frac{1}{N} \sum_{n=1}^N (\hat{y}_n - y_n)^2, \quad (5.11)$$

where N is the number of predictions. As before, $\hat{y}_n$ is the predicted class for data-point n , and y_n is the true class of data-point n . Note that in classification problems, y will be either 1 or 0, but $\hat{y}$ will be a real number between one and zero. Both of these loss functions quantify the difference between the NN's predictions and the true values, with lower values indicating better performance.

It is also possible to weight the data-points when computing the loss function by including a multiplicative factor per data-point:

$$\text{Loss} \rightarrow \text{Weighted loss} = \frac{1}{N} \sum_{n=1}^N w_n \epsilon(y_n, \hat{y}_n), \quad (5.12)$$

where w_n is the weight associated with data-point n , and ϵ is the error (non-reduced loss) on the prediction for data-point n . These weights can be used to adjust the relative penalty

for miss-predictions on a per data-point level. In high-energy physics, for example, such weighted losses can be used to account for class imbalances in the training data (e.g. signal and background, and particle process), provide information about the relative importance of different processes (e.g. a dominant background is much more important to identify than a process which only contributes a few events), and to flatten out data-occupancy of kinematic regions (e.g. to build a model invariant to (p_T, η) distributions).

Optimisers

Equation 5.5 described the standard update rule for SGD, however in recent years more advanced optimisers have been developed to allow networks to converge quicker (see e.g. [170] for an overview).

Due to the high-dimensional parameter-space that NNs have, it is likely that the update rule can become “trapped” in narrow, gently sloping valleys of the loss-surface, as illustrated in Fig. 5.5. In these cases, the walls of the valley have a high-gradient, forcing the update rule to move almost perpendicular to them, but the narrowness of the valley causes the update to move onto the opposite wall; effectively the NN oscillates between two sets of parameters without ever improving. Really, the update should either: use a lower learning rate, to avoid jumping over the valley floor; or move parallel with the valley walls down the gentle slope.

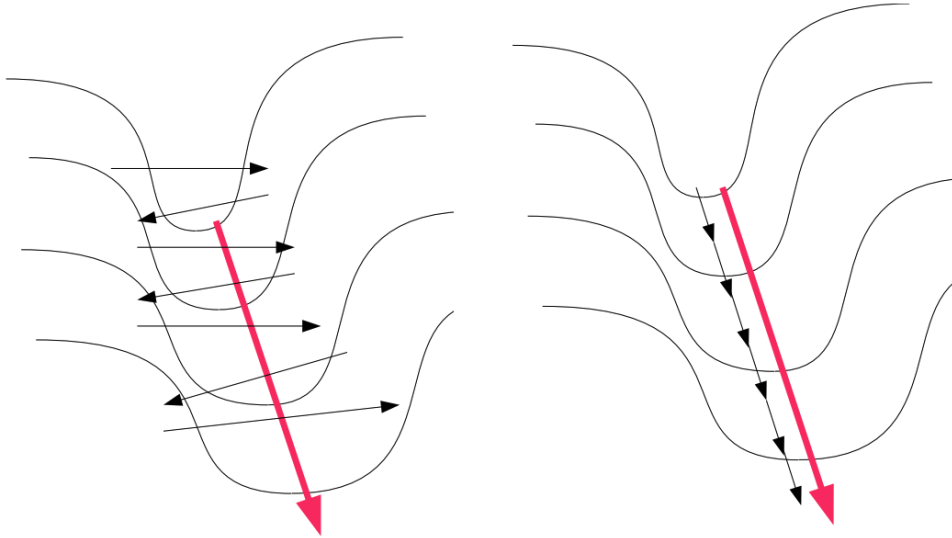


Figure 5.5: Parameter update moves in a “narrow loss valley”: Left, the updates in the direction of steepest gradient; Right, the ideal update moves. Image source [171].

SGD with momentum One way to help prevent this oscillating state is to allow the update rule to accumulate momentum from previous updates. In this case the slight movements parallel to the valley walls will eventually increase due to the accumulated momentum, whereas the movements perpendicular to the valley walls will be suppressed, since the momentum accumulated in that direction is cancelled out with each oscillation. This idea was successfully applied to NNs in 1986 by Rumelhart, Hinton, and Williams [172].

For SGD with momentum, the update rule becomes:

$$m_{t+1} = \alpha m_t - \gamma \nabla_{\theta_t} \mathcal{L}, \quad (5.13)$$

$$\theta_{t+1} = \theta_t + m_{t+1}, \quad (5.14)$$

where α is the momentum coefficient between zero and one (but generally close to one), and $m_0 = 0$.

Nesterov momentum A further improvement to the basic momentum implementation is to consider when it gets applied. In Eq. 5.14 we can see that regardless of the loss gradient at θ_t , the m_{t+1} update will always include αm_t . In *Nesterov momentum* [173], what is done instead is to make the parameter update in two stages: first the αm_t move, then the loss-gradient update, except that this may now be computed at $\theta_t + \alpha m_t$ instead of θ_t :

$$m_{t+1} = \alpha m_t - \gamma \nabla_{\theta_t + \alpha m_t} \mathcal{L}, \quad (5.15)$$

$$\theta_{t+1} = \theta_t + m_{t+1}. \quad (5.16)$$

This effectively provides a one-step lookahead to the gradient update, and can prevent it from overshooting changes in the gradient.

Adaptive learning-rates A second way to deal with narrow valleys is to allow the learning rate to adapt to the current gradient. For steeper gradients the update should be smaller to reduce the chance of overshooting the valley floor, whereas for shallower gradients, the update should be larger to allow for quicker training. The solution, then is to allow each parameter to track its own learning rate based on past gradients.

In ADAGRAD [174], the learning rate for parameter i at iteration $t + 1$ is:

$$\gamma_{i,t+1} = \frac{\gamma_0}{\sqrt{\sum_{\tau=0}^t \nabla_{\theta_{i,\tau}} \mathcal{L}^2}}, \quad (5.17)$$

where γ_0 is the initial, global learning-rate. This scaling allows the parameters associated with large gradients to update with lower learning rates and vice versa. However it does mean that all learning rates will eventually tend to zero, due to the accumulating denominator.

RMSPROP [175] gets around this decreasing learning rate by instead tracking a moving average of past squared-gradients:

$$v_{i,t+1} = \lambda v_{i,t} + (1 - \lambda) \nabla_{\theta_{i,t}} \mathcal{L}^2, \quad (5.18)$$

$$\gamma_{i,t+1} = \frac{\gamma_0}{\sqrt{v_{i,t+1}}}, \quad (5.19)$$

where $v_{i,0}$ is zero and λ is a coefficient between zero and one, and generally close to one.

Adaptive momentum Both methods of accumulating momentum and adapting the learning rate were seen to offer improvements in NN convergence. ADAM [176] combines these approaches by tracking moving averages of both the past gradients (momentum) and the past squared-gradients (adaptive learning rate):

$$m_{i,t+1} = \beta_1 m_{i,t} + (1 - \beta_1) \nabla_{\theta_{i,t}} \mathcal{L}, \quad (5.20)$$

$$v_{i,t+1} = \beta_2 v_{i,t} + (1 - \beta_2) \nabla_{\theta_{i,t}} \mathcal{L}^2, \quad (5.21)$$

where β_1 is the momentum coefficient (equivalent to α in Eq. 5.14), and β_2 is the adaptive coefficient (equivalent to λ in Eq. 5.18). Since both $m_{i,0}$ and $v_{i,0}$ are initialised to zero this would later introduce a bias in the adaptive scaling. ADAM corrects for this by dividing $m_{i,t}$ and $v_{i,t}$ by their β coefficients:

$$\hat{m}_{i,t+1} = m_{i,t+1} / (1 - \beta_1), \quad (5.22)$$

$$\hat{v}_{i,t+1} = v_{i,t+1} / (1 - \beta_2). \quad (5.23)$$

The learning rate now becomes:

$$\gamma_{i,t+1} = \frac{\gamma_0 \hat{m}_{i,t+1}}{\sqrt{\hat{v}_{i,t+1} + \epsilon}}, \quad (5.24)$$

where ϵ is a small number to prevent division by zero.

NADAM [177] is a adapted version of ADAM which includes Nesterov momentum.

Regularisation

As hinted at in Sec. 5.2.3, it is possible for the NN to memorise the training data, which impacts its ability to generalise to new datasets. Techniques and properties of the data and model which act to reduce this *over fitting*, are referred to as *regularisation*. Regularisation can come from many sources, not just intentional, tuneable ones, but also from data or model complexity. Reference [178] lists many such sources, however I will describe here a few of the most pertinent ones. In general, regularisation is something which must be tuned to both the data and the model; as too much can harm model performance on both training and unseen data, whilst too little can cause the model to perform well on training data and poorly on unseen data.

L_2 regularisation and weight decay L_2 [179] regularisation adds a penalty term to the loss, which penalises the NN based on the current magnitudes of its parameters:

$$\mathcal{L}_{\theta_t} \leftarrow \mathcal{L}_{\theta_t} + 0.5L_2 \sum_i \theta_{i,t}^2, \quad (5.25)$$

where L_2 is a coefficient that is normally small compared to $\mathcal{L}_{\theta_t}$. In this case, the model must attempt to minimise the loss without moving its parameters far from zero. Since this forces it to only make changes to the weights which consistently bring improvements to the loss, this can prevent the model from fitting to *noise* in the data, which are unrepresentative of genuine trends in the data.

A similar type of regularisation, called *weight decay*, instead penalises the updates of the parameters:

$$\theta_{t+1} = \theta_t - \gamma \left(\nabla_{\theta_t} \mathcal{L} + w_d \sum_i \theta_{i,t} \right), \quad (5.26)$$

where w_d is again a coefficient that is normally small compared to $\mathcal{L}_{\theta_t}$. Since the derivative of the sum of squares of parameters with respect to the parameters is equal to two times the sum of the parameters, the weight decay and L_2 approaches are equal for SGD updates. Most libraries therefore implement L_2 regularisation by adding a fraction of the sum of weights to the loss gradient, and refer to the two approaches interchangeably, e.g. L_2 in KERAS [180] and weight decay in PYTORCH [181].

For more advanced optimisers, however, the two approaches are not strictly the same since the L_2 penalisation gets combined into the moving averages tracked by the optimisers. Reference [182] explores the differences between the two approaches and concludes that a true implementation of weight decay for advanced optimisers like ADAM provides the greater benefit (see also Ref. [183] for further discussion and experimental results). The NNs trained in this thesis, however, use neither L_2 regularisation nor weight decay (although it is sometimes tested for some models). Because of this I use the library-native implementations of the optimisers, rather than versions which correctly implement weight decay.

Dropout Popularised¹ in 2012, Dropout [186, 187], is a very common and effective technique for regularising NNs. During training, neurons within given layers are randomly “dropped” with a given probability. That is to say that their outputs are ignored by other neurons in the network.

In Dropout, the probability for a neuron to be dropped is normally set per layer, and the choice is reevaluated after each iteration, e.g. neuron a may be dropped in one iteration

¹Reference [184] shows that Dropout is actually a special case of the more general *stochastic delta rule* [185] (1990), in which network weights are sampled as random variables from Gaussian distributions on each forward pass of the network.

but be present in the next iteration. Since the network is now unable to rely on a given series of input transformations to provide good predictions, it must instead learn multiple processing routes, effectively this is internal ensembling (see Sec. 5.4). At application time, all neurons are available, and so the output of the network can be thought of as the averaged prediction of many smaller networks. This does, however, lead to one subtlety in implementation that the outputs of the neurons must either be scaled up during training, or scaled down during application in order to have consistent levels of output during both phases. Another key point is that by effectively having to train multiple networks, the convergence time for the full network may be longer, however the time per epoch may be shorter due to there being fewer calculations to perform (depending on how Dropout is implemented).

Early stopping and model checkpointing One of the most effective ways to regularise a network is to continually evaluate its loss on a second dataset (referred to occasionally as *validation data*), which is separate to the training data. Whilst the training loss will normally continue to decrease, or reach a plateau, the network is actually performing best when the validation loss is lowest. The validation loss is normally seen to either reach a plateau as well, or to begin increase, indicating the network is over fitting to the training data and can no longer generalise well to unseen data. In these cases the network can be referred to as *over trained*. An example of such a loss curve is shown in Fig. 5.6.

The process of *early stopping* refers to monitoring the validation loss, and ending training once it has not decreased (by a minimal amount) for a prescribed number of epochs. This can be used to reduce training times when one does not know *a priori* how long the network will take to converge. Additionally, by saving a copy of the network whenever the validation loss improves, the best performing set of parameters can be loaded after training.

5.3 Random Forests

5.3.1 Decision-tree learning

A decision tree is a class of machine learning algorithm which can be applied to both classification and regression tasks. During the training phase a series of binary splits are applied to the starting dataset based on features of the data, which can be different at each split, as illustrated in Fig. 5.7. This process of splitting the data continues until some stopping criteria are met. The series of decisions can now be applied to new data. There are several implementations for decision tree methods, however I only use CART [188] (Classification And Regression Trees) in this thesis, which I will describe below.

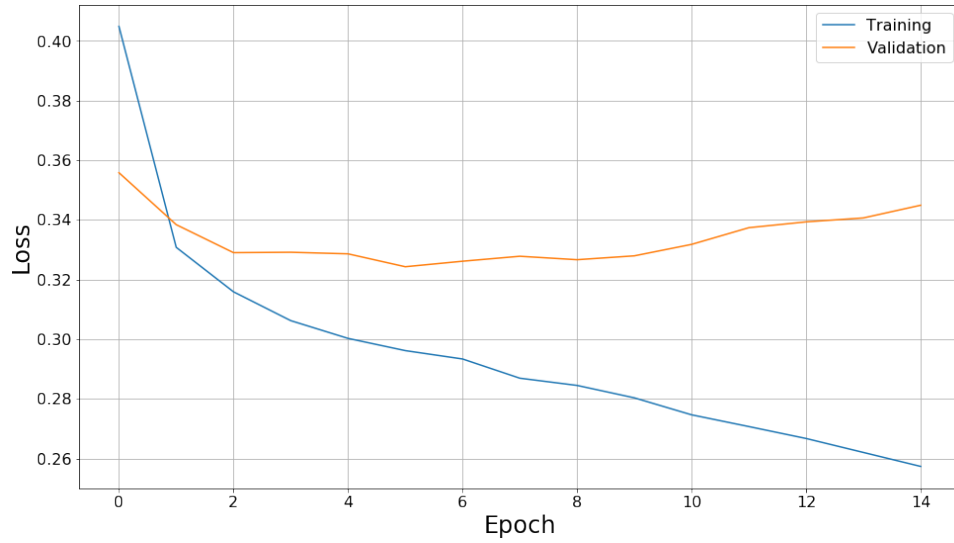


Figure 5.6: An example of overtraining. The training loss is the loss computed on the training data, and the validation loss is the loss computed on the validation dataset, which is not used to update the network parameters and so represents an unbiased estimate of the performance of the model. The validation loss decreases as training begins (epoch zero), reaches a minimum (epochs five to eight), and then begins to increase (epoch eight onwards). The training loss, however, continues to decrease. The best performing state of the neural network is when its validation loss is lowest, i.e. around epoch five. Continuing to train further does not improve the general performance of the model, only its performance on the specific set of data that is used during training.

General procedure

Beginning from the full set of training data (the *root node*), all possible cuts for all available features are examined by evaluating the value of an objective function. The split which results in the most optimal value for the objective function is applied resulting in two *child nodes*, one for each set of the data which falls on either side of the split. This process then repeats recursively on each of the child nodes, and continues until some stopping criterion is met. The resulting terminal nodes are referred to as *leaf nodes*.

Stopping criteria There are several ways to control the number of splits (the *depth* of the tree), and to stop the creation of new child nodes. These are hyper parameters which must be tuned to the given dataset and task, and can be used to control the amount of regularisation present in the model to reduce overfitting. Some common criteria are:

- Maximum depth: the maximum number of levels of splitting the tree can contain,

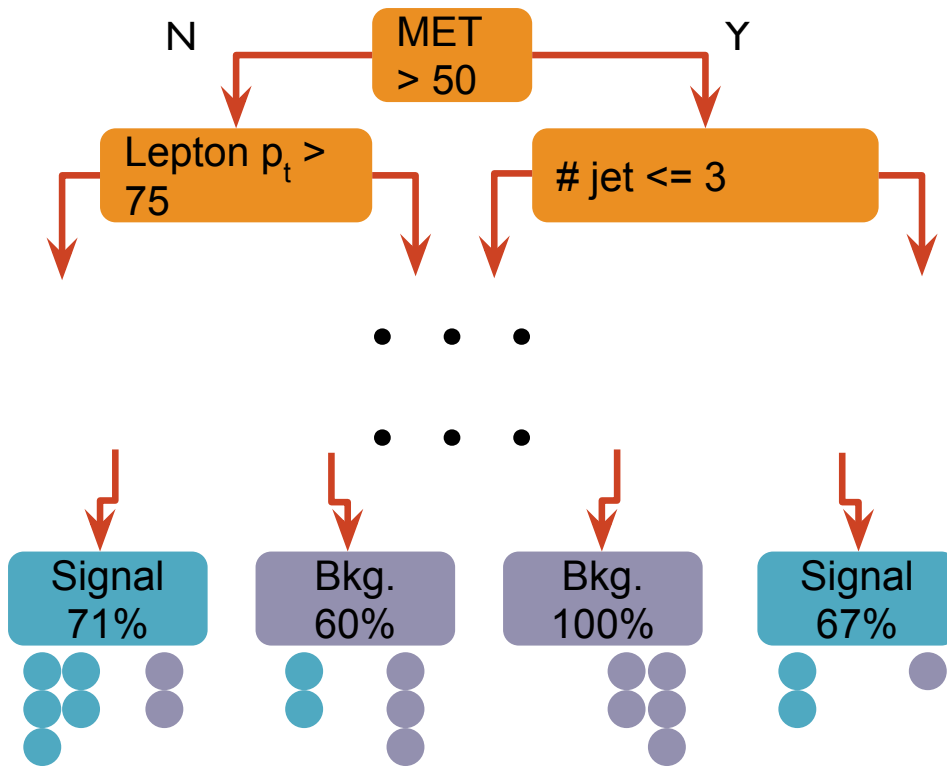


Figure 5.7: Illustrative layout of a decision tree. Data are split by sequential thresholds of individual features until some stopping criterion is met, resulting a set of leaf nodes with fixed predictions.

- Minimum number of samples per split: used to only allow splits to be performed on nodes which contain at least a certain number of datapoints,
- Minimum number of samples per node: used to limit the available split points by requiring that the resulting split leaves a minimum number of datapoints in each child node.

Classification trees

In classification tasks, the target feature is categorical, so the objective function being optimised should reflect the probability of correctly identifying the inputted datapoints as belonging to their respective classes.

Gini impurity In CART, the Gini impurity is the metric to be minimised in classification problems. It can be described as the sum of the probabilities of misclassifying

datapoints which are randomly sampled if the datapoints were labelled randomly according to the class distributions within a node. For example, given a node which contains two datapoints of class a , three datapoints of class b , and five datapoints of class c . The probability to misclassify datapoints of class a is: the probability to pick a datapoint of class a ($p_a = 0.2$), times the probability to mislabel it ($1 - p_a = 0.8$), resulting in $p_a \times (1 - p_a) = 0.2 \times 0.8 = 0.16$. Summing over these class misclassification probabilities, results in the mathematical expression for the Gini impurity:

$$I_G = \sum_{c=1}^C p_c \times (1 - p_c) = 1 - \sum_{c=1}^C p_c^2, \quad (5.27)$$

where $c \in [1, C]$ are the class indices. Nodes which only contain one class have a Gini impurity of zero, e.g. the “100 % Bkg.” node in Fig. 5.7.

Similar to weighted training of neural networks (Sec. 5.2.4), trees can also be trained using weighted data by summing up the total weights of points of each class, rather than the simply counting the number of datapoints of each class, during training and prediction.

Prediction Having trained a classification decision tree, it can be applied by feeding a new datapoint through the series of decisions until arriving at a leaf node. The prediction for the class of the datapoint is then the modal class of the training data in that node. E.g. given the node in the previous paragraph, the datapoint would be predicted as belonging to class c . Alternatively, it is sometimes useful to instead know the probabilities of belonging to each class, in which case the prediction would be $\{a : 0.2, b : 0.3, c : 0.5\}$.

Regression trees

In regression tasks we instead wish for the algorithm to predict the value of a continuous target feature. Similarly to regression tasks in neural networks (Sec. 5.2.4), we can use the mean-squared error of the prediction of the node. The prediction of the node in this case is the mean values of the target features of the training data in the node. Again, datapoints can be weighted, in which case the weight means are computed.

Similarly to applying a classification tree, when applying the regression tree to new data, the datapoint is fed through the tree until arriving at a leaf node. Since the mean values of the target features of the training data in the leaf node is now fixed, the regression tree does not have a continuous response, like a NN would, but rather attempts to model continuous functions with a series of discrete steps, as demonstrated in Fig. 5.8.

The usefulness of decision tree methods

Decision trees are useful in the sense that they work with both continuous and categorical data (although optimal treatment of categorical features may be useful), and are invariant

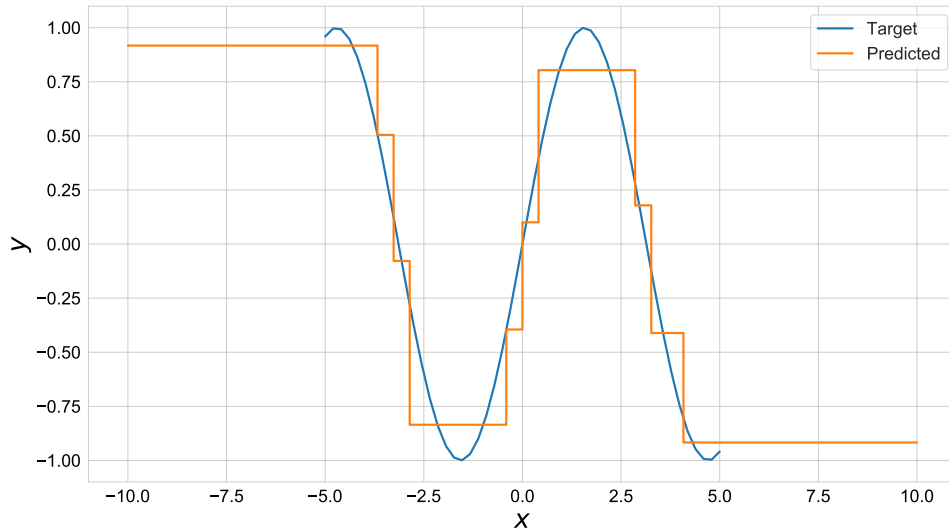


Figure 5.8: Example of the discrete nature of decision-tree-based regression; the target function is continuous, however a decision tree only has a finite number of leaf nodes, and hence has a discrete output, leading the approximation of the function as a series of steps.

to the scale and monotonic transformations of the features due to only considering the rank-order of datapoints. This means that they can be easily applied to a dataset with little to no preprocessing beforehand. However, since they rely on memorising the training data (their predictions are based on the exact values and class distributions seen during training), they are easily affected by outliers in the training data and can also overfit to the training data, even with the regularising effects of their stopping criteria.

5.3.2 Random Forests

Published in 2001, Random Forests [189], attempt to improve on CART by training many trees together as an *ensemble* (a concept further discussed in Sec. 5.4.4). The prediction of this set of trees is now either: the mean prediction of the trees, for regression models; or the modal class of the mean class probabilities, for classification models. By averaging the prediction over multiple models, the effect of overfitting can be further reduced, provided some randomness is present to prevent all the trees being constructed in the same way. The key point of Random Forests is the way they incorporate randomness.

The aim of a Random Forest is to make each tree as different as possible to all the other trees in the ensemble, whilst still allowing them to perform reasonably well in individually; effectively to make the ensemble be a collection of highly de-correlated *weak learners*.

Bagging The first way that randomness is introduced is via *bootstrap aggregation*, or *bagging* for short: each new tree is trained on a dataset that has been randomly sampled with replacement (i.e. the same sample can be drawn multiple times) from the original dataset. In this way, the effect of outliers is reduced, since they are not guaranteed to always be present in the training data.

Additionally, bagging provides a second set of datapoints which were not included in the training data. This *out-of-bag* (OOB) sample can then be used to provide an unbiased estimate of the performance of the tree. The OOB score can then be averaged across all the trees in the model. To provide a metric indicative of the model performance. Note, however, that it is not expected to be equal to the model performance, since it does not benefit from the improved performance that comes from combining many trees into an ensemble.

Feature sub-sampling The second way which Random Forests include randomness, is by limiting the features which are available when constructing splits: each time a tree is searching for an optimal split, rather than being able to consider all features, the set of features are instead randomly subsampled to produce a smaller set. If, for instance, there is a very strong feature in the data, then trees which are unable to split using this feature early on in the decision process are likely to display a very different structure to those which do see the feature early on. The number of features available is another hyper parameter for the model, which can be tuned: too smaller a set and the tree will not be able to pick good splitting points, limiting its performance; too larger a set, and the trees are likely to be more correlated, decreasing the generalisation performance of the Random Forest.

Use of Random Forests

Random Forests inherit the benefits of decision trees, and alleviate some of their problems. In addition to this they can be trained to reasonable performance in relatively small datasets. Whilst the main classification and regression tools in this thesis are built from neural networks, Random Forests are used Chap. 9 onwards to help quickly optimise the input features that are then used for training the NNs.

5.4 Assorted data-science techniques

This section includes discussions and explanations of various common data-science techniques which are relevant to the work in this thesis.

5.4.1 Pre-processing

The mathematically derived weight initialisation schemes for neural networks (see Sec. 5.2.4) are derived to be optimal for unit Gaussian inputs. However, it is likely that the input features in the data are not close to unit-Gaussian distributions. In order to make use of the weight initialisation schemes, the input features must be transformed such that they have a mean of zero and a standard deviation equal to one. Additionally, these transformations should be fixed using the training data, and then applied to both the training and testing data. This prevents information of the testing data from leaking into the training data, which could prevent the testing data from being considered as *unseen*, and therefore unable to provide an unbiased estimate of model performance.

5.4.2 Stratified splitting

Many times in data-science problems, one must split datasets whilst attempting to keep both resulting datasets as similar to the original as possible, in terms of datapoint composition. In a *stratified split* one assigns groups of datapoints unique IDs, referred to in this thesis as a their *stratification key*, and then splits such that the fractional composition of datapoints by stratification key in both datasets are equal to those of the original dataset.

As an example, given a typical HEP signal-search dataset with both signal and background events, one could split such that the resulting datasets each contain the same fraction of signal and background events as the full dataset. This can also be extended to consider categorical features (e.g. splitting such that the fraction of signal events with two jets matches in both datasets), and binned continuous features (e.g. splitting such that the fraction of signal events with two jets and a lepton $p_T \leq 50 \text{ GeV}$ matches in both datasets).

Since most functions in libraries such as SKLEARN [190] only accept a single stratification key, one technicality is generating this single ID such that it uniquely identifies the various factors considered. The approach taken in this thesis is to first assign an integer value to each of the factors, e.g. background and signal are 0 and 1 respectively, 1-jet, 2-jet, 3-jet events are 0, 1, 2, and lepton $p_T \leq 50 \text{ GeV}$ and $p_T > 50 \text{ GeV}$ are 0 and 1. These integer values can then be used as powers of prime numbers greater than one and the product computed (e.g. $2^1 \times 3^1 \times 5^0 = 6$). This allows each possible combination of factors to be assigned their own unique ID. I call this unique ID a *stratification key*.

5.4.3 Cross validation

k -fold cross-validation is a technique which can have multiple uses. It first involves splitting a dataset into k equally sized portions. A model can then be trained and tested k times from scratch, each time using a different portion for validation and the remaining $k - 1$ for training. By not having to split off a dedicated validation sample, this provides a

data efficient method of training, which can have many applications such as: showing the general response of a model, rather than how it happened to respond to a particular set of data; parameter optimisation; and training and evaluating many models for an ensemble (see next Sec. 5.4.4).

The k -fold splitting can also be performed in stratified manner, in order to keep the fractional composition of events consistent within each fold. This helps to ensure that each model receives a balanced training.

5.4.4 Ensembling

As previously introduced in terms of Random Forests in Sec. 5.3.2, ensembling is an approach in which multiple trained models are combined to offer a greater performance than each trained model on its own. Provided they give predictions in the same output space, the models don't even have to be of the same kind of algorithm, e.g. a Random Forest could be ensembled with a neural network.

The simplest method of ensembling is to take the mean prediction across all of the sub-models, however one can instead take the weighted mean by assigning each model a given weight. Often in this thesis, the responses of the multiple NNs are weighted according to their performance as demonstrated on a separate testing set (e.g. their validation fold during k -cross validation training). This allows the most performant NNs to have the greatest influence, but still be supported by other, weaker NNs.

Additionally, Sec. 9.4.8, explores a method in which an ensemble is not constructed by averaging in *model space*, but rather by tracking an average of NN parameters during training (averaging in *parameter space*), which should provide the benefits of ensembling without the increased run-time cost of having to train and evaluate multiple models.

Chapter 6

Motivation: The search for di-Higgs production

6.1 Motivation

6.1.1 Probing the Higgs potential - the κ_λ framework

Whilst the Higgs boson was discovered in 2012 by the ATLAS and CMS collaborations [14, 15, 16] at a mass of around 125 GeV, this alone is not enough to accurately determine the structure of the Higgs potential.

Expanding the Higgs potential produces [191, 192]:

$$\begin{aligned} V^{\text{SM}}(H^\dagger H) &= -\mu^2 H^\dagger H + \lambda (H^\dagger H)^2 \\ V^{\text{SM}}(h) &= \frac{1}{2}m_h^2 h^2 + \lambda_3^{\text{SM}} v h^3 + \frac{1}{4}\lambda_4^{\text{SM}} h^4 + \dots, \end{aligned} \quad (6.1)$$

where

$$\lambda_3^{\text{SM}} = \lambda_4^{\text{SM}} = \frac{m_h^2}{2v^2}, \quad (6.2)$$

and

$$v = \sqrt{\frac{\mu^2}{\lambda}}. \quad (6.3)$$

The theoretical values of the running-coupling λ can be computed to NNLO accuracy [193] (see Fig. 6.1), and it, along with the top Yukawa coupling, y_t , determine the stability of the SM vacuum. From Fig. 6.2, we see that current experimental measurements put the Universe in the region of meta-stability and it is therefore of interest to ensure the Higgs potential is well understood.

The lambda values can affect particle interactions in a number of ways, and their effects can be probed by both direct and indirect measurements, for example by the processes

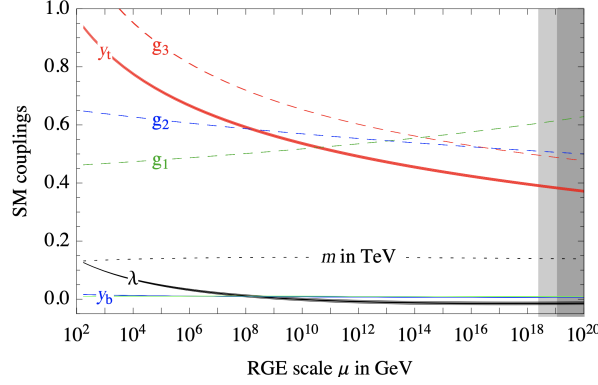


Figure 6.1: Theoretical values of standard-model couplings as a function of renormalisation group evolution scale. Image source [193].

shown in Fig. 6.3. The challenge is how to interpret the experimental measurements in order to constrain parameters. Direct measurements allow for a more *model-independent* interpretation, whereas indirect measurements will require more careful consideration.

Working in an effective field theory (EFT), the SM Higgs potential can be modified to allow for BSM scenarios [194]:

$$V^{\text{BSM}}(\Phi) = \frac{1}{2}m_h^2 h^2 + \lambda_3 v h^3 + \frac{1}{4}\lambda_4 h^4 + \sum_n \frac{c_{2n}}{\Lambda^{2n-4}} \left(\Phi^\dagger \Phi - \frac{v^2}{2} \right)^n, \quad (6.4)$$

where the lambda values are now modified with respect to their SM values:

$$\lambda_3 = \kappa_\lambda \lambda_3^{\text{SM}} \quad \lambda_4 = \kappa_{\lambda_4} \lambda_4^{\text{SM}}. \quad (6.5)$$

Considering just up to $n = 3$ in Eq. 6.4 (c_6 , six-dimension EFT), λ_3 and λ_4 are still related:

$$\begin{aligned} \kappa_\lambda &= 1 + \frac{c_6 v^2}{\lambda \Lambda^2} \\ \kappa_{\lambda_4} &= 1 + \frac{6c_6 v^2}{\lambda \Lambda^2} = 6\kappa_\lambda - 5. \end{aligned} \quad (6.6)$$

Moving further to $n = 4$ (c_8 , eight-dimension EFT) allows λ_3 and λ_4 to be modified independently.

Considering just the six-dimension EFT, deformations to the SM Higgs potential can be made by varying just a single parameter, κ_λ , where:

$$\kappa_\lambda = \frac{\lambda_{hhh}}{\lambda_{hhh, \text{SM}}}. \quad (6.7)$$

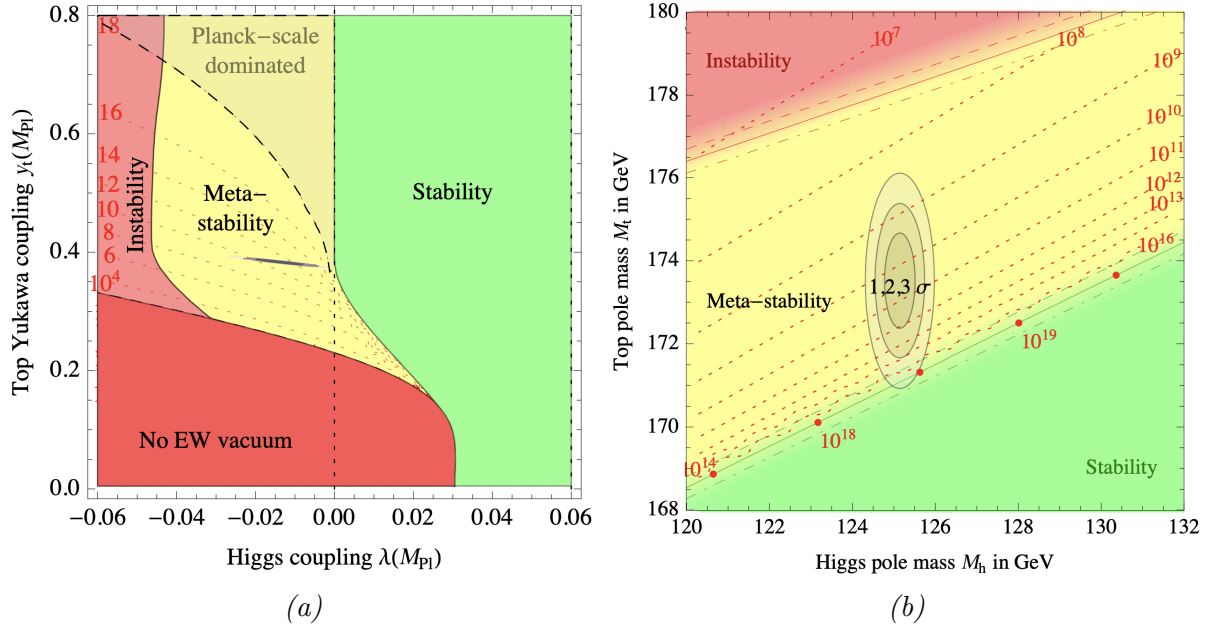


Figure 6.2: SM vacuum stability regions as a function of: (a) Higgs and top couplings (λ and y_t), (b) Higgs and top masses. Ellipses represent current experimental values. Image sources [193].

As shown in Fig. 6.4, this deformation by varying κ_λ directly leads to changes in the production cross-section of di-Higgs. Since the production cross-section is a property which can potentially be measured at particle colliders, this relationship allows experimental results to be reinterpreted as constraints on theory parameters.

6.1.2 The necessity of di-Higgs measurements

Single-Higgs constraints

As demonstrated in Fig. 6.3, the Higgs self-coupling can be probed using only single-Higgs measurements. Reference [196] computes constraints on κ_λ using the current experimental measurements of the Higgs-boson properties [197] (Fig. 6.5a). At the 95 % confidence level, κ_λ is constrained to be in the range $[-9.4, 17.0]$, using gluon-gluon fusion (ggF) and vector-boson fusion (VBF) and assuming that the true value of κ_λ is one.

The reference proceeds to make projections assuming improvements to the measurements of the Higgs boson properties as computed in Ref. [198] (Fig. 6.5b). At the HL-LHC (3000 fb^{-1}), the 95 % CL constraints on κ_λ are expected to be reduced to $[-2.0, 6.8]$.

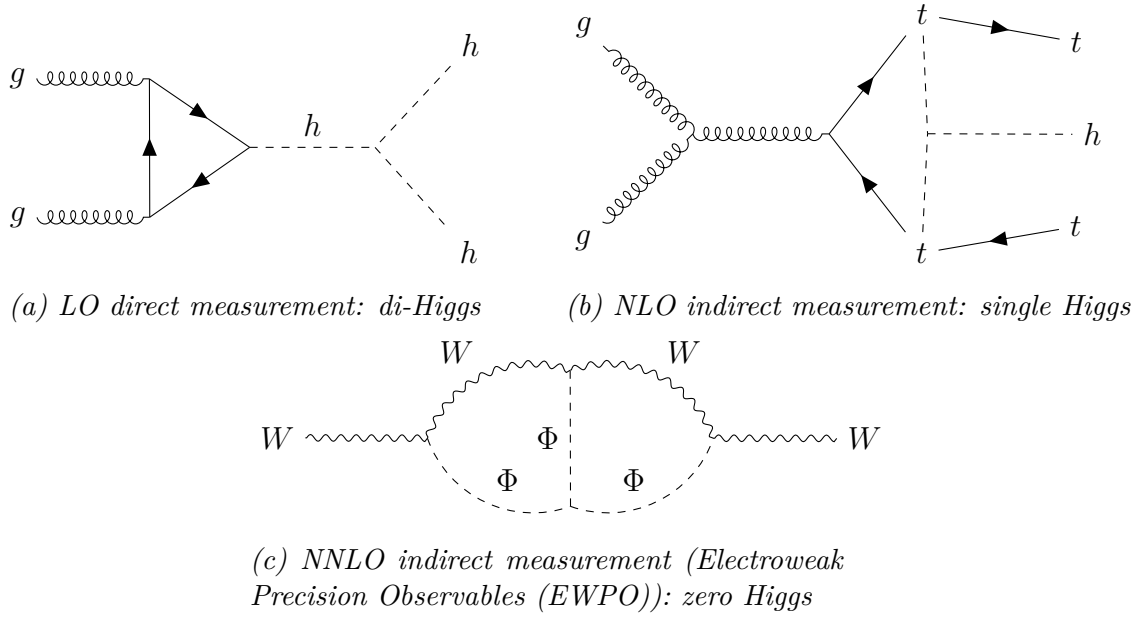


Figure 6.3: Examples of direct and indirect probes of λ_3 and λ_4 .

EWPO constraints

Figure 6.3 also shows that the Higgs self-coupling can also be probed indirectly using electroweak measurements. Reference [199] uses the mass of the W -boson as measured by the ATLAS collaboration [200], and the *electroweak-mixing parameter* as averaged from measurements by the CDF and D0 collaborations [201, 202]. The EWPO measurements can be used to improve the constraints on κ_λ coming from current ggF and VBF measurements (Fig. 6.5a) resulting in bounds of $[-8.2, 13.7]$ at 95 % CL.

Global fitting

The constraints derived in Sections 6.1.2 and 6.1.2 are done so under the assumption that only λ_{hhh} is varied and all other parameters remain the same - an *exclusive* fit. Reference [191], however, demonstrates that alterations to λ_{hhh} also leads to alterations of other Higgs interactions, including single-Higgs couplings. The constraints of Sections 6.1.2 and 6.1.2 are therefore optimistic, and instead the constraints must be derived using a global fit in which distortions to the other Higgs parameters are properly accounted for - an *inclusive* fit. Reference [191]’s results of such a fit are shown in Fig. 6.7, in which the global fit becomes almost flat, indicating very little sensitivity to λ_{hhh} and hence the form of the Higgs potential. Moreover, the prospect of a large κ_λ can potentially spoil previous fits to single-Higgs couplings and imply that they should also include κ_λ as a free-parameter.

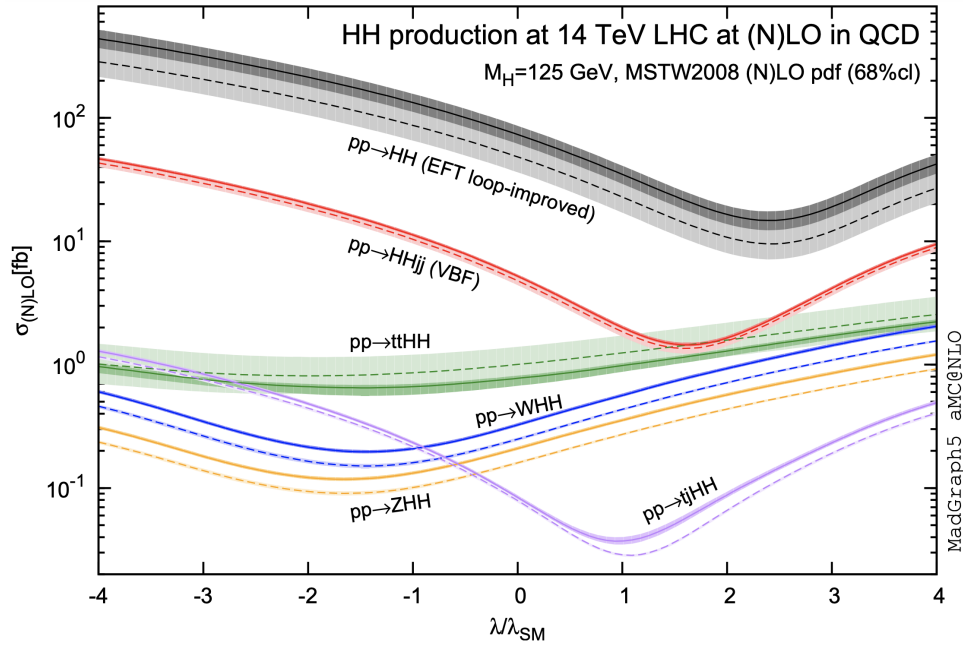


Figure 6.4: Production cross-section for di-Higgs in various production channels as a function of κ_λ . Image source [195]. $pp \rightarrow HH$ (EFT loop-improved) refers to production via gluon-gluon fusion with a virtual fermion loop, as discussed in Sec. 6.2.

Di-Higgs measurements

Including di-Higgs measurements can help lift the flat fit, restoring sensitivity to κ_λ in a global fit. Reference [191] goes on to do this, resulting in Fig. 6.8. In Fig. 6.8a, although sensitivity to κ_λ has been restored, the fit now results in a pair of degenerate minima, one centred at the expected value of $\delta\kappa_\lambda = 0$ ($\delta\kappa_\lambda = \kappa_\lambda - 1$) and another at $\delta\kappa_\lambda \approx 5$. Reference [203] shows that the invariant mass of the di-Higgs pair is particularly sensitive to deviations to the Higgs properties, and including the observables sensitive to the di-Higgs mass (a differential analysis) results in Fig. 6.8b, in which the second minima has been raised, with κ_λ expected to be constrained to the range $[-0.7, 7.5]$ with 95% confidence. The use of differential observables can also have an impact on the single-Higgs measurements, as explored in Fig. 6.9.

6.1.3 Resonant di-Higgs production

Whilst di-Higgs production is predicted to occur within the Standard Model, other BSM theories propose heavy particles which could decay to a pair of SM Higgs bosons. This is referred to as *resonant* di-Higgs production, since the invariant mass of the di-Higgs

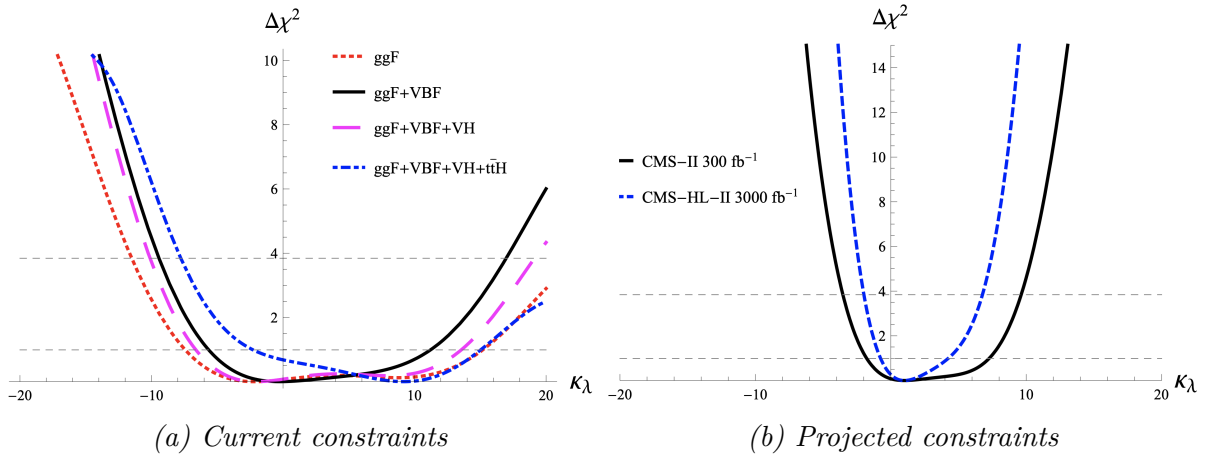


Figure 6.5: Constraints on κ_λ using single-Higgs measurements, assuming $\kappa_\lambda = 1$: (a) constraints using current measurements of Higgs-boson properties; (b) projected constraints using expected improvements to Higgs-boson measurements for two CMS scenarios. Image source [196].

now forms a peak at the mass of the heavy particle, as opposed to *non-resonant* di-Higgs production examined in Sec. 6.1.2 in which the di-Higgs invariant mass forms a wide distribution.

Two Higgs-Doublet Models (2HDM, see e.g. Ref. [204]) predict, amongst other new particles, the existence of a pair of neutral spin-0 particles, one of which could be the SM Higgs boson (h) and the other is a *heavy* Higgs boson (H). Examples of such 2HDMs include the *habemus* parameterisation of the minimal supersymmetric model (hMSSM) [205, 206]. The supersymmetric (SuSy) model is an appealing extension to the SM which positing the existence of a *supersymmetric* partner for every SM particle, the spins of which differ by one half. Such an addition *solves* several of the SM's *unnatural issues* by providing an explanation for the vast difference in strengths of different forces and allowing strong, weak, and electromagnetic couplings to unify at high energy. Additionally this provides a candidate particle for dark matter.

Theories of *warped extra dimensions* such as that of Randall and Sundrum [207] predict the existence of two heavy, neutral particles: one with spin-0 (a *Radion*) and one with spin-2 (a *Graviton*), both of which can decay to SM Higgs bosons. Warped extra dimension models seek to include the gravitational force in the SM, and indeed the Graviton would be the mediator boson for gravity.

The fact that the new particles proposed by these theories are expected to have high branching fractions to SM pairs of SM Higgs bosons, means that without great modification to the non-resonant analysis strategy, a search for heavy spin-0 and spin-2 resonances can be made. As we shall see in Sec. 9.9 discovery of SM di-Higgs production ($\kappa_\lambda = 1$)

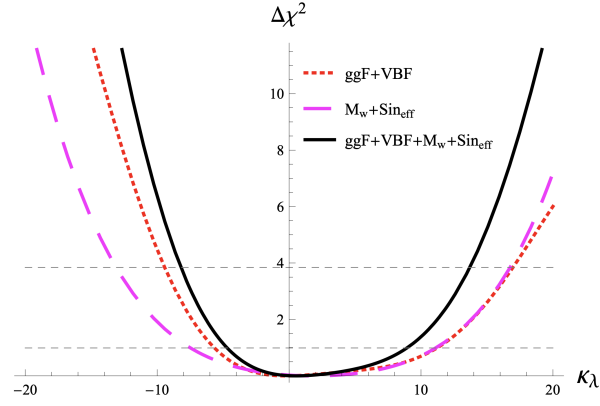


Figure 6.6: Constraints on κ_λ using EWPO measurements and current measurements of the Higgs-boson properties, assuming $\kappa_\lambda = 1$. Image source [199].

is not expected even at the HL-LHC, so the addition of a resonant search provides the chance of finding hints of New Physics, or further constraining such BSM theories, in the meantime.

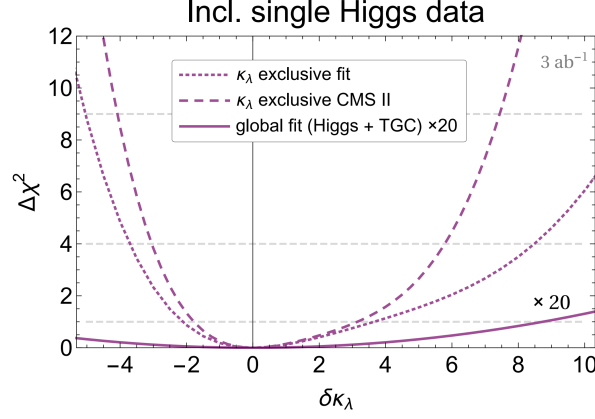


Figure 6.7: HL-LHC projected constraints on κ_λ using single-Higgs measurements. $\delta\kappa_\lambda$ is the correction to κ_λ , i.e. $\delta\kappa_\lambda = 0$ is the point at which $\kappa_\lambda = 1$. Image source [191].

- κ_λ exclusive fit: exclusive fit in which all other Higgs coupling deviations are set to zero.
- κ_λ exclusive CMS II: the exclusive fit results shown in Fig. 6.5b for comparison.
- Global fit: inclusive fit of single-Higgs and Higgs trilinear gauge couplings (TGCs), multiplied by 20 for visibility.

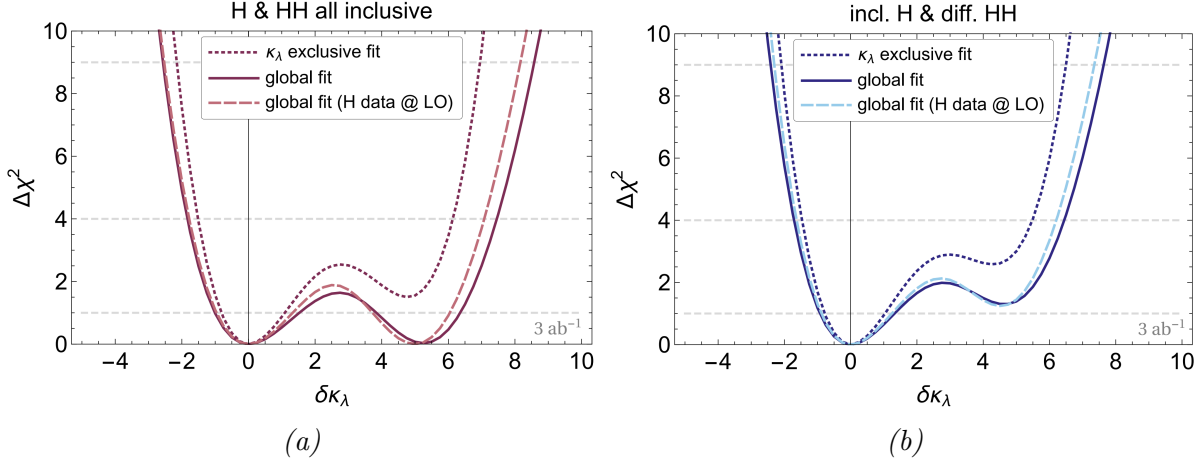


Figure 6.8: HL-LHC projected constraints on κ_λ using single-Higgs and di-Higgs measurements, assuming $\kappa_\lambda = 1$. In (b) a differential analysis is used which is sensitive to the invariant mass of the di-Higgs, which helps to raise the second minima. Image source [191].

- κ_λ exclusive fit: exclusive fit in which all other Higgs coupling deviations are set to zero.
- Global fit: inclusive fit of single-Higgs and Higgs trilinear gauge couplings
- Global fit (H data @ LO): inclusive fit but neglecting the impact of κ_λ on single-Higgs processes.

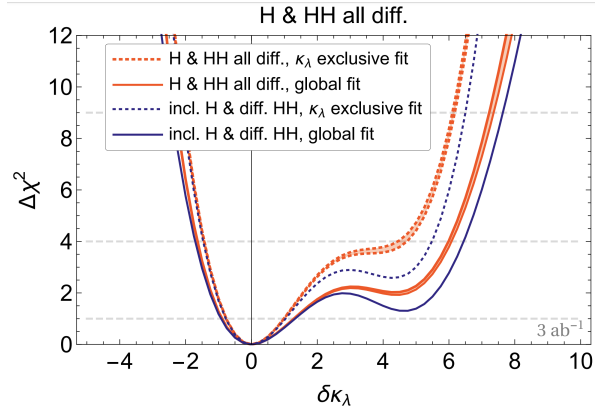


Figure 6.9: Impact of differential observables on the projected HL-LHC constraints on κ_λ using single-Higgs and di-Higgs measurements, assuming $\kappa_\lambda = 1$. Image source [191].

- *H & HH all diff*: differential observables were used for both single- and di-Higgs measurements.
- *incl. H & diff. HH*: differential observables were only used for di-Higgs measurements.
- κ_λ *exclusive fit*: exclusive fit in which all other Higgs coupling deviations are set to zero.
- *Global fit*: inclusive fit of single-Higgs and Higgs trilinear gauge couplings.

6.2 Production mechanisms

There are several groups of ways in which a di-Higgs pair can be produced using proton-proton collisions at the LHC in a non-resonant fashion. Figure 6.4 details the production cross-section of these channels as a function of κ_λ , complementary to this, Fig. 6.10 shows the production cross-section for $\kappa_\lambda = 1$ as a function of centre-of-mass energy.

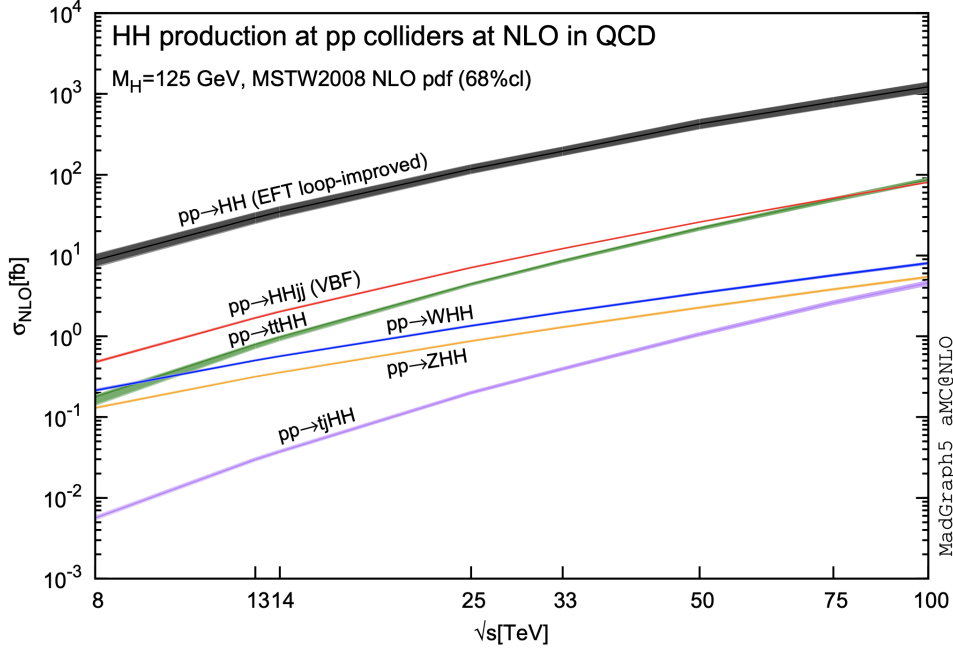


Figure 6.10: Production cross-section for di-Higgs in various production channels as a function of centre-of-mass energy for $\kappa_\lambda = 1$. Image source [195]. $pp \rightarrow HH$ (EFT loop-improved) refers to production via gluon-gluon fusion with a virtual fermion loop, as discussed in Sec. 6.2.

From these two plots we can see that consistently $pp \rightarrow HH$ (EFT loop-improved) has the highest production cross-section. This production method is referred to in this thesis as gluon-fusion, gluon-gluon fusion, and ggF. The second highest production cross-section process is $pp \rightarrow HHjj$ (VBF), where VBF stands for Vector-Boson Fusion.

Gluon fusion

Whilst the Higgs boson only couples to mass, a pair of (massless) gluons may still form a pair of Higgs bosons via an intermediate loop of (virtual) fermions, e.g. top quarks. Whilst such modelling would normally require an NLO ME, generators such as MADGRAPH [208] can generate such processes at LO using a Higgs EFT model (NLO simulation is also

possible, as demonstrated in Fig. 6.10). As shown in Fig. 6.11 there are two possible diagrams: Fig. 6.11a the “box” diagram with no Higgs self-coupling, and Fig. 6.11b the “triangle” diagram which includes Higgs self-coupling.

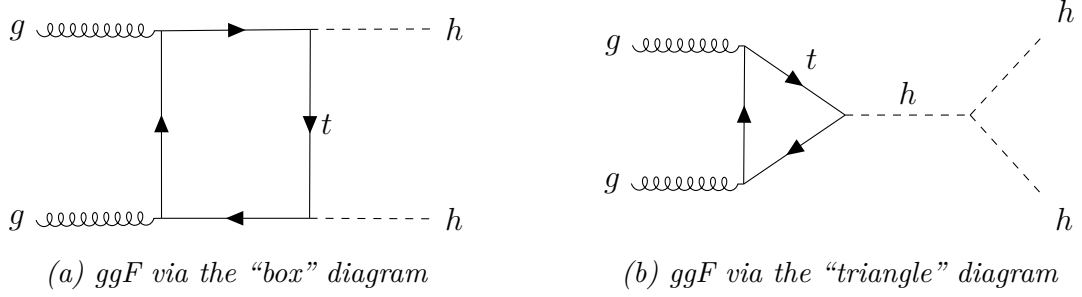
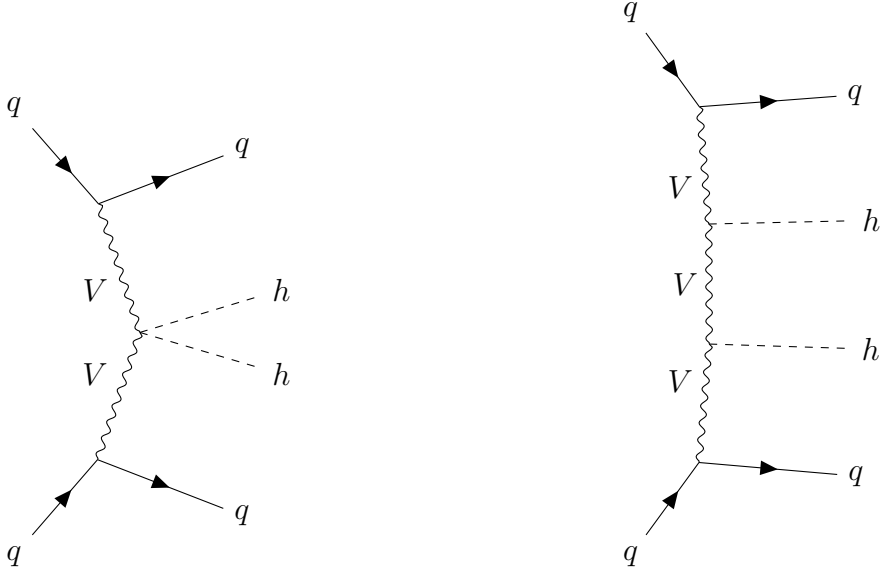


Figure 6.11: Diagrams for non-resonant di-Higgs production via gluon fusion.

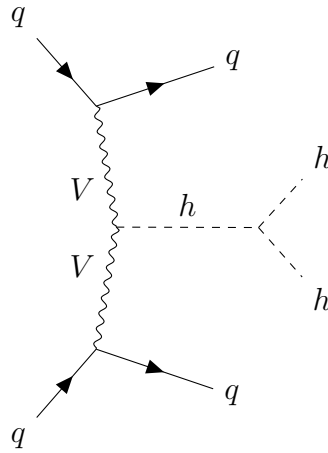
Vector-boson fusion

This production mechanism involves the Higgs coupling to massive vector bosons (W and Z). A pair of quarks radiate vector bosons which generate a pair of Higgs bosons: Fig. 6.12a through a four-point interaction; Fig. 6.12b through radiation of separated Higgs bosons; or through Higgs self-coupling. A key point is that despite having a lower cross-section compared to ggF , VBF results in two separated jets, which can potentially be used to identify it amongst other background processes.



(a) Direct production via VBF through the quadrilinear $VVHH$ coupling

(b) Separated production via VBF



(c) Production via VBF and Higgs self-coupling

Figure 6.12: Diagrams for non-resonant di-Higgs production via vector-boson fusion.

6.3 The $\tau^+ \tau^- b \bar{b}$ decay channel

6.3.1 Higgs decay channels

The Higgs boson decays prior to reaching the detector, therefore in order to identify di-Higgs production we must instead identify the decay products of the di-Higgs. As shown in Tab. 6.1, each Higgs boson can decay to a range of particles, some of which will undergo further decays.

6.3.2 Di-Higgs decay channels

Each decay channel comes with three concerns: the raw branching ratio - how often do such decays occur?; the selection efficiency - given such decays occur, how well can their final-states be reconstructed in the detector?; the background rates - what other processes can also produce similar final-states, and how often?

From Tab. 6.1 we can calculate that $h h \rightarrow b \bar{b} b \bar{b}$ has the highest branching ratio, and from Sec. 3.4.4 we know that the CMS detector has good algorithms for identifying the resulting b -jets, however QCD interactions will form a major background to this search. Whilst QCD can be simulated in Monte Carlo generators, the requirement of four b -jets will severely limit the selection efficiency, meaning that a vast amount of QCD samples would have to be generated in order to reduce the statistical uncertainty on yield rates to a manageable fraction. Normally, QCD contributions rely on extrapolation from control-regions of collider data, however this only provides differential yields, rather than individual events, which limits the complexity of analysis that may be performed.¹

What is needed in order to limit the contributions from QCD is a source of leptons: the CMS detector can reconstruct electrons and muons with high efficiency, and reconstruction of hadronic taus includes discriminators to reject fake taus due to QCD. Whilst the next most probable decay channel for a Higgs boson, $h \rightarrow W^+ W^-$, can provide a source of leptons in the subsequent W decays, W bosons decay 67.41 % of the time to hadrons [53], thereby reducing the effective cross-section.

$h \rightarrow \tau^+ \tau^-$ can also provide a source of leptons: the taus will decay either leptonically to electrons or muons, or hadronically to quarks which can be reconstructed as a hadronic tau. $h h \rightarrow \tau^+ \tau^- \tau^+ \tau^-$, however has a rather low effective cross-section. Instead, $h h \rightarrow \tau^+ \tau^- b \bar{b}$ combines the high branching ratio of $h \rightarrow b \bar{b}$, along with the source of triggering objects of $h \rightarrow \tau^+ \tau^-$.

Given the work required to complete an analysis in just one decay channel, the search effort for di-Higgs production at Run-II of the LHC has been divided amongst groups of

¹The eventual CMS search for non-resonant $h h \rightarrow b \bar{b} b \bar{b}$ [210] published in 2018 used a more advanced method [211] of building an event-wise background sample from collider data, which then allowed the training of a boosted decision tree to help classify signal and background.

Channel	Branching fraction
$h \rightarrow b \bar{b}$	5.824×10^{-1}
$h \rightarrow W^+ W^-$	2.137×10^{-1}
$h \rightarrow g g$	8.187×10^{-2}
$h \rightarrow \tau^+ \tau^-$	6.272×10^{-2}
$h \rightarrow c \bar{c}$	2.891×10^{-2}
$h \rightarrow Z Z$	2.619×10^{-2}
$h \rightarrow \gamma \gamma$	2.270×10^{-3}
$h \rightarrow Z \gamma$	1.533×10^{-3}
$h \rightarrow \mu \bar{\mu}$	2.176×10^{-4}

Table 6.1: A summary of the significant decay channels for a Higgs boson with mass 125 GeV, ordered by decreasing branching fraction as sourced from [209].

researchers by decay channel. In this thesis, I focus on the efforts in the $h h \rightarrow \tau^+ \tau^- b \bar{b}$ decay channel.

6.3.3 $h h \rightarrow \tau^+ \tau^- b \bar{b}$ sub-categories and decay channels

$h \rightarrow \tau^+ \tau^-$ decay-channels

The $h \rightarrow \tau^+ \tau^-$ decay can lead to three possible types of decay channels, as illustrated in Fig. 6.13: Fig. 6.13a fully-leptonic, in which both W bosons decay to light leptons ($\ell \ell$ visible final states); Fig. 6.13b fully-hadronic, in which both W bosons decay to quarks, i.e. two hadronic taus ($\tau_h \tau_h$ visible final states); and Fig. 6.13c semi-leptonic, in which one W boson decays to a light lepton and the other to hadrons ($\ell \tau_h$ visible final states). The branching fractions for the W boson are listed in Tab. 6.2.

From these, we can compute the branching fractions for $h \rightarrow \tau^+ \tau^-$. From Tab. 6.3 we can see that 94 % of di-tau decays involve at least one hadronic tau; the fully-leptonic channels only account for 6 % of decays and are therefore unlikely to contribute heavily to di-Higgs sensitivity at this stage. Due this, the only channels investigated in this thesis are $e \tau_h$, $\mu \tau_h$, and $\tau_h \tau_h$. Note that while electron and muon reconstruction efficiencies are high for CMS, τ_h reconstruction requires passing the HPS selection and the anti-fake discriminators which will reduce slightly the selection efficiency of the $\tau_h \tau_h$

$h \rightarrow b \bar{b}$ jet categories

The hadronisation of the b -quarks resulting from the $h \rightarrow b \bar{b}$ decay will result in b jets. The b -tagging algorithms at CMS, however, are not 100 % efficient meaning that not all of the b jets will be tagged. These missed jets can potentially be recovered by relaxing the

Channel	Branching fraction
$W^+ \rightarrow \bar{e} \nu_e$	0.1071
$W^+ \rightarrow \bar{\mu} \nu_\mu$	0.1063
$W^+ \rightarrow \bar{\tau} \nu_\tau$	0.1138
$W^+ \rightarrow q q'$	0.6741

Table 6.2: A summary of the experimentally measured branching fractions for the W boson. Data sourced from [53].

Channel	Branching fraction
$\tau^+ \tau^- \rightarrow e e$	0.015
$\tau^+ \tau^- \rightarrow e \mu$	0.030
$\tau^+ \tau^- \rightarrow \mu \mu$	0.015
$\tau^+ \tau^- \rightarrow e \tau_h$	0.18
$\tau^+ \tau^- \rightarrow \mu \tau_h$	0.18
$\tau^+ \tau^- \rightarrow \tau_h \tau_h$	0.58
$\tau^+ \tau^- \rightarrow \ell \ell$	0.06
$\tau^+ \tau^- \rightarrow \ell \tau_h$	0.36

Table 6.3: A summary of the branching fractions for a pair of tau leptons. Note, only the visible products are shown and branching fractions account for the charge combinations.

constraints on the jet requirements, e.g. requiring two jets but only one one which need to be tagged. Additionally, in some cases the Higgs boson may have been sufficiently boosted that the resulting b jets are merged into a single “fat” jet. These different possible jet specifications are likely to contain different populations of signal and background and so analysis sensitivity can be improved by devising multiple jet categories which are mutually exclusive.

6.3.4 Major background processes to the $hh \rightarrow \tau^+ \tau^- b\bar{b}$ decay channel

As alluded to in Sec. 6.3.2, several other physical processes can contribute final-states which can match those of $hh \rightarrow \tau^+ \tau^- b\bar{b}$. For quantitative contributions, one can check Tab. 10.5, for example.

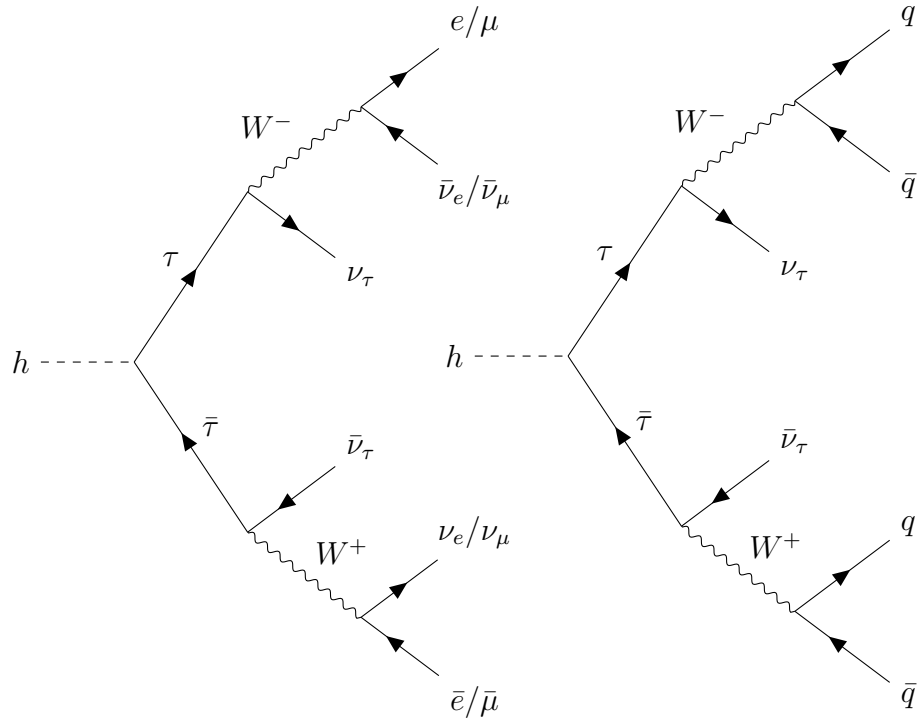
Semi-leptonic channels

In the semi-leptonic channels, top-quark pair-production ($t\bar{t}$) forms the dominant background. As shown in Fig. 6.14 $t\bar{t}$ can easily produce two b -quarks and a light lepton. The hadronic tau can then either be genuine (Fig. 6.14b), or faked via mistagging a light lepton or quark-jet (Fig. 6.14a). $t\bar{t}$ is usually completely modelled using MC.

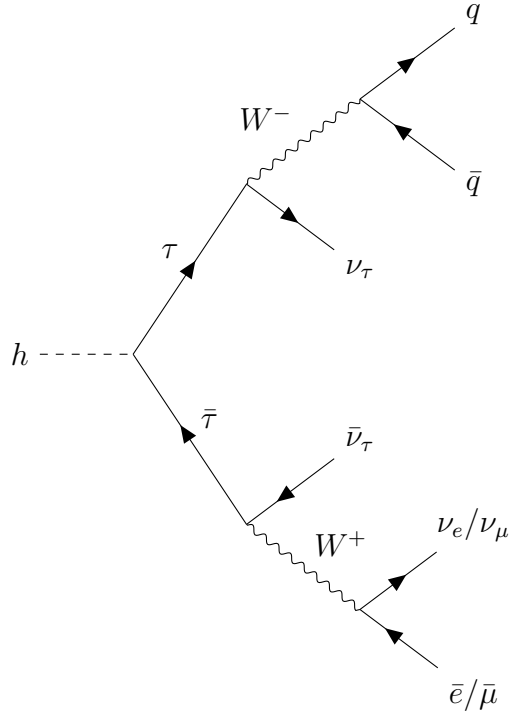
Lepton-pair production via quark-pair annihilation (Drell-Yan (DY)) also contributes a background, although at a reduced rate compared to $t\bar{t}$. As shown in Fig. 6.15, this can contribute by either mistagging an electron or muon as a hadronic tau, or generate a genuine di-tau pair. Given the busy environment of a LHC collision, it is not inconceivable that a pair of b jets, either real or mistagged, could be found within the same event, perhaps from initial-state radiation of a gluon from one of the incoming quarks. DY can be modelled using MC, however the event weights are usually rescaled using collider data in control regions.

Fully-hadronic channels

The fully hadronic channel ($\tau_h \tau_h b\bar{b}$) cuts down quite heavily on the $t\bar{t}$ background since two hadronic taus must be present, genuine, faked, or a mixture of both. This requirement similarly cuts down DY contributions. The removal of the light-lepton requirement, however, may make it easier for QCD to contribute, via jets faking hadronic taus. QCD in this analysis is completely inferred from collider data control regions.

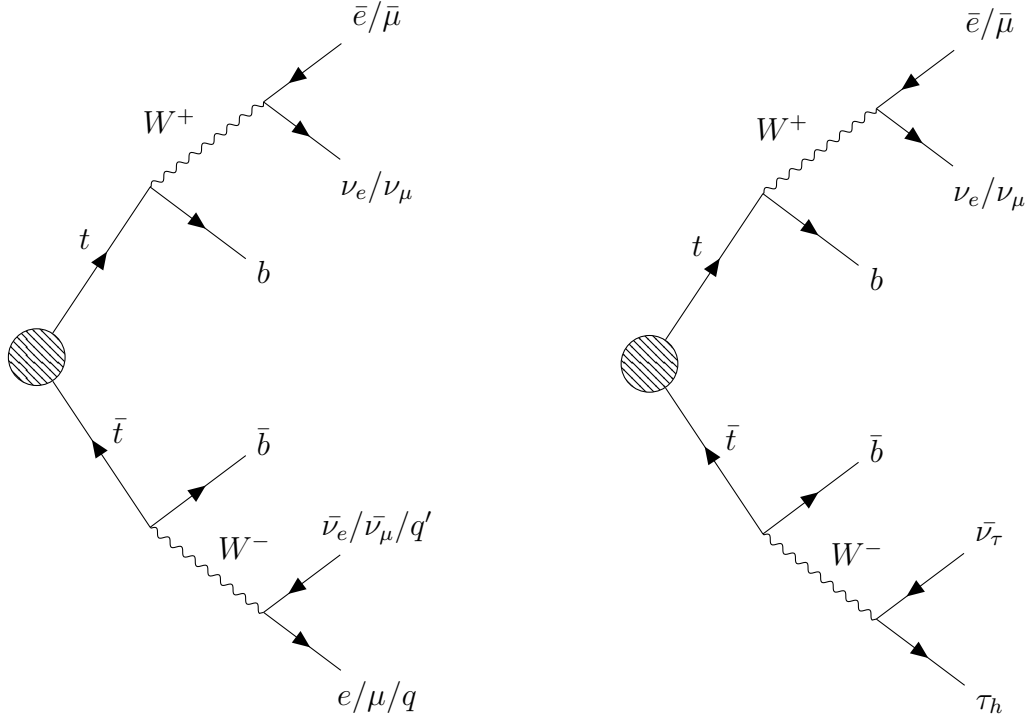


(a) Fully-leptonic decay channel (b) Fully-hadronic decay channel



(c) Semi-leptonic decay channel

Figure 6.13: Diagrams for di-tau decay channels.



(a) Faked τ_h through mistagging of an electron, muon, or jet

(b) Genuine τ_h production

Figure 6.14: $t \bar{t}$ background contributions to semi-leptonic $h h \rightarrow \tau^+ \tau^- b \bar{b}$.



(a) Faked τ_h through mistagging of an electron, muon

(b) Genuine di-tau production

Figure 6.15: Drell-Yan background contributions to semi-leptonic $h h \rightarrow \tau^+ \tau^- b \bar{b}$.

6.4 Basic analysis chain

The basic analysis chain for the (non-)resonant search proceeds thusly:

1. High-level triggers are applied to collider and MC data, with the MC data being rescaled if necessary to account for known discrepancies. The triggers used select events with electrons, muons, or taus;
2. Basic kinematic and multiplicity requirements are applied to the physics objects in the events. Again, corrections are applied to MC data to account for known discrepancies;
3. Events are assigned exclusively to decay channels ($e \tau_h b b$, $\mu \tau_h b b$, $\tau_h \tau_h b b$).
4. Events within each channel are then assigned to jet categories (e.g. two resolved b -jets, two resolved jets of which one is b -tagged, and one double- b -tagged jet (in which hadronisation products of both b -quarks are clustered in a single large- R jet, referred to as a *fat-jet*);
5. Candidates for $h_{b\bar{b}}$ and $h \rightarrow \tau^+ \tau^-$ are constructed;
6. Depending on the channel, category, and search, further cuts might be made (e.g. mass requirements on the Higgs candidates), or a machine learning algorithm may be applied, either to provide a prediction on which to cut, or a new highly-discriminant variable to use for hypothesis testing;
7. The presence of a signal is checked and in the case of a lack of signal, upper limits on the production cross-section of the signal processes are inferred.

The above is an overview of the full analysis process, however in the preliminary chapters of this thesis, not all of these steps are made.

