

Efficiency Improvements in Monte Carlo Algorithms for High-Multiplicity Processes

Master-Arbeit
zur Erlangung des Hochschulgrades
Master of Science
im Master-Studiengang Physik

vorgelegt von

Katharina Danziger
geboren am 12.07.1995 in Ebersdorf jetzt Saalburg-Ebersdorf

Institut für Kern- und Teilchenphysik
Fakultät Physik
Bereich Mathematik und Naturwissenschaften
Technische Universität Dresden
2020

Eingereicht am 31. März 2020

1. Gutachter: Dr. Frank Siegert
2. Gutachter: Prof. Dr. Arno Straessner

Abstract

Monte Carlo event generators are essential tools bridging the gap between theoretical predictions and measurements from contemporary collider experiments. Due to their importance and intensive use, it is crucial that event generators provide precise predictions while preserving reasonable computational costs. This thesis presents efficiency improvements for the SHERPA event generator, in particular for high-multiplicity processes. A short introduction to Monte Carlo techniques as well as concepts of perturbative QCD is given to serve as a basis for the following discussions. Subsequently, the idea behind the Machine Learning based unweighting is presented and the optimisation of the utilised Neural Network architecture described. Resulting performance improvements in comparison to the “classic” unweighting prescription are investigated. Significant time improvements are achieved for all examined processes. Finally, the problem of negative weighted events, as well as mechanisms to reduce the negative weight fraction, are discussed with focus on Z and $t\bar{t}$ production in association with jets. In both cases, the amount of negative weighted event can roughly be halved for the investigated process setups.

Kurzdarstellung

Monte Carlo Ereignisgeneratoren sind unentbehrlich für den Vergleich zwischen theoretischen Berechnungen und Messdaten gegenwärtiger Collider Experimente. Aufgrund ihrer Relevanz und intensiven Nutzung ist es essentiell, dass Ereignisgeneratoren genaue Vorhersagen bei gleichzeitig vertretbarem Rechenaufwand liefern. Diese Arbeit präsentiert Effizienzsteigerungen für den SHERPA Ereignisgenerator, insbesondere für Prozesse mit hoher Multiplizität im Endzustand. Eine kurze Einführung zu Monte Carlo Methoden sowie Konzepten der perturbativen QCD dienen als Grundlage für die darauffolgenden Diskussionen. Die grundlegende Idee des auf Maschinellen Lernens basierenden Unweightings und die Optimierung des dafür benutzten Neuralen Netzes werden anschließend beschrieben. Resultierende Leistungssteigerungen im Vergleich zum “klassischem” Unweighting werden untersucht. Für alle untersuchten Prozesse ergeben sich signifikante Zeitverbesserungen. Die Problematik negativ gewichteter Ereignisse sowie Mechanismen zur Reduzierung dieser werden diskutiert, im Speziellen mit Fokus auf Z und $t\bar{t}$ Produktion in Verbindung mit Jets. Für die beiden untersuchten Prozesskonfigurationen kann der Anteil an Ereignissen mit negativem Gewicht ungefähr halbiert werden.

Contents

1	Introduction	7
2	Monte Carlo Techniques	9
2.1	Monte Carlo Integration	9
2.1.1	Importance Sampling	10
2.1.2	Stratified Sampling	11
2.1.3	Multi-Channel Method	11
2.2	Monte Carlo Sampling	12
2.2.1	Hit-or-Miss Method	13
2.2.2	Veto Algorithm	13
2.3	Unweighting	14
2.4	Monte Carlo Event Generators	16
3	Perturbative QCD in Monte Carlo Event Generators	19
3.1	Perturbation Theory and Fixed Order Calculations	19
3.2	Parton Showers	21
3.3	Matching Parton Showers and Fixed-Order Calculations at NLO	23
3.4	Multijet Merging of Matrix Elements and Parton Showers	26
3.5	Combining Matching and Merging	28
4	Improvements of Unweighting Performance	29
4.1	Machine Learning Based Unweighting	29
4.1.1	Introduction to Machine Learning	29
4.1.2	Basic Idea	30
4.1.3	Software and Workflow	34
4.2	Neural Networks	34
4.2.1	Fundamentals of Neural Networks	34
4.2.2	Optimisation of the Neural Network	37
4.2.3	Examination of the Optimised Neural Network	46
4.3	Results	50
4.3.1	Z +jets	50
4.3.2	W +jets	54

4.3.3	$t\bar{t}$ +jets	57
5	Reduction of Negative Weights	61
5.1	Origin and Problem of Negative Weights	61
5.2	Mechanisms to Improve Negative Weight Fraction	62
5.2.1	Leading Colour Approximation	62
5.2.2	$\mathbb{H}$ -Event Shower Interplay	63
5.2.3	Local K-Factor	63
5.3	Results	64
5.3.1	Z +jets	64
5.3.2	$t\bar{t}$ +jets	69
6	Conclusion and Outlook	75
Appendix A	Appendix	77
A.1	Activation Functions	77
	List of Figures	79
	List of Tables	81
	Bibliography	83

1 Introduction

The field of particle physics studies the elementary constituents of matter and the interactions between them. To this date, the Standard Model of particle physics (SM) is the most successful theory describing all known fundamental particles and their non-gravitational interactions.

Many experiments have shown very good agreement with SM predictions, with a highlight being the discovery of the Higgs boson in 2012 [1, 2]. Despite its success, the Standard Model leaves some open questions unanswered and is thus not a complete theory of the fundamental interactions. For instance, it does not incorporate the theory of gravity nor does it contain an explanation for the existence of dark matter. The Nobel prize in physics 2015 was awarded for the discovery of neutrino oscillations [3, 4]. However, this is only possible if neutrinos carry mass contradicting the Standard Model which predicts them to be massless.

These and other observations are the motivation for searches of new physics beyond the Standard Model. One approach to study the properties of elementary particles and their interactions are particle colliders, where particles are accelerated to very high kinetic energies and brought to collision. As a result of the mass-energy equivalence, those accelerators enable the production of heavy and possibly unknown particles, either by their direct measurement or through the observation of deviations of known interactions measured at high precision. The most famous example of such particle colliders is the Large Hadron Collider (LHC) at CERN near Geneva [5]. The LHC is a circular collider with a circumference of 27 km primarily colliding proton beams accelerated to near the speed of light. With a centre-of-mass energy of 13 TeV, first reached in 2015 during its second operational run, the LHC holds the present world record.

Probing the Standard Model and the search for new physics necessitate the comparison of experimental measurements and theoretical predictions. Due to the complexity of such particle collisions, which involve a large number of final-state particles as well as non-trivial interactions according to the theory of strong interactions, Quantum Chromodynamics (QCD), the method of choice are computer simulation programs employing Monte Carlo methods. These so-called Monte Carlo event generators are computational frameworks incorporating theoretical models in order to simulate particle collisions. The resulting final-states, called events, are simulated at a level as seen by particle detectors. They can subsequently be fed into dedicated detector simulations. Hence, event generators are essential tools used both by experiments and

theorists. They enable the confrontation of measurements with generated data, allowing for tests of predictions made by theoretical models as well as the interpretation of experimental results. In addition, they can be used to simulate possible backgrounds and study detector requirements, as well as determine necessary corrections to data. There are various examples of available general-purpose Monte Carlo event generators, of which PYTHIA [6], HERWIG [7, 8] and SHERPA [9, 10] are the most prominent ones.

The importance and intense use of Monte Carlo event generators from experiments implicate high demands in terms of precision at reasonable computing performance. However, the ongoing efforts and progress for more accurate predictions impose computational challenges. Improving the efficiency of the SHERPA event generator, especially for high-multiplicity processes, is the subject of this work.

This thesis starts with an introduction to Monte Carlo methods in Chapter 2, followed by a brief discussion on aspects of perturbative QCD in Chapter 3, which both build the theoretical frame of this work. In particular, Monte Carlo integration and sampling techniques are described and Monte Carlo event generators are discussed in some more detail. Furthermore, fixed-order calculations, the concept of parton shower simulations as well as matching and merging techniques, which combine the former two, are reviewed.

The subsequent Chapter 4 introduces the idea of the so-called Machine Learning based unweighting, aiming for performance improvements with regard to the unweighted event generation. The optimisation procedure of the utilised Neural Network architecture is described and its performance investigated. Lastly, results for processes belonging to the $W/Z+4$ jets and $t\bar{t}+3$ jets groups are presented.

Finally, Chapter 5 deals with the problem of negative weighted events, which naturally arise when going beyond the leading order in the perturbative expansion. They spoil the statistical accuracy of event samples in comparison to only positive weighted events, making their reduction desirable. Mechanisms to decrease the negative weight fraction are reviewed and results are presented for the production of Z as well as $t\bar{t}$ in association with jets.

2 Monte Carlo Techniques

The aim of this chapter is to introduce some of the Monte Carlo techniques commonly used in phenomenological particle physics. Broadly speaking, Monte Carlo methods are computational algorithms exploiting the concept of randomness. Starting with the basic idea behind Monte Carlo integration along with some variance reduction techniques, the sampling according to a given distribution using Monte Carlo methods is discussed. Finally, Monte Carlo event generators are introduced – the indispensable tools of particle physics, which form the basis of this thesis. They utilise the above methods in order to integrate cross-sections and simulate high-energy particle collision events, enabling the crucial confrontation between theoretical predictions and experimental measurements.

2.1 Monte Carlo Integration

The calculation of particle collision cross-sections involves the integration over the momenta of all final-state particles. The dimensionality d of this phase-space integral is set by the number n of particles in the final-state, namely $d = 3n - 4$. Here, the constraints due to four-momentum conservation and on-shell conditions for all outgoing particles are already included. Traditional numerical quadrature methods are insufficient to perform this type of integration as they will not converge fast enough for large n . Table 2.1 shows a comparison of the convergence of different numerical integration techniques in d dimensions. As can be seen, the error estimate of the Monte Carlo technique scales like $\mathcal{O}(N^{-1/2})$ independent of the dimensionality of the problem, making it the method of choice for high-dimensional numerical integration.

The basic idea of the Monte Carlo approach is the approximation of the integral I of $f(\vec{x})$ by uniformly sampling its integration domain Ω [11, 12]

$$\int_{\Omega} d\vec{x} f(\vec{x}) = I \approx I_N = V \cdot \frac{1}{N} \sum_{i=1}^N f(\vec{x}_i) = V \cdot \langle f \rangle_x. \quad (2.1)$$

Here, V is the volume encompassed by Ω and $\langle f \rangle_x$ denotes the average of f with respect to randomised, uniformly distributed $\vec{x}_i = (x_i^1, x_i^2, \dots, x_i^d) \in \Omega$. N is the number of sample points. The law of large numbers ensures that I_N is a consistent estimator¹ of I as it converges to the

¹The function f must be integrable and is generally required to be finite everywhere and at least piecewise continuous [11].

exact value of the integral for $N \rightarrow \infty$. We define the weights $w_i := V \cdot f(\vec{x}_i)$, yielding

$$I \approx I_N = \frac{1}{N} \sum_{i=1}^N w_i. \quad (2.2)$$

Thus, the integral is given by the average of the weights w_i . I_N follows a Gaussian distribution as a result of the central limit theorem. The corresponding standard deviation of the Monte Carlo estimate is therefore given by

$$\sigma_I = V \sqrt{\frac{\langle f^2 \rangle_x - \langle f \rangle_x^2}{N-1}} = V \sqrt{\frac{\sigma^2(f)}{N-1}}. \quad (2.3)$$

At this, $\sigma^2(f)$ denotes the variance of f .

While independent of d , an error scaling of $1/\sqrt{N}$ is still not a good convergence. Various methods were developed in order to accelerate the error reduction by reducing the variance. In the following, the most prominent ones, namely importance sampling, stratified sampling and the multi-channel method, are discussed in more detail.

Numerical Integration Method	Error Scaling
Trapezium rule	$\propto 1/N^{2/d}$
Simpson's rule	$\propto 1/N^{4/d}$
Monte Carlo	$\propto 1/N^{1/2}$

Table 2.1: Comparison of the convergence of different numerical integration techniques in d dimensions. Here, N is the number of evaluation points.

2.1.1 Importance Sampling

The uncertainty of the Monte Carlo estimate will be sizeable if the integrand $f(\vec{x})$ has large fluctuations. Calculating the cross-section of a process involves the integration over the absolute squared matrix element, which exhibits sharp peaks in various different regions of the phase-space. Importance sampling addresses this problem by sampling from a well-known distribution $g(\vec{x})$ with a similar shape as $f(\vec{x})$ instead of uniformly sampling the integration domain. With this method, we effectively try to draw more samples in regions where $f(\vec{x})$ has a higher contribution (“important” regions). Mathematically, importance sampling utilises the relation $d\vec{x} = dG(\vec{x})/g(\vec{x})$, which corresponds to a change of integration variables. Hence,

$$V \cdot \langle f \rangle_x = \int_{\Omega} d\vec{x} f(\vec{x}) = \int_{\Omega} d\vec{x} \frac{f(\vec{x})}{g(\vec{x})} g(\vec{x}) = \int_{\Omega} dG(\vec{x}) \frac{f(\vec{x})}{g(\vec{x})} = V \cdot \langle f/g \rangle_G, \quad (2.4)$$

where $G(\vec{x})$ is the primitive of $g(\vec{x})$. The relevant variance is now $\sigma^2(f/g)$. In particle physics, this corresponds to using a mapping of random numbers to four-momenta, which should capture the distribution of the squared matrix element for the process of interest as far as possible. By choosing $g(\vec{x})$ to resemble $f(\vec{x})$ in shape, the variance can consequently be significantly reduced. The aim is to find such a $g(\vec{x})$, which must be integrable and invertible in order to allow for fast sampling. Furthermore, $g(\vec{x})$ has to be a density, i.e. be non-negative in Ω and normalised to integrate to unity over the sampling space.

2.1.2 Stratified Sampling

The general idea behind stratified sampling is to divide the full integration region Ω into M sub-domains Ω_j , called stratum, with

$$\bigcup_{j=1}^M \Omega_j = \Omega, \quad (2.5)$$

and performing a Monte Carlo integration in each sub-domain. The integral is then given by the sum over the partial results

$$V \cdot \langle f \rangle_x = \sum_{j=1}^M V_j \langle f \rangle_{x \in \Omega_j}. \quad (2.6)$$

The overall variance is just the sum over the variances of the individual pieces, which will be minimized by defining the domains dynamically, such that each contributes equally to the total variance. This can be achieved in an adaptive way by introducing a different a priori probability g_j for picking a sub-domain Ω_j to generate a phase-space point in it and, after a sufficiently large number of function calls, updating them. After the optimisation step, regions with a smaller variance, i.e. less fluctuations of $f(\vec{x})$, will have a smaller g_j and will therefore have fewer sampling points. On the other hand, domains with a high variance, that is with more fluctuations, will obtain a larger g_j and an increased number of points sampling the phase-space in this sub-domain. Theoretically, the best scenario is to have a priori weights g_j that behave like the variance in each domain. However, it should be noted, that this method can also lead to a larger variance if the choice is not appropriate.

2.1.3 Multi-Channel Method

As mentioned above, crude Monte Carlo integration will usually lead to poor results if the integrand $f(\vec{x})$ has sharp peaks. This can be improved by flattening the integrand by using a remapping of variables as shown in Section 2.1.1. However, it is often not possible to find a suitable variable transformation that simultaneously remaps all peaks if the integrand exhibits

different resonances in various regions of the phase-space. One solution to this problem is the multi-channel method, first presented in the scope of particle physics in [13]. The underlying idea is to construct a mapping function $g(\vec{x})$ as a weighted sum of m individual mappings $g_i(\vec{x})$, each capturing a single peak structure:

$$g(\vec{x}) = \sum_{i=1}^m \alpha_i \cdot g_i(\vec{x}) \quad \text{with} \quad \sum_{i=1}^m \alpha_i = 1. \quad (2.7)$$

These single transformations are called channels and the α_i the corresponding channel weights. A specific channel is then selected with probability α_i . Thus, the integral is given by

$$I = \int_{\Omega} d\vec{x} f(\vec{x}) = \sum_{i=1}^m \alpha_i \int_{\Omega} dG_i(\vec{x}) \frac{f(\vec{x})}{g(\vec{x})} \approx \frac{V}{N} \sum_{i=1}^m \sum_{n_i=1}^{N_i} \frac{f(\vec{x}_{n_i})}{g(\vec{x}_{n_i})} = I_N, \quad (2.8)$$

with $N_i = \alpha_i \cdot N$. The integral estimate I_N is independent of the choice of the α_i , while the integration error is not. Therefore, choosing the channel weights to match their relevance in the structure of $f(\vec{x})$, one can often significantly reduce the error of the Monte Carlo estimate. More details on multichanneling and how to choose the α_i can e.g. be found in [13] and [14].

Technically, the multi-channel method is a combination of the traditional Monte Carlo integration methods importance and stratified sampling. Namely, it performs a stratified sampling over various importance samplings. In particle physics, the form of the matrix element is known. The single phase-space mappings g_i can be constructed with the help of the individual Feynman diagrams, each representing a resonance driven by its propagator term.

The work of this thesis is done within the Monte Carlo event generator SHERPA [10] which employs the adaptive multi-channel method of [13, 15] and, by default, is used together with a VEGAS [16] optimisation to calculate initial-state and final-state integrals.

2.2 Monte Carlo Sampling

In the previous section, the Monte Carlo approach was introduced in order to deal with the evaluation of high-dimensional integrals as needed for the calculation of cross-sections. A further advantage of the Monte Carlo method is the possibility to generate events, since each phase-space point corresponds to a set of momenta of the involved particles. With the help of these simulated events, the distribution of observables can be histogrammed or the generated events can be passed to a dedicated detector simulation in order to study its response and to estimate signals and backgrounds in high-energy processes.

There are various methods to sample according to a function $f(x)$ [14, 17]. In particle physics, the function of interest might for example be a differential cross-section or a fragmentation

function. If $f(x)$ is non-negative in the range of interest $[x_{min}, x_{max}]$ and has an invertible primitive $F(x)$, an x can be selected using direct sampling

$$\begin{aligned} \int_{x_{min}}^x dx' f(x') &= R \int_{x_{min}}^{x_{max}} dx' f(x') \\ \Rightarrow x &= F^{-1}\left(F(x_{min}) + R(F(x_{max}) - F(x_{min}))\right), \end{aligned} \quad (2.9)$$

where $R \in [0, 1]$ is an uniformly distributed random number. However, the function $f(x)$ is usually not that simple, requiring the usage of more complex procedures.

2.2.1 Hit-or-Miss Method

In the case that the maximum of $f(x)$ is known, $f(x) \leq f_{max} \quad \forall x \in [x_{min}, x_{max}]$, a hit-or-miss is a failsafe sampling method that will always give the correct answer. The algorithm works as follows:

1. Select an $x = x_{min} + R(x_{max} - x_{min})$, where $R \in [0, 1]$ is uniformly distributed
2. Generate (another) uniformly distributed random number $R' \in [0, 1]$
3. If $f(x) > f_{max} \cdot R'$, then retain x as final answer
4. Otherwise reject the x and return to 1. for a new try

The drawback of this method is the low sampling efficiency, if the function $f(x)$ fluctuates too wildly. Further methods, aiming at an improved sampling performance, can for instance be found in [11].

2.2.2 Veto Algorithm

The sampling according to a Poisson distribution requires further considerations. First consider the simple example of a radioactive decay. There, the probability that something happens at a scale t , i.e. that a nucleus decays at time t , is given by

$$\mathcal{P}(t) = -\frac{dN(t)}{dt} = \lambda N(t), \quad (2.10)$$

where λ denotes the decay constant and N the number of nuclei. Assuming the process to start at time $t_0 = 0$ with $N(t_0) = 1$, the solution is determined by

$$\mathcal{P}(t) = \lambda \exp\{-\lambda t\}. \quad (2.11)$$

Now adding an explicit time dependence, i.e. $\lambda = f(t)$, Equation (2.11) transforms into

$$\mathcal{P}(t, t') = f(t) \exp\left\{-\int_t^{t'} d\bar{t} f(\bar{t})\right\}. \quad (2.12)$$

In the case that $f(t)$ has a primitive function $F(t)$ with known inverse, t values can be selected correctly in a straightforward way, namely $t = F^{-1}(F(t') + \ln(R))$, where R is an uniformly distributed random number between zero and one. However, if $f(t)$ is not sufficiently nice behaved, one may attempt to find a simpler function $g(t)$, with $f(t) \leq g(t)$ for all t as well as known and invertible primitive $G(t)$. Just picking values according to $g(t)$ and accepting them with probability $f(t)/g(t)$ would yield the wrong result, since the effects of the exponential term in $\mathcal{P}(t)$ are not accounted for correctly. A solution to this problem is an extension of the hit-or-miss Monte Carlo to Poisson distributions, called *veto algorithm* [17, 18]:

1. Initialise with $i = 0$ and $t = t_0$
2. Increment i by one and select new $t_i = G^{-1}(G(t_{i-1}) - \ln(R))$, i.e. according to $g(t)$ with constraint $t_i > t_{i-1}$
3. Pick (new) uniformly distributed random number $R \in [0, 1]$
4. If $f(t_i)/g(t_i) > R$, proceed, otherwise return to 2.
5. Retain last generated t_i as final answer

With that, the probability that something happens at scale t with n intermediate rejections is given by

$$\begin{aligned} \mathcal{P}_n(t, t') &= \frac{f(t)}{g(t)} g(t) \exp \left\{ - \int_t^{t_1} d\bar{t} g(\bar{t}) \right\} \\ &\times \prod_{i=1}^n \left[\int_{t_{i-1}}^{t_{i+1}} dt_i \left(1 - \frac{f(t_i)}{g(t_i)} \right) g(t_i) \exp \left\{ - \int_{t_i}^{t_{i+1}} d\bar{t} g(\bar{t}) \right\} \right], \end{aligned} \quad (2.13)$$

where $t_{n+1} = t'$ and $t_0 = t$. Disentangling the nested integrals in Equation (2.13) and summing over n results in the exponentiation of $g(t) - f(t)$ and thus Equation (2.12) is recovered [17] as desired.