6.5 Current state of the art

Both the ATLAS and CMS collaborations have performed searches for di-Higgs production in a variety of decay channels, and so far no signals have been observed. Reported below are summaries of the $h h \rightarrow \tau^+ \tau^- b \bar{b}$ analyses.

Existing results in 2016

On beginning my PhD research in March 2016, an existing LHC result for di-Higgs production searches in the $h h \rightarrow \tau^+ \tau^- b \bar{b}$ channel via gluon fusion came from the ATLAS

collaboration in Ref. [212] (September, 2015). This used 20.3 fb^{-1} of proton-proton collision data for Run-1 of the LHC with a centre-of-mass energy of 8 TeV.

Their analysis only considered the semi-leptonic decays of $h \rightarrow \tau^+ \tau^-$, where one tau decays hadronically, and the other leptonically. Window-cuts are applied to the Higgs candidates to select events in the Higgs-mass region, and another mass-requirement is used to reduce specifically contributions from $t\bar{t}$. Events are then divided into four exclusive categories according to b -jet multiplicity and the transverse momentum of the $h_{\tau^+ \tau^-}$ candidate. Variable distributions within each category are used for limit extraction: for the non-resonant search $h_{\tau^+ \tau^-}$ mass is used, and for the resonant search the di-Higgs mass is used.

Ref. [212] places an observed 95 % confidence upper-limit on non-resonant di-Higgs production of 160 times the SM expectation (130 expected). Limits on heavy spin-0 particles are shown as a function of mass in Fig. 6.16.

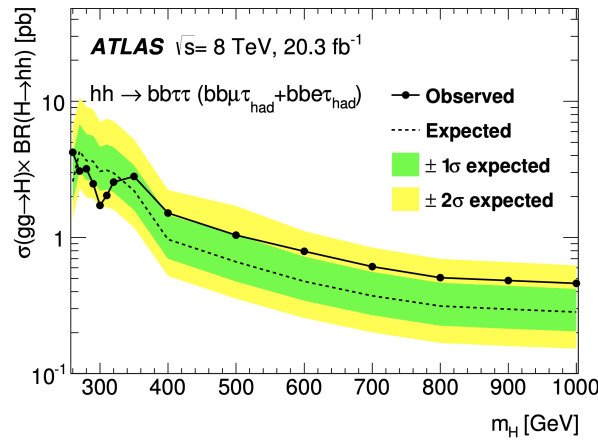


Figure 6.16: ATLAS $\sqrt{s} = 8 \text{ TeV}$: Observed and expected 95 % confidence upper-limit on spin-0 heavy resonances. Image source [212].

CMS 2017

CMS released results for a similar analysis in March 2017 [213] (later published in Ref. [214]), this took place during my preliminary exploration of this search [41] (February, 2017) and [42] (October, 2017), reported in Secs. 7 and 8.

The analysis used 35.9 fb^{-1} of proton-proton collision data for Run-2 of the LHC with a centre-of-mass energy of 13 TeV to study non-resonant and resonant (spin-0 and spin-2) di-Higgs production via gluon fusion. The semi-leptonic and fully-hadronic channels are used. Mass requirements are again placed on the Higgs candidates. Two BDTs are trained, one for low-mass resonances and non-resonant searches, and another for high-mass

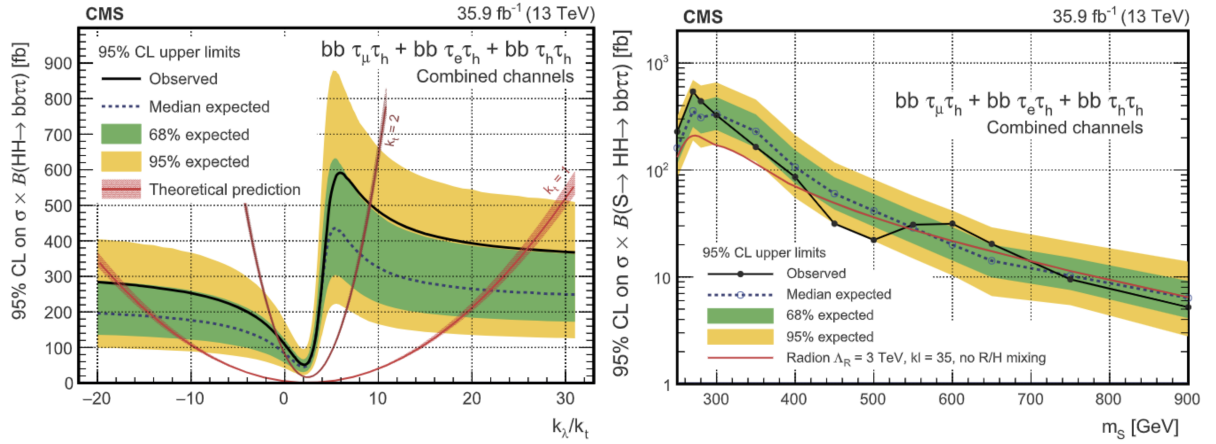
resonant searches, to classify signal and $t\bar{t}$. The BDTs are only used in the semi-leptonic channels; the $\tau_h\tau_h b\bar{b}$ channel is left as is. Thresholds are placed on the BDT predictions in order to cut out background contributions.

Three jet categories are used per channel:

1. 1 jet and 1 b jet;
2. 2 b jets;
3. double b fat-jet.

For the non-resonant search, limits are extracted via fits to the transverse mass (Ref. [215] - a variable with good discriminating power in the search) in the nine categories (three channels with three jet-categories). For the resonant search the di-Higgs mass is used as the fitting variable, again in nine categories.

Ref. [213] places an observed 95 % confidence upper-limit on non-resonant di-Higgs production as a function of $\kappa_\lambda/\kappa_t^2$ (Fig. 6.17a), reaching 28 times the SM expectation (25 expected) for $\kappa_\lambda = \kappa_t = 1$. Limits on heavy spin-0 and spin-2 particles are shown as a function of mass in Fig. 6.17b.



(a) Non-resonant limits as a function of κ_λ/κ_t (b) Resonant limits on spin-0 and spin-2 (radion) heavy resonances

Figure 6.17: CMS $\sqrt{s} = 13$ TeV: Observed and expected 95 % confidence upper-limit. **N.B.** in comparison to Fig. 6.16, cross-sections here are in fb. Image source [214].

² κ_t is the ratio of the top Yukawa coupling with respect to its SM value.

CMS 2018

Whilst not currently approved by CMS, the $hh \rightarrow \tau^+ \tau^- b \bar{b}$ search was the subject of a master's degree thesis completed in April 2018 [216]. The analysis used the same dataset as the CMS 2017 result, but improved on the signal classification and limit extraction.

BDTs were used for all channels with a dedicated training for non-resonant searches, and three trainings for the resonant search in groups of masses, but still only considering $t \bar{t}$ for the background. The BDTs used an extended set of input features as selected using a series of statistical methods. Additionally, the BDTs take parameterisation features as input, where appropriate, to allow them to fit better to a range of signal samples, these features are: κ_λ , channel, spin, and resonant mass. Hypothesis testing used instead the shape of the BDT prediction within each category, rather than the di-Higgs or transverse mass. These improvements provide an expected 95% confidence upper-limit on non-resonant di-Higgs production of 15.5 times the SM expectation, with improvements also to the resonant-search limits.

ATLAS 2018

The most stringent publicly available and collaboration-endorsed limits on $hh \rightarrow \tau^+ \tau^- b \bar{b}$ currently come from ATLAS in Ref. [217]. The analysis used 36.1 fb^{-1} of proton-proton collision data for Run-2 of the LHC with a centre-of-mass energy of 13 TeV to study non-resonant and resonant (spin-0 and spin-2) di-Higgs production via gluon fusion.

Building on top of the previous ATLAS analysis, they introduce the fully-hadronic decay channel in addition to the semi-leptonic ones, and BDT discriminators. These are trained separately per channel and search to classify signal and background, where background contains a variety of processes rather than just $t \bar{t}$. For the resonant search, BDTs are trained for each mass-point considered. For event selection, a relaxed mass requirement is used, consisting of just a lower bound on the di-tau mass, presumably to increase selection efficiency and rely more on the BDTs. Fits are made using the BDT predictions.

Ref. [217] places an observed 95% confidence upper-limit on SM non-resonant di-Higgs production of 12.7 times the SM expectation (14.8 expected). Limits on heavy spin-0 particles are shown as a function of mass in Fig. 6.18.

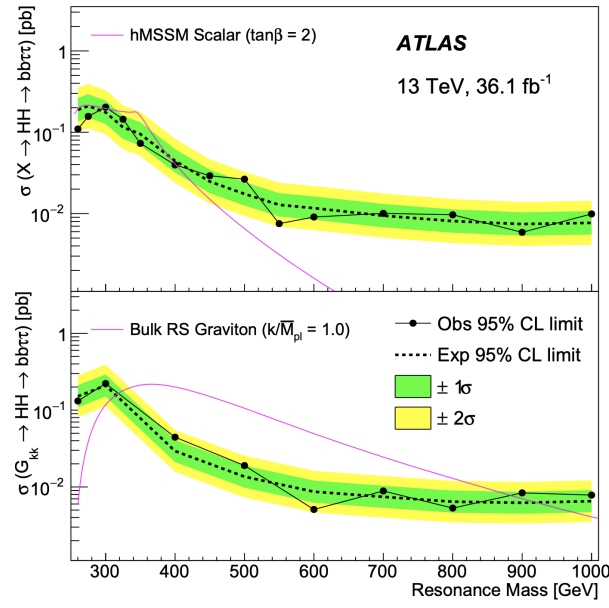


Figure 6.18: ATLAS $\sqrt{s} = 8 \text{ TeV}$: Observed and expected 95 % confidence upper-limit on spin-0 heavy resonances. **N.B.** in comparison to Fig. 6.17, cross-sections here are in pb. Image source [217].

Chapter 7

Preliminary investigation: Regression in di-Higgs searches

7.1 Overview

7.1.1 Event reconstruction in $\tau^+ \tau^- b \bar{b}$

As discussed in Sec. 3.4, reconstruction of particle collisions at the CMS detector involves compacting the detected final-states into a reduced set of *physics objects*. Of primary interest to the search for $h h \rightarrow \tau^+ \tau^- b \bar{b}$ are: muons, electrons, hadronically decaying tau leptons (τ_h), jets, and missing transverse momentum. As mentioned in Sec. 6.4, Higgs boson candidates are reconstructed from a pair of jets and from a pair of tau-candidates (either light leptons or hadronic tau leptons), and the di-Higgs vector then reconstructed from the Higgs candidates.

One limitation of the CMS detector, though, is its inability to detect neutrinos, which can only be inferred by computing the vector of missing momentum in the transverse plane ($\vec{p}_T^{\text{miss}}$) which is necessary to balance the transverse momenta of the detected final-states in each event. This is a particular problem for computing the $h_{\tau^+ \tau^-}$ momentum vector, since the subsequent decay of the tau leptons provides a natural source of neutrinos. Computing the $h_{\tau^+ \tau^-}$ momentum vector using only the visible decay products is therefore inaccurate and results in a systematic underestimation of the Higgs mass.

One attempt to improve the $h_{\tau^+ \tau^-}$ reconstruction is to include also the $\vec{p}_T^{\text{miss}}$ vector, under the assumption that all of the missing energy in the event is due to the neutrinos in the tau decays. This however is not completely true, since limitations in the detector's energy resolution, detection inefficiencies, and other sources of neutrinos in the event also contribute to the missing energy. This, in turn, can lead to an overestimation of the $h_{\tau^+ \tau^-}$ mass.

Additionally, the $h_{b \bar{b}}$ vector is subject to mis-reconstruction: the subsequent hadronisation of the b -quarks means that the $h_{b \bar{b}}$ must be estimated using b -tagged jets, whose

kinematic properties are due to the particles which were clustered into them. However, not all of the b -quark hadronisation products may be clustered within the jet, and some of these may be neutrinos anyway.

The inability to precisely reconstruct the Higgs and di-Higgs momentum vectors, means that two of the key features of the signal events, the masses of the Higgs, are not sharp peaks at specific values, but rather have wider distributions which are not necessarily centred at the expected values. Similarly, the mass of the di-Higgs candidate can be expected to be quite different for signal and background, but will be poorly estimated under naïve reconstruction. This makes the signal events more difficult to extract from the background events.

7.1.2 Regression in $\tau^+ \tau^- b \bar{b}$

Whilst the amount of missing transverse momentum can be estimated, this will be a combination of both neutrinos and detector effects, and since both the $h_{\tau^+ \tau^-}$ and $h_{b \bar{b}}$ decays can give rise to neutrinos, it is unclear exactly how the missing momentum should be distributed when computing them. Ideally, what we want to know is, given the observable features of the event ($\bar{x}$), what the mass is of a particular object (y):

$$y = f(\bar{x}), \quad (7.1)$$

where $f(\bar{x})$ is some function which computes the mass. Given the uncertainties in correctly distributing $\vec{p}_T^{\text{miss}}$, finding an analytical form for f is difficult, however we can instead approximate f with a neural network by training a regression model.

By training on Monte Carlo data, we know what the true 4-momenta of each particle in the event is, and through detector simulation, we can approximate reconstructed values of the 4-momenta of each final-state particle. This means we can train the regressor by using the generator-level values as targets y and the reconstructed features as x . A well-trained neural network, $f_\theta(\bar{x})$, then accounts for the correct distribution of the missing transverse momenta and the detector effects.

The aim is to train regressors such that we can compute more precise masses for the Higgs and di-Higgs candidates in signal, therefore we should only use signal data during training. The next problem, however, is that we then have to be able to apply the regressor to non-signal data and get a different response; if the output of the regressor were the same for signal and background then there would be no way to tell the classes apart using the mass features. Whilst for non-resonant di-Higgs production the true di-Higgs mass is a continuous distribution, the true values of the Higgs masses are constant. Training a regressor to directly output the Higgs mass would therefore lead to a model which consistently output 125 GeV, with no regard to its inputs, therefore a more indirect approach must be taken when attempting to regress to singular valued features.

7.1.3 Feature optimisation via regression

Aside from offering more precise and accurate estimates for the true values of particle properties, regression can be thought of as a way of optimising the features received by a classifier: by refining high-level features, such as the masses of resonant particles, for given particle processes, the differences between the probability density functions (PDFs) of the particle processes can become more apparent, thereby aiding the classifier in discriminating between them.

Whilst one of the main advantages of DNNs is that they have the potential to learn their own high-level features, they are limited by the fact that the inputs are reconstructed estimates of the underlying particle interactions, and hence carry a resolution determined by the quality of the experimental apparatus. By regressing the input features, it is possible that our theoretical knowledge of the *true* values can be folded into the input features. This would provide extra information which is otherwise unavailable to the NN.

Whilst this section deals primarily with developing well-performing regressors, these ideas will be further explored in Chap. 8 by comparing the performance of classifiers which receive an optimised dataset against those which receive the base-reconstructed dataset.

7.1.4 Regression approaches

As mentioned above, when attempting to regress to singularly valued features, one must take an indirect approach in order to ensure that the actual targets used during the regressor training have a distribution of values. This means we cannot directly regress to the Higgs mass. One way to indirectly regress to the Higgs mass is to instead regress to the 3-momenta of the pair of objects which result from its decay, e.g. regressing to the 3-momenta of the tau leptons resulting from the $h_{\tau^+\tau^-}$ decay. The invariant mass of the parent Higgs-boson can then be calculated by using the known masses of the child particles.

In this approach of *pair-object* (PO) momenta-regression, the regressor produces six outputs, corresponding to each component of the 3-momentum of each child particle. Due to the closed boundaries of the azimuthal angle ($-\pi = +\pi$) regressing to the ϕ component of momentum is non-trivial, instead momenta are transformed into a Cartesian coordinate system, i.e. p_x , p_y , and p_z .

Result summary

Despite regressing to the Higgs-boson mass indirectly, I eventually found that the regressor still produces a peak around the Higgs-boson mass when run on background events. From this I inferred that the NN learns that the Higgs mass is a constant value and uses it to aid regression, despite not being explicitly provided with it. In order to prevent this background peak, I introduced a second approach called *single-object* (SO) regression.

In this system, the child particles are regressed one at a time, with input features being removed if including them results in a background peak. SO regression results in an improved mass-estimate, compared to that of base reconstruction, but it leads to poorer momentum regression than PO regression.

For non-resonant di-Higgs production, the di-Higgs mass can be regressed to directly, however it is found that it is more useful to use instead the outputs of the previous regressors to provide optimised values of the momenta of the particles, and to also regress to the 3-momentum of the di-Higgs vector prior to regressing to its mass.

7.2 Datasets, event selection, and feature definitions

7.2.1 Training samples

The studies performed in this chapter, and later in Chap. 8, only consider SM non-resonant di-Higgs production and the dominant background for $hh \rightarrow \tau^+ \tau^- b \bar{b}$ decay channel, inclusive $t \bar{t}$ production.

The samples were produced via Monte Carlo simulation for a centre-of-mass energy of 13 TeV, using GEANT 4 [39, 134] to simulate the response of the CMS detector. The events reconstruction is performed via CMSSW [75], in order to obtain the final-state physics objects.

The signal sample is used for training and evaluating the regression models, and the background sample is only used to check the response of the trained model on some non-signal data.

Signal sample

The signal sample of SM non-resonant $hh \rightarrow \tau^+ \tau^- b \bar{b}$, uses leading-order Matrix-Elements generated by MADGRAPH 5 [208]. The parton showering, hadronisation, and decays were handled by PYTHIA 8 [117]. PYTHIA 8 is applied such that the decays of the Higgs bosons are forced into $hh \rightarrow \tau^+ \tau^- b \bar{b}$ channel.

The ME production cross section reported by MADGRAPH 5 is $14.518 \pm 0.001 \text{ fb}$. Using next-to-next-to-leading-log (NNLL) calculations matched to next-to-next-to-leading-order (NNLO) and accounting for top-quark mass effects to next-to-leading-order (NLO), the LHC Higgs Cross-Section Working Group [218] calculates that the production cross section for $gg \rightarrow hh$ at $\sqrt{s} = 13 \text{ TeV}$ is $33.5_{-2.3}^{+1.8} \text{ fb}$, for $m_h = 125 \text{ GeV}$, and that the branching ratios for $h \rightarrow b \bar{b}$ and $h \rightarrow \tau^+ \tau^-$ are $0.5824_{-0.0074}^{+0.0072}$ and 0.06272 ± 0.0010 , respectively. The theoretical production cross section is therefore $2.45_{-0.17}^{+0.14} \text{ fb}$.

Inclusive $t\bar{t}$ background sample

The background sample of $t\bar{t}$ production is the CMS sample, uses NLO MEs generated by POWHEG BOX 2 [219, 124, 220] using the `hvq` process [221]. Again, the parton showering, hadronisation, and decays were handled by PYTHIA 8 [117].

The production cross section reported by POWHEG BOX is 73.04 ± 0.06 pb. Using NNLL matched to NNLO calculations and assuming a top-quark mass of 173.2 GeV, the production cross section for $pp \rightarrow t\bar{t}$ at $\sqrt{s} = 13$ TeV is $816.0^{+39.5}_{-44.7}$ pb [218].

7.2.2 Object definitions

Primary vertices The interaction vertices in each event are identified using tracker information via a deterministic annealing algorithm [222, 223]. The primary vertex from which the final-states are considered to originate is selected as the vertex with the highest sum of p_T^2 of associated tracks after basic selection.

Electrons Electrons are reconstructed using information from the tracker and the ECAL of the simulated CMS detector. Reconstructed electrons are then selected according using a multivariate-based ID with $p_T > 24$ GeV, $|\eta| < 2.4$, maximum transverse separation from the primary vertex $|d_{xy}| < 0.045$ mm, and maximum longitudinal separation from the primary vertex $|d_z| < 0.2$ mm.

Muons Muons are reconstructed using tracker information. The reconstructed muons are then required to pass the following cuts: $p_T > 22$ GeV, $|\eta| < 2.4$, maximum transverse separation from the primary vertex $|d_{xy}| < 0.045$ mm, and maximum longitudinal separation from the primary vertex $|d_z| < 0.2$ mm.

Lepton Isolation Leptons are required to be *isolated* in terms of a relative isolation variable, I_{rel} . This variable is defined as the ratio of the scalar p_T sum of photons, and charged and neutral hadrons, which emerge from the same vertex as the selected electron (muon) and are within a cone of radius $\Delta R = \sqrt{(\Delta\phi)^2 + (\Delta\eta)^2} = 0.3$ (0.4) centred around the reconstructed lepton candidate. The isolation of particle p is defined as:

$$I_{\text{rel},p} = \frac{\sum p_{T,R < R_{\text{max}}}^{CH} + \sum E_{T,R < R_{\text{max}}}^{NH} + \sum E_{T,R < R_{\text{max}}}^{\gamma} - 0.5 \times \sum p_{T,R < R_{\text{max}}}^{PU}}{p_{T,p}}, \quad (7.2)$$

where $p_{T,R < R_{\text{max}}}^{CH}$ is the p_T of charged hadrons, $E_{T,R < R_{\text{max}}}^{NH}$ is the E_T of neutral hadrons, $E_{T,R < R_{\text{max}}}^{\gamma}$ is the E_T of photon, and $p_{T,R < R_{\text{max}}}^{PU}$ is the p_T of pile-up objects, all of which only contribute if within a cone of radius R_{max} .

The isolation variable is corrected for energy deposits from pileup interactions and is required to be less than 0.10 (0.15) for an electron (muon).

Hadronic τ s A hadronic τ (τ_h) is the decay of a τ lepton into hadrons and a neutrino. These are reconstructed using the “Hadrons Plus Strips” algorithm [224]. The basic kinematic requirements are: $p_T > 20 \text{ GeV}$, $|\eta| < 2.4$, and isolation $I_{\text{rel}} < 0.5$. Additional discriminators are then applied to reduce the chance of ‘fake’ τ_h s being created by jets, electrons, or muons.

Jets Particle-flow [89] objects are clustered into jets using the anti- k_T algorithm [96], with $R = 0.4$. Jets are required with $p_T > 20 \text{ GeV}$, $|\eta| < 2.7$, and to not be within a radius $\Delta R < 0.4$ of any electrons or muons in the event. Version 2 of the Combined Secondary Vertex (CSV) algorithm [225] is then applied to jets in order to associate a discriminator value for each jet, indicating its likelihood to have originated from the hadronisation of a b -quark, though no minimum requirement on the discriminator value is applied.

Missing transverse momentum The missing transverse momentum ($\vec{p}_T^{\text{miss}}$) is calculated for each event using the Particle-Flow $\vec{E}_T^{\text{miss}}$ algorithm [89].

7.2.3 Event selection

In this preliminary investigation, only the $\mu \tau_h b \bar{b}$ final-state category is considered, where τ_h indicates the hadronic tau decay with a resolved τ -tagged jet, and b indicates a resolved b -tagged jet.

Following the object definitions, I require events to contain exactly one muon, exactly one hadronic tau of opposite charge, no electrons, and at least two jets. In the case of more than two jets, the two with the highest b -tag CSV values are chosen as the candidate jets resulting from the decay of $h \rightarrow b \bar{b}$. These selected jets are subsequently referred to as the b -jets.

Signal events passing the selection requirements undergo an additional check using information from the MC generator (MC-truth) to set a flag indicating whether or not the selected final states correspond to the true decay products of both Higgs bosons.

- For the b -jets, this requires the generator-level b -quarks produced by the decay of a generator-level Higgs boson to be within $\Delta R = 0.5$ of their respective b -jets.
- For the hadronically decaying τ lepton, the generator-level τ lepton coming from the decay of a Higgs boson must be within $\Delta R = 0.5$ of the selected τ -jet.
- For the leptonically decaying τ lepton, the generator-level muon coming from the decay of a τ lepton coming from the decay of a Higgs boson must be within $\Delta R = 0.5$ of the reconstruction-level muon, and the difference in p_T between the two must be less than 20 GeV .

Sample	Acceptance [%]	Acceptance $\times$ cross-section [pb]
Signal	0.640 ± 0.005	$(1.6_{-0.1}^{+0.09}) \times 10^{-5}$
Signal MM	0.739 ± 0.005	$(1.8_{-0.1}^{+0.1}) \times 10^{-5}$
Bkg.	0.1788 ± 0.0004	$1.46_{-0.08}^{+0.07}$

Table 7.1: Acceptance and acceptance times production cross-section of events for each final state category. “Signal MM” (mis-matched) indicates events from the signal sample which fail the MC-truth match. Acceptance uncertainties are statistical only, acceptance $\times$ cross-section uncertainties include the theoretical uncertainties on the cross-section from Sec. 7.2.1.

If all four final states are correctly selected, the event is flagged as passing the MC-truth match.

Table 7.1 gives the values of acceptance and acceptance times production cross-section for signal, for signal failing the MC-truth check (“signal MM”, where incorrect final-states are selected), and for background events. The acceptance is estimated as the number of events passing the final event selection divided by the number of generated events in the signal sample ($h h \rightarrow \tau^+ \tau^- b \bar{b}$).

In total, just under 20 000 correctly selected signal events are available for training and evaluating the regressors, and just over 150 000 background events are available for checking the background response of the trained regressors.

7.2.4 Feature definition

A range of features are calculated for the data. Whilst only a subsection of them will be considered in this section, Chap. 8 also uses this same dataset and considers a greater range of features, so for simplicity I will describe all of the features together below.

Basic features

Once the events are selected, I reconstruct Higgs and di-Higgs candidates using the selected final states. First, the two τ -leptons are defined: for the hadronically decaying τ lepton, the 4-momentum of the τ -tagged jet is used; for leptonically decaying lepton (τ_μ), the 4-momentum of the muon is used. The 4-momentum of the parent Higgs-boson ($h_{\tau^+ \tau^-}$) is then reconstructed from the vectorial sum of the 4-momenta of the two τ leptons and $\vec{p}_T^{\text{miss}}$. Next, the 4-momentum of the Higgs boson which decays to $b \bar{b}$ ($h_{b \bar{b}}$) is reconstructed from the vectorial sum of the 4-momenta of the two selected jets. Finally, the combined 4-momentum of the two Higgs bosons is calculated (the di-Higgs vector).

Eighteen low-level final state features are calculated during the event selection process: p_T , η , and ϕ , for each selected final state, the magnitude of missing transverse momentum ($\vec{p}_T^{\text{miss}}$), and the azimuthal angle of the vector of the missing momentum ($\phi_{\vec{p}_T^{\text{miss}}}$).

Twelve reconstructed features are also calculated: p_T , η , ϕ , and the invariant masses of the three systems which correspond to: the Higgs boson which decays to $b\bar{b}$ ($h_{b\bar{b}}$), the Higgs boson which decays to $\tau^+\tau^-$ ($h_{\tau^+\tau^-}$), and the di-Higgs-boson system (hh). The transverse mass (m_T) of τ_μ is also calculated, according to:

$$m_T = \sqrt{2p_{T,\ell} \times \vec{p}_T^{\text{miss}} \times \left(1 - \cos \Delta\phi_{\ell, \vec{p}_T^{\text{miss}}}\right)}. \quad (7.3)$$

The 3-momenta components of the particles are also converted into Cartesian form, i.e. p_x , p_y , and p_z to be used as inputs to the regressors, however the (p_T, η, ϕ) form is still useful for calculating other features.

Example distributions of some of these features are shown in Fig. 7.1, and the invariant masses of the reconstructed Higgs bosons in Fig. 7.2.

High-level features

Hypothesis features The mass of the reconstructed $h_{\tau^+\tau^-}$ assumes that all of the missing energy in an event is due to the neutrinos in the decays of the τ leptons. This is a fairly crude assumption, since other neutrinos may be generated (for example in the decays of B hadrons) or detector-acceptance effects might lead to visible energy being missed. The **SVFit** algorithm [108] (see Sec. 3.4.5) uses a likelihood approach to better estimate the mass of the τ -lepton pair by assuming that they came from the decay of the same particle. This gives rise to an additional feature, the $h_{\tau^+\tau^-}$ mass as computed by **SVFit**.

Similar to how the $h_{\tau^+\tau^-}$ mass can be better estimated by assuming the production method of the final-states, the di-Higgs mass can also be better estimated. **HHKinFit** [112] (see Sec. 3.4.5) uses the momenta of the final-states to calculate the di-Higgs mass under the hypothesis of $hh \rightarrow b\bar{b}\tau^+\tau^-$. This gives rise to two additional features, the di-Higgs mass as computed by **HHKinFit**, and the probability of the hypothesis being correct as indicated by the convergence of the algorithm. Both $h_{\tau^+\tau^-}$ and di-Higgs masses computed by these approaches are illustrated in Fig. 7.3.

Global kinematic-features Four features help characterise global event by returning the kinematics of physics objects:

- H_T , the scalar sum of the transverse energy of both b -jets and the τ_h ,
- s_T , the scalar sum of $\vec{p}_T^{\text{miss}}$, muon p_T , and H_T ,
- E_{vis} , the sum of the energy of both b -jets, the τ_h , and the muon,

- Centrality, the scalar sum of the p_T of both b -jets, the τ_h , and the muon, divided by E_{vis} .

These features are plotted in Fig. 7.4.

Event-shape features 12 features which characterise the shape of events, defined using the eigenvalues of the following tensors:

$$\text{sphericity tensor} = \frac{1}{\sum_i |\vec{p}_i|^2} \sum_i \begin{bmatrix} p_{i,x}p_{i,x} & p_{i,x}p_{i,y} & p_{i,x}p_{i,z} \\ p_{i,y}p_{i,x} & p_{i,y}p_{i,y} & p_{i,y}p_{i,z} \\ p_{i,z}p_{i,x} & p_{i,z}p_{i,y} & p_{i,z}p_{i,z} \end{bmatrix}, \quad (7.4)$$

and

$$\text{spherocity tensor} = \frac{1}{\sum_i |\vec{p}_i|} \sum_i \left[\frac{1}{|\vec{p}_i|} \begin{bmatrix} p_{i,x}p_{i,x} & p_{i,x}p_{i,y} & p_{i,x}p_{i,z} \\ p_{i,y}p_{i,x} & p_{i,y}p_{i,y} & p_{i,y}p_{i,z} \\ p_{i,z}p_{i,x} & p_{i,z}p_{i,y} & p_{i,z}p_{i,z} \end{bmatrix} \right], \quad (7.5)$$

where $\sum_i$ is a summation over the selected final-states (both b jets, the hadronic tau, and the muon). The eigenvalues of the tensors are then calculated (six features), ordered, and normalised such that: $\lambda_1 \geq \lambda_2 \geq \lambda_3$ and $\lambda_1 + \lambda_2 + \lambda_3 = 1$. The following features are then calculated according to Ref. [226], and are illustrated in Fig. 7.5:

- Sphericity and spherocity: $S = \frac{3}{2}(\lambda_2 + \lambda_3)$: 2 features,
- Aplanarity and aplanority: $A = \frac{3}{2}\lambda_3$: 2 features,
- Υ of sphericity: $\Upsilon = \frac{\sqrt{3}}{2}(\lambda_2 - \lambda_3)$: 1 feature,
- Shape of spherocity: $D = 27\lambda_1\lambda_2\lambda_3$: 1 feature.

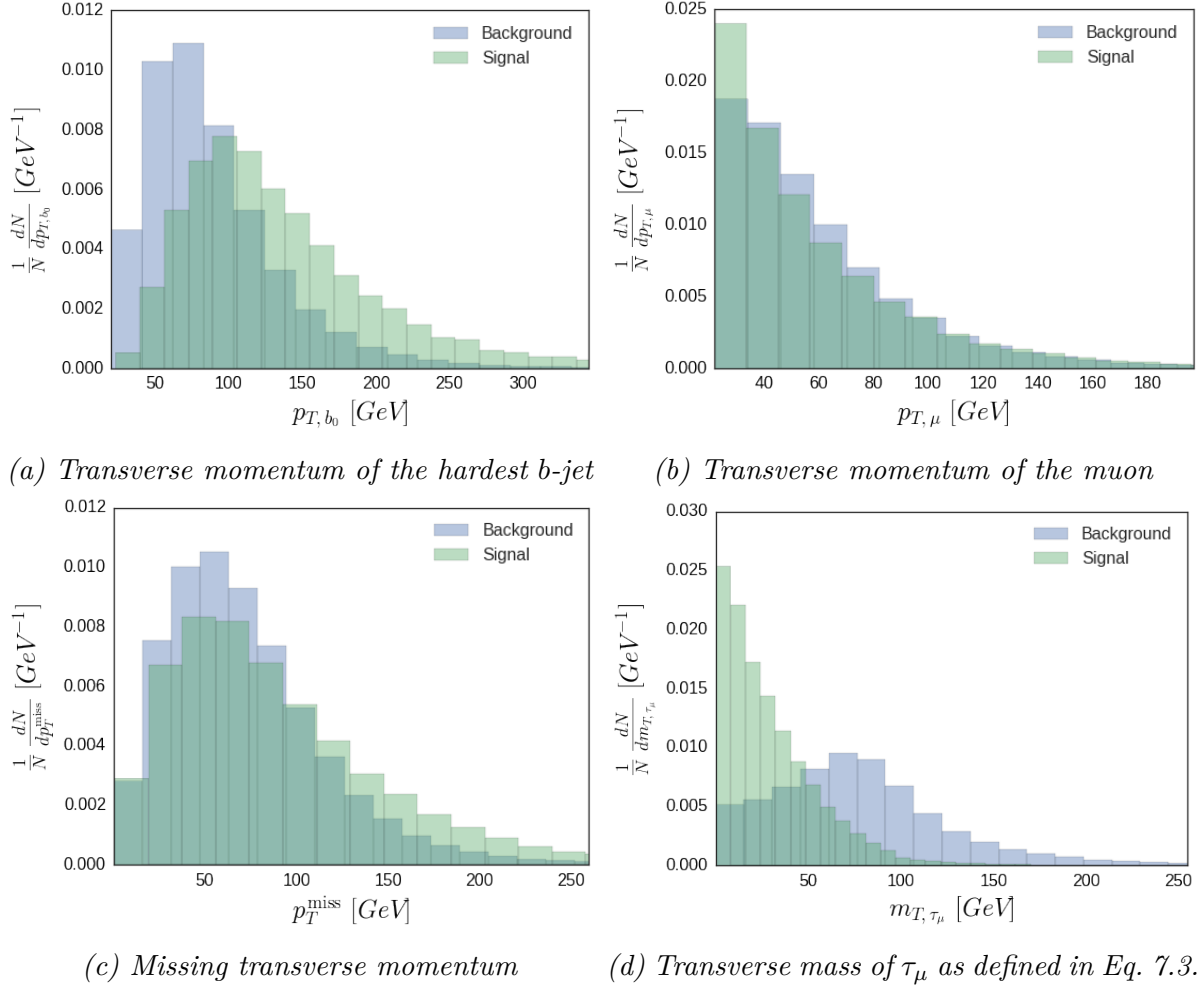
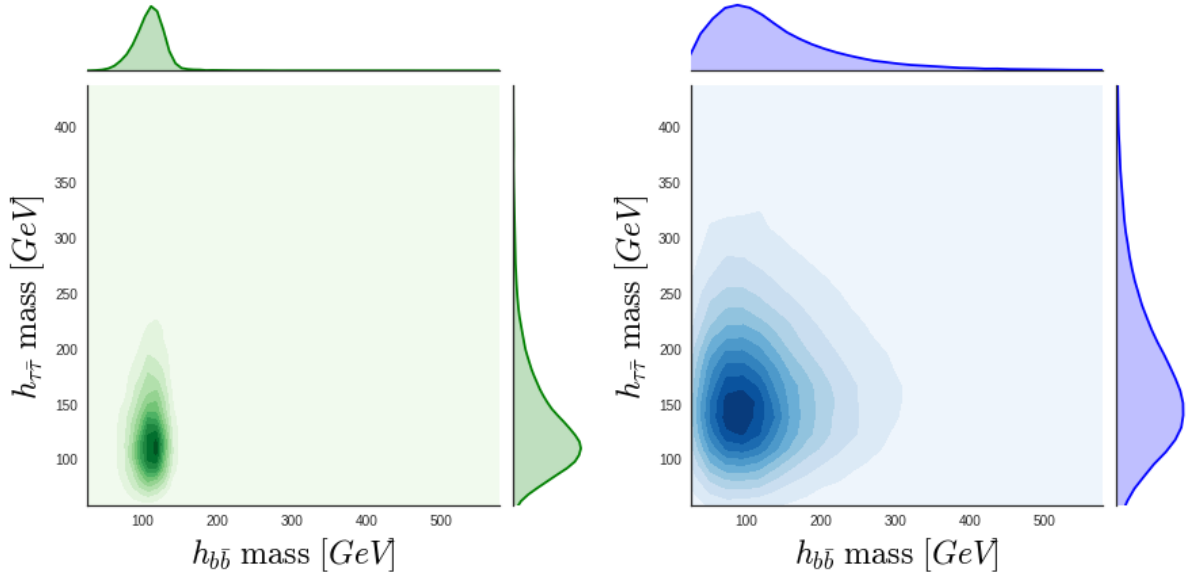


Figure 7.1: Example distributions for signal and background ($t\bar{t}$) processes of some of the low-level features for the final-state objects.



(a) Higgs boson masses in signal

(b) Higgs boson masses in background

Figure 7.2: 2D distributions of the invariant masses of the reconstructed Higgs bosons ($h_{\tau+\tau-}$ and $h_{b\bar{b}}$) in signal and background ($t\bar{t}$). 1D invariant-mass distributions are projected on the corresponding axes.

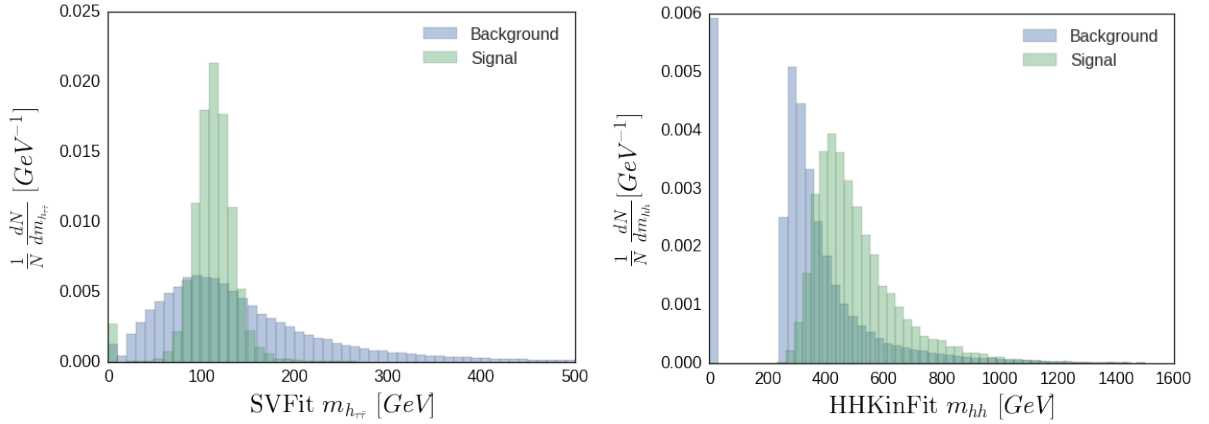
(a) $h_{\tau+\tau-}$ mass as computed by *SVFit*.(b) Di-Higgs mass as computed by *HHKinFit*.

Figure 7.3: Distributions of the $h_{\tau+\tau-}$ and di-Higgs masses for signal and background ($t\bar{t}$) processes as computed by *SVFit* and *HHKinFit*, respectively. Events with zero mass correspond to events for which the fitting method did not converge to a minimum, or no solution was found.

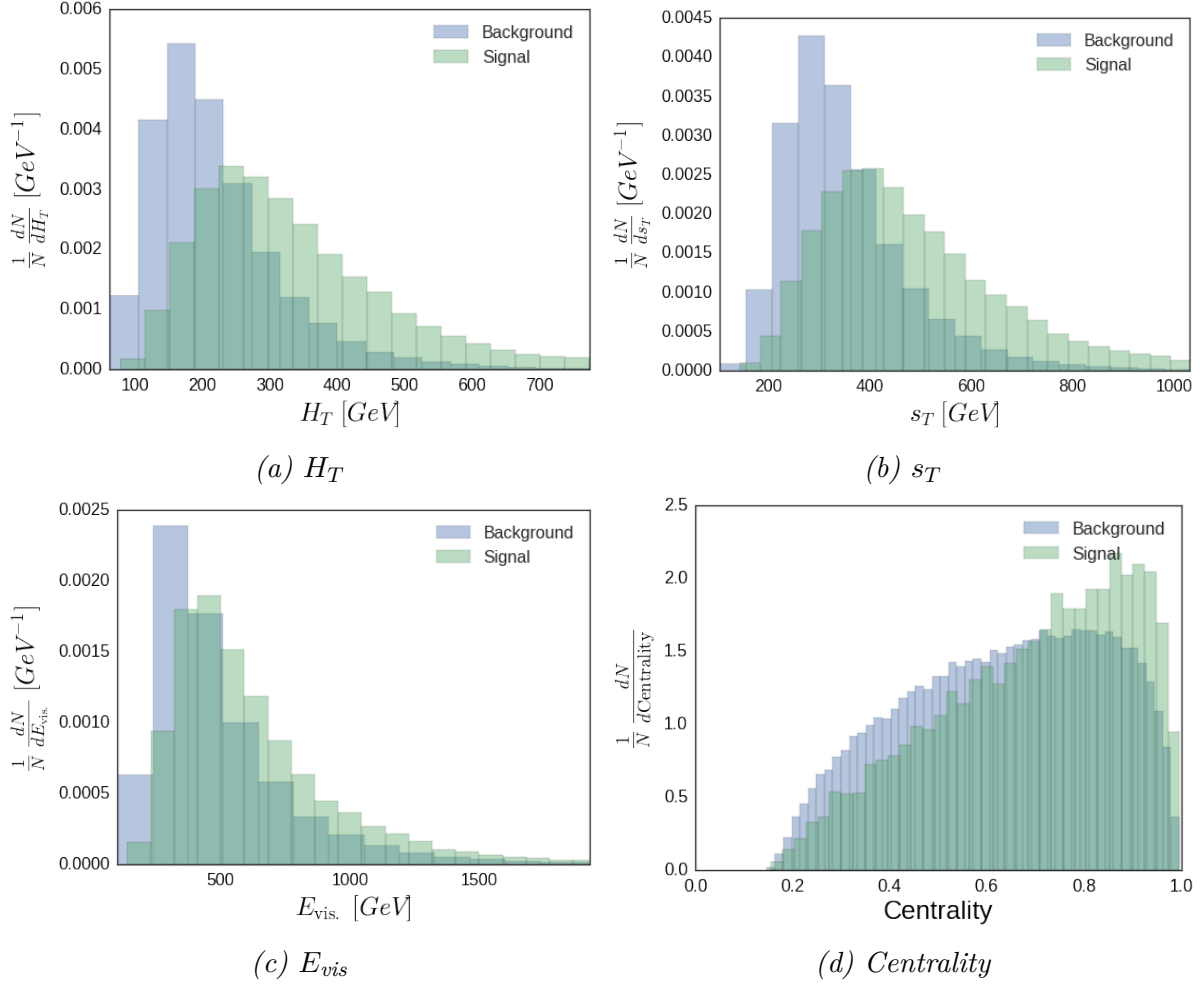


Figure 7.4: Distributions of four global event features for signal and background ($t\bar{t}$) processes calculated using the final-state objects (see Sec. 7.2.4).

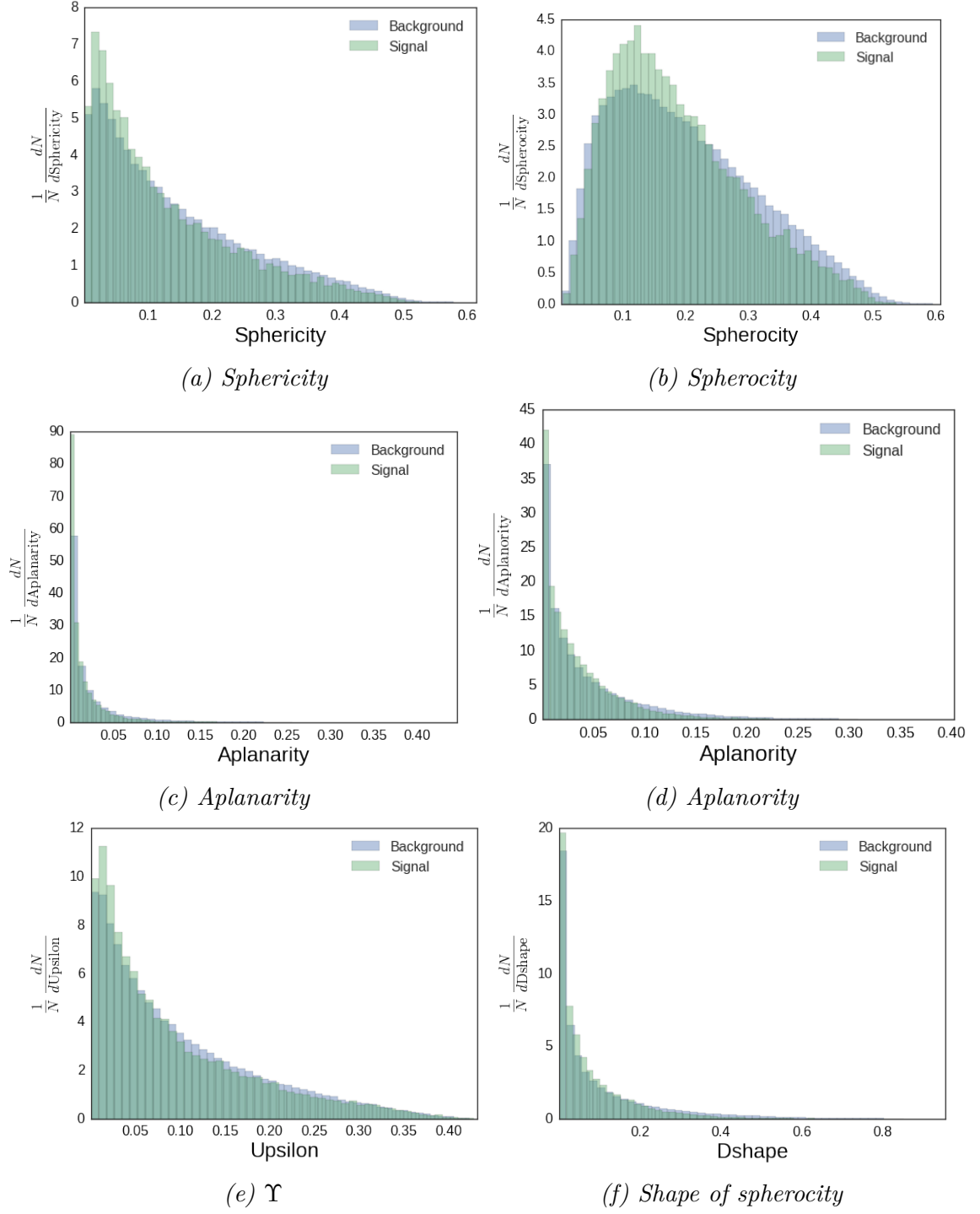


Figure 7.5: Distributions of the shape features for signal and background ($t\bar{t}$) processes calculated using the final-state objects (see Sec. 7.2.4).

7.3 Regressor architecture

KERAS [180] is used to implement neural networks. These consist of three hidden layers, each containing 100 neurons. All layers are initialised according to He-normal initialisation, except for the output nodes, which use Glorot initialisation. Internal activation functions are SELU [164], and the output nodes use linear activation, since the regressor target-features will be in the region $[-\infty, \infty]$.

The input features are scaled, standardised, and decorrelated via principle-component analysis ¹ using SKLEARN [190]. The target features are also scaled and standardised to account for the difference in scale between the p_z momentum component and those in the transverse plane.

The regressors are trained via 10-fold cross-validation (see Sec. 5.4.3) on 80% of the signal events which pass the MC-truth match (the correct final-states were chosen). This portion is referred to as the *training* set, while the remaining 20% is reserved for testing. The same split into trained and testing is made for all the regressors, to avoid misleading results on performance, whilst making efficient use of the available data.

Each regressor is trained in batches of 64 events of signal data, for a maximum limit of 10 000 epochs. The Nesterov-Adam optimiser [177] (see Sec. 5.2.4) is used to minimise the mean squared-error (MSE) of predictions (the loss function). The regressor's performance is evaluated after every epoch on the testing fold, and if the MSE does not decrease for ten epochs the training ceases; invariably, the upper limit of 10 000 is never reached. Additionally, the model is saved after each epoch if it provides the best performance on the testing set seen for the current training fold. Once training finishes, the best model is loaded. This early-stopping acts as a regulariser for the network reducing the possibility of over-training.

All ten NNs are then combined into an ensemble (see Sec. 5.4.4) by weighting their outputs according to the inverse of their loss on the testing fold. The response of the regressor ensemble is then confirmed on the testing data.

7.4 Higgs-boson mass regression

7.4.1 Pair-object momenta-regression

To begin with I will try training single regressors which will output all six components of the 3-momenta of a pair of Higgs decay products (e.g. p_x , p_y , and p_z for both of the taus resulting from the $h_{\tau^+\tau^-}$ decay). As mentioned in Sec. 7.1.4, this method of *pair-object* (PO) regression is later found to result in an adverse effect on background data. For Higgs-boson mass-regression the *single-object* (SO) approach (Sec. 7.4.2) is found to be

¹PCA [227] - a method of rotating the data in feature space, such that they are maximally decorrelated in terms of a new set of features.

more favourable, however the pair-object approach does result in better regression for the final-state momenta which is later used to aid the regression of the di-Higgs mass in Sec. 7.5.

Regression of the 3-momenta is performed in sequential steps, with the previously regressed features being used to help train later regressors. First the b -jets are regressed to their generator-level b -quark 3-momenta using only reconstructed features. Next the 3-momenta of the τ_h and the muon are regressed to those of the generator-level taus, using reconstructed features and the regressed momenta of the b -jets. Finally the 3-momenta of the di-Higgs are regressed to their generator-level values using the regressed features of the b -jets and the taus.

Features

The following 45 input features are used for each regressor:

- 4-momenta, and magnitude of 3-momenta of both b -jets, the muon, and the τ_h : 20 features,
- Mass of both b -jets and the τ_h : 3 features,
- 4-momentum, magnitude of 3-momentum, and mass of both Higgs bosons and di-Higgs system: 18 features,
- p_x , p_y , and magnitude of missing transverse momentum: 3 features,
- Transverse mass of leptonic tau ($m_{T,\mu}$): 1 feature.

In addition to these 45 features, the regression of the momenta of the b -jets also use the CSV values of each jet (47 features in total).

Momentum-regression performance

Fig. 7.6 shows examples of the regressors' responses on the signal samples for the x coordinate of the momentum of the hardest of the two b -jets, the τ -jet (only one is required in the selection), and the momentum of the di-Higgs system.

In the left column, the momenta distributions are shown for what was reported by the reconstruction algorithms (green), what was obtained after regression (magenta, cyan, and red), and what the true momenta of the object is at generator level (blue). Here, the closer the “regressed” distribution is to matching the “true” distribution the better. The right column shows the residual distributions, i.e. the difference between the estimated (reconstructed or reconstructed+regressed) and the true value, per event. Here, the closer the distribution is to a narrow peak centred at zero the better.

From the results, we can see that regression results in momenta estimations which more closely match the true values than what can be obtained by a simple base-level

reconstruction. This is most notable for the τ -jets, where energy is lost due to the presence of neutrinos in the decay of the τ lepton. However, regression also offers improvements in the b -quark and di-Higgs vector momenta estimations. From the residual distributions, we see that the standard deviations of the regressed distributions are smaller than those of the reconstructed ones, and in two of the three examples, the mean of the distribution is also closer to zero.

Pair-object Higgs-mass estimates

Having regressed the 3-momenta of all four decay products, we can now compute regressed masses for each of the two Higgs bosons. This is done by assuming the masses of the b quarks and the τ leptons and computing the invariant masses of the pairs using the regressed 3-momenta. The distributions of these masses are shown in Fig. 7.7 for both signal and background events, alongside the distributions computed using the base-level reconstructed final-states.

We can see that the regressed distributions in signal are closer to the Higgs mass value (125 GeV) than the reconstruction-only estimates, and have less than half the width. Unfortunately, we can also see that the regression causes the background distribution to move from being relatively flat to also peaking close to the Higgs mass, thus spoiling the discrimination between signal and background.

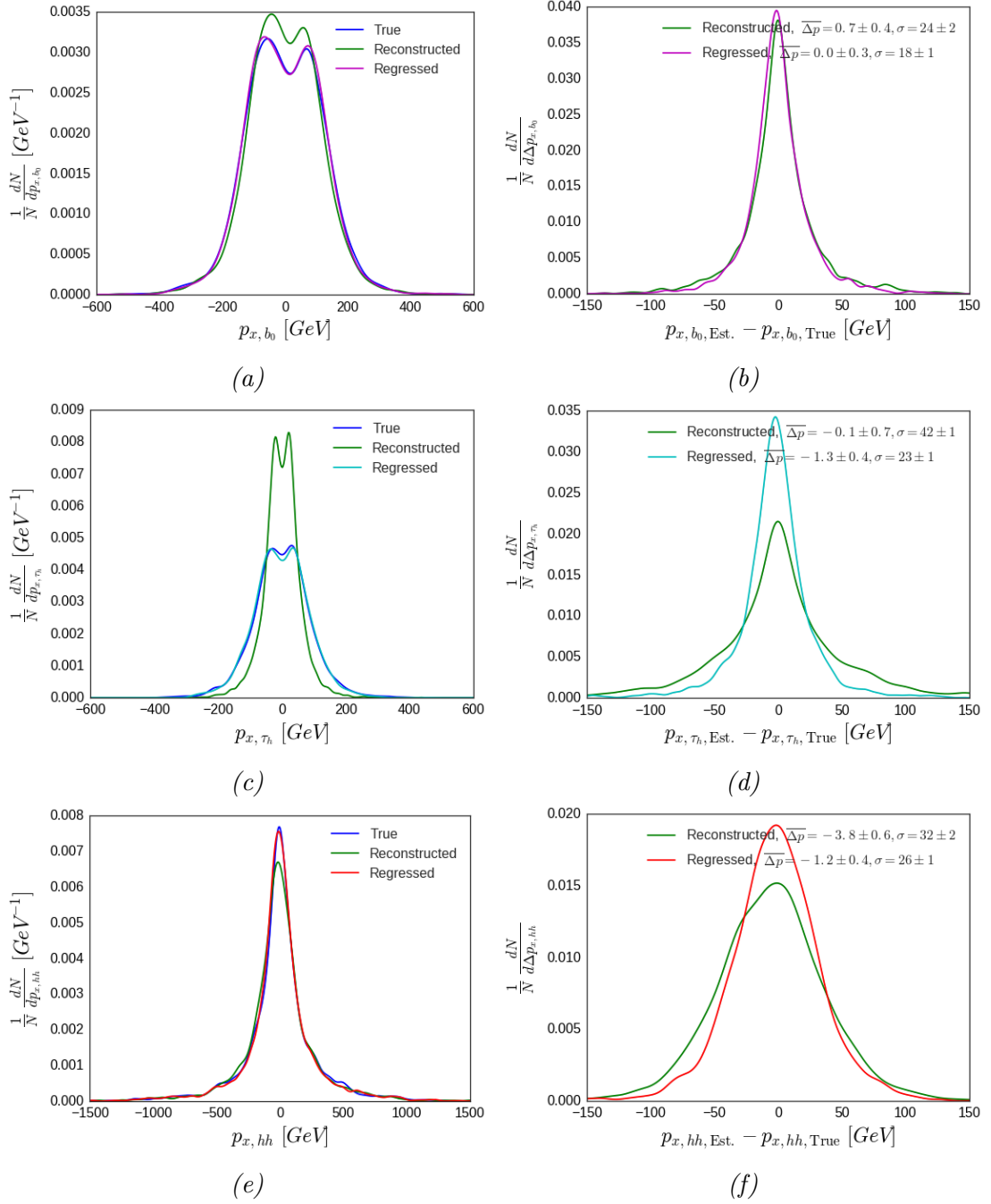


Figure 7.6: Examples of the performance of the pair-object regression in the $\tau^+ \tau^- b \bar{b}$ signal sample: hardest b-quark momentum (top), τ -lepton momentum (middle), di-Higgs-vector momenta (bottom). p_x estimations (after reconstruction and after reconstruction+regression), and true values are shown in the left column. Residual distributions of the estimations (after reconstruction and regression) are shown on the right column.

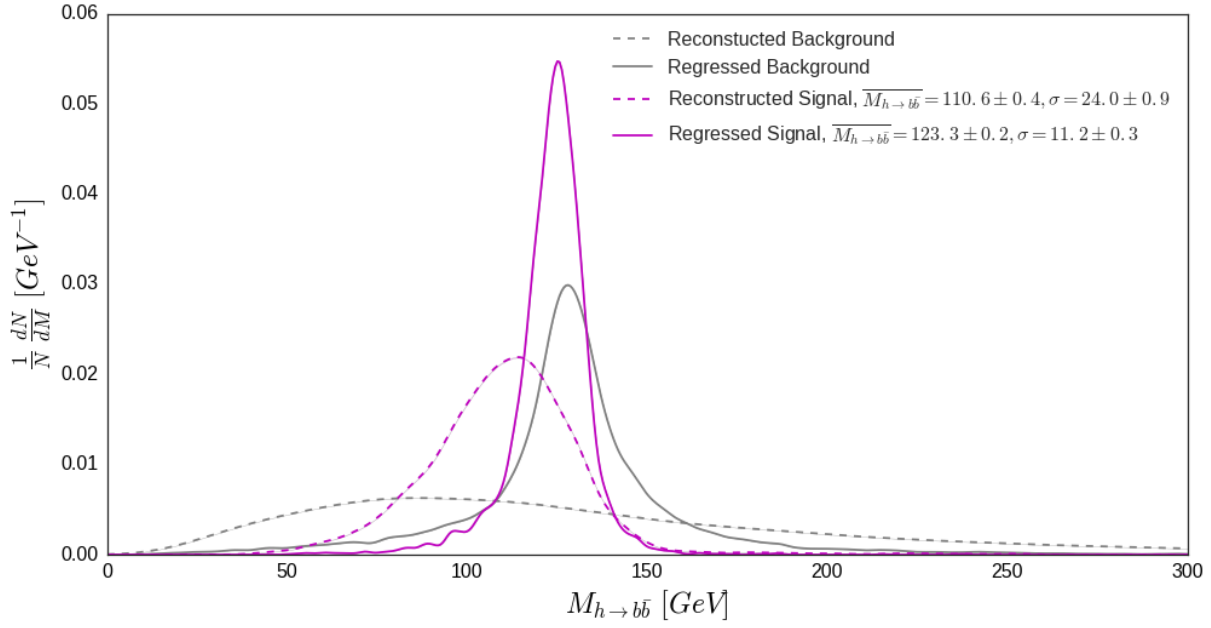
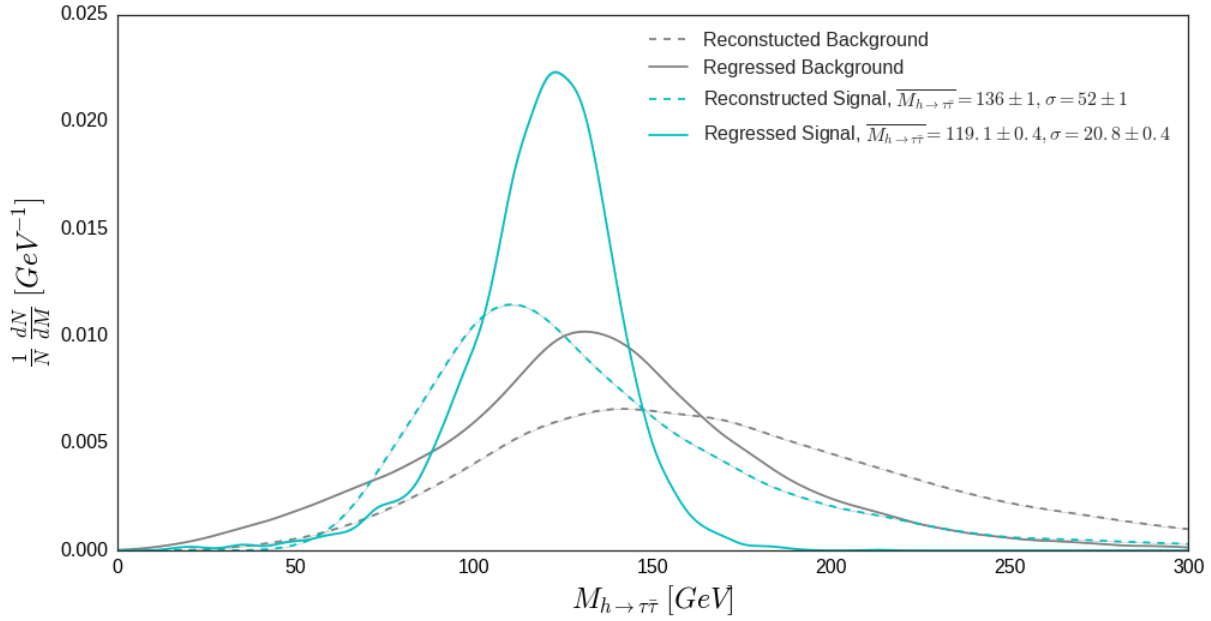
(a) Invariant-mass estimations of $b\bar{b}$.(b) Invariant-mass estimations of $\tau\bar{\tau}$

Figure 7.7: Invariant-mass estimations of $b\bar{b}$ (top) and $\tau\bar{\tau}$ (bottom) pairs from pair-object regression for both Higgs bosons for the $\tau^+ \tau^- b\bar{b}$ sample as well as for the corresponding distributions of the background sample ($t\bar{t}$).

7.4.2 Single-object regression

As mentioned in Sec. 7.1.4, the peaking in the regressed background distributions for the Higgs mass is thought to come from the regressor learning to compute the Higgs-boson mass, and using it to help better regress the object momenta in signal. This would result in the similar response when applied to background events. In order to prevent the formation of this peak in background, and thus retain or improve the separation between signal and background offered by the Higgs-boson masses, it is necessary to perform regression in such a way that the Higgs-boson mass cannot be computed by the regressor. This system involves regressing the momenta of each visible final-state individually, and removing features from the training if they lead to a peak in the background mass distribution. I refer to this approach as single-object (SO) regression. Once all four final-state objects are regressed, the regressed Higgs-boson masses may be computed as normal.

Features

A reduced selection of base-reconstruction level features are used for the SO regression. The reduced number of features acts to prevent the regressor from being able to determine the mass of the Higgs boson, by removing features related to the other object in each pair. ‘Hiding’ the constant Higgs mass from the regressor should prevent the background from peaking in the invariant mass distributions. For the case of the b_0 -jet (i.e. the leading b -jet) the features used are:

- 4-momentum, magnitude of 3-momentum of b_0 , τ_h , and τ_μ : 15 features,
- Mass of b_0 and τ_h : 2 features,
- Transverse mass of τ_μ : 1 feature,
- CSV values of b_0 : 1 feature,
- 4-momentum, magnitude of 3-momentum, and mass of $h_{\tau\tau}$: 6 features,
- Energy of h_{bb} : 1 feature,
- Energy, mass, and magnitude of 3-momentum of di-Higgs: 3 features,
- p_x and p_y of missing momentum: 2 features.

31 base-reconstruction level features are used for the single-object regression of the τ_h :

- 4-momentum, magnitude of 3-momentum, and mass of b_0 , b_1 , and τ_h : 18 features,
- Transverse mass of τ_μ : 1 feature,
- 4-momentum, magnitude of 3-momentum, and mass of h_{bb} : 6 features,

- Energy of $h_{\tau\tau}$: 1 feature,
- Energy, mass, and magnitude of 3-momentum of di-Higgs: 3 features,
- p_x and p_y of missing momentum: 2 features.

30 base-reconstruction level features are used for the single-object regression of the τ_μ , these are the same as those used for the regression of the τ_h , except that the 4-momentum and magnitude of 3-momentum of the τ_μ are used in place of those of the τ_h , and the mass of the τ_h is not used.

Performance

Figure 7.8 shows the mass distributions obtained from SO regression for both signal and background events. Comparing to the results obtained with the PO regression (Fig. 7.7), we can see the SO regression produces background distributions which are both wider and farther away from the Higgs-boson mass.

For the $h_{b\bar{b}}$ mass, the regression succeeds in producing a more accurate estimation, whilst precision (i.e. the standard deviation (σ) of the distribution) remains the same. However, we can see regression does still improve the estimation for the $h_{\tau^+\tau^-}$ mass, both for the mean and for the spread of the distribution. Most importantly, the regressed background distribution do not peak, and instead remain as flat as their corresponding reconstructed distributions, meaning that the discrimination offered by the mass features might have improved, even for the regressed $h_{b\bar{b}}$ mass.

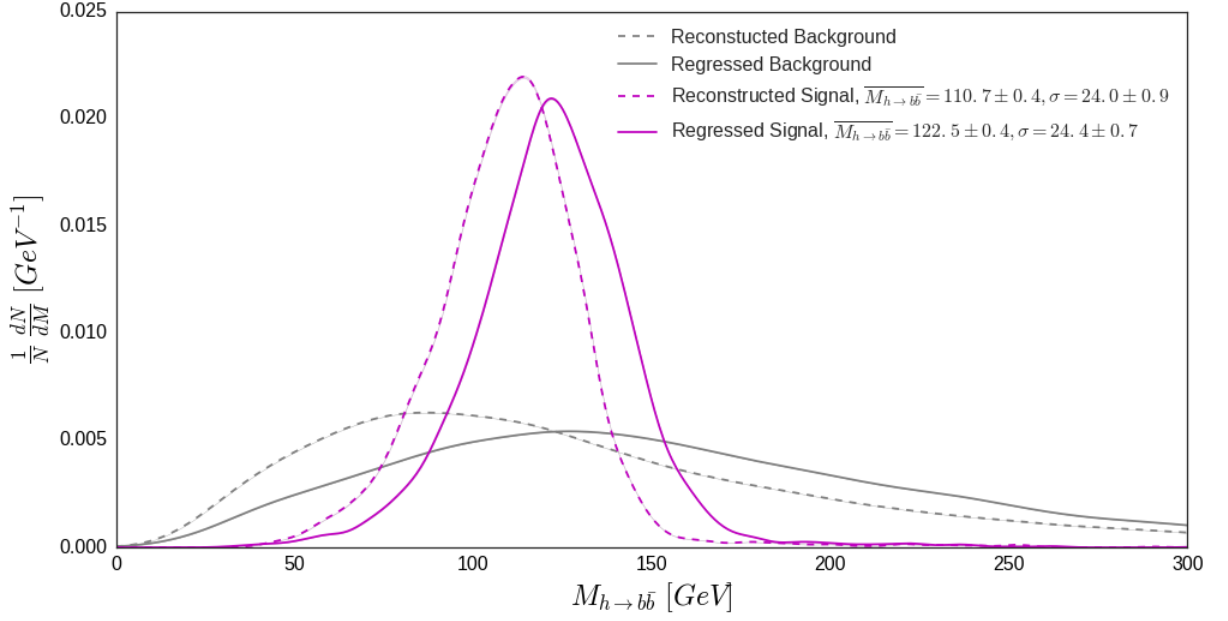
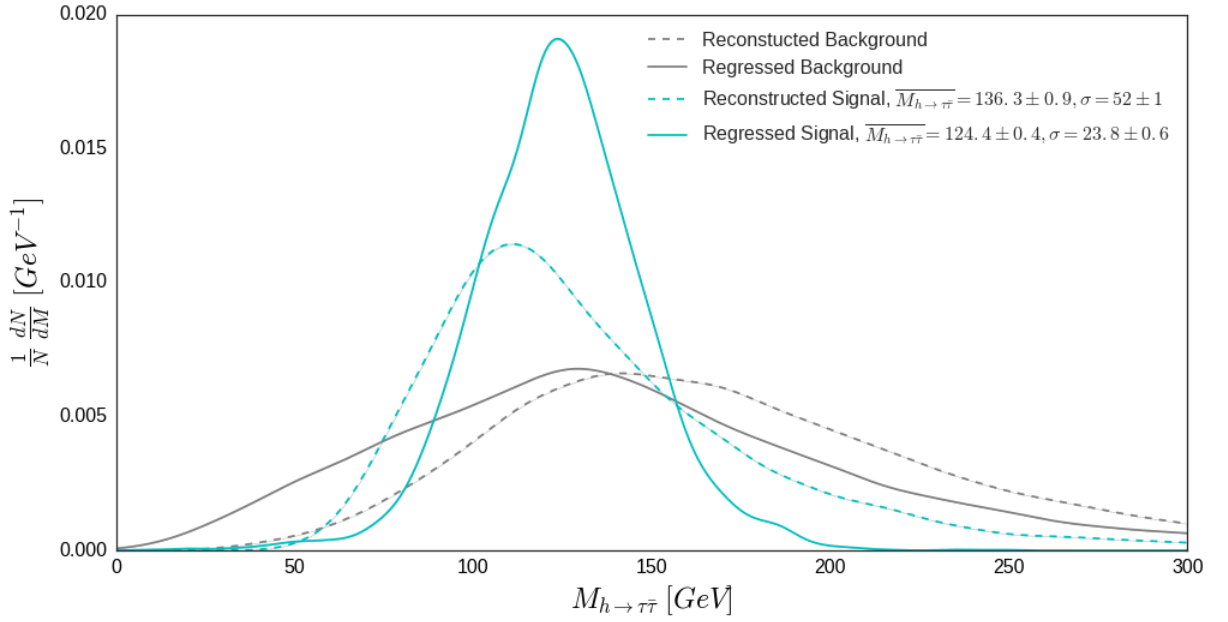
(a) Invariant-mass estimations of $b\bar{b}$.(b) Invariant-mass estimations of $\tau\bar{\tau}$

Figure 7.8: Invariant-mass estimations of $b\bar{b}$ (top) and $\tau\bar{\tau}$ (bottom) pairs from single-object regression for both Higgs bosons for the $\tau^+ \tau^- b\bar{b}$ sample as well as for the corresponding distributions of the background sample ($t\bar{t}$).

7.5 Di-Higgs mass regression

As mentioned in Sec. 7.1.4, regressors trained to reproduce non-delta distributions (i.e. the target value varies as a function of the inputs), rather than distributions which are delta functions (i.e. the target is a constant value), should produce different responses when applied to different samples (e.g. signal and background). Therefore, a regressor trained to output the invariant mass of the di-Higgs vector in signal should produce a different response when applied to background, since it is a non-delta distribution in signal. For this the same regression architecture as before will be used, with the single output being the mass. The regressor will also use PO regressed features where possible, since SO regression results in poorer momenta estimations and is only useful for regressing the individual Higgs-boson masses.

7.5.1 Features

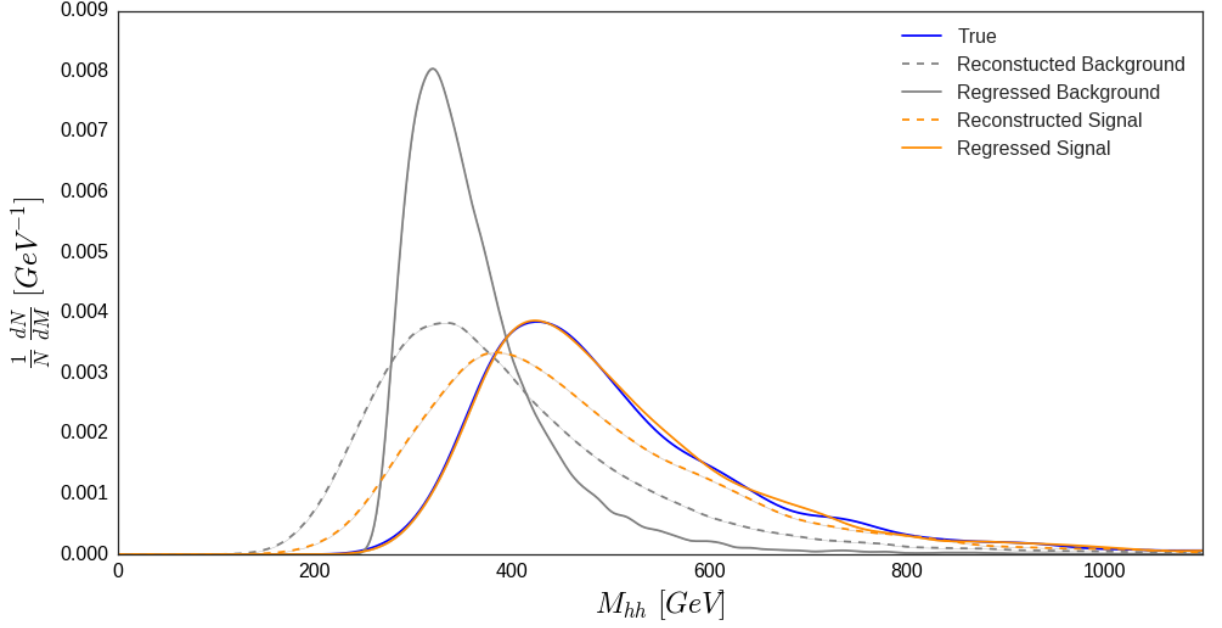
The following 45 features are used for the regression:

- Regressed 4-momentum, magnitude of 3-momentum, and mass of selected final-states: 24 features,
- Regressed 4-momentum, magnitude of 3-momentum, and mass of both Higgs and di-Higgs: 18 features,
- p_x and p_y of missing momentum: 2 features,
- Transverse mass of leptonic tau ($m_{T,\mu}$): 1 feature.

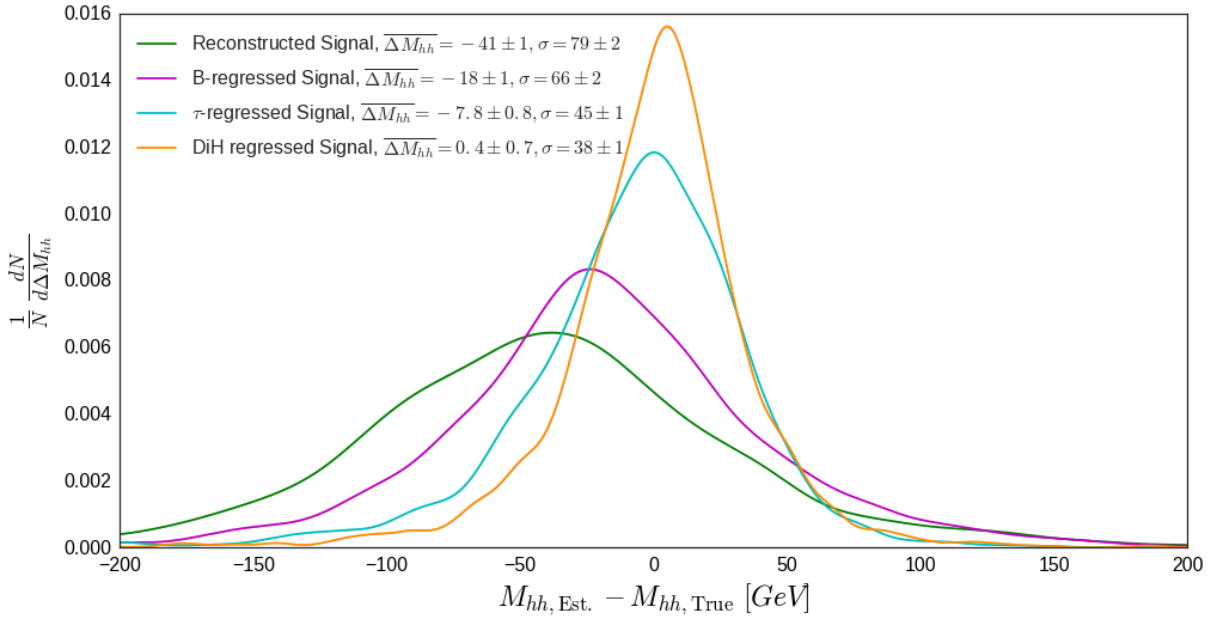
7.5.2 Performance

We can see from the results shown in Fig. 7.9a, that applying the regressor improves the estimation of the di-Higgs mass by correcting the overestimation of the low-mass region, which was presumably caused by mistreatment of the missing energy.

We can also see that the reconstructed distributions for signal and background are similar; however, when the regression is applied, the background accumulates in a region at a lower mass than the signal, thereby providing separation between signal and background. Residual distributions are shown in Fig. 7.9b, after reconstruction and after different steps of regression, i.e. b , τ , and di-Higgs.



(a) Di-Higgs-mass estimations.

(b) Residual distributions between estimated and true di-Higgs invariant mass values, after reconstruction and after different steps of regression (b, τ , and di-Higgs).Figure 7.9: Performance of Di-Higgs-mass regression on signal and background data ($t\bar{t}$).

7.6 Summary of regression studies

We can see from Fig. 7.8a that the SO regression method provides some improvement in determining the mean of the $h_{b\bar{b}}$ mass, and this is done without causing the background distribution to peak in the signal region. Furthermore, Fig. 7.8b shows that the regression method fixes the overestimation of the $h_{\tau^+\tau^-}$ mass distribution leading to large improvements in both mean and standard deviation, again without causing a background peak. Overall, it can be expected that the SO regression will improve classification power which can be achieved through the use of the Higgs-boson mass features.

As mentioned in Sec. 7.2.4, there is a CMS algorithm to estimate the $h_{\tau^+\tau^-}$ mass, called **SVFit** [108]. In Fig. 7.10 we can see that in terms of resolutions, the two approaches offer similar performance, and in neither case does the background peak. The regression approach developed here has a mean closer to the Higgs-boson mass, and it is quicker to evaluate; however, the training of the regressor will be specific to the signal phase-space, and porting the method to other analyses will require retraining. However, it is possible that with a more general training, a more widely applicable regressor could be constructed. In Chap. 8 I will examine which of the two methods offers the most advantage to a classifier.

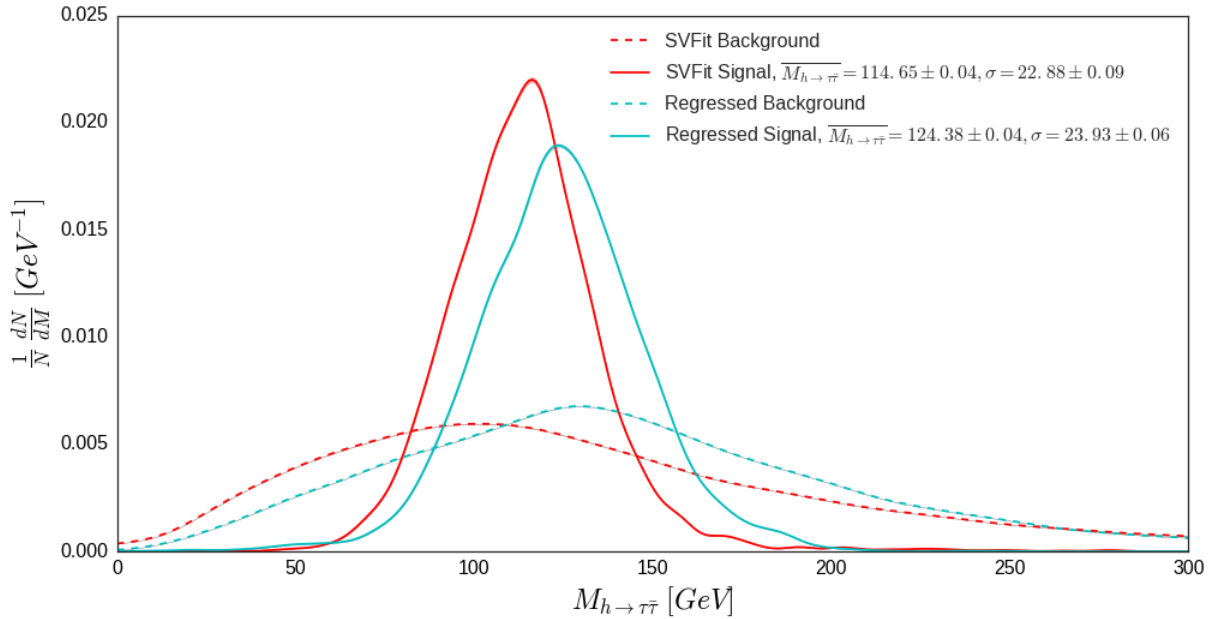
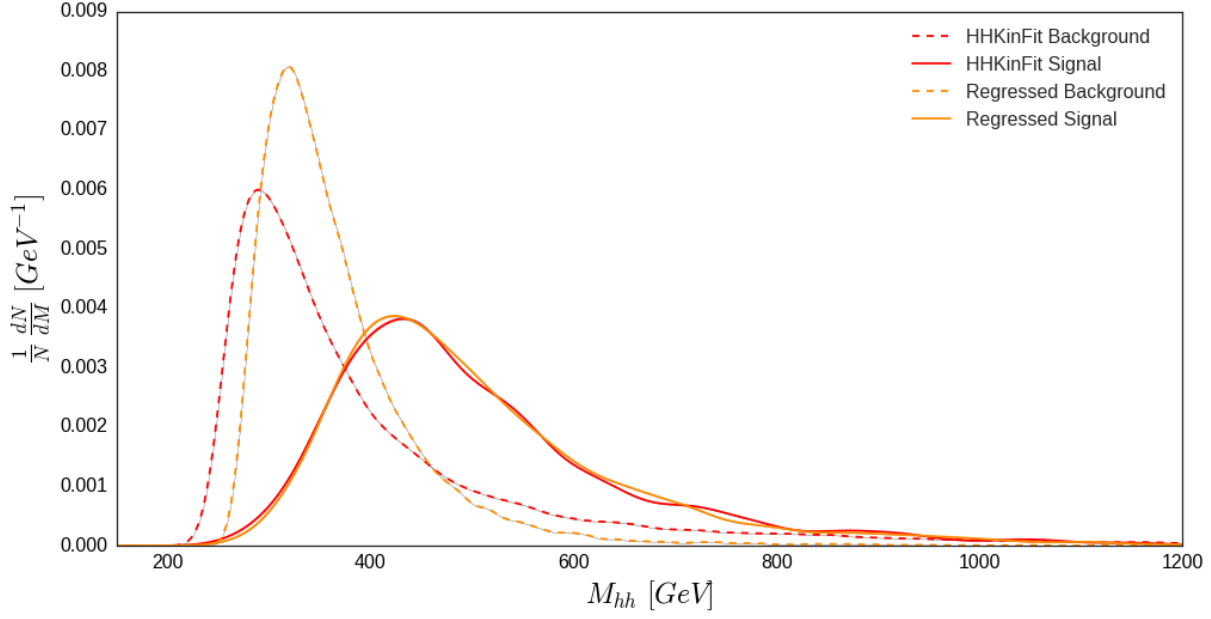
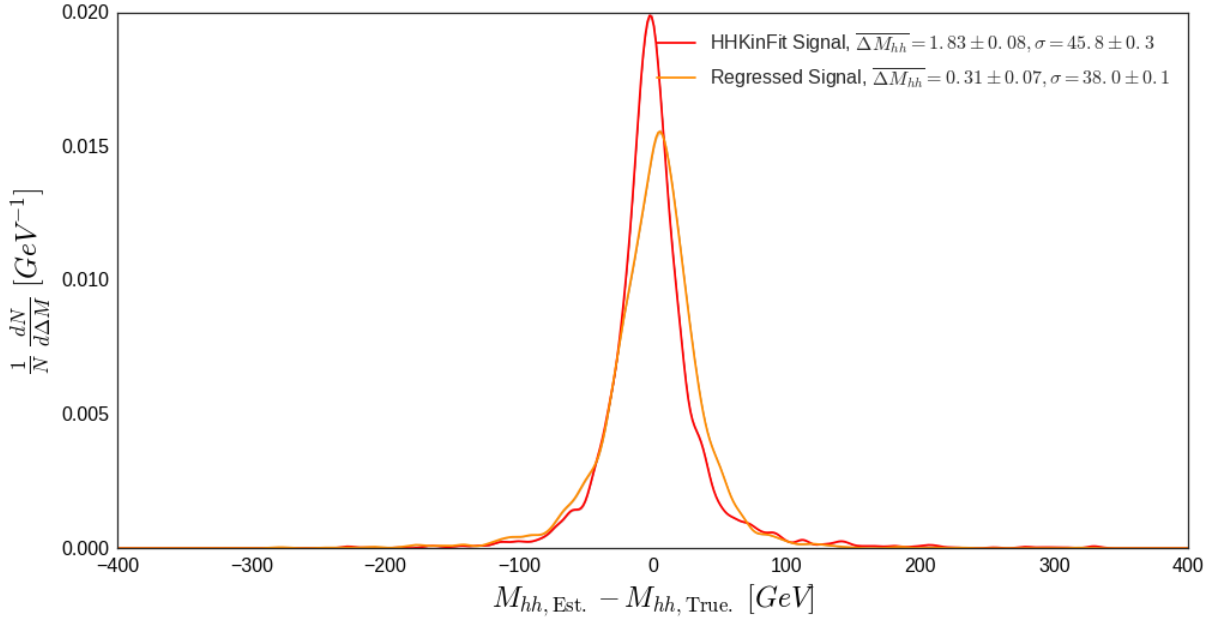


Figure 7.10: Comparison of the performances of **SVFit** and the regression methods developed in this thesis. Background only consists of $t\bar{t}$

Again, CMS has also developed a method to estimate the di-Higgs mass, called

`HHKinFit` [112]. The comparison of `HHKinFit` to the regression method is shown in Fig. 7.11, where we can see that both methods offer similar performance on the signal. `HHKinFit` does however occasionally severely overestimate the di-Higgs mass, hence why its standard deviation in Fig. 7.11b is larger, despite the peak being higher than the regressed distribution. Additionally, `HHKinFit` sometimes fails converge, whereas the regressor will always produce a valid output. `HHKinFit`'s distributions are only populated with events for which it converges, but the regressed distributions contain all events.

The background response is also slightly different for the two approaches: although both methods push the background to a low-mass peak, the `HHKinFit` peak is farther from the signal peak but contains a slowly decreasing tail; in the regression, the background peak is closer to the signal, but is sharper, and the tail decays more quickly. Similar to `SVFit`, `HHKinFit` is more widely applicable (and even to resonant hh production), while the regressor is likely to require analysis-specific training. Again, I will examine the impact of the two approaches on the overall classifier performance in Chap. 8.

(a) *Di-Higgs-mass estimations.*(b) *Residual distributions between estimated and true di-Higgs invariant mass values, after reconstruction and after different steps of regression (b, τ , and di-Higgs).*Figure 7.11: Comparison of the performance of **HHKinFit** and regression methods. Background only consists of $t\bar{t}$.

Chapter 8

Preliminary investigation: Classification in di-Higgs searches

8.1 Overview

8.1.1 The aim of event classification

As discussed in Chap. 6, discovery of di-Higgs production, or placing experimental limits on its production mechanisms, will be done using the statistical techniques described in Chap. 4. The sensitivity of these methods, and hence the power of the analysis, can be improved by finding regions of the data phase-space which are less-likely to be populated if the signal process does not exist in the collider data, and more likely to be populated if the signal process does exist.

In the previous chapter, we already examined the rates of production of the signal and main background process, and from Tab. 7.1 we can see that after the basic event selection, the ratio of signal to background is about 2×10^{-5} , and the background yield uncertainty is almost 10%, meaning that any contribution from the signal process will likely be indistinguishable from the background-only contributions.

This, however, is if we were to consider just the expected yield rates after event selection but in Fig. 7.9a and Fig. 7.8, we can already see that there are regions of the phase-space where the ratio of signal to background is more favourable. By *cutting* on these features (i.e. removing events which do not lie within certain regions of the feature), background events can be preferentially removed, thereby improving the signal-to-background ratio of the yield rates. Additionally, multiple features can be considered, e.g. background events in the region of the Higgs mass cannot be cut purely using the Higgs mass without also removing a significant portion of the signal events, however there may be other features in which these same events can be safely removed.

This process is generally referred to as a *cut-based analysis*. Since one is limited in terms of the dimensionality of probability density functions which may be easily perceived,

this kind of analysis has a high reliance on domain knowledge to propose a small subset of theoretically motivated features that provide decent separation between signal and background, either in isolation or when combined in pairs. Having optimised cuts on these (pairs of) features, one can then either perform the statistical analysis of the remaining yields (a *cut and count* analysis), or perform multiple yield measurements in bins of another feature in which the signal to background ratio varies (a *shape* analysis).

This process of cutting away events, however, can also reduce signal yield rate, and for very rare processes, such as di-Higgs production, even high-efficiency cuts can still remove enough signal to spoil the analysis. Additionally, fully optimising a cut-based approach would require considering all possible features and all possible cuts, as well as cuts which are parameterised by other features. Ideally one should consider all features of the data and attempt to find a projection in which the signal and background are most linearly separable. This univariate distribution can then be binned, or fitted to, in order to perform the statistical analysis without having to cut away data.

The advantage of machine learning is being able to do just this; to consider all features simultaneously and to reduce the dimensionality down to a single projection in which the classes are most separable: $y = f(\vec{x})$, in which the observable features of the event, x , are mapped by some function f to the class y (signal or background). Treating the problem as a binary classification task and using the approach in which one class is mapped to a zero, and the other to a one (see Sec. 5.1.2), the prediction of a trained classifier will fulfil our requirements. In the most extreme case, all of the signal will have a prediction of one, and all of the background, zero. However in reality since perfect separation is unlikely, the majority of the background will be close to zero, with a tail extending towards the signal region, and vice versa for signal.

8.1.2 Classification approaches

This chapter aims to check what kinds of approaches are viable for the classification of non-resonant di-Higgs production. I will initially test the use of the Matrix Element Method (MEM - a method of building a discriminator using the ME's for signal and background), and subsequently the use of a neural-network based classifier. I will use the same datasets as described in the previous section on regression (Chap. 7 - i.e. only the dominant background of $t\bar{t}$ and the $\mu\tau_h b\bar{b}$ decay channel are considered), and will also consider the benefit of using the regressed features over their more basic reconstructed counterparts. Additionally, I will begin to take steps towards selecting more optimal subsets of training features using specific feature-selection methods.

Result summary

Whilst MEM on its own is found to be unable to provide sufficient discrimination, the weights that are computed using it are found to be useful as inputs to a DNN-based

discriminator. Indeed the DNN-based discriminator is found to offer good performance in separating signal and background, and benefits from the regressed versions of the features. In checking the sensitivity of the analysis using the DNN, it is found that a shape analysis using the DNN prediction offers a greater expected significance than a cut-and-count analysis by cutting on the prediction.

8.2 Classification of di-Higgs signal with MEM

8.2.1 Overview

The Matrix Element Method (MEM) [111] aims at providing discrimination between two hypotheses (e.g. signal α , and background β) via the likelihood ratio:

$$\Lambda(x) = \frac{L(x|\alpha)}{L(x|\beta)}, \quad (8.1)$$

where $L(x|\alpha)$ is the likelihood to observe the experimental event x under hypothesis α . This likelihood can be calculated from particle theory by using the matrix element calculations and accounting for the effects of the particle detector and reconstruction. Specifically, the likelihood is calculated by evaluating the integral:

$$L(x|\alpha) = \frac{1}{\sigma_\alpha} \int \mathrm{d}\Phi(y) \, \mathrm{d}x_1 \mathrm{d}x_2 f(x_1) f(x_2) |\mathcal{M}_\alpha(y, x_1, x_2)|^2 W(x|y), \quad (8.2)$$

where: σ_α^{-1} is a normalisation factor based on the cross section of the process; $f(x_1) f(x_2)$ are the parton distribution functions for the incoming partons with momentum fractions x_1 and x_2 ; x is a set of momenta associated with the experimental event, y is the set of momenta corresponding generator-level event, and $W(x|y)$ is the transfer function from x to y , i.e. the probability for a partonic configuration y to evolve to reconstructed momenta x ; $|\mathcal{M}_\alpha(y, x_1, x_2)|$ is the theoretical matrix element (in this case calculated by the Monte Carlo generator); and $\mathrm{d}\Phi(y)$ is the phase-space measure, and corresponds to integrating over the whole phase space of production.

$L(x|\alpha)$ is henceforth referred to as the event weight under hypothesis α . In order to calculate these weights, the MOMEMTA [228] package is used. This package functions in a modular fashion and provides *blocks* to help simplify the integral via variable transformations and then run the integration. Since the hypotheses in the investigation are *background-only* and *signal-plus-background*, each event should have two weights, one computed under the signal hypothesis and the other under the background hypothesis. The discriminator in Eq. 8.1 then becomes:

$$\Lambda(x) = \frac{L(x|\text{Signal})}{L(x|\text{Signal}) + c \times L(x|\text{Bkg.})}, \quad (8.3)$$

where c accounts for the relative expected yield of signal and background, and is defined as $c = (\epsilon^{bkg.} \times \sigma^{bkg.}) / (\epsilon^{sig.} \times \sigma^{sig.})$, where ϵ and σ are the acceptance and the cross section, respectively, for signal (sig) and for background (bkg).

MADGRAPH [208] is used to compute the relevant matrix elements (m_α in Eq. 8.2), and $f(x_1)$ and $f(x_2)$ are sampled from the CT10n1o PDF [229] included via LHAPDF [230].

Computing weights via MEM for a given process hypothesis requires first computing the matrix element for that process, and then running the computation over each event. I will apply MEM to the datasets described in Sec. 7.2, however due to the necessary use of the regressors developed in Chap. 7 for the signal hypothesis, only the testing dataset for matched signal is used for the signal. The background events, however, are not used in training the regressors, so the entire dataset may be used.

8.2.2 Signal weights

In order to compute the signal weights, $L(x|\text{Signal})$, the outgoing particles of the matrix element must correspond to the 4-vectors of the reconstructed final-state particles being fed into the MEM configuration. The matrix element for $gg \rightarrow h h \rightarrow \tau^+ \tau^- b \bar{b}$ is generated and the regressors developed in Chap. 7 are then used to convert the reconstructed $\tau^+ \tau^-$ decay products back to generator level τ leptons. The regressors are also used to convert the b -jets back to generator level b quarks, thus avoiding the need for any transfer functions.

Figure 8.1 shows the distribution of the $-\log_{10}(w)$ of the weights (w) obtained for signal and background events under the signal hypothesis.

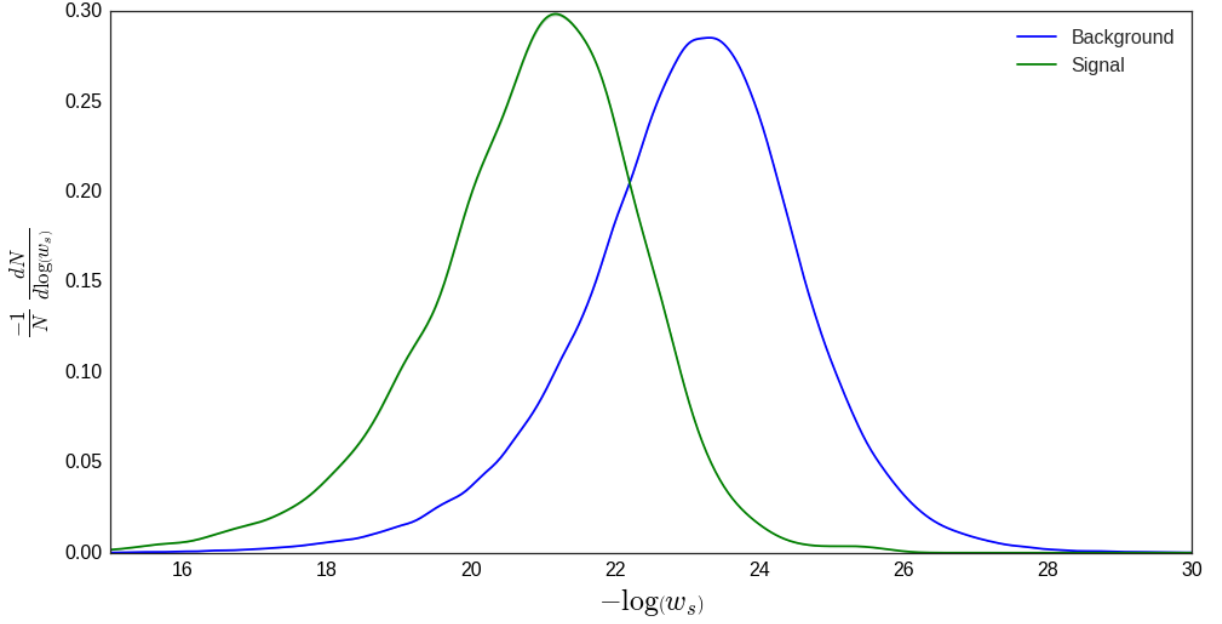


Figure 8.1: Distributions of the negative log of the MEM weights obtained for signal and background ($t\bar{t}$) under the hypothesis of signal: $gg \rightarrow hh \rightarrow \tau^+ \tau^- b\bar{b} \rightarrow \mu\tau_h b\bar{b}$.

8.2.3 Background weights

The inclusive $t\bar{t}$ background can contribute to the selected events in two ways: one involves the genuine production of a hadronic τ , such as $t\bar{t} \rightarrow b\bar{b}W^+W^- \rightarrow b\bar{b}\mu\bar{\nu}_\mu\tau^+\nu_\tau \rightarrow b\bar{b}\mu\bar{\nu}_\mu\tau_h\bar{\nu}_\tau\nu_\tau$; the second involves the hadronic decay of a W boson, where one of the resulting jets is mistagged as a τ_h , $t\bar{t} \rightarrow b\bar{b}W^+W^- \rightarrow b\bar{b}\mu\bar{\nu}_\mu q\bar{q}'$. The latter process, due to branching ratios and mistag rates, is expected to contribute approximately ten times less than the former and so emphasis is placed on the former contribution.

The matrix element for $t\bar{t} \rightarrow b\bar{b}W^+W^- \rightarrow b\bar{b}\mu\bar{\nu}_\mu\tau\nu_\tau$ is generated, and the τ is left undecayed. The reconstructed hadronic taus are then treated as corresponding exactly to the tau leptons, i.e. the energy of the ν_τ resulting from the decay of the τ is assumed to be small. To simplify the integral, the final states are assumed to be reconstructed with perfect angular resolution. This means that transfer functions are only required for the energies of the particles. For this preliminary investigation, these are approximated as Gaussian functions with resolutions of 10% for the b -jets and the τ_h , and 5% for the μ .

Permutations are made over the possible assignments of the b -jets and MOMEMTA is configured to account for the neutrinos and simplify the integration by performing a change of variable to integrate over the mass of the W bosons.

Figure 8.2 shows the distribution of the $-\log_{10}(w)$ of the weights (w) obtained for signal and background events under the background hypothesis.

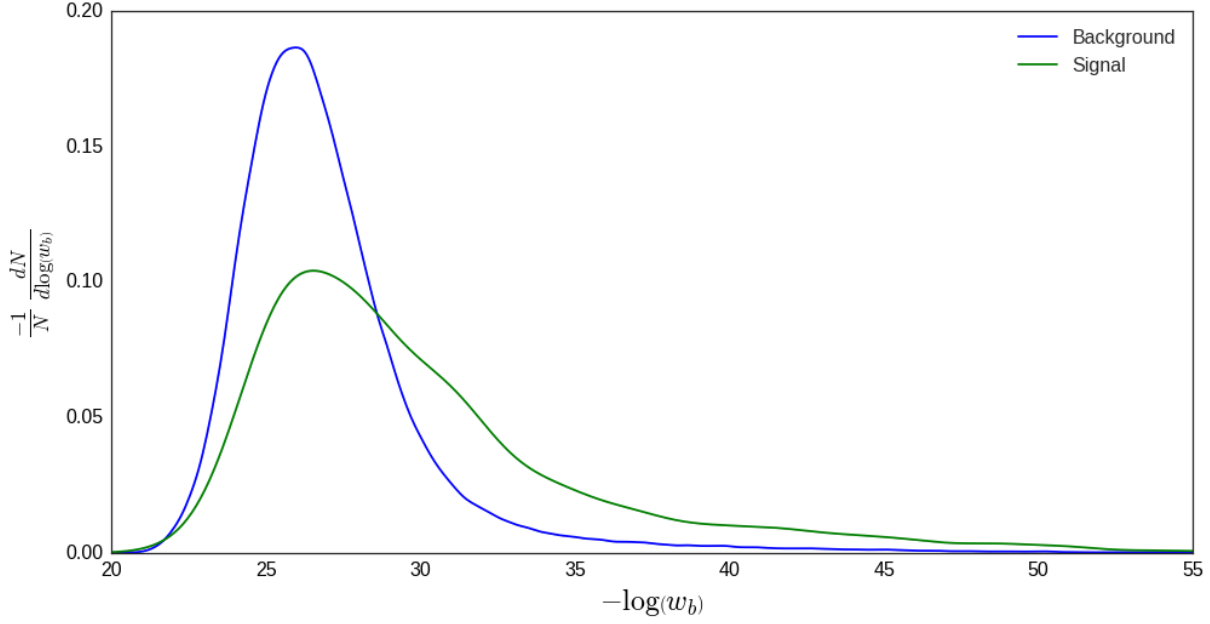


Figure 8.2: Distributions of the negative log of the MEM weights obtained for signal and background ($t\bar{t}$) events under the background-only hypothesis: $t\bar{t} \rightarrow b\bar{b}W^+W^- \rightarrow b\bar{b}\mu\nu_\mu\tau\nu_\tau$.

8.2.4 Discriminator construction

Having computed the weights under the signal and background hypothesis for each event, the discriminator $\Lambda(x)$ can be calculated according to Eq. 8.3.

By performing incrementally tighter cuts on the discriminator, the receiver operating characteristic (ROC) curve may be constructed as shown in Fig. 8.4. The ROC illustrates the trade-off between signal acceptance and background acceptance; effectively, it tells how well the classifier is able to discriminate between signal and background. The area under the ROC curve (ROC AUC) should be as close to one as possible; this would correspond to a perfect separation.

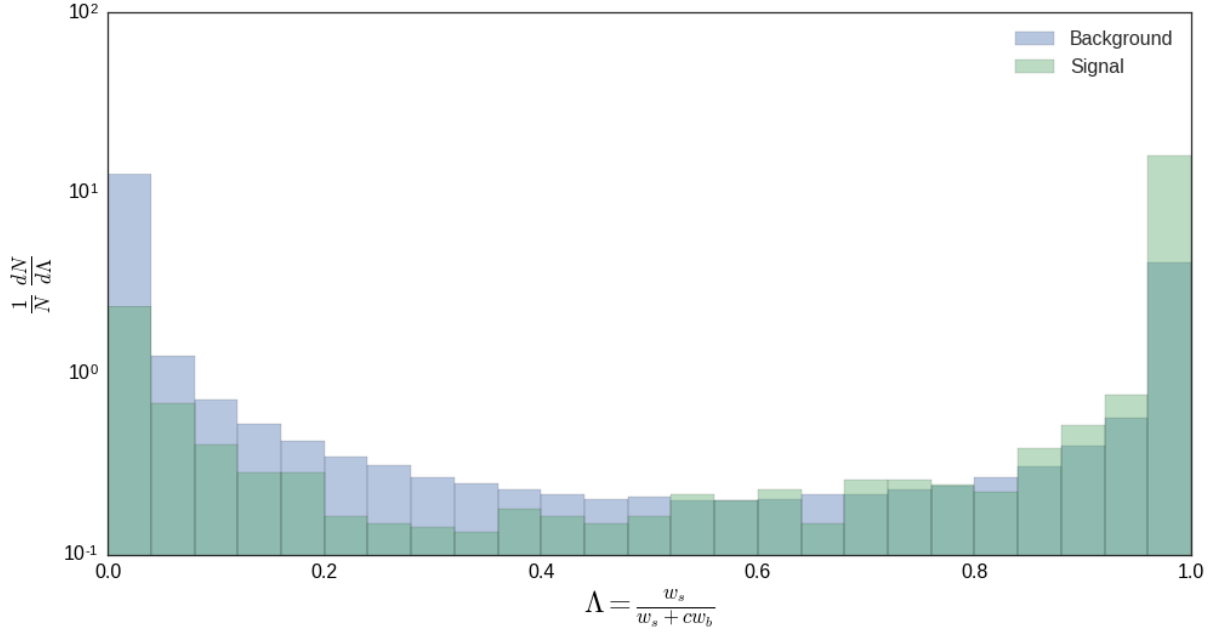


Figure 8.3: Distributions of Λ (Eq. 8.3), i.e. the ratio of the weights for the signal and signal-plus-background hypotheses (where background only consists of $t\bar{t}$ production).

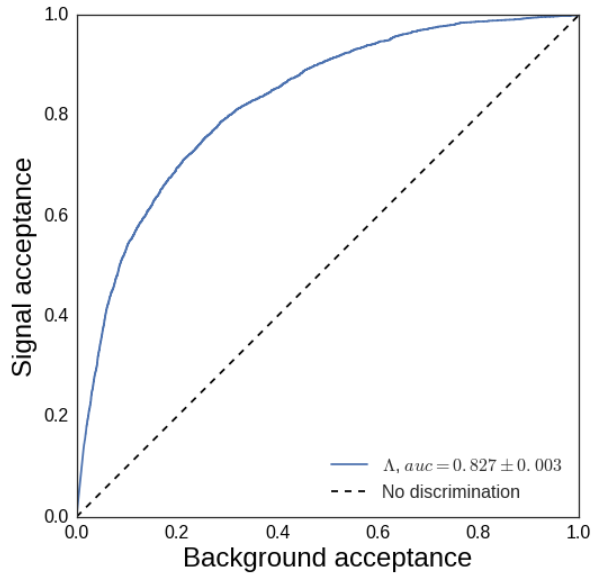


Figure 8.4: ROC curve for the Λ discriminator (Eq. 8.3). Lines closer to the top-left corner offer better signal-background separation.

8.2.5 Discussion

The separation shown in Fig. 8.3 is lower than desirable for an adequate signal-background separation performance in this analysis. This appears to be due to large portions of misclassified events. Several assumptions were made in the application of MOMEMTA, which could be improved with further work:

1. The transfer functions on the background are approximations and could be replaced with regression similar to the signal process;
2. The background can contribute in two ways; however, only the dominant method was accounted for;
3. The regression method is assumed to be 100% accurate in outputting the generator-level momenta; however, this is not the case.

Whilst the MEM on its own does not offer the desired performance, it is possible that the weights computed by MOMEMTA can be used as input features for a DNN classifier; I will examine this in Sec. 8.3.

8.3 Classification of di-Higgs signal with DNNs

8.3.1 Overview

In Chap. 7 I used NN-based regressors to improve the estimations of the kinematic properties of the particles in the signal data. The aim was to also increase the separation between signal and background in these distributions, in order to optimise the features for classifying signal and background. In this section, I will train dedicated classifiers on both the base-level reconstruction dataset and the optimised dataset in order to determine how well the samples may be separated, and whether or not the regression procedure really did indeed optimise the features.

8.3.2 The classifier

The network architecture, preprocessing, training, and ensembling, of the classifiers follows that used for the regressors (Sec. 7.3), with the exceptions that the NNs are trained on a mixture of both signal and background, the loss function is set to binary cross-entropy, and the output neuron uses a sigmoid activation function, since the outputs should be between zero (background) and one (signal). The same splitting of signal events into *training* and *testing* sets is made to avoid the possibility that the classifiers artificially show greater performance due to the regressors performing better on previously seen data-points. Also, during training, signal and background events were weighted such

that the summed weights of both classes were equal. This ensures that any imbalance in the sample sizes doesn't lead to a classifier which is biased towards signal or background identification.

8.3.3 Verification of feature optimisation

In order to determine whether or not the regressors developed in Chap. 7 improve the level of discrimination between signal and background, I will test on a reduced set of 38 features:

- 4-momenta, and magnitude of 3-momenta of selected final-states: 20 features,
- 4-momenta, magnitude of 3-momenta, and mass of both Higgs bosons and di-Higgs system: 18 features.

where the *Reconstructed* feature-set uses only reconstructed features, and the *Regressed* feature-set replaces the $h_{\tau^+\tau^-}$, $h_{b\bar{b}}$, and di-Higgs masses with their regressed version, as well as the di-Higgs momenta.

To compare the two features sets, I train classifiers via 10-fold stratified cross-validation on the training dataset and record the lowest loss achieved by each on the validation folds, and then compute the mean value. Table Tab. 8.1 reports the mean losses, along with the ensembled classifier performance on the training and testing datasets. From the results, we can see that the classifiers trained with the regressed features show a systematic improvement over those trained only on base-level reconstructed features.

Feature set	Mean ROC AUC	Mean loss
Reconstructed	0.9741 ± 0.0002	0.152 ± 0.004
Regressed	0.9829 ± 0.0005	0.125 ± 0.003
Improvement	0.9%	17.7%

Feature set	Training ROC AUC	Testing ROC AUC
Reconstructed	0.9926 ± 0.0002	0.9826 ± 0.0008
Regressed	0.9952 ± 0.0001	0.9849 ± 0.0007
Improvement	0.3%	0.2%

Table 8.1: Comparison of the classifiers trained on the base-level reconstructed features and the regressed ones. The mean values are computed over the 10 classifiers trained during cross-validation (i.e. single model performance). The Training and Testing results are for the resulting ensembles of the 10 classifiers. Loss is the binary cross-entropy of predictions.

8.3.4 Feature selection

In Sec. 7.2.4, a large number of features were defined, however only a subset of these will carry a significant discriminating power. All of the features defined in Sec. 7.2.4 were considered, along with the MEM weights computed in item 8.2.5. After testing several different combinations, the following set of 47 features was found to offer the greatest performance for the fewest number of features:

- 4-momenta, and magnitude of 3-momenta of selected final-states: 20 features,
- Mass of both b -jets and the τ_h : 3 features,
- CSV of both b -jets: 2 features,
- 4-momenta, magnitude of 3-momenta, and regressed mass of both Higgs bosons: 12 features,
- Regressed 4-momentum, magnitude of 3-momentum, and mass of di-Higgs: 6 features
- p_x , p_y , and magnitude of missing momentum: 3 features,
- Transverse mass of leptonic tau decay (m_T): 1 feature.

However, since manually checking the effect of the addition or removal of features is a time consuming process, not all possible combinations were tested. Instead, an alternative approach used in which a decision-tree ensemble was trained on all of the features and then rank them in order of importance based on how often each feature was used to split the data. Doing so and then removing all features with zero importance (i.e. the features which were never used) resulted in a reduced set of 38 features:

- The mass and CSV of both b -jets, and the E and $|p|$ of b_1 : 6 features,
- The energy of the muon: 1 feature,
- $p_x, p_y, |p|, m, E$ of the τ_h : 5 features,
- $p_x, p_y, |p|, E$, and regressed mass of both Higgs bosons: 10 features,
- Regressed m, E, p_x, p_y and $|p|$ of di-Higgs: 5 features,
- The `HHKinFit` mass of the di-Higgs: 1 feature,
- $|p|$ and p_x of missing momentum: 2 features,
- The $-\log$ of the `MOMEMTA` weights under signal and background, their sum, and their ratio: 4 features,
- H_T, m_T, s_T , and centrality: 4 features.

This set was found to outperform the larger, manually selected, feature set. It is interesting to note the automatic feature set does not use any z components of momenta, does find the `MOMEMTA` weights to be useful, and finds it is best to include the di-Higgs mass as computed by both regression and `HHKinFit`. This process of automatically selecting features is something I will expand upon and improve later in Chap. 10.

8.3.5 Performance

The classifier ensemble using the automatically selected feature set was trained on the training dataset via 10-fold stratified cross-validation. The distributions of the predicted class for signal and background events in the testing dataset are shown in Fig. 8.5. The corresponding ROC curves for the ensemble on the training and testing datasets are shown in Fig. 8.6. From these we can see that signal and background distributions are clustered towards one and zero, respectively, indicating that the classifier is able to separate the classes of events. Comparing the performance to that of the matrix-element method alone (Fig. 8.4), we can see that the DNN classifier is able to better utilise the MEM weights to achieve better performance.

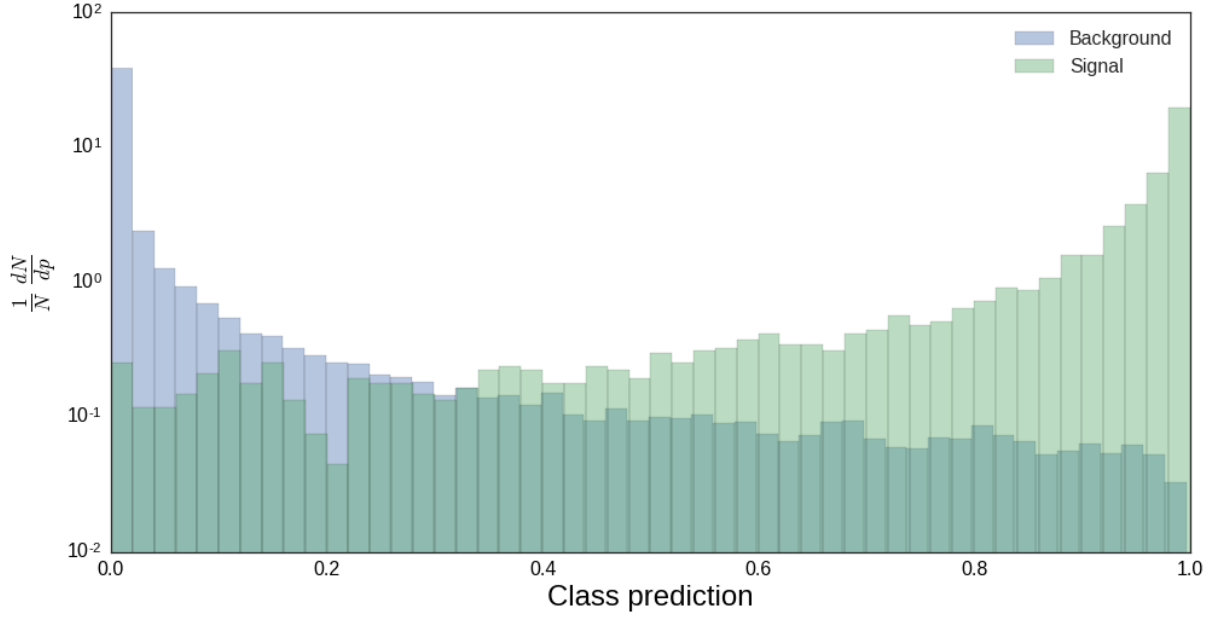


Figure 8.5: Class predictions of the classifier on testing data. 0 indicates ‘background-like’ and 1 indicates ‘signal-like’. Both signal and background ($t\bar{t}$) are normalised to unit area.

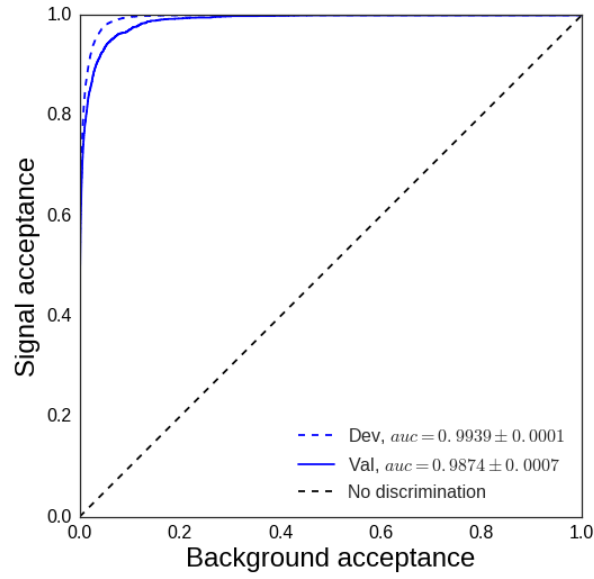


Figure 8.6: ROC curves for the classifier on the training and testing datasets. Lines closer to the top left-hand corner are better.

8.3.6 Summary of classification studies

From the studies performed in this Section, it seems that NNs are highly applicable to the separation of signal ($h h \rightarrow \tau^+ \tau^- b \bar{b}$) and background (inclusive $t \bar{t}$) events in the $\mu \tau_h b b$ final state.

I also found that the feature optimisation performed in Chap. 7 led to genuine improvements in the discrimination between signal and background. It is important to remember that NNs function by learning their own high-level trends in the data, so the fact that the ‘optimised’ dataset leads to an increase in performance despite the features having the same basis, means that the optimisation procedure really does fold in extra information about the event, which would otherwise be unavailable to the networks.

8.4 Inference of di-Higgs signal

In Sec. 8.3 I studied the performance of NN-based classifiers in terms of how well they can classify signal and background events. In the simulated sample we know whether an event is signal or background, however the data collected at the LHC is ‘unlabelled’, and might possibly be a mixture of signal and background events, with no way of saying for sure whether an event really is signal or background. As discussed in Chap. 4, what we want to do is to determine, for a given sample of collider data, what the likelihood is that the sample contains signal or whether it can be accounted for by only background processes. If the presence of signal in the sample is not deemed to be *significant*, then it can be used to place an upper limit on the rate of production of the signal process.

As discussed in Sec. 8.1.1, there are two ways in which the classifiers may be used for the hypothesis test. First, a signal-enriched region may be defined by requiring events to have a predicted class value greater than some cut-off point - a *cut and count* approach; alternatively, the density distribution of the predicted classes of events may be used - a *shape analysis*. In this section I will examine the use of both approaches.

The data used for the hypothesis tests is the testing sample used for evaluating the regressors and classifiers developed in the earlier sections. This ensures that the expected performance is not biased by the NNs having been trained on the data.

8.4.1 Cut and count

The signal significance can be approximated as $s/\sqrt{b}$, where s and b are the expected number of signal and background events, respectively. The expected rate of signal and background events, however, are not precisely known and are subject to systematic uncertainties. As discussed in Sec. 4.4, a more complex approximation, which accounts for the systematic uncertainty on the background production rate, is the Approximate Median

Uncertainty	Signal	Background
Luminosity	2.6%	2.6%
Lepton trigger and reconstruction	2%	2%
τ energy scale	3%	3%
Jet energy scale	2%	4%
b -tag efficiency	6%	6%
MC cross-section	8%	8%
Quadrature sum	11%	12%

Table 8.2: Systematic uncertainties affecting the normalisation of signal and background contributions (as used in Ref. [213]).

Significance (AMS):

$$\text{AMS} = \sqrt{2(s+b) \ln \left(\frac{(s+b)(b+\sigma_b^2)}{b^2 + (s+b)\sigma_b^2} \right) - \frac{b^2}{\sigma_b^2} \ln \left(1 + \frac{\sigma_b^2 s}{b(b+\sigma_b^2)} \right)}, \quad (8.4)$$

where σ_b is the uncertainty on the number of expected background events [142]. The AMS can be used as an intermediate metric to optimise the cut placed on the class prediction.

The signal region is defined by scanning across the possible cut values, and finding the cut for which the value of Eq. 8.4 is greatest. The cut choice is also subject to the requirement that the statistical uncertainties on signal and background (due to the limited number of MC events) are each less than 20%. This is done for a range of integrated luminosities based on those obtained, and expected to be obtained, at the LHC: 35.9 fb, 100 fb, 150 fb, and 500 fb.

In order to give a realistic idea of how systematic uncertainties are likely to affect the signal and background production rate estimations, in Tab. 8.2 I adopt the same values as those used in a similar CMS investigation documented in Ref. [213]. The value of σ_b used in Eq. 8.4 is therefore 12%. The optimum cut is found to be at 0.974 for all luminosities and Tab. 8.3 details the number of signal and background events which pass this cut.

The systematic uncertainties are then modelled via a profile-likelihood fit assuming log-normal distributions. The COMBINE tool [231] is used to perform the fits and hypothesis tests, with values for signal significance being calculated in the asymptotic limit [139] and the expected upper-limits on signal production being calculated using the fully frequentist method (LHC-style CL_s -limits) recommended by the LHC Higgs Combination Group. The cut-and-count results are reported in Tab. 8.4.

Integrated luminosity [fb^{-1}]	Signal yield	Background yield	AMS
35.9	0.250 ± 0.006	43 ± 8	0.031
100	0.70 ± 0.02	120 ± 20	0.040
150	1.04 ± 0.03	180 ± 30	0.042
500	3.48 ± 0.09	600 ± 100	0.047

Table 8.3: Expected number of signal (non-resonant $h h \rightarrow \tau^+ \tau^- b \bar{b} \rightarrow \mu \tau_h b b$) and background ($t \bar{t}$) events passing a classifier cut of 0.974 with associated uncertainties (statistical only), and the predicted significance (AMS) according to Eq. 8.4, for different integrated luminosities.

Integrated luminosity [fb^{-1}]	Expected significance	$\frac{\text{Upper-limit}}{\sigma_{\text{SM}}}$ @ 95% CL_s
35.9	0.019	91
100	0.023	74
150	0.024	73
500	0.027	69

Table 8.4: Expected signal significances and expected 95% CL_s upper limits on signal cross-section as obtained by the cut-and-count approach, for different integrated luminosities.

8.4.2 Shape analysis

As mentioned in Sec. 4.1.1, instead of cutting on the classifier output, the alternative approach is to use the density distributions of signal and background events (their *shapes*) to evaluate the presence of signal in the observed data. Here, I parametrise the density using histograms of varying bin width in order to best capture the shape of the distributions whilst not causing the statistical uncertainties in each bin to become too large.

Signal and background MC data are binned such that in a given bin, the statistical uncertainty on the population of each signal and background sample is less than 15%, unless the statistical uncertainty of either signal or background is less than 5%, in which case the statistical uncertainty of the other distribution may reach up to 25%. This results in the distribution shown in Fig. 8.7 - containing 28 bins - which shows the classifier prediction for signal and background.

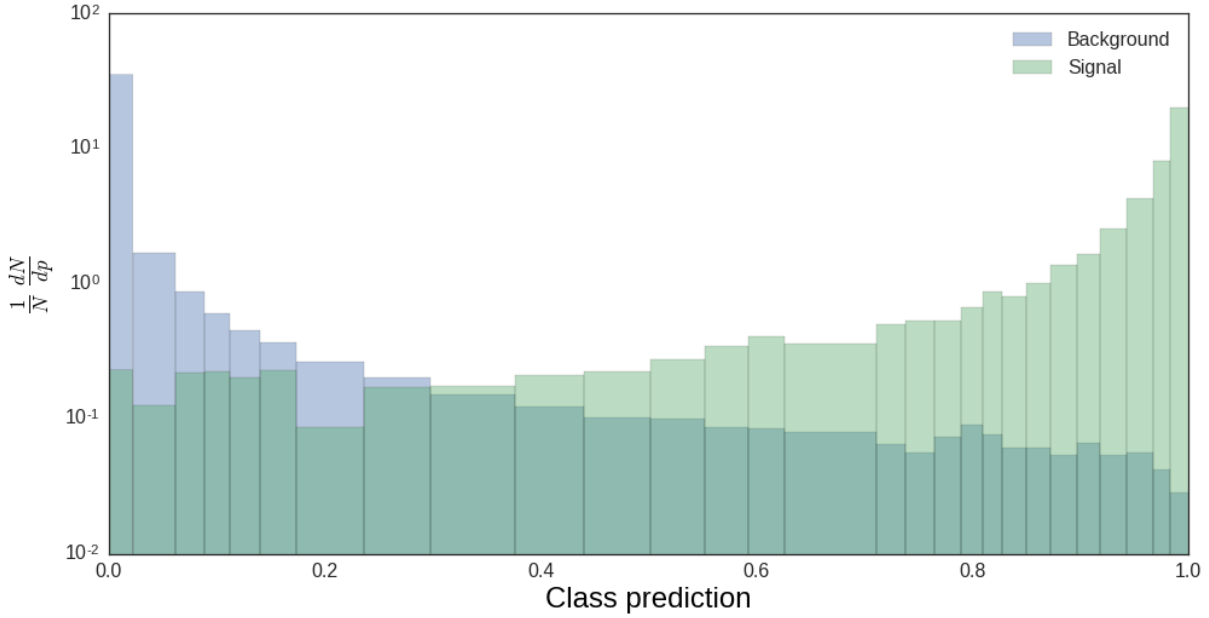


Figure 8.7: Class predictions of the classifier on testing data: 0 indicates ‘background-like’, and 1 indicates ‘signal-like’ events. Both signal and background ($t\bar{t}$) are normalised to one.

Values for signal significance and expected upper limit on signal production, again calculated using COMBINE with the asymptotic approximation and the frequentist method, respectively, are reported in Tab. 8.5. Comparing these results to those of the cut and count method (Tab. 8.4), we can see that the shape analysis is the more preferable approach due to consistently providing higher expected significances and lower expected limits.

Integrated luminosity [fb^{-1}]	Expected significance	$\frac{\text{Upper-limit}}{\sigma_{\text{SM}}}$ @ 95% CL_s
35.9	0.022	81
100	0.026	68
150	0.027	67
500	0.028	63

Table 8.5: Expected signal significances and expected 95% CL_s upper limits on signal cross-section as obtained by the shape analysis approach, for different integrated luminosities.

8.4.3 Discussion

Reference [213] reports that the most recent CMS study of the di-Higgs signal process sets an expected upper limit on the production cross section of 25 times the SM rate for a luminosity of 35.9 fb^{-1} , using nine separate channels and considering all sources of background. In this preliminary investigation I instead only studied one of the channels ($\mu \tau_h$). Whilst the $\tau_h \tau_h$ channel is expected to be the most sensitive channel for this search, the $\mu \tau_h$ channel provides a higher acceptance rate, which combined with the limited MC datasets better allowed me to study the usage of DNN classifiers in this search.

8.5 Conclusion

Deep neural networks were found to be highly applicable as event-level classifiers for signal and background, in the search for non-resonant di-Higgs production, achieving very strong levels of separation between the classes with a ROC AUC of 0.99. Additionally, the regressors developed in Chap. 7 were found to optimise the input features to the classifiers. The matrix-element method was also tested, but was not able to provide sufficient discrimination on its own. MEM was, however, found to provide features which were useful as an input to the DNN classifiers.

8.6 Retrospective thoughts on the preliminary investigations

The past two chapters have documented my preliminary investigation into the usage of neural networks for classification and regression in the search for SM non-resonant di-Higgs production $\tau_h \tau_h b b$ decay channel. This was an important step since it was my first experience with working with these kinds of algorithms in a HEP analysis. This work was performed separate to the official analysis, and so afforded me greater freedom to

explore and learn, but perhaps at the expense of becoming a contributor to the official analysis later than might have been optimal. This is not to say that I was not in contact with the CMS subgroup; indeed I gave summary presentations during working meetings and workshops, wrote internal analysis notes, and was generally known to be working towards merging with the official analysis. Given that this research accounted for the first year and a half of my PhD, and was driven by deadlines as part of my contributions to AMVA4NewPhysics, it is difficult to see how I could have been involved with the CMS search earlier.

Aside from becoming familiar with using neural networks, this investigation allowed me to explore and test different solutions and approaches. Whilst the MEM was found to provide useful inputs to a DNN classifier, it nonetheless is quite complicated and time-consuming to apply fully, as one needs to account for all possible ways in which the backgrounds may contribute and compute individual weights under each possible hypothesis. Therefore moving towards a full analysis and considering all the background processes, I will drop the use of the MEM.

The regressor system developed showed similar performance to the existing CMS algorithms, `SVFit` and `HHKinFit`, whilst being quicker to compute, and provided regression for b -jet pairs, for which, at the time, there was no commonly used CMS algorithm (although Ref. [232] used a BDT regressor). The current disadvantage is that the regression system should ideally be retrained for the specific analysis on which one wishes to apply it. Whilst this is not a hard problem, I went on to find that when working in a collaboration, the generator-level information required to train the regressors is often not included in the skimmed data files, or filtered out during generation to reduce file-size. Knowledge that the generator-level information is required must therefore be present right at the beginning of the analysis.

There are a few ways in which the regression system could be generalised such that a pretrained model could be applied to a new analysis without further training. In these cases it is necessary to adapt the networks to only use inputs related to the relevant decay-system, and to make the regression work for a range of resonant masses, rather than just the 125 GeV Higgs boson, or non-resonant di-Higgs system, as mine did.

Such an approach of training on multiple mass-points for the $h_{b\bar{b}}$ and $h_{\tau^+\tau^-}$ regressors was discussed during the preliminary investigation, however at that point our priority was only the SM Higgs boson, and generating samples for that took time enough on its own. Eventually, I did go on to start work on a $h_{\tau^+\tau^-}$ regressor trained using a range of mass-points, however upcoming deliverable deadlines meant that I was unable to pursue the development further. The fact that the $h_{\tau^+\tau^-}$ regressor developed here for the SM Higgs boson should form an upper bound in performance, and that this was consistent with `SVFit`, meant that the more general case was not a priority. I would like to revisit this, though, given my further experience in DNN training.

Whilst `SVFit` performs well, its main disadvantage is the evaluation time per event,

which can quickly become large if the same event must be evaluated multiple times to compute different systematic variations. Another approach could be to regress to the `SVFit` prediction for a range of input data. Whilst the mass resolution would not improve, and would probably degrade slightly, the DNN based version would be far quicker to evaluate, and may allow the use of a mass-resolution-improvement algorithm in analyses in which `SVFit` was previously unusable due to data-processing time-constraints.

In the meantime, since this preliminary investigation, another approach for b -jet regression has been developed in Ref. [233]. A DNN is used to predict the energy correction and energy resolution of b jets. Whilst this doesn't provide a refined momentum vector, during my investigations I had found that an accurate estimate of the jet energy alone was sufficient in providing a much improved invariant-mass resolution. The resolution predictions, too, are useful, potentially allowing a search to categorise not just by jet ID, but also by energy resolution. For the official CMS $hh \rightarrow \tau^+ \tau^- b\bar{b}$ search, however, b -jet regression was not identified as a priority task and is not used in the search documented in Chap. 10.

In Sec. 8.3.4 a method for automated feature selection was discussed. This concept is slightly improved in Sec. 9.4.2, and further built upon in Sec. 10.7, becoming a robust method for filtering down a large menu of feature to just the most useful ones.

Chapter 9

Optimisation studies: Advanced methods in deep learning

9.1 Overview

As mentioned in Sec. 1.2, modern techniques are being used in HEP applications, however these techniques are normally developed outside of HEP, and are then adapted for use in our domain-specific solutions. These techniques are continually being refreshed and updated, and are normally presented on benchmark datasets for some specific task, such as image recognition on ImageNet [234]. However, it is not always obvious whether improvements on such datasets would be reflected by similar successes when applied in other domains. I consider this a strong motivation to study how large an improvement these new techniques can bring. To do so I will use a very well-studied, HEP-specific benchmark dataset: that of the Higgs ML Kaggle challenge [235].

In this chapter I examine the benefits and cross-domain applicability of several recent advances in deep-learning, including: a method to quickly optimise the learning rate; newer activation functions; learning rate scheduling; data augmentation; and alternative ensembling techniques. Following these experiments I present a solution which is able to achieve consistently the same performance as the winning solution, whilst being significantly quicker to train and apply, and being suitable for use with both GPU and CPU hardware setups. These reductions in timing and hardware requirements potentially allow the use of more powerful algorithms in HEP analyses, where models must be retrained frequently, sometimes at short notice, by small groups of researchers with limited hardware resources.

I begin with a more detailed description of both the general problem (Sec. 1.2) and the specific challenge (Sec. 9.3). In Sec. 9.4 I describe the baseline model and the various improvements, reporting performances in terms of a range of optimisation metrics. In Sec. 9.5 I report the final performance on the testing data, expanding to a fuller com-

parison with the winning solutions. Section 9.6.1 studies the individual benefits of each method on the final model. The investigation and the main results are summarised in Sec. 9.7.

Following this investigation I describe in greater detail a software package, LUMIN [236], I developed and released which allows researchers to easily apply these state-of-the-art techniques in their own analyses (Sec. 9.8). In Sec. 9.9 I report an addendum study in which some of these results were reconfirmed in a separate analysis, which ran projections for the sensitivity of the HL-LHC to the search for di-Higgs production.

This study was actually run twice: the first time was to fulfil a of AMVA4NewPhysics (Deliverable 1.4), available in Ref. [43]; I then revisited the study using during the development of LUMIN, working in greater detail in terms of performance improvements, running with new optimisation metrics, and testing a few extra methods. It is this second study which is reported here, and was published in Ref. [237].

The framework and notebook experiments are freely available at Ref. [238].

9.2 The HiggsML dataset

9.3 Higgs ML challenge

In 2014 the Higgs ML challenge was launched on Kaggle (<https://www.kaggle.com/c/higgs-boson>). The challenge was designed to help stimulate outside interest and expertise in solving high-energy physics problems. Participants were tasked with working on a specific kind of problem one often faces in HEP: the search for a rare signal by classification of data based on its features. This signal process was the production of a Higgs boson decaying to a pair of tau leptons, against a background comprised of several more common processes.

The competition was highly successful with 1785 teams competing ¹, and helped to introduce many new methods to HEP, as well as produce more widely used tools, such as XGBOOST [239]. Given the level of expertise and effort that went into the solutions, the challenge forms a viable method to benchmark models and quantitatively measure the impact of new methods and approaches. The challenge is presented in Ref. [235], and the results of the solutions are discussed in Ref. [240].

¹Typical Kaggle competitions can attract anywhere between 50 and 9000 teams. 2000 teams puts this competition in approximately the top 20 % of the most popular competitions. What is worth noting, though, is that this competition required participants to become familiar with a complex field, which could have potentially discouraged people from joining, and yet it still managed to attract so many participants.

9.3.1 Challenge description

Overview

The data used in the challenge consist of simulated particle collisions, generated to mimic those expected at the LHC during its 2012 running. These are fed through a detailed GEANT 4 [39, 134] simulation of the ATLAS detector [241], and finally through the ATLAS reconstruction algorithms, resulting in a set of physics objects per simulated collision. These *events* are then filtered to select those compatible with containing the semi-leptonic decay of a pair of tau leptons, i.e. events which contain a hadronically decaying tau lepton and either an electron or a muon. The properties of the reconstructed physics objects are then recorded in columnar-data format, with each row corresponding to a separate collision event, along with the process label and a weight. The event labels are: “0” for background and “1” for signal. The weight of an event is the product of the production cross-section of the particular physical process that gave rise to the collision (effectively how likely the process was to occur), the probability of the event then meeting the requirements to be included in the final dataset (in this case containing a hadronically decaying tau and either as electron or muon), and the amount of data collected for which the sample was simulated. It is used to normalise the contributions of the events. Both the labels and the weights are only known due to the events being simulated.

The top solutions to the challenge made heavy use of ensembling techniques, combining tens to hundreds of models. The bases of the models were mostly either decision trees/forests, or neural networks. A lot of work seemed to go into feature engineering and selection, with a new fit-based high-level feature (CAKE²) being developed towards the end of the competition. This appeared to improve the results for tree-based models, but gave no significant improvements to DNN-based models, indicating that sufficiently well trained networks were able to learn their own versions of it. The other focus appeared to be optimising the way in which models were ensembled, e.g regularised greedy forests [242].

Scoring metric

The performance of a solution is measured using the Approximate Median Significance [142], as computed in Eq. 9.1:

$$\text{AMS} = \sqrt{2 \left((s + b + b_{\text{reg.}}) \ln \left(1 + \frac{s}{b + b_{\text{reg.}}} \right) - s \right)}, \quad (9.1)$$

in which s is the sum of weights of true positive events (signal events determined by the solution to be signal), b is the sum of weights of false positive events (background events

²CAKE discussion: <https://www.kaggle.com/c/higgs-boson/discussion/10329>
 CAKE code: <https://bitbucket.org/tpgillam/lesterhome/src/master/>

determined by the solution to be signal), and $b_{\text{reg.}}$ is a regularisation term that was set to 10 by the competition organisers. This term effectively ensures that a minimum number of background events are always present in the signal region, which reduces the variance of the AMS (see Sec. 4.3.2 of Ref. [240]). As discussed in Sec. 4.4, the AMS provides a quick, analytical value which is an approximation of the expected significance one would obtain after a full hypothesis test of signal+background versus background only. Note that this is a simplified version of the AMS which was used in Sec. 8.4.1, and does not account for the uncertainty on the background.

As discussed in Sec. 8.1.1 the common practice for these kinds of problems is to cut events from the data in order to remove preferentially the background events whilst retaining the signal events, in order to improve the AMS of the remaining data. Either a single cut can be used on some highly discriminating feature of the data, or multiple cuts can be used over several features. The common ML approach is to adapt the former procedure by using the features of the data to learn a new single highly discriminating feature, place a threshold on it, and then only accept events which pass the threshold. The feature learnt in this approach is simply the predicted class distribution of events, in which background events will be clustered towards zero, and signal events towards one. The AMS can then be optimised by only accepting events with a class prediction greater than some value.

In a full HEP analysis, the sensitivity is usually improved further by fitting histograms of the data along one or two discriminating features, and computing the significance using multiple bins of the data. This can either involve cutting on the ML prediction and then fitting to a different feature, or not cutting on the prediction and instead directly fitting to the histogram of the prediction. Additionally, the data may be split into analysis-specific sub-categories to further increase sensitivity, due to varying background contributions in each category. Examples of such sub-categories for this search might be: channel-wise “tau+electron” and “tau+muon”; and jet-wise “zero jets”, “one jet”, “two or more jets”. However, for simplicity the challenge here requires only a single cut on the ML prediction.

The threshold cut can easily be optimised by scanning across the range of possible cuts (either at fixed steps, or checking at each data point) and picking the value which maximises the AMS. This is likely, however, to be optimal only for the dataset on which it is optimised, and performance can be expected to drop when applying the cut to unseen data. This is reflected in the challenge by requiring the solutions to predict on “test” data for which the class labels are not provided. It is important then that one is more careful when choosing a cut, in order to generalise better to unseen data. The approach adopted here is to consider the top 10 % of events as ranked by their corresponding AMS, and compute the arithmetic mean of their class predictions and use this as the cut. This reduces the influence of statistical fluctuations in the AMS, resulting in a more generalising cut.

9.3.2 Current state of the art

It is my understanding that the winning solution to the Higgs ML Challenge, with an AMS of 3.805 81, has yet to be outperformed under challenge conditions and so will be used as a comparison to the solution developed in this chapter. Whilst Ref. [243] claims to achieve an AMS of 3.979 using a method of converting the tabular data to images, I find that this result is likely a mistake in the authors' handling of the testing data which led them to compute the AMS on double the integrated luminosity, thereby increasing their score by a factor of approximately $\sqrt{2}$. I assert that their method actually achieves an AMS of around 2.8 and document my study in App. 12.1.

9.3.3 ATLAS geometry and coordinate system

Similar to the CMS detector the ATLAS detector [241] is cylindrical in shape, displaying rotational and forwards-backwards symmetry in the transverse and longitudinal planes, respectively, with respect to the beam-axis. It uses the same coordinate systems as CMS, as described in Sec. 3.1.3, and illustrated in Fig. 9.1.

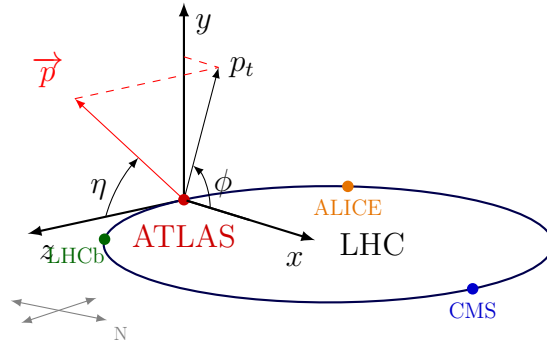


Figure 9.1: Illustration of the coordinate systems used at the ATLAS experiment in the geographical context of the LHC. Image notice [244].

9.3.4 Data and processing

Although the full dataset created for the challenge has now been made public [245], in order to provide an accurate comparison to the previous scores only the subset that was used during the challenge is used here, both for training and testing. The training and testing sets consist of 250 000 and 550 000 events, respectively. Both contain a mixture of both classes: signal ($h \rightarrow \tau^+ \tau^-$); and background ($t\bar{t}$, $Z \rightarrow \tau^+ \tau^-$, and W boson decay). The full dataset is only slightly larger, containing an extra 18 238 events (an increase of 2.3 %), and so it is unlikely to offer much in the way of benefits to training or evaluation.

Use of the full dataset was therefore not thought to be worth the cost of being unable to compare to the baseline results on a like-for-like basis.

As mentioned in Sec. 9.3.1, each event is characterised by two sets of training features: primary - lower level information such as tau p_T ; and derived - higher-level information calculated via (non)-linear combinations of the low-level features or from hypothesis-based fitting procedures. Tables 9.1 and 9.2 list and describe the features, and further details are available in Ref. [235]. The coordinate system of the data is originally provided in terms of (p_T, η, ϕ) , however initial tests on this dataset, and experience from the preliminary chapters (Chap. 7 and Chap. 8) dictates that NN-based models perform better when vectors are mapped into a Cartesian coordinate system, due to the cyclical nature of the azimuthal angle, and the non-linear nature of the pseudorapidity, η :

$$\eta = \tanh^{-1} \left(\frac{p_z}{|\vec{p}|} \right),$$

where p_z is the component of 3-momenta along the beam axis, and $|\vec{p}|$ is the magnitude of the 3-momenta.

The dataset also contains information about the hardest two jets in each event. For cases in which there were not enough jets reconstructed, the features are set to default values of -999 . In order to prevent these values from having an adverse effect on the normalisation and standardisation transformations that will later be applied, these values were replaced with *NaN* (Not a Number - a special value that is ignored by the pre-processing methods), which prevents them influencing the transformations during fitting, and being altered by the transformations during application. Additionally, one of the features, `DER_mass MMC`, which is the mass of the Higgs boson estimated by a hypothesis based fitting technique, sometimes does not converge, in which case the value is set to *NaN*.

Since scoring the testing data requires optimising a threshold on some data, and also to provide some way to compare architectures without relying on too much on the public score of the testing data, an 80 : 20 random, stratified split (as described in Sec. 5.4.2)³ into training and validation sets is performed on the original training data. Since it was found that the validation set had a large impact on the test scores, due to the cut optimisation, experiments are repeated multiple times using different random seeds for the splitting.

After processing the data consist of 31 training features. The training data features are transformed to have a mean of zero, and a standard deviation of one. The exact same transformation is then applied to both the validation and testing data. Following the pre-processing stage, all *NaN* values are replaced with zeros. The training and validation datasets are then each split into ten folds via random stratified splitting on the event class

³Initially the stratification key is the event class, but from Sec. 9.4.4 onwards, the stratification key is based on the event class within categories of jet multiplicity.

(signal or background). The testing data is also split into ten folds, but via simple random splitting. This was done to allow for easy training of ensembles and to make the process of augmenting the data easier. Additionally, I henceforth redefine one epoch as being a complete use of one of these folds, and during training will load each fold sequentially.

Each event in the training and validation data also contains a *weight*. This is normalised to be used to evaluate the AMS, but also reflects the relative importance of the particular event. It is a product of the production cross-section of the underlying process and the acceptance efficiency of the initial skimming procedure. Since the model is unlikely to achieve perfect classification, it is important that it focuses on learning to classify the higher weighted events better than other, less important events. This can be achieved by applying the weights as multiplicative factors during the evaluation of the loss:

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^N w_n \times \epsilon(y_n, \hat{y}_n), \quad (9.2)$$

where w_n is the weight associated with event n in a mini-batch of N events. This method of loss weighting is also used to account for class imbalances in the data by normalising the event weights in training data such that the sum of weights for each class are equal:

$$\sum w_{\text{signal}} = \sum w_{\text{background}} = 1. \quad (9.3)$$

This has the advantage of retaining the relative importance of different events within each class, whilst balancing the importance of each class overall.

9.3.5 Spirit of the investigation

I use the HiggsML dataset due to it being a publicly accessible dataset which is representative of a typical HEP problem in both size and scoring metric, and already has strong baseline solutions from the 2014 Kaggle challenge.

The public release of the dataset now includes the truth labels of the original Kaggle test set meaning that solutions can be fully scored without requiring submissions to Kaggle. This also means that the test set can be further subdivided and mean scores computed allowing both a score and an uncertainty to be reported.

In the preceding version of this investigation, available in Ref. [43], I attempted to reproduce the challenge conditions by only evaluating the public score during development and then checking the private score at the end. Additionally, that investigation only considered single trainings with a fixed seed for the train/validation splitting. The investigation presented here instead attempts to evaluate the general effects that new techniques have on model performance, and so I will report the mean scores from multiple trainings, each with unique random seeds for train/validation splitting. Additionally, the mean public score over ten stratified folds will be computed.

Whilst this metric would be unavailable in true challenge conditions, the final score is a product of both model optimisation and cut optimisation. Since the method to pick the cut is fixed (as per 9.3.1), the problem of *cut optimisation* reduces to *cut compatibility*. Even so, the validation AMS alone is not representative of the true performance of the method; the score on some withheld data at the chosen cut is required. Data limitations mean that it is necessary to rely on the public scores to provide some indication of true model performance. However in order to avoid overfitting to the public score, the mean score will be used, and this will be averaged over repeated trainings.

This all is to say that, although the private scores will be kept blind until the end, this investigation does not fully reproduce challenge conditions and instead aims to demonstrate the general effect of different architectures and training choices. Additionally, the solutions developed will consider the train and inference time requirements, not just performance, and hopes to produce solutions and approaches which are appropriate for use in other HEP applications.

9.3.6 Reported metrics

Several different metrics will be reported during the investigation and are averaged over six repeated trainings, each of which uses a unique train/validation splitting. The metrics are:

- The Mean Maximal Validation AMS (MMVA); the maximum AMS achievable on the validation data. MMVA can be thought of as representing an optimistic upper-limit of solution performance;
- The Mean Validation AMS at Cut (MVAC); the AMS on the validation data at the cut chosen according to the technique described in Sec. 9.3.1. MVAC offers a more general measure of performance, but assumes that the cut used generalises well to unseen data;
- The Mean Averaged Public AMS (MAPA); the value of the AMS on the public part of the testing data at the chosen cut averaged over ten sub-splittings. MAPA represents a more realistic indication of performance since it accounts for cut generalisation.

MAPA will be the main metric for determining improvements, with MMVA and MVAC available to provide further indications of performance. Uncertainties computed for these metrics only account for statistical fluctuations in performance as systematic uncertainties due to theoretical modelling and experimental precision are not accounted for in the challenge.

Additionally, the time taken to train and apply each setup will be reported as the fractional increase in time compared to the baseline setup (see Sec. 9.4.3), and this will

be averaged over the six hardware setups used. A more complete set of timings will be reported in Sec. 9.5.

9.4 Method testing

9.4.1 Machinery

Since this chapter will report the timings of the solutions, I introduce here the machinery used. There are five machines, two of which have the same configuration, and one of which is used in two different hardware configurations (CPU or GPU). Each solution is trained once per hardware configuration, meaning that six trainings are performed. Whilst RAM and VRAM capacity will be stated, I emphasise that this memory requirements by the solutions are minimal and less than 1 GB.

Desktop

- CPU: Intel Core i7-8700K, 6 cores, 2 threads per core, clock-speed 3.7 GHz
- GPU: Nvidia GeForce GTX 1080 Ti (11 GB VRAM)
- RAM: 32 GB, DDR4, clock-speed 2666 MHz
- Data loaded from SSD hard-drive (SATA connection), read speed 550 MB s^{-1}
- Used to provide two hardware configurations: CPU or GPU

MacBook Pro (2018, 13")

- CPU: Intel Core i7-8559U, 4 cores, 2 threads per core, clock-speed 2.7 GHz
- GPU: N/A
- RAM: 16 GB, LPDDR3, clock-speed 2133 MHz
- Data loaded from internal SSD hard-drive, read speed 2520 MB s^{-1}

Virtual Machine (m2.medium)

- CPU: Intel Xenon Skylake CPU, 2 virtual single-thread cores, clock-speed 2.2 GHz
- GPU: N/A
- RAM: 3.7 GB, unknown clock-speed
- Unknown hard-drive

2 X Virtual Machine (m2.large)

- CPU: Intel Xenon Skylake CPU, 4 virtual single-thread cores, clock-speed 2.2 GHz
- GPU: N/A
- RAM: 7.3 GB, unknown clock-speed
- Unknown hard-drive

9.4.2 Feature selection

The definitions of the features available, and selection of the training features to use. All training features are found to be useful.

As discussed in Sec. 9.3.4, the dataset consists of both low-level and high-level features: 18 low-level features consisting of 3-momenta (p_T, η, ϕ), missing transverse momentum, and sum of transverse momenta of all jets; and 13 physics-inspired high-level features such as invariant masses and angles between final-state objects. Tables 9.1 and 9.2 describe the features used along with the associated name that will be used throughout this chapter. Figure 9.2 illustrates the density distributions of two example features, one high-level and the other low-level. It should be borne in mind that 3-momenta of final-states are the result of complex reconstruction techniques and so are not strictly “low-level” features, these would be the tracking hits and energy deposits in the particle detector, however they are the lowest-level of information provided in the dataset.

The mass of the Higgs boson candidate is expected to be a particularly useful feature: the $\tau^+ \tau^-$ final-state candidates should form an invariant mass around 125 GeV for signal, 91 GeV the $Z \rightarrow \tau^+ \tau^-$ background, and a broad range of masses for the W -decay (with perhaps a peak around 80 GeV) and $t\bar{t}$ backgrounds. The tau decays, however, involve neutrinos, which cannot be detected at the LHC detectors leading to a loss of kinematical information. `DER_mass_vis` is an estimation of the Higgs mass using the visible decay products, and underestimates the Higgs boson mass. `DER_mass MMC` instead attempts to account for the missing energy and produce a more accurate estimate for the Higgs mass by performing a kinematic fit using the visible decay products and the missing transverse momentum under the hypothesis of a di-tau decay [246].

My hope is that the networks will be able to exploit the low-level information, but without this exploitation the low-level features risk being removed during feature selection. I am therefore more interested in seeing whether all of the high-level features are necessary. Testing will use a simplified type classification model called a Random Forest [189] (see Sec. 5.3.2). I use these as they are quick and easy to train, provide decent performance, and do not require the data to be transformed in any particular fashion prior to training.

Feature name	Description	Grouping
DER_mass_MMC	Mass of the Higgs boson estimated by a hypothesis based fitting technique	Mass, Higgs
DER_mass_transverse_met_lep	Transverse mass of the lepton and $\vec{p}_T^{\text{miss}}$	Mass, Higgs
DER_mass_vis	Invariant mass of the lepton and the tau	Mass, Higgs
	Higgs boson	
DER_pt_h	Transverse momenta of the vector sum of the lepton, tau, and $\vec{p}_T^{\text{miss}}$	3-momenta, Higgs
DER_deltaeta_jet_jet	Absolute difference in pseudorapidity of the leading and subleading jets (undefined for less than two jets)	Angular, Jet
DER_mass_jet_jet	Invariant mass of the leading and subleading jets (undefined for less than two jets)	Mass, Jet
DER_prodetta_jet_jet	Product of the pseudorapidities of the leading and subleading jets (undefined for less than two jets)	3-momenta, Jet
DER_deltar_tau_lep	Separation in $\eta - \phi$ space of the lepton and the tau	Angular, Final-state
DER_pt_tot	Transverse momentum of the vector sum of the transverse momenta of the lepton, tau, the leading and subleading jets (if present), and $\vec{p}_T^{\text{miss}}$	Sum, Final-state
DER_sum_pt	Sum of the transverse momenta of the lepton, tau, and all jets	Sum, global event
DER_pt_ratio_lep_tau	Transverse momenta of the lepton divided by the transverse momenta of the tau	3-momenta, Final-state
DER_met_phi_centrality	Centrality of the azimuthal angle of $\vec{p}_T^{\text{miss}}$ relative to the lepton and the tau	Angular, Final-state
DER_lep_eta_centrality	Centrality of the pseudorapidity of the lepton relative to the leading and subleading jets (undefined for less than two jets)	Angular, Jet

Table 9.1: Glossary of high-level (derived) features used for training, along with their grouping.

Feature name	Description	Grouping
PRI_tau_[px/py/pz]	3-momenta of the tau in Cartesian coordinates	3-momenta, Final-state
PRI_lep_[px/py/pz]	3-momenta of the lepton in Cartesian coordinates	3-momenta, Final-state
PRI_met_[px/py]	Components of the vector of missing transverse momentum in Cartesian coordinates	3-momenta, Final-state
PRI_met	Modulus of the vector of missing transverse momentum in Cartesian coordinates	3-momenta, Final-state
PRI_met_sumet	Sum of all transverse energy	Energy, Final-state
PRI_jet_num	Number of jets in event	Multiplicity, Jet
PRI_jet_leading_[px/py/pz]	3-momenta of the leading jet in Cartesian coordinates (undefined if no jets present)	3-momenta, Jet
PRI_jet_subleading_[px/py/pz]	3-momenta of the subleading jet in Cartesian coordinates (undefined if < than two jets present)	3-momenta, Jet
PRI_jet_all_pt	Sum of transverse momenta of all jets in Cartesian coordinates	3-momenta, Jet

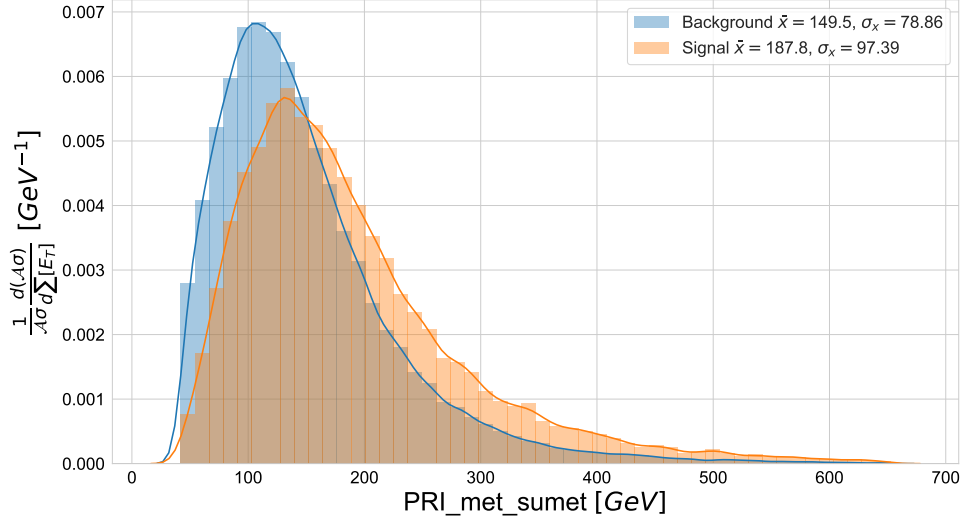
Table 9.2: Glossary of low-level (primary) features used for training, along with their grouping. Note that *PRI_jet_all_pt* can be expected to be different from the sum of p_T of leading and subleading jets, since there can be events which contain more than two jets, however the dataset only contains details of the two jets with the highest p_T .