The veto algorithm is important for the construction of parton showers by means of a Monte Carlo algorithm, which will be discussed in more detail in Section 3.2.

2.3 Unweighting

As already mentioned in Section 2.1.3, SHERPA utilises an adaptive multi-channel method to compute cross-sections. At the same time, this integrator is nothing else than a phase-space generator. Thus, the generated phase-space points can also be used to produce events. The problem, however, is that these points will not be generated in accordance with the physical phase-space measure but according to the adaptive functions of the multi-channel technique. As a result, each event will need to obtain a weight w , given by the absolute squared matrix

element $|\mathcal{M}|^2$ times a phase-space weight $\mathcal{PS}$

$$w = |\mathcal{M}|^2 \cdot \mathcal{PS}, \quad (2.14)$$

corresponding to the underlying probability distribution. Following this, we effectively produce so-called *weighted events*.

The generation of weighted events is undesirable from a statistical point of view. There may be a multitude of events with small weights, which is equivalent to a low likelihood to occur, resulting in large samples being required in order to receive a reasonable statistical accuracy. While this may not be a major issue in terms of CPU time on the event generation side, these events will have to enter a dedicated detector simulation to reconstruct the interactions with detector material, which is the driving factor with regard to computational costs. Furthermore, producing events pursuant to the underlying physical distribution will not only result in smaller sample sizes, but also give a realistic representation of actual physical events.

The procedure to recover the desired underlying physics distribution from weighted events is called *unweighting*. The simple idea is to generate weighted events and accept them with a probability w/w_{max} , where the maximal weight w_{max} is determined during the adaptive optimisation process of the integration phase. All accepted events then obtain an unit weight. Events with small weights are more likely to be rejected than those with large weights and vice versa, such that the probability for events in certain regions of the phase-space is directly related to the number of events in this region.

A huge drawback of this procedure are low unweighting efficiencies, given by

$$\varepsilon = \frac{\bar{w}}{w_{max}}, \quad (2.15)$$

where $\bar{w}$ is the mean weight, especially when going to higher multiplicities in the final-state or higher orders in the perturbative expansion, as a consequence of difficulties to sample the underlying physics distribution. The unweighting efficiency signals how many events need to be generated on average to obtain one unweighted one. This efficiency could be improved by utilising an appropriate envelope $g(\vec{x})$ of the weight distribution $w(\vec{x})$ as a function of the respective phase-space point $\vec{x}$. However, the distribution $w(\vec{x})$ is too complex to find a suitable $g(\vec{w})$ due to its high-dimensional dependencies, such that usually a constant maximum w_{max} is employed.

Figure 2.1 shows the event weight distributions for processes producing Z plus up to four jets generated with the Monte Carlo event generator SHERPA. The difficulty to approximate the underlying matrix element when going to higher multiplicities is reflected by broadening weight distributions towards lower weights with more particles in the final-state. On top, the

evaluation of the event weights gets increasingly expensive in terms of CPU time, leading to severe performance losses, which is a huge challenge encountered in current event generators.

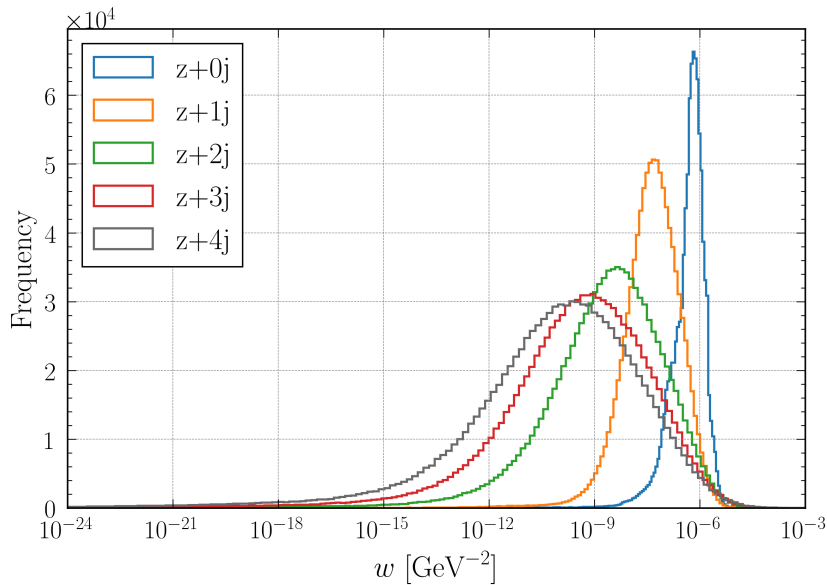


Figure 2.1: Weight distributions for one million events for the production of Z plus jets at leading order generated with the Monte Carlo event generator SHERPA.

2.4 Monte Carlo Event Generators

Monte Carlo event generators are indispensable devices in the field of particle physics. They build the connection between experimental measurements and theoretical predictions in collider experiments, enabling for precision tests of the Standard Model and the search for new physics. Moreover, the tuning and design of detectors as well as the development of analysis strategies rely on the exploitation of simulated events produced by Monte Carlo event generators.

The simulation of high-energy particle collisions is an extremely challenging task. Each collision leads to complex final-states involving a large number of particles. At present, these calculations are mostly done for proton-proton collisions as investigated at the LHC. The colliding particles are partons while experimentally hadrons, their colourless bound states, are measured. This requires the modelling of the transition from the perturbative to the non-perturbative regime of QCD, involving phenomenological fragmentation models which need to be adjusted by comparison to data.

The basic idea of Monte Carlo event generators to deal with the above mentioned theoretical challenges is to factorise particle scattering processes into several well-defined event stages, each corresponding to a different kinematic regime. Figure 2.2 illustrates this exemplary for the production of a Higgs boson in association with a top quark pair. Starting with the

hard-scattering of incoming partons stemming from the colliding protons, highly energetic and heavy particles may be produced by means of appropriate matrix elements, which are exact to some order in perturbation theory. The heavy resonances with narrow widths, such as top quarks, then decay. Going to lower energetic scales, the incoming hadrons as well as the outgoing particles radiate lighter particles, like gluons and photons, producing initial- and final-state radiation, respectively. The emitted particles tend to be collinear and are softer than their emitters. Likewise, they are able to radiate further, producing a cascade of particles which is realised through a parton shower algorithm. Parton showers approximately model QCD Bremsstrahlung to exact perturbation theory, which is accurate to leading logarithmic order. These emissions start at comparatively high energetic scales and evolve down to the hadronization scale, which is of the order of a few Λ_{QCD} . The subsequently formed hadrons might be unstable and decay further into practically stable hadrons that would finally reach the detector. For hadronic initial-states, an additional difficulty arises. There, remnants of the incoming hadrons may undergo secondary interactions, called underlying event, contributing substantially to the final-state multiplicity. While being of great importance, such effects are beyond QCD factorisation theorems and thus again require the usage of phenomenological models that rely on parameter tuning with the help of experimental data.

Due to the great relevance of such Monte Carlo event generators, it is of utter importance that they perform well in terms of precision while retaining decent computational costs. This gets increasingly difficult when going to higher multiplicities in the hard-scattering and higher orders in the perturbative expansion. Such efficiency problems are the concern of this thesis and are dealt within the SHERPA event generator.

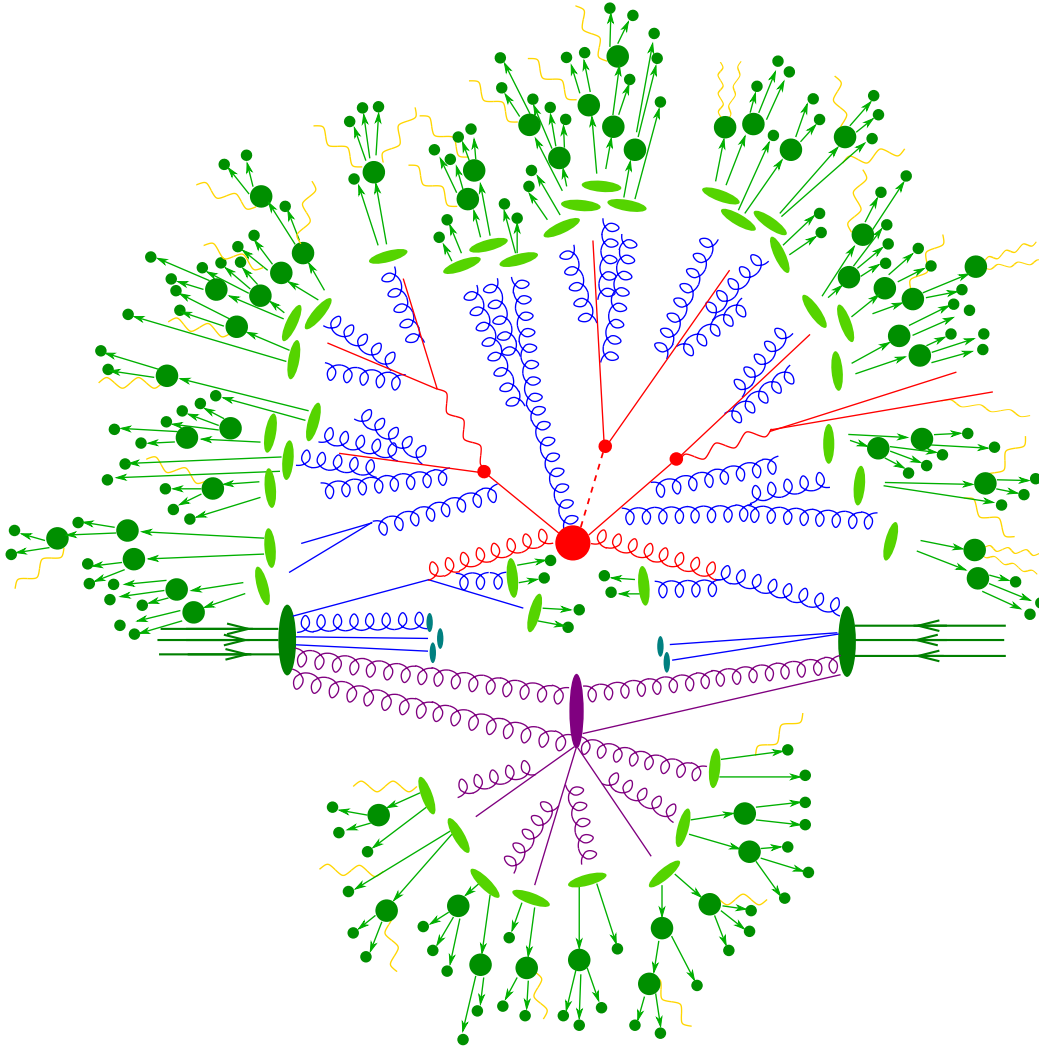


Figure 2.2: Illustration of an event as produced by a Monte Carlo event generator. Depicted is the production of a Higgs boson in association with a top and anti-top quark. The big red blob represents the hard scattering, followed by the decay of the top quark pair and the Higgs portrayed by the smaller red blobs. Additional hard QCD radiation is produced, illustrated in blue, and a secondary interaction occurs (purple). At lower energies, the partons hadronize (light green). The produced hadrons might be unstable and thus decay further, depicted in dark green. The photon radiation portrayed in yellow can happen at any stage.

3 Perturbative QCD in Monte Carlo Event Generators

With the need and importance of Monte Carlo event generators reviewed in the previous chapter, the focus hereafter is on a brief discussion on the physics implemented within them, as needed within the scope of this thesis. Precise predictions are essential for the assessment of experimental measurements and their comparison with theoretical calculations, for instance with regard to potential deviations of the Standard Model. In the context of proton-proton colliders such as the LHC, the theory of strong interactions is of particular interest. This chapter will, therefore, focus on concepts of perturbative QCD which will build the foundation for the discussion on the reduction of negative weighted events in Chapter 5.

Starting with a brief overview of fixed-order calculations, the underlying idea of parton shower simulations will be discussed. Finally, matching and merging techniques will be reviewed, which combine the two former concepts and aim at improving the perturbative description of Monte Carlo event generators.

3.1 Perturbation Theory and Fixed Order Calculations

The perturbative approach to quantum field theory, more specifically the order-by-order assessment of quantum mechanical transition matrix elements, is an essential tool to make predictions for high energetic particle collisions. The contributions to transition amplitudes at a fixed perturbative order can be represented pictorially by Feynman diagrams, encoding mathematical prescriptions to evaluate them. The explicit computation gets increasingly complex with a growing number of external legs and internal loops and necessitates the usage of computer codes which automate their evaluation.

The calculation of hard scattering processes can be decoupled into the high-energy partonic interactions and the low energetic evolution of each initial hadron using collinear factorisation¹ [19, 20, 21]. With that, the inclusive cross-section for the collision of two hadrons h_1 and h_2

¹Collinear factorisation has only been proven for Drell-Yan production as well as deep-inelastic scattering [12]. Its application to other processes is not based on a mathematical proof, but is only justified by its successful description.

leading to an n particle final-state is given by

$$\sigma_{h_1, h_2 \rightarrow n} = \sum_{a,b} \int dx_a \int dx_b f_a^{h_1}(x_a, \mu_F) f_b^{h_2}(x_b, \mu_F) \hat{\sigma}_{ab \rightarrow n}(\mu_F, \mu_R), \quad (3.1)$$

where a and b denote the colliding partons. In this context, the $f_p^{h_i}$ are called *parton distribution functions* (PDF). At leading order, they can be interpreted as probabilities to find a parton p with momentum fraction x_p inside the hadron h_i at a resolution scale μ_F . The renormalisation scale μ_R and the factorisation scale μ_F are process-dependent quantities, where the former signifies the scale at which the coupling constants are evaluated. For simple processes, both scales are typically assumed to be equal.

On the other hand, the partonic cross-section $\hat{\sigma}_{ab \rightarrow n}$ is evaluated as the integral of the absolute squared matrix element $|\mathcal{M}_{ab \rightarrow n}|^2$ over the available n -particle phase-space

$$\hat{\sigma}_{ab \rightarrow n} = \int d\hat{\sigma}_{ab \rightarrow n} = \int \left[\prod_{i=1}^n \frac{d^3 \vec{p}_i}{(2\pi)^3 2E_i} \right] \delta^4 \left(p_a + p_b - \sum_{i=1}^n p_i \right) |\mathcal{M}_{ab \rightarrow n}|^2. \quad (3.2)$$

Four-momentum conservation is encoded within the delta distribution. The partonic cross-section can be made accessible by perturbative expansion in the strong coupling constant α_S , using the fact that α_S is small in high-energy or short distance interactions, yielding

$$\hat{\sigma} = \alpha_S^k \sum_{n=0}^{\infty} \left(\frac{\alpha_S}{\pi} \right)^n \hat{\sigma}^n = \hat{\sigma}^{\text{LO}} + \frac{\alpha_S}{\pi} \hat{\sigma}^{\text{NLO}} + \left(\frac{\alpha_S}{\pi} \right)^2 \hat{\sigma}^{\text{NNLO}} + \dots \quad (3.3)$$

Different hard processes will contribute with distinct leading powers k in the coupling constant. For instance, in the case of vector boson production k is equal to zero, while for jet or heavy quark production $k = 2$. While the leading order term (LO) gives a first approximation of the total cross-section, the inclusion of higher order corrections, i.e. the next-to-leading order (NLO), next-next-to-leading order (NNLO) term and so forth, adds to the convergence of the result. The determination of higher order contributions gets increasingly difficult and cumbersome. Some recent state-of-the-art results accomplish up to N³LO accuracy in QCD [22, 23, 24] and NNLO in the electroweak (EW) sector [25].

Exemplary Feynman diagrams contributing at leading and next-to-leading order QCD to quark-antiquark scattering are shown in Figure 3.1. At NLO, not only the leading order matrix element, sometimes called Born-level term, but also virtual and real corrections contribute. The former is given by adding a closed loop to the Born term, while the latter is obtained by attaching an additional outgoing particle, i.e. an extra leg. It should be stressed that the overall virtual correction is given by multiplication of the Born-level and the virtual correction, whereas the real correction is derived from the absolute square of matrix elements with one

additional emitted parton.

Equation (3.2) only specifies the total cross-section for the final-state n and is therefore called the inclusive cross-section. Any number of particles may emerge in addition to n . In fact, the particles involved in the hard-scattering process are forcefully accelerated, leading to additional quark and gluon radiation off coloured particles, which themselves carry colour charges and therefore emit further. Principally, exclusive cross-sections can be accessed by demanding a specific number of additional particles or restricting the available phase-space, or both. In event generators, an inclusive cross-section is produced according to Equation (3.2) and made exclusive using a Markov chain Monte Carlo algorithm, known as parton shower, supplementing the process with additional particles. This finally results in a high-multiplicity final-state, while respecting the inclusivity condition of the original production of n .

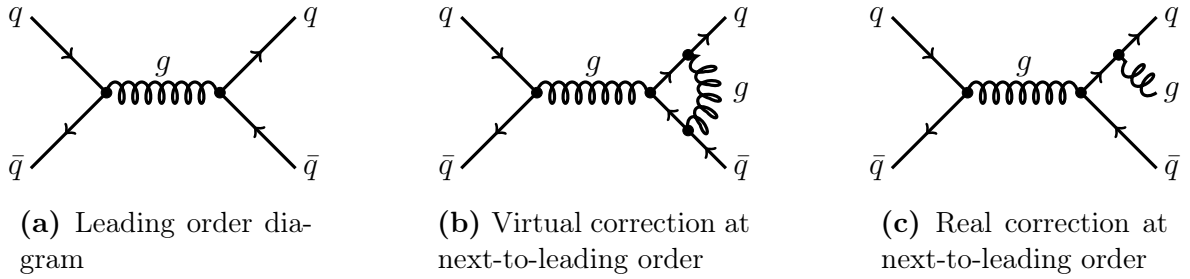


Figure 3.1: *Feynman diagrams contributing to quark-antiquark scattering at LO and NLO.*

3.2 Parton Showers

The task of parton showers is to approximate higher-order real emission corrections to the hard scattering process. They are numerical tools generating emissions with gradually lower energy off the initial- and final-state partons down to the validity region of perturbative QCD. Substantially, they describe the evolution of partons from the scale of the hard process down to a cut-off value, which is usually chosen to be the scale associated with the fragmentation of partons into hadrons. The emissions are independent of each other and modelled in an iterative way as a Markov chain. The parton shower approximation includes the dominant contributions in each order, which are given by collinear and soft parton splittings, which factorise off of the Born matrix element in their respective limits. Furthermore, they are based on a leading colour approximation (large N_c -limit²), such that a colour flow can be assigned to each parton configuration, effectively neglecting colour correlations.

²Technically, QCD is a non-abelian gauge theory with $SU(3)$ gauge symmetry. In terms of group theory, the large N_c -limit consists of replacing the colour-octet gluons by a colour triplet-antitriplet pair, since $1/N_c$ terms are omitted, whereas $N_c = 3$ is the number of colours. This leads to a simple colour topology, amounting to a planar flow. Hence, each gluon is colour connected to only two and each quark to only one other parton.

The construction of parton showers can be done with the aid of *Sudakov form factors*, defined as

$$\begin{aligned}\Delta^{(\text{PS})}(t, t') &= \exp \left\{ - \int_t^{t'} \frac{d\bar{t}}{\bar{t}} \int dz \int \frac{d\phi}{2\pi} J(\bar{t}, z, \phi) \frac{\alpha_S}{2\pi} K(\bar{t}, z, \phi) \right\} \\ &= \exp \left\{ - \int_t^{t'} d\Phi_1 \frac{\alpha_S}{2\pi} K(\Phi_1) \right\},\end{aligned}\tag{3.4}$$

which can be interpreted as the probability of a parton to not undergo a branching between two scales t and t' . Here, ϕ is the azimuthal angle, z the splitting and t the evolution variable. The sequential emissions are ordered in terms of the parameter t , which is typically related to the opening angle, virtuality or transverse momentum of the splitting, dependent on the explicit shower implementation. These three variables together define the phase-space of the extra emission Φ_1 , where J is the Jacobian factor stemming from the relevant phase-space mappings. The kinematic details of the branching are encoded in the splitting kernel K .