Quantification of a feature’s importance is done using its *permutation importance*. This aims to quantify how much a given feature influences the model’s output. Features with a higher importance will have a greater influence overall than those of lesser importance. The permutation importance (PI) is computed by first training the model as normal and evaluating its performance on the training set. Next, a copy of the training data is made and one of the features is randomly shuffled datapoint-wise, thereby destroying any information carried by that feature for each datapoint, whilst keeping its values at the same scale as seen during training. The performance of the model on the shuffled data is computed and the difference in performance compared to the original performance is the permutation importance of the feature that was shuffled. If shuffling the feature caused a large change in performance, then that feature was important to the model. If the change was low, then the feature was less important. By then dividing the PI by the original performance we arrive at the *fractional* permutation importance.

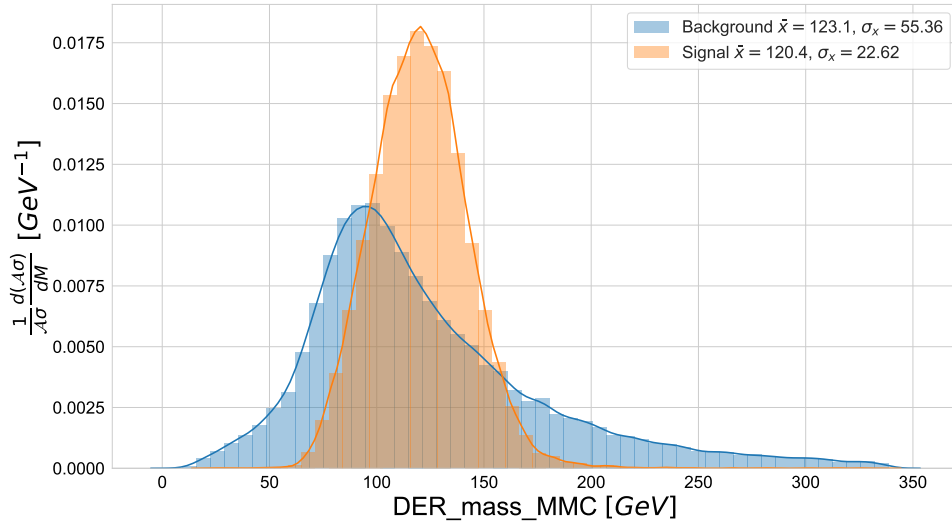
This process of copying and shuffling is repeated for every feature. By retraining new models, an average PI can be computed for each feature. Although the dataset is the same each time, the feature subsampling at each split acts as a source of stochasticity. Here I compute PIs using an average of five retrainings. A threshold on the PI can then be used to select features by minimum importance. I use a threshold of 0.005 as to identify features as “important”. From Fig. 9.3, we can see that all features except the four nearest the top of the plot, i.e. those associated with jet pairs, appear to be important.

The lack of importance of jet-pair features is to be expected: since the majority of events do not contain two or more jets, the values for these features are often the default value of zero. When the feature is permuted, most of the zeros will be replaced with another zero, and the model response remains mostly unaffected. Effectively the di-jet features are only important for a subset of events, but their importance is smeared out across all events. In actual application, the jet number will be encoded in a way such that the model will be able to more easily target these di-jet events and make use of their features. The di-jet features will therefore be kept. As additional motivation, training a new Random Forest without these features showed a drop in performance on unseen data.

In order to check for monotonic relationships between features I then compute the Spearman’s rank-order correlation coefficients [247] for every pair of high-level features and do not find any pairs that are fully (anti-)correlated with one another. As a final check I examine the mutual dependencies of the features, i.e. how easy it is to predict the value of a feature using the other features as inputs to a Random Forest regressor. As shown in Fig. 9.4, none of the non-di-jet high-level features are completely predictable using the other features (dependencies are never equal to one). We can therefore expect that each high-level feature is bringing at least some unique information not carried by the others.



(a) Example low-level feature: the sum of missing transverse energy ($\sum E_T$)



(b) Example high-level feature: the masses of the Higgs candidates as computed by a hypothesis-based fitting algorithm ($M_{\tau\tau, MMC}$)

Figure 9.2: Example distributions for low-level and high-level features in the data.

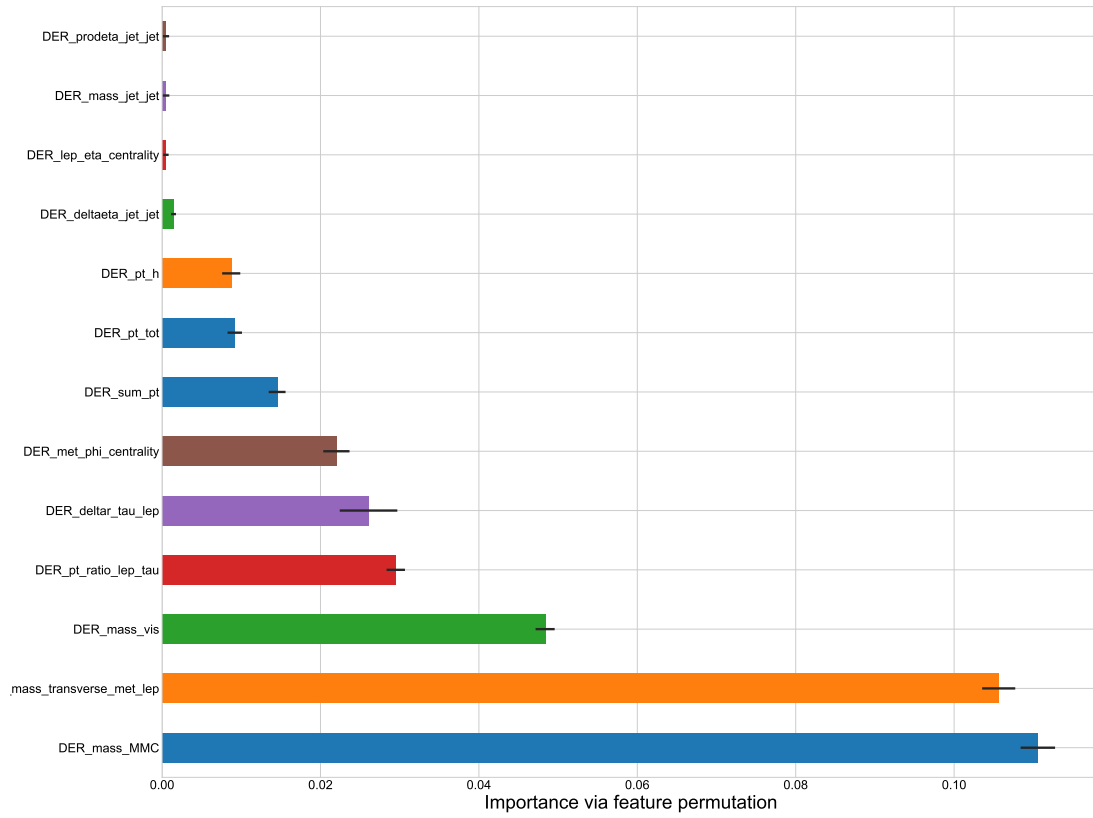


Figure 9.3: Illustration of the fractional permutation importance of the high-level features as estimated using the average importance over five Random Forest models.

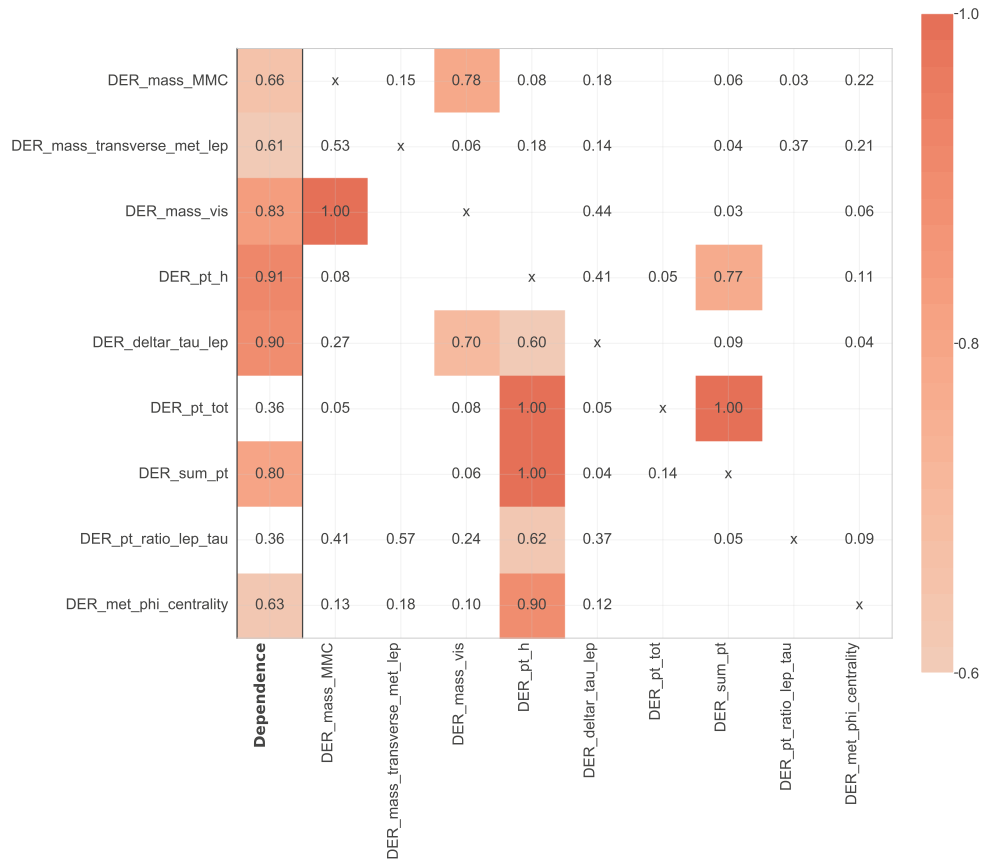


Figure 9.4: Illustration of feature dependence and associated feature importance using Random Forest regression. Considering each row separately, the feature on the y axis is one being regressed to. The value in the “Dependence” column is the coefficient of determination (R^2); higher means easier to predict the feature values. The remaining columns indicate how much each feature on the x axis contributes to the performance of the regressor; higher means more important. As an example: **DER_pt_h** is the easiest feature to predict (dependence of 0.92), and **DER_sum_pt** is moderately important to the regressor.

9.4.3 Baseline model

The basic architecture used for future comparisons, and also a demonstration of the benefits of ensembling models.

Training and inference of all solutions developed in this study are performed in LUMIN [236], a wrapper library for PYTORCH [181] which I developed during the course of this investigation.

Architecture

The baseline model consists of a simple neural network architecture. Specifically, it contains four fully-connected layers, each of 100 neurons, with weights initialised according to the He-normal [168] scheme. ReLU activation functions [162] follow each fully-connected layer. The output layer consists of a single neuron, whose weights are initialised according to the Glorot-normal [167] scheme. A sigmoid activation function is applied to the output of the network. The starting biases for all layers are set to zero.

As usual, the parameters (weights and biases) of the network are optimised through backpropagation [157] of a loss-gradient estimated by averaging batches of data points (here I use a minibatch size of 256 events). Given that the problem involves binary classification, the loss function is chosen to be the binary cross-entropy of network predictions. The update steps are determined using ADAM [176] with relatively standard parameter settings of $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1 \times 10^{-8}$. The learning rate (LR) is kept constant during training at 2×10^{-3} , which was found using a LR Range Test as described in Ref. [248]. Training continues until the loss on a validation fold of the training data has not decreased in the last 50 epochs (a *patience* of 50), after which the model state corresponding to the lowest validation loss is loaded.

Performance

Single model A single model is trained on nine out of the ten training folds, with the tenth fold acting as validation data for loss monitoring. Once finished, the model is applied to both the validation and testing data, and the required metrics recorded. This process is run once per hardware configuration, as defined in Sec. 9.4.1 (six times in total), each using a unique random seed for the train/validation splitting as described in Sec. 9.3.4. Performance metrics for the single model are reported in Tab. 9.3.

Ensemble An ensemble of ten models is trained, again using nine out of the ten training folds, with the tenth fold acting as validation data. Each model uses a different fold of the training data for validation. The resulting models are ensembled by weighting their predictions proportionally to the inverse of their associated loss on their validation folds.

Setup	MMVA	MVAC	MAPA	Fractional time-increase	
				Training	Inference
Single model	3.43 ± 0.04	3.38 ± 0.04	3.44 ± 0.02	-	-
Ensemble	3.70 ± 0.05	3.64 ± 0.04	3.664 ± 0.007	10.4 ± 0.6	5.9 ± 0.9

Table 9.3: Validation-performance comparison between using the single model and weighted ensemble setups. Higher values of MMVA, MVAC, and MAPA indicate better performance, and higher values of fractional time-increase indicate that the solution takes longer to train and/or apply, i.e. the ensemble solution takes about 10 times longer to train than the single model. The inference time increase for the ensemble is less than 10, since inputs are kept in memory between calls to each model, so loading times are not repeatedly incurred. The best values for each metric are shown in bold, and the setup chosen is also indicated in bold.

The prediction of the ensemble is then:

$$p_{\text{ensemble}}(x) = \sum_i^{10} w_i p_i(x), \quad (9.4)$$

where $p_{\text{ensemble}}(x)$ is the prediction of the ensemble for datapoint x , $p_i(x)$ is the prediction of model i for datapoint x , and w_i is the importance weight of model i . Importance weights are normalised such that they sum to one. This means that models that reached a lower validation loss during training have a higher importance weight and consequent a greater influence over the ensemble prediction than those with a higher loss.

Ensembling is seen to provide improvements in all metrics compared to the single model setup, with expected increases in train and inference time. Note that the inference-time increase appears to be sub-linear due to the fact that the data are kept in memory between model evaluations. Based on the improved performance, I proceed to adopt the ensemble version as the baseline model for later comparisons.

9.4.4 Categorical entity embedding

The testing of an efficient way to encode categorical information, which results in mild improvements.

In Sec. 9.4.3 I considered all features in the data to be continuous variables, however the number of jets in an event (PRI_jet_num) could instead be thought of as a categorical feature. Although two jets are more than one jet, this can also be treated as defining subcategories of particle interactions, e.g. a two-jet event and a one-jet event, in which

case there is no longer a numerical comparison. Optimal treatment of categorical features requires encoding the different categories in a way such that the ML algorithm can easily access them individually. A common way to do this is to assign each category an integer code which begins at zero and sequentially increases, and then treat these codes as row indices of an *embedding* matrix. The elements in the indexed row are then used to represent the corresponding category.

Under the *1-hot encoding* scheme, the embedding matrix is a $N \times N$ identity matrix (where N is the number of categories in a feature; its *cardinality*), i.e. each category is represented by a vector of length N in which only one value is non-zero (one element is *hot*). Whilst this approach provides unique access to each category, the number of input variables is now N . For high cardinality features, or many low-cardinality features, this can quickly lead to a huge increase in the number of model parameters which must be learnt.

Reference [249] instead suggests for neural networks to use an $N \times M$ weight matrix for the embedding, where $M < N$ is a hyper-parameter which must be chosen. Each category is now represented by a real-valued vector of length M , i.e. in a more compact fashion than 1-hot encoding would provide. The values of the weight matrix are then learnt via backpropagation during training. This approach of *categorical entity embedding* works because, for example, it is likely that it is more important to know whether an event contains zero, one, several, or many jets, rather than exactly how many jets, but rather than manually compacting the categories (and having to define a priori “several” and “many”), a compact, information rich, and task specific representation can be learnt. The additional hyper-parameter M can be estimated using domain knowledge or by using some rule-of-thumb: the original paper suggests to use $M = N - 1$, and FASTAI suggests to use half the feature cardinality or 50, whichever is smaller ($M = \min(50, (N + 1) // 2)$ [250])

For embedding `PRI_jet_num`, M is chosen to be three and initial weights for the 4×3 matrix are sampled from a unit-Gaussian distribution. As shown in Tab. 9.4, this is found to provide metric improvements across the board, for only minor increases in train and inference time. Figure 9.5 shows an example of one of the learned embeddings. Please be aware that Tab. 9.4 includes all improvements that are discussed in this thesis, and so contains solutions which may not have been introduced at the time of reading

Whilst not required for this particular study, as discussed in Sec. 9.3.1, HEP analyses potentially categorise data in order to further improve the sensitivity of their search. The definition of these inference categories depends on the search being performed. Examples could be categorisation by di-tau decay-channel, and categorisation by jet-multiplicity. Flags indicting which categories the events belong to can potentially be encoded as categorical features and embedded. This then allows the classifier to better parameterise its responses to each inference category, whilst benefitting from a larger training sample (as opposed to training separate classifiers for each category), and learning information

common to all categories.

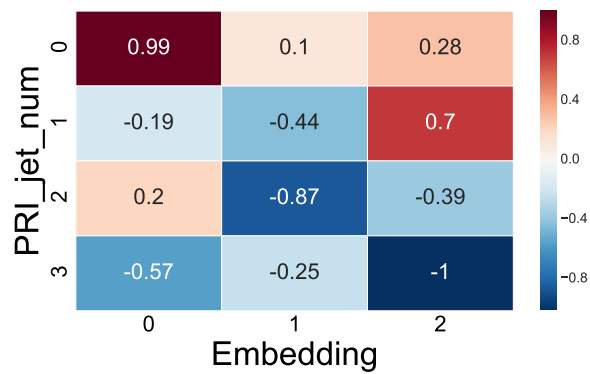


Figure 9.5: Illustration of an example of a learned embedding matrix for the number of jets. Note that each row is unique, indicating that the NN found it useful to learn separate embeddings for each category, and was able to transform the feature into a lower dimensional representation.

Setup	MMVA	MVAC	MAPA	Fractional time-increase Training	Inference
Baseline	3.70 ± 0.05	3.64 ± 0.04	3.664 ± 0.007	-	-
+Embedding	3.78 ± 0.05	3.70 ± 0.06	3.71 ± 0.02	0.13 ± 0.03	0.08 ± 0.04
+Augmentation	3.96 ± 0.04	3.89 ± 0.05	3.79 ± 0.01	1.45 ± 0.08	14 ± 5
+SGDR	3.94 ± 0.06	3.90 ± 0.04	3.80 ± 0.02	2.12 ± 0.07	14 ± 5
+Swish	3.96 ± 0.06	3.86 ± 0.05	3.81 ± 0.02	2.32 ± 0.07	17 ± 4
-SGDR + SWA	3.96 ± 0.06	3.88 ± 0.05	3.81 ± 0.02	2.03 ± 0.07	17 ± 4
-SWA + 1cycle	4.04 ± 0.06	3.85 ± 0.04	3.81 ± 0.02	0.66 ± 0.04	17 ± 4
+Dense	3.95 ± 0.04	3.89 ± 0.05	3.82 ± 0.02	0.81 ± 0.08	17 ± 5

Table 9.4: A summary of solution evolution in terms of the optimisation metrics and the time impact. Fractional time-increases are computed with respect to the baseline model, which is the ensembled solution from Sec. 9.4.3. A ‘+’ indicates the addition of a new method to the solution, and ‘-’ indicates the removal of a previously added method. The best values for each metric are shown in bold, and the setup chosen is also indicated in bold.

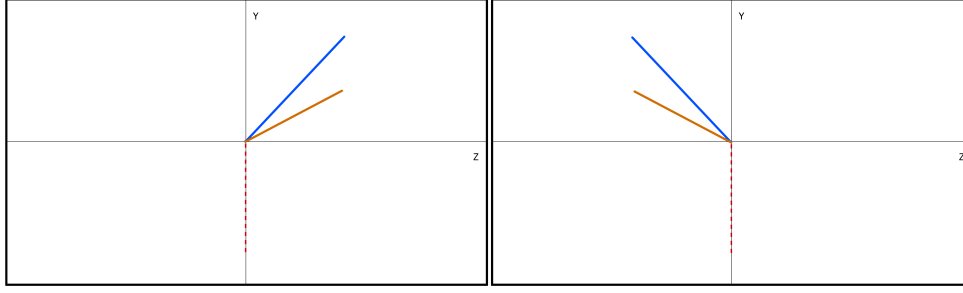
9.4.5 Data symmetry exploitations

Discussion and application of HEP-specific data augmentation, which result in large improvements.

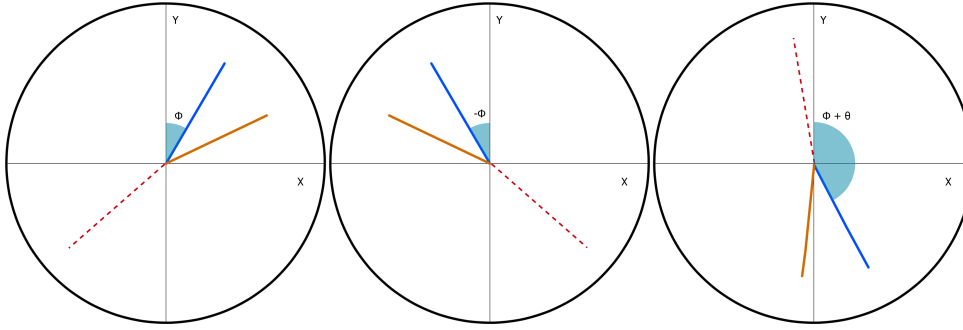
Overview of data symmetries

At the LHC [35], beams of protons are collided head-on with approximately zero transverse momentum. Because of this the resulting final states can be expected to be produced isotropically in both the transverse plane (x, y) and the longitudinal axis (z). Particle detectors such as CMS [66] and ATLAS [241] are built to account for these isotropies by being symmetric in both azimuthal angle (ϕ) and pseudorapidity (η). All this is to say that the class of physical process that gave rise to the collision event is only related to the relative separation between the final states, and not to the absolute position of the event.

Since the data used in this problem are simulated for such a collider and detector combination, the class is invariant to flips in the transverse or longitudinal planes, and to rotations in the azimuthal angle, as illustrated in Fig. 9.6. Note that it is important to reconsider the set of class-preserving transformations if intending to use the following techniques to analyse data for a different experimental configuration, such as: colliding beams of different particles, using a fixed-target experiment, or an asymmetrical detector.



(a) Left: an example event in the y – z plane, with blue and orange lines representing arbitrary final-state vectors and the dashed line representing missing transverse momentum. Right: the same event flipped in the longitudinal axis.



(b) Left: an example event in the transverse plane, with blue and orange lines representing arbitrary final-state vectors and the dashed line representing missing transverse momentum. Centre: the same event rotated in the azimuthal angle (ϕ). Right: the same event flipped in the y axis.

Figure 9.6: Illustrations of class-preserving transformations for particle collisions.

Data fixing

The existence of class-invariant symmetries in the data acts to increase the data complexity; a classifier must learn to become invariant to the absolute orientation of events and instead focus on the positions of the particles relative to one-another. This complexity can easily be removed by rotating all events to have a common alignment. Note that rotation in the transverse plane is effectively computing the $\Delta\phi$ angles of other final-states with respect to the light-leptons, and indeed this is how the fixing is implemented in the code.

I refer to this pre-processing step as *data fixing*, and implement it by first rotating events in the transverse plane and reflecting in the longitudinal plane such that the light-leptons (PRI_lep) are at $\phi = 0$ and $\eta \geq 0$, and then reflecting the event in the transverse plane such that the tau leptons (PRI_tau) are in the positive ϕ region. Since the y component of the light-lepton's 3-momentum is now constant (zero) it can be dropped as an input feature during training.

Setup	MMVA	MVAC	MAPA	Fractional time-increase	
				Training	Inference
Fixing	3.90 ± 0.04	3.83 ± 0.05	3.76 ± 0.01	-	-
Augmentation	3.96 ± 0.04	3.89 ± 0.05	3.79 ± 0.01	1.11 ± 0.05	13 ± 5

Table 9.5: Comparison of data symmetry exploits in terms of the optimisation metrics and time impact. The best values for each metric are shown in bold, and the setup chosen is also indicated in bold.

Data augmentation

An alternative approach to dealing with the data symmetry is instead to exploit it as a means of generating augmented copies of the events. Data augmentation is a common technique for improving the generalisation power of a model. It involves applying class-preserving transformations to the data in order to exploit or highlight certain invariances between the input and target features. In the field of image classification, these transformations could be small rotations, flips, zooms, or colour and lighting adjustments, e.g. Ref. [186]. The data augmentation applied here consists of event flips in the y and z planes and rotations in the transverse plane. Application of the augmentations may be done during training in order to artificially inflate the size of the training data (train-time augmentation), or during inference by taking the mean prediction for a set of augmentations (inference-time augmentation).

Train-time augmentation was performed by applying a random set of augmentations before each data point is used (implemented by augmenting each fold when loaded). Inference-time augmentation was performed by taking the mean prediction for each data point over a set of eight transformations: each possible combination of flips in y and z for two different ϕ orientations (original ϕ and $\phi + \pi$).

Comparison

Table 9.5 compares the two approaches in terms of the optimisation metrics. Data augmentation produces more favourable scores, but at the expense of a large increase in inference time. Nevertheless, the absolute inference time is still acceptable (15 s on a GPU (Nvidia GeForce GTX 1080 Ti) and 2 min on the slowest CPU (Intel Xenon Skylake CPU, 2 virtual single-thread cores, clock-speed 2.2 GHz)), so the augmentation approach was chosen going forwards. Table 9.4 compares the improvements against the baseline solution.

9.4.6 Stochastic gradient descent with warm restarts

Testing of learning-rate annealing schedules. Whilst this results in moderate improvements it eventually becomes obsolete by the schedule in Sec. 9.4.9.

According to Ref.[251], the learning rate (LR) is one of the most important parameters to set when training a neural network. Whilst initial, optimal values can be found using a LR Range Test [248], it is unlikely that this will remain optimal throughout training. A common approach is to decrease the LR in steps [252] during training (either at preset points or when the loss reaches a plateau).

Reference [253] instead proposes a *cosine-annealing* approach in which the LR is decreased as a function of mini-batch iterations according to half a cycle of a cosine function. This means that initial training is performed at high LRs, where the model can quickly move across the loss surface towards a minima, and the final training is spent at lower LRs allowing the model to converge to a minima, with a smooth, approximately linear transition between these states. Additionally, by repeating this cosine decay, the paper suggests that the rate of convergence can be further improved, a process described as *warm restarts*.

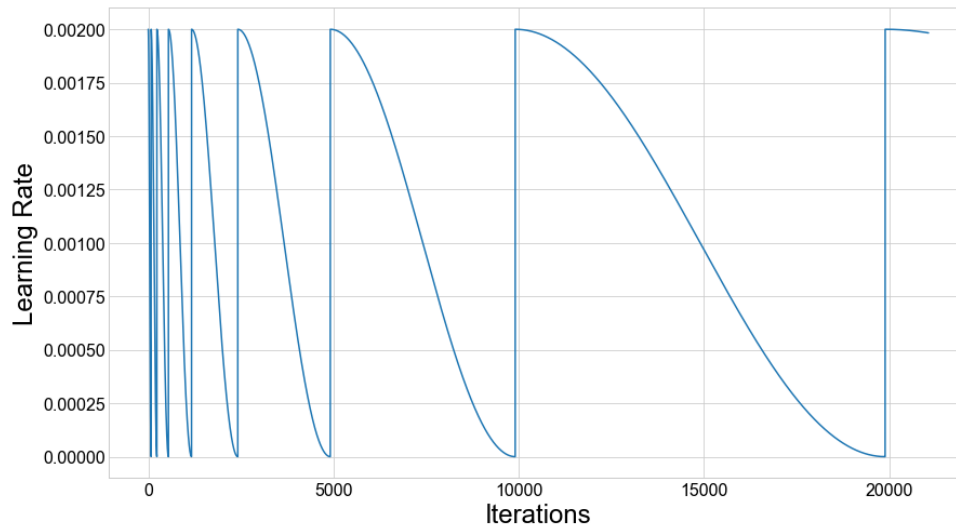
Reference [254] suggests these sudden changes in the LR allow the model to explore multiple minima. This exploration of minima could in turn lead to the discovery of wider, less steep minima, which should generalise better to unseen data since changes in the centre of the minima will have less of an effect on the loss than if the minima were steeply sided.

Figure 9.7 illustrates the LR schedule and an example of the validation-loss evolution.

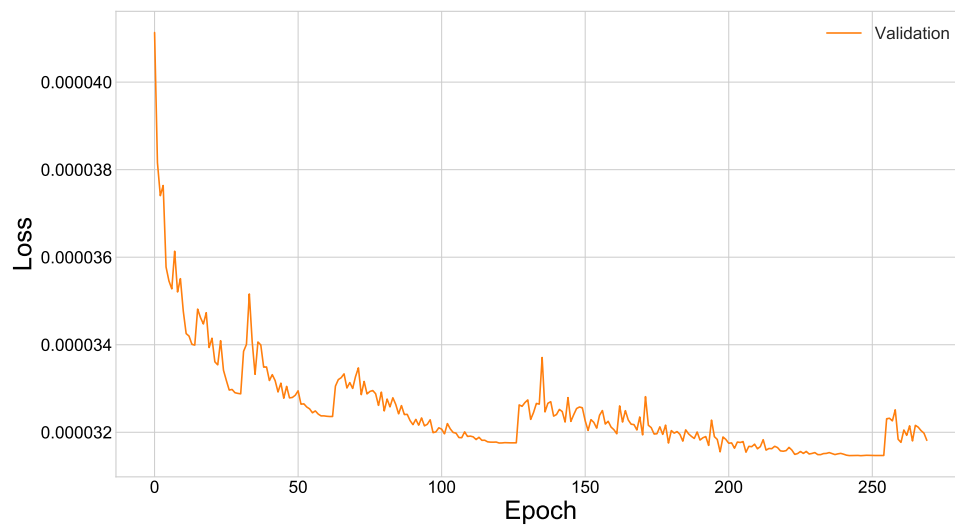
1. It begins at an initial LR, γ_0 , of 2×10^{-3} and an initial cycle length, T , of one complete use of the training folds.
2. During the cycle, the LR is decreased according to $\gamma_t = 0.5 \gamma_0 \times (1 + \cos(\pi t/T))$, where t is incremented after each minibatch update
3. Once the LR would become zero ($t = T$), t is reset to zero, resulting in a sudden increase in the LR (the warm restart). Additionally, the cycle length T is doubled allowing the model to eventually spend longer and longer time at lower LRs, improving convergence.

The warm restarts are visible in the validation loss, appearing as small increases in the loss. Initially, the early stopping criterion was modified such that training stopped once the model went one entire cycle without an improvement in the validation loss, however since testing showed models always reached this point in the same annealing cycle, the number of training epochs was instead fixed to avoid unnecessary training time.

The results (Tab. 9.4 Row “+SGDR”) show a minor improvement in both MVAC and MAPA, and a decrease in MMVA, indicating a less overly optimistic solution with better generalisation.



(a) Learning-rate schedule



(b) Validation loss evolution

Figure 9.7: Cosine annealed learning-rate schedule with increasing cycle length and example of the corresponding loss evolution for a model trained with such a schedule

9.4.7 Activation function

The inclusion of a newer activation function, which provides slight improvements.

Whilst ReLU is a good default choice for an activation function it does exhibit several problems, which more recent functions attempt to address:

- The Parametrised ReLU (PReLU) [168] is similar to ReLU, but can feature a constant, non-zero gradient for negative inputs, the coefficient of which is learnt via backpropagation during training;
- The *Scaled Exponential Linear Unit* (SELU) [164] uses careful derived scaling coefficients to keep signals in the network approximately unit Gaussian, without the need for manual transformation via batch normalisation [165]. It is recommended to use the `lecun` initialisation scheme [169] for neurons using the SELU activation function;
- The Swish function [166] was found via a reinforcement learning based search, and features the interesting characteristic of having a region of negative gradient, allowing outputs to decrease as the input increases: $\text{Swish}(x) = x \cdot \text{sigmoid}(\beta x)$, where β can either be kept constant or be learnt during training. The recommended weight initialisation scheme for Swish is the same as the one for ReLU, i.e. He.

A comparison of these functions is shown in Fig. 9.8.

Previous testing, documented in Ref. [43], found that the Swish activation function provided superior performance, and so its improvements were again tested here. I use the Swish-1 version, in which $\beta = 1$ and is not changed during training. As shown in Row “+Swish” of Tab. 9.4, its inclusion provides slight improvements to MAPA, and so its use was accepted. Note that the small time increase is due to the exponentials in Swish being more complicated to compute than the `max` operation of ReLU.

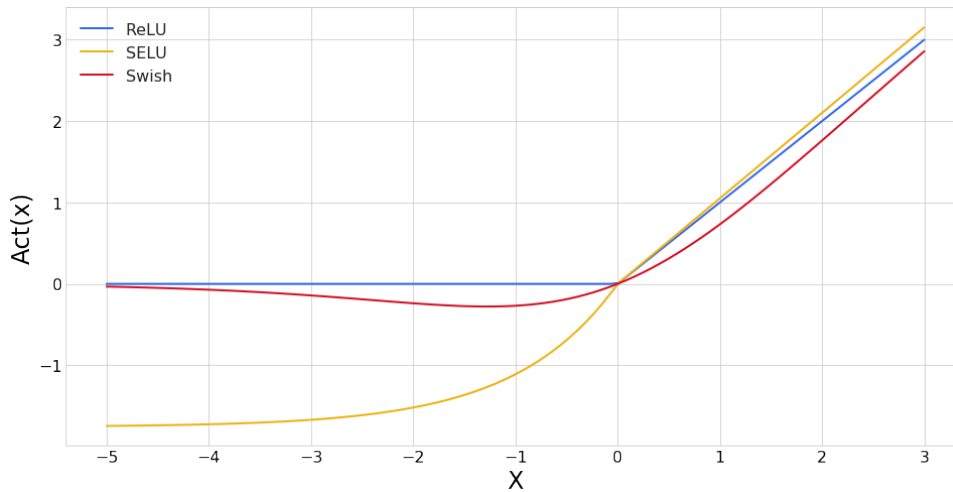


Figure 9.8: Responses ($Act(x)$) for several activation functions for a range of input values, x . PReLU (not shown) is the same as ReLU in the positive region, $Act(x) = x$, and with a constant, non-zero gradient in the negative region, $Act(x) = \alpha x$, where α is a parameter of the model.

9.4.8 Advanced ensembling

A discussion and experimental test of several more complicated methods of building an ensemble of models. Additionally a new implementation of Stochastic Weight Averaging is introduced which does not rely on a predefined starting point. Whilst this provides slight improvements in training time it is eventually made obsolete by the schedule in Sec. 9.4.9.

As mentioned in Sec. 9.4.6, Ref. [254] suggests that Ref. [253]’s process of restarting the LR schedule allows the training to discover multiple minima in the loss surface. The paper builds on this by introducing a method in which an ensemble of networks results from a single training cycle by saving copies of the networks just before restarting the LR schedule, i.e. when the model is in the minima. This process of *Snapshot Ensembling* (SSE) is further refined into *Fast Geometric Ensembling* (FGE) in Ref. [255], by forcing the model evolution along curves of constant loss between connected minima (simultaneously and independently discovered by Ref. [256]).

The problem with these methods is that whilst they allow one to train an ensemble in a much reduced time, one still has to run the data through each model individually, so the application-time is still increased. Reference [257] instead finds an approach which leads to an approximation of FGE in a single model. This is done by averaging the models in *weight space*, rather than in *model space*. The general method involves training a model as normal, until the model begins to converge. At this point the model continues to train

as normal, but after each epoch a running average of the model’s parameters is updated.

In the training, one has to decide on when to begin collecting the averages: *too early*, and the model average is spoiled by ill-performing weights; *too late*, and the model does not explore the loss surface enough to allow SWA to be of use. Additionally, SWA may be combined with a cyclical learning rate, in which case weight averaging should take place at the end of each cycle.

Implementation

SSE SSE was tested by training ten networks as usual, with a cosine-annealed LR (initial LR = 2×10^{-3}) with a constant cycle length of 50 folds. Training continued until the validation loss failed to decrease for two complete cycles. Snapshots were then loaded, starting with the best performing set of weights, and then up to four previous snapshots. A weighting function of the form $w = n^{-1}$ was used, where n is the 1-ordered number of weight loads (i.e. best performing weights = 1, first previous snapshot = 2, *et cetera*). This means that snapshots later in the training are weighted higher than earlier ones, in order to balance the trade off between greater generalisation due to a larger ensemble being used and the poorer performance of earlier snapshots. Additionally, snapshots loaded from different model trainings were not reweighted according to validation loss as they were in Sec. 9.4.3.

FGE Since FGE expects to send models along curves of near constant loss between minima, it employs a higher frequency saving of snapshots. A linearly cycled LR [248] is used, moving from LR = 2×10^{-4} to 2×10^{-3} and back again over the course of five folds. Training continued until the validation loss had not decreased for nine cycles. Snapshots were loaded in a similar fashion to SSE except up to 8 cycles, as well as the best performing weights, were loaded, and no cycle weighting was used (the losses of the cycles were approximately equal, as expected).

SWA Rather than having to pick a starting point to begin SWA (which would require running the training once beforehand without SWA), SWA was begun early during training and later a second SWA model was started. At set points during training the SWA averages were compared. If the older average performed better, then the younger average was reset to the current model parameters and the time to the next comparison was doubled. If the younger average was better, then the older average was reset to the current model parameters and the time to the next comparison was set back to its initial value (its *renewal period*). Effectively, a range of possible start points for SWA models are tested, and the optimum start position is automatically selected.

Additionally, three LR schedules were tested:

Setup	MMVA	MVAC	MAPA	Fractional time-increase	
				Training	Inference
Current solution	3.96 ± 0.06	3.86 ± 0.05	3.81 ± 0.02	-	-
SSE	3.95 ± 0.05	3.85 ± 0.05	3.81 ± 0.02	0.15 ± 0.03	1.5 ± 0.3
FGE	3.95 ± 0.06	3.86 ± 0.05	3.80 ± 0.02	-0.13 ± 0.03	6 ± 1
SWA constant	3.96 ± 0.06	3.88 ± 0.05	3.81 ± 0.02	-0.09 ± 0.01	-
SWA linear cycle	3.97 ± 0.06	3.87 ± 0.04	3.81 ± 0.02	0.25 ± 0.03	-
SWA cosine	3.95 ± 0.05	3.89 ± 0.05	3.81 ± 0.02	0.22 ± 0.04	-

Table 9.6: Comparison of the various advanced ensembling approaches. “Current solution” refers to the solution as of Sec. 9.4.7. The best values for each metric are shown in bold, and the setup chosen is also indicated in bold.

- A constant LR of 2×10^{-3} running with a patience of 50 folds with SWA beginning on the fifth fold, and comparisons between averages taking place with an initial separate of five folds.
- A Cosine annealed LR between 2×10^{-3} and 2×10^{-4} over five folds, a patience of 9 cycles, and SWA beginning on the second cycle with a renewal period of two cycles.
- A linearly cycled LR between 2×10^{-4} and 2×10^{-3} over five folds, a patience of 9 cycles, and SWA beginning on the second cycle with a renewal period of two cycles.

Comparison

Comparing the approaches to the current best solution in Tab. 9.6, we can see that no approach is able to improve the current MAPA, however SWA with a constant LR achieves the same score in a shorter train time. From an example plot of the validation loss over time (Fig. 9.9) we can see that SWA not only demonstrates better performance, but it also shows a heavy suppression of loss fluctuations and converges to a loss plateau, whereas the non-averaged model eventually over-fits (signified by an increasing loss in the later stages of training). The sharp drops in SWA loss are due to the old average being replaced with the newer average.

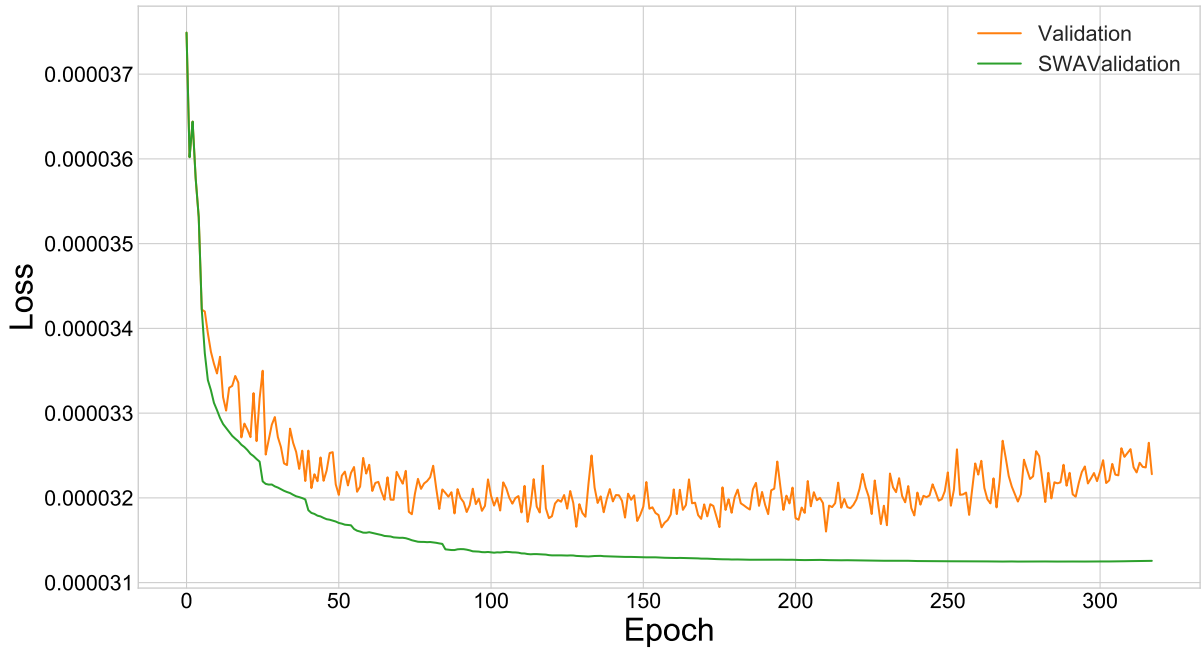


Figure 9.9: Example evolution of validation loss over training time for both model being trained and the stochastic weight-average of the model states.

9.4.9 Super-convergence

The application of a schedule for both learning rate and momentum, which provides a significant reduction in training time.

Reference [258] introduces the concept of *super convergence*, in which a specific LR schedule, *1cycle*, is used to achieve convergence much quicker (between five and ten times) than traditional schedules allow. This is further discussed in Ref. [259]. The *1cycle* policy combines both cyclical LR and cyclical momentum⁴ (or β_1 in case of ADAM), and evolves both hyper-parameters simultaneously and continuously from their starting values to another value and then back again over the course of training in a single cycle. Hyper-parameter evolution takes place in opposite directions, with the LR initially increasing and the momentum initially decreasing, i.e. when the LR is at its maximum, the momentum is at its minimum, as illustrated in Fig. 9.10. In this way the two parameters help to balance each other allowing the use of higher LR without the network becoming unstable and diverging. In the final stages of training, the LR is allowed to decrease several orders of magnitude below its initial value.

⁴Momentum here refers to optimiser momentum and not physical momentum, as might potentially be inferred.

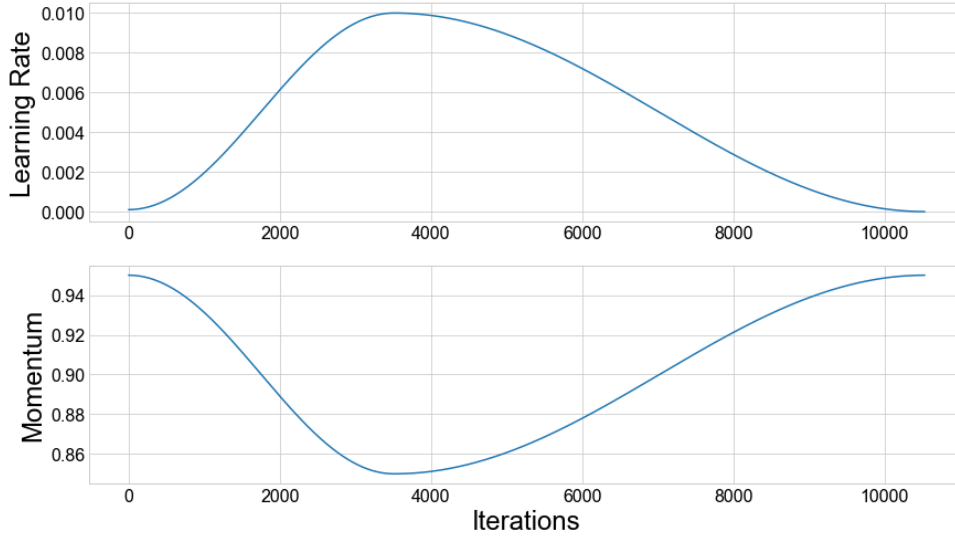


Figure 9.10: Illustration of the evolution schedule of the learning rate and momentum (β_1) used during testing. Iterations are minibatch updates.

One of the key points of Ref. [259] is that super-convergence can only occur if the amount of regularisation ⁵ present in the setup is below a certain amount, but that once super-convergence can be achieved, other forms of regularisation can then be tuned accordingly. In my current setup, I have avoided explicitly adding tuneable forms of regularisation ⁶ for this reason. However Sec. 9.4.5’s choice to use data augmentation rather than data fixing equates to choosing not to reduce regularisation through data complexity. Because of this, I will be conservative in my testing of super-convergence and attempt to only halve the training time.

Implementation of 1cycle follows the suggestion of fast.ai [261], in which the halves of the cycles are half-cosine functions, rather than linear interpolations. This can be expected to offer the same advantages of the cosine annealing schedule of Ref. [253] and also provide a smooth transition between the directions of parameter evolution. The total length of the cycle was set to 135 folds (approximately half that required for the solutions of Sec. 9.4.6 and Sec. 9.4.8). The lengths of the first and second parts of the cycle were set to 45 folds and 90 folds, respectively, i.e. a ratio of 1:2, allowing a longer time for final convergence. Based on a LR range test performed at a constant β_1 of 0.9 and the knowledge that β_1 would be at a minimum when the LR is maximised, a slightly higher maximum LR of 1×10^{-2} and an initial value of 1×10^{-4} were chosen.

⁵Techniques, or aspects of the data, model, or task, which limit overfitting of the model to the training data or aim to improve the score on the testing dataset. See Ref. [260] for an in depth discussion.

⁶Such as L_2 [179] or Dropout [187].

As shown in Fig. 9.10, during the first part of the cycle, the LR is increase from its initial value (1×10^{-4}) to its maximum of 1×10^{-2} while β_1 is decreased from 0.95 to 0.85. In the second half, β_1 returns back up to 0.95, but the LR is allowed to tend to zero. As shown in Row “-SWA +1cycle” of Tab. 9.4, although 1cycle provides the same level of MAPA performance as SWA, it is able to reach this level of performance in half the time.

9.4.10 Densely connected networks

An investigation into different connections between layers, which is found to provide a mild improvement in performance whilst reducing the number of free parameters in the networks.

The fully-connected architectures used so far aim to learn a better representation of the data at each layer, however this means that potentially useful information is lost after each layer if it has not been sufficiently well represented by the latest layer. Being able to do so requires that the layers have a sufficient number of free parameters to avoid losing useful information. Due to limited training data, this limits the depth of the networks which can be trained and so potentially limits the performance of the model.

Consider an example as illustrated in Fig. 9.11 for a set of inputs a, b, c, d, e, f in the case in which it is optimal to learn first a new feature based on inputs a, b, c, d, e using hidden layer 1 (e.g. the angle between the tau and the electron or muon) and then use hidden layer 2 to combine this new feature with input f (e.g. the angle multiplied by the amount of missing transverse energy). Hidden layer 1 must now learn either an identity representation of input f or a monotonic transformation of it, which requires “budget” from both the available free parameters and the amount of training data. It would be more efficient if instead hidden layer 2 were to have direct access to input f . Consider further the case in which it is hidden layer 3 that requires access to input f , now both hidden layer 1 and 2 must learn identity representations of f .

In their 2016 paper, “Densely Connected Convolutional Networks” [262], Huang and Weinberger present an architecture in which, within subgroups of layers, the outputs of all previous layers are fed as inputs into all subsequent layers via channel-wise concatenation. This means that lower-level representations of the data are directly accessible by all parts of the block. This potentially allows both a more diverse range of features to be learnt, and for layers to be trained via *deep supervision* [263] due to more direct connections when back-propagating the loss gradient. Additionally, it means that the weights which were previously required to encode the low-level information may no longer be necessary.

Whilst the paper presents this dense connection in terms channel-wise concatenation of the outputs of convolutional filters, the same idea can be applied to fully connected networks by concatenating width-wise the output tensor of each linear layer with the that layer’s input tensor; i.e. $x_{i+1} = D_i(x_i) \oplus x_i$, where x_i is the hidden state before the i^{th}

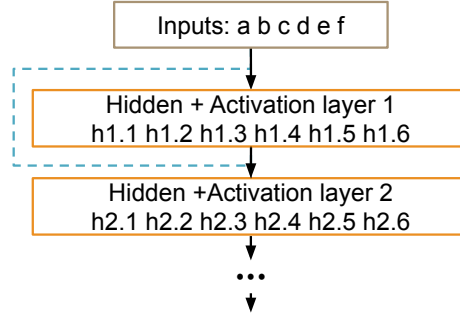


Figure 9.11: A hypothetical scenario in which it is optimal for hidden layer 2 to have access to both a learned representation of the inputs from hidden layer 1 and one or more of the original inputs. Hidden layer 1 must now learn an identity representation of the required inputs in order for hidden layer 2 to have access to the information it requires. A more efficient approach is for hidden layer 2 to access both the hidden state from layer 1 and the original inputs, as indicated by the dashed line.

linear layer (D_i), as illustrated in Fig. 9.12.

This concept can also be thought of as accumulating a tensor of hidden states, as illustrated in Fig. 9.13, in which each hidden layer takes as input the current state of the accumulated tensor and then concatenates its output to the tensor. In this way every subsequent hidden layer has direct access to all previous representations of the inputs, meaning that no identity representations need to be learnt and hidden layers are more easily optimised due to the more direct paths for the loss gradient during back-propagation. As discussed in Ref. [252], in the context of residual connections, such “skip connections” also help mitigate the effects of over-parameterising a model; rather than forcing redundant layers to learn identity mappings, they can instead be skipped (ignored) or pushed towards a zero output. This can be expected to offer some level of leniency when choosing the number of hidden layers to used in the network.

Testing

The architecture up to now has used four hidden layers, each with 100 neurons. Including the embedding and output layers, the total number of free parameters is 33 813⁷. For the densely connected architecture I reduce the width of linear layers to 33 (the number of continuous inputs plus the width of the categorical embedding output), and increase the depth of the network to six hidden layers. The outputs of each hidden layer except the

⁷Computable thusly: Embedding matrix $(4 \times 3) = 12$ + first fully-connected hidden layer of 100 neurons and 33 inputs $((33 \times 100) + 100 = 3400)$ + three subsequent 100 neuron fully-connected hidden layers $(3 \times ((100 \times 100) + 100) = 30\,300)$ + single-neuron output layer $(100 + 1) = 101$. 33 813 parameters total.

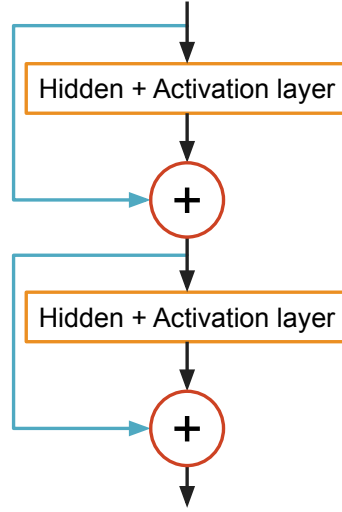


Figure 9.12: Illustration of a densely-connected, non-convolutional network. $\oplus$ indicates concatenations of hidden states.

last are concatenated with their inputs to compute the input tensor to the next hidden layer. This results in a total of 23 113 free parameters ⁸, i.e. about two thirds of the original size. This architecture is illustrated in Fig. 9.14.

Training of the networks used the same 1cycle schedule as presented in Sec. 9.4.9. As shown in Row “+Dense” of Tab. 9.4, the densely connected architecture provides mild improvements in both MVAC and MAPA, despite being significantly smaller, however due to the extra depth and concatenation operations, training and inference timing is approximately the same as the previous architecture.

Unfortunately, a mistake entered into the code during this experiment that led to the ensemble of trained models being uniformly weighted, rather than performance weighted (as per Sec. 9.4.3) as intended. This mistake was only discovered during an ablation study performed at the end of the analysis after the private AMS values had been computed. In order to reflect the information seen during the course of the study, and not to draw conclusions from the private AMS, Tab. 9.4 shows the performance of the uniformly-weighted ensemble, which achieves a MAPA of 3.82 ± 0.02 . A retraining using performance-weighting achieves a MAPA of 3.81 ± 0.01 (comparable with the 3.81 ± 0.02 achieved by the solution of Sec. 9.4.9), which might have caused me not to use dense connections: use of dense connections is slightly slower to train, but the slight reduction in variance of performance would have been seen as favourable. This mistake remained in the code

⁸Computable thusly: Embedding matrix (4×3) = 12 + six fully-connected hidden layers of 33 neurons with input multiplicities scaling according to $33 \times n$: $\sum_{n=1}^6 [((33 \times n) \times 33) + 33] = 23067$ + single-neuron output layer ($33 + 1$) = 34. 23 113 parameters total.

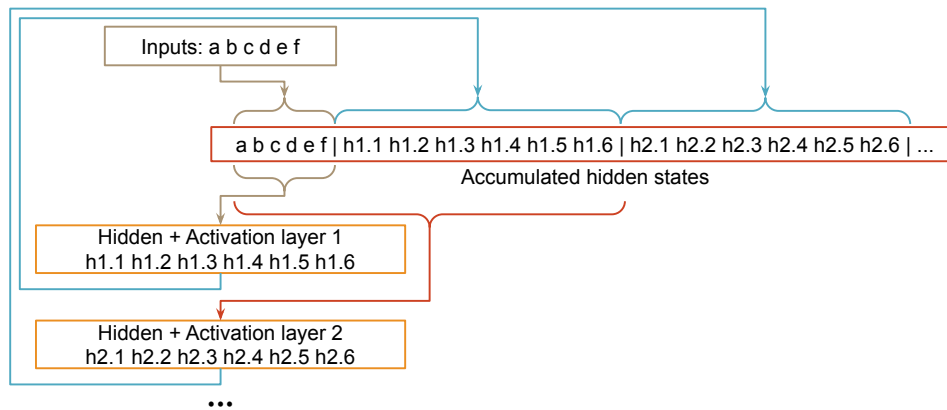


Figure 9.13: An alternative visualisation of a densely-connected network in which hidden layers append to an ever-growing accumulation of hidden states, taking as input the current version of the accumulation.

throughout Sec. 9.4.12 and App. 9.4.11, meaning that the comparisons of metrics to what was seen in this section are still valid.

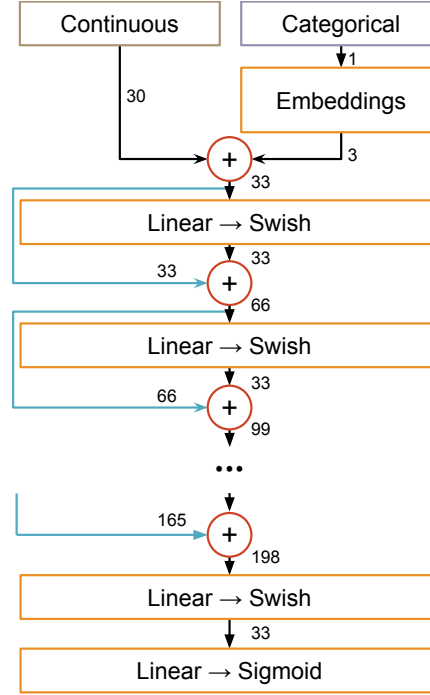


Figure 9.14: Illustration of the architecture described in Sec. 9.4.10. $\oplus$ indicates concatenations of hidden states and the numbers show the widths of the hidden states at each point. A total of six linear layers are used, each with 33 neurons. Two of these layers are compacted into “...”. As a point of reference, this is the architecture that is used to get the final results of this part of the investigation.

9.4.11 Parallel networks and feature splitting

An investigation into more complex architectures which split the input features along separate processing paths. Whilst this section does not result in an improvement, it does illustrate how these architectures can be used to further interpret the data.

So far I have treated all the input features in the same manner (aside from embedding the jet multiplicity) and passed them all through the networks together. However, in particle physics we have lower-level features, such as final-state momenta, but we often compute several high-level features, which are non-linear combinations of low-level features, such as the Higgs mass, that should theoretically be useful variables for the task at hand. Indeed, the Higgs ML dataset contains both high (DER) and low (PRI) features.

As was shown in Fig. 9.2, the high-level features in the data already demonstrate a good degree of class separation compared to the low-level features. Because of this, it is possible that I could reduce the number of free parameters of the model; although the

low-level features might require a deeper network to learn a useful representation for them, perhaps the high-level features might simply require a single layer to encode them. Such architectures have been referred to in literature as *Wide & Deep learning*, e.g. Ref. [264].

It is likely, however, that the optimal model will require some interactions between the high- and low-level features. I can consider two extra architectures in addition to completely splitting the features:

In the first architecture, all of the input features are passed through the deep network as normal, but additionally the high-level features are passed through a single layer which is then concatenated with the output of the deep block. In this case the single layer can encode easily the high-level information, allowing the deep block to learn to represent better the low-level information and to capture the interactions between the low- and high-level features.

In the second architecture, the low- and high-level features are split, but two *bottleneck* layers are used to encode a low-dimensional representation of the sets of features. The outputs of the bottleneck layers are then concatenated with their opposite set of features to provide inputs to each block: i.e. the compressed representation of the high-level features is concatenated with the low-level features and passed to the deep block, and vice versa. In this case, the bottleneck layers should learn the most useful representation of the information which each network block would otherwise not have available, whilst still allowing the number of free parameters in the model to be comparatively small.

Testing

Three architectures were tested:

1. Full-split: The low-level features were passed through a six-layer-deep, 20 neuron-wide, densely connected set of hidden layers, and the high-level features were passed through a single 50-neuron-wide layer. This results in 9303 free parameters.
2. Full-split+bottlenecks: Each set of low- and -high level features are passed through single neuron bottleneck layers. The low-level features, concatenated with the high-level-bottleneck output, were passed through a six-layer-deep, 21 neuron-wide, densely connected set of hidden layers, and the high-level features were concatenated with the low-level-bottleneck output and passed through a single 51-neuron-wide layer. This results in 10 272 free parameters.
3. Semi-split: Both the low and high level features are passed through a six-layer-deep, 33 neuron-wide, densely connected set of hidden layers, and additionally the high-level features were passed through a single 50-neuron-wide layer. This results in 23 863 free parameters.

In all cases, Swish activation functions were placed after every linear layer, and the outputs of the wide and deep blocks were then concatenated and passed through the output

neuron. Figure 9.15 illustrates the three network architectures. For reference the current solution as of Sec. 9.4.10 uses 23 113 free parameters.

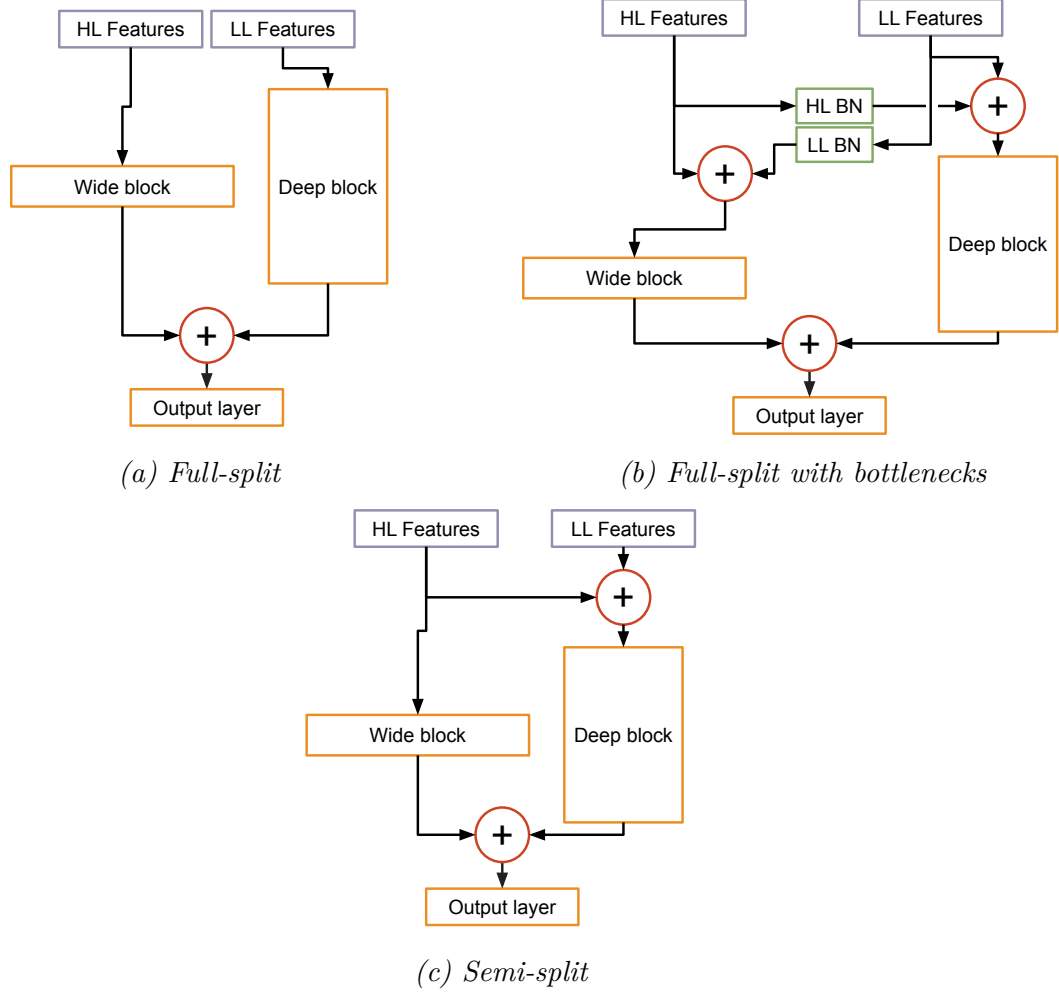


Figure 9.15: Architecture diagrams for the parallel networks considered. $\oplus$ indicates concatenations of hidden states, BN stands for “bottleneck”, and HL and LL stand for “high-level” and “low-level”, respectively.

From Tab. 9.7, we can see that completely splitting the features severely reduces model performance, but that allowing the network to learn compressed representations of the features goes some way to recover performance. We can also see that the addition of the linear embedding for high-level features, offered by the semi-split architecture does not provide any improvement over the current solution, and only serves to increase the model size and timings.

Setup	MMVA	MVAC	MAPA	Fractional time-increase	
				Training	Inference
Current	3.95 ± 0.04	3.89 ± 0.05	3.82 ± 0.02	-	-
Full-split	3.69 ± 0.06	3.64 ± 0.06	3.596 ± 0.006	0.02 ± 0.03	-0.10 ± 0.04
Full-split+BN	3.80 ± 0.05	3.72 ± 0.05	3.647 ± 0.006	0.14 ± 0.05	0.00 ± 0.05
Semi-split	3.93 ± 0.05	3.86 ± 0.05	3.82 ± 0.02	0.12 ± 0.02	0.1 ± 0.1

Table 9.7: Comparison of the various parallel network architectures. “Current” refers to the solution as of Sec. 9.4.10. ‘BN’ in this case stands for BottleNeck. The best values for each metric are shown in bold, and the setup chosen is also indicated in bold.

Interpretation

Whilst it was found that none of the parallel architectures provided improvements over the current solution, they do allow for several informative interpretation methods due to the fact that the features are being split into subgroups.

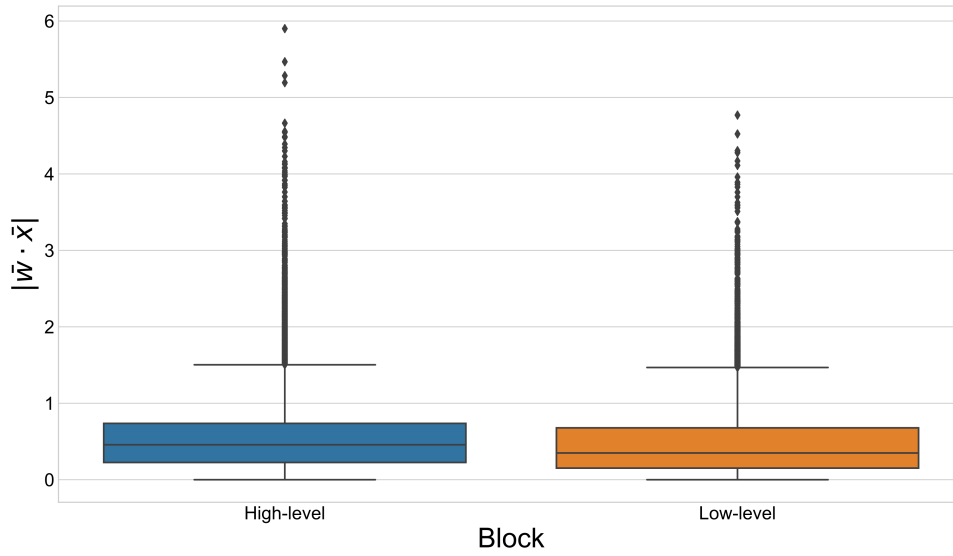
Since the output layer in each model consists of a single neuron which directly takes its inputs from the wide and deep block outputs, we can compute the absolute values of the dot products of slices of the output neuron’s weights ($\bar{w}$) with the corresponding inputs ($\bar{x}$) from each of the wide and deep blocks for a range of data points. This then gives an idea of the reliance of the model’s prediction on particular subsets of inputs.

From Fig. 9.16 we can see that the “Full-split” architecture shows a slightly higher dependence on the single layer fed with the high-level features, but the fact that this difference in dependence is only minor suggests that the deep block is learning a powerful representation of the low-level features. The “Semi-split” model, on the other hand, shows a strong reliance on the deep block (which contains both high and low features), confirming the result that the single-layer embedding of the high-level features does not add much to the model performance.

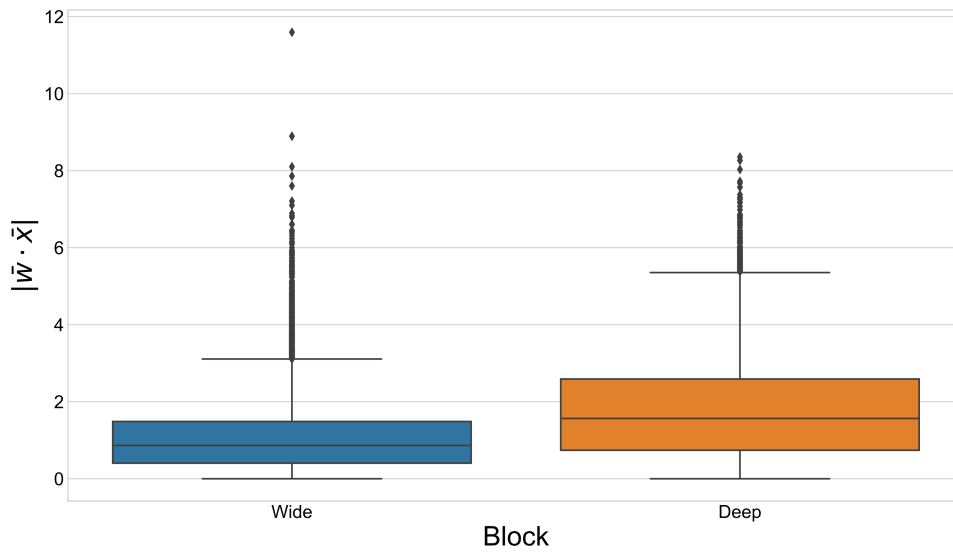
In a similar fashion, the bottleneck layers of the “Full-split+bottlenecks” architecture can be interpreted by computing the absolute values of the weights (w_i) times the feature values (x_i , for feature i) for a range of data points. Comparing Fig. 9.17b to Fig. 9.3, we can see that whilst the high-level bottleneck representation does rely heavily on the two most important features (the MMC mass and the transverse mass), the most used feature is the $p_{t,h}$, which has comparatively lower importance (as shown in Fig. 9.3). This is perhaps an indication that the deep block is being used to refine this weaker feature by combining it with low-level information, or the response of the deep block is being parametrised according to the Higgs p_T . It is also interesting to note in Fig. 9.17a that the low-level bottleneck appears to be used for passing information about jet multiplicity and missing transverse energy; features which could potentially be used to parametrise

the response of the wide layer to the jet-related features.

As a final stage of interpretation of the bottleneck layers, we can plot the distributions of the their outputs for a range of data points (Fig. 9.18). We can see that both bottlenecks learn features which display decent separation between the data classes.

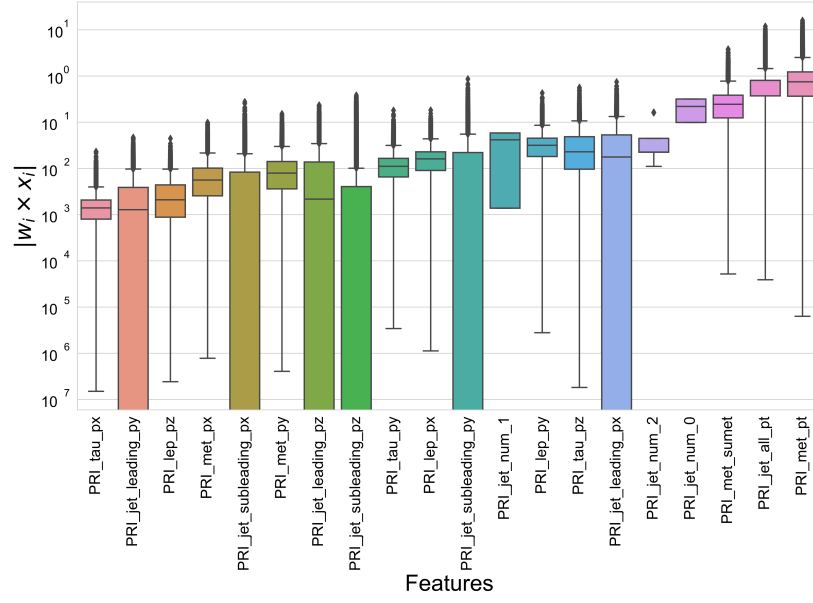


(a) Model reliance on high-level and low-level features as processed by the “Full-split” architecture.

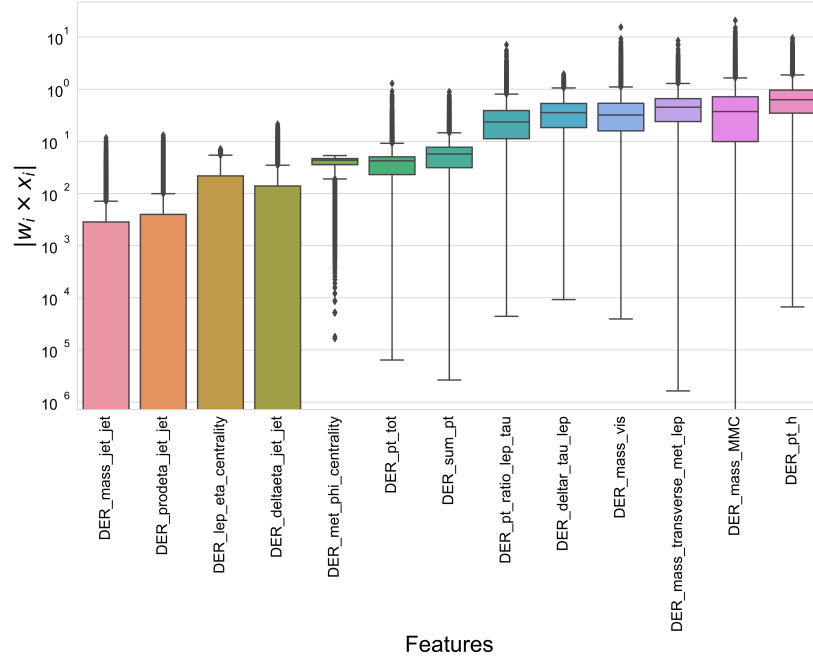


(b) Model reliance on high-level and high+low-level features as processed by the “Semi-split” architecture.

Figure 9.16: Model reliance on subsets of features. Model reliance is computed as the absolute values of the dot products of slices of the output neuron’s weights ($\bar{w}$) with the corresponding inputs ($\bar{x}$) from each of the wide and deep blocks for a range of data points.

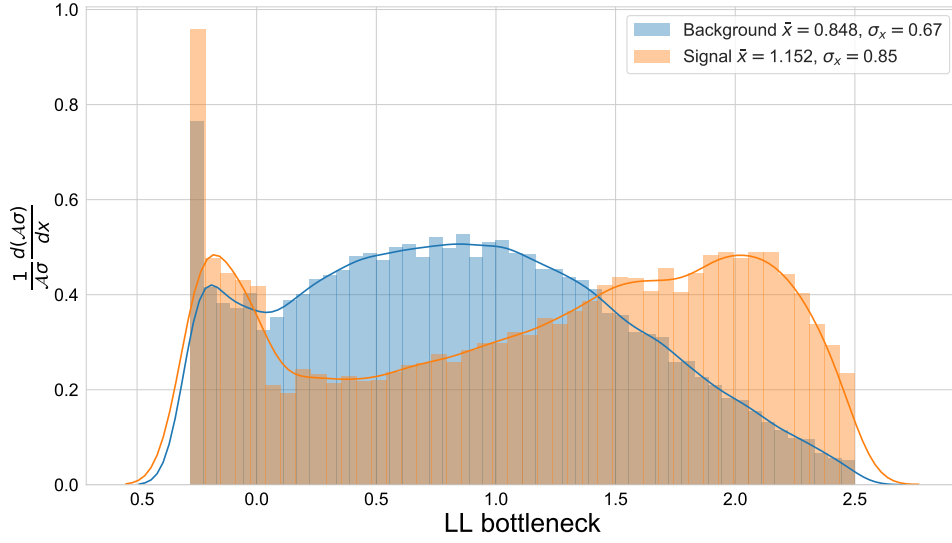


(a) Low level

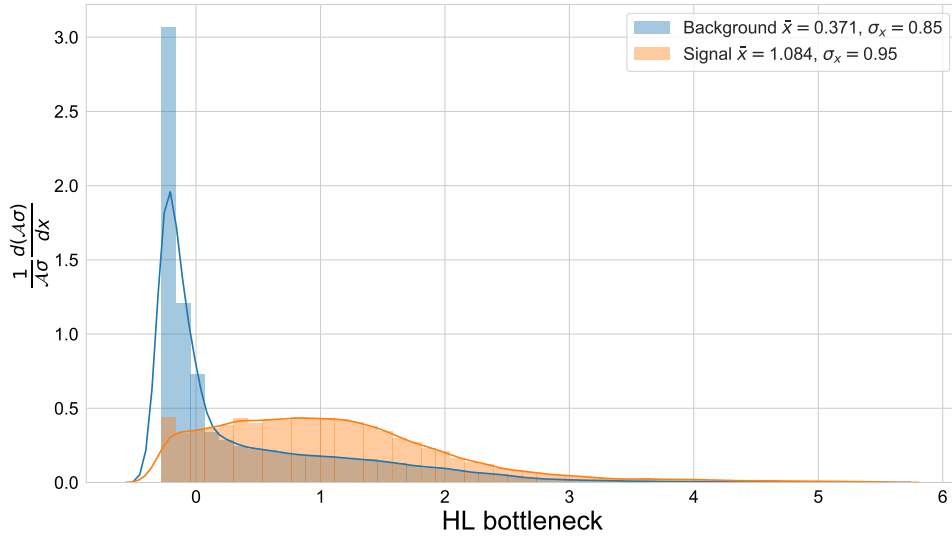


(b) High level

Figure 9.17: Bottleneck reliance on individual features. Bottleneck reliance is computed as the absolute values of the weights(w_i) times the feature values (x_i , for feature i) for a range of data points.