In order to generate emissions according to this probability distribution, the veto algorithm as described in Section 2.2.2 is employed. In accordance with the notation established there, $f(t)$ corresponds to the splitting kernel K , integrated over the splitting variable z .

With that, the leading order cross-section in the parton-shower approximation to first order in α_S is given as

$$\sigma_{\text{PS}}^{\text{LO}} = \int d\Phi_B B(\Phi_B) \left[\Delta^{(\text{PS})}(t_c, \mu_Q^2) + \int_{t_c}^{\mu_Q^2} d\Phi_1 \frac{\alpha_S}{2\pi} K(\Phi_1) \Delta^{(\text{PS})}(t, \mu_Q^2) \right].\tag{3.5}$$

Here and in the following, μ_Q denotes the parton shower start scale, defining the upper limit of the shower evolution variable t , while t_c is the cut-off scale specifying the transition to the non-perturbative region. $B(\Phi_B)$ is the absolute squared leading-order matrix element, incorporating flux and symmetry factors, whereas $d\Phi_B$ means the corresponding Born phase-space element. While the first term in the square bracket in Equation (3.5) describes the probability of no further emission to occur above t_c , the latter produces one additional branching at t . Parton showers respect unitarity, which is reflected by the fact that the square bracket altogether integrates to one, leaving the total cross-section of an event sample unaffected, thus

$$\int d\sigma_{\text{PS}}^{\text{LO}} = \int d\sigma^{\text{LO}} = \int d\Phi_B B(\Phi_B).\tag{3.6}$$

In other words, a parton may split or not, both probabilities adding up to one. Virtual corrections are taken into account by the parton shower approximation as well, albeit more implicitly through the no-branching probability between two scales, given by the Sudakov form

factor. Hence, the shower resums the leading logarithmic terms to all orders, which originate from virtual and unresolvable diagrams, as well as the corresponding logarithms stemming from resolvable emissions. These logarithmically enhanced terms spoil the convergence of the perturbative expansion such that any fixed-order calculation on its own will not give correct results, necessitating either analytical or numerical resummation techniques to complement the fixed-order results, whereas the latter can be achieved by means of a parton shower.

More detailed discussions on parton shower algorithms can be found in the literature, for instance in [12, 17, 26, 27].

3.3 Matching Parton Showers and Fixed-Order Calculations at NLO

The previous sections briefly introduced both fixed-order calculations as well as the concept of parton showers. As outlined above, the interplay between leading order matrix elements and the parton shower is trivial. In this simple case no overlap in the respective radiation patterns occurs and the shower algorithm can simply be attached on top of the LO result with a starting scale related to the hard scattering. When going to NLO in order to improve the perturbative description, the situation gets more complicated.

The main idea behind next-to-leading order matching is the consistent combination of parton shower simulations with fixed-order NLO results. Here, we have to address the problem of double-counting of terms present both in the real NLO emission and the parton shower calculation, while preserving the inclusive NLO accuracy. The two major approaches to tackle this challenge are the MC@NLO [28] and the POWHEG [29] method, respectively. In the following, the focus will be on a variant of the MC@NLO matching prescription, called S-MC@NLO [30], as implemented in SHERPA.

First of all, NLO calculations involve (infrared) divergent structures both in the virtual as well as the real correction terms, however, these divergences cancel each other precisely due to the KLN theorem [31, 32]. Yet, both contributions have separately unbounded weight distributions, which can not be calculated numerically within Monte Carlo event generators. By isolating the divergent structures, subtracting these dipoles $D^{(S)}$ from the real emission matrix element R and adding the integrated subtraction terms $I^{(S)}$ back to the Born-like contribution, one obtains two separately finite terms. With that, the differential dipole-Subtracted cross-section at NLO can be written as

$$d\sigma^{\text{NLO}} = d\Phi_B \left[B(\Phi_B) + \tilde{V}(\Phi_B) + I^{(S)}(\Phi_B) \right] + d\Phi_R \left[R(\Phi_R) - D^{(S)}(\Phi_R) \right], \quad (3.7)$$

where B and $\tilde{V}$ are the matrix elements for the Born and renormalised virtual contribution, while $d\Phi_B$, $d\Phi_R$ denote the phase-space element for the Born and real configuration, respectively. The Born and virtual term cover the same phase-space, while the real emission phase-space is enlarged by one additional emitted particle, $\Phi_R = \Phi_B \otimes \Phi_1$.

As mentioned above, applying a parton shower resummation on top of an NLO calculation results in the problem that both the real emission term as well as the parton shower cover the same phase-space, leading to an unwanted double-counting if employed without correction. Figure 3.2 pictorially illustrates this issue. The basic idea to deal with this problem is to subtract the parton shower contributions from R and, in order to preserve the shower unitarity, re-add the “integrated” parton shower contribution with Born kinematics, effectively performing an Additional subtraction with subtraction terms $D^{(\mathcal{A})}$. With that, Equation (3.7) becomes

$$\begin{aligned} d\sigma^{\text{NLO}} = & d\Phi_B \left[B(\Phi_B) + \tilde{V}(\Phi_B) + I^{(\mathcal{S})}(\Phi_B) \right] + d\Phi_R \left[D^{(\mathcal{A})}(\Phi_R) - D^{(\mathcal{S})}(\Phi_R) \right] \\ & + d\Phi_R \left[R(\Phi_R) - D^{(\mathcal{A})}(\Phi_R) \right]. \end{aligned} \quad (3.8)$$

In general, it is not possible to subtract all soft divergences since the parton shower is based on a leading colour approximation, therefore missing soft singularities in sub-leading colour configurations. Hence, taking the usual shower kernels to define $D^{(\mathcal{A})}$ would lead to non-finite integration results.

In the S-MC@NLO prescription, $D^{(\mathcal{A})}$ is therefore directly set to the dipole terms $D^{(\mathcal{S})}$ in the soft and collinear region and zero everywhere else [33]:

$$D^{(\mathcal{A})} = D^{(\mathcal{S})} \Theta(\mu_Q^2 - t). \quad (3.9)$$

Now applying the parton shower resummation using $D^{(\mathcal{A})}$ as splitting kernels yields

$$\begin{aligned} \sigma_{\text{MC@NLO}} = & \int d\Phi_B \bar{B}^{(\mathcal{A})}(\Phi_B) \left[\underbrace{\bar{\Delta}^{(\mathcal{A})}(t_c, \mu_Q^2)}_{\text{unresolved}} + \underbrace{\int_{t_c}^{\mu_Q^2} d\Phi_1 \frac{D^{(\mathcal{A})}(\Phi_B, \Phi_1)}{B(\Phi_B)} \bar{\Delta}^{(\mathcal{A})}(t, \mu_Q^2)}_{\text{resolved, singular}} \right] \\ & + \int d\Phi_R \underbrace{\left[R(\Phi_R) - D^{(\mathcal{A})}(\Phi_R) \right]}_{\text{resolved, non-singular, } \equiv H^{(\mathcal{A})}}, \end{aligned} \quad (3.10)$$

where $\bar{B}^{(\mathcal{A})}$ denotes the so-called next-to-leading order weighted Born cross-section given by

$$\bar{B}^{(\mathcal{A})}(\Phi_B) = B(\Phi_B) + \tilde{V}(\Phi_B) + I^{(\mathcal{S})}(\Phi_B) + \int d\Phi_1 \left[D^{(\mathcal{A})}(\Phi_B, \Phi_1) - D^{(\mathcal{S})}(\Phi_B, \Phi_1) \right] \quad (3.11)$$

and $H^{(\mathcal{A})} = R - D^{(\mathcal{A})}$ describes the hard remainder term. The first emission is described in terms of a modified Sudakov form factor

$$\bar{\Delta}^{(\mathcal{A})}(t, t') = \exp \left\{ - \int_t^{t'} d\Phi_1 \frac{D^{(\mathcal{A})}(\Phi_B, \Phi_1)}{B(\Phi_B)} \right\}. \quad (3.12)$$

The first line in Equation (3.10) corresponds to so-called $\mathbb{S}$ -events, where the first emission is generated as in the Standard parton shower, but including the full colour structure. It corresponds to having either no resolvable emission between μ_Q^2 and t_c , or one emission generated by the modified shower at a scale t . In contrast, the second line refers to $\mathbb{H}$ -events, where a $\mathbb{H}$ ard emission is generated pursuant to the matrix element and the parton shower takes over afterwards. During event generation, either a $\bar{B}^{(\mathcal{A})}$ or $H^{(\mathcal{A})}$ seed event is chosen, according to their respective cross-sections. The MC@NLO algorithm reproduces the next-to-leading order accuracy while preserving the full (leading) logarithmic accuracy of the parton shower resummation [30, 33].

A potential pitfall of the S-MC@NLO method is the exponentiation non-positive subtraction terms within the Sudakov form factors as a result of sub-leading colour configurations in the real emission matrix elements. Consequently, Sudakov form factors may become larger than one, loosing their interpretability as no-branching probability. A solution to this is an extension of the veto algorithm [18, 30] outlined in Section 2.2.2. More precisely, an additional estimate $h(t)$ is introduced, whereas the additional weight $g(t)/h(t)$ is applied analytically and not in a Monte Carlo fashion. In other words, the S-MC@NLO prescription employs a weighted $N_C = 3$ one-step parton shower. With that, Equation (2.13) becomes

$$\begin{aligned} \mathcal{P}_n(t, t') &= \frac{f(t)}{g(t)} h(t) \exp \left\{ - \int_t^{t_1} d\bar{t} h(\bar{t}) \right\} \\ &\times \prod_{i=1}^n \left[\int_{t_{i-1}}^{t_{i+1}} dt_i \left(1 - \frac{f(t_i)}{g(t_i)} \right) h(t_i) \exp \left\{ - \int_{t_i}^{t_{i+1}} d\bar{t} h(\bar{t}) \right\} \right] \end{aligned} \quad (3.13)$$

The corresponding analytic weight for one accepted splitting with n intermediate rejections is given by

$$w(t, t_1, \dots, t_n) = \frac{g(t)}{h(t)} \prod_{i=1}^n \frac{g(t_i)}{h(t_i)} \frac{h(t_i) - f(t_i)}{g(t_i) - f(t_i)}. \quad (3.14)$$

These weights effectively absorb the negative sign of the MC@NLO splitting kernels and thus may become negative. Furthermore, the density $h(t)$ is not necessarily an overestimate of $f(t)$, leading to negative weights as well.

Negative weights are a severe problem of event generation, since they can spoil the statistical accuracy of samples significantly. This issue, as well as mechanisms to reduce the negative

weight fraction are discussed in more detail in Chapter 5. In contrast to the negative contributions that appear in next-to-leading order calculations due to the subtraction formalism, the positive and negative weight distribution in the MC@NLO method are separately finite, enabling the generation of unweighted events, in the sense of equally weighted, up to a sign.

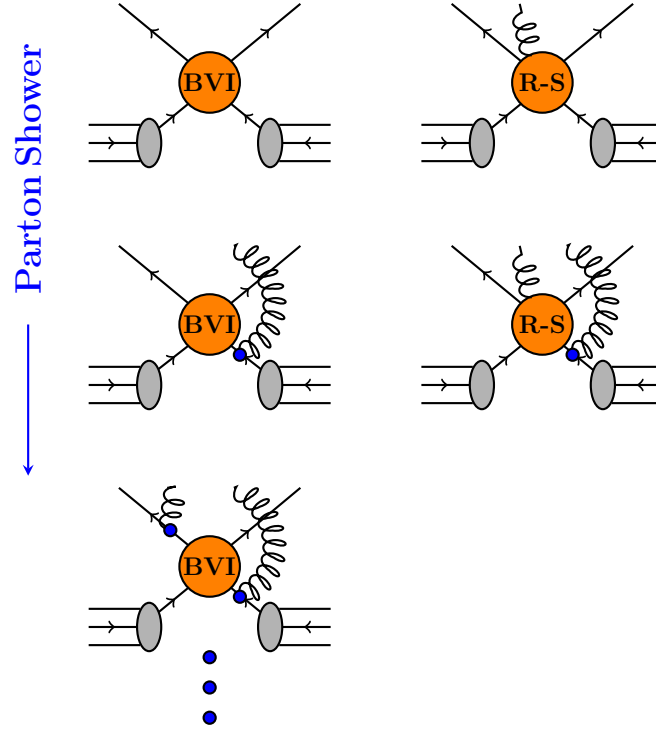


Figure 3.2: Pictorial illustration of the double-counting occurring during the application of a parton shower algorithm on top of a next-to-leading order calculation.

3.4 Multijet Merging of Matrix Elements and Parton Showers

Parton shower algorithms introduced in Section 3.2 only approximate higher-order real emissions in the soft-collinear region. Hence, they are not reliable in the description of additional hard emissions. The idea behind multijet merging is to improve the parton shower description by consistently including higher-multiplicity matrix elements. This is done by dividing the emission phase-space into two complementary regions, where the parton shower takes over the soft regime of jet evolution, while the hard regime is described by means of appropriate matrix elements, accounting for jet production. Generally, this decomposition can be described in terms of a theoretical merging scale parameter Q_{cut} . The first multijet merging algorithm was presented in [34], named the CKKW method after its authors. In the following, the focus is on the variant of this merging algorithm at leading order, called MEPS@LO, as implemented in SHERPA [35].

Including high-multiplicity matrix elements into the parton shower description necessitates the evolution of the shower from varying phase-space and colour configurations. In order to maintain the unitarity of the inclusive cross-section, matrix elements having m additional outgoing legs have to be reinterpreted as stemming from an underlying *core process*, which sets the resummation scale μ_Q^2 . This leads to the construction of an inverted parton shower history by means of a backward-clustering algorithm, that identifies intermediate splitting scales

$$t_m \leq t_{m-1} \leq \dots \leq t_2 \leq t_1 \leq t_{\text{core}} = \mu_Q^2, \quad (3.15)$$

effectively setting the initial conditions of the shower. Here, t_m denotes the scale of the first clustering step and so forth, down to the core scale t_{core} . Figure 3.3 pictorially exemplifies this clustering for $m = 2$ additional jets. The parton shower then starts off the identified core process, incorporating the predefined splitting scales using a so-called *truncated shower*. With that, the double-counting that would arise by plainly summing up all matrix elements with differing multiplicity is removed by appropriate vetoes on shower emissions producing additional hard jets above the merging scale Q_{cut} , called *Sudakov veto*. Furthermore, emission coming from the matrix element are restricted to be above Q_{cut} .

Considering the merging of matrix elements corresponding to the production of n and $(n+1)$ jets into one inclusive sample, the expectation value of a given infrared-finite observable O can be written as

$$\begin{aligned} \langle O \rangle_{\text{MEPs@LO}} = & \int d\Phi_n B_n(\Phi_n) \left[\Delta_n(t_c, \mu_Q^2) O(\Phi_n) \right. \\ & + \int_{t_c}^{\mu_Q^2} d\Phi_1 K_n(\Phi_1) \Delta_n(t_{n+1}, \mu_Q^2) \Theta(Q_{\text{cut}} - Q_{n+1}) O(\Phi_{n+1}) \left. \right] \\ & + \int d\Phi_{n+1} B_{n+1}(\Phi_{n+1}) \Delta_n(t_{n+1}, \mu_Q^2) \Theta(Q_{n+1} - Q_{\text{cut}}) O(\Phi_{n+1}). \end{aligned} \quad (3.16)$$

This describes the first emission step, coming either from the parton shower or a matrix element. Here, B_n and B_{n+1} describe the leading order matrix elements with n and $(n+1)$ outgoing legs, respectively. The matrix element region ($Q_{n+1} > Q_{\text{cut}}$) and the parton shower region ($Q_{n+1} < Q_{\text{cut}}$) are defined in terms of a hardness scale Q_{n+1} associated with the emission of the $(n+1)$ th particle. This method can be extended to multiplicities as high as desired.

The idea of multijet merging can be expanded to next-to-leading order accuracy, where the goal is to merge multiple MC@NLO matched event samples of increasing jet multiplicity in a consistent way. This method is called MEPS@NLO. Details on the NLO merging technique as employed within SHERPA are presented in [36, 37].

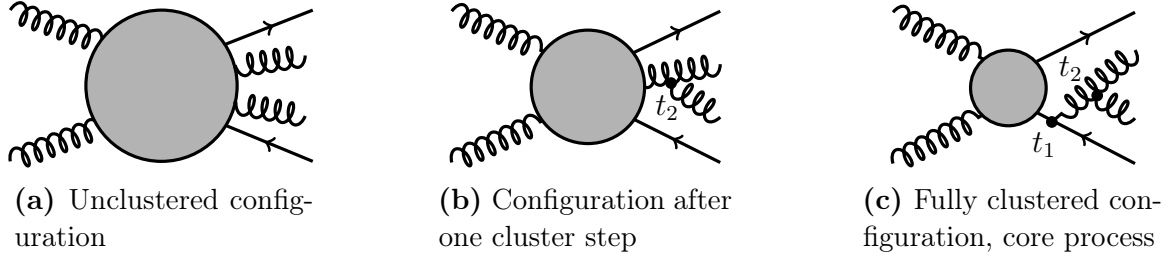


Figure 3.3: Pictorial representation of the determination of a clustering history involving two cluster steps.

3.5 Combining Matching and Merging

The matching of next-to-leading order calculations as well as the merging of multiple fixed-order matrix elements with parton shower algorithms, as discussed in the previous sections, can be combined into a hybrid approach called the MENLOPS method [36, 37, 38]. It combines NLO matching for the lowest-multiplicity with a LO multijet merging for the matrix elements with higher multiplicity. Especially the reliable modelling of processes contributing as background for potential physics beyond the Standard Model necessitates the inclusion of additional jet multiplicities at leading order.

In accordance with the leading order merging prescription, the NLO real emission correction has to be constrained to not produce any additional jet. The expectation value of an arbitrary infrared safe observable to $\mathcal{O}(\alpha_S)$ for an n , $(n+1)$ particle final-state, can be written as

$$\begin{aligned}
\langle O \rangle_{\text{MENLOPS}} = & \int d\Phi_n \bar{B}_n^{(A)} \left[\bar{\Delta}_n^{(A)}(t_c, \mu_Q^2) O(\Phi_n) \right. \\
& + \int_{t_c}^{\mu_Q^2} d\Phi_1 \frac{D_n^{(A)}(\Phi_n, \Phi_1)}{B_n(\Phi_n)} \bar{\Delta}_n^{(A)}(t_{n+1}, \mu_Q^2) \Theta(Q_{\text{cut}} - Q_{n+1}) O(\Phi_{n+1}) \Big] \\
& + \int d\Phi_{n+1} H_n^{(A)}(\Phi_{n+1}) \Delta_n^{(\text{PS})}(t_{n+1}, \mu_Q^2) \Theta(Q_{\text{cut}} - Q_{n+1}) O(\Phi_{n+1}) \\
& + \int d\Phi_{n+1} k_n^{(A)}(\Phi_{n+1}) B_{n+1}(\Phi_{n+1}) \Delta_n^{(\text{PS})}(t_{n+1}, \mu_Q^2) \Theta(Q_{n+1} - Q_{\text{cut}}) O(\Phi_{n+1}).
\end{aligned} \tag{3.17}$$

The higher-multiplicity part is multiplied by a local K-factor $k_n^{(A)}$, given by

$$k_n^{(A)}(\Phi_{n+1}) = \frac{\bar{B}_n^{(A)}(\Phi_n)}{B_n(\Phi_n)} \left(1 - \frac{H_n(\Phi_{n+1})}{R_n(\Phi_{n+1})} \right) + \frac{H_n(\Phi_{n+1})}{R_n(\Phi_{n+1})}, \tag{3.18}$$

which incorporates NLO corrections to the lowest multiplicity final-state n , effectively minimising discontinuities along Q_{cut} . This K-factor tends to one for hard emissions, since $H_n \rightarrow B_{n+1}$. On the other hand, $H_n \rightarrow 0$ for a soft additional parton, yielding $k_n^{(A)} \rightarrow \bar{B}_n^{(A)}/B_n$. Again, this method can be extended to multiplicities as high as desired.

4 Improvements of Unweighting Performance

The theoretical foundations of the generation of unweighted events were already discussed in Section 2.3. While the computational costs of event generation is increasing rapidly when going to higher-multiplicities in the hard scattering and higher orders in the perturbative expansion, the driving factor is the subsequent detector simulation. Thus, with regard to the statistical accuracy, it is preferable to use fewer equally weighted events than many events with large weight fluctuations. Certainly, unweighted events must also be preferred because of their natural-like distribution, i.e. they produce a much more realistic description of actual physical events. The unweighting of events can, however, become highly inefficient as a result of broadening weight distributions towards lower values as a consequence of the increasing difficulty to approximate the underlying matrix elements.

This chapter will introduce a method to tackle this problem by employing Machine Learning algorithms in order to increase the unweighting performance. After introducing the basic idea of the so-called Machine Learning based unweighting, the fundamentals of artificial neural networks, in the following implicitly referred to as neural networks (NN), are discussed. The optimisation procedure of the neural network architecture used throughout this thesis is described and its performance investigated. Finally, results for processes belonging to the $V+4$ jets and $t\bar{t}+3$ jets groups are presented.