(a) Low level



(b) High level

Figure 9.18: Output distribution of the bottleneck layers separated by event class. These are the outputs after the activation layers of the bottlenecks. It is interesting to note the “double-peak” distribution of the low-level bottleneck output; further inspection revealed that the peak close to zero was only present in events containing at least one jet, further supporting the hypothesis that the low-level bottleneck learns to pass jet-multiplicity information to the wide-block.

9.4.12 Architecture optimisation

In this section I perform a search over the remaining hyper-parameters to finalise my solution. Interestingly, though, the existing architecture is found to work best and the model displays only a weak dependence on its hyper-parameters.

Overview of remaining hyper-parameters

The choices made so far equate to rough optimisations of hyper-parameters and training-method, but the underlying architecture of the network has remained relatively constant. For the final stage of solution optimisation, I explore the impact of adjusting the remaining hyper-parameters: number of hidden layers (depth), number of neurons per hidden layer (width), dropout rates (DO) [187], and the amount of L_2 regularisation [179].

In order to reduce the dimensions of the architecture space the following choices were made:

- Due to the depth of networks explored, batch normalisation [165] was not considered.
- Having already performed feature selection (Sec. 9.4.2), L_1 regularisation [265], which encourages feature sparsity, was not considered.
- Rather than adjusting the widths of each layer individually, the widths were allowed to increase or decrease at a constant rate according to the depth of each layer (the *growth rate*), i.e. $w_d = w(1 + dg)$, where w_d is the width of the hidden layer at depth d (zero-ordered), w is the nominal layer width, and g is the growth rate. A growth rate of zero corresponds to constant widths for all layers.
- Dropout layers, when used, were placed after every activation function except the output, and a single rate was used for all dropout layers.
- The same value of L_2 was used for all weight updates.
- Training will use the 1cycle schedule with the same settings used in Sec. 9.4.9.
- All hidden layers are densely connected as per Sec. 9.4.10.

Whilst other forms of layer-width growth (quadratic, cubic, *et cetera*) could be considered, I assume that the main issue is to have *some* ability to change the layer widths, rather than the exact parametric form; linear growth offers the simplest form, introducing only one extra hyper-parameter. It should also be mentioned that in initial testing of various architectures, the optimal learning rates were all found to be at about the same point (1×10^{-2}), so the same LR was used for all parameter sampling, in order to reduce the search space by one dimension. The LR can of course be adjusted later for any promising parameter points found.

Parameter scan

While some choices have allowed a reduction of the number of dimensions of the parameter search space, the remaining space still has five dimensions (depth, width, dropout rate, L_2 , and growth rate), meaning a full grid-search of all possible combinations is unfeasible; instead a random search will be used. For each architecture, the following parameter-sampling rules were used:

- Depth sampled from $[2, 9) \in \mathbb{Z}$
- Width sampled from $[33, 101) \in \mathbb{Z}$
- Growth rate sampled from $[-0.2, 1) \in \mathbb{R}$
- L_2 sampled from $\{0, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\} \times 10^{-5}$
- Dropout rate sampled from $\{0, 0.05, 0.1, 0.25, 0.5\}$
- The total number of free parameters must be in the range $(5 \times 10^3, 1 \times 10^5)$
- In the case of negative growth rate, the width of the last hidden layer before the output layer must be greater than one.

Just over 350 parameter sets were tested. Each test consisted of training an ensemble of three networks with the specified architecture. The MVAC and MAPA were then calculated and recorded. Figure 9.19 illustrates the distributions of the sampled parameters. Figure 9.20 shows MAPA and MVAC as a function of the total number of free parameters in the networks. From the flat linear-fit, we can see that performance is unrelated to model complexity, and that whilst the data and training methodologies are sufficient for optimising large models, such models are unnecessary to reach better performance. This is possibly due to the leniency on over-parameterised networks discussed in Sec. 9.4.10.

The testing results were then used to fit Gaussian process models in order to evaluate the partial dependencies of the metrics on the hyper-parameters. Figure 9.21 illustrates the one- and two-dimensional partial dependencies of MVAC and MAPA on the hyper-parameters. Note that the machinery used expects a minimisation problem, hence why the negatives of the metrics are shown.

From the results, the trends in the partial dependencies of both MVAC and MAPA appear to favour minimal explicit regularisation. In terms of depth and width, MVAC shows weak dependence on width and growth rate, and slight dependence on depth, with networks with fewer than four hidden layers performing poorly compared to deeper networks. MAPA again shows weak dependence on width and growth rate, but demonstrates a bounded minimum for depth, favouring networks with around five hidden layers.

Architecture	MMVA	MVAC	MAPA	Fractional time-increase	
				Training	Inference
Current	3.95 ± 0.04	3.89 ± 0.05	3.82 ± 0.02	-	-
4 100 0.1	3.9 ± 0.05	3.83 ± 0.05	3.74 ± 0.01	0.23 ± 0.09	0.6 ± 0.1
4 40 0.45	3.96 ± 0.05	3.90 ± 0.05	3.82 ± 0.01	-0.02 ± 0.05	-0.01 ± 0.03
5 100 -0.1	4.03 ± 0.05	3.91 ± 0.05	3.80 ± 0.02	0.15 ± 0.07	0.6 ± 0.1
5 33 0.9	3.95 ± 0.05	3.89 ± 0.04	3.81 ± 0.02	0.19 ± 0.08	0.6 ± 0.2
8 33 -0.2	3.95 ± 0.06	3.89 ± 0.06	3.82 ± 0.02	0.4 ± 0.1	1.2 ± 0.4

Table 9.8: Comparison of the various explored architectures in terms of performance metrics. “Current” refers to the solution as of Sec. 9.4.10. Architecture names are of the form: [depth] [width] [growth rate], e.g. “4 100 0.1” has 4 layers, a base width of 100 neurons and a growth rate of 0.1. The best values for each metric are shown in bold, and the setup chosen is also indicated in bold.

Further testing

Based on the scan results and partial dependencies, a number of promising parameter sets were further explored with complete trainings, and the LRs set using LR range tests. The performances of these architectures are compared to the current solution (depth 6, width 33, growth rate 0, and no dropout or weight decay) in Tab. 9.8. As can be seen, none of the new architectures improve on the current architecture in terms of MAPA and although some are able to reach the same level of performance they do not provide significant reductions in terms of training or inference time.

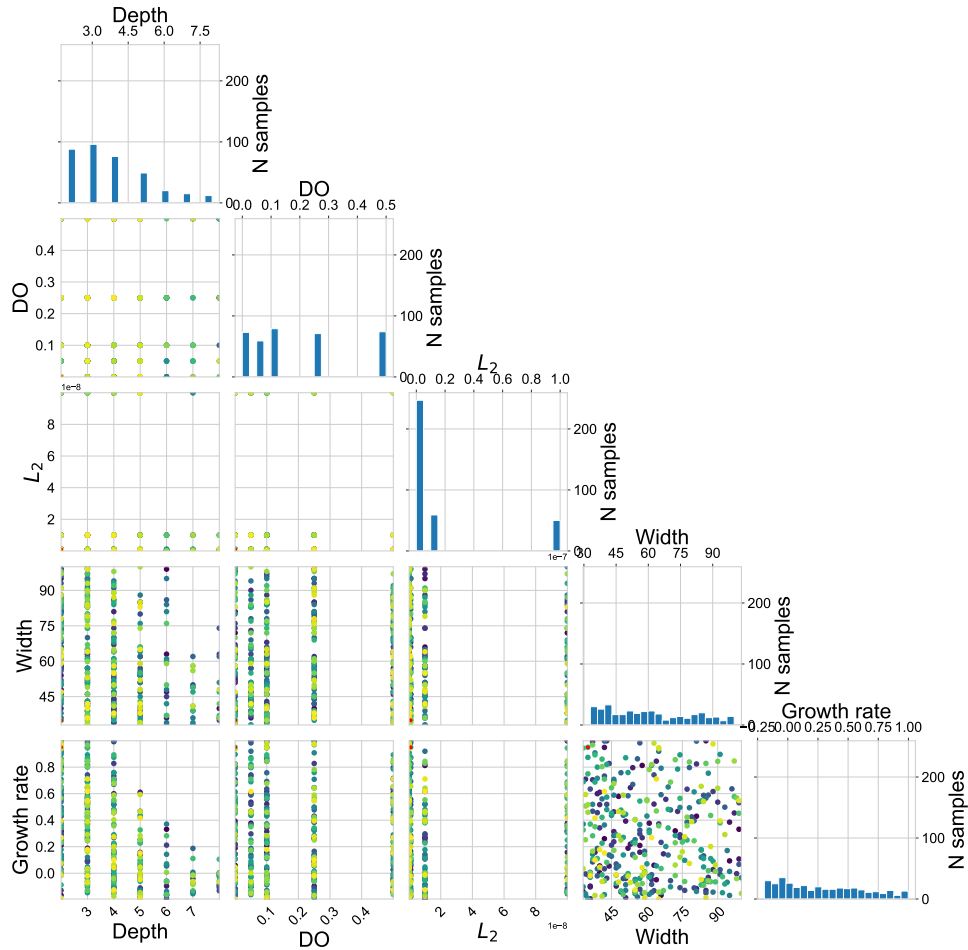


Figure 9.19: Illustration of the parameter sets sampled during testing. Point colour indicates order in which points were sampled (random). Sampling is approximately flat for each hyper-parameter, except for the depth of the networks; the limitation on number of parameters acts to suppress deeper architecture with high widths.

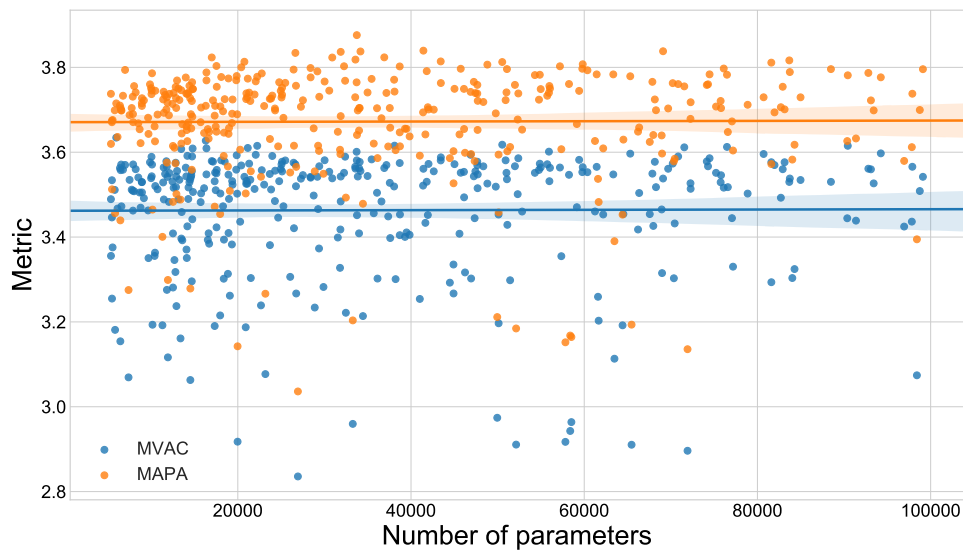
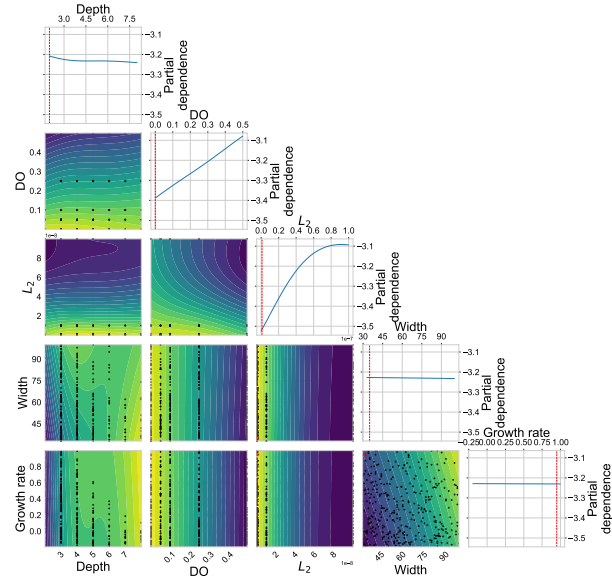
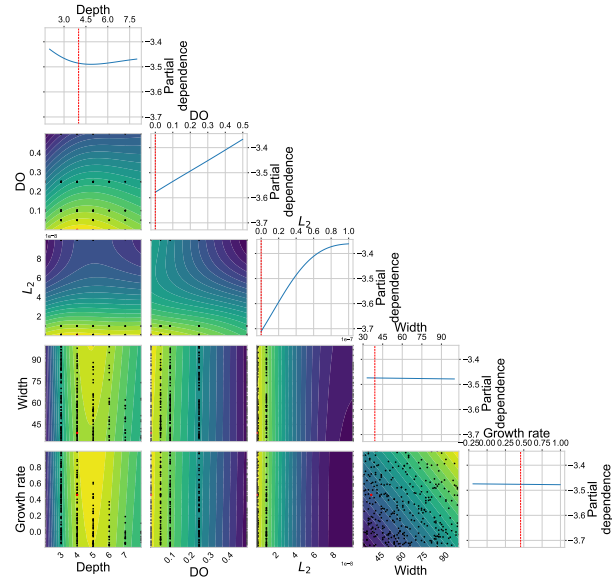


Figure 9.20: Dependence of the optimisation metrics (y -axis) on the total number of free parameters of the models sampled during testing (x -axis). Metric values show a flat dependence on parameter-count, indicating that larger DNNs are not necessary to achieve better performance.



(a) MVAC



(b) MAPA

Figure 9.21: Partial dependencies of negative MVAC and negative MAPA on network hyper-parameters (diagonal plots - lower partial dependence = better performance) and the 2D interaction between hyper-parameters (sub-diagonal plots - yellow region = better performance). Black points indicate the parameter-sets sampled, and the red points and lines indicate the position of the best parameter set found during sampling.

9.5 Solution evaluation

9.5.1 Test results

Having built up a solution using several optimisation metrics and timing changes as guides, I now come to checking the performance of the solution using the remaining metrics: the AMS on the public and private portions of the testing data (reminder: MAPA was only an average over subsamples of the public portion of the testing data). In Fig. 9.22 I show all metrics as a function of solutions developed, averaged over the six trainings that were run per solution. From the results, we can see that whilst MAPA consistently over-estimated actual performance, its general trend is nonetheless representative of changes in the public and private AMS.

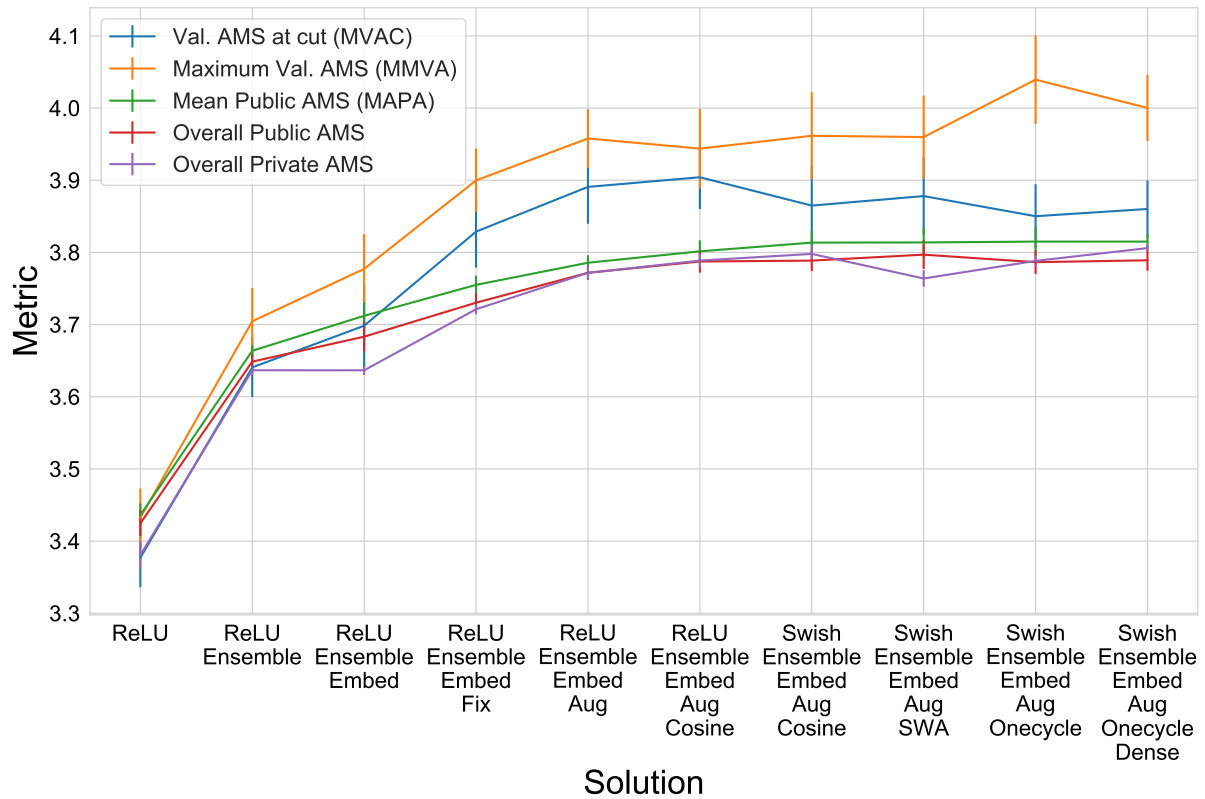


Figure 9.22: Illustration of the performance metrics (MVAC, MMVA, and MAPA (see Sec. 9.3.6)), and the scoring metrics (public and private AMS) for each solution developed.

9.5.2 Solution timing

Also of interest for real-world application is the time taken by each solution. Plotted in Figs. 9.23 and 9.24 are the absolute (wall clock) and relative timings for each solution for the variety of hardware tested, as computed by the `timeit` package. Note that one solution was run per hardware configuration, except for the quad-core Xenon setup, which shows the mean of two different machines with the same hardware specification. The numbers in parentheses indicate the number of cores and number of threads per core, respectively, except for the Xenon machines, which are cloud-based and are allocated virtual CPUs. From these results, we can see that whilst the GPU is clearly superior, the architectures run sufficiently well on CPU to be viable for use in a particle-physics analysis.

It is also interesting to note the fractional test-time increase for the GPU, which appears to show sub-linear scaling when ensembling, but poor scaling when running inference-time augmentation. This is because LUMIN currently loads each data-fold from the hard-drive into either RAM or VRAM sequentially, and the timing includes this. Once in memory, however, all models in the ensemble are run over the fold. Since the load time for VRAM is non-negligible compared to the prediction time, the fractional increase when evaluating the ten models is much less than nine. Additionally, LUMIN currently applies data-augmentation when loading the data fold, rather than augmenting the data in-place, meaning that the load time is incurred an extra seven times when running TTA, hence the apparent poor scaling when running TTA on a GPU.

9.5.3 Discussion

Of the solutions developed, the one from Sec. 9.4.10 (densely connected network with 1cycle training scheme) offered the highest value for MAPA seen during solution development, and is also much quicker to train compared to solutions that used constant or cosine-annealed learning-rate schedules. Its performance with unweighted ensembling is public : private AMS (3.80 ± 0.01 : 3.803 ± 0.005), and with weighted ensembling (3.79 ± 0.01 : 3.806 ± 0.005), see Sec. 9.6.2 for further discussion on this.

As can be seen from the architecture-search results, no explicit regularisation was needed for this problem, and performance was only slightly dependent on the widths of the hidden layers. The rule-of-thumb used to set the initial layer width for the densely connected network (width number = of inputs features (continuous + categorical embeddings)) seems to be appropriate, but further work will be necessary to see how well this rule generalises to other problems and applications.

A comparison to the solutions developed during the running of the challenge is presented in Tab. 9.9. In terms of hardware, for the solutions developed here, the GPU results are for a Nvidia GeForce GTX 1080 Ti, and the CPU result is for a Intel Core i7-8559U (2018 MacBook Pro). RAM and VRAM usage for each setup was found to be

less than 1 GB. Details of the other solutions are as follows:

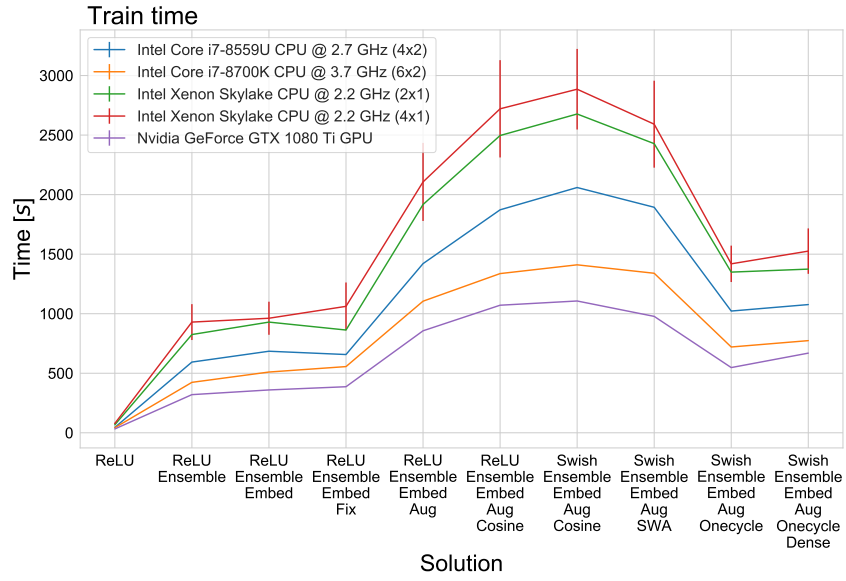
1. The 1st place solution (available at <https://github.com/melisgl/higgsml>) primarily used a setup with a Nvidia Titan GPU and 24 GB RAM, but Ref. [266] also details timings for an Amazon EC2 m2.4xlarge instance (8-vCPU & 15 GB RAM);
2. The 2nd place solution (<https://github.com/TimSalimans/HiggsML>) was run on an Amazon EC2 m2.4xlarge instance (8-vCPU & 64 GB RAM) and notes the high RAM requirements of the training;
3. And the 3rd place solution used a 2012 quad-core laptop (<https://www.kaggle.com/c/higgs-boson/discussion/10481>).

The difference in hardware makes a direct comparison of solution timing difficult, but we can try to scale the timings: The 1st place solution used a Nvidia Titan GPU (2013, 1500 gigaflops processing power in double precision) and my solution uses a Nvidia 1080 Ti (2017, 10 609 gigaflops processing power in single precision). Accounting for the changes in processing power, the 1st place solution should take around 100 min to train on my GPU. My solution therefore provides an estimated speedup of 13x on GPU and 7x on CPU (comparing to GPU). Given the uncertainties involved in the metric it would be reasonable to say that the solution developed here achieves comparable performance to winning solutions of the challenge in just a fraction of the time.

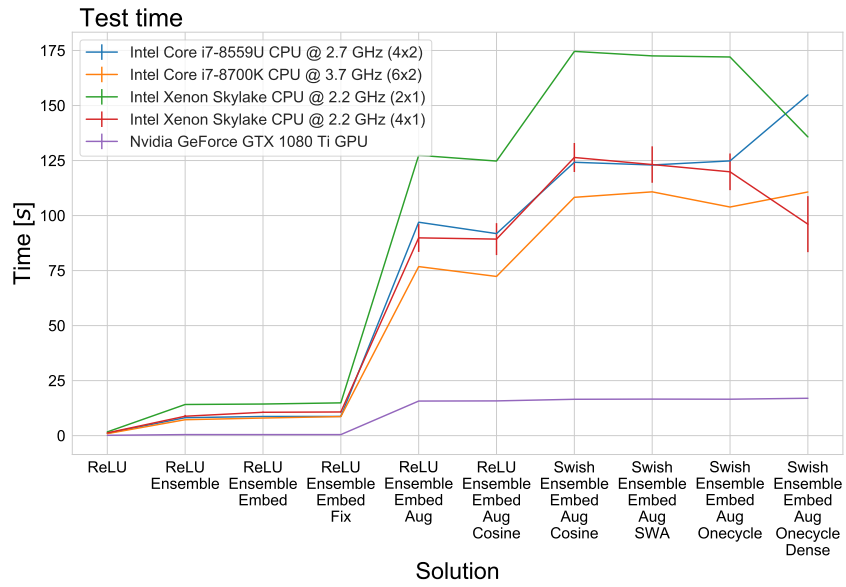
Whilst the Kaggle solutions only report the private AMS scores they achieved (single numbers with no uncertainties) it is of interest to know the general performance offered by each of the solutions. The summary paper for the challenge (Ref. [240]) estimates the statistical significance of the solution ranking by bootstrap resampling of the scored data. The authors find that at a 95 % confidence level, the 1st-place solution does indeed outperform the rest, but that the 2nd and 3rd-place solutions are indistinguishable in terms of performance. Ideally such a test should also be performed to compare my solution to the 1st-place solution, however this requires access to the scored data that was submitted during the competition, which is not publicly accessible.

	My solution	1 st place	2 nd place	3 rd place
Method	10 DNNs	70 DNNs	Many BDTs	108 DNNs
Train-time (GPU)	8 min	12 h	N/A	N/A
Train-time (CPU)	14 min	35 h	48 h	3 h
Test-time (GPU)	15 s	1 h	N/A	N/A
Test-time (CPU)	3 min	???	???	20 min
Score	3.806 ± 0.005	3.80581	3.78913	3.78682

Table 9.9: Comparison with challenge solutions. ‘N/A’ indicates that the solution was not run on a GPU and so no timing information is available. ‘???’ indicates that the solution was run on CPU but no timing information was reported. Timings for “My solution” are rounded to the nearest minute, or five seconds. Note that the training times differ from those shown in Fig. 9.23, which were trained whilst showing a live feedback of the loss to help diagnose any problems, but equated to a constant increase in time per epoch that was independent of the architecture. For the final timings shown here, the live plotting was turned off.



(a) Absolute train-time



(b) Absolute test-time

Figure 9.23: Absolute (wall-clock) timings for solutions on a variety of hardware. Note that the (4x1) Xenon line is an average of two machines, and hence has error bars; only single machines were available for the other hardware configurations and so their associated timing uncertainties cannot be evaluated.

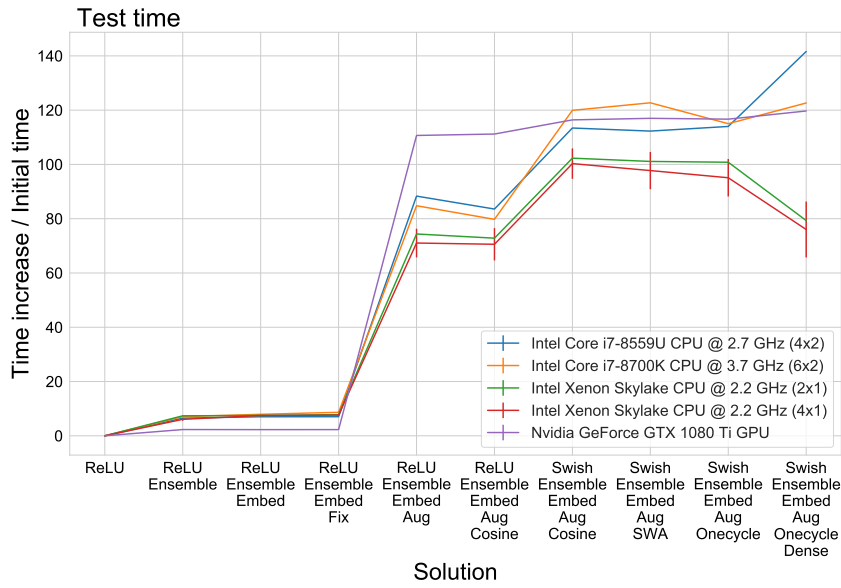
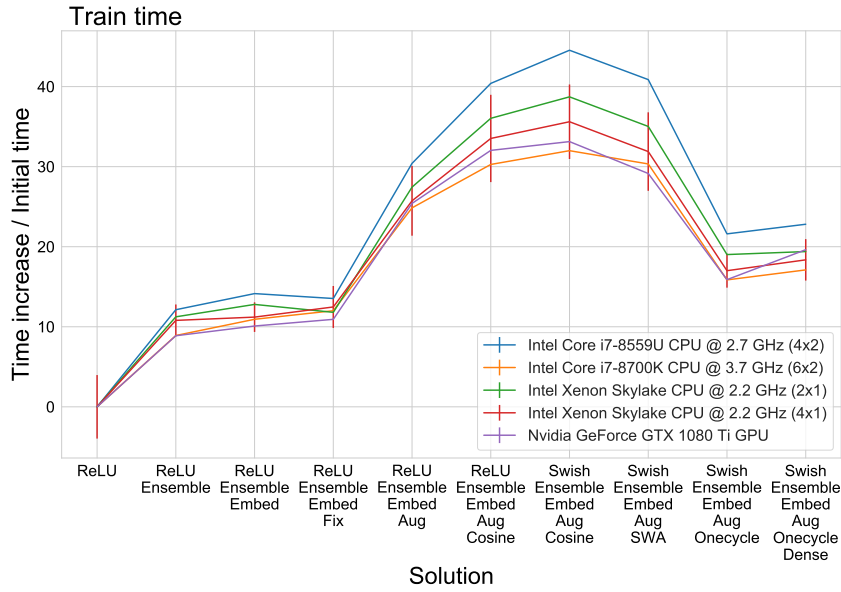


Figure 9.24: Fractional increases in timings of solutions for a variety of hardware, relative to “ReLU”. Note that the (4x1) Xenon line is an average of two machines, and hence has error bars; only single machines were available for the other hardware configurations and so their associated timing uncertainties cannot be evaluated.

9.6 Benefit and ablation studies

9.6.1 Individual benefits

Due to all possible permutations of new methods not being considered, it is plausible that some methods would show a greater or worse relative improvement if testing had been performed in a different order. This could happen for a number of reasons, such as: overlaps in improvements in methods; certain pairs of methods working well together; and the non-linear nature of the scoring metric. Table 9.10 addresses some these reasons by detailing results from retrainings of solutions in which only one new method was added. Whilst all new methods provide at least minor improvements in isolation, it is interesting to see that data augmentation actually provides a greater improvement than ensembling when applied individually, in comparison to Fig. 9.25 (which shows the relative contributions of each new method to the overall improvement in performance), indicating an overlap in the two methods. Also the use of dense connections on their own provides a smaller benefit than they do in the full solution, implying that they are most useful when applied in conjunction with other methods.

9.6.2 Ensemble weighting

In Sec. 9.4.3, performance-weighted ensembling was used. The more simple form of ensembling would have been to instead consider the predictions of each model equally (uniform weighting). The choice to use weighted ensembling was made based on previous experience with similar HEP classification tasks, and was not tested at the time so as to reduce the number of times the performance metrics were computed (reducing the chance of overfitting to the validation dataset). Having completed the study, though, it is of interest to check the impact of weighted versus unweighted ensembling. As discussed in Sec. 9.4.10, the final ensembles were accidentally built using uniform weighting, achieving a private AMS of 3.803 ± 0.005 , which was only noticed when performing this ablation study of the ensemble weighting. Retraining the ensembles using performance-weighted ensembling, as intended, results in a slightly higher private AMS of 3.806 ± 0.005 .

Setup	Mean improvement in private AMS
Baseline + Data augmentation	$7.98 \pm 0.05\%$
Baseline + Ensembling	$7.58 \pm 0.05\%$
Baseline + 1cycle	$4.00 \pm 0.03\%$
Baseline + Swish	$2.22 \pm 0.03\%$
Baseline + Embedding	$1.63 \pm 0.03\%$
Baseline + Dense connections	$1.25 \pm 0.03\%$

Table 9.10: Individual percentage improvements in mean private AMS offered by each new method over a baseline ReLU model. In contrast to Fig. 9.25 the model setups here do not include other methods, e.g. “Baseline + Embedding” does not also include ensembling.

9.7 Conclusion

In this study I have examined several recent advances in deep learning for neural network architectures and training methodologies, and their applicability to a typical problem in the field of high-energy particle physics, using data from a popular data-science competition. Whilst the solutions developed were unable to consistently improve on the performance of the winning solutions, they were able to reproduce their performance in a significantly shorter time period using less computational resources.

The solutions were developed in a systematic manner, rather than considering all possible permutations of hyper-parameters, using a variety of metrics (one of which was unavailable during the competition) and considering the time requirements. The improvements to the performance metric are broken down in Fig. 9.25, where we can see that most of the improvement beyond ensembling (which the winning solutions already used) is coming from domain-specific data augmentation (applied during training and testing), this is particularly interesting as it implies that the models are able to make good use of the low-level information in the data and learn their own high-level representations of it. Learning-rate scheduling provides not only a moderate reduction in train time, but also a small improvement in performance. Finally, using densely connected layers offers a further small improvement in performance as well as reducing the dependence of performance on the widths of network layers, potentially making it easier to find optimal architectures for similar applications.

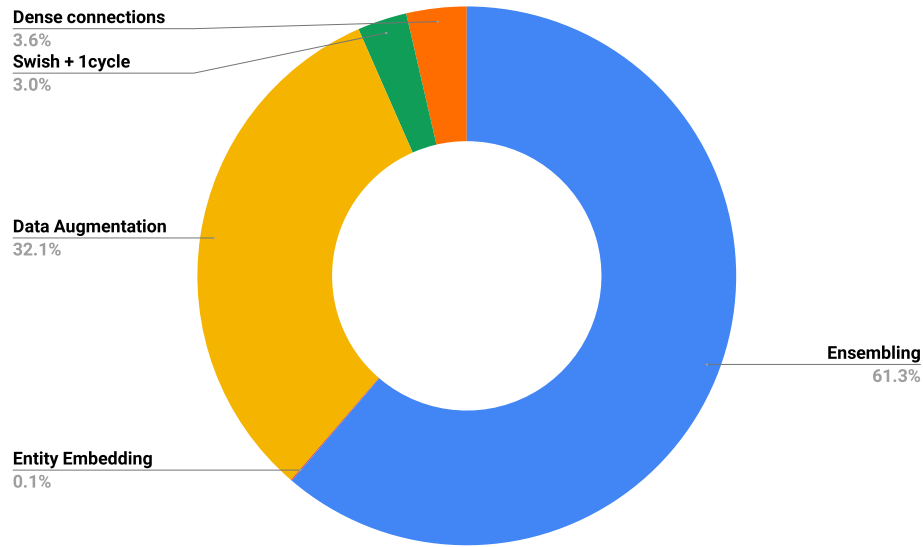


Figure 9.25: Illustration of the relative contributions of each new method to the overall improvement in mean private AMS over a baseline ReLU model. Note that contributions are calculated sequentially in the order tested, i.e. the entity embedding result includes ensembling, the augmentation result includes embedding and ensembling, et cetera.

9.8 Software package: LUMIN

Over the course of the investigations documented in Chap. 7, Chap. 8, and in the initial investigation for this chapter [43], I had built up a reasonably large repository of commonly used code and wrapper functions for implementing deep-learning techniques in KERAS, called HEPML_TOOLS, which is publicly available at https://github.com/GilesStrong/hepml_tools. Whilst KERAS provided me with an accessible introduction to neural networks, this repository had grown to the point that it did not use many of the functions which made KERAS useful, and some of the technique implementations involved interacting with the tensor library (TENSORFLOW) which KERAS was designed to abstract.

The report of my initial study on the Higgs ML Kaggle dataset formed the final deliverable for my role in the AMVA4NewPhysics Marie Curie Innovative Training Network and so prior to entering the fourth year of my PhD I decided to rewrite this repository. In place of KERAS I moved to using PYTORCH, a tensor library similar to TENSORFLOW but one with which I was more familiar, having completed several courses which used it (FastAI: <https://course.fast.ai/>). Interacting directly with the tensor library allowed for greater flexibility in implementing techniques and provided less reliance on

overriding existing functions and classes. Additionally I formed the library into an easily installable Python package, in the hope that it would of use to other members of the scientific research community.

LUMIN [236] (Lumin Unifies Many Improvements for Networks), as I named it, was made available to the public under an open-source licence in March 2019 on Github (<https://github.com/GilesStrong/lumin>) and Zenodo <https://doi.org/10.5281/zenodo.2601857>, and later as a PyPI package <https://pypi.org/project/lumin/>. Documentation is hosted at <https://lumin.readthedocs.io/>.

LUMIN not only provides implementations for a range of advanced techniques for neural network training and architecture, but also methods for statistics and data-processing specific to the domain of high-energy physics. The library can be described by six distinguishing characteristics:

Ensembling Ensembling is an easy way to increase the performance of a model, especially in HEP where datasets are often limited, and models are reasonably quick to train. Ensembling is a core aspect of LUMIN:

- Data files are pre-split to yield ‘folds’ of data with the intention that a model can be trained on $N - 1$ folds and validated on the remaining fold. Figure 9.26a shows the `FoldYielder` class being instantiated. These objects can be used to interface with data that has been pre-split into folds, and are used as input to a range of methods.
- Users do not create a network and train it, but instead define how networks should be created. By passing this definition to the core training-loop function a user-requested number of models can be trained and validated, instead of just one. Figure 9.26b shows how model definitions can be encapsulated in a `ModelBuilder` class, and can then be used to initialise trainable models.
- A dedicated ensemble class provides a convenient way to handle inference, interpretation, saving, loading, and exporting of many networks. Figure 9.26c shows the instantiation of an `Ensemble` class from a set of previously trained models.

```
from lumin.nn.data.fold_yielder import FoldYielder
train_fy = FoldYielder(PATH/'train.hdf5', input_pipe=input_pipe)
train_df = train_fy.get_df(inc_inputs=True, deprocess=True, nan_to_num=True); train_df.head()
```

200000 datapoints loaded

	DER_mass_MMC	DER_mass_transverse_met_lep	DER_mass_vis	DER_pt_h	DER_deltaeta_jet_jet	DER_mass_jet_jet	DER_prodeteta_jet_jet	DER_deltar_tau_lep
0	118.916000	44.963001	83.234001	0.434999	0.0	0.0	0.0	3.324
1	135.817001	26.292000	103.487999	4.588998	0.0	0.0	0.0	3.080
2	190.457993	14.596998	141.067001	7.952001	0.0	0.0	0.0	3.824
3	0.000000	81.817001	82.167999	0.650999	0.0	0.0	0.0	2.471
4	105.120003	10.250002	76.503998	4.723001	0.0	0.0	0.0	3.156

5 rows × 33 columns

(a) The *FoldYielder* class being instantiated, providing access to fold of data stored in *HDF5* format. The data remains out of memory until requested, e.g. by the *to_df* method, which loads the data into a *PANDAS DataFrame* object.

```
from lumin.nn.models.model_builder import ModelBuilder
from lumin.nn.models.model import Model
from lumin.nn.models.helpers import CatEmbedder

bs = 256
objective = 'classification'
model_args = {'body':{'act':'swish', 'width':100, 'depth':4}}
opt_args = {'opt':'adam', 'eps':1e-08}
cat_embedder = CatEmbedder.from_fy(train_fy)

n_out = 1
model_builder = ModelBuilder(objective, cont_feats=cont_feats, n_out=n_out, cat_embedder=cat_embedder,
                             model_args=model_args, opt_args=opt_args)
model = Model(model_builder)
```

(b) The *ModelBuilder* is a class that can be used to describe how models should be defined, via keyword arguments and partial classes. The model definition can then be used to initialise any number of models.

```
from lumin.nn.ensemble.ensemble import Ensemble

ensemble = Ensemble.from_results(results, n_models, model_builder, metric='loss', higher_metric_better=False)
```

Choosing ensemble by loss

100.00% [3/3 00:00]

Model 0 is 0 with loss = 3.1563054108119104e-06
 Model 1 is 1 with loss = 3.2229536373051815e-06
 Model 2 is 2 with loss = 3.3168385016324464e-06

(c) An *Ensemble* class is instantiated from a set of previously trained models, and internally weights them according to their performance. The class can then be used to predict, save, export, et cetera all of the models in a single method call.

Figure 9.26: Examples of classes related to ensembling.

Modularity LUMIN aims to be flexible and adaptable, allowing users to define and use their own classes without heavy ‘hacking’ of the core functions.

- As demonstrated in Fig. 9.27b, networks are also defined in a modular fashion consisting of: a Head, to input and encode data; a Body, to provide the hidden layers of the network; and a tail to construct the output of the network. Models can also have multiple heads, to read different forms of data inputs, and multiple bodies, to provide separate processing of different inputs. The modular nature of the networks allows users to plugin their own classes in specific areas without having to define the entire network class themselves.
- Abstract classes contain core functionality allowing users to easily inherit from them and override only the relevant functions, as shown in Fig. 9.27a.
- Main functions, such as the training loop (e.g. as shown in Fig. 9.27c), take as arguments these classes, meaning that user defined classes can be slotted into existing infrastructure, provided they conform to the expected API calls.
- The usage of ‘callback’ classes at set points of the training and evaluation cycles allows the user further options for customising the model and its training.

```
bs = 256
objective = 'classifier'
n_out = 1

def get_model_builder(fy: FoldYielder, depth: int, width: int, head: AbsHead=CatEmbHead) -> ModelBuilder:
    opt_args = {'opt': 'adam', 'eps': 1e-08}
    cat_embedder = CatEmbedder.from_fy(fy)
    body = partial(FullyConnected, depth=depth, width=width, act='swish')
    tail = partial(ClassRegMulti)
    model_builder = ModelBuilder(objective, cont_feats=fy.cont_feats, n_out=n_out, cat_embedder=cat_embedder,
                                opt_args=opt_args, body=body, head=head, tail=tail)
    print(Model(model_builder))
    return model_builder
```

(a) Here the Head, Body, and Tail of the `ModelBuilder` are being set using dedicated classes. Users may switch out any of these classes for their own, provided the new classes implement the expected core behaviour.

```
from lumin.nn.models.blocks.head import AbsConvIdHead

class ResNext(AbsConvIdHead):
    def get_layers(self, in_c: int, act: str='relu', bn: bool=False, **kwargs) -> Tuple[nn.Module, int]:
        layers = []
        layers.append(self.get_convld_block(in_c, 32, stride=1, kernel_sz=3, act='linear', bn=False))
        layers.append(self.get_convld_resNext_block(32, 4, 4, 32, stride=1, kernel_sz=3, act=act, bn=bn))
        layers.append(self.get_convld_resNext_block(32, 4, 4, 64, stride=2, kernel_sz=3, act=act, bn=bn))
        layers.append(nn.AdaptiveAvgPool1d(1))
        layers = nn.Sequential(*layers)
        return layers
```

```
head = partial(ResNext, vecs=vecs, feats_per_vec=fpv, act='swish', bn=True)
model_builder = get_model_builder(train_fy, depth=3, width=50, head=head)
```

(b) An example of inheritance from abstract classes: A new convolutional head is defined by overriding a key method of an abstract class that provides the necessary functionality for working with convolutional models. That class itself inherits from a class designed for working with matrix-data, which in turn inherits from an abstract class for head modules of the network.

```
from lumin.nn.training.fold_train import fold_train_ensemble

results, histories, cycle_losses = fold_train_ensemble(train_fy, n_models,
                                                       model_builder=model_builder,
                                                       bs=bs,
                                                       callback_partials=callback_partials,
                                                       eval_metrics=eval_metrics,
                                                       max_epochs=max_epochs, patience=patience)
```

Training model 1 / 3, Val ID = 0

(c) An example of training and ensemble of models: three models will be trained and saved. Note that the data is provided via a `FoldYielder` object, and model definitions via a `ModelBuilder` object, as opposed to instantiating a model and training it on data already in memory.

```
from lumin.nn.callbacks.cyclic_callbacks import *
from functools import partial

callback_partials = [partial(OneCycle, lengths=[45, 90], lr_range=[1e-4, 1e-2],
                             mom_range=[0.85, 0.95], interp='cosine')]
eval_metrics = {'AUC': AUC(wgt_name='weights')}
```

(d) An example of preparing callback classes for use in training. `OneCycle` will run a specific schedule for the learning rate and momentum during training. Note that they are defined as partial functions rather than instantiated objects, allowing them to be instantiated on demand without having to reset in internal variables.

Figure 9.27: Modularity.

Modern techniques Easy access to best practices and latest techniques: Whilst LUMIN aims to include state of the art models and techniques, it also aims to abstract away the ‘boilerplate’ code and configuration required for common tasks to lower the barrier of entry for users:

- Many useful techniques are implemented in LUMIN, however they are not enabled by default, to avoid confusing new users and having unexpected behaviour for users experienced with other frameworks. Figure 9.28 gives an example of this.
- Common defaults for problems are configured via a string-based argument, e.g. ‘classification’ will set the default output activation function of the model to be a sigmoid, the default loss function to be (binary) cross-entropy, and other similar changes depending on the function or class to which it is passed.

```
from lumin.optimisation.hyper_param import fold_lr_find
lr_finder = fold_lr_find(train_fy, model_builder, bs, lr_bounds=[1e-5, 1e-1], train_on_weights=True)
```

100.00% [10/10 00:03]

```
Loading embeddings: {0: 'weights/PRI_jet_num.h5'}
Loading embeddings: {0: 'weights/PRI_jet_num.h5'}
Loading embeddings: {0: 'weights/PRI_jet_num.h5'}
Loading embeddings: {0: 'weights/PRI_jet_num.h5'}
Loading embeddings: {0: 'weights/PRI_jet_num.h5'}
Loading embeddings: {0: 'weights/PRI_jet_num.h5'}
Loading embeddings: {0: 'weights/PRI_jet_num.h5'}
Loading embeddings: {0: 'weights/PRI_jet_num.h5'}
Loading embeddings: {0: 'weights/PRI_jet_num.h5'}
Loading embeddings: {0: 'weights/PRI_jet_num.h5'}
LR finder took 3.842s
```

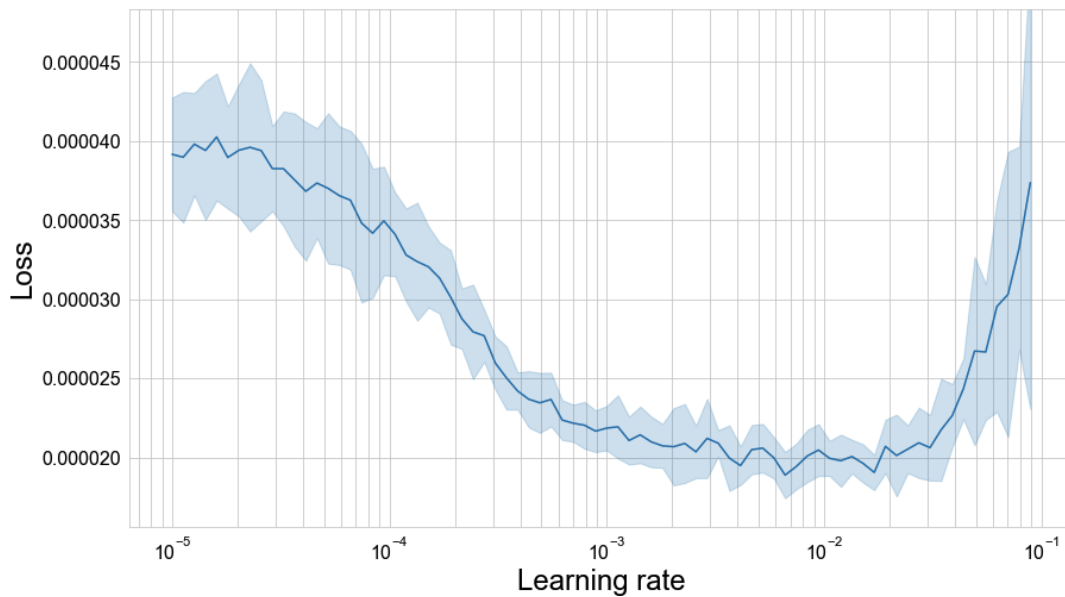


Figure 9.28: An example of the LR-range test [248] used to find the optimal LR for training a particular model and data configuration. This is repeated 10 times to get an average loss curve. Additionally, a pretrained embedding for a categorical feature (the number of jets in an event) is being loaded each time a new model is created. The use of pretrained weights was something explicitly requested during model definition, rather than being added automatically without the user's knowledge. Depending on the LR schedule used (if any) during training, the optimal LR for this setup would probably be between 3×10^{-3} and 3×10^{-4} ; the highest LR for when the loss is still decreasing, but allowing some safety margin.

Universal weight handling Weight handling: HEP data usually contains weights which characterise the acceptance and production cross-section of each event, flatten some distribution such as p_T and η , or balance out classes in training data.

- Relevant classes and functions support weighted data and act accordingly, e.g. during training, plotting, and interpretation (as shown in Fig. 9.29).
- The default losses in PYTORCH are expanded upon to better handle weighted data

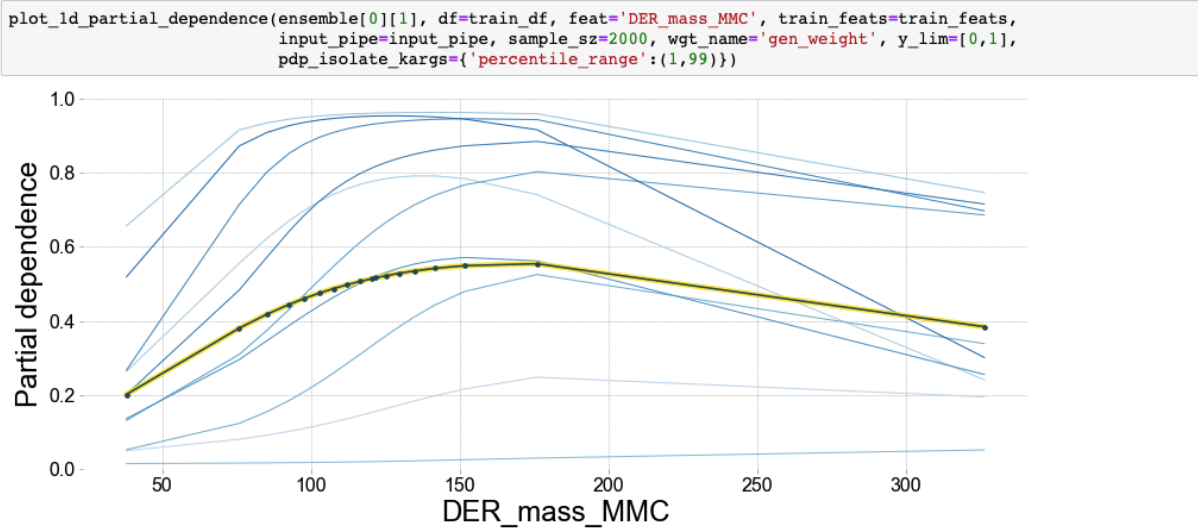


Figure 9.29: An example of interpreting the response of a trained model to a particular input feature, (the Higgs mass). Model predictions (y-axis) tend towards signal-like around the Higgs mass, as expected. The data, though, contains mostly background, however the passing of a weight feature to the method (`wgt_name='gen_weight'`), allows the weights of the data to be used to cure the class imbalance and centre the plot around a mean prediction of 0.5 (halfway between background and signal).

Uncertainties As an experimental science, knowledge about the uncertainties on results and measurements is essential. Where possible, methods in LUMIN compute the statistical uncertainties on quantitative results, in some cases via bootstrap resampling. Figure 9.30 shows an example of bootstrap resampling being used to plot the response of a regressor model, along with the moments of the regressed data for the Higgs to di-tau mass.

```

from lumin.utils.statistics import bootstrap_stats
from lumin.utils.multiprocessing import mp_run
x = np.linspace(0, 300, 500)
x_delta = np.linspace(-50, 200, 500)
x_pull = np.linspace(-1, 1, 500)

bs_stats = mp_run([{'data':df.h_tt_mass, 'name':'reco', 'x':x, 'kde':1, 'mean':1, 'std':1},
                  {'data':df.pred, 'name':'pred', 'x':x, 'kde':1, 'mean':1, 'std':1},
                  {'data':df.reco_delta, 'name':'reco_delta', 'x':x_delta, 'kde':1, 'mean':1, 'std':1},
                  {'data':df.reco_pull, 'name':'reco_pull', 'x':x_pull, 'kde':1, 'mean':1, 'std':1},
                  {'data':df.pred_delta, 'name':'pred_delta', 'x':x_delta, 'kde':1, 'mean':1, 'std':1},
                  {'data':df.pred_pull, 'name':'pred_pull', 'x':x_pull, 'kde':1, 'mean':1, 'std':1}], bootstrap_stats)

from lumin.plotting.data_viewing import plot_kdes_from_bs
plot_kdes_from_bs(x, {s:bs_stats[s] for s in bs_stats if 'reco' in s or 'gen' in s or 'pred' in s},
                  {'reco': {'condition': 'Reconstructed', 'ci': 'sd'},
                   'pred': {'condition': 'Regressed', 'ci': 'sd'}},
                  feat=r'$M_{h\rightarrow\tau\tau}$', units='GeV')

```

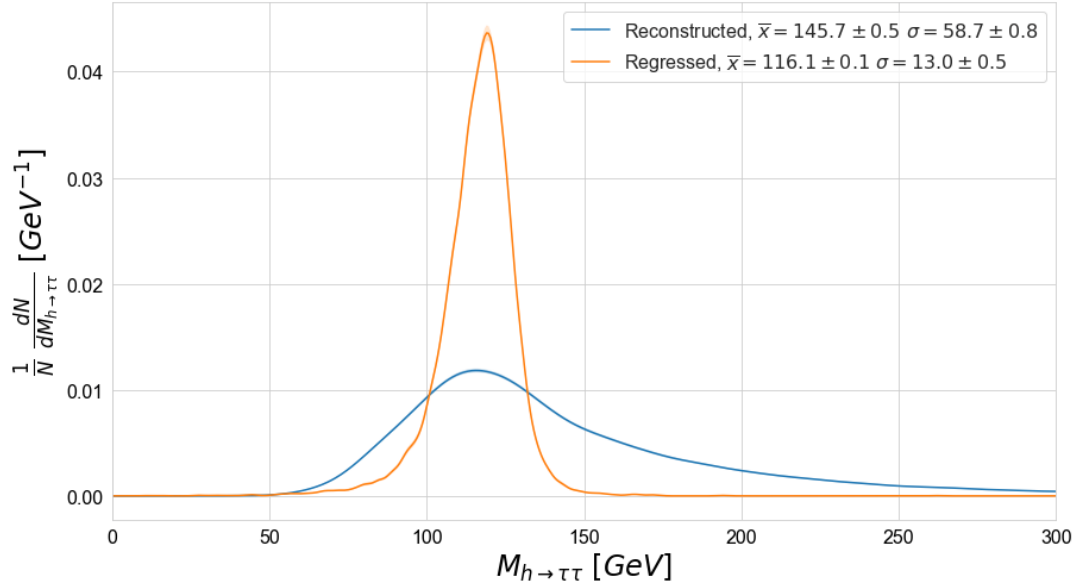


Figure 9.30: Bootstrap resampling is performed simultaneously on six datasets via multi-threading. The resampling is used to compute the moments and kernel density-estimates (KDEs) for the data distributions (two of which are then plotted), along with their uncertainties.

Uniform plot style Plotting methods take as argument a class which controls most aspects of plot appearance, colour style, save location and filetype, title, *et cetera*. The aim being to allow the user to make all plots conform to the same style and easily export them for publication without having to spend too much time on them.

9.9 Case study: HL-LHC projections

The investigations performed in Chap. 9 focused solely on a single dataset. Whilst it was representative of a typical binary classification problem in high-energy physics in both dataset size and metric, it does not guarantee that the improvements generalise to other datasets, even if the task is similar. In between completing the initial study and the followup study documented here, I co-supervised two master’s degree students for a summer studentship project studying the potential sensitivity of the High Luminosity LHC to non-resonant di-Higgs production; our contribution being the projections for the $hh \rightarrow \tau^+ \tau^- b\bar{b}$ channel for CMS.

As part of this study the two students, Miguel Bengala and Rodrigo Santo (Instituto Superior Técnico), re-derived the results and implementations of the initial study, finding similar improvements when using advance methods; a 30 % increase in sensitivity over the starting neural network. The results of this study are published in volume 7 of the 2019 CERN Yellow Reports [267], and a more in-depth description is documented in CMS-PAS-FTR-18-019 [268], which was also published as an addendum to the Yellow Report [269]. A further study was contributed to CERN-LHCC-2019-003 (CMS-TDR-020) [270]. A compact summary of the study is detailed below.

9.9.1 Motivation

As discussed in Sec. 3.3.7, the CMS detector will undergo significant upgrades to cope with the higher luminosity, and associated increases in pileup, at the HL-LHC. It is of interest to not only estimate the sensitivity that the upgraded detector will have in the search for rare phenomena, such as di-Higgs production, but also to quantify the effect that different parts of the detector, and their settings, will have.

Initially, an investigation was performed to estimate the expected significance for Standard Model, non-resonant $hh \rightarrow \tau^+ \tau^- b\bar{b}$, and the expected 95 %-confidence upper-limits on production cross-section of the signal as a function of the Higgs self-coupling. This was performed using dedicated Monte Carlo samples for the expected run conditions of the HL-LHC and upgraded CMS detector at its nominal phase-II specification. The study was later extended to better quantify the impact of the MIP timing detector [79] for two different timing resolutions.

9.9.2 Datasets and event selection

Dedicated Monte Carlo simulated datasets were centrally produced by CMS for the study of the expected performance under HL-LHC conditions. DELPHES was used to provide detector simulation assuming the presence of proposed CMS Phase-II upgrades, such as: improved inner tracker [76], high-granularity calorimeters [77, 78], and the minimum ionising particle timing detector (MTD) [79].

The samples used are: the signal, non-resonant Standard Model $h h \rightarrow \tau^+ \tau^- b \bar{b}$; and all of the major background processes, expanding upon Chap. 8, which only considered the dominant background $t \bar{t}$. The samples were generated at a centre of mass energy of 14 TeV, including an average of 200 pile-up collisions per event, and normalised to an integrated luminosity of 3000 fb^{-1} ; the expected run-conditions for the HL-LHC.

The event selection is similar to that used for the preliminary investigations (Chapters 7 and 8), except that the $e \tau_h b b$ and $\tau_h \tau_h b b$ decay channels are also considered in addition to $\mu \tau_h b b$ channel.

The resulting distributions for the Higgs masses are shown in Fig. 9.31. From these we can see that $t \bar{t}$ production is still the dominant background is $t \bar{t}$ production.

9.9.3 Neural network discriminator training

The events passing the selection were split into two sets, and a pair of discriminators were trained, one on each set. In order to get results, the discriminators were used to predict the class (signal or background) of the set on which they were not trained and the predictions were then combined for inference of the expected signal-significance and upper-limit on signal production cross-section. This allowed us to make full use of the data available to reduce the statistical uncertainty on yield and prediction shapes, whilst being protected against biasing the result, which would occur if the model were used to predict data points on which it had been trained. Each of these discriminators was an ensemble of ten neural networks.

Similar to Chap. 8, a mixture of low-level and high-level features were used as input, with some coarse tests to refine the selection used. The investigation into the impact of advanced techniques described in Ref. [43] was repeated (with the exception of SWA) using the AMS as an optimisation metric. The final model used SELU activation functions, cosine SGDR learning-rate schedules, and train-time/inference-time data-augmentation, resulting in an AMS 30 % higher than that of the starting model.

9.9.4 Results

Expected values of the discovery significance and 95 %-confidence upper-limit on production cross-section of the signal were evaluated using a simultaneous fit in each of the three

Channel	Significance		95% CL limit on $\sigma_{hh}/\sigma_{hh}^{\text{SM}}$	
	Stat. + syst.	Stat. only	Stat. + syst.	Stat. only
$b\bar{b}b\bar{b}$	0.95	1.2	2.1	1.6
$\tau^+\tau^-b\bar{b}$	1.4	1.6	1.4	1.3
$b\bar{b}WW(\ell\nu\ell\nu)$	0.56	0.59	3.5	3.3
$b\bar{b}gg$	1.8	1.8	1.1	1.1
$b\bar{b}ZZ(\ell\ell\ell\ell)$	0.37	0.37	6.6	6.5
Combination	2.6	2.8	0.77	0.71

Table 9.11: Summary of expected significance and upper limits of this study, the results of similar studies in the other four main search channels, and their combination. Table source [268].

decay-channels, using the event distributions as a function of the discriminator prediction, as shown in Fig. 9.32.

The results for the $\tau^+\tau^-b\bar{b}$ channel for Standard Model production are: an expected significance of 1.4σ , considering the systematic uncertainties, increasing to 1.6σ when only considering the statistical uncertainties on the yields; or a 95 %-confidence upper-limit of 1.4 times the Standard Model production cross-section with systematic uncertainties (1.3 for statistical uncertainties only). Combining the $\tau^+\tau^-b\bar{b}$ channel with the other major decay-channels for di-Higgs production, summarised in Tab. 9.11, the expected significance increases to almost 3σ (2.6σ with systematic uncertainties and 2.8σ for statistical uncertainties only) - i.e. almost “evidence” but falling short of “discovery”.

This, however, is evaluating the case in which the Higgs-boson self-coupling value is the Standard Model one ($\kappa_\lambda = \lambda_{hhh}/\lambda_{hhh,SM} = 1$). As discussed in Sec. 6.1.1, the production cross-section and kinematics for non-resonant di-Higgs varies as a function of κ_λ , and so the limit achievable can also vary. By reweighting events in the $\kappa_\lambda = 1$ sample as a function of the generator-level di-Higgs mass and $\cos\theta^*$ (the angle between one of the Higgs bosons and the beam-line in the centre-of-mass rest frame) we can transform the sample to any value of κ_λ , as described in Ref. [271]. For simplicity we do not retrain the discriminators on the reweighted signal sample, but rather just evaluate the predictions on the reweighted data. Having done this, we can evaluate the expected upper-limit as a function of κ_λ , as shown in Fig. 9.33. From this we can see that the limit obtainable does vary as a function κ_λ , reaching a minimum at around $\kappa_\lambda = 2.45$, corresponding to the minimum value of the theoretical di-Higgs production cross-section, which has a quadratic dependence on κ_λ .

Assuming instead that the di-Higgs signal does exist and $\kappa_\lambda = 1$, we can reinterpret the results to experimentally constrain the possible values of κ_λ through a likelihood scan,

as shown in Fig. 9.34. From this we can observe a double-minima structure for several channels which are sensitive to the di-Higgs mass. However, by combining the channels the second minima is raised just above the 95 % confidence level. Further improvements to the analysis can be expected to restrict the 95 % region, thereby reducing the experimental uncertainty on the value of κ_λ .

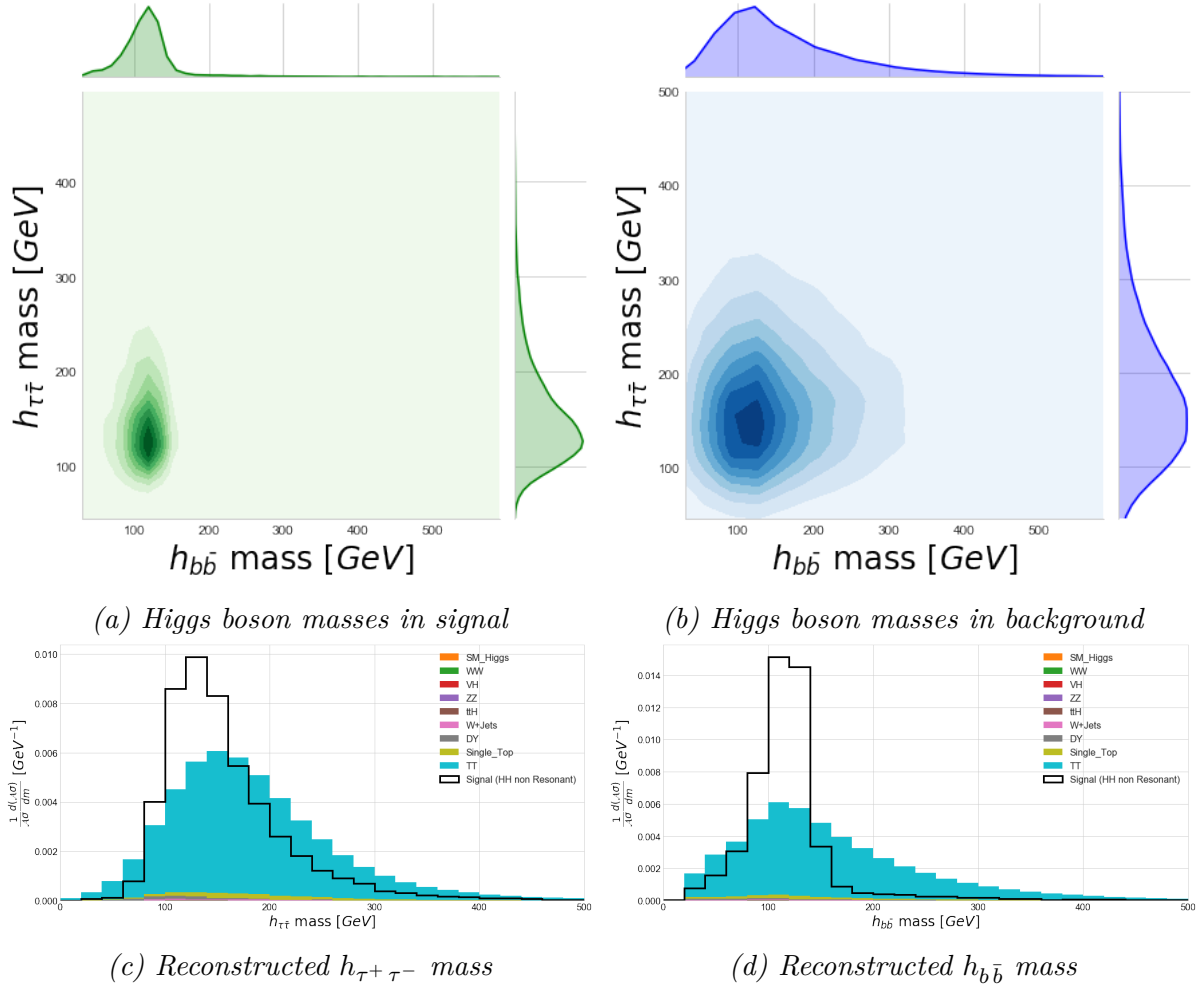


Figure 9.31: Top: Unity-normalised 2D distributions of the invariant masses of the reconstructed Higgs bosons ($h_{\tau^+\tau^-}$ and $h_{b\bar{b}}$) in signal and background ($t\bar{t}$). 1D invariant-mass distributions are projected on the corresponding axes. All channels, and background processes are shown together.

Bottom: Unity-normalised 1D invariant-mass distributions of each reconstructed Higgs mass, in which the relative contributions of each background process are indicated.

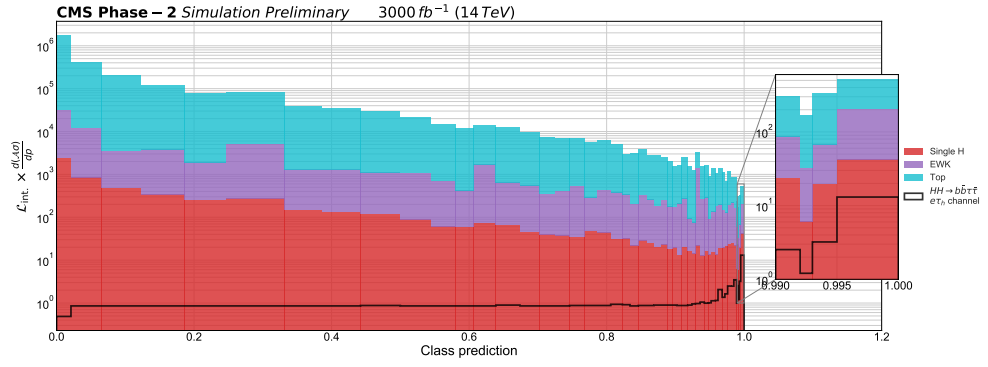
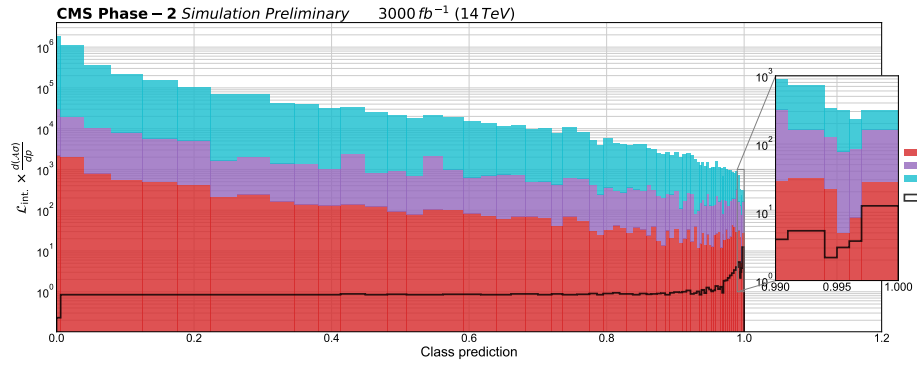
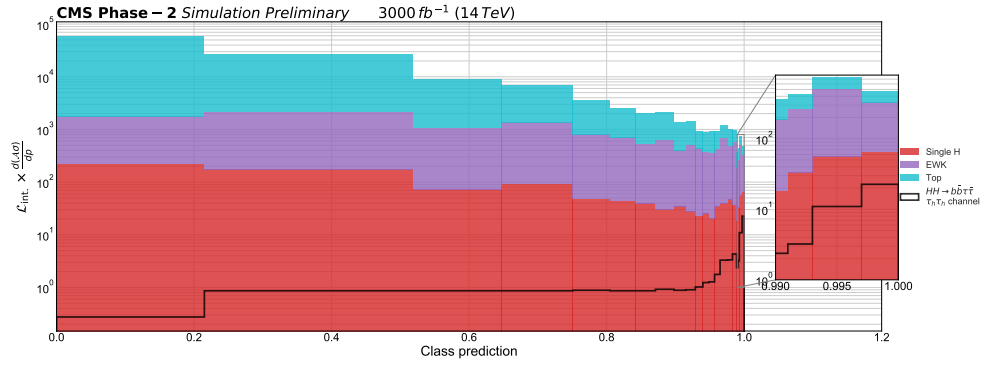
(a) $e\tau_h bb$ channel(b) $\mu\tau_h bb$ channel(c) $\tau_h\tau_h bb$ channel

Figure 9.32: Event-yield distributions as a function of the discriminator output per decay channel. The individual background-processes are compacted into overall categories: Single SM Higgs boson, Electroweak, and Top.

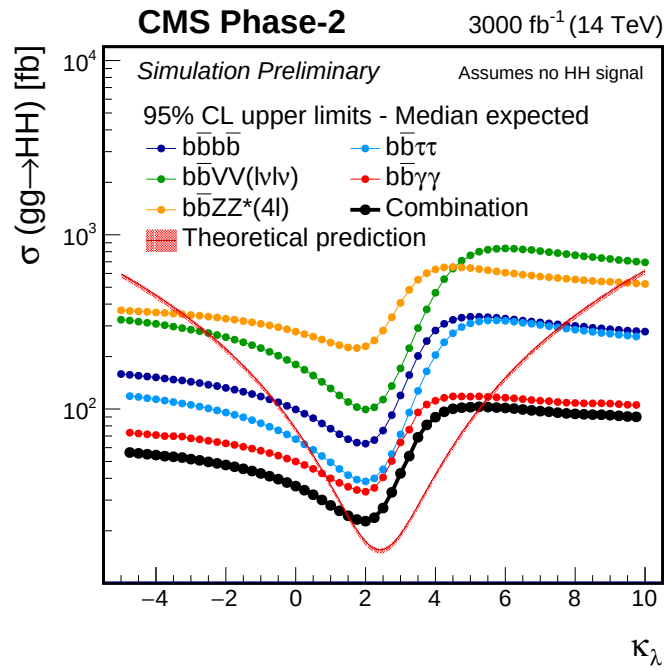


Figure 9.33: Expected 95 % upper-limit on the di-Higgs production cross-section as a function of κ_λ , assuming to di-Higgs signal.

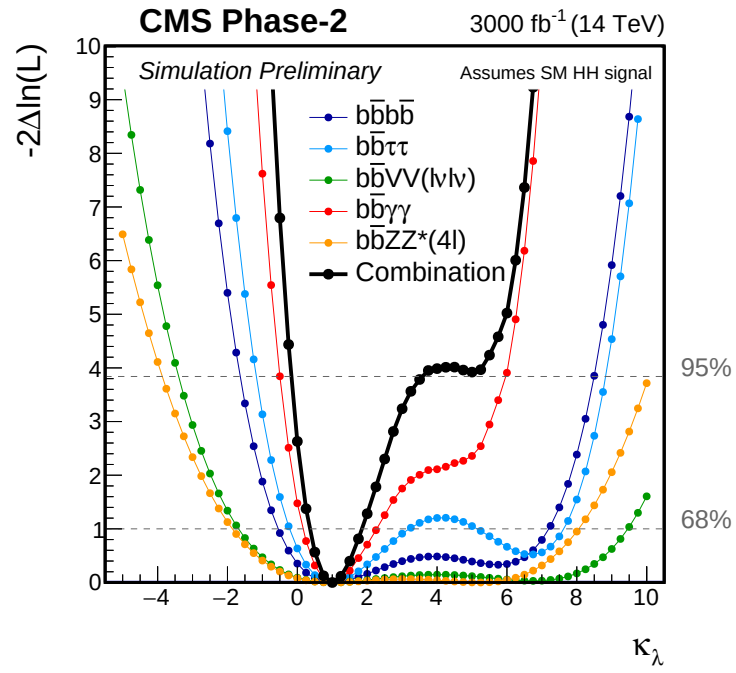


Figure 9.34: Expected log-likelihood scan of κ_λ assuming SM di-Higgs production at $\kappa_\lambda = 1$

9.9.5 Impact of the MIP timing detector on analysis sensitivity

The MIP Timing Detector is a new subsystem which will be installed at the HL-LHC, allowing time-of-flight information to be used during tracking. This is expected to be useful in mitigating the effects of the increased pile-up present under HL-LHC run-conditions, resulting in improved object-isolation and jet tagging efficiency. In order to help quantify the impact of the new subsystem, this study was revised in order to include the expected changes in sensitivity to di-Higgs production under several scenarios, using new samples with improved simulation of the MTD.

Figure 9.35, shows the improvement in sensitivity due to the MTD as a function of the κ_λ likelihood, and the expected significances under several timing resolutions are summarised in Tab. 9.12. From these results we can see that the presence and time-resolution of the MTD has a significant impact on the expected significance, particularly for the $\tau^+ \tau^- b \bar{b}$ channel, which benefits from efficiency improvements to four object types: b -tagging, electrons, muons, and taus.

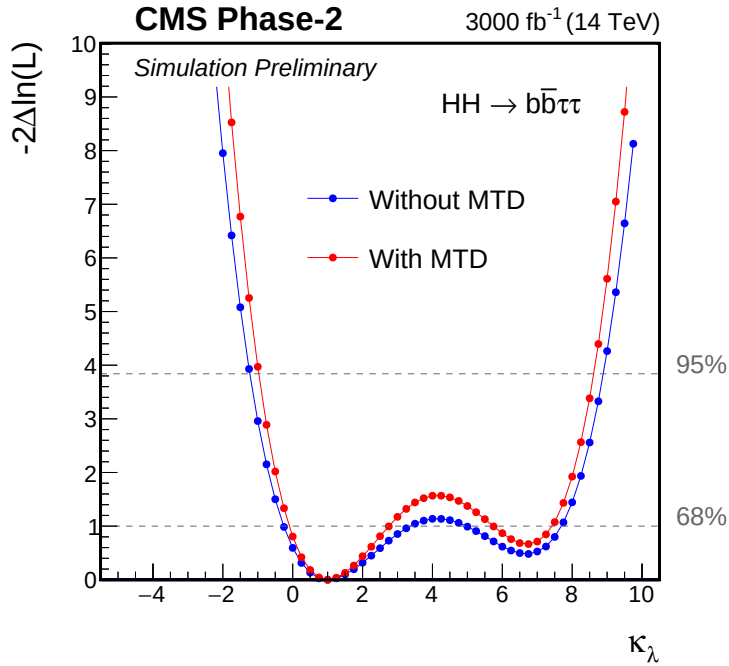


Figure 9.35: Expected log-likelihood scan of κ_λ assuming SM di-Higgs production at $\kappa_\lambda = 1$, with and without MTD@35 ps

Channel	Expected Significance [σ]		
	No MTD	MTD@50 ps	MTD@35 ps
$bbbb$	0.88	0.94	0.95
$bb\tau\tau$	1.3	1.48	1.6
$bb\gamma\gamma$	1.7	1.83	1.9
$bbWW$	0.53	0.58	0.58
$bbZZ$	0.38	0.42	0.42
Combined	2.4	2.63	2.7

Table 9.12: Impact of the MTD on the expected significance of SM di-Higgs production for various timing resolutions.