4.1 Machine Learning Based Unweighting

4.1.1 Introduction to Machine Learning

Machine Learning (ML) is the science of programming computer systems so they can perform a specific task without explicit instructions by learning from data [39]. A prominent daily example is a spam filter. There is a vast amount of different types of ML systems. A common way to classify them is to distinguish between whether they are trained with human supervision, *supervised learning*, or not, *unsupervised learning*.

In supervised learning, the algorithm obtains the training data including the desired output, called labels. The algorithm aims at learning the functional dependence between the given

input and output in order to predict the outcome of new, unseen data. Supervised learning tasks can be further categorised into *classification* and *regression* tasks. While classification algorithms try to predict discrete values, called classes, regressors have a continuous range as output. On the other hand, unsupervised learning algorithms receive unlabelled training data. They try to find patterns and structures within the data for example clustering or grouping of data points.

The problem at hand, which will be discussed in the following, is a supervised regression task.

4.1.2 Basic Idea

In Section 2.3, the decreasing unweighting efficiency when going to higher multiplicities was already discussed. Not only more events need to be generated on average in order to obtain one unweighted event, but also the time needed for the evaluation of the matrix element and phase-space weight per generated event is increasing significantly with more particles in the final-state. This is where Machine Learning comes into play.

As mentioned above, supervised learning algorithms try to learn the relationship between input and output variables to be able to estimate the outcome of new incoming data. The underlying idea of the Machine Learning based unweighting is to utilise the capability of these algorithms to predict approximated weights a , effectively learning the integrators, and using them to decide whether the event survives the unweighting criterion or not. The virtue of this method is the time saving concerning the calculation of the matrix element and phase-space weight, improving the unweighting performance if the Machine Learning algorithm is capable to make fast and precise predictions. In other words, the ML based unweighting employs Machine Learning models to determine whether an event is likely to survive the unweighting condition, saving CPU time otherwise required to explicitly calculate the event weight for events that will probably be thrown away anyway. The true weight is only calculated for events passing the unweighting criterion using the approximated weights.

The ML models are of course not able to predict the “true” weight w perfectly. Using the predicted weights a , without correction, would thus lead to a bias in the physics description. Hence, events surviving the unweighting procedure with the approximated weight will need to obtain the weight $\frac{w}{a}$. To see why this holds, consider the sum over the probability to accept an event P_i times its final relative weight r_i for the classic and ML based unweighting procedure

$$\sum_i P_i^{classic} \cdot r_i^{classic} = \sum_i \frac{w_i}{w_{max}} \cdot 1 = \sum_i \frac{a_i}{w_{max}} \cdot \frac{w_i}{a_i} = \sum_i P_i^{ML} \cdot r_i^{ML}. \quad (4.1)$$

A comparison of both unweighting methods is displayed in Figure 4.1. One can think of the weight $\frac{w}{a}$ as a sort of correction factor, taking the mismatch between w and a into account. If

a overestimates w , the event will be more likely to pass the unweighting step, but will receive a smaller weight $\frac{w}{a}$ in return. Equally, if the ML model underestimates the true weight, making it less likely to pass the unweighting criterion, it will obtain a larger weight.

Classic Unweighting	Machine Learning based Unweighting
1: Generate event 2: Calculate corresponding weight w 3: Generate uniformly distributed random number $R \in [0, 1]$ 4: if $w > R \cdot w_{max}$ then accept event and apply unit weight 5: else go back to 1 	1: Generate event 2: Calculate approximated weight a 3: Generate uniformly distributed random number $R \in [0, 1]$ 4: if $a > R \cdot w_{max}$ then go to 6 5: else go back to 1 6: Calculate event weight w and keep event with weight $x := \frac{w}{a}$

Figure 4.1: Comparison between the classic and the Machine Learning based unweighting procedure up to the first unweighting step.

The basic idea of the ML based unweighting was first formulated in [40], however without explicit implementation into SHERPA. Possible performance improvements were studied based solely on statistical calculations using Python, neglecting all times needed aside from the calculations of the approximated weights a . Besides assessing the actual possible time improvements, a direct implementation also allows to verify that the ML approach does not influence the physics description.

The workflow of SHERPA can be divided into an integration and a subsequent event generation phase. First, the cross-sections of the hard processes are evaluated and, simultaneously, the integration over the phase-space is optimised. Afterwards, events can be generated using the results obtained during the integration phase. In case of the ML approach, the training points needed to train the algorithm are obtained during the integration, reusing the phase-space points and corresponding weights needed to calculate the cross-section of the process in question. Thus, the integration phase is extended by the training of the Machine Learning algorithm. The trained model can then be used to predict the approximated weights during the event generation.

Events obtained using the Machine Learning prescription shown in Figure 4.1 are still weighted. Only in the case that the approximated weight a is equal to the true weight w , the fraction $x = \frac{w}{a}$ will be one. Hence, a second unweighting is needed.

Second Unweighting

As discussed above, the Machine Learning based unweighting approach necessitates the application of two unweighting procedures. The problem at hand, however, is to find the maximum x_{max} needed to perform the second unweighting step. To address this issue, a sufficiently large set of test points has to be generated in order to be able to estimate the second maximum. Therefore, a “second” integration phase is employed, effectively extending the regular integration stage by generating more phase-space points than necessary to evaluate the total cross-section. These test points are then feed to the trained algorithm. Just taking the maximum of the resulting $\frac{w}{a}$ distribution would lead to a too large x_{max} . The maximum would likely be an outlier, significantly underestimating the true weight w . Consequently, the corresponding event has a smaller chance to survive the first unweighting criterion and w might have been small (in comparison to w_{max}) already. Hence, the acceptance probability for this specific event tends to zero and using it for the definition of x_{max} would spoil the success of the Machine Learning based unweighting.

During this thesis, another approach is followed to determine an estimate of the second maximum. After the generation of the test points, which should be at least of size of the sample used for the training process to obtain a reliable result, one proceeds as follows:

1. Iterate through the test points k times.
2. Keep the maxima $x_{max}^k = \left(\frac{w}{a}\right)_{max}^k$ of events that would have survived the first unweighting criterion.
3. Take the median of the individual maxima $\bigcup_{i=1}^k x_{max}^i$ as final maximum x_{max} .

It cannot be guaranteed that no weight x will exceed x_{max} . Therefore, events generated using the Machine Learning based unweighting will effectively be *partially unweighted*. As a result, one may need to generate more events using the ML approach in order to achieve the same statistical accuracy in comparison to only equally weighted events. In order to take this fact into account, we define the second unweighting efficiency ε_2

$$\varepsilon_2 = \frac{(\sum_i x_i)^2}{\sum_i x_i^2} \cdot \frac{1}{N} = \frac{N_{eff}}{N}, \quad (4.2)$$

where the sum runs over all N final weights x_i of events passing the second unweighting. In general, the statistical accuracy of a sample consisting of N events with weights w_i can be calculated as

$$\frac{1}{\sqrt{N_{eff}}} = \frac{\sqrt{\sum_i w_i^2}}{\sum_i w_i} \stackrel{w_i=const=:c}{=} \frac{\sqrt{c^2 \sum_i 1}}{c \sum_i 1} = \frac{\sqrt{N}}{N} = \frac{1}{\sqrt{N}}. \quad (4.3)$$

If x_{max} is a true maximum, the effective number of events N_{eff} will be equal to N and consequently $\varepsilon_2 = 1$. The effective factor f_{eff} that the Machine Learning based unweighting is faster

than SHERPA is therefore given by

$$f_{\text{eff}} = \frac{t_{\text{SHERPA}}}{t_{\text{ML}}} \cdot \varepsilon_2. \quad (4.4)$$

The final weight w_i^{final} obtained after the second unweighting step is then

$$w_i^{\text{final}} = w_{\text{max}} \cdot x_{\text{max}} \cdot \Theta\left(\frac{x_i}{x_{\text{max}}} - R\right) \left[\Theta\left(1 - \frac{x_i}{x_{\text{max}}}\right) + \frac{x_i}{x_{\text{max}}} \Theta\left(\frac{x_i}{x_{\text{max}}} - 1\right) \right] \quad (4.5)$$

where $R \in [0, 1]$ is a uniform distributed random number. As will be discussed in Section 4.3, for the processes tested within this work, there are only a few events that exceed the second maximum and only by a small factor.

Error Definition

Mathematically speaking, optimisation problems consist of the maximisation or minimisation of a real function $f(\vec{\vartheta})$ by systematically selecting the values of the parameter vector $\vec{\vartheta}$ within an allowed domain and computing the value of f . An optimisation algorithm frequently used in Machine Learning is Gradient Descent as well as variants of it such as Batch, Stochastic and Mini-batch Gradient Descent. The general idea is to iteratively tweak the parameters of the Machine Learning algorithm in order to minimise a so-called cost function, also referred to as loss function, for example the mean squared error (MSE). The cost function is used to estimate the error between prediction and the correct result. During the training, the parameters are gradually adjusted, attempting to decrease the cost function, until a good prediction is received and the algorithm converges to a minimum. For more details on Gradient Descent (and variants of it), see e.g. [39, 41].

The values of the event weights span several orders of magnitude, making it very difficult for Machine Learning algorithms, such as neural networks, to learn their distribution. Thus, the logarithm of the weights is used for training. Using the MSE as cost function, one then effectively calculates the mean squared logarithmic deviation (MSLD), defined as ¹

$$\text{MSLD} = \frac{1}{N} \sum_{i=1}^N \left[\ln(a_i) - \ln(w_i) \right]^2 = \frac{1}{N} \sum_{i=1}^N \ln^2 \left(\frac{a_i}{w_i} \right) = \frac{1}{N} \sum_{i=1}^N \ln^2 \left(\frac{w_i}{a_i} \right), \quad (4.6)$$

often also called mean squared logarithmic error (MSLE). The advantage of this error definition is that the ratio $\frac{w}{a}$ is important instead of their difference. Considering truth prediction pairs

¹Usually the mean squared logarithmic deviation is defined as $\text{MSLD} = \frac{1}{N} \sum_{i=1}^N \left[\ln(a_i + 1) - \ln(w_i + 1) \right]^2$, i.e. with an additional “+1” in order to avoid taking the logarithm of zero, which is not defined. For the problem at hand, the plus one can be omitted, since events with weight equal to zero are unphysical in the sense that their probability to occur is zero and will thus be thrown away.

$(w_i, a_i) = (10^{-n}, 10^{-(n+1)})$ with $n \in \mathbb{N}$, one obtains the same weight $x_i = \frac{w_i}{a_i}$ for all n and consequently the same MSLD, but differing MSE. Moreover, $w = c \cdot a$ and $a = c \cdot w$, with c a constant larger than zero, will yield the same MSLD, as desired. Hence, the mean squared logarithmic deviation is used in the following to quantify the performance of the ML model.

4.1.3 Software and Workflow

Nowadays several machine learning algorithms are available that can be used to solve the regression problem at hand. During this thesis, the utility of (boosted) decision trees, random forests and neural networks were investigated in terms of speed and accuracy. While being faster², random forests and (boosted) decision trees turned out to be inadequate in comparison to employing neural networks. The reason is a decreased efficiency for the first unweighting as a result of poorer prediction accuracy and the resulting increased maximum for the second unweighting x_{max} , effectively nullifying their faster performance per approximated weight. Thus, only neural networks are discussed further in the following, see Section 4.2.

During this thesis, the neural networks are parametrised using Keras [42] with TensorFlow [43] backend. The input features and corresponding event weights used for training are obtained during the integration phase. After training the neural network in Python, its topology and weights are saved as JSON file. During event generation, the saved model is loaded in order to predict the approximated event weights a using frugally-deep [44] as basic frame, with several modifications made in order to be compatible with SHERPA. The main advantage is that SHERPA is only extended by a small header-only library without the need to link against TensorFlow. One CPU core is used per prediction. Event generation in SHERPA can be parallelized trivially by starting multiple instances of SHERPA with different random seeds. It should be noted that in the future, implementations of event generators on accelerators such as GPUs and TPUs may become available enabling a speed up of the neural network in terms of prediction time within SHERPA.

4.2 Neural Networks

4.2.1 Fundamentals of Neural Networks

Artificial neural networks are brain-inspired computational models. Biological brains are composed of numerous and strongly interconnected neurons and artificial neural networks try to loosely model them. They consist of connected units or nodes, also called artificial neurons, and just like biological neurons receive short electrical impulses via synapses in brains, each

²The prediction speed of the machine learning models is dependent on their architecture. One can construct simple neural networks which are able to predict faster than a very deep decision tree. However, the accuracy and ability to generalise may decrease with simpler topologies.

connection can transmit a signal represented by a real number to other neurons. One of the simplest neural networks architectures is the perceptron, invented in 1957 by Frank Rosenblatt [45]. In the following, only feedforward neural networks, as used within this thesis, are discussed. In contrast to recurrent neural networks, wherein connections between nodes can form cycles or loops, signals are only passed forward. More details on recurrent neural networks can for example be found in [39, 46].

Mathematically, artificial neurons are functions g_k of an input $\vec{x} = (x_1, x_2, \dots, x_n)$ weighted by a vector $\vec{w}_k = (w_{1k}, w_{2k}, \dots, w_{nk})$ and optionally supplemented with a bias b_k . The resulting real number is passed through a non-linear function known as activation function f , giving

$$y_k = g_k(\vec{x}) = f\left(b_k + \sum_{i=1}^n x_i w_{ik}\right). \quad (4.7)$$

Figure 4.2 schematically illustrates a single artificial neuron. A feedforward neural network consists of discrete layers of such neurons, where the output of a neuron of one layer becomes the input of neurons of the subsequent layer. While the size of the first (*input layer*) and last layer (*output layer*) are determined by the size of the features and corresponding labels according to required task, the dimension and number of intermediate *hidden layers* can be chosen freely. The activation functions may differ for each layer. A simple neural network with two input variables, one hidden layer with three nodes and two output variables is represented in Figure 4.3.

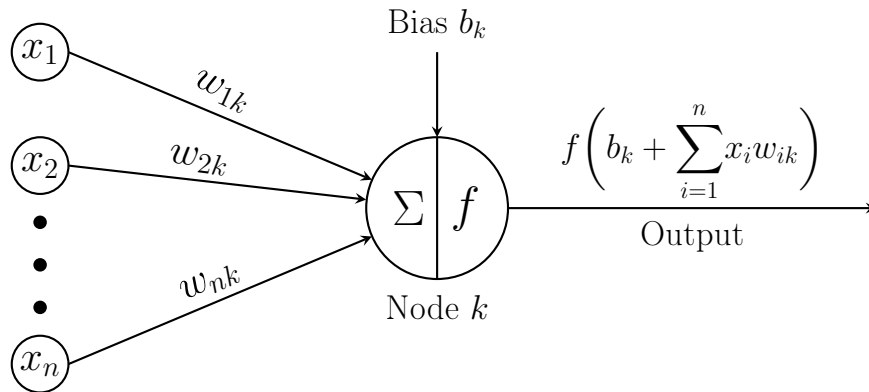


Figure 4.2: Schematic representation of a single artificial neuron (node k) receiving an input vector $\vec{x} = (x_1, x_2, \dots, x_n)$. Each input x_i is multiplied with a corresponding weight w_{ik} . The results are summed up, possibly adding a bias b_k , and passed through an activation function f .

The weights and biases are learnable parameters of the network that are optimised during the training process in order to minimise the loss function and obtain a good predictive model. Backpropagation [47] is a widely used algorithm to train feedforward neural networks for supervised learning tasks. The algorithm first makes a prediction, called *forward pass*, and measures the error between prediction and correct result in terms of the cost function. Using

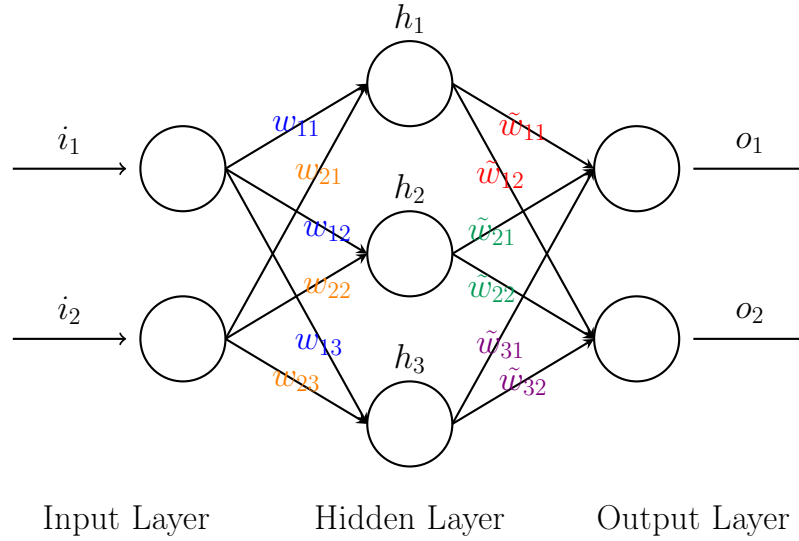


Figure 4.3: Illustration of a simple fully connected feedforward neural network with two input variables i_l , one hidden layer with three nodes h_k and two output variables o_m , where $k \in \{1, 2, 3\}$ and $l, m \in \{1, 2\}$. The values of the hidden layer are given by $h_k = f\left(\sum_l w_{lk} i_l + b_k\right)$ and the outputs by $o_m = f\left(\sum_k \tilde{w}_{km} h_k + b'_m\right)$. The activation functions f may differ.

the chain rule of differential calculus, the error gradient is propagated from output to input layer in reverse measuring the error contribution from each connection, called *backward pass*. A complete pass, forward and backward, through the entire training dataset is called *epoch*. Initially randomly assigned weights and biases are then gradually adjusted during the training process by utilising gradient descent or one of its variants. A proper tuning of the weights and biases ensures lower errors, making the model more reliable by increasing its ability to generalise.

In order to assess the performance of the neural network on new data, the dataset is usually divided into a training and a test set. While the former is used to actually train the algorithm, the latter enables an unbiased evaluation of the network's performance on a representative, statistically independent dataset. The test set is not used for the optimisation. In each epoch, the loss for both sets is calculated, called the training and validation loss, respectively. The difference between the two indicates how well the model is generalising to unseen records. If the training error is low but the validation error is high, the model has learned the training set too well and is hence overfitting. Ideally, both errors should become equal and decrease with each epoch until the validation loss starts converging or increasing again as a result of aforementioned memorising of the training data, indicating to stop the learning process. Commonly and henceforth, 80% of the dataset is used for training and the remaining 20% for validation. Additionally, a new test set containing one million points is utilised to assess the network's performance after the training.

Aside from the training process itself there are many parameters whose values are set before the learning process begins. The number of layers, neurons per layer, weight initialisation, number of epochs and the activation function used for each layer are just a few examples. These *hyperparameters* also need to be tuned to achieve a good performing model.

4.2.2 Optimisation of the Neural Network

The variety of tunable parameters of neural networks make the choice of an optimal network architecture a non-trivial task. In addition, with regard to the Machine Learning based unweighting, the model topology should not only perform well for one specific process but is ought to generalise to a multitude of processes.

Fine-tuning the parameters of a neural network can be done following different approaches. Two commonly deployed examples are grid and random search. The idea behind a grid search is to simply define a set of parameter values, train the model for all possible parameter combinations, and select the one performing best. If the hyperparameter search space is large and in order to make an educated guess on parameters for the grid search, it is often preferable to apply a randomised search. Instead of checking all possible combinations of hyperparameters within the customised ranges, a given number of random sets of the parameters is investigated.

In contrast, the optimisation of the neural network for the Machine Learning based unweighting is done following a heuristic approach for one representative process, namely $gg \rightarrow e^-e^+ggd\bar{d}$ at leading-order. For the examined process groups $V+4$ jets and $t\bar{t}+3$ jets, there are $\mathcal{O}(100)$ contributing processes. These processes are of particular interest as they take up a major part of the CPU resources of the ATLAS and CMS experiments at CERN with regard to event generator usage. Conducting the hyperparameter tuning for every single process would be unreasonable and spoil the potential benefit of the ML approach. The performance of the obtained configuration for other processes is checked afterwards, see Section 4.3. The investigation of high-multiplicity processes is not only interesting because of their low unweighting efficiencies, but they are also important for the search of new physics beyond the Standard Model where they contribute as background.

In the scope of this thesis, only simple fully connected feedforward neural networks are examined. One could think of more complex network structures, though the model should not only be able to give precise but also fast predictions for the problem at hand. The considered hyperparameters, their respective investigated ranges and options as well as the finally implemented values are listed in Table 4.1. The final values are obtained by taking the mean of five optimisation runs with different random seeds for each parameter and errors are estimated using the sample standard deviation. If not otherwise stated, a training period of 50 epochs is used throughout this chapter. The starting point is a network architecture consisting of two

hidden layers with 64 nodes each, random weight initialisation, rectified linear unit (ReLU) [48] activation function for the hidden layers and Adam [49] optimiser. While optimising one hyperparameter, all other parameters are left at their default values. The individual considered hyperparameters will be briefly reviewed in the following. Note that the optimal choice of some hyperparameters can be dependent on the value of others. Furthermore, it should be stressed, that there is no perfect NN architecture that is suited for all process groups and multiplicities.

Hyperparameter	Final Value	Tested Values
Hidden Layers	3	{1, 2, 3, 4}
Nodes per Layer	64	{32, 64, 128, 256}
Activation Function	ReLU	{Tanh, Sigmoid, ReLU, Elu, Leaky ReLU, Softsign}
Optimiser	Adam	{RMSProp, Adam}
Learning Rate	10^{-3}	$\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$
Dropout	0.0	{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}
Epochs	200	{[1, 300]}

Table 4.1: Hyperparameters investigated during the optimisation process of the deep feed-forward neural network used for the Machine Learning based unweighting. The examined parameter options or ranges as well as the eventually implemented values are listed.

Choice of Input Features

In order to train supervised regression algorithms a set of labelled training data is needed. The algorithm tries to model the relation between features and labels. Hence, the input needs to be chosen carefully in order to get a precise prediction.

To predict the event weight w , which is given by the absolute squared matrix element times a phase space weight, different input features are feasible. One obvious possibility, for instance, is to use the four-momenta of the hard scattering. However, the performance of a Machine Learning algorithm can sometimes be improved by reducing the number of features and selecting a subset of relevant inputs, effectively filtering out features that are not useful in addition to the remaining ones. Moreover, reducing the number of input variables cuts down the computational costs of the training process and (slightly) decreases the size of the neural network. Hence, also the three-momenta, as well as the energies of the particles in the hard scattering are tested for their utility as input features.

Also the random numbers, used to calculate the four-momenta, can serve as input to the ML algorithm. Considering a $2 \rightarrow n$ process, in total $3n$ random numbers are generated per event.

Two of these are needed for the determination of the energy fraction of the initial-state particles according to their PDFs. Further, $3n - 4$ random numbers need to be generated, since each of the n outgoing particles has three degrees of freedom, the minus four stemming from energy-momentum conservation. The two remaining random numbers are used for the selection of the initial-state and final-state channel³. The meaning of the other random numbers is defined by these channels. They specify the mapping from the random numbers to the four-momenta. The random numbers naturally lie between zero and one. To compare the different features, the momenta and energies are hence scaled to a range of $[0, +1]$ as well.

Figure 4.4 (a) shows the MSLD obtained for the different input features. Using the random numbers as input clearly results in the highest error among the tested features. A reason for this is the large number of channels, especially concerning the final-state. As mentioned above, the two random numbers used for the channel selection determine the transformation utilised to calculate the four-momenta from the other random numbers. Explicitly adding the channel indices as input did not improve the performance. While an explicit channel separation, i.e. using a prescribed initial- and final-state channel, enhances the predictive power in comparison to the general case, the random numbers still result in the highest error among the tested features. The more high-level features, such as the momenta, are independent of the respective channels, making it easier for the ML algorithm to learn the weight distribution.

Lacking the additional information of the spatial momentum components, the energies give higher deviations in comparison to the four- and three-momenta. The best performing under the tested features are the three-momenta with a slightly lower error than the four-momenta. The particles involved in the scattering process are light partons and leptons which are treated massless during the calculations. The additional information on the energies can hence be calculated using the three-momenta and is therefore obsolete. In other words, the neural network is able to learn the underlying Lorentz structure. Hence, the additional energy input even slightly decreases the performance of the network as it increases its complexity while, in fact, being redundant. As a result, the three-momenta are used as features. For heavy particles with non-negligible masses such as top quarks, one could still think about using the four-momenta instead. The learning curves for the three-momenta, averaged over the five runs, are plotted in 4.4 (b).

Feature Scaling

Some Machine Learning algorithms⁴, such as neural networks, do not perform well if the input exhibits widely varying scales, which is due to individual features governing the employed

³SHERPA uses an adaptive multi-channel method for the evaluation of the initial- and final-state integrals, see 2.1.3. Roughly speaking, each propagator term is mapped to a different channel.

⁴For instance, tree-based algorithms, such as decision trees and random forests, are independent of the feature scaling.

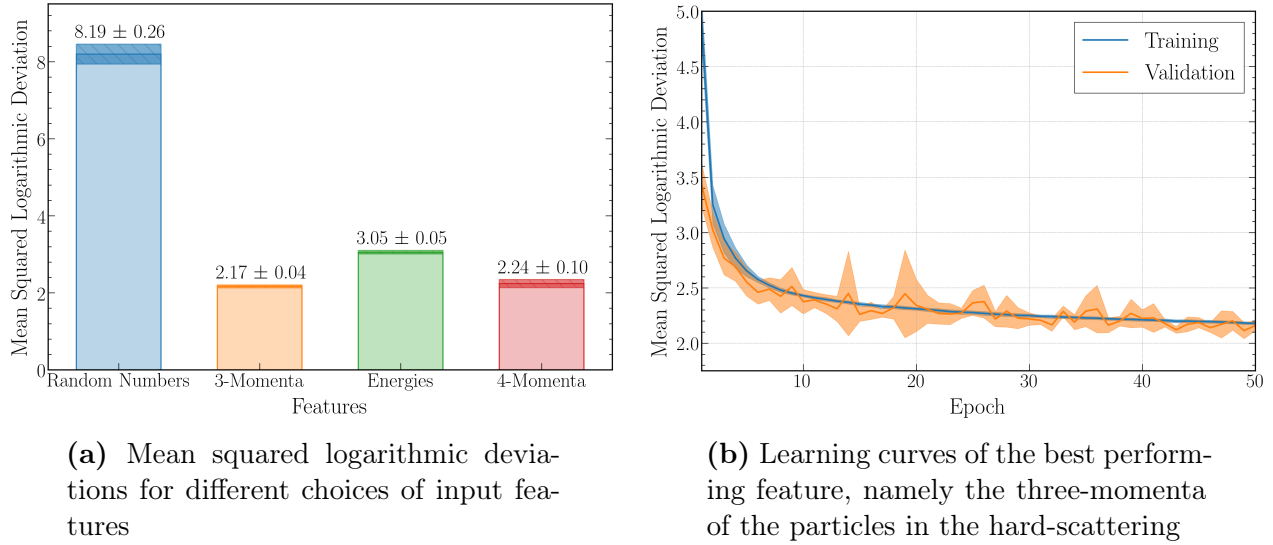


Figure 4.4: Comparison of the performance of different input features. The given values and their corresponding uncertainties are calculated as the means and standard deviations obtained from five optimisation runs for each set of features.

metric. Performed during the data preprocessing step, data normalisation can improve the training process by transforming the data into a particular range. In addition, gradient descent has been shown to converge substantially faster with an appropriate feature scaling [50]. Scaling the target values is generally not required.

So far, the three-momenta were scaled to a range of $[0, +1]$ in order to match the natural range of the random numbers. In addition, a scaling of $[-1, +1]$ and $[-0.5, +0.5]$, as well as mean normalisation and standardisation, where the features are scaled to have zero mean and unit variance, are tested. A comparison of the obtained deviations are illustrated in Figure 4.5 (a). Aside from the rescaling to $[0, +1]$, all investigated feature scaling behave roughly equally well. In the following, a normalisation to $[-1, +1]$ is chosen. Figure 4.5 (b) shows the learning curves received for this scaling.

Hidden Layers and Nodes per Layer

As described in Section 4.2.1, the number of nodes in the input and output layer are predetermined by the dimensions of the features and labels of the respective task. In contrast, the size and number of hidden layers can be tuned in order to optimise the performance of a neural network. Using just one hidden layer already enables to model very complex functions, provided it has enough nodes. However, deeper networks, i.e. with more hidden layers, exhibit a higher parameter efficiency than shallow nets. Shallow networks can be exponentially less efficient in terms of complexity than deeper ones [51, 52], making the latter much faster to train. On the other hand, deep networks are often affected by the problem of vanishing or exploding gradients making them trickier to train. Furthermore, they are prone to overfitting. The van-

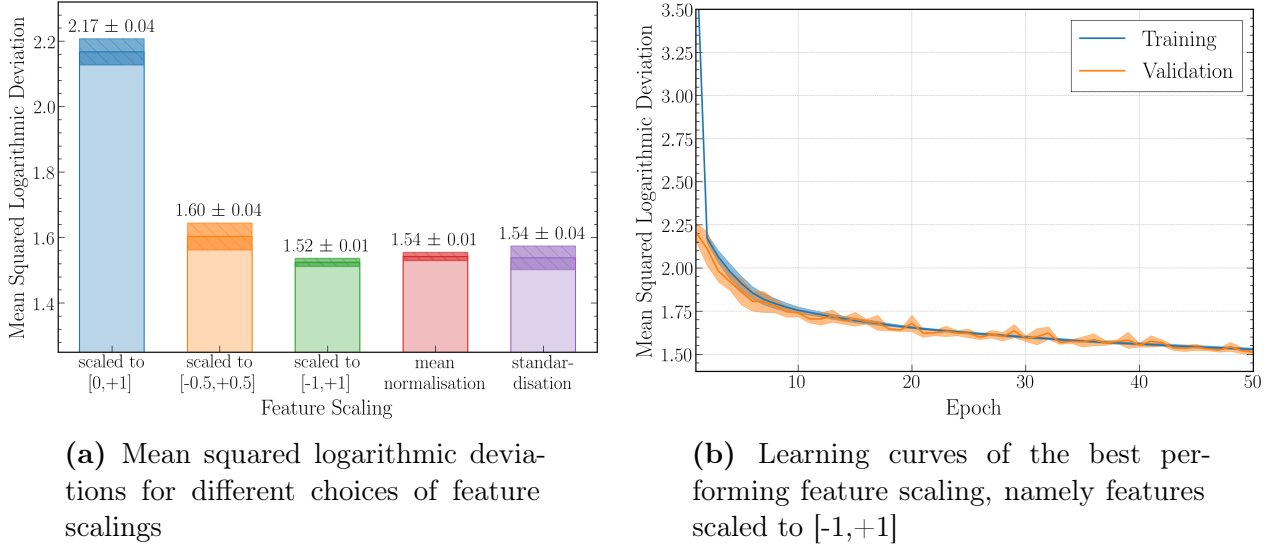


Figure 4.5: Comparison of the performance of different feature scalings. The given values are calculated as the means obtained from five optimisation runs for each scaling variant.

ishing and exploding gradient problems are difficulties encountered in the training of neural networks with gradient-based learning methods and backpropagation, mentioned in Section 4.2.1. The former is the issue of vanishingly small gradients, effectively preventing the weights to change during the update and, in the worst case, completely stopping the network from further training. The latter is the problem of accumulating large error gradients, resulting in very large updates of the networks weights during the training process with the effect of an unstable model which is unable to learn from the training data.

In terms of nodes per layer, many different approaches are common. One example are funnel-shaped architectures, with a decreasing number of nodes for deeper intermediate layers. A simpler approach, also used in this study, is to use the same number of nodes for every hidden layer, additionally reducing the number of hyperparameters.

Figure 4.6 (a) shows the errors obtained for different hidden layers and nodes per layer combinations. It can be seen that the error decreases when going to deeper and wider network architectures but the improvement seems to converge. The more complex topologies require a longer training process to optimise the weights and biases and possibly more training points to further decrease the obtained deviations, which should be avoided due to performance reasons. Regarding the Machine Learning based unweighting, precision is not the only concern but also the prediction speed. The respective time improvements obtained by using the NN to predict the approximated event weight a in comparison to the time SHERPA needs to calculate the matrix element and the phase-space weight (making up the event weight w) are displayed in Figure 4.6 (b). As a trade-off between precision and speed, an architecture consisting of three hidden layers and 64 nodes per layer is selected and used in the following. This topology enables an over 800 times faster evaluation of a than required by SHERPA for the exact

calculation of w , while also showing a comparatively low error.

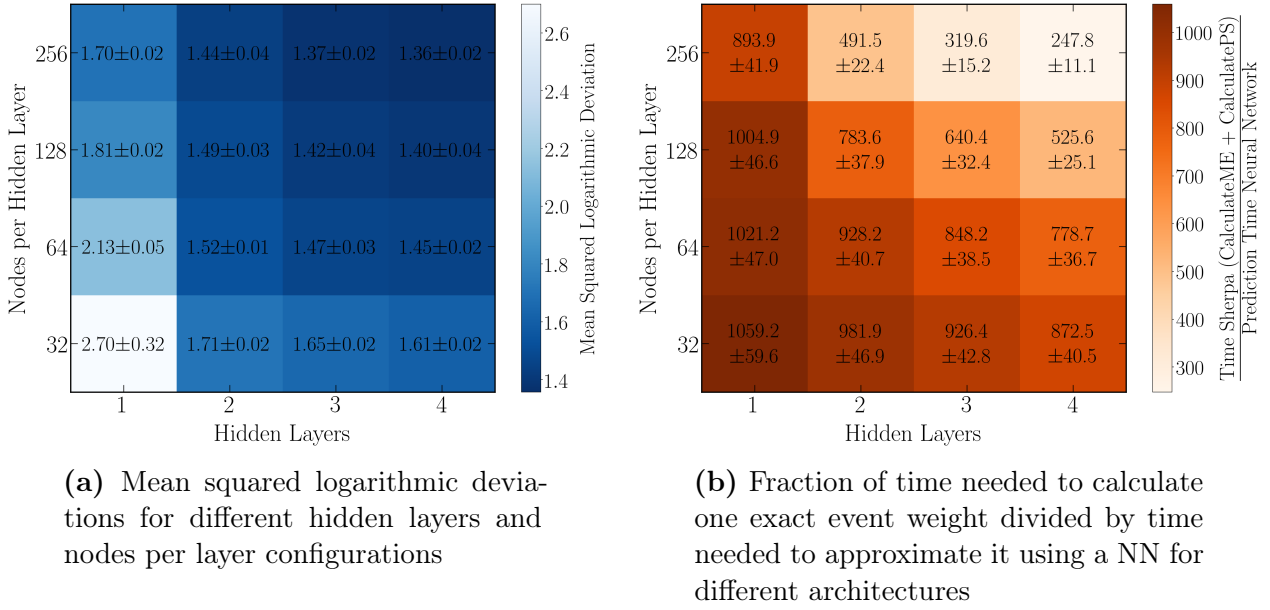


Figure 4.6: Comparison of the performance of different hidden layer and nodes per layer configurations in terms of accuracy and speed. The mean squared logarithmic deviations are calculated by taking the mean of five optimisation runs for each combination. The factor that the neural network is faster than SHERPA per event weight is determined by implementing one model for each configuration into SHERPA and measuring the times needed to calculate as well as to approximate the weight by generating ten thousand weighted events, respectively.

Activation Function

The output of a node is calculated in terms of a transfer function called activation function f . The prediction of the event weight is a regression task resulting in a real number as output. Therefore, a simple linear activation function ($f(x) = x$) is used for the output layer⁵, leaving its input unchanged. For the hidden layers, a multitude of different activations are possible. In the interest of simplification, the same activation is used for all intermediate layers, again reducing the number of tunable hyperparameters. An overview and definitions of the investigated activation functions can be found in the appendix, see Figure A.1. Activation functions including adjustable parameters are left at their default values. As can be seen in Figure 4.7 (a), ReLU, defined as

$$f(x) = x^+ = \max(0, x), \quad (4.8)$$

yields the lowest error and is hence implemented in the final NN architecture. Figure 4.7 (b) shows the training and the validation curve received for this activation, respectively. In addition, the ReLU activation is more computationally efficient than the also widespread charac-

⁵Classification task require different activation functions for the output layer, such as softmax.

teristic “S-shaped” curves (sigmoid curves), since it just picks the positive part of its argument without the need to perform expensive exponential operations as in sigmoid-like functions. The major benefit of ReLUs, however, is the reduced likelihood of vanishing gradients due to the fact that it does not saturate for large input values as opposed to the hyperbolic tangent or sigmoid function, which saturate at one. A faster learning is further achieved thanks to the constant gradient of ReLUs. Another benefit is sparsity for $x \leq 0$, effectively reducing the complexity of the neural network, leading to decreasing time and energy consumptions due to a reduced number of computations while potentially increasing its ability to generalise [53].

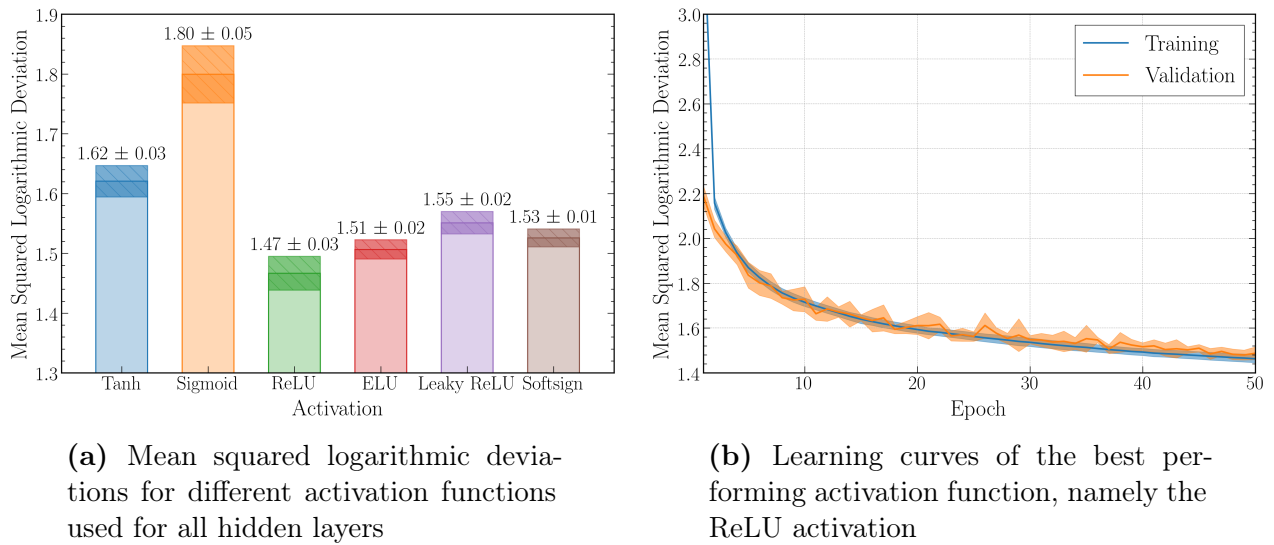


Figure 4.7: Comparison of the performance of different activation functions used for all hidden layers. The given values are calculated as the means obtained from five optimisation runs for each activation.

Weight Initialisation

So far, the weights of the NN were initialised randomly before the training process. A proper weight initialisation is important to alleviate the problem of vanishing or exploding gradients. Furthermore, it can also speed up the training process considerably. Two modern activation-aware techniques to initialise the weights are Xavier [54] (or Glorot) initialisation established for the hyperbolic tangent and sigmoid activations, and He [55] initialisation used for ReLU and its variants such as ELU. The underlying idea behind Xavier as well as the He initialisation techniques is to make the variance of the outputs of a layer equal to the variance of its inputs. In other words, they try to balance the variance of the activation, effectively accelerating the learning process and reducing the exploding and vanishing gradient problem. Since ReLU is the activation used for the hidden layers of the neural network architecture within this thesis, the He weight initialisation technique is used in the following.

Optimiser and Learning Rate

Optimisation algorithms try to minimise the cost function, which is dependent on the model's internal learnable parameters, the weights and biases in the case of neural networks and calculated by comparing the models predictions to the expected output values. These parameters are updated in the direction of the optimal solution, e.g. minimising the loss for instance by utilising gradient descent based algorithms. Different optimisation algorithms with varying degrees of accuracy and rapidness like classical stochastic gradient descent (SGD), Root Mean Square Propagation (RMSProp) [56] and Adam [49] optimisation are available.

While SGD maintains a single unchanged learning rate throughout the learning process, Adam computes individual adaptive learning rates for the different parameters based on estimations of first and second-order moments of the gradients. On the other hand, RMSProp adapts per-parameter learning rates based on the average first moment (the mean) of the gradients for the weight. The learning rate determines how much the model is changed in response to the estimated error in each update of the model parameters. Too small learning rates result in a longer training process with the chance to get stuck in a local minimum. Choosing a too high learning rate, on the other hand, may result in an unstable training process that does not converge or learning a sub-optimal set of weights too fast.

The Adam and RMSProp optimisers are tested with varying initial learning rates. All other parameters of the optimisers are left at their default values, as recommended for both algorithms. As can be seen in Figure 4.8, the Adam optimiser with a learning rate of 10^{-3} yields the lowest deviation among the tested setups and is therefore implemented.