Chapter 10

Comprehensive investigation: The search for di-Higgs production at CMS

10.1 Introduction

The preliminary investigation performed in Chapters 7 and 8 studied the applicability of deep neural networks (DNNs) to the problem of identifying non-resonant di-Higgs production in the $\tau^+ \tau^- b \bar{b}$ decay channel. I found that a DNN-based event-level classifier for signal and background was able to lead to encouraging expected significances, and additionally that whilst DNN-based regressors for the Higgs and di-Higgs masses were able to significantly improve the resolution of the mass in signal over basic reconstruction, the improvement was on par with the algorithms already approved for use in CMS analyses at the time.

In Chap. 9, I studied how modern advancements in deep-learning may be used to further improve the performance of DNNs by demonstrating first on a benchmark HEP dataset, and then reporting a projection study which also benefitted from the use of the advanced techniques studied. This investigation also gave rise to a software package, LUMIN, for implementing improved DNNs on HEP data.

In this final part of my thesis, I report my contributions to the LHC Run-II search for $h h \rightarrow \tau^+ \tau^- b \bar{b}$ performed by CMS. This analysis studies all three years of data collected during 2016, 2017, and 2018 (137.19 fb^{-1} of integrated luminosity) to search for both non-resonant and resonant di-Higgs production, via both gluon-gluon fusion (ggF) and vector-boson fusion (VBF) production methods. The full analysis is the result of the hard work of many researchers, each with their own specific work tasks, and indeed builds upon the work of the whole CMS collaboration through the use of the hardware, software, algorithms, methods, and experience.

My personal and direct contributions to the analysis are:

- The event-level classifiers for the ggF production mechanism (both resonant and non-resonant), described in Sections 10.9 and 10.10, including the input-feature selection (Sec. 10.7);
- The optimisation of the cuts on the Higgs candidate masses (Sec. 10.3.9);
- An initial investigation on the 2016 data, documented in Ref. [272];
- Contributions to the writing of analysis notes, the publication, and internal presentations.

For completeness I will provide an overview of the details of the analysis. The complete documentation may be found in Ref. [273]. At the time of writing of this thesis, all results being finalised.

10.2 Datasets

10.2.1 Collider data

Three years worth of collider data collected from $\sqrt{s} = 13$ TeV proton-proton collisions at the LHC during 2016, 2017, and 2018 are used. In total this equates to 137.19 fb^{-1} of integrated luminosity, the yearly breakdown of which is documented in Tab. 10.1.

10.2.2 Simulated samples

Simulated data are produced via Monte Carlo generation for each year separately. Full detector simulation is applied via GEANT 4. The PDF used is NNPDF3.1 [274] and the underlying event is modelled using the CP5 tune [275]. The parton showering is performed in PYTHIA 8, with appropriate matching/merging procedures with the ME.

Year	Integrated Luminosity [fb^{-1}]
2016	35.92
2017	41.53
2018	59.74
Sum	137.19

Table 10.1: Yearly breakdown of integrated luminosity.

Benchmark	κ_λ	κ_t	c_2	c_g	c_{2g}
1	7.5	1.0	-1.0	0.0	0.0
2	1.0	1.0	0.5	0.8	0.6
3	1.0	1.0	-1.5	0.0	-0.8
4	-3.5	1.5	-3.0	0.0	0.0
5	1.0	1.0	0.0	0.8	-1.0
6	2.4	1.0	0.0	0.2	-0.2
7	5.0	1.0	0.0	0.2	-0.2
8	15.0	1.0	0.0	-1.0	1.0
9	1.0	1.0	1.0	-0.6	0.6
10	10.0	1.5	-1.0	0.0	0.0
11	2.4	1.0	0.0	1.0	-1.0
12	15.0	1.0	1.0	0.0	0.0
SM	1.0	1.0	0.0	0.0	0.0

Table 10.2: Benchmark nodes for exploring the space of anomalous couplings in non-resonant di-Higgs production.

- $\kappa_\lambda = \lambda_{hhh}/\lambda_{hhh,SM}$, the ratio of the Higgs trilinear coupling to the SM value;
- $\kappa_t = y_t/y_{t,SM}$, the ratio of the top Yukawa coupling to the SM value;
- c_2 , c_g , and c_{2g} parameters controlling non-SM Higgs-interactions with gluons and top quarks.

Table source [276].

Di-Higgs production

LO samples for resonant and non-resonant di-Higgs production via gluon fusion (ggF) and vector-boson fusion (VBF) are produced using MADGRAPH 5. The resonant samples cover graviton and radion masses in the range 250-3000 GeV. The decays are restricted to the $\tau^+ \tau^- b \bar{b}$ channel to improve generation efficiency. The non-resonant samples include the Standard Model (SM) expectation and samples in which non-SM coupling strengths are included. These are generated at 12 benchmark “nodes” according to Ref. [276] to sample adequately the phase-space of anomalous couplings. These benchmarks are summarised in Tab. 10.2. The full BSM coupling range may be explored by reweighting the node according to the generator-level di-Higgs mass and $\cos \theta^*$ (the angle between one of the Higgs bosons and the beam-line in the centre-of-mass rest frame), as described in Ref. [271]. To improve the inference of signal, NLO samples of signal are used, however these are small samples, and so the LO samples are used to train the DNN discriminators and the NLO samples are then scored for inference.

Backgrounds

Drell-Yan Drell-Yan (DY) production of di-lepton plus jets ($Z/\gamma \rightarrow \ell\ell + \{n_{\text{jets}}\}$) is modelled using LO MADGRAPH 5 and NLO using MADGRAPH 5 AMC@NLO. In order to help cover the full production phase-space, several samples are generated, and unless specified, the invariant mass of the di-lepton is required to be at least 50 GeV:

- A series of samples in which the MEs have exactly 0, 1, 2, 3, and 4 extra outgoing partons;
- Samples in which the MEs have two outgoing b quarks;
- A series of samples generated in bins of H_T ;
- Samples covering the case in which the di-lepton invariant mass is in the range 10-50 GeV.

LO modelling of DY processes is known to be imperfect [277], however only small NLO samples are available, therefore the LO samples are first reweighted to match the NLO samples, and then the normalisation is corrected using collider data. This will be discussed further in Sec. 10.4.

Electroweak Contributions from $W \rightarrow \ell\nu_\ell$ and $Z/\gamma \rightarrow \ell\ell$, with the possible addition of extra jets but only allowing EWK interactions, are modelled using LO MADGRAPH 5.

Top-quark pair-production $t\bar{t}$ contributions are modelled using NLO POWHEG BOX 2.0 rescaled using NNLL matched to NNLO calculations and assuming a top-quark mass of 172.5 GeV. Generation is performed separately for the fully- and semi-leptonic, and fully-hadronic decay channels.

Single top Single top-quark contributions are also modelled using NLO POWHEG BOX 2.0, and generated separately for top scattering and tW production, as well as separately for top and anti-top quarks. Similar to top-quark pair-production, the cross-section is rescaled using higher-order calculations.

W +jets Contributions from $W \rightarrow \ell\nu_\ell$ are modelled using LO MADGRAPH 5, via both inclusive and H_T -binned generation.

Multi-boson Double and triple production of W and Z bosons is modelled at NLO for a variety of processes and decay channels, using POWHEG BOX 2.0 and MADGRAPH 5 AMC@NLO+MADSPIN.

Single Higgs-boson Though single Higgs decays to $b\bar{b}$ or $\tau^+\tau^-$ can contribute. Production mechanisms include ggF, VBF, and vector boson association. These are modelled as NLO using POWHEG BOX 2.0 assuming a SM Higgs boson with mass 125 GeV.

$t\bar{t}X$ production $t\bar{t}$ in association with one or two bosons can also contribute: $t\bar{t}H$ is modelled at NLO using POWHEG BOX 2.0; $t\bar{t}W$ +jets and $t\bar{t}Z$ are modelled at NLO using MADGRAPH 5 AMC@NLO; and $t\bar{t}ZZ$, $t\bar{t}WW$, $t\bar{t}WZ$, $t\bar{t}WH$, and $t\bar{t}ZH$ are modelled at LO using MADGRAPH 5.

QCD multi-jet As discussed in Sec. 6.3.4, QCD production of jets can form a small, but difficult to model, background. QCD contributions are instead inferred from collider data and will be discussed further in Sec. 10.4.

10.3 Event selection

10.3.1 High-level trigger requirements

Events must fire at least one relevant high-level trigger (HLT) in order to be considered. The HLTs considered include:

- Electrons: tight-ID electron, either single or crossed with a loose-ID τ_h ;
- Muons: isolated muon, either single or crossed with a loose-ID τ_h ;
- Di-tau: double medium-ID $\tau_h \tau_h$ (loose-ID for VBF searches).

10.3.2 Lepton selection

Kinematic and ID requirements are applied to electrons, muons, and taus according to which HLTs were fired by the event, and the year in which the event was recorded. The p_T requirements are set such that leptons are accepted within the region of flat triggering efficiency. These requirements are summarised in Tab. 10.3. Additionally, the Hadrons-Plus-Strips (see Sec. 3.4.4) decay modes are filtered to only include decay modes certified for use in analyses.

10.3.3 Channel assignment

Events are assigned to an appropriate channel in the following order:

1. If a muon and τ_h pair separated by $\Delta R > 0.5$ is present, the event is assigned to the $\mu \tau_h$ channel;

Electrons					
Year	HLT	Min. p_T [GeV]	Max. $ \eta $		
2016	Single e	26	2.1		
2017/18	Single e	33	2.1		
2017/18	Cross $e \tau_h$	25	2.1		
Year	HLT	MVA ID	Max. $ dxy $ [cm]	Max. $ dz $ [cm]	
All	All	Tight	0.045	0.2	

Muons					
Year	HLT	Min. p_T [GeV]	Max. $ \eta $		
2016	Single μ	23	2.1		
2016	Cross $\mu \tau_h$	20	2.1		
2017/18	Single μ	25	2.1		
2017/18	Cross $\mu \tau_h$	21	2.1		
Year	HLT	ID	Max. Iso.	Max. $ dxy $ [cm]	Max. $ dz $ [cm]
All	All	Tight	0.15	0.045	0.2

Taus					
Year	HLT	Min. p_T [GeV]	Max. $ \eta $		
2016/17/18	Single $e \mu$	20	2.3		
2016	Cross $\mu \tau_h$	25	2.1		
2016	Di-tau	40	2.1		
2017/18	Cross $e \tau_h$	35	2.1		
2017/18	Cross $\mu \tau_h$	32	2.1		
2017/18	Di-tau	40	2.1		
2017/18	VBF $h \rightarrow \tau_h \tau_h$	25	2.1		
Year	HLT	VSe	VS μ	VSjet	Max. $ dz $ [cm]
All	Single e	Tight	Tight	VVVLoose	0.2
All	Cross $e \tau_h$	Tight	Tight	VVVLoose	0.2
All	Single μ	VLoose	Tight	VVVLoose	0.2
All	Cross $\mu \tau_h$	Tight	Tight	VVVLoose	0.2
All	Di-tau	VLoose	Tight	VVVLoose	0.2
All	VBF $h \rightarrow \tau_h \tau_h$	VLoose	VVLoose	VVVLoose	0.2

Table 10.3: Kinematic requirements on leptons by particle, year, and HLT. Tau ID is performed using DeepTau.

Lepton	Min. p_T [GeV]	Max. $ \eta $	ID	Max.Iso.	Max. $ dxy $ [cm]	Max. $ dz $
e	10	2.5	MVA Veto	0.3	0.045	0.2
μ	10	2.4	Medium	0.3	0.045	0.2

Table 10.4: Kinematic requirements on veto leptons by particle.

2. If an electron and τ_h pair separated by $\Delta R > 0.5$ is present, the event is assigned to the $e \tau_h$ channel;
3. If a $\tau_h \tau_h$ pair separated by $\Delta R > 0.5$ is present, the event is assigned to the $\tau_h \tau_h$ channel;
4. If a $\mu \mu$ pair separated by $\Delta R > 0.5$ is present, the event is assigned to the $\mu \mu$ channel.

The $\mu \mu$ channel is not used for signal investigation, but is instead used to correct the DY background, which will be discussed further in Sec. 10.4.2. Events not assigned to a specific final-state are rejected.

10.3.4 Third-lepton veto

To decrease contamination from the DY background, particularly in the $\mu \tau_h$ and $e \tau_h$ channels, events in any channel which contain either an additional electron or muon are rejected. The requirements for the veto light-leptons are relaxed compared to those used for primary lepton selection, and are detailed in Tab. 10.4.

10.3.5 τ_h selection

Following the 3rd-lepton veto, in the $e \tau_h$ and $\mu \tau_h$ channels there will be one candidate to use as the light-lepton, however there may be multiple candidates to choose from to use as the hadronic tau. Similarly in the $\tau_h \tau_h$ channel there may be multiple τ_h candidates. In the $e \tau_h$ and $\mu \tau_h$ channels, the τ_h is chosen to be the most isolated τ_h which is at least $\Delta R > 0.5$ away from the light lepton. In the $\tau_h \tau_h$, channel the first τ_h is chosen to be the most isolated τ_h , and the second one is the next most isolated τ_h which is at least $\Delta R > 0.5$ away. τ_h isolation here refers to the anti-jet discriminator score as computed by DeepTau [106] (see Sec. 3.4.4).

10.3.6 MC-correction categories

Corrections to MC data will be applied to allow the simulation to better match the collider-data, however the corrections applied will depend on the origins of the reconstructed

objects. The origins of reconstructed (reco) particles are determined by checking their ΔR separations with the generator-level (gen) particles prior to detector simulation, and by using the fractional difference in p_T computed using:

$$\Delta p_T(\text{gen, reco}) = \frac{p_{T,\text{gen}} - p_{T,\text{reco}}}{p_{T,\text{gen}}}. \quad (10.1)$$

Matching is performed by checking criteria in the following order:

1. Prompt electron: gen-level prompt electron with $p_T > 8 \text{ GeV}$ and $\Delta R(\text{gen, reco}) < 0.3$;
2. Prompt muon: gen-level prompt muon with $p_T > 8 \text{ GeV}$ and $\Delta R(\text{gen, reco}) < 0.3$;
3. $\tau \rightarrow e$: gen-level prompt-decay from τ electron with $p_T > 8 \text{ GeV}$, $\Delta R(\text{gen, reco}) < 0.3$, and $\Delta p_T(\text{gen, reco}) < 0.5$;
4. $\tau \rightarrow \mu$: gen-level prompt-decay from τ muon with $p_T > 8 \text{ GeV}$, $\Delta R(\text{gen, reco}) < 0.3$, and $\Delta p_T(\text{gen, reco}) < 0.5$;
5. $\tau \rightarrow \tau_h$: gen-level jet clustered from prompt-decay products of gen-tau (excluding muons and electrons) with $p_T > 15 \text{ GeV}$, $\Delta R(\text{gen, reco}) < 0.3$, and $\Delta p_T(\text{gen, reco}) < 1.0$;
6. Fake- τ_h via Jet/Pile-up: Any reco candidate not able to fulfil the previous criteria.

10.3.7 Jet selection

Jets are corrected for pile-up using PUPPI [97], as described in Sec. 3.4.3. *Resolved jets* are clustered using $R = 0.4$ anti- k_t under the assumption that they contain the hadronisation products of a single b quark, and *fat jets* are clustered using $R = 0.8$ anti- k_t under the assumption that they contain the hadronisation products of a pair of b quarks.

b -jet candidates are selected from $R = 0.4$ anti- k_t jets with $p_T > 20 \text{ GeV}$ and $|\eta| < 2.4$ passing tight jet-ID or loose pile-up ID¹ if $p_T < 50 \text{ GeV}$. The selection is performed by a neural-network-based system called HHBTAG [278], which is trained to score jets in the event according to their likelihood of originating from $h_{b\bar{b}}$ decays.

VBF jets are selected from $R = 0.4$ anti- k_t jets with $p_T > 20 \text{ GeV}$ and $|\eta| < 4.7$ passing tight jet-ID or loose pile-up ID if $p_T < 50 \text{ GeV}$.

Fat-jets are clustered using $R = 0.8$ anti- k_t jets with at least two sub-jets and a soft-drop mass [279] greater than 30 GeV . The highest p_T sub-jets are then required to lie within $\Delta R < 0.4$ separations of two b -jet candidates which pass the loose b -tag WP.

¹Jet and pileup IDs are quality checks based on the hadronic and electromagnetic composition of the jets in order to cut out poorly reconstructed jets and pileup

10.3.8 Jet categorisation

Events are further categorised by their jets and b tag, as computed using the DeepFlavour algorithm [100] (see Sec. 3.4.4), in the following order:

1. Less than two resolved jets: Reject event;
2. Event contains one fat-jet: Assign event to the “Two b boosted” category;
3. Event contains at least one b -jet candidate that passes the medium b -tag WP, and two VBF jets with $\Delta\eta_{jj} > 2$ and $m_{jj} > 650$ GeV: Assign to the “VBF category”;
4. Both b -jet candidates pass the medium b -tag WP: Assign event to the “Two b resolved” category;
5. One b -jet candidate passes the medium b -tag WP: Assign event to the “One b one jet resolved” category;
6. Else: Assign event to the “Two jet resolved” category.

10.3.9 Mass cuts

The selection of events so far has been comparatively open, with any necessary cuts set to minimal reasonable values. A significant portion of the background (along with some signal) can be removed via cuts on the reconstructed Higgs masses ($h_{b\bar{b}}$ and $h_{\tau^+\tau^-}$). These events, however, can potentially be useful in training the DNN discriminator, and the discriminator may well be able to separate signal and background in the excludable regions better than simple cuts on the Higgs masses.

The cut can take the form of an ellipse in the 2D mass plane of the Higgs masses. This is traditionally parameterised as:

$$\frac{(m_{\tau^+\tau^-} - m_{\tau^+\tau^-,\text{offset}})^2}{m_{\tau^+\tau^-, \text{res.}}^2} + \frac{(m_{b\bar{b}} - m_{b\bar{b},\text{offset}})^2}{m_{b\bar{b}, \text{res.}}^2} < 1; \quad (10.2)$$

where the offsets correct for biases in the mass measurements in signal, the resolutions are the expected widths of the mass distributions in signal. $m_{\tau^+\tau^-}$ is as computed by SVFit.

The preceding version of this search (Ref. [214] - 2016 data only), used an ellipse of the form:

$$\frac{(m_{\tau^+\tau^-} - 116)^2}{35^2} + \frac{(m_{b\bar{b}} - 111)^2}{45^2} < 1. \quad (10.3)$$

for non-boosted jet-categories. Boosted jet-categories used mass-window cuts of the form:

$$\begin{aligned} 80 \text{ GeV} < m_{\tau^+\tau^-} < 160 \text{ GeV} \\ 90 \text{ GeV} < m_{b\bar{b}} < 160 \text{ GeV} \end{aligned} \quad (10.4)$$

In this analysis, such cuts result in a 68 % signal efficiency and a 83 % background rejection in the non-boosted jet-categories.

The ellipse cut can be adjusted via randomly sampling values for the four parameters and selected the set corresponding to the highest background reject for a signal efficiency above a specified threshold. This can be performed separately for boosted and non-boosted jet-categories. In testing we found that DNNs trained on using data passing 90 % efficient cuts were able to outperform DNNs trained using data passing the previous mass-cut settings. These cuts are as follows:

$$\frac{(m_{\tau^+\tau^-} - 129)^2}{53^2} + \frac{(m_{b\bar{b}} - 169)^2}{145^2} < 1, \quad (10.5)$$

for non-boosted jet-categories, and

$$\frac{(m_{\tau^+\tau^-} - 128)^2}{60^2} + \frac{(m_{b\bar{b}} - 159)^2}{94^2} < 1, \quad (10.6)$$

for boosted jet-categories. Figure 10.1 illustrates the distributions of the Higgs-candidate masses in signal and background, and the elliptical cuts.

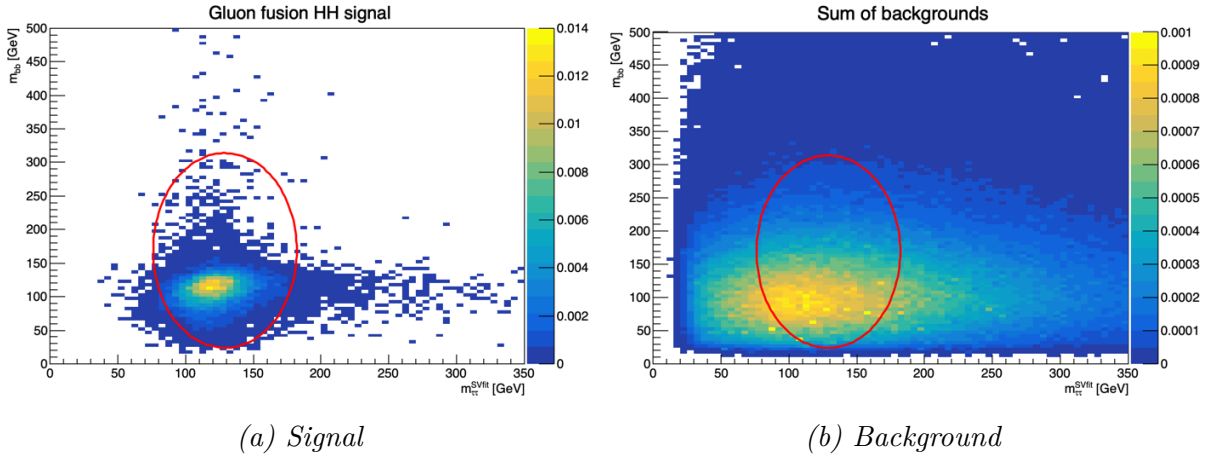


Figure 10.1: 2D distributions of the Higgs masses in signal and background in the 2018 “Two b resolved” category. All decay channels and MC backgrounds are merged and weighted. The red line indicates the elliptical cut. Image source [273].

Selection of events in the $\mu\mu$ channel (which are used to compute DY corrections) use slightly tighter resolutions of 40 GeV and 50 GeV for $m_{\tau^+\tau^-}$ and $m_{b\bar{b}}$, respectively. Additionally, an alternative $\mu\mu$ sample is defined, which only requires $m_{\tau^+\tau^-} > 50$ GeV. These two samples are referred to as the MH and MTT samples, respectively.

10.3.10 Regional categorisation

Remaining events are now sorted into exclusive *regions*, in order to further remove background contamination and to allow the contributions from QCD multi-jets to be inferred (see Sec. 10.4.1). Regions are defined as follows for the semi-leptonic channels:

1. Signal region (*A*): $h_{\tau^+\tau^-}$ leptons are oppositely charged and hadronic tau passes the medium anti-jet WP;
2. Control Region *B*: $h_{\tau^+\tau^-}$ leptons are similarly charged and hadronic tau passes the medium anti-jet WP;
3. Control Region *C*: $h_{\tau^+\tau^-}$ leptons are oppositely charged and the hadronic tau fails the medium anti-jet WP;
4. Control Region *D*: $h_{\tau^+\tau^-}$ leptons are similarly charged and the hadronic tau fails the medium anti-jet WP.

and for the $\tau_h\tau_h$ channel:

1. Signal region (*A*): $h_{\tau^+\tau^-}$ leptons are oppositely charged and both hadronic taus pass the medium anti-jet WP;
2. Control Region *B*: $h_{\tau^+\tau^-}$ leptons are similarly charged and both hadronic tau pass the medium anti-jet WP;
3. Control Region *C*: $h_{\tau^+\tau^-}$ leptons are oppositely charged and one hadronic tau fails the medium anti-jet WP, but the other passes;
4. Control Region *D*: $h_{\tau^+\tau^-}$ leptons are similarly charged and one hadronic tau fails the medium anti-jet WP, but the other passes;
5. Reject: Both hadronic taus fail the medium anti-jet WP.

Only the signal region is used for signal inference, however the remaining regions are necessary for QCD estimation.

10.3.11 Process contributions

Table 10.5 details the expected yields per channel and jet category for the non-resonant gluon-fusion search. The Drell Yan yields include scale factors, the computation of which will be described in Sec. 10.4.2.

Process	Expected yield by channel			
	$e \tau_h b b$	$\mu \tau_h b b$	$\tau_h \tau_h b b$	Total
Non-resonant ggF (LO)	3.2	6.6	4.1	13.9
Overall bkg.	54665.4	115627.7	3809.6	174102.7
TT	42962.8	87272.8	1298.1	131533.6
DY	5385.8	15093.8	2134.1	22613.6
Wjets	2855.2	6195.5	74.3	9125.0
ST	2928.9	5862.7	103.5	8895.1
VV	292.3	666.2	73.0	1031.6
EWK	107.3	251.8	37.2	396.2
SM Higgs	53.1	123.1	54.4	230.6
ttH	54.9	109.6	17.7	182.2
VH	19.1	42.8	16.4	78.3
VVV	6.0	9.5	1.0	16.4
AMS	0.014	0.019	0.067	

Process	Expected yield by jet category			
	2j1bR noVBF	2j2bR noVBF	2j2LbB noVBF	2j1b VBF
Non-resonant ggF (LO)	5.4	6.2	0.9	1.4
Overall bkg.	102053.1	62815.4	281.5	8952.7
TT	66263.6	57206.1	162.7	7901.2
DY	19503.5	2508.3	65.0	536.8
Wjets	8269.2	707.2	8.7	140.0
ST	6516.7	2082.2	12.7	283.5
VV	846.4	135.3	16.7	33.2
EWK	326.3	49.1	5.2	15.7
SM Higgs	194.4	18.2	2.4	15.7
ttH	72.3	83.9	3.0	22.9
VH	48.2	23.0	4.9	2.3
VVV	12.6	2.0	0.3	1.5
AMS	0.017	0.025	0.054	0.014

Table 10.5: Expected yields per process for all three years (137.19fb^{-1} of integrated luminosity), along with the Approximate Median Significance (AMS) for channels or jet-categories currently present through the event selection alone, i.e. prior to any event classification or discriminating cut.

10.4 Collider-data-driven background samples and corrections

10.4.1 QCD multi-jet

QCD contributions to the signal region (same-sign $h_{\tau^+\tau^-}$ leptons with hadronic taus passing the medium anti-jet WP of DeepTau) are inferred via extrapolation from the other regions, as defined in Sec. 10.3.10 and illustrated in Fig. 10.2, in a process referred to as the ABCD method.

In the ABCD method, signal contribution in the control regions B , C , and D (as defined in Sec. 10.3.10) are assumed to be negligible. This is reasonable given that the signal process gives rise to genuine leptons of opposite charge. We then assume that any deficit in collider-data modelling by MC backgrounds is due to QCD multi-jet. The third assumption is that the probability for QCD multi-jet “leptons” to pass the medium WP of the DeepTau anti-jet discriminator (VS jet) is independent of the charge likeness of the pair of “leptons” in the event. Again, this is reasonable, since DeepTau acts on a per τ_h basis, so has no knowledge of other leptons in the event, and is not expected to have a charge bias.

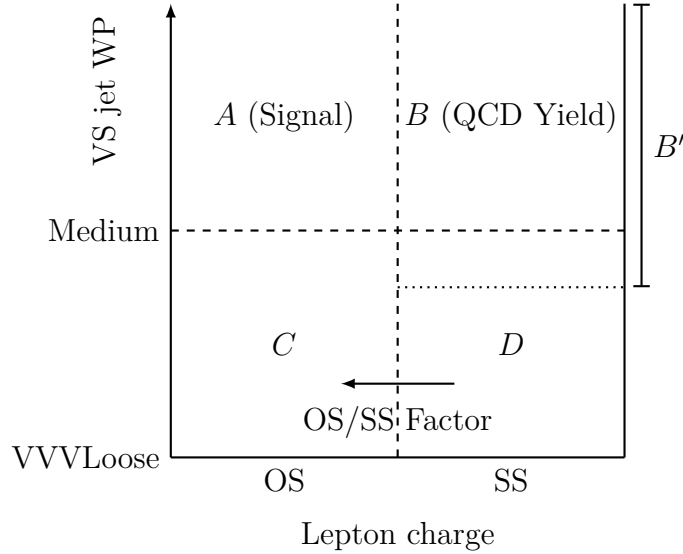


Figure 10.2: Illustration of the ABCD regions used for QCD contribution estimation. B' represents the extension of control region B used for modelling the distribution of QCD in Region A , having been rescaled by the OS/SS factor, as discussed in the text.

The deficits in regions B , C , and D are computed:

$$N_i = N_i^{\text{data}} - N_i^{\text{MC}} \quad i \in \{B, C, D\}, \quad (10.7)$$

where N_i is the number of events in region i . An OS/SS factor is derived by computing the ratio of QCD yields in regions C and D N_C/N_D . This can then be used to extrapolate the QCD yield passing the anti-jet discriminator (N_B) from the same-charge region to the opposite-charge region (signal region).

This, however, only gives the expected yield of QCD multi-jets. Inevitably, we will want to estimate the differential distribution of QCD as a function of, for example, and the di-Higgs mass or the output of a neural network. Region B alone, does not have enough events to accurately model the distribution. Instead, the requirement on VS jet may be slightly relaxed, allowing the addition of parts of Region D to be included in a larger Region B' , and so improving the accuracy of the shape modelling of QCD in Region A . Shapes are computed by estimating the density using histograms and then computing the data deficit in each bin.

10.4.2 Drell-Yan background corrections

Corrections to the Drell-Yan samples uses events in the $\mu\mu$ channel, in which a pair of isolated, opposite-sign muons are present. First the LO samples are reweighted to match the NLO samples in the generator-level Z -boson p_T distributions according to the generator-level multiplicity of b -jet and light-jet outgoing partons in the samples.

Having corrected the shapes of the LO DY samples, the normalisation is corrected. These corrections take the form of multiplicative scale factors that further reweight the reweighted LO DY samples in bins of reconstructed Z -boson p_T and generator-level b -jet multiplicity. Computation of the nominal scale factors is performed using events in the MH sample (as detailed in Sec. 10.3.9). The uncertainties on the scale factors are estimated by repeating the process on events in the MTT sample (as detailed in Sec. 10.3.9) and checking the differences in the scale factors compared to those computed for the MH sample. In order to focus more on the Drell-Yan background, an additional cut of $\vec{p}_T^{\text{miss}} < 45 \text{ GeV}$ is applied to help suppress the other dominant background, $t\bar{t}$.

The samples are then split into six jet-categories according to the number of jets passing the medium b -tag WP and the number of light jets present. Scale factors are computed via a simultaneous fits to the observed data for the $Z \rightarrow \mu\mu$ candidate p_T in three control regions defined using the generator-level b -jet multiplicity: 0, 1, and ≥ 2 b jets.

10.5 Analysis and inference approach

10.5.1 Non-resonant search

The non-resonant search consists of two sub-searches: production via gluon fusion (ggF) and production via vector-boson fusion. The general aim is to classify signal ($hh \rightarrow$

$\tau^+ \tau^- b \bar{b}$) and background using DNN discriminants and use the distributions of the events as a function of the discriminators in order to infer the presence of signal in the collider data. In order to highlight the contributions to signal from VBF (which has a much lower production cross section compared to ggF, but provides a source of light-jets which can be triggered on), the VBF jet categories are used. From Tab. 10.5, though, we can see that a significant amount of ggF signal also enters the VBF category and effectively acts as a background to the VBF search. If the ggF signal were to be treated as a background, however, it would be washed out by the presence of much larger backgrounds, and the discriminant would likely overlook its contributions. Instead, a second approach is used in which the discriminant in the VBF categories is trained to predict multiple different process categories: VBF signal, ggF signal, $t \bar{t}$, and Drell-Yan - a *multi-class classification*. In this way the contributions of VBF, ggF, and other backgrounds may be weighted equally during training.

The multi-class classifier (described in Ref. [280]) was not developed by me, and only acts on events entering the VBF category. Events entering the VBF category are assigned entirely to the category with the highest score according to the multi-class classifier. In the remaining jet categories, VBF signal is not considered and binary classification of signal and background is performed. Following the training of both the multi-class classifier and my binary classifier, it was found that it was optimal to then use the my binary classifier to re-score events in each of the multi-class categories.

Figure 10.3 illustrates the event categorisation and the discriminants responsible for each category. A simultaneous fit to the collider data of the density-distributions of events in each of the inference categories, per decay channel, will be performed in order infer the presence of signal.

10.5.2 Resonant search

For the resonant search, only ggF production is considered, enabling the use of a single binary classifier. As discussed further in Sec. 10.8, to help deal with the range of signals being searched for, a parameterised classifier may be trained, which adjusts its response according to the mass and spin of particle being hypothesised. Whilst we shall see that such a parameterised classifier is able to fit well to a range of signal, and interpolate between them, it comes with the downside that data must be passed through the classifier multiple times, which vastly increases the inference time of the analysis.

An alternate approach is to instead remove any notion of the di-Higgs mass from the classifier inputs and train an unparameterised model. At inference time, a two-dimensional fit to the event distributions in the DNN prediction and the reconstructed di-Higgs mass (using `HHKinFit`) may be performed, in order to combine the accurate classification of background of the DNN, with the signal sensitivity of the di-Higgs mass.

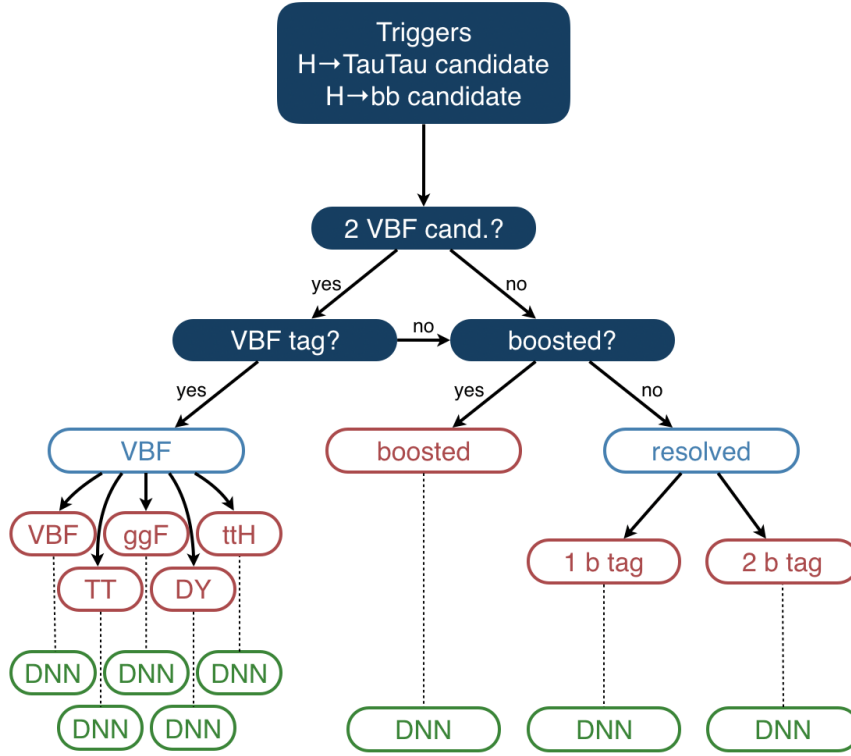


Figure 10.3: Illustration of the analysis categories per channel per year. Events entering the VBF jet-category are classified by the multi-class classifier into five categories: two signal (VBF and ggF) and three backgrounds. Events in each category (multi-class and jet) are scored by my ggF binary-classifier (DNN in the diagram). A total of eight inference categories for each channel means 24 categories per year and 72 categories in total. Image source [273].

10.5.3 Systematic uncertainties

Various sources of systematic uncertainty are considered when inferring the presence of signal in the data. Some of these only affect the overall yield of contributing processes, others can affect the shape of feature-distributions and hence the shape of the discriminator distribution.

Yield systematics

- Luminosity: Uncertainties in the range 2.3-2.5 % [281] account for the uncertainty on the luminosity measurement in each year. Each uncertainty is fully correlated between processes in a given year, but each year is fully uncorrelated with the other

years. These uncertainties are not applied to the QCD and Drell-Yan processes, which are estimated from collider data.

- e and μ ID efficiency: 1 %, as computed from MC-Collider scale-factors.
- L1 pre-firing: 2 % for MC processes in 2016 and 2017.
- b -tag efficiency: Uncertainties vary in the range 2-6 % [282] as a function of jet p_T and $|\eta|$.
- HH cross-section: Uncertainties on the theoretical cross-sections for the signal production-processes as computed in Ref. [283] are considered when computing limits with respect to the Standard Model coupling ($\kappa_\lambda = 1$):
 - ggF: $\sigma_{-5.5\%}^{+2.2\%}(\text{scale}) \pm 3\%(\text{PDF} + \alpha_s) \pm 2.6\%(m_{\text{top}})$
 - VBF: $\sigma_{-0.04\%}^{+0.03\%}(\text{scale}) \pm 2.1\%(\text{PDF} + \alpha_s)$
- Higgs branching fractions: Assuming $m_H = 125 \text{ GeV}$, total uncertainties on the branching fractions are $+1.25\% - 1.27\%$ for $h \rightarrow b\bar{b}$ and $\pm 1.65\%$ for $h \rightarrow \tau^+ \tau^-$ [284].
- MC background cross-sections: Uncertainties vary by process but average to 5 % due to factorisation and renormalisation scales, and 3 % due to PDFs and α_s .
- Collider-data-driven background yields: For Drell-Yan, the uncertainties computed when estimating the Z +jets normalisation are propagated to the signal region. For QCD contributions, the uncertainties are computed assuming Poisson fluctuations on the number of events landing in the three control regions.

Shape systematics

- τ_h energy scale: For true τ_h and fake τ_h via electrons, the uncertainties on the measurement of a given τ_h -candidate's energy varies according to its decay mode and p_T . For fake τ_h via muons, a flat 1 % uncertainty is used. Uncertainties are provided by the CMS Tau POG in Ref. [285].
- DeepTau scale factors: Uncertainties are derived by the CMS Tau POG in Ref. [285] and affect the overall event-weights. For the anti-jet discriminator, uncertainties are derived according to candidate p_T . The uncertainties for anti-electron and anti-muon discriminators are derived in bins of pseudorapidity.
- Tau-trigger scale factors: These uncertainties also modify the overall event weights, and vary as a function of the decay modes of the tau-candidates matched to the trigger object. Again, they are provided by the CMS Tau POG in Ref. [285].

- Jet energy scale: 11 sources of uncertainty are considered, as provided by the CMS JetMET POG in Ref. [286].
- Top-quark p_T reweighting: The effect of the uncertainty on the p_T of top-quarks is modelled by producing alternative shapes through p_T reweighting, as recommended by the CMS Top POG in Ref. [287].

10.6 Feature definitions

10.6.1 Approach

For feature extraction from the datasets I take an inclusive approach, computing a large range of possible features. Invariably, this will lead to a large number of features, too many to validate and use as inputs to a ML classifier. In Sec. 10.7 I will discuss how we can filter the full feature-menu down to just the most important features.

Depending on the search being performed, some features, particularly the VBF-related ones, may not be definable. Similarly, `HHKinFit` and `SVFit` are not guaranteed to converge and therefore their related features are not always valid. Invalid or undefined features are set to `NaN` (Not a Number), for now and will be processed further, later.

For convenience, I'll refer to the light lepton or most isolated τ_h as τ_1 and the τ_h or next most-isolated τ_h as τ_2 . The two b -jet candidates are ordered by DeepFlavour score: b_1 has the higher score and b_2 the lower.

10.6.2 Higgs and di-Higgs construction

The $h_{b\bar{b}}$ candidate is constructed from the vector sum of the two b -jet candidates. Several $h_{\tau^+\tau^-}$ vectors are built:

- $h_{\tau^+\tau^-,\text{vis.}} = \tau_1 + \tau_2$; $h_{\tau^+\tau^-}$ built from the visible decay-products.
- $h_{\tau^+\tau^-,\text{MPT}} = \tau_1 + \tau_2 + \vec{p}_T^{\text{miss}}$; $h_{\tau^+\tau^-}$ assuming that all of the missing transverse momentum (MPT) was due to the neutrinos in the tau decays.
- $h_{\tau^+\tau^-,\text{SV}}$; $h_{\tau^+\tau^-}$ as computed using `SVFit`, if it converges.

The di-Higgs vector is defined using the best available information for each event:

- Ideally the 3-momenta is set using the sum of $h_{b\bar{b}}$ and $h_{\tau^+\tau^-,\text{SVFit}}$, and the mass is set using `HHKinFit`;
- If `HHKinFit` does not converge but `SVFit` does, then both the 3-momenta and mass are set using $h_{b\bar{b}}$ and $h_{\tau^+\tau^-,\text{SVFit}}$;

- Else if `HHKinFit` does converge but `SVFit` does not, then the 3-momenta is set to the sum of $h_{b\bar{b}}$ and $h_{\tau^+\tau^-,\text{MPT}}$, and the mass is set using `HHKinFit`;
- When neither `HHKinFit` nor `SVFit` converge, then both the 3-momenta and mass are set using $h_{b\bar{b}}$ and $h_{\tau^+\tau^-,\text{MPT}}$.

10.6.3 Feature definitions

Momenta and energy features

The Cartesian 3-momenta, p_T , and energy of the lepton, b -jet, $\vec{p}_T^{\text{miss}}$ (p_z not included), and VBF jet (when defined) candidates are extracted; 34 features in total. The p_T and energies of $h_{\tau^+\tau^-,\text{SV}}$, $h_{b\bar{b}}$, and hh are also extracted; six features.

For VBF searches, the sign of the product of the pseudorapidities of the two VBF jets is also considered; i.e. whether both jets have the same longitudinal direction.

Masses

Six transverse masses are computed:

- The transverse masses (m_T , see Eq. 7.3) of the two lepton candidates, $h_{\tau^+\tau^-,\text{MPT}}$, and $h_{\tau^+\tau^-,\text{SV}}$ computed with respect to `MPT`.
- $m_{T,\text{tot.}}$ the quadratic sum of m_{T,τ_1} , m_{T,τ_2} , and the transverse mass of one lepton computed with respect to the other.
- The transverse mass (m_{T2} see Ref. [215]) of the event.

Higgs masses considered are those of $h_{b\bar{b}}$, $h_{\tau^+\tau^-,\text{vis.}}$, and $h_{\tau^+\tau^-,\text{SV}}$. Five di-Higgs masses are also computed:

- $m_{hh,\text{KinFit}}$, the `HHKinFit` mass;
- $m_{hh,\text{MPT}}$ the invariant mass of $h_{b\bar{b}}$ and $h_{\tau^+\tau^-,\text{MPT}}$;
- $m_{hh,\text{vis.}}$ the invariant mass of $h_{b\bar{b}}$ and $h_{\tau^+\tau^-,\text{vis.}}$;
- $m_{hh,\text{SV}}$ the invariant mass of $h_{b\bar{b}}$ and $h_{\tau^+\tau^-,\text{SV}}$;
- $m_{hh,\text{red.}}$ the reduced di-Higgs mass, computed as $m_{hh,\text{MPT}} - m_{h_{\tau^+\tau^-,\text{MPT}}} - m_{h_{b\bar{b}}}$.

In order to help reject $t\bar{t}$ contributions, mass pairs are computed under the hypothesis of a $t\bar{t}$ decay for all combinations of the leptons, b -jets, and `MPT`. The pair of masses which together most closely match the top-quarks mass are used; two features $m_{\text{top},1}$ and $m_{\text{top},2}$.

Angular separations

A range of $|\Delta\phi|$, $|\Delta\eta|$, and ΔR separations in the lab rest-frame are computed between the leptons, b -jets, VBF jets, Higgs candidates, and MPT: 14 $|\Delta\phi|$, five $|\Delta\eta|$, and five $|\Delta R|$ features. Additionally the $|\Delta R|$ between the b -jets is computed in the $h_{b\bar{b}}$ rest-frame, and the ΔR between the leptons is computed in the rest-frames of both $h_{\tau^+\tau^-,\text{MPT}}$ and $h_{\tau^+\tau^-,\text{SV}}$; three boosted ΔR features.

p_T scaled ΔR features are also computed: $\Delta R(b_1, b_2) \times p_{T,h_{b\bar{b}}}$, $\Delta R(\tau_1, \tau_2) \times p_{T,h_{\tau^+\tau^-,\text{MPT}}}$, and $\Delta R(\tau_1, \tau_2) \times p_{T,h_{\tau^+\tau^-,\text{SV}}}$.

Seven Φ angles are computed for a variety of particles and reference frames, according to:

$$\begin{aligned}\Phi(\vec{v}_0, \vec{v}_1, \vec{v}_2, \vec{v}_3, \vec{r}) &= \frac{\arccos(\vec{n}_0, \vec{n}_1)}{|\vec{n}_0| |\vec{n}_1|} \\ \vec{n}_0 &= \vec{v}_{0,\vec{r}} \times \vec{v}_{1,\vec{r}}, \quad \vec{n}_1 = \vec{v}_{2,\vec{r}} \times \vec{v}_{3,\vec{r}},\end{aligned}\tag{10.8}$$

where $\vec{v}_{i,\vec{r}}$ is vector $\vec{v}_i$ boosted to the rest-frame of reference vector $\vec{r}$. $\vec{n}_0$ and $\vec{n}_1$ are therefore the normals to the planes defined by two pairs of boosted vectors and Φ is the angle between the two normals.

19 $\cos(\theta)$ angles are computed of particles and reference frames, according to:

$$\cos(\theta_{\vec{v}_0, \vec{v}_1, \vec{r}}) = \vec{v}_{0,\vec{r}} \cdot \vec{v}_1,\tag{10.9}$$

where $\vec{v}_{i,\vec{r}}$ is vector $\vec{v}_i$ boosted to the rest-frame of reference vector $\vec{r}$, and $\vec{v}_1$ is normally the same vector as $\vec{r}$. For the cases in which $\vec{v}_1$ is the z -axis, the feature is labelled $\cos(\theta^*)$.

Jet flags

Information about the jets is encoded in two quality flags for the b -jets, indicating the most stringent DeepFlavour WP they pass, and whether or not the event fulfils the conditions to be characterised as *boosted*.

Miscellaneous features

Two topological discriminants, p_ζ and $p_{\zeta,\text{vis.}}$, are computed by projecting τ_1 , τ_2 , and MPT (if p_ζ) onto an axis given by the bisector of the momenta in the transverse plane of the visible decay products of the tau leptons, as defined in Ref. [288].

The χ^2 of the `HHKinFit` is used. It indicates both the quality of the fit, and whether or not the fit actually converged. Similarly, a flag for whether or not `SVFit` converged is used.

10.7 Methodologies for automated feature-selection

In Sec. 9.4.2 I discussed a process of filtering a set of possible input features down to only the most useful and unique features. However in the context of the HiggsML dataset, such methods were of limited use due to the features available already being quite limited. In the case of the search performed here, though, we have taken the approach of computing a large range of features in the hope that a subset of them are actually useful. In this section I introduce an expanded version of the process in Sec. 9.4.2 which offers several benefits: being automated by a few hyper-parameters; running more quickly on larger datasets with more features; providing resistance to noisy features; and accounting for dataset bias when looking to train a pair of discriminators on each halve of the full dataset, as we are in this case.

Dataset bias

The datasets used here originally came from the same dataset, and the split was performed in a way that was invariant of any physical process, class, kinematic phase-space, *et cetera*. It is not expected, then, that a given feature should be found to be useful for one dataset and not the other, unless said feature is being used to fit to some noise particular to one of the datasets, and is not representative of the underlying data distribution. Using such a feature may well provide improved performance on one dataset, but this improvement would not generalise to the other dataset. We therefore want to ensure that our feature selection is performed in a way which selects features that are useful on both datasets.

One possible way to achieve this might be to perform feature selection twice, from scratch, on each dataset separately, and then only use the features used by both datasets. Consider, though, the case in which a pair of strong features, A and B , are correlated to the extent that one of the pair can be removed without a drop in performance, but the removal of both does. If in the first feature selection A is removed, and in the second feature selection B is removed, then when taking the set of mutually useful features, neither A nor B will be included, resulting in a drop in potential performance. Ideally, both feature selections should have removed the same feature, allowing the other to be included, however this is not guaranteed.

The approach used here is to run feature selection on the first dataset. Feature selection is then performed on the second dataset, but only considering the features found to be useful on the first set. In this case any features useful only for the first dataset will be removed in the second selection, and any features useful only for the second dataset will not be considered. In the above example, the decision of whether to select A or B is fixed in the first selection, removing the possibility of both A and B being accidentally excluded.

10.7.1 Constant features

The number of unique values of each feature are checked, and any features with just one unique value are removed. This step is mostly a sanity check; no features should have constant values except those appropriate to the search e.g. the resonant mass will be at a constant default value for the non-resonant search.

10.7.2 Subsampling

The methods which follow make heavy use of ML algorithms, which can be slow to run if a large dataset is used. Whilst a large dataset can be expected to improve the performance of the ML algorithms, we are more interested in the relative changes in performance. Because of this we can save time by only running a fraction of the full dataset: either a fixed subsample can be used, or subsamples can be extracted on demand. I opt for the latter since the change in performance can then be averaged over several subsamples.

Subsamples are extracted by specifying the number of datapoints to extract and then running a stratified split (see Sec. 5.4.2), with an appropriate stratification key. In this case the key is generated using the MC process ID and the jet category ID.

10.7.3 Random Forest hyper-parameter optimisation

Similar to Sec. 9.4.2, the feature selection methods will use Random Forests (RFs, see Sec. 5.3.2). Since the number of features used will change considerably throughout the selection process, it is important that the RFs remain optimal, meaning that their hyper-parameters must be re-optimisable in an quick and automated fashion.

The two most important considerations for an RF are the stopping criterion and the feature-subsampling fraction. To control the stopping criterion, I adjust the minimum number of samples required per node, which prevents the creation of new child nodes if the one of them would contain fewer datapoints than the minimum amount, thereby preventing the RF from working in phase-space regions where the statistical uncertainties on the data distributions are high. Other hyper-parameters are left at suitable default values.

Optimisation works by first selecting the optimal value for the minimum number of samples required per node using several possible values, with the feature sub-sample fraction set to the default value of $\sqrt{N}^{-1}$, where N is the total number of features. Next, the optimal feature-subsample fraction is optimised, again by checking a range of values. For each test, an RF is trained on a subsampled training set and then evaluated on a separate validation set, which may or may not be resampled.

An example of this hyper-parameter optimisation is shown in Fig. 10.4. The lack of a full grid-search, or more thorough methods, is in line with the requirement that

the optimisation is quick and *good enough*. Whenever the RF hyper-parameters are re-optimised, a new subsample of training data is extracted for this purpose, in order to limit the effect of overfitting the hyper-parameters to the data.

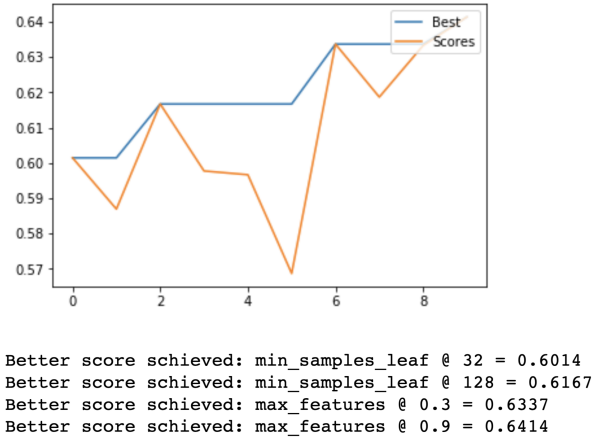


Figure 10.4: Example of RF hyper-parameter optimisation, checking various metric values (in this example the weighted accuracy) for minimum number of samples required per node and the feature sub-sample fraction. The y-axis shows the metric value for tested sets of hyper-parameters (indexed on the x-axis).

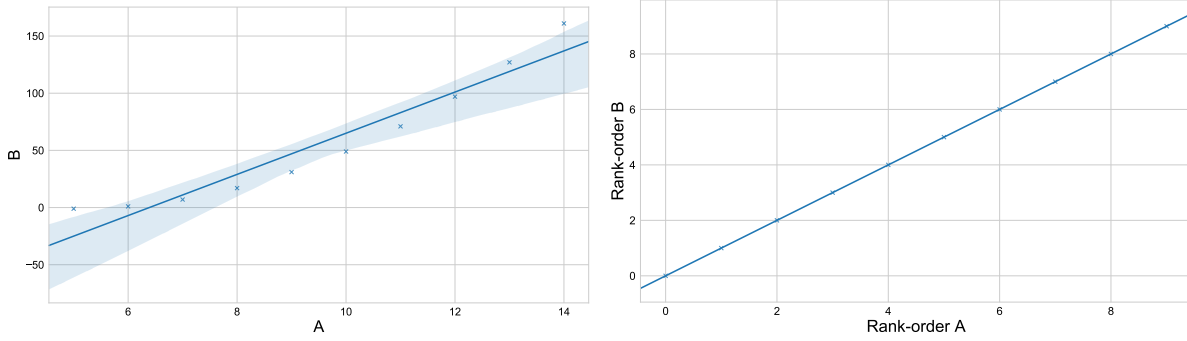
10.7.4 Monotonic relationships

Spearman's rank-order correlation coefficient

The use of Spearman's rank-order correlation coefficient (SROCC) [247] was mentioned in Sec. 9.4.2. This can be used to highlight monotonic relationships between pairs of features by transforming by ordering datapoints according to increasing values one feature (the *rank order*) and another. The correlation between a pair of features can then be computed using the linear correlation under rank-ordering for that pair of features. A high degree of correlation means that one of the features is just a monotonic transformation of the other, and little to no new information is included. Lower correlation means that each feature contains a higher degree of unique information. An example of this is shown in Fig. 10.5. Note that SROCC is only computed for continuous features.

Hierarchical clustering

The problem is how to scale this method to deal with all combinations of features. The solution is to compute the absolute values (to account for anti-correlation) of SROCC for



(a) Example of the correlation between two features A and B . We can clearly see that the two features are related and need to quantify this. (b) The same example with rank-ordering. The features now show perfect linear correlation and we can assume that A and B are monotonic transformations of one another.

Figure 10.5: Example of the use of rank-ordering to identify monotonic relationships.

all pairs and perform hierarchical clustering of features using SROCC as a Euclidean distance. This means that the threshold on SROCC begins from one (completely correlated) and is decreased. When a pair of features has a SROCC equal to the threshold, they are clustered. This continues, with the possibility of new features joining previously clustered features, until all features have been clustered. Pairs which were clustered at a lower distance are more correlated than those clustered at a greater distance. For the purpose of feature selection, we can say that clusters below a certain distance are candidates for removal. An example of this clustering is shown in Fig. 10.6.

Feature removal

Correlated features share a greater amount of mutual information and so it is likely that the removal of one of them will not decrease performance. To test this I train a certain number (in this case five) of RFs on subsamples of the training data, evaluate their performance on the validation data (or one could use the out-of-bag score), and compute the average performance. This is performed three times: once on the full dataset to get a baseline score, twice more each time with the removal of a different feature. If the removal of one of the features does not decrease performance compared to using all features then it is removed. If both features do not cause a decrease in performance, then the feature whose removal produces the better performance is removed. An example of this is shown in Fig. 10.7.

10.7.5 Feature importance

Having removed correlated features, the next step is to select features which are useful in solving the given problem. This is based around computing the permutation importance of each feature.

Permutation importance

This aims to quantify how much a given feature influences the model's output. Features with a higher importance will have a greater influence overall than those of lesser importance. The permutation importance (PI) is computed by first training the model as normal and evaluating its performance on the training set. Next, a copy of the training data is made and one of the features is randomly shuffled datapoint-wise, thereby destroying any information carried by that feature for each datapoint, whilst keeping its values at the same scale as seen during training. The performance of the model on the shuffled data is computed and the fractional difference in performance compared to the original performance is the *fractional permutation importance* of the feature which was shuffled. If shuffling the feature caused a large change in performance, then that feature was important to the model. If the change was small, then the feature was less important.

This process of copying and shuffling is repeated for every feature. By retraining new models, an average PI can be computed for each feature. Although the dataset is the same each time, the feature subsampling at each split acts as a source of stochasticity. Here I compute PIs using an average of five retrainings. A threshold on the PI can then be used to select features by minimum importance. As a final check, one can train several models using all features and then just the selected features and compare the average performance to ensure that the removal of the features did not lead to a decrease in performance. An example of selection by averaged PI is shown in Fig. 10.8.

It is important that PI is computed after correlated features have been removed: if a pair of strong but correlated features were present, then the removal of either would not necessarily decrease the model performance dramatically, since the model may have learnt to use both of them, in which case the feature importance would be shared by the pair of features. The fact that neither feature alone has the full feature importance it deserves, might mean that both features end up failing to meet the PI threshold, resulting in the removal of both features.

Repeated PI selection

Note in Fig. 10.8 that the error bars are quite large. It is possible that due to statistical fluctuations unimportant features may pass the selection, or important features may be left out. To better account for these cases, the process of feature selection can be repeated several times on different subsamples of training data, and resamples of the validation data. We can then set a threshold on the minimum number of times (or fraction of times)

a feature must pass selection by PI in order to be considered *important*. Again, RFs can be used to check that the filtered set of features do not lead to a drop in performance. An example of this repeated selection by PI is shown in Fig. 10.9, where feature selection is repeated ten times. Since most of the time is spent computing the PI, the whole process can be sped up by running multiple feature-selections in parallel, given sufficient CPU cores and memory.

10.7.6 Removal by mutual dependence

Mutual dependence

Whilst SROCC can help remove monotonically related features, it is not guaranteed to highlight features which display non-monotonic relationships with other (sets of) features. To highlight this *mutual dependence* we can instead train regression models to predict the value of a given feature using the other features as inputs. If a feature can be predicted well, then we can compute the PI of the regressor inputs in order to find which features are used.

Feature which can be well predicted are then candidates for removal testing, provided they themselves are not important in predicting the values of other regressable features. An example of the mutual dependence table that can be computed is shown in Fig. 10.10. Each row corresponds to a separate regression training to predict that row's feature. The dependence column indicates the score of the regressor (higher indicates better regression), and the remaining columns indicate the PI of each feature as used by that regressor (only PIs above a certain level are shown). I consider features with a dependence above 0.8 as candidates for removal.

Feature removal

Removal by mutual dependence follows a similar strategy to that of removal by SROCC, except now that multiple features can be tested, but the mutual dependencies must periodically be reevaluated.

Using the order of increasing feature importance as computed in the previous section, features with MD higher than the threshold are added to a testing set provided they are not important in predicting a feature already added to the testing set. As an example, in Fig. 10.10 in the first round of feature testing we would add `dR_b1_b2_x_h_bb_pT`, `costheta_l2_httmet`, `deta_l1_l2`, `dphi_httmet_met`. However when we try to add `dphi_httvis_met` we see that it is important in predicting `dphi_httmet_met`, which is already in our testing set. At this point we would stop and perform a removal test by evaluating the average performance of RF models on the full range of remaining features and then by removing each feature in turn (the same as for removal by SROCC). The difference being that now we remove any feature that doesn't cause a decrease in per-

formance compared to using all features; by construction, the set of features tested are independent of one another, so the removal of one feature shouldn't impact the change in model performance when another feature is removed, allowing the safe removal of multiple features.

At this point, the MDs must be recomputed using the remaining features since some feature will no longer be regressable. The process then continues moving up the list of dependent features in order of importance.

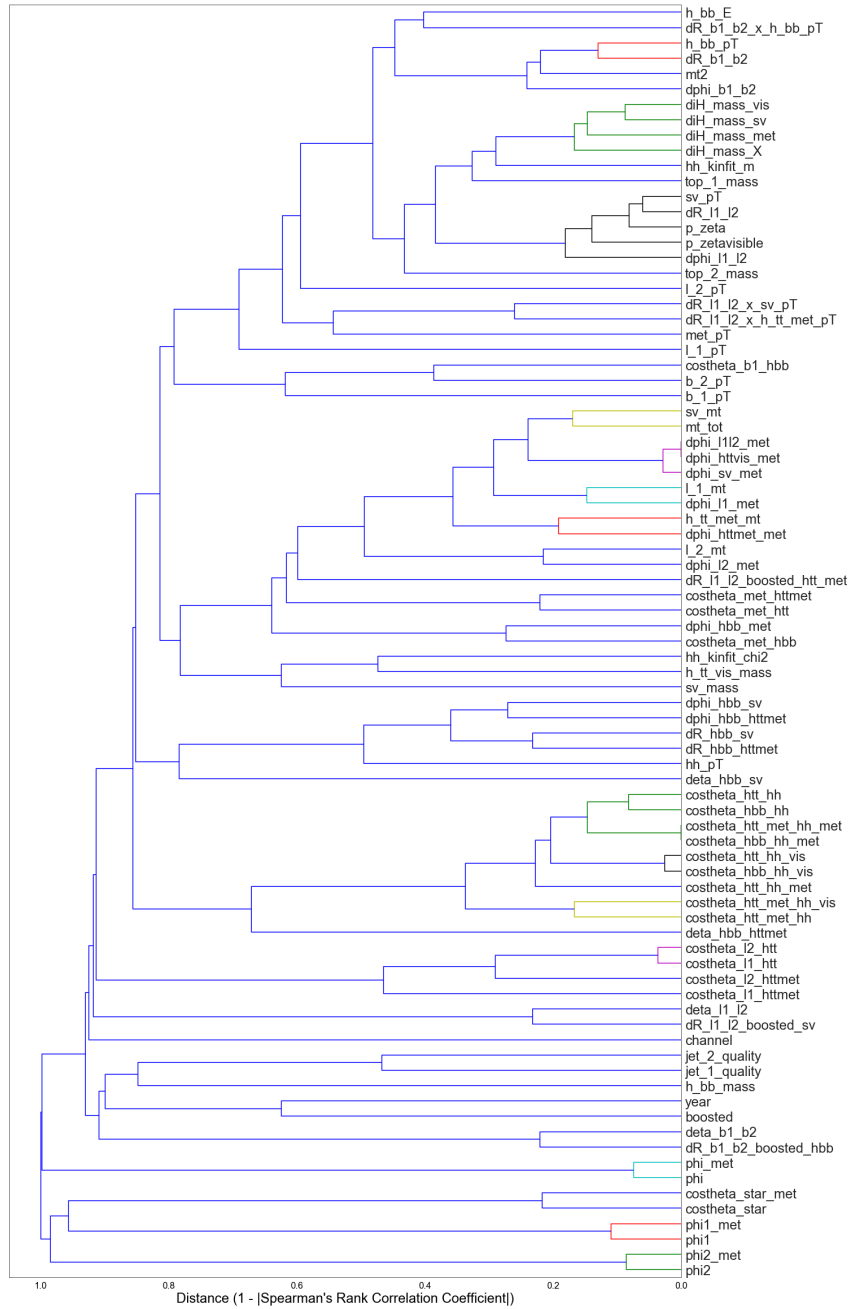


Figure 10.6: Dendrogram illustrating the hierarchical clustering of features by Spearman's rank-order correlation coefficient. Features clustered with a smaller distance display greater linear correlation. Pairs with a distance less than the threshold (0.2) are coloured non-blue.

Checking pair: ['costheta_htt_met_hh', 'costheta_htt_met_hh_vis']

Removed	OOB Score	Val Score
None	0.732±0.009	0.779±0.009
costheta_htt_met_hh	0.742±0.006	0.791±0.003
costheta_htt_met_hh_vis	0.732±0.01	0.786±0.005

Dropping costheta_htt_met_hh

Figure 10.7: Example removal check for two correlated features. The removal of either results in a validation score improvement, however the removal of `costheta_htt_met_hh` results in the high score, therefore it is removed.

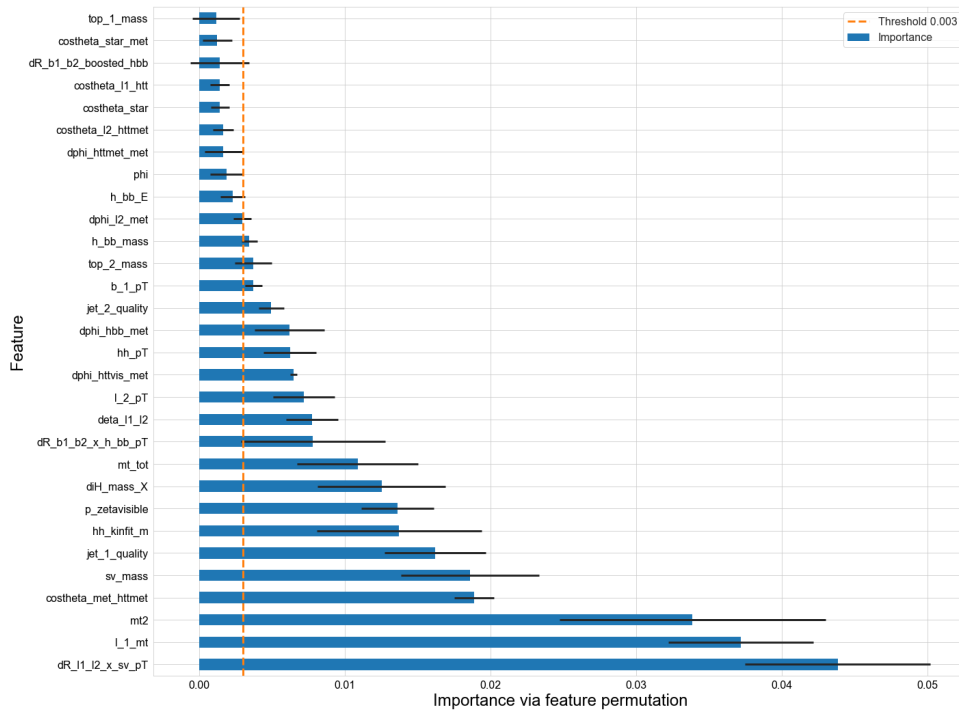


Figure 10.8: Example feature selection by averaged permutation importance. **N.B.** only the top 30 features are shown.

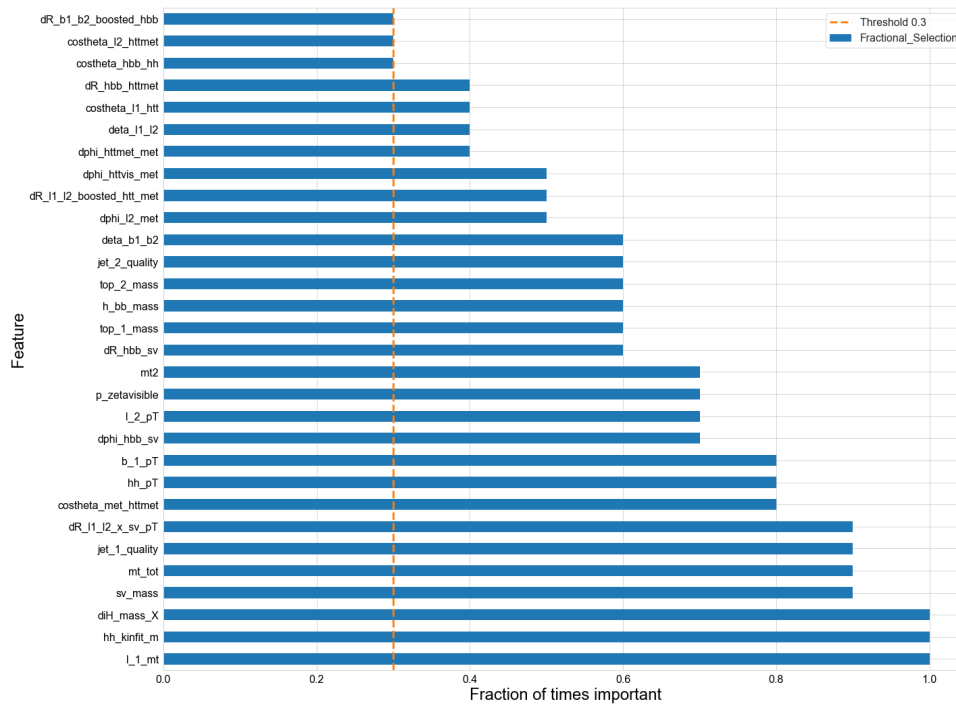


Figure 10.9: Example repeated feature selection by averaged permutation importance. **N.B.** only the top 30 features are shown.

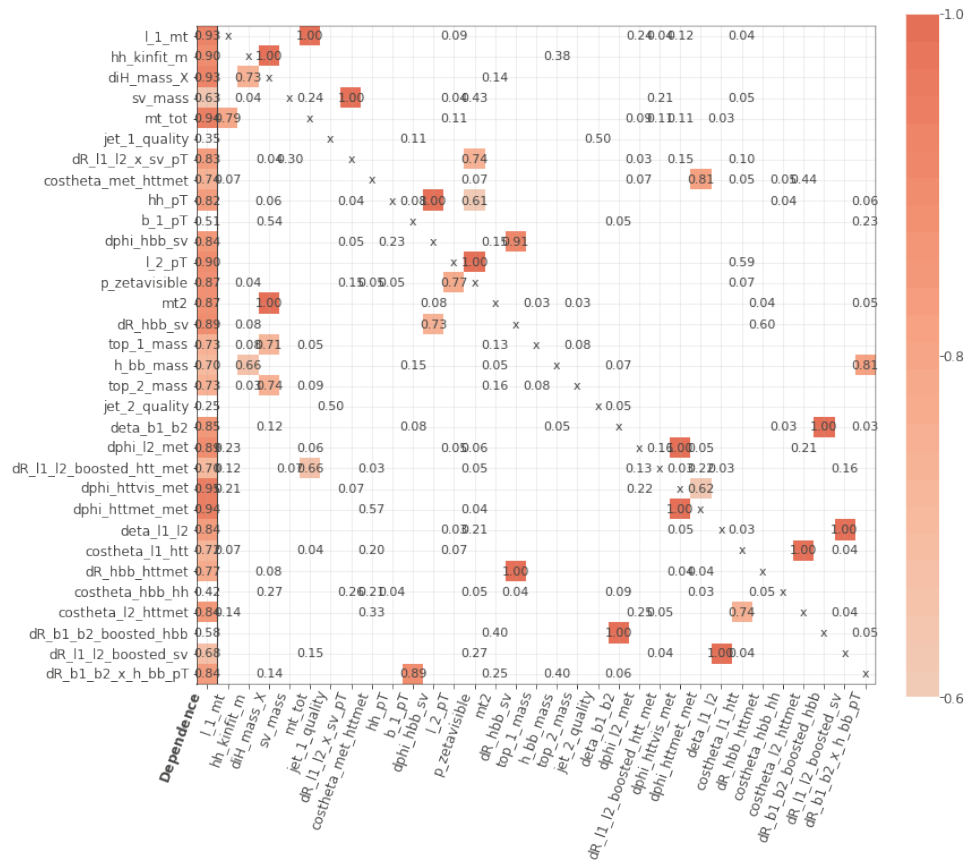


Figure 10.10: Mutual dependence of features as estimated via Random Forest Regression. The “Dependence” column indicates ease of prediction of that row’s feature values, and the other columns indicate the relative importance of each feature in predicting the dependent feature.

As an example, `l_2_pT` is easily predictable (dependence of 0.90), and the most important feature in predicting its values is `p_zetavisible` (importance of 1.00).


```

Checking ['l_2_pT', 'hh_pT', 'mt_tot', 'diH_mass_X']
+-----+-----+-----+
| Removed | OOB Score | Val Score |
+-----+-----+-----+
| None    | 0.516±0.002 | 0.631±0.006 |
| l_2_pT  | 0.512±0.004 | 0.637±0.001 |
| hh_pT   | 0.52±0.003  | 0.637±0.005 |
| mt_tot  | 0.52±0.003  | 0.627±0.002 |
| diH_mass_X | 0.512±0.003 | 0.627±0.001 |
+-----+-----+-----+
Dropping l_2_pT
Dropping hh_pT

```

Figure 10.11: Example removal check for several dependent features. The removal of two of which results in a validation score improvement, therefore they are both removed.

10.8 General approach to discriminator training

10.8.1 General approach

The general approach used here follows that which was used in the HL-LHC projection study (see Sec. 9.9), using the best performing architecture setup found during Sec. 9.4. In this approach the Monte Carlo data are split evenly into two sets. A pair of neural-network discriminators are then trained, each on different halves of the data. At inference time the discriminators are used to predict the classes of events in the halves of the data on which they were not trained. In doing so, all of the Monte Carlo data may then be used for inference, thereby reducing the associated statistical uncertainty on the sample density-distributions, whilst not being subject to a systematic bias on the predictions which would otherwise arise if the discriminators were applied to data on which they were trained.

It should be noted that this approach does not provide an explicit validation sample and does therefore not permit any fine-tuning of hyper-parameters without the risk of unknowingly biasing the model to the testing data. From Sec. 9.4.12, though, we can see that the architecture and training scheme has only a very weak dependence on architecture parameters, so I do not expect the lack of tuning to damage our potential performance too drastically. What changes I do test will be sufficiently coarse as to either provide a definite improvement or deterioration, which can be expected to be reproduced on any dataset similar to the training and testing sets.

10.8.2 Architecture and training

Each discriminator is an ensemble of ten neural networks trained via ten-fold stratified cross-validation of their training data, and weighted by validation-fold performance:

1. A given half of the training data is split into ten folds ensuring that the fractions of samples, channels, and jet categories in each fold are as close as possible to the overall fractions in that half of the training data;
2. A network is then trained on nine of the folds using the tenth fold as validation data to monitor the generalisation of the network during training;
3. Once the training cycle is finished, the model state corresponding to the best performance on the validation data is loaded and the model is assigned an importance weight based on the inverse of its loss (event-weighted binary cross-entropy) on the validation data;
4. Further networks are trained, each using a different fold of the data as their validation sample, until all ten folds of the training data have been used exactly once;

5. The importance weights of the ensemble are normalised such that their sum is one;
6. The prediction of the ensemble for a given data point is the sum of each model's prediction for that data point times that model's importance weight: $p_{\text{ensemble}}(x) = \sum_i^N w_i p_i(x)$.

The architecture of the networks used is the same as the optimal architecture developed in Sec. 9.4.12, with the addition of Dropout [187]:

- Entity embedding of categorical features with embedding size $M = (1 + N)/2$ for feature cardinality N [249, 250];
- Six densely connected hidden layers of widths equal to the number of continuous input features plus the sum of each embedding sizes for categorical input feature (see Sec. 9.4.10);
- Swish-1 activation functions for hidden layers [166];
- Single output neuron with sigmoid activation function.

Training of the model uses:

- Learning rate range tests to find optimal learning rate [248];
- ADAM optimisation [176];
- 1cycle schedule for learning rate and β_1 over 15 epochs with cosine interpolation [258, 261];
- Dropout of 10 % for non-resonant models and 5 % for resonant models.

10.8.3 Parametrised learning

Since the resonant searches use multiple hypotheses for the mass and spin of the resonant particle, it is important that the classifier is optimal for all mass points. One way to achieve this would be to train a single classifier for each mass point, as was done in the 2018 ATLAS analysis (Ref. [217]), however for the number of mass-points considered in this study, this approach is a time-consuming and resource-heavy one. Additionally, some mass-points have very low numbers of events for the signal, meaning that training is difficult. Ideally, the training method should use all of the training data to be used in a fashion that allows performance at under-populated mass-points to be increased by extrapolation from, or interpolation between, neighbouring mass-points.

Described in Ref. [31], parametrised networks can be trained and applied to classification tasks in HEP, in which only one of the classes has a true value for the parametrisation

feature by assigning the true value to events in that class, and a randomly sampled value for the other class during training. For at application time, the parametrisation feature can be set to a specific value, to evaluate the response of the model at that feature point.

Effectively this means that the dataset is assigned a continuous feature to provide the resonant mass. For signal events this feature is set to the generator level mass of the heavy Higgs for which the sample was generated. For background events during training this resonant mass is sampled randomly from the set of possible masses. Evaluation then takes place at each resonant mass point, during which only signal events with that specific mass point are considered, and all the background events are considered but their resonant mass is set to that of the resonant mass currently being evaluated. Additionally, the spin of the resonance can be used for parameterised training using a single categorical feature. Parameterised learning has already been tested in this scenario and found to be useful in Ref. [216].

10.8.4 Feature pre-processing

Continuous features are normalised to have a mean of zero and a standard deviation of one. The necessary transformations are computed twice, once on each of the two halves of the dataset. When training each half is pre-processed using the transformation that was fitted to the half of the dataset. During application, when each discriminator is applied to the opposite halves of the data, the features are pre-processed using the opposite transformation as well, in order to ensure optimal compatibility.

The transformations are computed whilst invalid or undefined feature values are still `NaN`, in a way that ignores `NaN` values. Following the application of the transformations, any `NaN` values are replaced with zeros.

10.8.5 Weighted training

In Chap. 8 and Chap. 9 (including the HL-LHC study, Sec. 9.9) the data were weighted during training in order to balance the contributions from signal and background. The relative weighting of backgrounds were preserved during reweighting, meaning that the most important backgrounds had a higher priority than those of lesser importance, but the final classifier would identify signal and background with equal performance.

In this expanded analysis with multiple channels and jet-categories, I eventually find it to be more optimal to weight the sum of signal and sum of background to be equal in each jet category per channel. Looking at Tab. 10.5, for example, we can see that the categories with the higher sensitivity are also the ones with the lower yields. By simply reweighting by class without regard for category, the discriminators are implicitly encouraged to learn to classify events better in the categories with lower sensitivity to signal. By instead balancing within categories, the discriminators' responses to the more sensitive categories improve. The impact of different weighting considerations was examined in an initial

study on the 2016 samples, and it was found that balancing the weight sums of signal and background in each channel and jet category, provided a set of discriminators that resulted in a more sensitive analysis than was possible with similar discriminators which had been trained on samples in which the signal and background had only been balanced globally, without regard to subcategories.