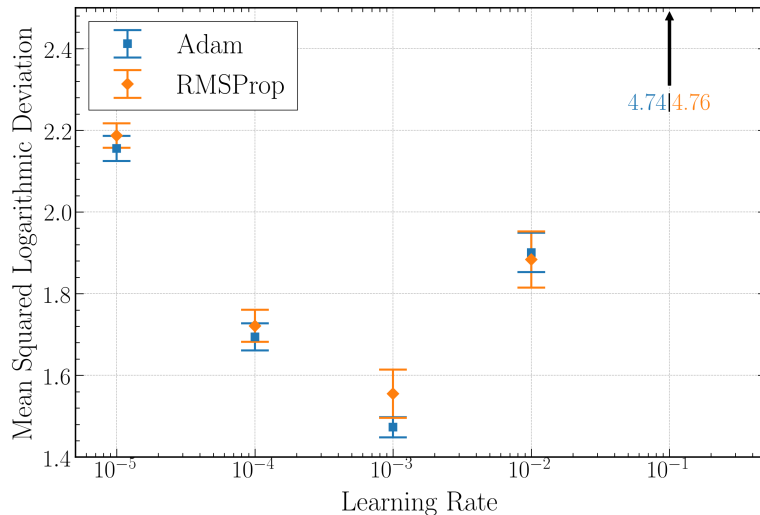


Figure 4.8: Comparison of the Adam (blue) and RMSProp (orange) optimisers with different learning rates. All other tunable parameters of the optimisers were left at their default values. The deviations are calculated as the means obtained from five optimisation runs for each learning rate and optimiser. The MSLD obtained for a learning rate of 0.1 are outside the presented value range.

Dropout

Deep neural networks tend to easily overfit too small training datasets. Dropout [57] is a method to prevent this issue by ignoring a certain, randomly chosen amount of neurons along with their connections during each training step. Neurons trained with dropout are able to co-adapt with their adjacent nodes and tend to be less sensitive to slight changes in the inputs, thus raising the networks ability to generalise. It is a computationally light regularisation method to reduce overfitting and enhance the networks generalisation capability.

Various dropout rates were tested in the course of this thesis, see Table 4.1. However, non-vanishing dropout rates lead to a deterioration of the NN prediction capability. As can be seen in the previous learning curves, e.g in Figure 4.7 (b), the employed model does not show signs of overfitting, therefore rendering the use of dropout obsolete.

Number of Training Points

Until now, one million phase-space points are used for the training process motivated by the number of phase-space points typically generated during the integration phase, which are of the order of $\mathcal{O}(10^5 - 10^6)$, depending on the process in question. Here, 80% of the dataset is used for the actual training of the network and the remaining 20% are utilised for validation. Choosing a too small training sample generally poses the risk of overfitting. The model learns the “small” training set too well, thereby loosing its ability to correctly predict the output of new input samples. On the contrary, using a larger training sample is almost always better when it comes to generalising to unseen data. The drawback of an increased number of training points, however, is the likewise extended training time. In case of the ML based unweighting, one advantage is the reuse of phase-space points generated to evaluate the cross-section of the given process in order to train the neural network, effectively recycling them. Hence, to enlarge the training set, one also has to artificially extend the integration phase, which might spoil the potential time improvements of the ML unweighting approach.

The number of training points was varied between 10^4 , which is of the order of the size of one iteration during the integration step, and 2.5 million. The obtained deviations are illustrated in Figure 4.9. As can be seen, increasing the training size beyond one million points does not reduce the mean squared logarithmic deviation by more than its indicated uncertainties. On the other hand, decreasing the number of training points below that mark leads to a poorer performance of the neural network, due to the model memorising the small dataset. Thus, a training set size of one million points is retained.

It should be mentioned that one could think about training the algorithm in parallel to the integration phase by feeding the $\mathcal{O}(10^4)$ points generated during each integration iteration to the NN and training it for several epochs until the new set of phase-space points is generated,

effectively employing online learning. However, in the course of this thesis, the entire training set is used to train the neural network, for simplicity.

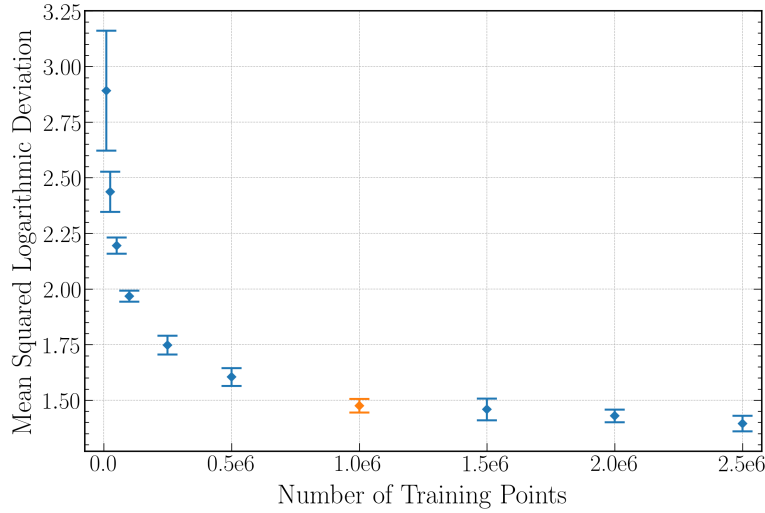


Figure 4.9: Comparison of differing numbers of phase-space points used to train the neural network. The mean squared logarithmic deviations are calculated by taking the mean of five optimisation runs for each training set size. The finally employed number of training points is coloured orange.

Epochs

Finally, the number of epochs, that is the number of complete passes through the entire training data as part of the iterative optimisation procedure, is tweaked. The neural network was trained for 300 epochs in total. The resulting learning curves, averaged over five runs, are displayed in Figure 4.10. The validation error converges around 200 epochs, while the training error is still slightly decreasing due to the model memorising the training set. Training the model beyond that does not improve its prediction capacity, hence, in the following, the networks are trained for 200 epochs overall.

4.2.3 Examination of the Optimised Neural Network

The hyperparameter choices derived in the previous section, see Table 4.1, are used to train the neural networks in the remaining part of this chapter. In the following, the performance of the resulting models, exemplary for the process $gg \rightarrow e^-e^+ggd\bar{d}$, is examined.

In Section 4.2.2, it was already pointed out that the employed NN architecture is able to predict the approximated weight a roughly 800 times faster compared to the time SHERPA needs to evaluate the exact event weight w by calculating the matrix element and the phase-space weight. However, for the success of the Machine Learning based unweighting, a precise prediction capability of the model is just as important. Figure 4.11 shows the distribution of

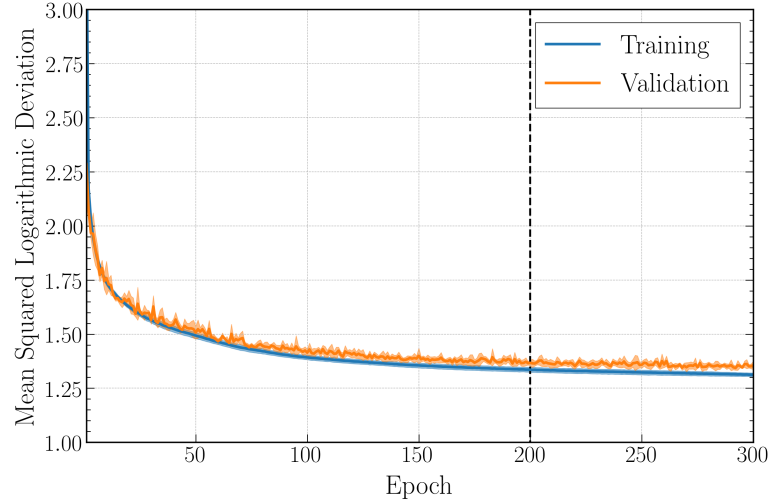


Figure 4.10: Training (blue) and validation (orange) loss of the optimised neural network trained for 300 epochs. The given learning curves are determined by taking the mean obtained from five individual runs. The dashed line illustrates the stopping point for the training of the finally implemented model.

relative weights $x = \frac{w}{a}$ obtained by generating one million weighted events within SHERPA. Roughly 51.4% of the weights x lie in a range of $[0.5, 2.0]$,

$$\frac{\left| \{x \in \mathcal{A} \mid (x > 0.5) \wedge (x < 2)\} \right|}{|\mathcal{A}|} = 0.514 \quad \text{with} \quad \mathcal{A} = \bigcup_i x_i. \quad (4.9)$$

It can be seen, that the model has a slight tendency to overestimate the event weight for this specific process.

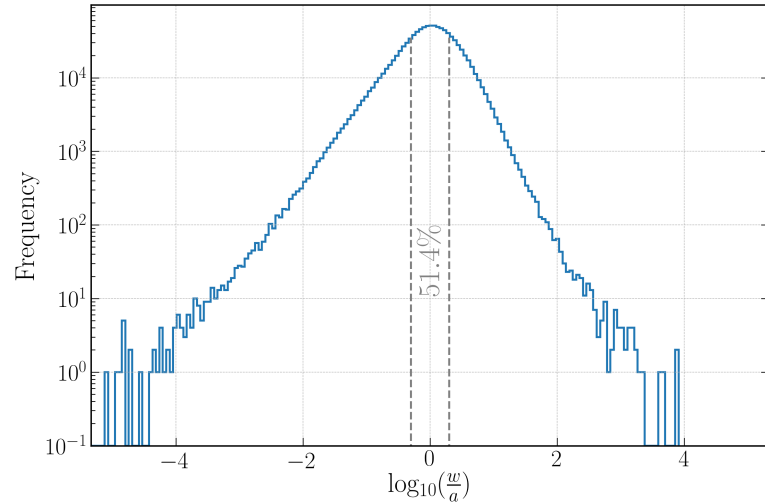


Figure 4.11: Distribution of weights $x = \frac{w}{a}$ using one million test points generated with SHERPA for the process $gg \rightarrow e^-e^+ggd\bar{d}$ at a centre-of-mass energy of 13 TeV. Roughly 51.4% of the weights lie in a range of $[0.5, 2.0]$, illustrated by the dashed lines.

As already discussed, the Machine Learning based unweighting necessitates a second unweighting step. The procedure to determine the second maximum x_{max} is described in Section 4.1.2. Again using one million test points, the evolution of x_{max} with the number of iterations is displayed in Figure 4.12. The value of x_{max} converges slowly and, after 50 iterations, a maximum of 25.32 is received and used in the following.

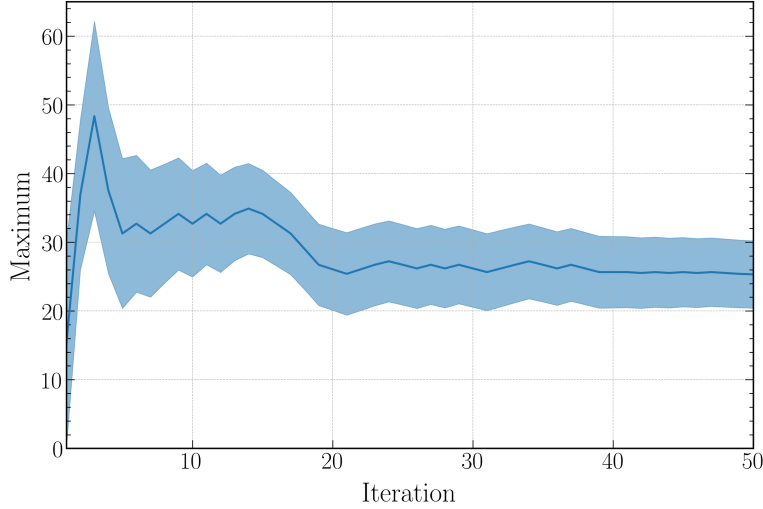


Figure 4.12: Evolution of the second maximum x_{max} with the number of iterations using the procedure as described in Section 4.1.2 for the process $gg \rightarrow e^-e^+gg\bar{d}\bar{d}$. The error band is given by the standard deviation of the respective iterations. One million test points are used for each iteration.

While the previous results try to estimate the performance of the neural network, a true investigation of its capabilities can only be made by running SHERPA with and without the ML based unweighting and comparing the results. The event generation of high-multiplicity processes is CPU intensive, making the use of local devices unfeasible. The simulations are therefore run on the Bull high-performance computing (HPC) cluster Taurus of the ZIH⁶, which is part of the TU Dresden. By default, jobs on Taurus run in shared-mode, i.e. multiple jobs can run on the same compute nodes, leading to interferences spoiling the reliability of time measurements. For benchmarking purposes, an exclusive job allocation is necessary, which ensures that no other jobs run on the exploited computing nodes.

Table 4.2 lists the results obtained for various random seeds using exclusive job allocation. The times per (partially) unweighted event are determined by generating one thousand events per random seed using the “classic” and the ML based unweighting prescription within SHERPA. The second unweighting efficiencies ε_2 and effective factors f_{eff} that the ML approach is faster than plain SHERPA are calculated using Equation (4.2) and (4.4), respectively. It is apparent that the second unweighting efficiencies are close to one for all investigated random seeds. This is reflected by the relative event weight distribution shown in Figure 4.13. Consequently, most

⁶See <https://tu-dresden.de/zih/hochleistungsrechnen/hpc> for details.

events do not exceed the second maximum. Those exceeding x_{max} are rare and do so only by a small factor. Overall, the Machine Learning approach accelerates the unweighted event generation by a factor of 5.8 for $gg \rightarrow e^-e^+ggd\bar{d}$. The time needed to train the neural network and to determine the second maximum x_{max} are considered to be negligible. This is justified since the training process takes roughly one hour on a single CPU core (Intel® Core™ i5-6200U CPU@2.30GHz) and could easily be speeded up by utilising GPUs. Taking these additional times into account, the ML approach would catch up with plain SHERPA after less than 300 events and will outperform it significantly afterwards.

Random Seed	t_{SHERPA} per unw. event [s]	t_{ML} per (part.) unw. event [s]	ε_2	f_{eff}
1234	24.96	4.16	0.94	5.67
2468	22.05	4.00	0.99	5.47
3702	22.41	4.14	0.99	5.38
4936	25.48	4.09	0.99	6.16
6170	24.98	3.83	0.96	6.23
7404	24.68	4.09	0.99	6.01
8638	24.56	3.97	0.98	6.08
9872	21.89	4.07	0.99	5.30
11106	23.69	3.92	0.99	6.00
12340	23.62	3.92	0.98	5.92

Table 4.2: Average times needed to generate one (partially) unweighted event for the process $gg \rightarrow e^-e^+ggd\bar{d}$ in SHERPA at a centre-of-mass energy of 13 TeV with and without the Machine Learning approach for different random seeds. Generating one thousand events per random seed and using exclusive job allocation, the timings t , second unweighting efficiencies ε_2 and the resulting f_{eff} are determined.

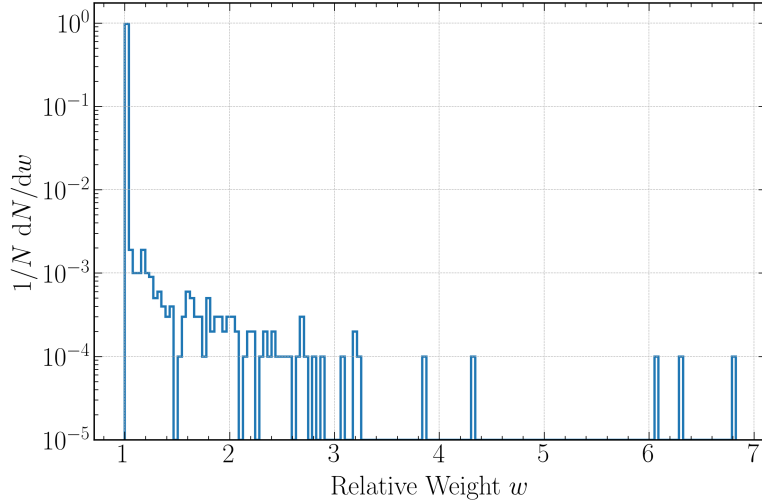


Figure 4.13: Relative event weight distribution of ten thousand events generated using the Machine Learning approach for the process $gg \rightarrow e^-e^+ggd\bar{d}$ at a centre-of-mass energy of 13 TeV. The relative weights w_i are calculated after the second unweighting step employing Equation (4.5).

4.3 Results

So far, the optimisation of the neural network and its resulting performance were described using the example of the process $gg \rightarrow e^-e^+ggd\bar{d}$. This section presents results for other processes belonging to the $Z+4$ jets, $W+4$ jets and $t\bar{t}+3$ jets process groups. In addition, only time improvements were studied until now, without verifying that the physics description is not altered by the use of the Machine Learning based unweighting approach. Hence, a confrontation of important Monte Carlo validation observables between SHERPA with and without the ML based unweighting method is illustrated for one exemplary process for each group in the following. It is worth mentioning that all presented time improvements can likewise be translated into energy savings as no notable additional computing resources are needed for the Machine Learning approach.

SHERPA has two built-in tree-level matrix element generators, namely AMEGIC [58] and COMIX [59]. Within the scope of this thesis, AMEGIC is used to generate matrix elements in the context of the Machine Learning based unweighted.

4.3.1 Z +jets

Performance Improvements

The optimisation procedure of the neural network was described in detail in Section 4.2.2. The hyperparameters are tuned with respect to one illustrative process as an individual tuning is not practicable. A priori, it is not clear how well these parameter choices will work for other processes.

Results obtained for other processes belonging to the $Z+4$ jets group as well are presented in Table 4.3. The cross-sections σ are obtained by generating between $5 \cdot 10^3$ and $5 \cdot 10^4$ unweighted events in SHERPA with and without the ML approach. The respective processes are majorly selected because of their comparatively high cross-sections, hence making up the bulk of this specific process group. One process with a relatively low cross-section, namely $u\bar{u} \rightarrow e^-e^+gggg$, is investigated in the interests of comparability.

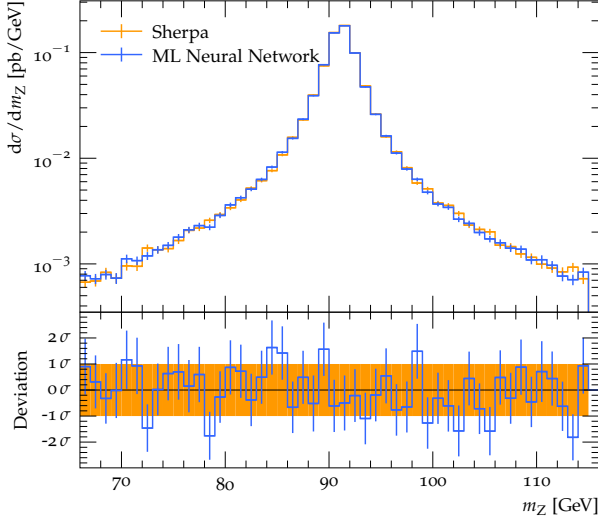
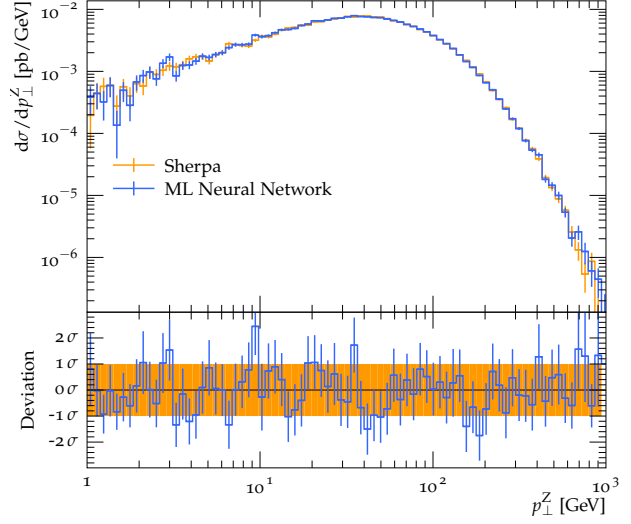
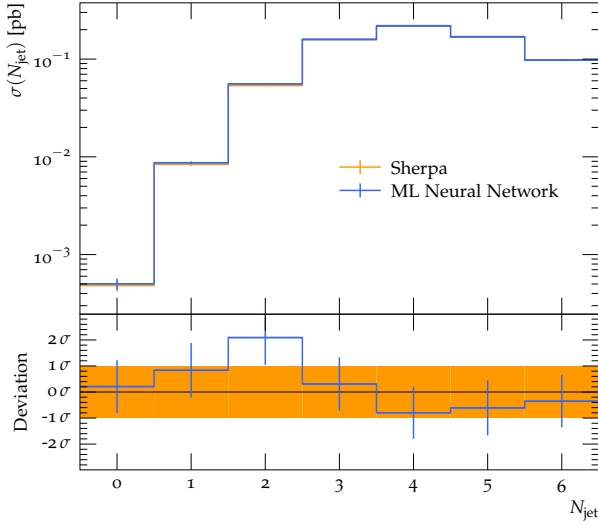
All considered process are speeded up by the use of the ML prescription. The time improvements vary between a few percent and a factor larger than eight. In general, the processes experience significant time improvements. The neural network has difficulties approximating the weights of $du \rightarrow e^-e^+ggdu$, reflected by a comparatively high second maximum x_{max} . A reason for this is that SHERPA seems to have problems in sampling this specific process, resulting in a broader weight distribution making it increasingly difficult for the model to learn it.

Process	x_{max}	σ_{SHERPA} [pb]	σ_{ML} [pb]	$t_{\text{SHERPA}}/t_{\text{ML}}$	f_{eff}
$ug \rightarrow e^-e^+gggu$	44.26	8.82(6)	8.84(6)	6.4(3)	6.1(3)
$gd \rightarrow e^-e^+gggd$	30.06	5.34(3)	5.40(4)	9.3(5)	8.4(6)
$du \rightarrow e^-e^+ggdu$	52.55	1.614(10)	1.619(12)	1.33(12)	1.14(13)
$gg \rightarrow e^-e^+ggd\bar{d}$	25.32	1.272(8)	1.281(8)	5.9(4)	5.8(4)
$u\bar{u} \rightarrow e^-e^+gggg$	41.35	0.194(3)	0.191(4)	8.74(14)	6.2(3)

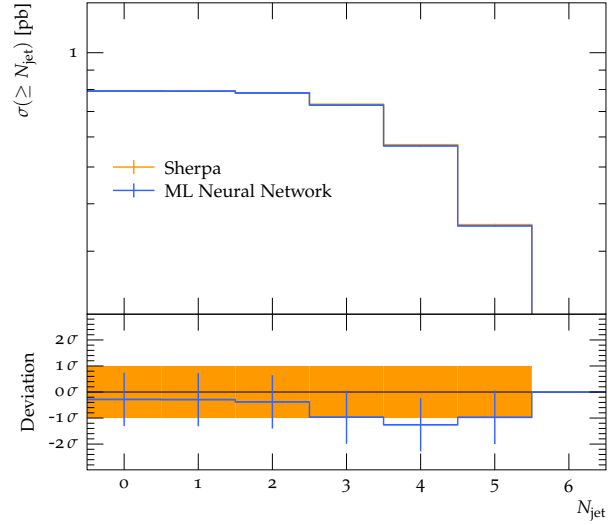
Table 4.3: Results for processes belonging to the Z +jets group for pp collisions at a centre-of-mass energy of 13 TeV using the NNPDF 3.0 NNLO PDF set [60]. An invariant mass cut of $m_{e^-e^+} > 66$ GeV is applied and jets are defined using the anti- k_T clustering algorithm with $R = 0.4$ and $p_T > 20$ GeV. The cross-sections σ are obtained by generating between $5 \cdot 10^3$ and $5 \cdot 10^4$ unweighted events in SHERPA with and without the ML approach. Using different random seeds and exclusive job allocation, the timings t , second unweighting efficiencies ϵ_2 and the resulting f_{eff} are determined.

Physics Validation

The discussed time improvements would be meaningless if the Machine Learning procedure influences the physics description, making it unusable for practical usage. To verify the validity of the Machine Learning based unweighting, results for different relevant observables are investigated. Figures 4.14 to 4.15 show various important Monte Carlo validation observables comparing the physics description of SHERPA with and without the ML approach, exemplarily for the process $gg \rightarrow e^-e^+ggd\bar{d}$. As can be seen, the prediction of SHERPA and ML SHERPA agree with each other within the given uncertainties. The statistical deviations can be reduced by generating more events.

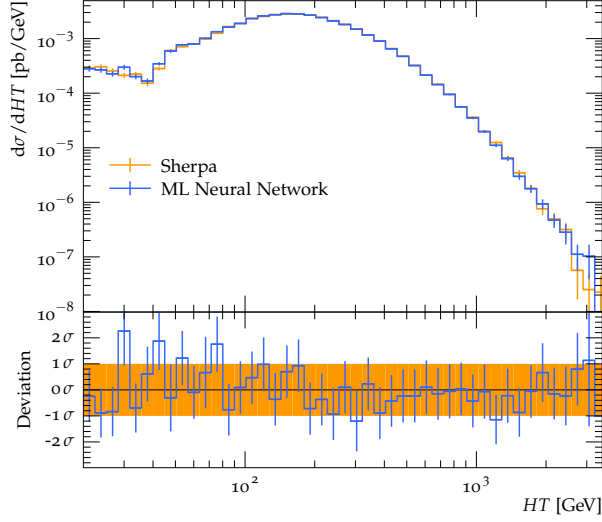
(a) Z mass m_Z (b) Transverse momentum of Z p_T^Z 

(c) Exclusive jet multiplicity

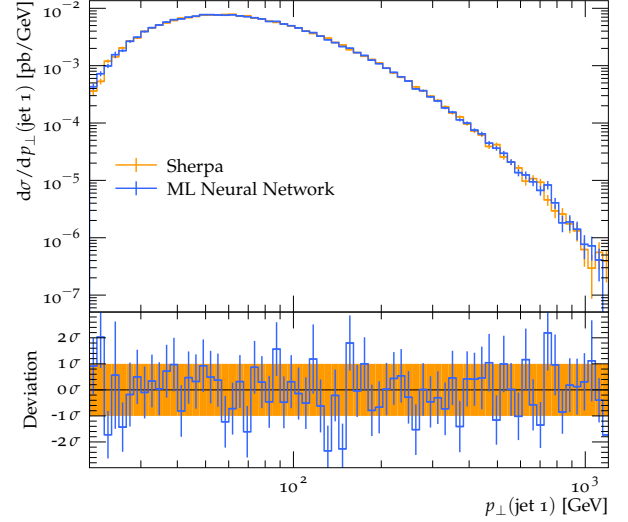


(d) Inclusive jet multiplicity

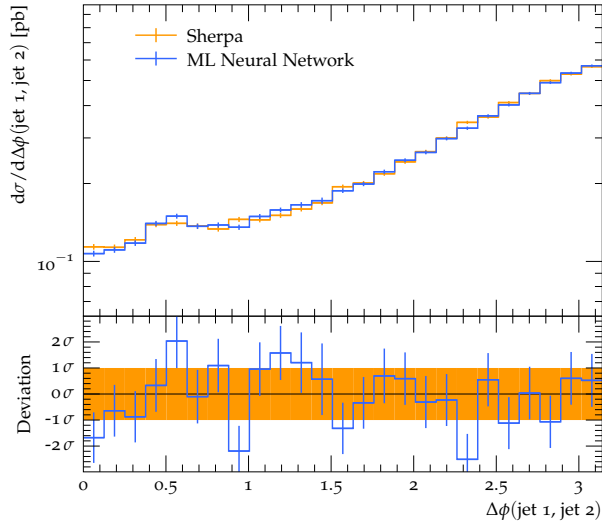
Figure 4.14: Comparison of different Monte Carlo validation observables generated using SHERPA with (blue) and without (orange) the Machine Learning approach for the process $gg \rightarrow e^-e^+ggd\bar{d}$ at a centre-of-mass energy of 13 TeV. An invariant mass cut of $m_{e^-e^+} > 66$ GeV is applied and jets are defined using the anti- k_T clustering algorithm with $R = 0.4$ and $p_T > 20$ GeV.



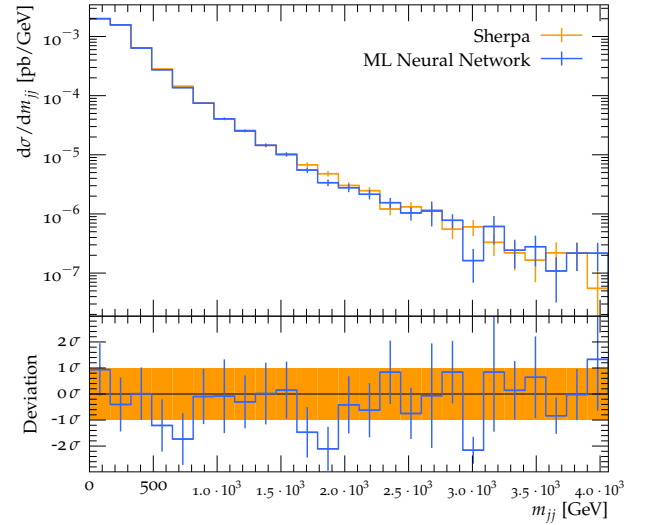
(a) Scalar sum of jet transverse momenta HT



(b) Transverse momentum of leading jet $p_T(\text{jet } 1)$



(c) Azimuthal separation between jets $\Delta\phi(\text{jet } 1, \text{jet } 2)$



(d) Dijet invariant mass spectrum m_{jj}

Figure 4.15: Comparison of different Monte Carlo validation observables generated using SHERPA with (blue) and without (orange) the Machine Learning approach for the process $gg \rightarrow e^-e^+ggd\bar{d}$ at a centre-of-mass energy of 13 TeV. An invariant mass cut of $m_{e^-e^+} > 66$ GeV is applied and jets are defined using the anti- k_T clustering algorithm with $R = 0.4$ and $p_T > 20$ GeV.

4.3.2 W +jets

Up to now, only processes belonging to the Z +4 jets group were examined. In order to check how well the ML procedure generalises to other processes, this section investigates its utility for the production of W bosons in association with jets, where the same Neural Network architecture as described in Section 4.2.2 is used. Table 4.4 shows results for W +4 jets processes.

Performance Improvements

Again, significant time savings could be achieved for all evaluated processes. In general, the NN models are not able to learn the underlying weight distribution as well as in the case of Z +jets, reflected by the slightly higher values obtained for the second maximum x_{max} . Nevertheless, all investigated processes exhibit major performance improvements of at least 90%.

Process	x_{max}	σ_{SHERPA} [pb]	σ_{ML} [pb]	$t_{\text{SHERPA}}/t_{\text{ML}}$	f_{eff}
$dg \rightarrow e^- \bar{\nu}_e g g g u$	47.38	46.6(5)	47.1(6)	9.5(8)	8.6(10)
$gg \rightarrow e^- \bar{\nu}_e g g u \bar{d}$	34.54	11.37(7)	11.46(11)	4.29(11)	4.0(5)
$dd \rightarrow e^- \bar{\nu}_e g g d u$	55.46	8.01(5)	7.97(7)	2.6(3)	1.9(3)
$du \rightarrow e^- \bar{\nu}_e g g u u$	47.74	7.42(7)	7.42(8)	3.28(13)	2.6(3)
$d\bar{u} \rightarrow e^- \bar{\nu}_e g g g g$	53.84	0.908(13)	0.897(23)	5.4(3)	3.8(6)

Table 4.4: Results for processes belonging to the W +jets group for pp collisions at a centre-of-mass energy of 13 TeV using the NNPDF 3.0 NNLO PDF set [60]. Jets are defined using the anti- k_T clustering algorithm with $R = 0.4$ and $p_T > 20$ GeV. The cross-sections σ are obtained by generating between $5 \cdot 10^3$ and $5 \cdot 10^4$ unweighted events in SHERPA with and without the ML approach. Using different random seeds and exclusive job allocation, the timings t , second unweighting efficiencies ε_2 and the resulting f_{eff} are determined.

Physics Validation

In order to verify that the Machine Learning based unweighting preserves the physics modelling of SHERPA in the case of W boson production as well, various Monte Carlo validation observables are examined and illustrated in Figures 4.16 to 4.17 using the example of the process $gg \rightarrow e^- \bar{\nu}_e g g u \bar{d}$. Again, both unweighting prescriptions are compatible with each other.

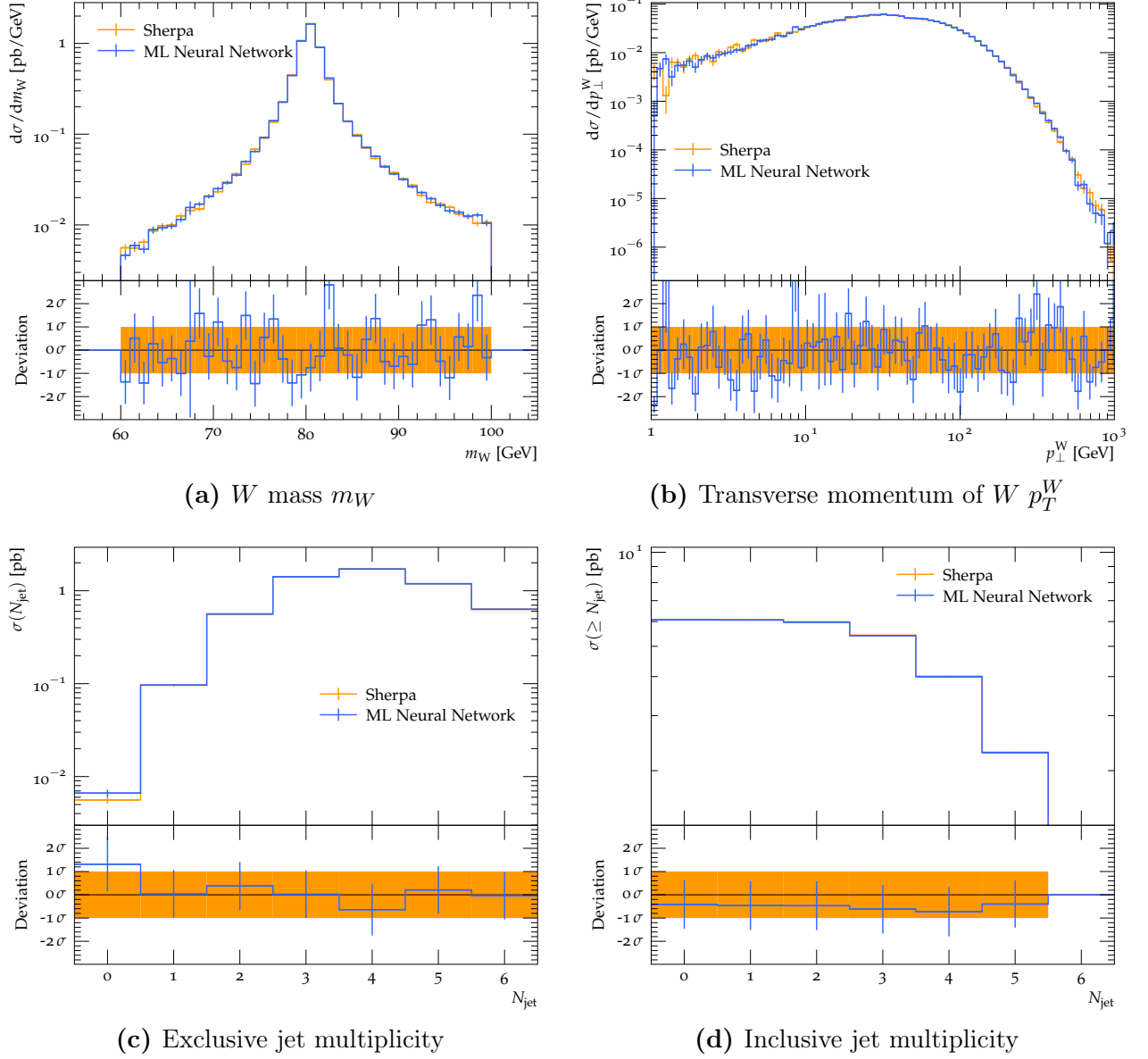
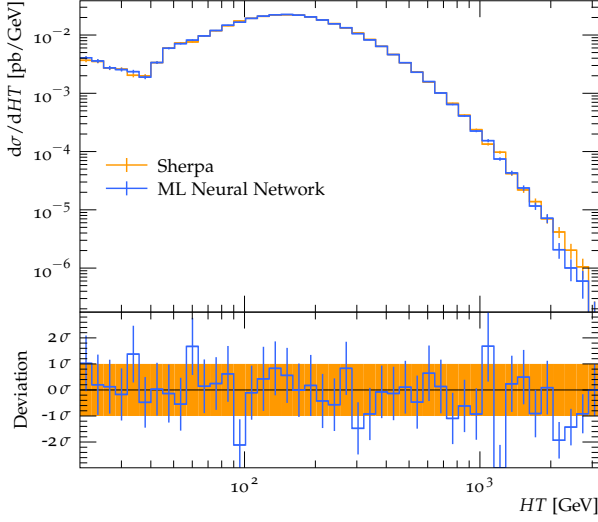
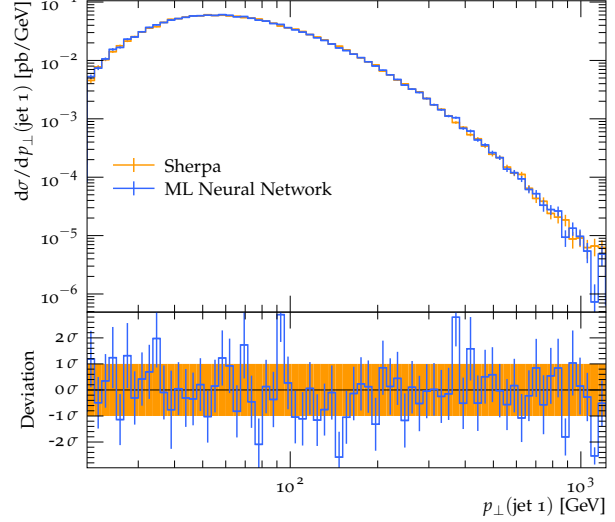


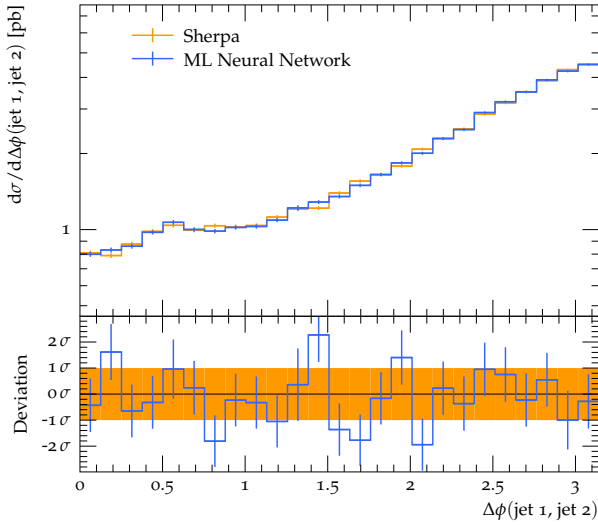
Figure 4.16: Comparison of different Monte Carlo validation observables generated using SHERPA with (blue) and without (orange) the Machine Learning approach for the process $gg \rightarrow e^- \bar{\nu}_e gg u \bar{d}$ at a centre-of-mass energy of 13 TeV. Jets are defined using the anti- k_T clustering algorithm with $R = 0.4$ and $p_T > 20$ GeV.



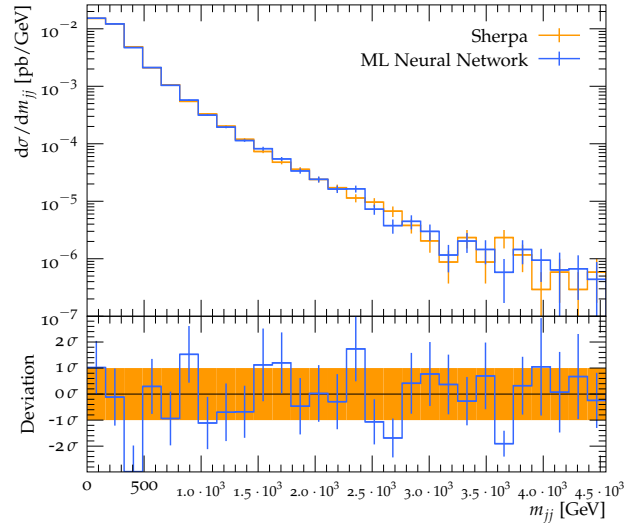
(a) Scalar sum of jet transverse momenta HT



(b) Transverse momentum of leading jet $p_T(\text{jet } 1)$



(c) Azimuthal separation between jets $\Delta\phi(\text{jet } 1, \text{jet } 2)$



(d) Dijet invariant mass spectrum m_{jj}

Figure 4.17: Comparison of different Monte Carlo validation observables generated using SHERPA with (blue) and without (orange) the Machine Learning approach for the process $gg \rightarrow e^- \bar{\nu}_e gg u \bar{d}$ at a centre-of-mass energy of 13 TeV. Jets are defined using the anti- k_T clustering algorithm with $R = 0.4$ and $p_T > 20$ GeV.

4.3.3 $t\bar{t}$ +jets

Finally, the performance of processes belonging to the $t\bar{t}$ +3 jets group are examined, effectively checking the generalisation capability beyond the production of electroweak bosons in association with jets. Even though the top quarks masses are not neglected during calculations, the same neural network architecture using the three-momenta as input features is utilised.