10.9 Non-resonant search

10.9.1 Input features

The full menu of input features is described in Sec. 10.6.3, and after applying the feature selection process described in Sec. 10.7, the input features to the non-resonant classifiers are:

1. DeepFlavour *charm versus bottom* score of the b_1 binned in working points;
2. DeepFlavour *charm versus light-jet* score of the b_1 binned in working points;
3. HHBTAG score for b_2 ;
4. $m_{hh, \text{KinFit}}$;
5. χ^2 of HHKinFit;
6. m_{T, τ_1} ;
7. $m_{h_{\tau^+ \tau^-}, \text{SV}}$;
8. $m_{h_b \bar{b}}$;
9. $m_{hh, \text{SV}}$;
10. $p_{T, h_b \bar{b}}$;
11. p_{T, τ_1} ;
12. p_{T, τ_2} ;
13. p_{T, b_1} ;
14. $\Delta R(\tau_1, \tau_2)$;
15. $\Delta R(\tau_1, \tau_2) \times p_{T, h_{\tau^+ \tau^-}, \text{SV}}$;
16. $\Delta R(b_1, b_2)$ in the rest-frame of $h_{\tau^+ \tau^-}$,MPT;

17. $|\Delta\phi(h_{\tau^+\tau^-}, \text{sv}, \vec{p}_T^{\text{miss}})|$;
18. $|\Delta\phi(h_{\tau^+\tau^-}, \text{sv}, h_{b\bar{b}})|$;
19. $\Phi(\tau_1, \tau_2, b_1, b_2, h, h)$ (defined in Sec. 10.6.3);
20. $\cos(\tau_2, \vec{p}_T^{\text{miss}})$ (defined in Sec. 10.6.3).

These are then combined with six categorical input features, which either provide discrimination alone (e.g. b -tag working point) or can allow the DNN to parameterise its response to subsets of the data (e.g. year and channel):

21. Boosted-category flag;
22. VBF-category flag
23. b_1 b -tag working point;
24. b_2 b -tag working point;
25. Event channel;
26. Event year.

10.9.2 Interpretation

Computing the feature importance for the trained discriminators (Fig. 10.12), we can see that the models are not overly reliant on any one feature, but have a distributed reliance over many features. The top features seem to be the ΔR angle between the tau candidates, and whether or not the event is boosted (indicating that the event topology is reflected in the feature values and the models learn to parameterise their responses to this flag). The next most important features relate to the Higgs and di-higgs masses; with only very loose cuts on the masses, these were expected to be quite discriminating features.

It is also interesting to check the embedding learnt for the channel flag (Fig. 10.13). The similarity in embedding values for the $\mu\tau_h$ and $e\tau_h$ channels indicates that the models focus more on whether the event is fully-hadronic or semi-leptonic, and less on what the exact light lepton is in the semi-leptonic channel.

10.9.3 Binned predictions

Event predictions are binned per category by considering the uncertainty on the yield of signal and background in each bin due to the number of MC events. The maximum permitted uncertainty ranges between 3-5% according to the channel and jet-category

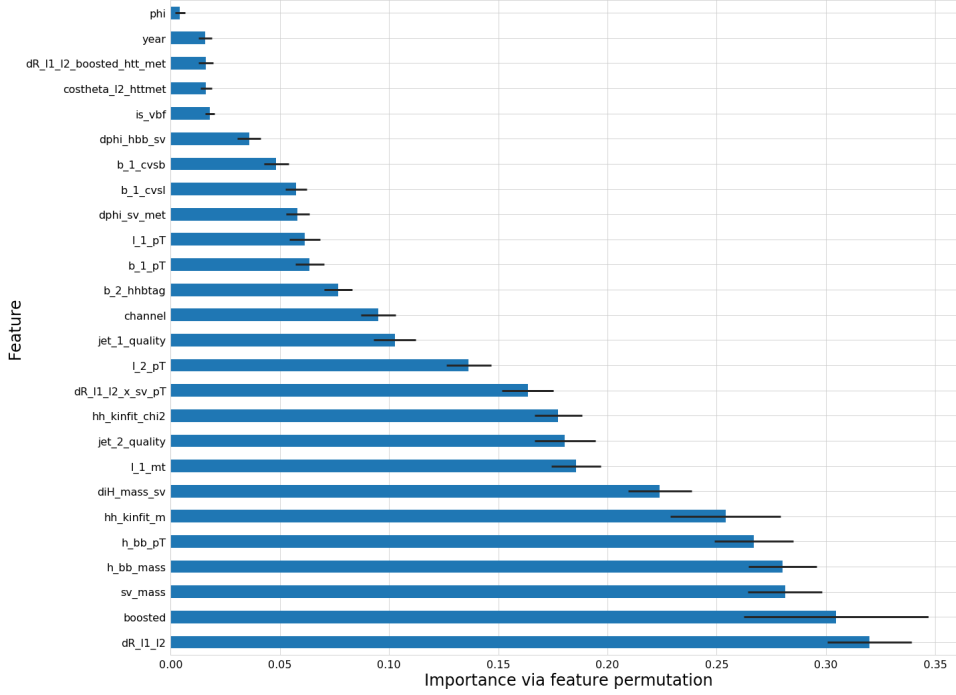


Figure 10.12: Importance of each feature used as input to one of the non-resonant ensembles, as computed using weighted binary cross-entropy.

for 2017 and 2018. For 2016, a constant maximum uncertainty of 6% is used. Examples of the binned predictions by decay channel for the two resolved b -jets category are shown in Fig. 10.14 for 2018. Computing the AMS summed in quadrature over all bins and categories and assuming a systematic uncertainty of 10% and considering the yield uncertainty due to the number of MC events, results in an expected significance of 0.593.

10.9.4 Inference and interpretation

At the time of writing of this thesis, all results are being finalised; they include both the ggF and VBF signals, and all systematic uncertainties on yield and shape, except for: uncertainties on b -tag scale factors, uncertainty on the anti-jet discriminator in the $\tau_h \tau_h$ channel, and the jet energy-scale uncertainty per year (a single uncertainty is instead used for all three years).

For the SM point ($\kappa_\lambda = 1$), an expected upper-limit at the 95% confidence level on the cross-section of SM non-resonant di-Higgs production divided by the SM prediction of $4.5\sigma/\sigma_{\text{SM}}$ is achieved. This is shown in Fig. 10.15 broken down into the three years, and the overall combination. Limits as a function of κ_λ can also be derived in terms of channels and jet-categories, as illustrated in Fig. 10.16. Figure 10.17 shows the expected

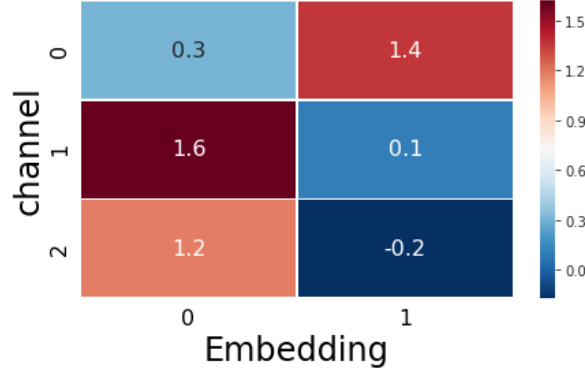


Figure 10.13: Learned embeddings for the channel flag. $0 = \tau_h \tau_h$, $1 = \mu \tau_h$, and $2 = e \tau_h$. Values in the row corresponding to the event channel are fed to the model.

exclusion of di-Higgs production as a function of κ_λ . Figure 10.18 shows the sensitivity to κ_λ assuming SM di-Higgs production at $\kappa_\lambda = 1$; at the 95% confidence level, κ_λ is constrained to lie within the range $[-3.5, 10]$, and $[-1.5, 8]$ at the 68% confidence level. Figure 10.19 illustrates the impact and pulls of the top-30 systematic uncertainties, as ranked by decreasing impact. From this we can see that the top systematic uncertainties are due to theory uncertainties on signal production, tau-ID scale-factors, and jet energy-scale uncertainties (`prop_binch*_bin*` entries refer to statistical uncertainties due to MC populations in each bin).

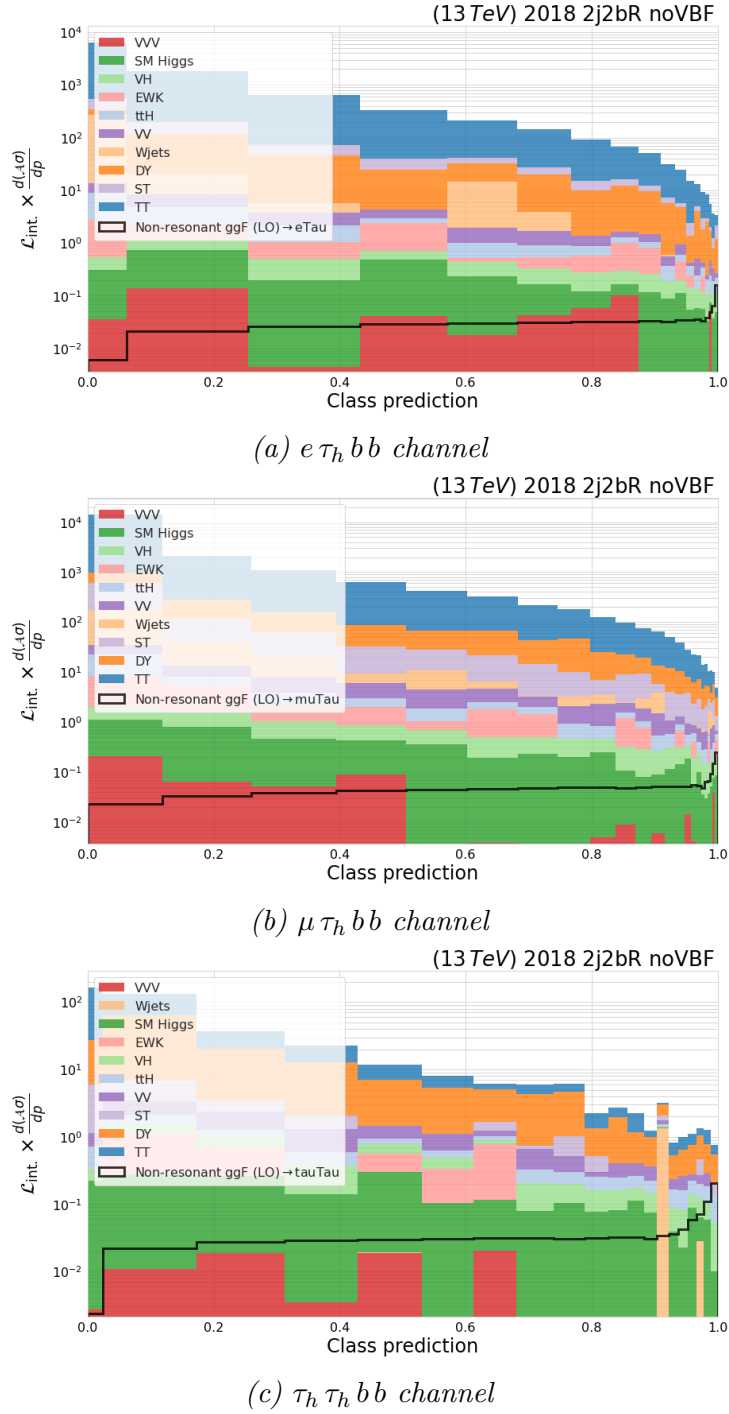


Figure 10.14: Event-yield distributions for non-resonant signal and SM backgrounds as a function of the discriminator output. The most significant jet-category in 2018 (two resolved b-jets) for each decay channel is shown.

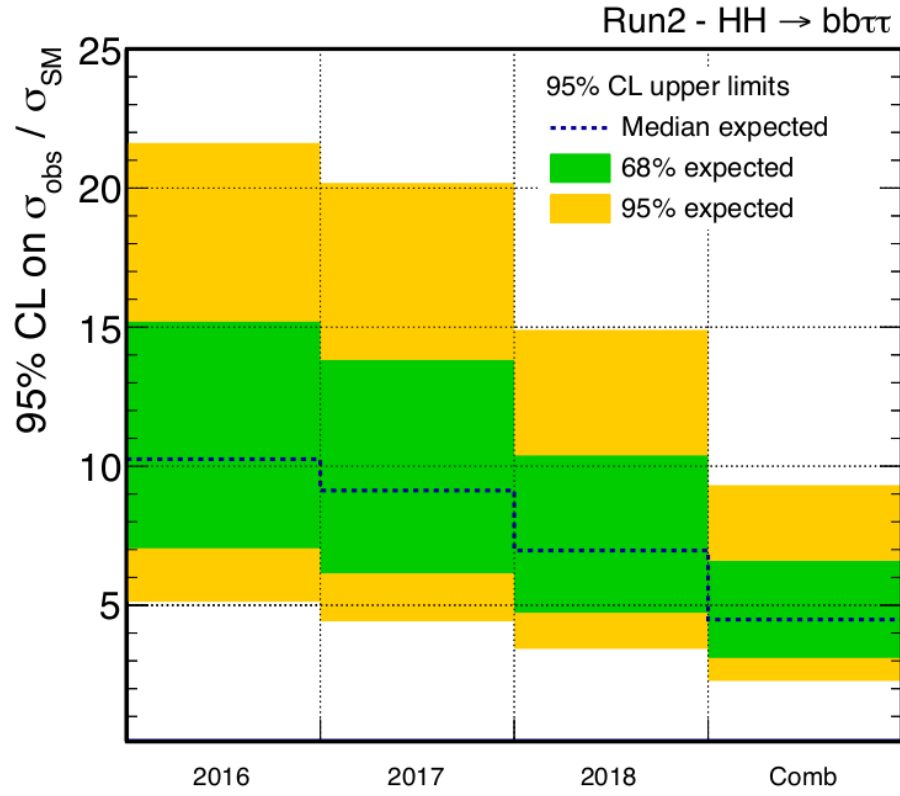
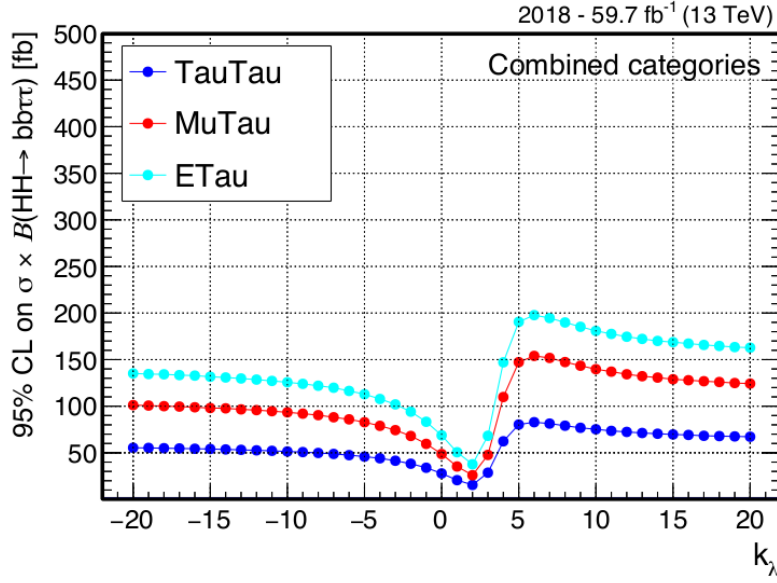
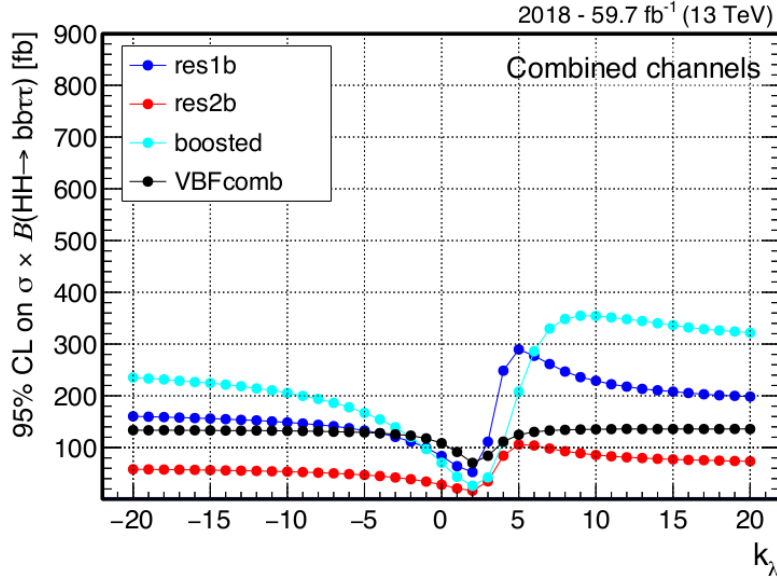


Figure 10.15: Expected upper-limits at the 95 % confidence level on the cross-section of SM non-resonant di-Higgs production divided by the SM prediction. The combination of all three years provides a median expected upper limit of $4.5\sigma/\sigma_{SM}$.



(a) Limits for 2018 in terms of channels.



(b) Limits for 2018 in terms of jet-categories.

Figure 10.16: Examples of the expected upper-limits at the 95 % confidence level on the cross-section of non-resonant $hh \rightarrow \tau^+ \tau^- b\bar{b}$ as a function of κ_λ .

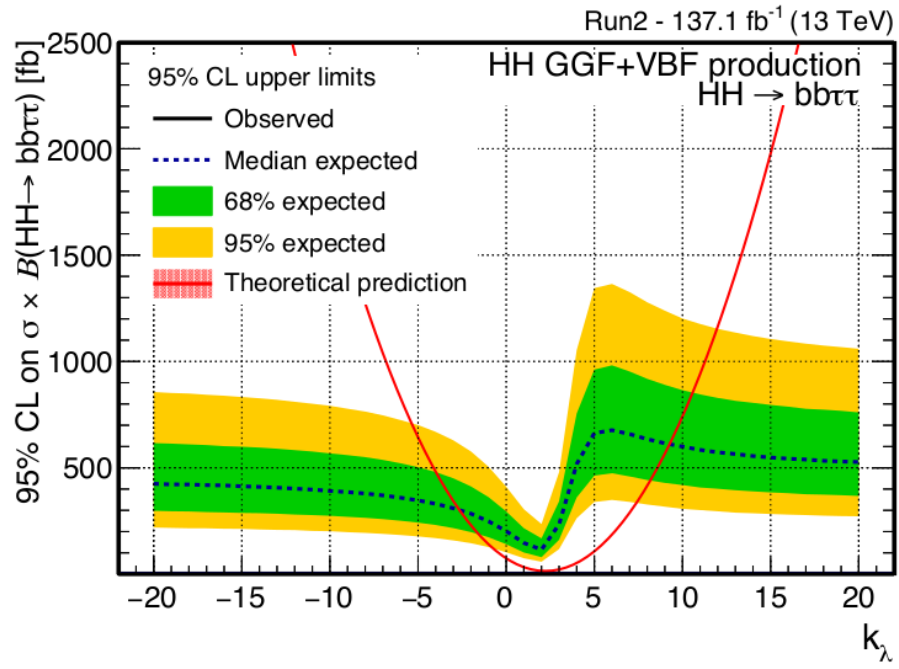


Figure 10.17: Expected upper-limits at the 95 % confidence level on the production cross-section of SM non-resonant di-Higgs in the $\tau^+ \tau^- b \bar{b}$ decay channel as a function of κ_λ .

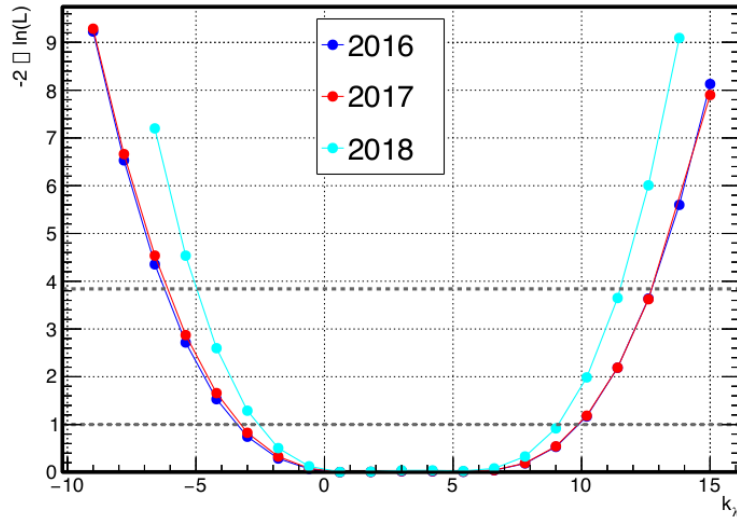
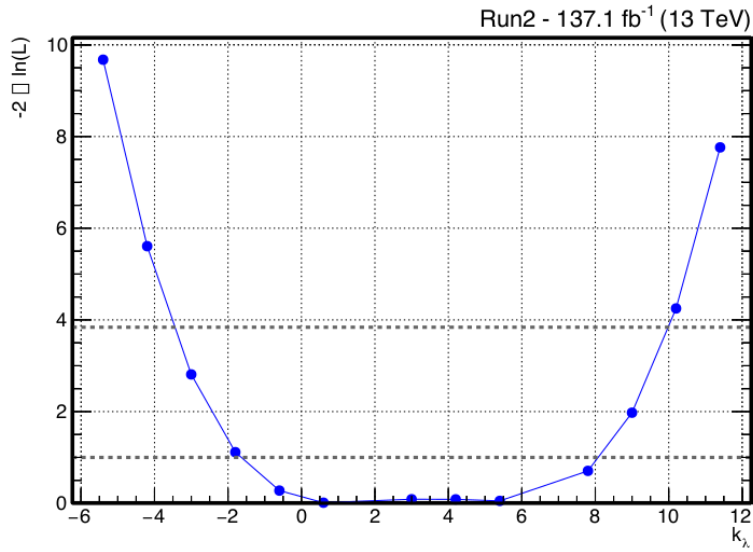
(a) Full scan of κ_λ , split by years.(b) Combined scan of κ_λ .

Figure 10.18: Expected log-likelihood scan of κ_λ assuming SM di-Higgs production at $\kappa_\lambda = 1$.

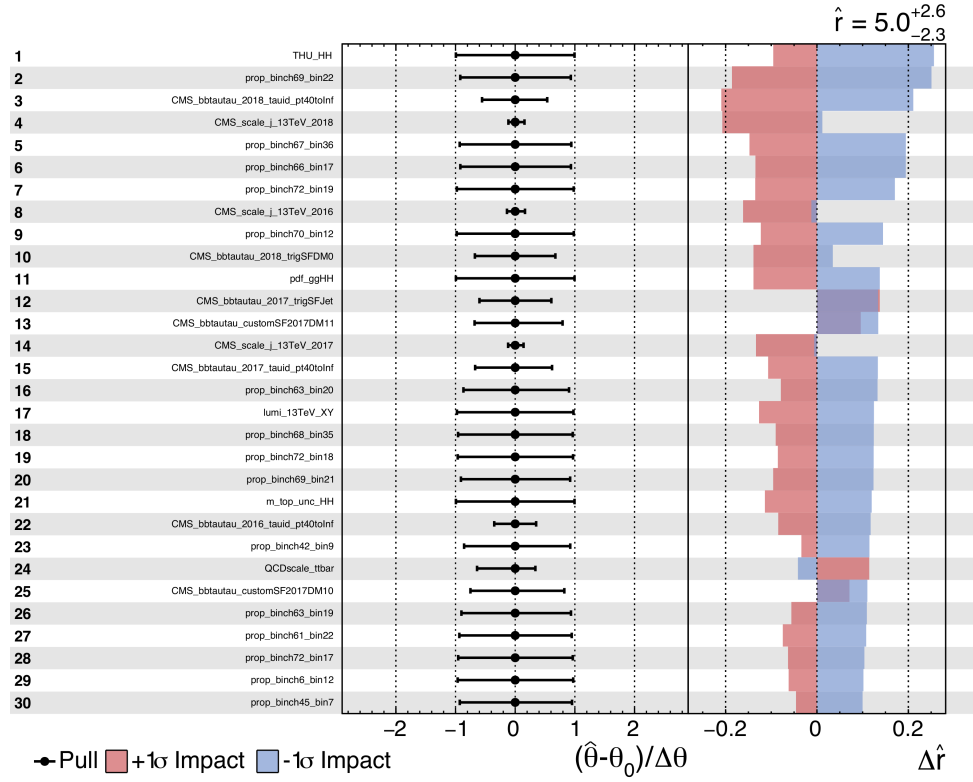


Figure 10.19: Pulls and impacts of the top-30 systematic uncertainties, as ranked by decreasing impact.

10.10 Resonant search

10.10.1 Parameterised model

Input features

Following the feature selection process described in Sec. 10.7, the input features (as defined in Sec. 10.6.3) the parameterised resonant-classifiers are:

1. The true resonant mass (correct for signal and randomly sampled for background);
2. DeepFlavour *charm versus bottom* score of the b_1 binned in working points;
3. DeepFlavour *charm versus bottom* score of the b_2 binned in working points;
4. DeepFlavour *charm versus light-jet* score of the b_1 binned in working points;
5. HHBTAG score for b_2 ;
6. χ^2 of `HHKinFit`;
7. $m_{hh, \text{KinFit}}$;
8. $m_{hh, \text{MPT}}$;
9. $m_{h_{\tau^+ \tau^-}, \text{SV}}$;
10. $m_{h_{b\bar{b}}}$;
11. m_{T, τ_1} ;
12. $m_{T, h_{\tau^+ \tau^-}, \text{SV}}$;
13. p_{T, τ_2} ;
14. $\Delta R(\tau_1, \tau_2) \times p_{T, h_{\tau^+ \tau^-}, \text{SV}}$;
15. $\Delta R(\tau_1, \tau_2)$;
16. $|\Delta\eta(h_{\tau^+ \tau^-}, \text{SV}, h_{b\bar{b}})|$.

These are then combined with seven categorical input features, which either provide discrimination alone (e.g. b -tag working point) or can allow the DNN to parameterise its response to subsets of the data (e.g. year and channel)

17. The true spin of the resonant particle (correct for signal and randomly sampled for background);

- 18. Boosted-category flag;
- 19. VBF-category flag
- 20. b_1 b -tag working point;
- 21. b_2 b -tag working point;
- 22. Event channel;
- 23. Event year.

Interpretation

Computing the feature importance for the trained discriminators (Fig. 10.20), we can see that in contrast to the non-resonant case, the resonant models are extremely reliant on just a few features (mainly the true resonant mass and di-Higgs mass as computed by `HHKinFit`). This is to be expected, since the models are required to parameterise their response to a large range of possible signals. The other features, however, are still important in terms of absolute values, and are simply overshadowed by the two features most sensitive to the signal.

We can also check whether the models really parameterise their responses to the resonant mass. From Fig. 10.21, we see that the models predict background data to be background-like with little to no dependence on the resonant mass, as desired. There is a slight bump towards signal-likeness at the low-mass region, however in this region the reconstructed resonant mass for signal and background display a significant overlap, and signal/background identification is more difficult.

Looking, now, at signal-only data we can see in Fig. 10.22 that the models do correctly parameterise their response to the signal data; e.g. in Fig. 10.22a the model predictions peak at 300 GeV (the true mass of the heavy Higgs boson) but once the resonant mass input feature (`res_mass`) is adjusted away from the true value, the predictions drop towards background-like. Due to the fewer signal hypotheses at the high-mass region, the response of the model for high-masses is comparatively widened, although there is still a definite parameterisation there, as shown by Fig. 10.22c.

Binned predictions

Examples of the binned predictions by decay channel and jet category are shown in Fig. 10.23 for 2018. From these we can see that the model is able to identify both background and the signal events for the specific hypothesis (a 400 GeV radion). Note that the signal has an arbitrary normalisation.

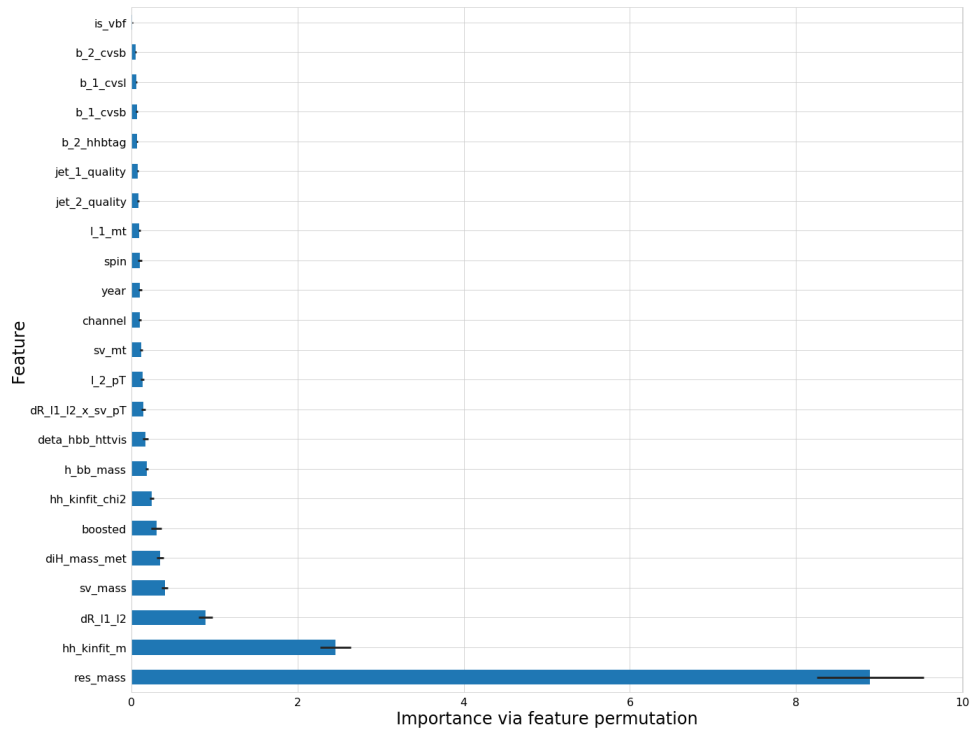


Figure 10.20: Importance of each feature used as input to one of the parameterised resonant ensembles, as computed using weighted binary cross-entropy.

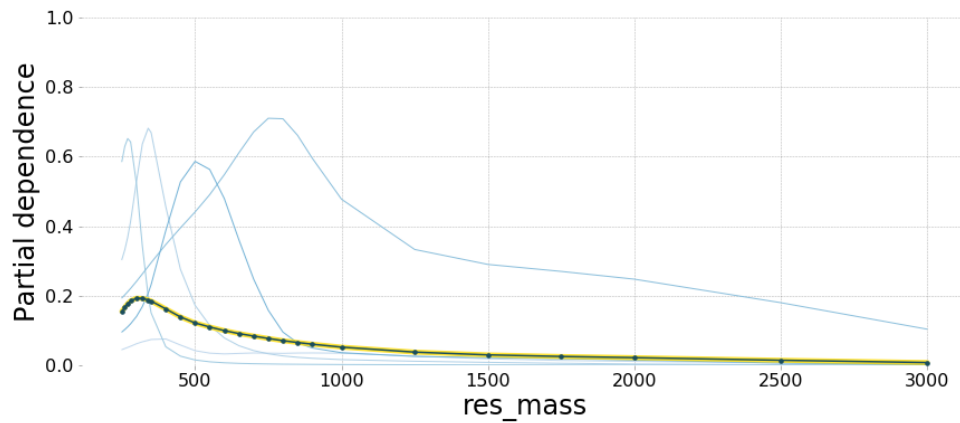
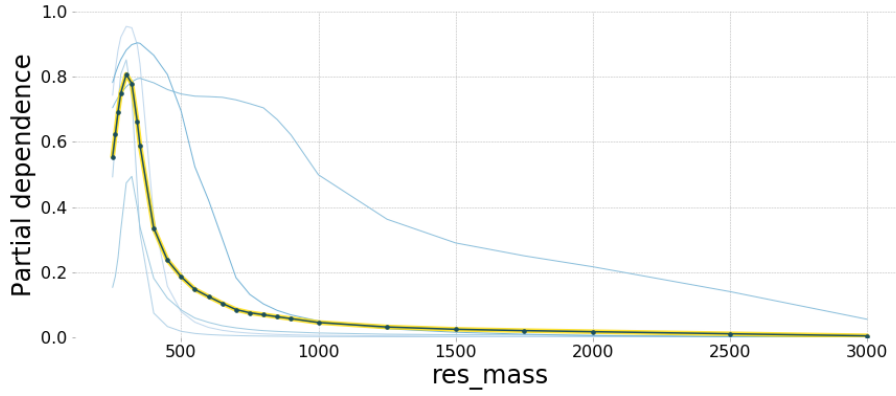
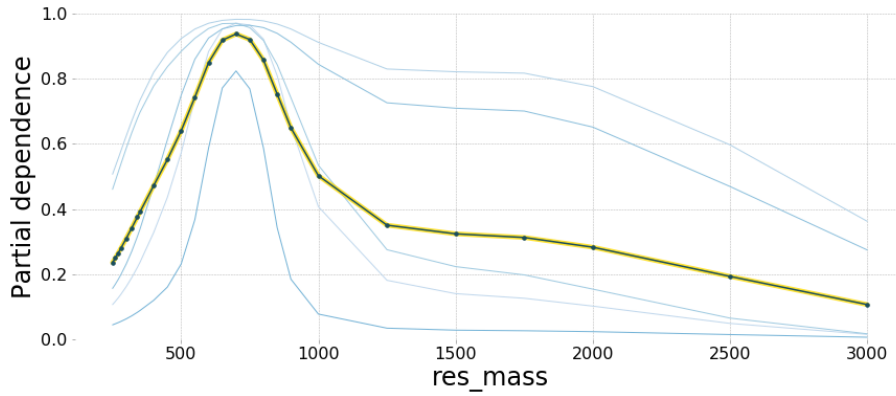


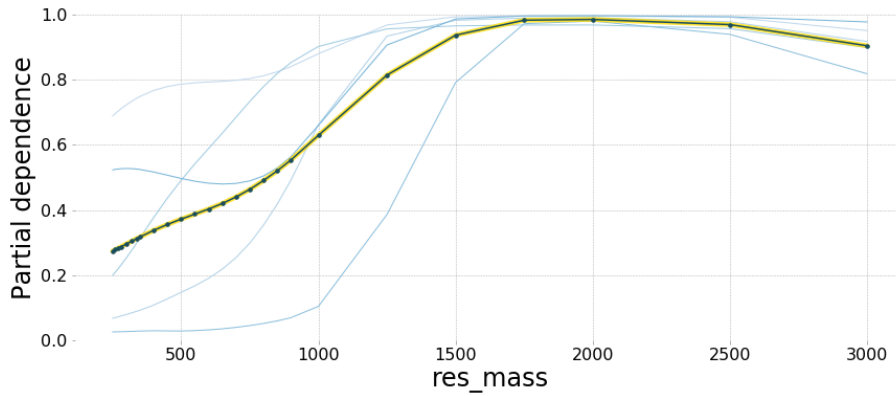
Figure 10.21: Model predictions on background-only data in which the resonant mass input feature (**res_mass**) is adjusted to a range of different values. Model predictions are made on all of the datapoints adjusted to each value of the resonant mass input feature, and the yellow line indicates the median prediction at each value. The blue lines represent the behaviour of subsets of the data, and are obtained by combining the individual datapoint responses into five clusters.



(a) 300 GeV signal mass.



(b) 700 GeV signal mass.



(c) 2000 GeV signal mass.

Figure 10.22: Model predictions on signal-only data for specific resonant masses in which the resonant mass input feature (`res_mass`) is adjusted to a range of different values. Model predictions are made on all of the datapoints adjusted to each value of the resonant mass input feature, and the yellow line indicates the median prediction at each value. The blue lines represent the behaviour of subsets of the data, and are obtained by combining the individual datapoint responses into five clusters.

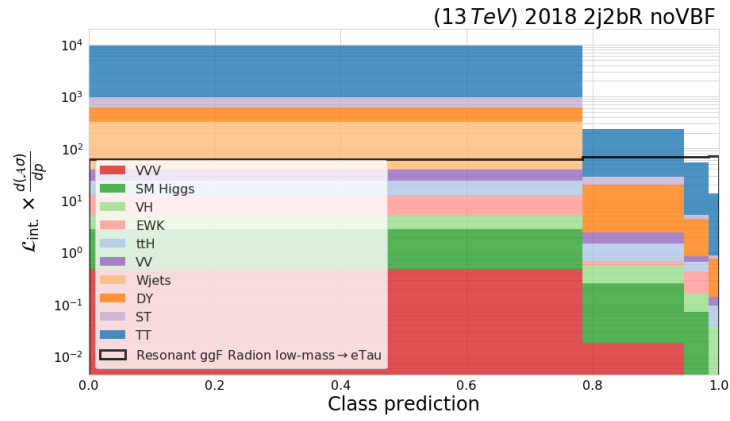
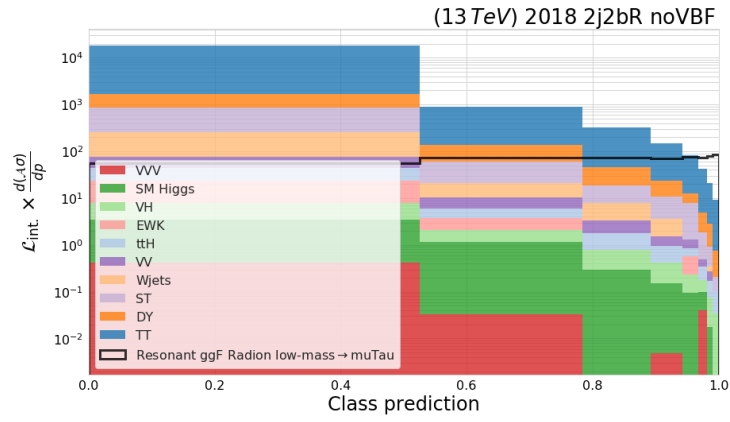
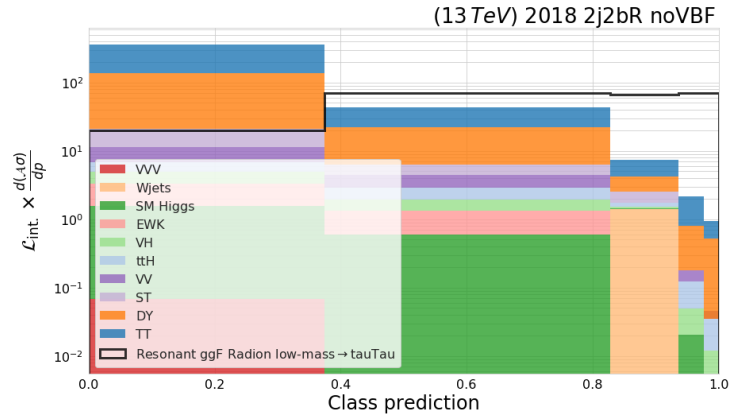
(a) $e\tau_h bb$ channel(b) $\mu\tau_h bb$ channel(c) $\tau_h\tau_h bb$ channel

Figure 10.23: Event-yield distributions as a function of the parameterised discriminator output for a 400 GeV radion signal. The two resolved b -jets jet-category in 2018 is shown for each decay channel. The signal has an arbitrary normalisation.

10.10.2 Non-parameterised model

Input features

Following the feature selection process described in Sec. 10.7 and additional removal of features correlated with the di-Higgs mass, the input features (as defined in Sec. 10.6.3) to the non-parameterised resonant-classifiers are:

1. DeepFlavour *charm versus bottom* score of the b_1 binned in working points;
2. DeepFlavour *charm versus bottom* score of the b_2 binned in working points;
3. DeepFlavour *charm versus light-jet* score of the b_1 binned in working points;
4. HHBTAG score for b_1 ;
5. χ^2 of HHKinFit;
6. $m_{h_{\tau^+\tau^-},\text{SV}}$;
7. $m_{h_b\bar{b}}$;
8. $m_{T,h_{\tau^+\tau^-},\text{SV}}$;
9. $m_{T,\tau_1\tau_2}$ (the transverse mass of τ_1 computed with respect to τ_2);
10. m_{T,τ_1} ;
11. $m_{\text{top},1}$;
12. $\vec{p}_T^{\text{miss}}$;
13. p_{T,τ_2} ;
14. $p_{T,hh}$;
15. $\Delta R(b_1, b_2)$;
16. $E_{h_{\tau^+\tau^-},\text{SV}}$;
17. $\Delta R(\tau_1, \tau_2) \times p_{T,h_{\tau^+\tau^-},\text{SV}}$;
18. $\Delta\phi(h_b\bar{b}, \text{MPT}) \times p_{T,h_{\tau^+\tau^-},\text{SV}}$;
19. $|\Delta\eta(\tau_1, \tau_2)|$.

These are then combined with six categorical input features, which either provide discrimination alone (e.g. b -tag working point) or can allow the DNN to parameterise its response to subsets of the data (e.g. year and channel):

20. Boosted-category flag;
21. VBF-category flag
22. b_1 b -tag working point;
23. b_2 b -tag working point;
24. Event channel;
25. Event year.

Interpretation

Computing the feature importance for the trained discriminators (Fig. 10.24), we can see that once again the models are displaying a distributed reliance on the input features. Similar to the non-resonant classifier, the ensembles here rely on the Higgs masses, the boosted flag and the angle between final-states (in this the b -jets rather than the taus).

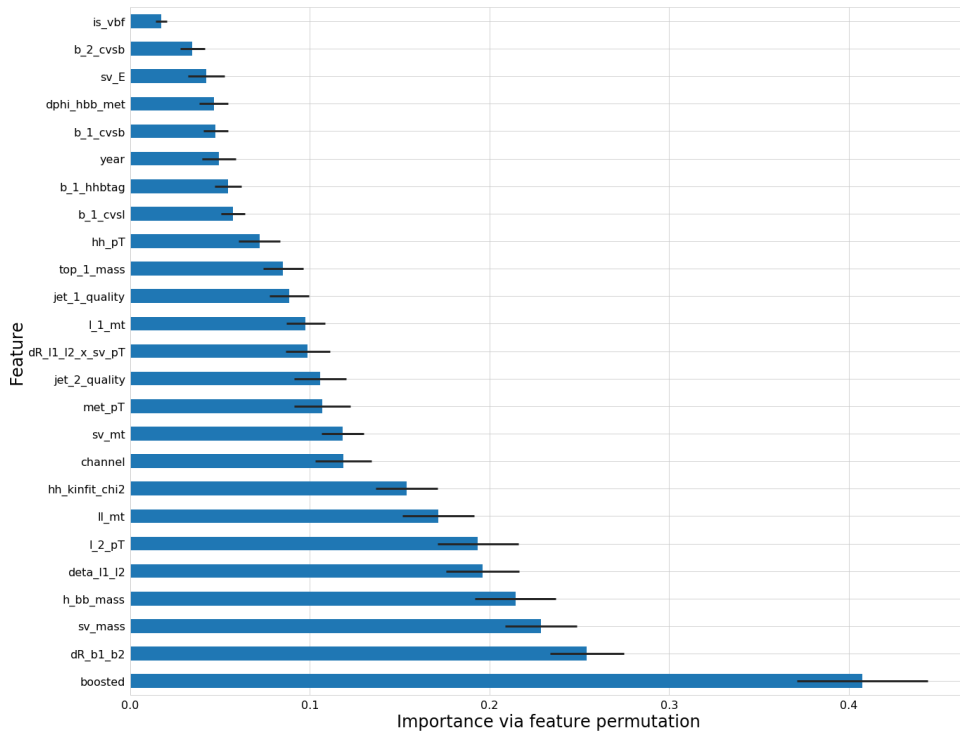


Figure 10.24: Importance of each feature used as input to one of the non-parameterised resonant ensembles, as computed using weighted binary cross-entropy.

Predictions

Examples of the binned predictions by decay channel and jet category are shown in Fig. 10.25 for 2018. In comparison to the parameterised approach, the discriminators here struggle to identify signal events, leading to an almost flat distribution. Background, however, is well identified, and clustered towards zero.

We can exploit the good identification of the background of the classifier, and the fact that the input features were deliberately filtered to remove features related to the di-Higgs mass, by checking the two-dimensional distribution of the data in terms of class prediction and the `HHKinFit` mass. Figure 10.26 provides a demonstration of this for a few masses in the two resolved- b -jets jet-category in 2018. Note that for visualisation purposes, the logit of the class predictions (network output before the sigmoid activation function) is plotted to allow the KDEs to be plotted on an unbounded variable, rather than the class prediction, which is bounded between zero and one. Although the high-mass resonances show good separation from the background, it is worth noting that for clarity the lowest density-contours of the samples are not plotted and the high-mass resonances do overlap with background contributions.

10.10.3 Discussion

Due to the timescale of the CMS analysis associated with this work, no limits on resonant production of di-Higgs are currently extracted. Nonetheless, I have demonstrated two approaches for applying DNN-based classifiers to the study of such multi-hypothesis signals. The parameterised classifier is able to adjust its response to varying masses, and shows good identification of both signal and background for a range of signal hypotheses. It does, however, require evaluating the signal and background, plus all the systematic shape-variations, once per signal hypothesis. In contrast, the non-parameterised classifier struggles to identify signal, but does manage to identify the background. However, the deliberate decorrelation of its response with the di-Higgs mass allows one to use a 2D fit with the `HHKinFit` mass to recover signal separation. The main advantage being that all of the data, and systematic shape-variations, only need to be evaluated once.

From Fig. 10.26 we can see that the `HHKinFit` mass provides naturally good separation above masses of around 600 GeV, and so a hybrid approach might be to only apply the parameterised classifier to signal hypotheses below this mass, where background contamination is higher. This would reduce the number of times that the classifier would have to be evaluated and help recover sensitivity to low-mass resonances.

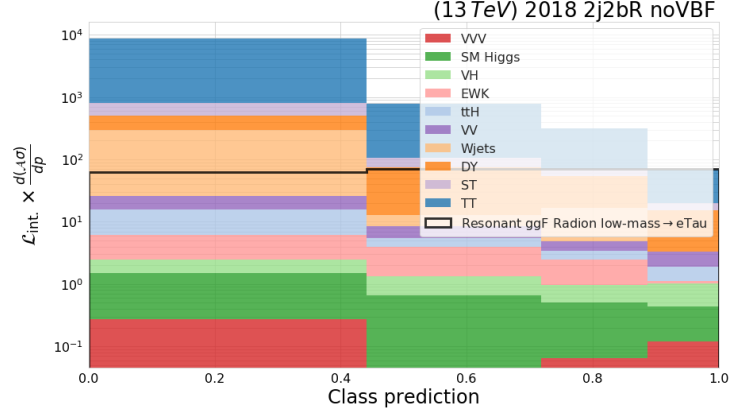
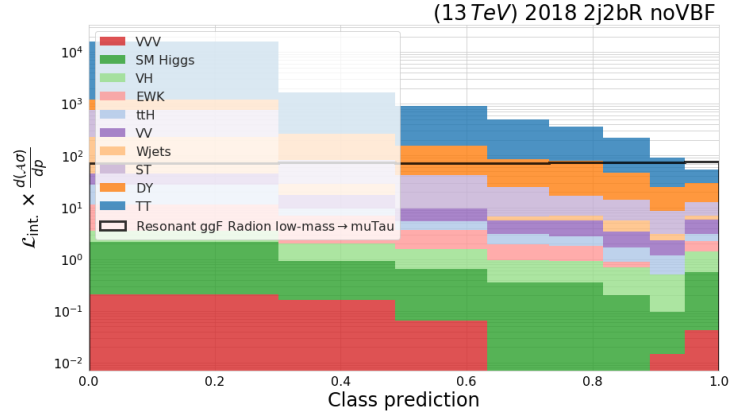
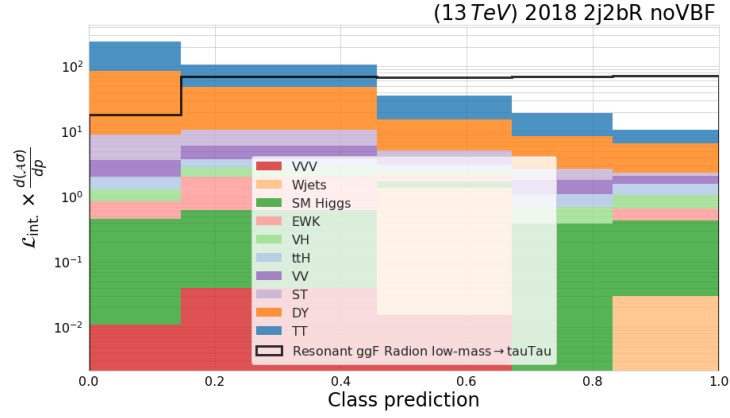
(a) $e\tau_h bb$ channel(b) $\mu\tau_h bb$ channel(c) $\tau_h\tau_h bb$ channel

Figure 10.25: Event-yield distributions as a function of the non-parameterised discriminator output for a 400 GeV radion signal. The two resolved b -jets jet-category in 2018 is shown for each decay channel. The signal has an arbitrary normalisation.

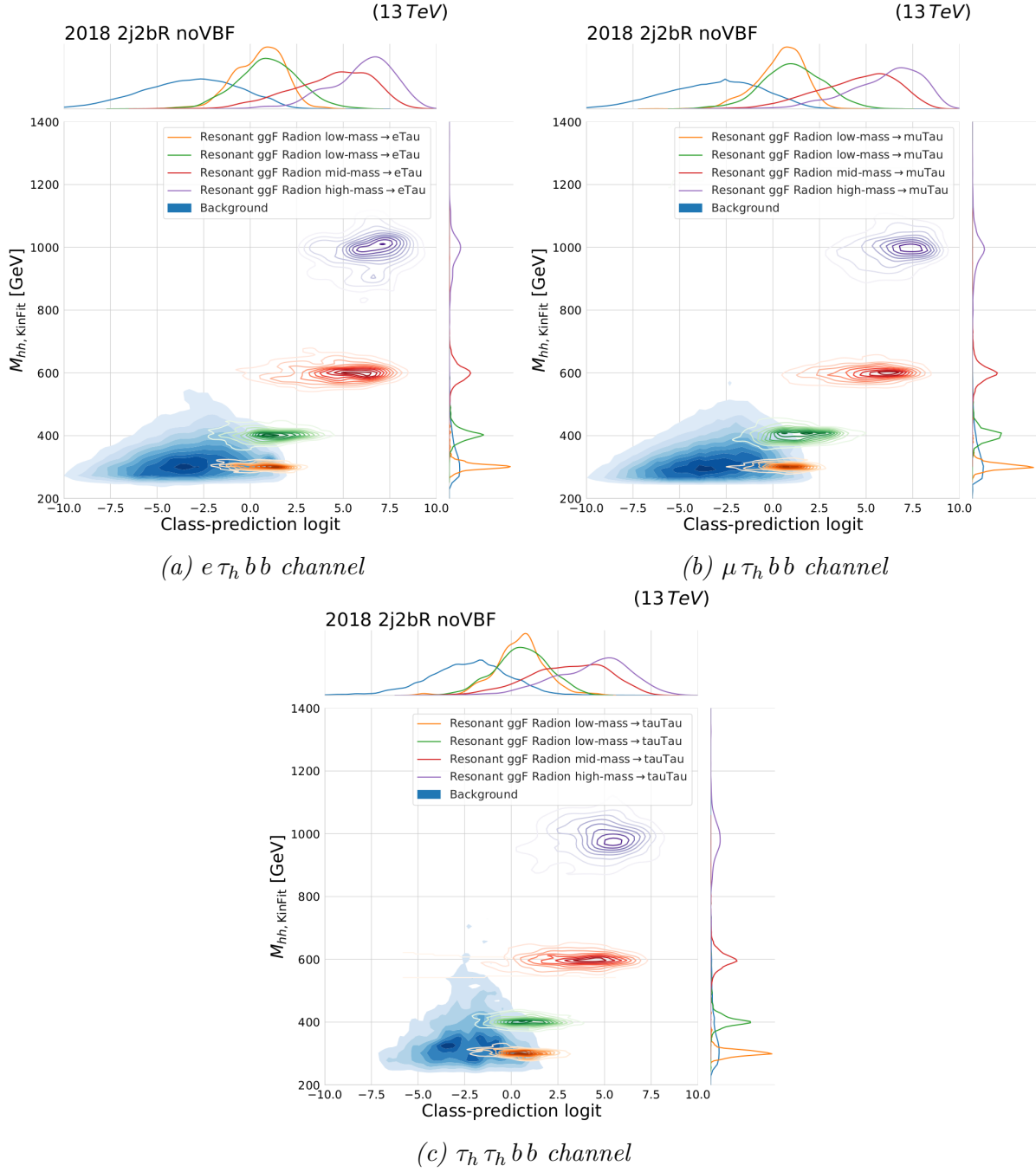


Figure 10.26: Unity-normalised shape-distributions as a function of the logit of the non-parameterised discriminator output and the $HH\text{KinFit}$ mass, for a range of radion signal masses (300, 400, 600, and 1000 GeV). The distributions of the two resolved b -jets jet-category in 2018 are shown for the three decay channels.

10.11 Conclusion

Whilst results have yet to be finalised, the search for non-resonant $hh \rightarrow \tau^+ \tau^- b\bar{b}$ indicates large improvements thanks to the addition of two years of new data, and the use of more advanced tools and algorithms, and the inclusion of the VBF production mechanism. The previously published result (Refs. [213] and [214]) for the 2016 CMS dataset (integrated luminosity of 35.9 fb^{-1}) showed an expected 95 %-confidence upper-limit on the SM non-resonant di-Higgs production cross-section of 25 times the SM prediction. Reference [216] suggested that through a more complete MVA treatment of all channels, this limit could be further improved towards 15.5 times the SM prediction. Meanwhile, ATLAS had also adopted a more open selection, without the use of mass-window cuts around the Higgs masses, and achieved an expected upper limit of 14.8 times the SM [217] (integrated luminosity of 36.1 fb^{-1}).

This analysis revisited the 2016 CMS dataset and achieved an upper limit of 10.3 times the SM prediction. This was driven by the use of a more open event-selection, a different method of feature selection, optimised weighting of the training data, the inclusion of the VBF jet-category, and advanced DNN classifiers for signal/background prediction, b -jet assignment, and b and τ_h tagging. Extending the analysis to the 2017 and 2018 datasets (integrated luminosity of 137.19 fb^{-1}), the preliminary results indicate an upper limit of 4.5 times the SM prediction. Assuming SM production at $\kappa_\lambda = 1$, κ_λ may be treated as a parameter of interest and a log-likelihood scan can be performed. At 95 % confidence, κ_λ is constrained to lie within $[-3.5, 10]$.

Unfortunately, limits on resonant production are not yet available in this analysis, however two possible training and inference methodologies were presented: one in which the classifier parameterises its response according the signal hypothesis being tested, and a second in which the classifier is trained to allow optimal pairing with the reconstructed di-Higgs mass in a 2D fit for inference.

In this case, the non-resonant classifiers were trained only for the SM signal ($\kappa_\lambda = 1$), which means that their responses for $\kappa_\lambda \neq 1$ signals may not be optimal. The concept of parameterised learning could easily be extended to the non-resonant search, and indeed this was done in Ref [216], however one would then suffer from increased inference time. Additionally, the likelihood scan of κ_λ would no longer be possible, since (currently) constructing it assumes that only the weights of the data change, and not their associated predictions.

The value of κ_λ affects mainly kinematic properties associated with the di-Higgs mass [203], and when designing the classifiers for this analysis, I focussed mainly on optimising the sensitivity at the SM point. In order to provide improved sensitivity to the value of κ_λ , an alternate training method would be to deliberately filter the input features to reduce the dependence on the di-Higgs mass (and hence the value of κ_λ), similar to what I do for the one of the methods of training the resonant classifier. A test in which the feature selection and the classifier training were performed on $\kappa_\lambda = 5$ signal data showed

improved sensitivity for medium to high absolute values of κ_λ , but reduced sensitivity at the SM point. For $\kappa_\lambda = 5$ the di-Higgs mass-distribution displays a large overlap with the background, and the feature selection naturally ignored it, and features associated with it, so the classifier displayed a weaker dependence on κ_λ . In a more convoluted publication, one could provide two measurements: a limit on the SM point using a classifier specialised for κ_λ , and a κ_λ scan using a second classifier with reduced dependence on κ_λ .

Chapter 11

Summary and conclusions

This thesis documents the majority of my research work between 2016 and 2020 in which I studied the various applications of neural networks to the search for di-Higgs production at the LHC. It is a fairly long document, so my thanks (and commiserations) to anyone tasked with reading through the whole thing, and my congratulations (and concern) to any who have done so voluntarily. I hope, though, that individual sections and chapters will be of interest and use to a range of people, and so for the readers coming here straight from the abstract, I will do my best to summarise the outcomes of each part below.

In Chapters 7 and 8 I performed a preliminary investigation in which I found that DNNs were applicable to the task of event-level classification of non-resonant di-Higgs production at the LHC, and that the subsequent predictions of such classifiers were useable as summary statistics for inferring the presence of signal in a hypothesis test. I also found that the use of DNN-based regression could be a means by which to optimise the input features to the classifiers, thereby further improving the sensitivity to signal. However, technical requirements meant that the information necessary to train such regressors is not always present in the data formats used in collaboratory investigations, and so must be a key concern near to the start of an analysis.

In Chap. 9 I optimised the training and architecture of my DNNs by quantitatively testing the benefits of a range of recently published techniques. This testing was done using a well-studied HEP benchmark dataset (the 2014 Higgs ML Kaggle Challenge). Through the use of domain-specific data augmentation at train and test time, densely connected layers, a 1cycle training schedule, and a newer activation function I was able to match the performance of the winning solution with a light-weight model which could be trained seven times quicker on a laptop CPU (13 times quicker on GPU), and also be applied significantly quicker, too. These techniques therefore allow one to rapidly train and compare models and quickly reach good performance in HEP tasks without extended study. They also broaden the possibilities for developing models by researchers who do not have access to expensive hardware, such as GPUs.

Concurrently to the study in Chap. 9, I developed a software package called LUMIN.

This acts as a framework to allow users to quickly access the latest techniques in machine learning and deep learning, and develop and train models in a modular fashion. Although the framework is ostensibly designed for HEP data, it can indeed be applied to many other domains. LUMIN was released as an open-source software package in early 2019, and is still under active development by myself. Through various talks and project activities it is gradually attracting the attention of the scientific community, and recently began receiving contributions from other developers.

The techniques studied in Chap. 9 were also re-verified in a projection study for the sensitivity of the HL-LHC to di-Higgs, in which they provided a 30 % increase in sensitivity over a more basic DNN. This study indicated that CMS would likely reach just under 3σ significance of SM di-Higgs production for 3000 fb^{-1} of integrated luminosity. A combination study with ATLAS, however, would no doubt increase the overall sensitivity. This study also examined the impact of the resolution of the proposed MIP Timing Detector at CMS, quantifying directly its effect on the expected discovery significance.

Having shown that DNNs are applicable to di-Higgs searches, and optimised their architectures and training, I contributed to the Run-II CMS search for di-Higgs production in the $\tau^+ \tau^- b \bar{b}$ decay channel. My contributions of event-level classifiers for resonant and non-resonant di-Higgs production via gluon fusion are documented in Chap. 10. Whilst the full analysis is still being finalised, preliminary results show that the DNN classifiers perform well: at the 95 % confidence level, an expected upper-limit on the cross-section of SM non-resonant di-Higgs production of 4.5 times the SM prediction, and the ratio of the Higgs self-coupling to the SM prediction is constrained to lie between $[-3.5, 10]$. Additionally in this chapter I document an approach to automated feature-selection for inputs to ML models, which has gradually been developed over the course of this thesis into something that provides a reliable and robust filtering of large menus of possible inputs.

Bibliography

- [1] **L3** Collaboration, *Search for the Standard Model Higgs boson in e^+e^- interactions at $161 \leq s \leq 172 \text{ GeV}$* , Physics Letters B **411** (Oct, 1997)373–386.
[https://doi.org/10.1016/S0370-2693\(97\)01033-2](https://doi.org/10.1016/S0370-2693(97)01033-2).
- [2] **D0** Collaboration, *Search for single top quark production at D0 using neural networks*, Physics Letters B **517** no. 3, (2001)282 – 294.
<http://www.sciencedirect.com/science/article/pii/S0370269301010097>.
- [3] P. Bhat, *Run II physics at the Fermilab Tevatron and advanced analysis methods*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **502** no. 2, (2003)327 – 333.
<http://www.sciencedirect.com/science/article/pii/S0168900203004388>.
Proceedings of the VIII International Workshop on Advanced Computing and Analysis Techniques in Physics Research.
- [4] F. Abe *et al.*, *The CDF detector: an overview*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **271** no. 3, (1988)387 – 403.
<http://www.sciencedirect.com/science/article/pii/0168900288902987>.
- [5] **D0** Collaboration, S. Abachi *et al.*, *The D0 Detector*, Nucl. Instrum. Meth. A **338** (1994)185–253.
- [6] **CMS** Collaboration, S. Chatrchyan *et al.*, *Measurement of the t -channel single top quark production cross section in pp collisions at $\sqrt{s} = 7 \text{ TeV}$* , Phys. Rev. Lett. **107** (2011) 091802, [arXiv:1106.3052](https://arxiv.org/abs/1106.3052) [hep-ex].
- [7] L. Breiman, *Arcing the edge*, tech. rep., 1997. <http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.367.9480&rep=rep1&type=pdf>.
- [8] Y. Freund and R. E. Schapire, *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting*, Journal of Computer and System Sciences **55** no. 1, (1997)119 – 139.
<http://www.sciencedirect.com/science/article/pii/S002200009791504X>.
- [9] J. H. Friedman, *Greedy function approximation: A gradient boosting machine.*, Ann. Statist. **29** no. 5, (10, 2001)1189–1232. <https://doi.org/10.1214/aos/1013203451>.

- [10] J. H. Friedman, *Stochastic gradient boosting*, Computational Statistics & Data Analysis **38** no. 4, (2002) 367 – 378.
<http://www.sciencedirect.com/science/article/pii/S0167947301000652>.
Nonlinear Methods and Data Mining.
- [11] L. Mason, J. Baxter, P. L. Bartlett, and M. R. Frean, *Boosting Algorithms as Gradient Descent*, in *Advances in Neural Information Processing Systems 12*, S. A. Solla, T. K. Leen, and K. Müller, eds., pp. 512–518. MIT Press, 2000. <http://papers.nips.cc/paper/1766-boosting-algorithms-as-gradient-descent.pdf>.
- [12] L. Mason, J. Baxter, P. Bartlett, and M. Frean, *Boosting Algorithms as Gradient Descent in Function Space*, 1999.
<http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=560C3EDBA817577256D6915FC7297EFD?doi=10.1.1.51.6893&rep=rep1&type=pdf>.
- [13] CMS Collaboration, *A search using multivariate techniques for a standard model Higgs boson decaying into two photons*, Tech. Rep. CMS-PAS-HIG-12-001, CERN, Geneva, 2012. <https://cds.cern.ch/record/1429931>.
- [14] ATLAS Collaboration, *Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC*, Physics Letters B **716** no. 1, (Sep, 2012) 1–29. <http://dx.doi.org/10.1016/j.physletb.2012.08.020>.
- [15] CMS Collaboration, *Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC*, Physics Letters B **716** no. 1, (Sep, 2012) 30–61.
<http://dx.doi.org/10.1016/j.physletb.2012.08.021>.
- [16] CMS Collaboration, S. Chatrchyan *et al.*, *Observation of a New Boson with Mass Near 125 GeV in pp Collisions at $\sqrt{s} = 7$ and 8 TeV*, JHEP **06** (2013) 081, arXiv:1303.4571 [hep-ex].
- [17] K. Albertsson *et al.*, *Machine Learning in High Energy Physics Community White Paper*, Jul, 2018. <https://cds.cern.ch/record/2630209>. Editors: Sergei Gleyzer, Paul Seyfert and Steven Schramm.
- [18] D. Guest, K. Cranmer, and D. Whiteson, *Deep Learning and its Application to LHC Physics*, Ann. Rev. Nucl. Part. Sci. **68** (2018) 161–181, arXiv:1806.11484 [hep-ex].
- [19] M. Andrews, M. Paulini, S. Gleyzer, and B. Poczós, *End-to-End Physics Event Classification with the CMS Open Data: Applying Image-based Deep Learning on Detector Data to Directly Classify Collision Events at the LHC*, arXiv:1807.11916 [hep-ex].
- [20] G. Louppe, K. Cho, C. Becot, and K. Cranmer, *QCD-Aware Recursive Neural Networks for Jet Physics*, JHEP **01** (2019) 057, arXiv:1702.00748 [hep-ph].

- [21] M. Paganini, L. de Oliveira, and B. Nachman, *CaloGAN : Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks*, Phys. Rev. **D97** no. 1, (2018) 014021, [arXiv:1712.10321 \[hep-ex\]](#).
- [22] P. Musella and F. Pandolfi, *Fast and Accurate Simulation of Particle Detectors Using Generative Adversarial Networks*, Comput. Softw. Big Sci. **2** no. 1, (2018) 8, [arXiv:1805.00850 \[hep-ex\]](#).
- [23] S. Otten *et al.*, *Event Generation and Statistical Sampling for Physics with Deep Generative Models and a Density Information Buffer*, [arXiv:1901.00875 \[hep-ph\]](#).
- [24] CMS Collaboration, *Heavy flavor identification at CMS with deep neural networks*, CMS-DP-2017-005 (Mar, 2017). <https://cds.cern.ch/record/2255736>.
- [25] A. Butter *et al.*, *The Machine Learning Landscape of Top Taggers*, SciPost Phys. **7** (2019) 014, [arXiv:1902.09914 \[hep-ph\]](#).
- [26] E. A. Moreno *et al.*, *JEDI-net: a jet identification algorithm based on interaction networks*, Eur. Phys. J. **C80** no. 1, (2020) 58, [arXiv:1908.05318 \[hep-ex\]](#).
- [27] LHCb Collaboration, R. Aaij *et al.*, *Design and performance of the LHCb trigger and full real-time reconstruction in Run 2 of the LHC*, JINST **14** no. 04, (2019) P04013, [arXiv:1812.10790 \[hep-ex\]](#).
- [28] J. Brehmer, G. Louppe, J. Pavez, and K. Cranmer, *Mining gold from implicit models to improve likelihood-free inference*, Proceedings of the National Academy of Sciences **117** no. 10, (Feb, 2020) 5242–5249, [arXiv:1805.12244 \[stat.ML\]](#).
<http://dx.doi.org/10.1073/pnas.1915980117>.
- [29] P. De Castro and T. Dorigo, *INFERNO: Inference-Aware Neural Optimisation*, Comput. Phys. Commun. **244** (2019) 170–179, [arXiv:1806.04743 \[stat.ML\]](#).
- [30] M. Stoye *et al.*, *Likelihood-free inference with an improved cross-entropy estimator*, [arXiv:1808.00973 \[stat.ML\]](#).
- [31] P. Baldi, K. Cranmer, T. Faucett, P. Sadowski, and D. Whiteson, *Parameterized neural networks for high-energy physics*, Eur. Phys. J. **C76** no. 5, (2016) 235, [arXiv:1601.07913 \[hep-ex\]](#).
- [32] A. Blance, M. Spannowsky, and P. Waite, *Adversarially-trained autoencoders for robust unsupervised new physics searches*, JHEP **10** (2019) 047, [arXiv:1905.10384 \[hep-ph\]](#).
- [33] G. Louppe, M. Kagan, and K. Cranmer, *Learning to Pivot with Adversarial Networks*, [arXiv:1611.01046 \[stat.ML\]](#).
- [34] C. Englert, P. Galler, P. Harris, and M. Spannowsky, *Machine Learning Uncertainties with Adversarial Neural Networks*, Eur. Phys. J. **C79** no. 1, (2019) 4, [arXiv:1807.08763 \[hep-ph\]](#).

- [35] L. Evans and P. Bryant, *LHC Machine*, JINST **3** (2008) S08001.
- [36] A. L. Read, *Presentation of search results: the CLs technique*, Journal of Physics G: Nuclear and Particle Physics **28** no. 10, (sep, 2002)2693–2704.
<https://doi.org/10.1088/0954-3899/28/10/313>.
- [37] A. L. Read, *Modified frequentist analysis of search results (the CLs method)*,
<https://cds.cern.ch/record/451614>.
- [38] A. Buckley *et al.*, *General-purpose event generators for LHC physics*, Physics Reports **504** no. 5, (Jul, 2011) 145–233, [arXiv:1101.2599](https://arxiv.org/abs/1101.2599) [hep-ph].
<http://dx.doi.org/10.1016/j.physrep.2011.03.005>.
- [39] J. Allison *et al.*, *Geant4 developments and applications*, IEEE Transactions on Nuclear Science **53** no. 1, (Feb, 2006)270–278.
- [40] **DELPHES 3** Collaboration, J. de Favereau *et al.*, *DELPHES 3, A modular framework for fast simulation of a generic collider experiment*, JHEP **02** (2014) 057,
[arXiv:1307.6346](https://arxiv.org/abs/1307.6346) [hep-ex].
- [41] **AMVA4NewPhysics** Collaboration, *Multivariate Analysis Methods for Higgs Boson Searches At The Large Hadron Collider*, tech. rep., Feb, 2017.
<https://docs.infn.it/share/s/VnuDzh8WSfmXME2cuVyKLQ>.
- [42] **AMVA4NewPhysics** Collaboration, *Multivariate Analysis Methods for Higgs Boson Searches At The Compact Muon Solenoid*, tech. rep., Oct, 2017. CMS-internal analysis note: CMS-AN-2016/407.
- [43] **AMVA4NewPhysics** Collaboration, *Classification and Regression Tools in Higgs Measurements*, tech. rep., Oct, 2018.
<https://docs.infn.it/share/s/c9veKvvfT3-WmoC6TW66jQ>.
- [44] M. E. Peskin and D. V. Schroeder, *An introduction to quantum field theory*. Westview, Boulder, CO, 1995. <https://cds.cern.ch/record/257493>.
- [45] D. Griffiths, *Introduction To Elementary Particles*. John Wiley & Sons, 1987.
- [46] K. S. Krane, *Introductory nuclear physics*. Wiley, New York, NY, 1988.
<https://cds.cern.ch/record/359790>.
- [47] R. M. Eisberg and R. Resnick, *Quantum physics of atoms, molecules, solids, nuclei and particles; 2nd ed.* Wiley, New York, NY, 1985. <https://cds.cern.ch/record/105889>.
- [48] C. Maxwell, *The Standard Model*, Lectures presented at the School for Young High Energy Physicists, Somerville College, Oxford (Sep., 2012). Lecture notes used as supplementary material for the gauge field theory sub-module of the particle theory lecture course at Durham University in 2012-13.

- [49] S. L. Glashow, *The renormalizability of vector meson interactions*, Nucl. Phys. **10** (1959)107–117.
- [50] S. Weinberg, *A Model of Leptons*, Phys. Rev. Lett. **19** (Nov, 1967)1264–1266.
<https://link.aps.org/doi/10.1103/PhysRevLett.19.1264>.
- [51] A. Salam and J. C. Ward, *Weak and electromagnetic interactions*, Il Nuovo Cimento (1955-1965) **11** no. 4, (1959)568–577. <https://doi.org/10.1007/BF02726525>.
- [52] F. Capozzi, E. Lisi, A. Marrone, D. Montanino, and A. Palazzo, *Neutrino masses and mixings: Status of known and unknown 3 ν parameters*, Nuclear Physics B **908** (Jul, 2016) 218–234. <http://dx.doi.org/10.1016/j.nuclphysb.2016.02.016>.
- [53] **Particle Data Group** Collaboration, M. Tanabashi *et al.*, *Review of Particle Physics*, Phys. Rev. D **98** (Aug, 2018) 030001.
<https://link.aps.org/doi/10.1103/PhysRevD.98.030001>.
- [54] N. Cabibbo, *Unitary Symmetry and Leptonic Decays*, Phys. Rev. Lett. **10** (Jun, 1963)531–533. <https://link.aps.org/doi/10.1103/PhysRevLett.10.531>.
- [55] M. Kobayashi and T. Maskawa, *CP-Violation in the Renormalizable Theory of Weak Interaction*, Progress of Theoretical Physics **49** no. 2, (02, 1973)652–657,
<https://academic.oup.com/ptp/article-pdf/49/2/652/5257692/49-2-652.pdf>.
<https://doi.org/10.1143/PTP.49.652>.
- [56] O. W. Greenberg, *Spin and Unitary-Spin Independence in a Paraquark Model of Baryons and Mesons*, Phys. Rev. Lett. **13** (Nov, 1964)598–602.
<https://link.aps.org/doi/10.1103/PhysRevLett.13.598>.
- [57] M. Y. Han and Y. Nambu, *Three-Triplet Model with Double SU(3) Symmetry*, Phys. Rev. **139** (Aug, 1965)B1006–B1010.
<https://link.aps.org/doi/10.1103/PhysRev.139.B1006>.
- [58] H. Fritzsch, M. Gell-Mann, and H. Leutwyler, *Advantages of the color octet gluon picture*, Physics Letters B **47** no. 4, (1973)365 – 368.
<http://www.sciencedirect.com/science/article/pii/0370269373906254>.
- [59] P. W. Anderson, *Plasmons, Gauge Invariance, and Mass*, Phys. Rev. **130** (Apr, 1963)439–442. <https://link.aps.org/doi/10.1103/PhysRev.130.439>.
- [60] F. Englert and R. Brout, *Broken Symmetry and the Mass of Gauge Vector Mesons*, Phys. Rev. Lett. **13** (Aug, 1964)321–323.
<https://link.aps.org/doi/10.1103/PhysRevLett.13.321>.
- [61] P. W. Higgs, *Broken Symmetries and the Masses of Gauge Bosons*, Phys. Rev. Lett. **13** (Oct, 1964)508–509. <https://link.aps.org/doi/10.1103/PhysRevLett.13.508>.

- [62] G. S. Guralnik, C. R. Hagen, and T. W. B. Kibble, *Global Conservation Laws and Massless Particles*, Phys. Rev. Lett. **13** (Nov, 1964) 585–587.
<https://link.aps.org/doi/10.1103/PhysRevLett.13.585>.
- [63] M. Lamont, *Status of the LHC*, J. Phys.: Conf. Ser. **455** (2013) 012001.
- [64] Image compiled from code made available under CC BY-SA 4.0 by Izaak Neutelings and further modified by myself. Original code source:
https://wiki.physik.uzh.ch/cms/latex:example_spherical_coordinates
(accessed: 2020/03/06), modified version available at https://github.com/GilesStrong/cc_by_sa_4/blob/master/cms_coord_system.tex.
- [65] Image made available under CC BY-SA 2.0 by OpenStreetMap. Image source
https://commons.wikimedia.org/wiki/File:Location_Large_Hadron_Collider.PNG
(accessed: 2020/03/06).
- [66] CMS Collaboration, *The CMS Experiment at the CERN LHC*, JINST **3** (2008) S08004.
- [67] D. Barney, *CMS Detector Slice*, tech. rep., Jan, 2016.
<https://cds.cern.ch/record/2120661>. CMS Collection.
- [68] CMS Collaboration, *The CMS magnet project: Technical Design Report*. Technical Design Report CMS. CERN, Geneva, 1997. <https://cds.cern.ch/record/331056>.
- [69] CMS Collaboration, M. Lipinski, *The Phase-1 Upgrade of the CMS Pixel Detector*, Tech. Rep. CMS-CR-2017-135. 06, CERN, Geneva, May, 2017.
<https://cds.cern.ch/record/2265423>.
- [70] J. Mans *et al.*, *CMS Technical Design Report for the Phase 1 Upgrade of the Hadron Calorimeter*, Tech. Rep. CERN-LHCC-2012-015. CMS-TDR-10, Sep, 2012.
<https://cds.cern.ch/record/1481837>. Additional contact persons: Jeffrey Spalding, Fermilab, spalding@cern.ch, Didier Contardo, Universite Claude Bernard-Lyon I, contardo@cern.ch.
- [71] S. I. Cooper, *Phase I Upgrade of the CMS Hadron Calorimeter*, Nuclear and Particle Physics Proceedings **273-275** (2016) 1002 – 1007.
<http://www.sciencedirect.com/science/article/pii/S240560141500646X>. 37th International Conference on High Energy Physics (ICHEP).
- [72] CMS-Totem Collaboration, M. Albrow *et al.*, *CMS-TOTEM Precision Proton Spectrometer*, Tech. Rep. CERN-LHCC-2014-021. TOTEM-TDR-003. CMS-TDR-13, Sep, 2014. <https://cds.cern.ch/record/1753795>.
- [73] CMS Collaboration, G. L. Bayatyan *et al.*, *CMS TriDAS project: Technical Design Report, Volume 1: The Trigger Systems*. Technical Design Report CMS.
<https://cds.cern.ch/record/706847>.

- [74] **CMS** Collaboration, S. Cittolin, A. Rácz, and P. Sphicas, *CMS The TriDAS Project: Technical Design Report, Volume 2: Data Acquisition and High-Level Trigger. CMS trigger and data-acquisition project*. Technical Design Report CMS. CERN, Geneva, 2002. <https://cds.cern.ch/record/578006>.
- [75] **CMS** Collaboration, *CMS Software: CMSSW*, 2005. <https://github.com/cms-sw/cmssw>.
- [76] **CMS** Collaboration, *The Phase-2 Upgrade of the CMS Tracker*, Tech. Rep. CERN-LHCC-2017-009. CMS-TDR-014, CERN, Geneva, Jun, 2017. <https://cds.cern.ch/record/2272264>.
- [77] **CMS** Collaboration, *The Phase-2 Upgrade of the CMS Barrel Calorimeters*, Tech. Rep. CERN-LHCC-2017-011. CMS-TDR-015, CERN, Geneva, Sep, 2017. <https://cds.cern.ch/record/2283187>. This is the final version, approved by the LHCC.
- [78] **CMS** Collaboration, *The Phase-2 Upgrade of the CMS Endcap Calorimeter*, Tech. Rep. CERN-LHCC-2017-023. CMS-TDR-019, CERN, Geneva, Nov, 2017. <https://cds.cern.ch/record/2293646>. Technical Design Report of the endcap calorimeter for the Phase-2 upgrade of the CMS experiment, in view of the HL-LHC run.
- [79] **CMS** Collaboration, *TECHNICAL PROPOSAL FOR A MIP TIMING DETECTOR IN THE CMS EXPERIMENT PHASE 2 UPGRADE*, Tech. Rep. CERN-LHCC-2017-027. LHCC-P-009, CERN, Geneva, Dec, 2017. <https://cds.cern.ch/record/2296612>.
- [80] S. Amrouche *et al.*, *The Tracking Machine Learning Challenge: Accuracy Phase*, The Springer Series on Challenges in Machine Learning (Nov, 2019) 231–264. http://dx.doi.org/10.1007/978-3-030-29135-8_9.
- [81] **CMS** Collaboration, *Description and performance of track and primary-vertex reconstruction with the CMS tracker*, Journal of Instrumentation **9** no. 10, (Oct, 2014) P10009–P10009. <http://dx.doi.org/10.1088/1748-0221/9/10/P10009>.
- [82] P. Billoir, *Progressive track recognition with a Kalman-like fitting procedure*, Computer Physics Communications **57** no. 1, (1989)390 – 394. <http://www.sciencedirect.com/science/article/pii/001046558990249X>.
- [83] P. Billoir and S. Qian, *Simultaneous pattern recognition and track fitting by the Kalman filtering method*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **294** no. 1, (1990)219 – 228. <http://www.sciencedirect.com/science/article/pii/016890029091835Y>.
- [84] R. Mankel, *A concurrent track evolution algorithm for pattern recognition in the HERA-B main tracking system*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **395** no. 2, (1997)169 – 184. <http://www.sciencedirect.com/science/article/pii/S0168900297007055>.

- [85] R. Frühwirth, *Application of Kalman filtering to track and vertex fitting*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **262** no. 2, (1987)444 – 450.
<http://www.sciencedirect.com/science/article/pii/0168900287908874>.
- [86] R. Frühwirth, W. Waltenberger, and P. Vanlaer, *Adaptive Vertex Fitting*, Tech. Rep. CMS-NOTE-2007-008, CERN, Geneva, Mar, 2007.
<https://cds.cern.ch/record/1027031>.
- [87] CMS Collaboration, *Particle-Flow Event Reconstruction in CMS and Performance for Jets, Taus, and MET*, Tech. Rep. CMS-PAS-PFT-09-001, CERN, Geneva, Apr, 2009.
<https://cds.cern.ch/record/1194487>.
- [88] CMS Collaboration, *Commissioning of the Particle-flow Event Reconstruction with the first LHC collisions recorded in the CMS detector*, Tech. Rep. CMS-PAS-PFT-10-001, 2010. <https://cds.cern.ch/record/1247373>.
- [89] CMS Collaboration, A. M. Sirunyan *et al.*, *Particle-flow reconstruction and global event description with the CMS detector*, JINST **12** (2017) P10003, [arXiv:1706.04965](https://arxiv.org/abs/1706.04965) [physics.ins-det].
- [90] CMS Collaboration, *Performance of CMS muon reconstruction in pp collision events at $\sqrt{s} = 7\text{TeV}$* , Journal of Instrumentation **7** no. 10, (Oct, 2012) P10002–P10002.
<http://dx.doi.org/10.1088/1748-0221/7/10/P10002>.
- [91] E. Meschi, T. Monteiro, C. Seez, and P. Vikas, *Electron Reconstruction in the CMS Electromagnetic Calorimeter*, Tech. Rep. CMS-NOTE-2001-034, CERN, Geneva, Jun, 2001. <https://cds.cern.ch/record/687345>.
- [92] W. Adam, R. Frühwirth, A. Strandlie, and T. Todorov, *Reconstruction of electrons with the Gaussian-sum filter in the CMS tracker at the LHC*, Journal of Physics G: Nuclear and Particle Physics **31** no. 9, (Jul, 2005) N9–N20.
<http://dx.doi.org/10.1088/0954-3899/31/9/N01>.
- [93] G. P. Salam and G. Soyez, *A practical seedless infrared-safe cone jet algorithm*, Journal of High Energy Physics **2007** no. 05, (May, 2007) 086–086.
<http://dx.doi.org/10.1088/1126-6708/2007/05/086>.
- [94] S. D. Ellis and D. E. Soper, *Successive combination jet algorithm for hadron collisions*, Physical Review D **48** no. 7, (Oct, 1993) 3160–3166.
<http://dx.doi.org/10.1103/PhysRevD.48.3160>.
- [95] Y. Dokshitzer, G. Leder, S. Moretti, and B. Webber, *Better jet clustering algorithms*, Journal of High Energy Physics **1997** no. 08, (Aug, 1997) 001–001.
<http://dx.doi.org/10.1088/1126-6708/1997/08/001>.
- [96] M. Cacciari, G. P. Salam, and G. Soyez, *The Anti- $k(t)$ jet clustering algorithm*, JHEP **04** (2008) 063, [arXiv:0802.1189](https://arxiv.org/abs/0802.1189) [hep-ph].

- [97] D. Bertolini, P. Harris, M. Low, and N. Tran, *Pileup Per Particle Identification*, JHEP **10** (2014) 059, [arXiv:1407.6013 \[hep-ph\]](#).
- [98] CMS Collaboration, A. M. Sirunyan *et al.*, *Identification of heavy-flavour jets with the CMS detector in pp collisions at 13 TeV*, JINST **13** no. 05, (2018) P05011, [arXiv:1712.07158 \[physics.ins-det\]](#).
- [99] CMS Collaboration, S. Chatrchyan *et al.*, *Identification of b-Quark Jets with the CMS Experiment*, JINST **8** (2013) P04013, [arXiv:1211.4462 \[hep-ex\]](#).
- [100] CMS Collaboration, *CMS Phase 1 heavy flavour identification performance and developments*, <https://cds.cern.ch/record/2263802>.
- [101] M. Stoye *et al.*, *Deep learning in jet reconstruction at CMS*, Journal of Physics: Conference Series **1085** (sep, 2018) 042029. <https://doi.org/10.1088%2F1742-6596%2F1085%2F4%2F042029>.
- [102] Image source <https://twiki.cern.ch/twiki/pub/CMSPublic/BTV13TeV2017DeepJet/DeepJet-schematic.pdf> (accessed: 2020/03/18).
- [103] CMS Collaboration, *Performance of τ -lepton reconstruction and identification in CMS*, Journal of Instrumentation **7** no. 1, (Jan., 2012) 1001, [arXiv:1109.6034 \[physics.ins-det\]](#).
- [104] CMS Collaboration, V. Khachatryan *et al.*, *Reconstruction and identification of τ lepton decays to hadrons and ν_τ at CMS*, JINST **11** no. 01, (2016) P01019, [arXiv:1510.07488 \[physics.ins-det\]](#).
- [105] CMS Collaboration, A. M. Sirunyan *et al.*, *Performance of reconstruction and identification of τ leptons decaying to hadrons and ν_τ in pp collisions at $\sqrt{s} = 13$ TeV*, JINST **13** no. 10, (2018) P10005, [arXiv:1809.02816 \[hep-ex\]](#).
- [106] CMS Collaboration, *Performance of the DeepTau algorithm for the discrimination of taus against jets, electron, and muons*, <https://cds.cern.ch/record/2694158>.
- [107] K. Androsov, R. Manzoni, J. Steggemann, and R. Wolf, *DeepTau: DNN-based multi-class tau identification algorithm*, tech. rep., 2020. CMS-internal analysis note: CMS-AN-2019/260.
- [108] L. Bianchini, J. Conway, E. K. Friis, and C. Veelken, *Reconstruction of the Higgs mass in $H \rightarrow \tau\tau$ Events by Dynamical Likelihood techniques*, Journal of Physics: Conference Series **513** no. 2, (2014) 022035. <http://stacks.iop.org/1742-6596/513/i=2/a=022035>.
- [109] L. Bianchini *et al.*, *Reconstruction of the Higgs mass in events with Higgs bosons decaying into a pair of τ leptons using matrix element techniques*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **862** (2017) 54 – 84. <http://www.sciencedirect.com/science/article/pii/S0168900217305259>.

- [110] CMS Collaboration, S. Chatrchyan *et al.*, *Search for Neutral Minimal Supersymmetric Standard Model Higgs Bosons Decaying to Tau Pairs in pp Collisions at $\sqrt{s} = 7$ TeV*, Phys. Rev. Lett. **106** (Jun, 2011) 231801.
<https://link.aps.org/doi/10.1103/PhysRevLett.106.231801>.
- [111] K. Kondo, *Dynamical Likelihood Method for Reconstruction of Events with Missing Momentum. I. Method and Toy Models*, Journal of the Physical Society of Japan **57** no. 12, (1988)4126–4140, <https://doi.org/10.1143/JPSJ.57.4126>.
<https://doi.org/10.1143/JPSJ.57.4126>.
- [112] M. Hoffmann, *Search for the decay of a heavy Higgs boson decaying to two light Higgs bosons using a kinematic fit*. PhD thesis, Hamburg U., 2016.
<https://inspirehep.net/record/1503235/files/ThesisHoffmannMalte.pdf>.
- [113] R. Placakyte, *Parton Distribution Functions*, arXiv:1111.5452 [hep-ph].
- [114] A. Martin, W. Stirling, R. Thorne, and G. Watt, *Parton distributions for the LHC*, Eur. Phys. J. C **63** (2009)189–285, arXiv:0901.0002 [hep-ph].
- [115] G. Marchesini and B. Webber, *Simulation of QCD coherence in heavy quark production and decay*, Nuclear Physics B **330** no. 1, (1990)261 – 283.
<http://www.sciencedirect.com/science/article/pii/055032139090310A>.
- [116] E. Norrbin and T. Sjöstrand, *QCD radiation off heavy particles*, Nuclear Physics B **603** no. 1-2, (Jun, 2001) 297–342. [http://dx.doi.org/10.1016/S0550-3213\(01\)00099-2](http://dx.doi.org/10.1016/S0550-3213(01)00099-2).
- [117] T. Sjöstrand *et al.*, *An Introduction to PYTHIA 8.2*, Comput. Phys. Commun. **191** (2015)159–177, arXiv:1410.3012 [hep-ph].
- [118] S. Catani, S. Dittmaier, M. H. Seymour, and Z. Trócsányi, *The dipole formalism for next-to-leading order QCD calculations with massive partons*, Nuclear Physics B **627** no. 1-2, (Apr, 2002) 189–265. [http://dx.doi.org/10.1016/S0550-3213\(02\)00098-6](http://dx.doi.org/10.1016/S0550-3213(02)00098-6).
- [119] M. Bähr *et al.*, *Herwig++ physics and manual*, The European Physical Journal C **58** no. 4, (Nov, 2008) 639–707. <http://dx.doi.org/10.1140/epjc/s10052-008-0798-9>.
- [120] J. Bellm *et al.*, *Herwig 7.0/Herwig++ 3.0 release note*, Eur. Phys. J. C **76** no. 4, (2016) 196, arXiv:1512.01178 [hep-ph].
- [121] T. Gleisberg, S. Hoeche, F. Krauss, M. Schonherr, S. Schumann, *et al.*, *Event generation with SHERPA 1.1*, JHEP **0902** (2009) 007, arXiv:0811.4622 [hep-ph].
- [122] S. Hoeche, F. Krauss, S. Schumann, and F. Siegert, *QCD matrix elements and truncated showers*, JHEP **05** (2009) 053, arXiv:0903.1219 [hep-ph].
- [123] S. Frixione and B. R. Webber, *Matching NLO QCD computations and parton shower simulations*, JHEP **06** (2002) 029, arXiv:hep-ph/0204244 [hep-ph].

- [124] S. Frixione, P. Nason, and C. Oleari, *Matching NLO QCD computations with Parton Shower simulations: the POWHEG method*, JHEP **11** (2007) 070, [arXiv:0709.2092 \[hep-ph\]](#).
- [125] T. Gehrmann, S. Hoeche, F. Krauss, M. Schonherr, and F. Siegert, *NLO QCD matrix elements + parton showers in $e^+e^- \rightarrow \text{hadrons}$* , JHEP **01** (2013) 144, [arXiv:1207.5031 \[hep-ph\]](#).
- [126] B. Andersson, G. Gustafson, G. Ingelman, and T. Sjöstrand, *Parton fragmentation and string dynamics*, Physics Reports **97** no. 2, (1983)31 – 145.
<http://www.sciencedirect.com/science/article/pii/0370157383900807>.
- [127] B. Andersson, *The Lund model*, Camb. Monogr. Part. Phys. Nucl. Phys. Cosmol. **7** (1997)1–471.
- [128] D. Amati and G. Veneziano, *Preconfinement as a property of perturbative QCD*, Physics Letters B **83** no. 1, (1979)87 – 92.
<http://www.sciencedirect.com/science/article/pii/0370269379908967>.
- [129] D. J. Lange, *The EvtGen particle decay simulation package*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **462** no. 1, (2001)152 – 155.
<http://www.sciencedirect.com/science/article/pii/S0168900201000894>.
BEAUTY2000, Proceedings of the 7th Int. Conf. on B-Physics at Hadron Machines.
- [130] P. Golonka *et al.*, *The tauola-photos-F environment for the TAUOLA and PHOTOS packages, release II*, Computer Physics Communications **174** no. 10, (May, 2006) 818–835. <http://dx.doi.org/10.1016/j.cpc.2005.12.018>.
- [131] S. Jadach, Z. Was, R. Decker, and J. Kühn, *The tau decay library TAUOLA, version 2.4*, Computer Physics Communications **76** no. 3, (1993)361 – 380.
<http://www.sciencedirect.com/science/article/pii/001046559390061G>.
- [132] S. Jadach, J. H. Kühn, and Z. Was, *TAUOLA - a library of Monte Carlo programs to simulate decays of polarized tau leptons*, Computer Physics Communications **64** no. 2, (1991)275 – 299.
<http://www.sciencedirect.com/science/article/pii/001046559190038M>.
- [133] V. Khachatryan *et al.*, *Event generator tunes obtained from underlying event and multiparton scattering measurements*, The European Physical Journal C **76** no. 3, (Mar, 2016). <http://dx.doi.org/10.1140/epjc/s10052-016-3988-x>.
- [134] S. Agostinelli *et al.*, *Geant4—a simulation toolkit*, Nucl. Instr.Meth. A **506** no. 3, (2003)250 – 303.
<http://www.sciencedirect.com/science/article/pii/S0168900203013688>.
- [135] M. Selvaggi, *DELPHES 3: A modular framework for fast-simulation of generic collider experiments*, J. Phys. Conf. Ser. **523** (2014) 012033.

- [136] A. Mertens, *New features in Delphes 3*, J. Phys. Conf. Ser. **608** no. 1, (2015) 012045.
- [137] T. Dorigo, *Extraordinary claims: the 0.000029% solution*, EPJ Web Conf. **95** (2015) 02003.
- [138] J. Neyman and E. S. Pearson, *On the Problem of the Most Efficient Tests of Statistical Hypotheses*, Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character **231** (1933)289–337. <http://www.jstor.org/stable/91247>.
- [139] G. Cowan, K. Cranmer, E. Gross, and O. Vitells, *Asymptotic formulae for likelihood-based tests of new physics*, Eur. Phys. J. **C71** (2011) 1554, [arXiv:1007.1727](https://arxiv.org/abs/1007.1727) [physics.data-an]. [Erratum: Eur. Phys. J. C73,2501(2013)].
- [140] S. Wilks, *The Large-Sample Distribution of the Likelihood Ratio for Testing Composite Hypotheses*, Annals Math. Statist. **9** no. 1, (1938)60–62.
- [141] A. Wald, *Tests of Statistical Hypotheses Concerning Several Parameters When the Number of Observations is Large*, Transactions of the American Mathematical Society **54** no. 3, (1943)426–482. <http://www.jstor.org/stable/1990256>.
- [142] G. Cowan, *Two developments in tests for discovery: use of weighted Monte Carlo events and an improved measure*, Progress on Statistical Issues in Searches, SLAC (2012). https://www.pp.rhul.ac.uk/~cowan/stat/cowan_slac_4jun12.pdf.
- [143] Fast Iron, *Blue Book for Bulldozers*, Kaggle (2012). <https://www.kaggle.com/c/bluebook-for-bulldozers>.
- [144] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Optimization by Simulated Annealing*, Science **220** no. 4598, (1983)671–680. <http://www.jstor.org/stable/1690046>.
- [145] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1998.
- [146] J. Mockus, *On Bayesian Methods for Seeking the Extremum and their Application.*, in *IFIP Congress*, pp. 195–200. 1977. <http://dblp.uni-trier.de/db/conf/ifip/ifip1977.html#Mockus77>.
- [147] J. B. Mockus and L. J. Mockus, *Bayesian approach to global optimization and application to multiobjective and constrained problems*, Journal of Optimization Theory and Applications **70** no. 1, (1991)157–172. <https://doi.org/10.1007/BF00940509>.
- [148] F. Rosenblatt, *The Perceptron: A probabilistic model for information storage and organization in the brain*, Psychological Review **65** no. 6, (1957)386–408.
- [149] A. Graves and J. Schmidhuber, *Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks*, Advances in Neural Information Processing Systems **22** (2009)545–552. <http://people.idsia.ch/~juergen/nips2009.pdf>.

- [150] G. E. Dahl, D. Yu, L. Deng, and A. Acero, *Large vocabulary continuous speech recognition with context-dependent DBN-HMMS*, 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2011)4688–4691.
- [151] Y. LeCun *et al.*, *Backpropagation Applied to Handwritten Zip Code Recognition*, Neural Comput. **1** no. 4, (Dec., 1989) 541–551.
<https://doi.org/10.1162/neco.1989.1.4.541>.
- [152] J. J. Hopfield, *Neural networks and physical systems with emergent collective computational abilities*, Proceedings of the National Academy of Sciences **79** no. 8, (1982)2554–2558, <https://www.pnas.org/content/79/8/2554.full.pdf>.
<https://www.pnas.org/content/79/8/2554>.
- [153] S. Hochreiter and J. Schmidhuber, *Long Short-Term Memory*, Neural Computation **9** no. 8, (1997)1735–1780, <https://doi.org/10.1162/neco.1997.9.8.1735>.
<https://doi.org/10.1162/neco.1997.9.8.1735>.
- [154] K. Cho *et al.*, *Learning phrase representations using RNN encoder-decoder for statistical machine translation*, in *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*. 2014.
- [155] J. Hadamard, *Mémoire sur le problème d'analyse relatif à l'équilibre des plaques élastiques encastrées*. Mémoires présentés par divers savants à l'Académie des sciences de l'Institut de France: Éxtrait. Imprimerie nationale, 1908.
<http://books.google.com.au/books?id=BTEPAAAAIAAJ>.
- [156] P. J. Werbos, *Applications of Advances in Nonlinear Sensitivity Analysis*, in *Proceedings of the 10th IFIP Conference, 31.8 - 4.9, NYC*, pp. 762–770. 1981.
- [157] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning representations by back-propagating errors*, Nature **323** (Oct., 1986)533–536.
- [158] S. Linnainmaa, *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors*, Master's thesis, Univ. Helsinki, 1970.
- [159] J. Schmidhuber, *Deep learning in neural networks: An overview*, Neural Networks **61** (Jan, 2015) 85–117. <http://dx.doi.org/10.1016/j.neunet.2014.09.003>.
- [160] J. Schmidhuber, *Learning Complex, Extended Sequences Using the Principle of History Compression*, Neural Computation **4** no. 2, (1992)234–242.
- [161] G. E. Hinton and R. R. Salakhutdinov, *Reducing the Dimensionality of Data with Neural Networks*, Science **313** no. 5786, (2006)504–507,
<https://science.sciencemag.org/content/313/5786/504.full.pdf>.
<https://science.sciencemag.org/content/313/5786/504>.

- [162] V. Nair and G. E. Hinton, *Rectified Linear Units Improve Restricted Boltzmann Machines*, in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, p. 807–814. Omnipress, Madison, WI, USA, 2010.
- [163] D. Clevert, T. Unterthiner, and S. Hochreiter, *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)*, CoRR **abs/1511.07289** (2015).
<http://arxiv.org/abs/1511.07289>.
- [164] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, *Self-Normalizing Neural Networks*, in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., pp. 971–980. Curran Associates, Inc., 2017. arXiv:1706.02515 [cs.LG].
<http://papers.nips.cc/paper/6698-self-normalizing-neural-networks.pdf>.
- [165] S. Ioffe and C. Szegedy, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, Stanford University, Tech. Rep. **37** (2015).
<http://jmlr.org/proceedings/papers/v37/ioffe15.pdf>.
- [166] P. Ramachandran, B. Zoph, and Q. V. Le, *Searching for Activation Functions*, CoRR **abs/1710.05941** (2017), arXiv:1710.05941. <http://arxiv.org/abs/1710.05941>.
- [167] X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, JMLR Workshop and Conference Proceedings **9** (2010).
<http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf>.
- [168] K. He, X. Zhang, S. Ren, and J. Sun, *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*, in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, p. 1026–1034. IEEE Computer Society, USA, 2015. arXiv:1502.01852 [cs.CV].
<https://doi.org/10.1109/ICCV.2015.123>.
- [169] Y. LeCun, L. Bottou, G. B. Orr, and K. R. Müller, *Efficient BackProp*, pp. 9–50. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.
<http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf>.
- [170] S. Ruder, *An overview of gradient descent optimization algorithms*, 2016.
<http://arxiv.org/abs/1609.04747>.
- [171] J. Martens, *Deep learning via Hessian-free optimization*, in *ICML*. 2010.
- [172] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning Internal Representations by Error Propagation*, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, D. E. Rumelhart and J. L. McClelland, eds., pp. 318–362. MIT Press, Cambridge, MA, 1986.
- [173] Y. Nesterov, *A method for unconstrained convex minimization problem with the rate of convergence $o(1/k^2)$* , Doklady AN USSR **269** (1983)543–547.

- [174] J. Duchi, E. Hazan, and Y. Singer, *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization*, J. Mach. Learn. Res. **12** (July, 2011)2121–2159.
<http://dl.acm.org/citation.cfm?id=1953048.2021068>.
- [175] T. Tieleman and G. Hinton, *Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude*, COURSERA: Neural Networks for Machine Learning (2012).
- [176] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, eds. 2015.
<http://arxiv.org/abs/1412.6980>.
- [177] T. Dozat, *Incorporating Nesterov momentum into Adam*, JMLR Workshop and Conference Proceedings **9** (2010).
http://cs229.stanford.edu/proj2015/054_report.pdf.
- [178] J. Kukacka, V. Golkov, and D. Cremers, *Regularization for Deep Learning: A Taxonomy*, CoRR **abs/1710.10686** (2017), arXiv:1710.10686.
<http://arxiv.org/abs/1710.10686>.
- [179] A. Y. Ng, *Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance*,
<https://doi.org/10.1145/1015330.1015435>.
- [180] F. Chollet, *Keras*, GitHub (2016). <https://github.com/fchollet/keras>.
- [181] A. Paszke et al., *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, arXiv:1912.01703 [cs.LG].
- [182] I. Loshchilov and F. Hutter, *Fixing Weight Decay Regularization in Adam*, CoRR **abs/1711.05101** (2017), arXiv:1711.05101. <http://arxiv.org/abs/1711.05101>.
- [183] S. Gugger and J. Howard, *AdamW and Super-convergence is now the fastest way to train neural nets*, fast.ai (Jul., 2018).
<https://www.fast.ai/2018/07/02/adam-weight-decay/>.
- [184] N. Frazier-Logue and S. J. Hanson, *The Stochastic Delta Rule: Faster and More Accurate Deep Learning Through Adaptive Weight Noise*, Neural Computation **32** no. 5, (2020)1018–1032, arXiv:1808.03578 [cs.LG].
- [185] S. J. Hanson, *A stochastic version of the delta rule*, Physica D: Nonlinear Phenomena **42** no. 1, (1990)265 – 272.
<http://www.sciencedirect.com/science/article/pii/016727899090081Y>.
- [186] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, in *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012, December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pp. 1106–1114. 2012.

- <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks>.
- [187] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*, Journal of Machine Learning Research **15** (2014) 1929–1958. <http://jmlr.org/papers/v15/srivastava14a.html>.
 - [188] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*, <https://cds.cern.ch/record/2253780>.
 - [189] L. Breiman, *Random Forests*, Mach. Learn. **45** no. 1, (Oct., 2001) 5–32. <https://doi.org/10.1023/A:1010933404324>.
 - [190] F. Pedregosa *et al.*, *Scikit-learn: Machine Learning in Python*, JMLR **12** no. 85, (2011) 2825–2830. <http://jmlr.org/papers/v12/pedregosa11a.html>.
 - [191] S. Di Vita, C. Grojean, G. Panico, M. Riembau, and T. Vantalon, *A global view on the Higgs self-coupling*, JHEP **09** (2017) 069, [arXiv:1704.01953](https://arxiv.org/abs/1704.01953) [hep-ph].
 - [192] S. Di Vita, *A global view on the Higgs self-coupling at future colliders*, 3rd CMS HH Workshop (2017). https://indico.cern.ch/event/685188/contributions/2931510/attachments/1628236/2593900/divita-global_view_higgs_selfcoupling.pdf. Accessed 2020/03/20.
 - [193] D. Buttazzo *et al.*, *Investigating the near-criticality of the Higgs boson*, JHEP **12** (2013) 089, [arXiv:1307.3536](https://arxiv.org/abs/1307.3536) [hep-ph].
 - [194] F. Maltoni, *Tri-linear HHH coupling extraction*, The 13th Workshop of the LHC Higgs Cross Section Working Group (2017). <https://indico.cern.ch/event/595100/contributions/2633727/attachments/1492334/2321397/HHH-maltoni.pdf>. Accessed 2020/03/22.
 - [195] R. Frederix *et al.*, *Higgs pair production at the LHC with NLO and parton-shower effects*, Phys. Lett. **B732** (2014) 142–149, [arXiv:1401.7340](https://arxiv.org/abs/1401.7340) [hep-ph].
 - [196] G. Degrandi, P. P. Giardino, F. Maltoni, and D. Pagani, *Probing the Higgs self coupling via single Higgs production at the LHC*, JHEP **12** (2016) 080, [arXiv:1607.04251](https://arxiv.org/abs/1607.04251) [hep-ph].
 - [197] **ATLAS, CMS** Collaboration, G. Aad *et al.*, *Measurements of the Higgs boson production and decay rates and constraints on its couplings from a combined ATLAS and CMS analysis of the LHC pp collision data at $\sqrt{s} = 7$ and 8 TeV*, JHEP **08** (2016) 045, [arXiv:1606.02266](https://arxiv.org/abs/1606.02266) [hep-ex].
 - [198] M. E. Peskin, *Estimation of LHC and ILC Capabilities for Precision Higgs Boson Coupling Measurements*, in *Proceedings, 2013 Community Summer Study on the Future of U.S. Particle Physics: Snowmass on the Mississippi (CSS2013): Minneapolis, MN, USA, July 29-August 6, 2013*. 2013. [arXiv:1312.4974](https://arxiv.org/abs/1312.4974) [hep-ph]. <http://www.slac.stanford.edu/econf/C1307292/docs/submittedArxivFiles/1312.4974.pdf>.

- [199] G. Degrandi, M. Fedele, and P. P. Giardino, *Constraints on the trilinear Higgs self coupling from precision observables*, JHEP **04** (2017) 155, [arXiv:1702.01737 \[hep-ph\]](#).
- [200] **ATLAS** Collaboration, M. Aaboud *et al.*, *Measurement of the W -boson mass in pp collisions at $\sqrt{s} = 7$ TeV with the ATLAS detector*, Eur. Phys. J. **C78** no. 2, (2018) 110, [arXiv:1701.07240 \[hep-ex\]](#). [Erratum: Eur. Phys. J. **C78**, no. 11, 898 (2018)].
- [201] **CDF** Collaboration, T. A. Aaltonen *et al.*, *Measurement of $\sin^2 \theta_{\text{eff}}^{\text{lept}}$ using e^+e^- pairs from γ^*/Z bosons produced in $p\bar{p}$ collisions at a center-of-momentum energy of 1.96 TeV*, Phys. Rev. **D93** no. 11, (2016) 112016, [arXiv:1605.02719 \[hep-ex\]](#). [Addendum: Phys. Rev. **D95**, no. 11, 119901 (2017)].
- [202] **D0** Collaboration, V. M. Abazov *et al.*, *Measurement of the Effective Weak Mixing Angle in $p\bar{p} \rightarrow Z/\gamma^* \rightarrow e^+e^-$ Events*, Phys. Rev. Lett. **115** no. 4, (2015) 041801, [arXiv:1408.5016 \[hep-ex\]](#).
- [203] A. Azatov, R. Contino, G. Panico, and M. Son, *Effective field theory analysis of double Higgs boson production via gluon fusion*, Phys. Rev. **D92** no. 3, (2015) 035001, [arXiv:1502.00539 \[hep-ph\]](#).
- [204] G. C. Branco *et al.*, *Theory and phenomenology of two-Higgs-doublet models*, Phys. Rept. **516** (2012) 1–102, [arXiv:1106.0034 \[hep-ph\]](#).
- [205] A. Djouadi *et al.*, *The post-Higgs MSSM scenario: Habemus MSSM?*, Eur. Phys. J. **C73** (2013) 2650, [arXiv:1307.5205 \[hep-ph\]](#).
- [206] A. Djouadi, L. Maiani, A. Polosa, J. Quevillon, and V. Riquer, *Fully covering the MSSM Higgs sector at the LHC*, JHEP **06** (2015) 168, [arXiv:1502.05653 \[hep-ph\]](#).
- [207] L. Randall and R. Sundrum, *A Large mass hierarchy from a small extra dimension*, Phys. Rev. Lett. **83** (1999) 3370–3373, [arXiv:hep-ph/9905221 \[hep-ph\]](#).
- [208] J. Alwall *et al.*, *The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations*, JHEP **07** (2014) 079, [arXiv:1405.0301 \[hep-ph\]](#).
- [209] **LHC Higgs Cross Section Working Group** Collaboration, D. de Florian *et al.*, *Handbook of LHC Higgs Cross Sections: 4. Deciphering the Nature of the Higgs Sector*, [arXiv:1610.07922 \[hep-ph\]](#).
- [210] **CMS** Collaboration, A. M. Sirunyan *et al.*, *Search for nonresonant Higgs boson pair production in the $b\bar{b}b\bar{b}$ final state at $\sqrt{s} = 13$ TeV*, JHEP **04** (2019) 112, [arXiv:1810.11854 \[hep-ex\]](#).
- [211] P. De Castro Manzano *et al.*, *Hemisphere Mixing: a Fully Data-Driven Model of QCD Multijet Backgrounds for LHC Searches*, PoS **EPS-HEP2017** (2017) 370, [arXiv:1712.02538 \[hep-ex\]](#).

- [212] **ATLAS** Collaboration, G. Aad *et al.*, *Searches for Higgs boson pair production in the $hh \rightarrow bb\tau\tau, \gamma\gamma WW^*, \gamma\gamma bb, bbbb$ channels with the ATLAS detector*, Phys. Rev. **D92** (2015) 092004, [arXiv:1509.04670 \[hep-ex\]](#).
- [213] **CMS** Collaboration, A. M. Sirunyan *et al.*, *Search for Higgs boson pair production in events with two bottom quarks and two tau leptons in proton-proton collisions at $\sqrt{s} = 13$ TeV*, [arXiv:1707.02909 \[hep-ex\]](#).
- [214] A. Sirunyan *et al.*, *Search for Higgs boson pair production in events with two bottom quarks and two tau leptons in proton-proton collisions at $s=13$ TeV*, Physics Letters B **778** (2018) 101 – 127.
<http://www.sciencedirect.com/science/article/pii/S037026931830008X>.
- [215] A. Barr, C. Lester, and P. Stephens, *$m(T2)$: The Truth behind the glamour*, J. Phys. **G29** (2003) 2343–2363, [arXiv:hep-ph/0304226 \[hep-ph\]](#).
- [216] A. Giraldi, *Optimisation of a multivariate analysis technique for the $t\bar{t}$ background rejection in the search for Higgs boson pair production in $b\bar{b}\tau^+\tau^-$ decay channel with the CMS experiment at the LHC*, Mar, 2018. <https://cds.cern.ch/record/2623854>. Presented 20 Apr 2018.
- [217] **ATLAS** Collaboration, M. Aaboud *et al.*, *Search for resonant and non-resonant Higgs boson pair production in the $b\bar{b}\tau^+\tau^-$ decay channel in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector*, Phys. Rev. Lett. **121** no. 19, (2018) 191801, [arXiv:1808.00336 \[hep-ex\]](#). [Erratum: Phys. Rev. Lett. 122, no. 8, 089901 (2019)].
- [218] **Particle Data Group** Collaboration, C. Patrignani *et al.*, *Review of Particle Physics*, Chin. Phys. **C40** no. 10, (2016) 100001.
- [219] P. Nason, *A New method for combining NLO QCD with shower Monte Carlo algorithms*, JHEP **11** (2004) 040, [arXiv:hep-ph/0409146 \[hep-ph\]](#).
- [220] S. Alioli, P. Nason, C. Oleari, and E. Re, *A general framework for implementing NLO calculations in shower Monte Carlo programs: the POWHEG BOX*, JHEP **06** (2010) 043, [arXiv:1002.2581 \[hep-ph\]](#).
- [221] K. Hamilton, P. Richardson, and J. Tully, *A Positive-Weight Next-to-Leading Order Monte Carlo Simulation for Higgs Boson Production*, JHEP **04** (2009) 116, [arXiv:0903.4345 \[hep-ph\]](#).
- [222] K. Rose, *Deterministic Annealing for Clustering, Compression, Classification, Regression, and Related Optimization Problems*, in *Proceedings of the IEEE*, pp. 2210–2239. 1998.
- [223] W. Erdmann and PSI, *Offline Primary Vertex Reconstruction with Deterministic Annealing Clustering*, CMS Internal Note: CMS-IN-2011-014 (Jun, 2011).
<https://cds.cern.ch/record/1365063>.

- [224] **CMS** Collaboration, *Performance of τ -lepton reconstruction and identification in CMS*, Journal of Instrumentation **7** (Jan., 2012) 1001, [arXiv:1109.6034](#) [physics.ins-det].
- [225] **CMS** Collaboration, S. Chatrchyan *et al.*, *Identification of b -quark jets with the CMS experiment*, JINST **8** (2013) P04013, [arXiv:1211.4462](#) [hep-ex].
- [226] J. Gallicchio *et al.*, *Multivariate discrimination and the Higgs + W/Z search*, JHEP **04** (2011) 069, [arXiv:1010.3698](#) [hep-ph].
- [227] K. Pearson, *LIII. On lines and planes of closest fit to systems of points in space*, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **2** no. 11, (1901)559–572.
- [228] S. Brochet *et al.*, *MoMEMta, a modular toolkit for the Matrix Element Method at the LHC*, Eur. Phys. J. **C79** no. 2, (2019) 126, [arXiv:1805.08555](#) [hep-ph].
- [229] H.-L. Lai *et al.*, *New parton distributions for collider physics*, Phys. Rev. **D82** (2010) 074024, [arXiv:1007.2241](#) [hep-ph].
- [230] A. Buckley *et al.*, *LHAPDF6: parton density access in the LHC precision era*, Eur. Phys. J. **C75** (2015) 132, [arXiv:1412.7420](#) [hep-ph].
- [231] **CMS** Collaboration, *CMS Higgs Combination toolkit*, 2017. <https://github.com/cms-analysis/HiggsAnalysis-CombinedLimit>.
- [232] **CMS** Collaboration, *Evidence for the decay of the Higgs Boson to Bottom Quarks*, Tech. Rep. CMS-PAS-HIG-16-044, CERN, Geneva, 2017. <https://cds.cern.ch/record/2278170>.
- [233] **CMS** Collaboration, A. M. Sirunyan *et al.*, *A deep neural network for simultaneous estimation of b jet energy and resolution*, [arXiv:1912.06046](#) [hep-ex].
- [234] J. Deng *et al.*, *ImageNet: A large-scale hierarchical image database*, in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pp. 248–255. IEEE Computer Society, 2009. <https://doi.org/10.1109/CVPR.2009.5206848>.
- [235] **ATLAS** Collaboration, *Dataset from the ATLAS Higgs Boson Challenge 2014*, CERN Open Data Portal (2014).
- [236] G. C. Strong, *LUMIN*, Mar., 2019. <https://doi.org/10.5281/zenodo.2601857>. Please check <https://github.com/GilesStrong/lumin/graphs/contributors> for the full list of contributors.
- [237] G. C. Strong, *On the impact of selected modern deep-learning techniques to the performance and celerity of classification models in an experimental high-energy physics use case*, Machine Learning: Science and Technology (Jun, 2020). <http://dx.doi.org/10.1088/2632-2153/ab983a>.

- [238] G. C. Strong, *HiggsML_Lumin*, Apr., 2020.
<https://doi.org/10.5281/zenodo.3754670>.
https://github.com/GilesStrong/HiggsML_Lumin.
- [239] T. Chen and C. Guestrin, *XGBoost: A Scalable Tree Boosting System*, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, p. 785–794. Association for Computing Machinery, New York, NY, USA, 2016. arXiv:1603.02754 [cs.LG].
<https://doi.org/10.1145/2939672.2939785>.
- [240] C. Adam-Bourdarios *et al.*, *The Higgs boson machine learning challenge*, in *Workshop on High-energy Physics and Machine Learning, HEPML NIPS 2014, Montreal, Quebec, Canada, December 8-13, 2014*, vol. 42 of *JMLR Workshop and Conference Proceedings*, pp. 19–55. JMLR.org, 2014. <http://proceedings.mlr.press/v42/cowa14.html>.
- [241] **ATLAS** Collaboration, G. Aad *et al.*, *The ATLAS Experiment at the CERN Large Hadron Collider*, JINST **3** (2008) S08003.
- [242] R. Johnson and T. Zhang, *Learning Nonlinear Functions Using Regularized Greedy Forest*, IEEE Transactions on Pattern Analysis and Machine Intelligence **36** no. 5, (2014)942–954, arXiv:1109.0887 [stat.ML].
- [243] B. Sun *et al.*, *SuperTML: Two-Dimensional Word Embedding and Transfer Learning Using ImageNet Pretrained CNN Models for the Classifications on Tabular Data*, CoRR abs/1903.06246 (2019), arXiv:1903.06246. <http://arxiv.org/abs/1903.06246>.
- [244] Image compiled from code made available under CC BY-SA 4.0 by Izaak Neutelings and further modified by me. Original code source:
https://wiki.physik.uzh.ch/cms/latex:example_spherical_coordinates
 (accessed: 2020/03/06), modified version available at https://github.com/GilesStrong/cc_by_sa_4/blob/master/atlas_coord_system.tex.
- [245] **ATLAS** Collaboration, *Dataset from the ATLAS Higgs Boson Machine Learning Challenge 2014*, CERN Open Data Portal (2014).
- [246] A. Elagin, P. Murat, A. Pranko, and A. Safonov, *A new mass reconstruction technique for resonances decaying to*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **654** no. 1, (Oct, 2011) 481–489. <http://dx.doi.org/10.1016/j.nima.2011.07.009>.
- [247] Y. Dodge, *The Concise Encyclopedia of Statistics*, pp. 502–505. Springer New York, New York, NY, 2008. https://doi.org/10.1007/978-0-387-32833-1_379.
- [248] L. N. Smith, *Cyclical Learning Rates for Training Neural Networks*, in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 464–472. 2017. arXiv:1506.01186.

- [249] C. Guo and F. Berkhahn, *Entity Embeddings of Categorical Variables*, CoRR abs/1604.06737 (2016), arXiv:1604.06737. <http://arxiv.org/abs/1604.06737>.
- [250] fast.ai, *Practical Deep Learning For Coders, Part 1 Lesson 4*, 2018. course18.fast.ai/lessons/lesson4.html. Accessed 2019/05/06.
- [251] Y. Bengio, *Practical Recommendations for Gradient-Based Training of Deep Architectures*, pp. 437–478. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. arXiv:1206.5533. https://doi.org/10.1007/978-3-642-35289-8_26.
- [252] K. He, X. Zhang, S. Ren, and J. Sun, *Deep Residual Learning for Image Recognition*, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. 2016. arXiv:1512.03385. <http://arxiv.org/abs/1512.03385>.
- [253] I. Loshchilov and F. Hutter, *SGDR: Stochastic Gradient Descent with Warm Restarts*, in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. arXiv:1608.03983. <https://openreview.net/forum?id=Skq89Scxx>.
- [254] G. Huang et al., *Snapshot Ensembles: Train 1, Get M for Free*, in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. arXiv:1704.00109. <https://openreview.net/forum?id=BJYwwY91l>.
- [255] T. Garipov et al., *Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs*, in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds., pp. 8789–8798. Curran Associates, Inc., 2018. arXiv:1802.10026 [stat.ML]. <http://papers.nips.cc/paper/8095-loss-surfaces-mode-connectivity-and-fast-ensembling-of-dnns.pdf>.
- [256] F. Draxler, K. Veschgini, M. Salmhofer, and F. Hamprecht, *Essentially No Barriers in Neural Network Energy Landscape*, in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, eds., vol. 80 of *Proceedings of Machine Learning Research*, pp. 1309–1318. PMLR, Stockholmsmässan, Stockholm Sweden, 10–15 Jul, 2018. arXiv:1803.00885 [stat.ML]. <http://proceedings.mlr.press/v80/draxler18a.html>.
- [257] P. Izmailov et al., *Averaging Weights Leads to Wider Optima and Better Generalization*, in *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, A. Globerson and R. Silva, eds., pp. 876–885. AUAI Press, 2018. arXiv:1803.05407. <http://auai.org/uai2018/proceedings/papers/313.pdf>.
- [258] L. N. Smith and N. Topin, *Super-convergence: very fast training of neural networks using large learning rates*, in *Artificial Intelligence and Machine Learning for*

- Multi-Domain Operations Applications*, T. Pham, ed., vol. 11006, pp. 369 – 386, International Society for Optics and Photonics. SPIE, 2019. arXiv:1708.07120. <https://doi.org/10.1117/12.2520589>.
- [259] L. N. Smith, *A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay*, CoRR **abs/1803.09820** (2018), arXiv:1803.09820. <http://arxiv.org/abs/1803.09820>.
- [260] J. Kukacka, V. Golkov, and D. Cremers, *Regularization for Deep Learning: A Taxonomy*, CoRR **abs/1710.10686** (2017), arXiv:1710.10686. <http://arxiv.org/abs/1710.10686>.
- [261] fast.ai, *fastai library documentation*, 2019. docs.fast.ai/. Accessed 2019/05/13.
- [262] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, *Densely Connected Convolutional Networks*, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269. 2017. arXiv:1608.06993.
- [263] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, *Deeply-Supervised Nets*, in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, G. Lebanon and S. V. N. Vishwanathan, eds., vol. 38 of *Proceedings of Machine Learning Research*, pp. 562–570. PMLR, San Diego, California, USA, 09–12 May, 2015. arXiv:1409.5185 [stat.ML]. <http://proceedings.mlr.press/v38/lee15a.html>.
- [264] H. Cheng *et al.*, *Wide & Deep Learning for Recommender Systems*, in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA, September 15, 2016*, A. Karatzoglou, B. Hidasi, D. Tikk, O. S. Shalom, H. Roitman, B. Shapira, and L. Rokach, eds., pp. 7–10. ACM, 2016. arXiv:1606.07792. <https://doi.org/10.1145/2988450.2988454>.
- [265] R. Tibshirani, *Regression Shrinkage and Selection via the Lasso*, Journal of the Royal Statistical Society. Series B (Methodological) **58** no. 1, (1996)267–288. <http://www.jstor.org/stable/2346178>.
- [266] G. Melis, *Dissecting the Winning Solution of the HiggsML Challenge*, in *Workshop on High-energy Physics and Machine Learning, HEPML NIPS 2014, Montreal, Quebec, Canada, December 8-13, 2014*, vol. 42 of *JMLR Workshop and Conference Proceedings*, pp. 57–67. JMLR.org, 2014. <http://proceedings.mlr.press/v42/meli14.html>.
- [267] E. M. Cepeda, S. Gori, P. Ilten, M. Kado, and F. Riva, *Higgs physics at the HL-LHC and HE-LHC*, CERN Yellow Reports: Monographs **7** no. 0, (2019) 221. <https://e-publishing.cern.ch/index.php/CYRM/article/view/952>.
- [268] CMS Collaboration, *Prospects for HH measurements at the HL-LHC*, Tech. Rep. CMS-PAS-FTR-18-019, CERN, Geneva, 2018. <https://cds.cern.ch/record/2652549>.

- [269] T. ATLAS and C. Collaborations, *Collection of notes from ATLAS and CMS*, CERN Yellow Reports: Monographs **7** no. 0, (2019).
<https://e-publishing.cern.ch/index.php/CYRM/article/view/956>.
- [270] CMS Collaboration, *A MIP Timing Detector for the CMS Phase-2 Upgrade*, Tech. Rep. CERN-LHCC-2019-003. CMS-TDR-020, CERN, Geneva, Mar, 2019.
<https://cds.cern.ch/record/2667167>.
- [271] A. Carvalho, F. Goertz, K. Mimasu, M. Gouzevitch, and A. Aggarwal, *On the reinterpretation of non-resonant searches for Higgs boson pairs*, [arXiv:1710.08261](https://arxiv.org/abs/1710.08261) [hep-ph].
- [272] G. Strong and M. Gallinaro, *A DNN-based classification approach for enhancing the search for Higgs boson pair production in the $b\bar{b}\tau\tau$ final state in pp collisions at $\sqrt{s} = 13$ TeV*, tech. rep., Dec, 2019. CMS-internal analysis note: CMS-AN-2019/188.
- [273] C. e. a. Amendola, *Search for Higgs boson pairs produced through gluon and vector boson fusion in the $b\bar{b}\tau\tau$ final state with Run-II data*, tech. rep., 2020. CMS-internal analysis note: CMS-AN-2018/121.
- [274] NNPDF Collaboration, R. D. Ball *et al.*, *Parton distributions from high-precision collider data*, Eur. Phys. J. C **77** no. 10, (2017) 663, [arXiv:1706.00428](https://arxiv.org/abs/1706.00428) [hep-ph].
- [275] CMS Collaboration, A. M. Sirunyan *et al.*, *Extraction and validation of a new set of CMS PYTHIA8 tunes from underlying-event measurements*, Eur. Phys. J. **C80** no. 1, (2020) 4, [arXiv:1903.12179](https://arxiv.org/abs/1903.12179) [hep-ex].
- [276] A. Carvalho *et al.*, *Higgs Pair Production: Choosing Benchmarks With Cluster Analysis*, JHEP **04** (2016) 126, [arXiv:1507.02245](https://arxiv.org/abs/1507.02245) [hep-ph].
- [277] CMS Collaboration, V. Khachatryan *et al.*, *Measurements of differential production cross sections for a Z boson in association with jets in pp collisions at $\sqrt{s} = 8$ TeV*, JHEP **04** (2017) 022, [arXiv:1611.03844](https://arxiv.org/abs/1611.03844) [hep-ex].
- [278] K. Androsov, M. A. Ciocci, and M. R. Di Domenico, *HH-BTag a ML-based method to select b jets from $H \rightarrow b\bar{b}$ decay, in $HH \rightarrow b\bar{b}\tau\tau$ searches*, tech. rep., 2020.
- [279] A. J. Larkoski, S. Marzani, G. Soyez, and J. Thaler, *Soft drop*, Journal of High Energy Physics **2014** no. 5, (May, 2014). [http://dx.doi.org/10.1007/JHEP05\(2014\)146](http://dx.doi.org/10.1007/JHEP05(2014)146).
- [280] J. L. Holgado, S. G. Lopez, and M. Rieger, *Search for $t\bar{t}H$ production in the $H \rightarrow b\bar{b}$ decay channel with leptonic $t\bar{t}$ decays in proton-proton collisions at $\sqrt{s} = 13$ TeV*, tech. rep., 2020. CMS-internal analysis note: CMS-AN-2020/095.
- [281] CMS Lumi POG, *Luminosity Physics Object Group (Lumi POG)*, TWiki .
<https://twiki.cern.ch/twiki/bin/view/CMS/TWikiLUM>.

- [282] CMS B tag POG, *Methods to apply b-tagging efficiency scale factors*, TWiki .
<https://twiki.cern.ch/twiki/bin/viewauth/CMS/BTagSFMethods>.
- [283] LHC Higgs Cross Section Working Group, *LHC Higgs Cross Section HH Sub-group*, TWiki . <https://twiki.cern.ch/twiki/bin/view/LHCPhysics/LHCHXSWGHH>.
- [284] LHC Higgs Cross Section Working, *SM Higgs Branching Ratios and Total Decay Widths (update in CERN Report4 2016)*, TWiki .
<https://twiki.cern.ch/twiki/bin/view/LHCPhysics/CERNYellowReportPageBR>.
- [285] CMS Tau POG, *Tau ID recommendations for Run-2: 2016, 2017, and 2018*, TWiki .
<https://twiki.cern.ch/twiki/bin/viewauth/CMS/TauIDRecommendationForRun2>.
- [286] CMS JetMet POG, *Run 2 reduced set of uncertainty sources*, TWiki .
https://twiki.cern.ch/twiki/bin/view/CMS/JECUncertaintySources#Run_2_reduced_set_of_uncertainty.
- [287] CMS Top POG, *The modeling of the top quark pT*, TWiki .
<https://twiki.cern.ch/twiki/bin/viewauth/CMS/TopPtReweighting>.
- [288] CMS Collaboration, *Reconstruction and identification of τ lepton decays to hadrons and ν_τ at CMS*, Journal of Instrumentation **11** no. 01, (Jan, 2016) P01019–P01019.
<http://dx.doi.org/10.1088/1748-0221/11/01/P01019>.
- [289] S. J. Pan and Q. Yang, *A Survey on Transfer Learning*, IEEE Trans. Knowl. Data Eng. **22** no. 10, (2010)1345–1359. <https://doi.org/10.1109/TKDE.2009.191>.
- [290] K. Weiss, T. M. Khoshgoftaar, and D. Wang, *A survey of transfer learning*, Journal of Big Data **3** no. 1, (2016) 9. <https://doi.org/10.1186/s40537-016-0043-6>.
- [291] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, *Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation*, in *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pp. 580–587. IEEE Computer Society, 2014. arXiv:1311.2524.
<https://doi.org/10.1109/CVPR.2014.81>.
- [292] L. Studer *et al.*, *A Comprehensive Study of ImageNet Pre-Training for Historical Document Image Analysis*, in *2019 International Conference on Document Analysis and Recognition, ICDAR 2019, Sydney, Australia, September 20-25, 2019*, pp. 720–725. IEEE, 2019. arXiv:1905.09113. <https://doi.org/10.1109/ICDAR.2019.00120>.
- [293] S. Merity, C. Xiong, J. Bradbury, and R. Socher, *Pointer Sentinel Mixture Models*, in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. arXiv:1609.07843. <https://openreview.net/forum?id=Byj72udxe>.

- [294] J. Howard and S. Ruder, *Universal Language Model Fine-tuning for Text Classification*, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, I. Gurevych and Y. Miyao, eds., pp. 328–339. Association for Computational Linguistics, 2018. arXiv:1801.06146. <https://www.aclweb.org/anthology/P18-1031/>.
- [295] K. He, R. B. Girshick, and P. Dollár, *Rethinking ImageNet Pre-training*, CoRR abs/1811.08883 (2018), arXiv:1811.08883. <http://arxiv.org/abs/1811.08883>.
- [296] R. A. Fisher, *The Use Of Multiple Measurements In Taxonomic Problems*, Annals of Eugenics **7** no. 2, (1936)179–188. <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-1809.1936.tb02137.x>.
- [297] M. Forina *et al.*, *PARVUS - An Extendible Package for Data Exploration, Classification and Correlation*. Institute of Pharmaceutical and Food Analysis and Technologies, Via Brigata Salerno, 16147 Genoa, Italy, 1991. <https://archive.ics.uci.edu/ml/datasets/wine>.
- [298] R. Kohavi, *Scaling up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid*, in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD'96*, p. 202–207. AAAI Press, 1996. <https://www.aaai.org/Papers/KDD/1996/KDD96-033.pdf>.
- [299] G. C. Strong, *SuperTML_HiggsML_Test*, Apr., 2020. <https://doi.org/10.5281/zenodo.3754668>. https://github.com/GilesStrong/SuperTML_HiggsML_Test.
- [300] J. Howard *et al.*, *fastai*, 2018. <https://github.com/fastai/fastai>.
- [301] J. Hu, L. Shen, and G. Sun, *Squeeze-and-Excitation Networks*, in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141. 2018. arXiv:1709.01507. <http://arxiv.org/abs/1709.01507>.
- [302] R. Brun and F. Rademakers, *ROOT: An object oriented data analysis framework*, Nucl. Instrum. Meth. **A389** (1997)81–86.
- [303] M. R. Whalley, D. Bourilkov, and R. C. Group, *The Les Houches accord PDFs (LHAPDF) and LHAGLUE*, in *HERA and the LHC: A Workshop on the implications of HERA for LHC physics. Proceedings, Part B*, pp. 575–581. 2005. arXiv:hep-ph/0508110 [hep-ph]. <http://lhapdf.hepforge.org>.
- [304] M. Cacciari, G. P. Salam, and G. Soyez, *FastJet User Manual*, Eur. Phys. J. **C72** (2012) 1896, arXiv:1111.6097 [hep-ph].
- [305] J. D. Hunter, *Matplotlib: A 2D graphics environment*, Computing In Science & Engineering **9** no. 3, (2007)90–95.

- [306] M. L. Waskom, *seaborn: statistical data visualization*, Journal of Open Source Software **6** no. 60, (2021) 3021. <https://doi.org/10.21105/joss.03021>.
- [307] Theano Development Team, *Theano: A Python framework for fast computation of mathematical expressions*, arXiv e-prints **abs/1605.02688** (May, 2016). <http://arxiv.org/abs/1605.02688>.
- [308] S. van der Walt, S. C. Colbert, and G. Varoquaux, *The NumPy Array: A Structure for Efficient Numerical Computation*, Computing in Science Engineering **13** no. 2, (March, 2011)22–30.
- [309] Wes McKinney, *Data Structures for Statistical Computing in Python*, in *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman, eds., pp. 56 – 61. 2010.
- [310] S. Seabold and J. Perktold, *Statsmodels: econometric and statistical modeling with Python*, in *Proceedings of the 9th Python in Science Conference*, pp. 57–61. 2010. <http://statsmodels.sourceforge.net/stable/>.
- [311] T. Parr and K. Turgutlu, *Feature importances for scikit random forests*,. <https://github.com/parrrt/random-forest-importances>. See also <https://explained.ai/rf-importance/index.html>.
- [312] T. Head *et al.*, *scikit-optimize/scikit-optimize: v0.5.2*, Mar., 2018. <https://doi.org/10.5281/zenodo.1207017>.
- [313] A. Murray, *python-pillow/Pillow: 6.0.0*, Apr., 2019. <https://doi.org/10.5281/zenodo.3733082>.
- [314] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, *Tensorflow: A system for large-scale machine learning*, in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283. 2016.
- [315] L. Jiangchun, *PDPbox*,. <https://github.com/SauceCat/PDPbox>.
- [316] J. Bai *et al.*, *ONNX: Open Neural Network Exchange*, 2017. <https://github.com/onnx/onnx>.
- [317] T. Jin *et al.*, *Tensorflow Backend for ONNX*, 2017. <https://github.com/onnx/onnx-tensorflow>.
- [318] M. Riga, *TensorFlow Interface for CMSSW*, 2018. <https://gitlab.cern.ch/mrieger/CMSSW-DNN>.

Chapter 12

Appendices

12.1 A study of the SuperTML result

As mentioned in Sec. 9.3.2, Ref. [243] claims to achieve an AMS of 3.979 through the use of pretrained models. If true, this would potentially be revolutionary for the HEP field, and so it is necessary to confirm and investigate further.

12.1.1 Pretraining and transfer learning

Overview

Pretraining is a method in which a model is trained to perform a task on a large dataset. The trained model can be used as a starting point for refinement on another dataset to perform a different task. This allows more powerful models to be used in situations where such models would otherwise not be able to be trained, or simply to provide a better initialisation than random weighting. This is known as transfer learning (see e.g. Ref. [289] and Ref. [290]).

One example is the pretraining of models on large collections of images, such as ImageNet [234], which allows the model to learn common and transferable features maps such as corners and edges, and more complex features like materials and objects. The large model can now be used as a starting point for learning on a much smaller dataset (e.g. R-CNN [291]), or for different tasks such as image analysis of historical texts [292] where only a small number of texts have human annotations due to the time and expertise required to label the data. Another example is the pretraining of models on large corpuses of texts, such as WikiText [293], in order to learn the basic grammar and vocabulary of a language, the trained model can then be further refined for other tasks [294].

Reference [295] notes, however, that the benefits of pretraining can be limited by the differences in target data-space, or task, notably in image data when the target task is sensitive to the spacial locations of objects in the image.

Transfer learning in tabular data

Model pretraining for computer vision and natural-language processing is achieved by using architectures which have some insensitivity to the dimensions of the input data; recursive and convolutional layers allow the model to be applied to images of different sizes, or sentences of different lengths, to those of the training data. In tabular data, however, the number of features can change considerably from one dataset to another, with no (current) architecture capable of dealing with this variation in an efficient manner. Even if there were such an architecture, the main benefit of pretraining comes from learning methods of feature extraction from unstructured data, whereas tabular data already has features extracted and these carry meanings specific to the dataset.

Imagine for example employee data for an international business. One could train a classification model to predict which of the employees leave during the data-taking period based on employee details. Now imagine similar data for employees working at a small startup company. The same features are potentially present for both datasets, so it is possible that the classifier trained for the international business might offer some transfer-learning benefits when fine-tuned on the smaller company, but severe differences in company and employee type will likely limit the benefits of using the pretrained model. Taking this idea further, the pretrained model is now fine-tuned to predict whether it rained on a given day using data which happens to have the same number as input features as the original training data. Here no transfer learning can take place; the input features do not share similar meanings and the target task is far too different to the original task.

12.1.2 The SuperTML method

The Super Tabular data Machine Learning method (SuperTML) is a way of allowing pretrained models to be applied to tabular data, such as the HiggsML dataset and potentially other HEP data. It avoids restrictions on the data dimensionality (number of features) by transforming each datapoint into its own image. Each image is black and the values of the features are written in white text on the image, as shown in Fig. 12.1.

Having transformed the data into images, it uses the data to fine-tune large models that have been pretrained on ImageNet. The authors demonstrate their method on four classification datasets: Iris [296], Wine [297], Adult [298], and HiggsML.

12.1.3 Re-implementation of the method

The authors do not provide a publicly accessible distribution of their method, and did not offer to provide their code during our correspondence; we are therefore left with re-implementing their method based on the paper description. In correspondence, the authors did clarify some technical points of their method, but not all. My implementation and experimental results are available at Ref. [299], and are documented below.

Image preparation

Images are prepared as 224x224 pixel three-channel (RGB) images. Pixels are initially set to (0,0,0) (black). Feature values are written in columns in the order as listed in the datasets in 13pt **FreeMonoBold** text in white (255,255,255) and saved in png format. The use of three-channel images, despite being greyscale, is due to the pretrained models expecting full-colour images. This method of data preparation is compatible with that of a later re-implementation to which the authors directed us (<https://github.com/EmjayAhn/SuperTML-pytorch>).

Model fine-tuning

There are several choices when fine-tuning models. A two-stage process can be adopted:

1. An initial training in which the pretrained model is frozen and not updated and extra linear layers are added at the end of the model and trained, allowing the new layers to adjust to the outputs of the pretrained model;
2. The full model is unfrozen and a second stage of training is performed, allowing more optimal feature maps to be learnt.

In correspondence, the authors confirmed that they did not add any extra layers to the end of the model, and that training was performed in a single stage on the unfrozen model. I perform the fine-tuning using the fast.ai library v1 [300], with 1cycle scheduling and no forms of data augmentation.

Implementation confirmation tests

To show that my implementation works I attempt to recover the authors' scores of 93.33 % accuracy on the Iris dataset and 97.30 % accuracy on the Wine dataset. I follow the authors' data split of 80 % training and 20 % validation. For the model I use ResNet-34 [252], instead of SE-net-154 [301] due to it being quicker to train for these confirmation tests. Despite the smaller model, my implementation achieves validation accuracies of 100 % and 97% on the Iris and Wine datasets, respectively. From these results I believe that our implementation of the method matches the authors'. Note that I do not test on the Adult dataset due to the authors' using one of their own methods to provide text embeddings for certain features in the data and I do not wish to introduce a source of uncertainty by implementing another of their techniques.

12.1.4 HiggsML tests

From the paper we read that the open-data version of the dataset is used, allowing the authors' to compute the AMS metric themselves on the testing data. They also say they use the same 250 000 events for training and the same 550 000 events for testing as were used in the Kaggle competition¹. To monitor overtraining I randomly sample 20 % of the training data for validation. The paper

¹The authors confirmed a typographical error in the paper where at one point the dataset sizes were quoted to be ten times smaller.

gives no mention of cut optimisation at all and the authors did not give any comment when asked, so we are left to assume that they assign events to the class with the highest prediction, i.e. a cut of 0.5 in the binary classification sense of the problem. The data images are prepared as normal, and I do not apply any preprocessing to them, (Fig. 4(a) in the paper gives no indication that the data are preprocessed). Transforming the dataset into images increases the size of the data from 187 MB to 9.8 GB (50 times larger).

For the model I fine-tune SE-net-154, and in testing found that the model converges within four epochs. The batch size was set to 20, which was the largest size that could safely fit in the 11 GB VRAM of the GPU (Nvidia 2080 Ti). Each epoch takes just over one hour 50 minutes; total training time is about seven and a half hours. A validation accuracy of around 83 % is reached.

The trained model is then applied to the testing dataset, which takes just over an hour. Computing the AMS for a cut of 0.5 gives public and private scores of 2.83 and 2.84, respectively, in disagreement with the authors' score of 3.98 ². One possibility is that the authors do optimise a cut, but computing the maximum possible AMS on both public and private datasets, results in AMS scores of 3.20 and 3.25, respectively.

During correspondence, the authors did not reply to questions about whether or not they split the testing dataset into the public and private samples; they are supplied together and must be manually separated by the user. This gives another possibility that the authors did not split the testing sample and instead computed the AMS on the entire sample. It should be noted from Eq. 9.1, that the AMS contains no self-normalising components, unlike other ML metrics such as accuracy. All normalisation comes from the weights of the samples on which it is computed. Since the public and private samples have the same normalisation (integrated luminosity) computing the AMS on the entire testing set equates to running with double the integrated luminosity and can be expected to increase the AMS by a factor of approximately $\sqrt{2}$: $\sqrt{2} \times 2.84 = 4.02$. Indeed computing the AMS on the entirety of the testing dataset produces an AMS of 4.01. Allowing for slight changes in model training and other sources of stochasticity, this score is compatible with the claimed score of 3.98. I therefore suspect that this is how the authors arrived at their score.

12.1.5 Closing remarks

Although the SuperTML provides a solution to the problem of running pretrained models on tabular data with differing numbers of features, it doesn't solve the problem that the features can have very different meanings. Additionally the target task is extremely different to the pretraining task; whilst the models may learn to recognise numerals during pretraining, the task never explicitly encourages the model to learn to perform mathematical operations on the numerals. It is therefore difficult to see how the pretraining really benefits the method, and indeed further testing showed that the same performance was achievable when starting from random initialisation.

²I indeed ran many experiments and were unable to recover the claimed score. These are all included in my GitHub repository.

Additionally I found that slight improvements were possible by transforming the feature values to the range $[0, 255]$ and encoding the data as solid, greyscale, blocks of pixels, as shown in Fig. 12.2; this cuts out the necessity of text recognition by the model and allows smaller images to be used since font size is no longer a limitation. Smaller images means that the increase in data file size is reduced, and larger batch-sizes may be used, which reduced the training time.

As it stands, though, the method is perhaps not best suited for data analysis in HEP, where we nowadays work with analysis-level data of at least several gigabytes. Even with the pixel-block encoding and 56×56 images, the increase in storage requirements is still a factor of 17. Additionally, the method is slow to train and to apply, and for final-state analysis the model will normally have to be retrained several times, occasionally at short notice. The final point is that not all analysers will have access to GPUs, and even fewer will have them on-demand. At least for a few years to come it will be necessary that analysis-level algorithms work reasonably on CPU, as well. Further discussion on the requirements and future goals for ML in HEP may be found in Ref. [17].

Certainly, though, the concept of pretraining domain-specific models in HEP is enticing; training data is usually computer simulated, and for analysis-level must be of very high quality, which requires the running of comparatively slow simulations, especially for detector simulations (e.g. GEANT [39, 134]). There are, however, software packages that still provide detector simulation but through the use of rough parameterisations of the detector responses (e.g. DELPHES [40, 135, 136]). It is possible then that powerful models could be pretrained using roughly simulated data, and then fine-tuned on smaller, but more accurate, samples of high-quality simulated data.

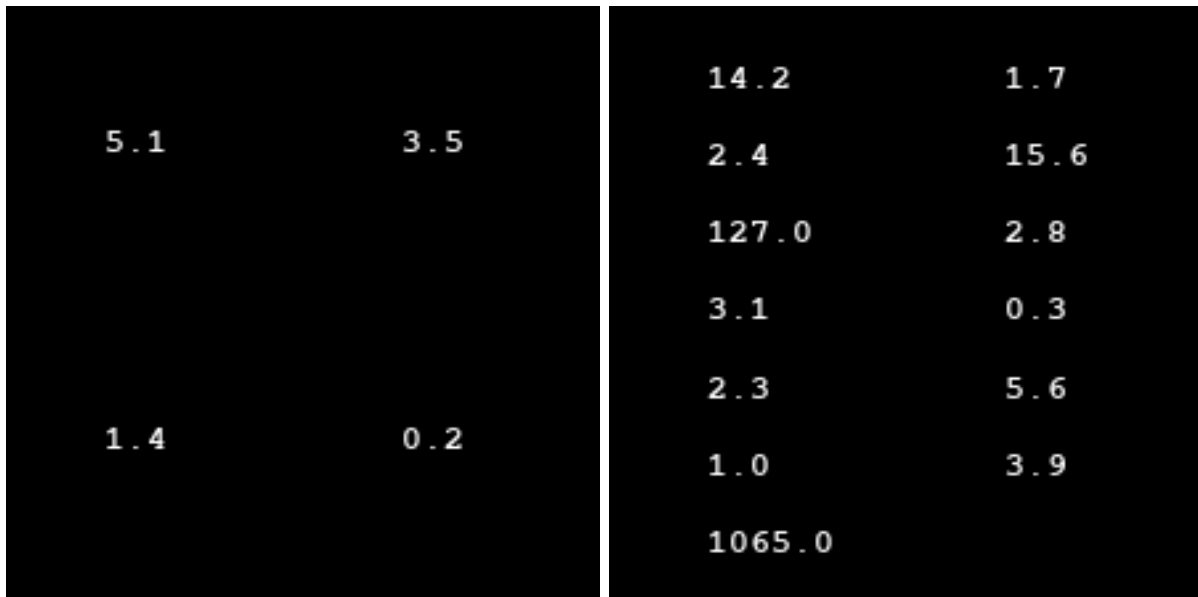
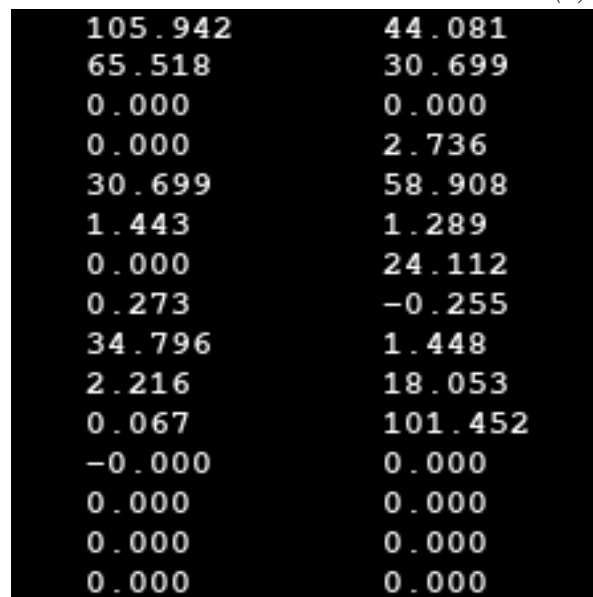
(a) *Iris*(b) *Wine*(c) *HiggsML*

Figure 12.1: Example images as encoded for the SuperTML method.

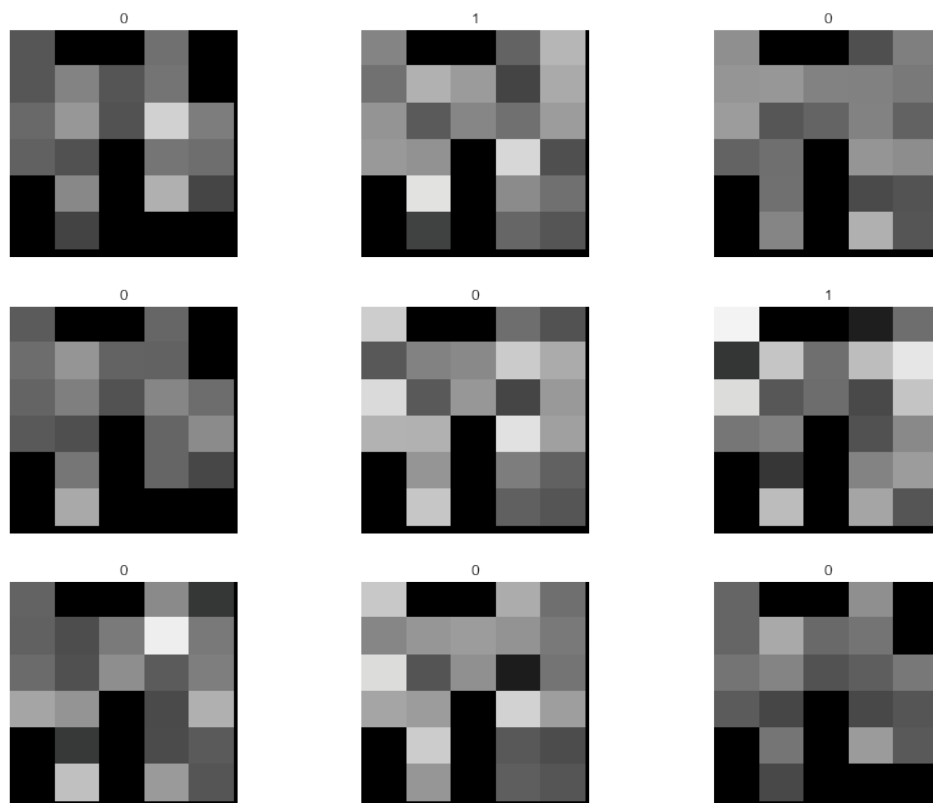


Figure 12.2: Examples of several datapoints encoded as greyscale blocks. The class of datapoint is indicated above each image: 0 - background, 1 - signal.

12.2 Software details

During my PhD, I made use of several software packages, which are detailed below. The preliminary investigation in Chapters 7 and 8 made use of the software detailed in Tab. 12.1. The optimisation studies performed in Chap. 9 and Sec. 12.1 made use of software detailed in Tab. 12.2, and the framework and notebook experiments are made available at Ref. [238]. The HL-LHC projection studies in Sec. 9.9 used the software detailed in Tab. 12.3. My contributions to the CMS search performed in Chap. 10 made use of the software detailed in Tab. 12.4. .

Software	Version	References	Use/Notes
KERAS	1.05 & 2.0.6	[180]	Implementing neural networks
ROOT	6.06/04	[302]	Analysis of Monte Carlo data
CMSSW	8.0.25	[75]	Analysis of Monte Carlo data
CMSSW	7.1.25 & 8.0.21	[75]	Production of MC data and event reconstruction
COMBINE	7.0	[231]	Significances and limits
MADGRAPH	5.2.4.2	[208]	MC generation
PYTHIA	8.219	[117]	MC generation
POWHEG BOX	2	[219, 124]	MC generation
		[220, 221]	
LHAPDF	5.9.1 & 6.1.6 & 6.2.1	[303, 230]	PDFs for MC generation
DELPHES	3.3.2	[40, 135, 136]	Detector simulation
GEANT	4	[39, 134]	Detector simulation
MoMEMTA	1.0.0 beta1	[228]	MEM implementation
XGBOOST	0.6a2	[239]	Implementing BDTs
FASTJET	3.1.3	[304]	Jet clustering
SCIKIT-LEARN	various	[190]	Cross-validation
MATPLOTLIB	various	[305]	Plot production
SEABORN	various	[306]	Plot production
THEANO	various	[307]	KERAS back-end
NUMPY	various	[308]	Data analysis and computation
PANDAS	various	[309]	Data analysis and computation
STATSMODELS	0.6.1	[310]	Kernel density-estimation

Table 12.1: Software used for the preliminary investigations

Software	Version	References	Use/Notes
LUMIN	0.0.0-0.3.1	[236]	Wrapping PYTORCH to implement new methods presented
PYTORCH	1.0.0	[181]	Implementing neural networks
SEABORN	0.8.1	[306]	Plot production
MATPLOTLIB	3.0.2	[305]	Plot production
PANDAS	0.23.4	[309]	Data analysis and computation
NUMPY	1.15.2	[308]	Data analysis and computation
SCIKIT-LEARN	0.20.0	[190]	Cross-validation and pre-processing
RFPIMP	1.3.4	[311]	Random Forest Permutation Importance and mutual dependency calculations
SCIKIT-OPTIMIZE	0.5.2	[312]	GP fitting and hyper-parameter partial-dependence calculations
FASTAI	1.0.55	[300]	Implementing the SuperTML method
PILLOW	6.0.0	[313]	Image preparation for SuperTML method

Table 12.2: Software used for the optimisation studies

Software	Version	References	Use/Notes
KERAS	2.1	[180]	Implementing neural networks
ROOT	6.0 & 6.12	[302]	Analysis of Monte Carlo data Preparation of histograms for COMBINE
CMSSW	8.10	[75]	Running COMBINE
COMBINE	7.0	[231]	Extracting significances and limits
SCIKIT-LEARN	0.19	[190]	Cross-validation, pre-processing and ROC construction
MATPLOTLIB	2.1	[305]	Plot production
SEABORN	0.8	[306]	Plot production
TENSORFLOW	1.8	[314]	KERAS back-end
NUMPY	1.14	[308]	Data analysis and computation
PANDAS	0.23	[309]	Data analysis and computation

Table 12.3: Software used for the HL-LHC investigations

Software	Version	References	Use/Notes
LUMIN	0.4.0-0.6.0	[236]	Wrapping PYTORCH to implement networks
PYTORCH	1.0.0-1.6.0	[181]	Implementing neural networks
SEABORN	0.9.0	[306]	Plot production
MATPLOTLIB	3.0.3 & 3.2.2	[305]	Plot production
PANDAS	1.0.5	[309]	Data analysis and computation
NUMPY	1.18.5	[308]	Data analysis and computation
SCIKIT-LEARN	0.23.0	[190]	Cross-validation and pre-processing
RFPIMP	1.3.5	[311]	Random Forest Permutation Importance and mutual dependency calculations
ROOT	6.22.00	[302]	Skimming of data
CMSSW	10.2.25	[75]	Skimming of data
PDPBOX	0.2.0	[315]	Partial dependencies for resonant models
ONNX	1.6.0	[316]	Exporting ONNX to TENSORFLOW
ONNX-TF	1.1	[317]	Exporting ONNX to TENSORFLOW
TENSORFLOW	1.6.0	[314]	Exporting ONNX to TENSORFLOW
CMSSW-DNN	N/A	[318]	Running exported models over data

Table 12.4: Software used for the CMS search