Performance Improvements

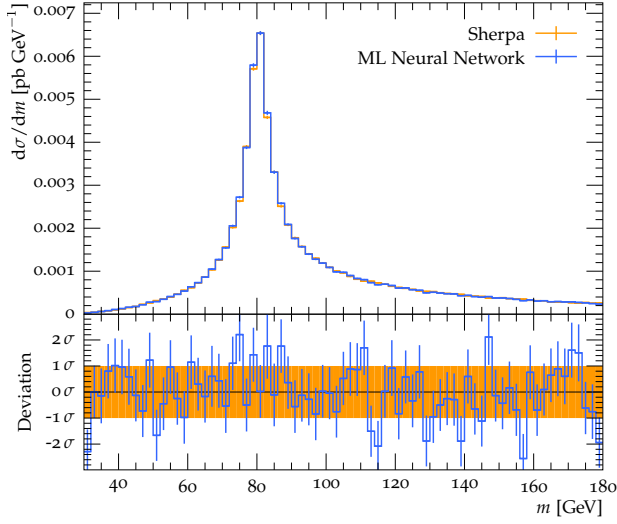
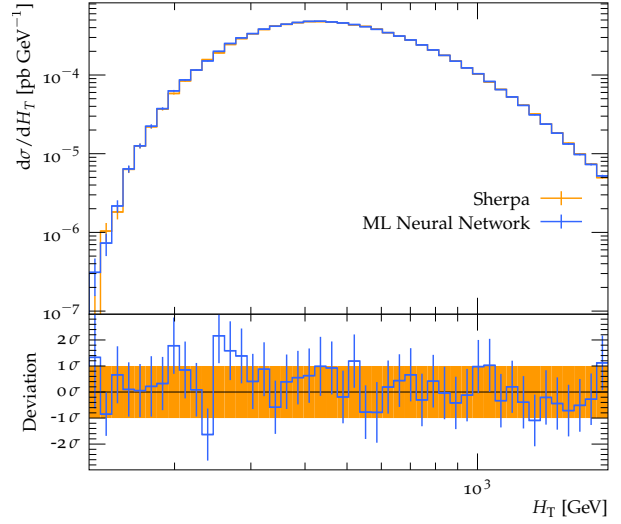
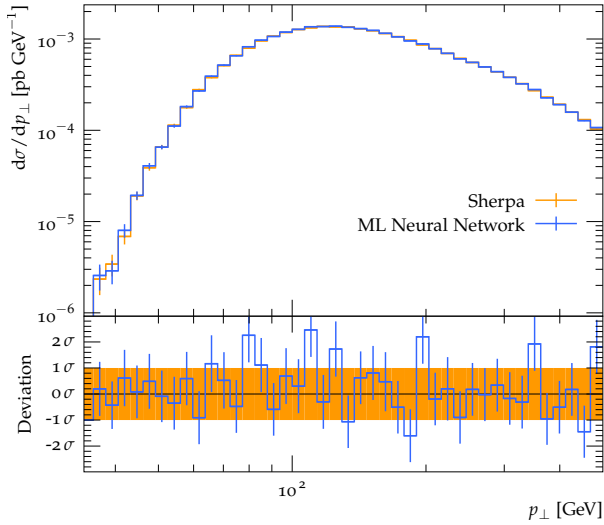
The results are listed in Table 4.5. All processes receive major performance improvements of at least 83%. In the case of $ug \rightarrow t\bar{t}ggu$, the unweighting speed could be increased by a factor of 23.6.

Process	x_{max}	σ_{SHERPA} [pb]	σ_{ML} [pb]	$t_{\text{SHERPA}}/t_{\text{ML}}$	f_{eff}
$ug \rightarrow t\bar{t}ggu$	20.78	48.9(7)	47.7(7)	24.0(27)	23.6(27)
$dg \rightarrow t\bar{t}ggd$	16.84	21.9(3)	21.6(4)	15.2(15)	14.3(15)
$uu \rightarrow t\bar{t}guu$	30.43	6.09(4)	6.02(4)	3.63(25)	3.50(27)
$du \rightarrow t\bar{t}gdu$	38.78	2.888(13)	2.895(13)	1.85(8)	1.83(8)

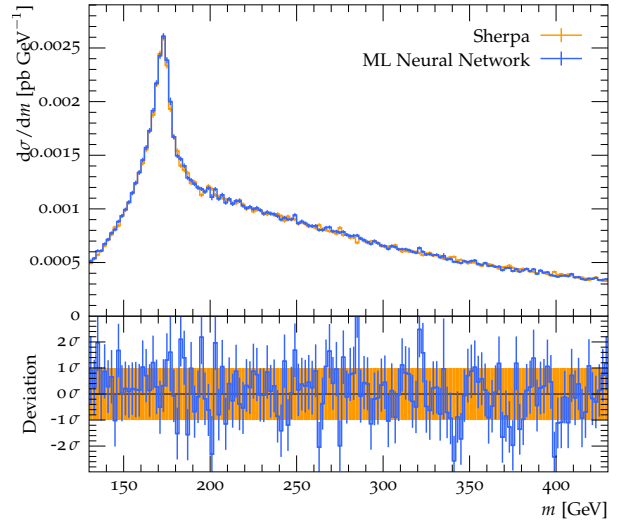
Table 4.5: Results for processes belonging to the $t\bar{t}$ +jets group for pp collisions at a centre-of-mass energy of 13 TeV using the NNPDF 3.0 NNLO PDF set [60]. Jets are defined using the anti- k_T clustering algorithm with $R = 0.4$ and $p_T > 20$ GeV. The cross-sections σ are obtained by generating between $5 \cdot 10^3$ and $5 \cdot 10^4$ unweighted events in SHERPA with and without the ML approach. Using different random seeds and exclusive job allocation, the timings t , second unweighting efficiencies ε_2 and the resulting f_{eff} are determined.

Physics Validation

Figures 4.18 and 4.19 illustrate several Monte Carlo validation observables for $t\bar{t}$ production in association with jets, exemplarily for the process $uu \rightarrow t\bar{t}guu$. As can be seen, the results obtained for SHERPA with and without the ML based unweighting are again in perfect agreement with each other.

(a) Mass distribution for W boson(b) H_T distribution for all jets

(c) Transverse momentum distribution for jet 1



(d) Mass distribution for reconstructed top

Figure 4.18: Comparison of different Monte Carlo validation observables generated using SHERPA with (blue) and without (orange) the Machine Learning approach for the process $uu \rightarrow t\bar{t}g u u$ at a centre-of-mass energy of 13 TeV. Jets are defined using the anti- k_T clustering algorithm with $R = 0.4$ and $p_T > 20$ GeV.

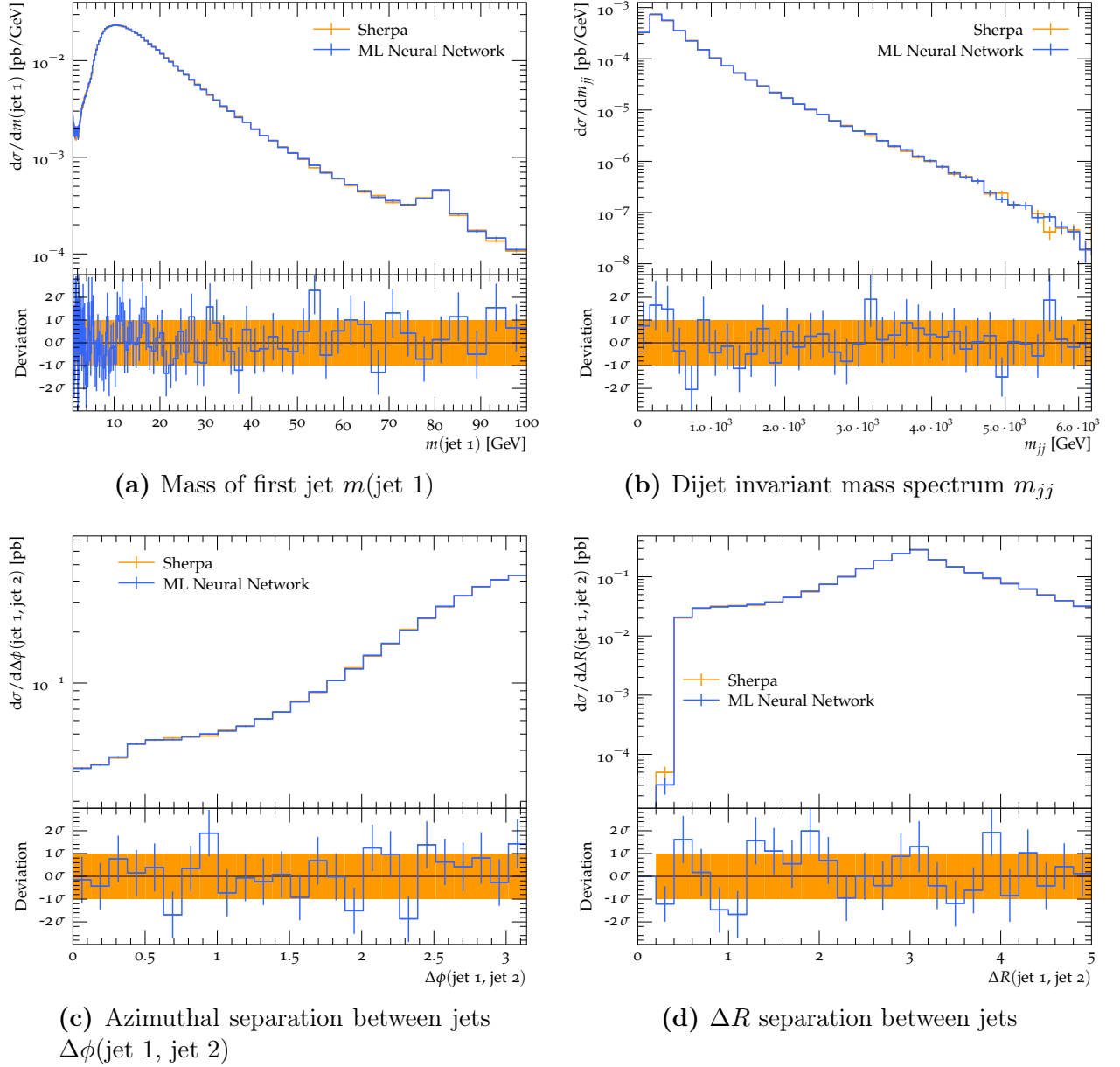


Figure 4.19: Comparison of different Monte Carlo validation observables generated using SHERPA with (blue) and without (orange) the Machine Learning approach for the process $uu \rightarrow t\bar{t}g uu$ at a centre-of-mass energy of 13 TeV. Jets are defined using the anti- k_T clustering algorithm with $R = 0.4$ and $p_T > 20$ GeV.

5 Reduction of Negative Weights

The various efforts in recent years to improve the predictive power of Monte Carlo event generators is accompanied by an increased need of computational resources. One major reason is the effect of negative weighted events, briefly mentioned in Section 3.3. While negative weights are not a problem from a theoretical point of view, they are a major concern with regard to the statistical uncertainty of Monte Carlo event samples. Significantly more events need to be generated to achieve a reasonable statistical accuracy if a large fraction of events is negative weighted. In the following, the origin of as well as mechanisms to decrease the amount of negative weights are reviewed. In particular, the focus hereafter is on Z +jets and $t\bar{t}$ +jets process setups, since they constitute a major portion of Monte Carlo samples produced for contemporary collider experiments.

5.1 Origin and Problem of Negative Weights

Negative weighted events are unavoidable when going beyond the leading order in theoretical calculations. First of all, the real and virtual contributions of next-to-leading order computations are separately divergent, as discussed in Section 3.3. This introduces subtraction terms which inevitably lead to negative weights. Furthermore, negative weights are a distinctive feature of MC@NLO-type matching procedures, which aim for a consistent combination of NLO calculations and parton shower algorithms. In the context of the S-MC@NLO prescription, the additional analytic weights may become negative due to sub-leading colour configurations.

Samples containing negative weighted events have a, comparatively, reduced statistical power. Considering a negative weight fraction ε and, up to a sign, equally weighted events with weights $w = \pm c$, where c is a constant larger than zero, the statistical accuracy of an event sample of size N can be evaluated as

$$\frac{\sqrt{\sum_i w_i^2}}{\sum_i w_i} = \frac{\sqrt{c^2 \cdot N}}{c \cdot (1 - 2\varepsilon) \cdot N} = \frac{1}{(1 - 2\varepsilon)} \cdot \frac{1}{\sqrt{N}} = \frac{1}{\sqrt{N_{eff}}}. \quad (5.1)$$

This yields the effective number of events, given by $N_{eff} = (1 - 2\varepsilon)^2 \cdot N$, which incorporates the diminished statistics in the presence of negative weighted events. Hence, we need to generate

a factor

$$f(\varepsilon) = \frac{1}{(1 - 2\varepsilon)^2} \quad (5.2)$$

more events to get the same statistical accuracy in comparison to only positive weighted events. Figure 5.1 illustrates this factor f as a function of the negative weight fraction ε . As can be seen, this factor increases drastically with growing ε . For instance, if a quarter of the events have a weight smaller than zero, a four times larger sample has to be produced. While this raises the computational cost on the event generation side, the more pressing issue are the subsequent detector simulations, making up the major part in terms of CPU consumption. If the negative weight fraction is too large, using full detector simulations becomes impractical. Thus, it is of utter importance to reduce the amount of negative weighted events.

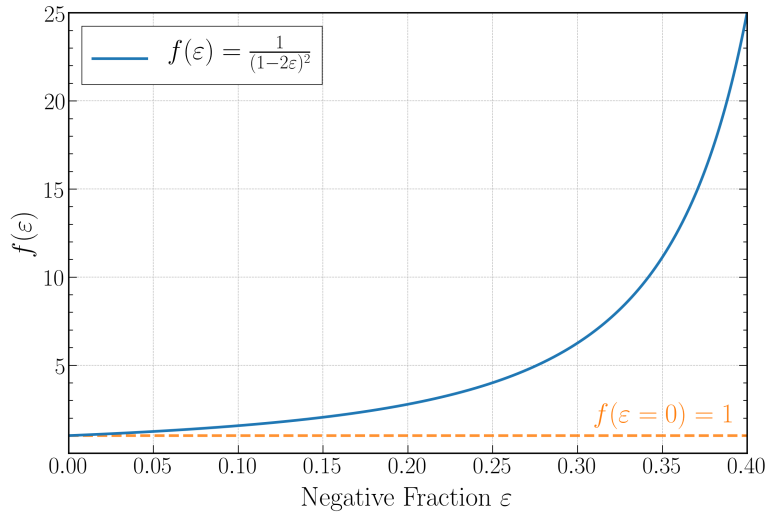


Figure 5.1: *Dependence of the factor f , quantifying the excess of events to be generated for the same statistical accuracy as in the case of solely positive weighted events, on the negative weight fraction ε .*

5.2 Mechanisms to Improve Negative Weight Fraction

5.2.1 Leading Colour Approximation

As can be seen in Table 5.1, S-events take up the major share of the negative weighted events for Z +jets and $t\bar{t}$ +jets processes. Hence, a first approach to decrease the negative weight fraction is to establish (and validate) a leading colour approximation in the matching of NLO calculations with the parton shower. There, sub-leading colour configurations in the real-emission matrix elements are a source of negative weights. Running the usual S-MC@NLO prescription, as described in Section 3.3, in a leading colour mode amounts to dropping the analytic weights introduced in connection with the colour correct one-step parton shower.

This approximation is further motivated by the fact that corrections to the large N_C -limit are typically suppressed by $1/N_C^2$, which would lead to corrections in the order of 10%. However, the effects are usually significantly smaller and can hence be neglected.

	Positive Fraction	Negative Fraction		Positive Fraction	Negative Fraction
$\mathbb{S} + \mathbb{H}$	82%	18%	$\mathbb{S} + \mathbb{H}$	75%	25%
thereof $\mathbb{S}$	88%	58%	thereof $\mathbb{S}$	91%	72%
thereof $\mathbb{H}$	12%	42%	thereof $\mathbb{H}$	9%	28%

(a) $pp \rightarrow Z+0,1,2 \text{ jets@NLO}+3 \text{ jets@LO}$ (b) $pp \rightarrow t\bar{t}+0,1 \text{ jets@NLO}+2,3 \text{ jets@LO}$

Table 5.1: Percentage of positive and negative weighted events and their composition of $\mathbb{S}$ and $\mathbb{H}$ -events for proton-proton collisions at a centre-of-mass energy of 13 TeV.

5.2.2 $\mathbb{H}$ -Event Shower Interplay

The previously discussed leading-colour approximation efficaciously reduces the amount of negative weighted $\mathbb{S}$ -events. A further approach is to allow for a shower veto on the first emission of $\mathbb{H}$ -events in the soft region, effectively treating them as in the merging prescription. This Sudakov suppression reduces the number of $\mathbb{H}$ -events in the soft domain, which, consequently, will be populated more by $\mathbb{S}$ -events. In combination with the above outlined leading colour approximation, this further decreases the negative weight fraction. This, however, does not influence ε in the high- p_T region.

5.2.3 Local K-Factor

The negative weight fraction at next-to-leading order is increasing when going to higher multiplicities in the final-state. In the context of multijet merging employing the MENLOPS method, see Section 3.5, an interpolating K-factor $k_n^{(\mathcal{A})}$ was introduced for the purpose of smooth merging between the matrix elements of varying multiplicity. This K-factor is usually defined on the highest multiplicity described at NLO QCD. By using $k_n^{(\mathcal{A})}$ from the underlying core process instead, the lower multiplicity matrix elements are effectively scaled up and hence contribute more to the total event sample. Considering their, comparatively, smaller negative weight fraction, this further reduces ε in higher transverse momentum region.

Defining the local K-factor on the lowest multiplicity has the drawback that electroweak corrections are effectively not taken into account. The weak bosons need additional partons to recoil against in order to receive any transverse momentum p_T . While the contributions of electroweak corrections to the total cross-section are comparatively small, they can contribute

significantly in the high p_T -region, where they are enhanced due to the appearance of electroweak Sudakov logarithms [61]. Consequently, modifying $k_n^{(\mathcal{A})}$ in such a way may become undesirable if these corrections get large. Hence, the applicability of this method needs to be assessed depending on the considered process.

5.3 Results

This section presents results obtained for the previously discussed mechanisms with the aim of reducing the amount of negative weighted events. More precisely, two process setups are investigated, namely

$$pp \rightarrow Z+0,1,2 \text{ jets@NLO}+3 \text{ jets@LO}$$

and

$$pp \rightarrow t\bar{t}+0,1 \text{ jets@NLO}+2,3 \text{ jets@LO},$$

both evaluated at a centre-of-mass energy of $\sqrt{13}$ TeV using the NNPDF 3.0 NNLO PDF set with $\alpha_S = 0.118$. In the case of Z production, an invariant mass cut of $66 \text{ GeV} < m_{e^-e^+} < 116 \text{ GeV}$ is applied. Furthermore, jets are defined using the anti- k_T clustering algorithm with $R = 0.4$ and $p_T > 20 \text{ GeV}$. The merging cut Q_{cut} is set to 20 GeV in the case of Z production and is equal to 30 GeV for $t\bar{t}$ +jets. All results are obtained using version 2.2.8 of SHERPA with virtual contributions provided by OPENLOOPS version 1.3.1 [62].

In the following, all presented uncertainties correspond to the statistical Monte Carlo error. Perturbative uncertainties can, for example, be accessed by variations of the renormalisation scale μ_R and the factorisation scale μ_F , which are typically scaled up and down by a factor of two.

5.3.1 Z +jets

The study of vector boson production in association with jets plays an important role in the ongoing physics program of the LHC, where they can be measured with high precision and thus enable stringent tests of the Standard Model. Furthermore, these processes contribute as background to some of the most important searches for physics beyond the Standard Model, e.g. dark matter or supersymmetric signatures. As a result, vector boson plus jet processes constitute a vast amount of the Monte Carlo samples produced for the ATLAS and CMS experiments at the LHC, making the reduction of negative weighted events for these processes crucial. In particular, this section focuses on the negative weights of the above outlined Z +jets setup.

	Negative Weight Fraction
Default	18.1%
Leading Colour Mode	14.0%
+ shower veto on $\mathbb{H}$ -events	9.6%
+ local K-factor from core	9.1%

Table 5.2: Evolution of the mean negative weight fraction for $pp \rightarrow Z+0,1,2 \text{ jets@NLO} + 3 \text{ jets@LO}$.

Table 5.2 shows the evolution of the negative weight fraction ε using the approaches introduced in the previous section for Z +jets. As can be seen, ε is reduced by roughly a factor of two, from 18.1% down to 9.1%. A reduction of negative weights is only useful if the physics description is not altered in the sense of poorer predictive power. Figures 5.2 and 5.3 show a comparison of the discussed approaches for various Monte Carlo validation observables. There, the lower two ratio plots display the negative fraction ε and the corresponding factor $f(\varepsilon)$, which indicates how much more events need to be generated on average in order to achieve the same statistical accuracy as compared to only positive weighted events in each bin. All observables show good agreement between default and the different mechanisms.

It is visible that the leading colour mode reduces the number of negative weights in the low p_T region, while it leaves ε unchanged for large transverse momenta. The additional Sudakov suppression on $\mathbb{H}$ -events further decreases the negative fraction for small p_T . Defining the local K-factor on the core process also lowers ε for large transverse momenta, where it nearly approaches zero for the investigated observables. The transverse momentum of the Z boson, shown in Figure 5.2 (a), is one of the standard candles of particle phenomenology and thus of special interest. This particular observable shows deviations around the merging cut $Q_{\text{cut}}=20$ GeV. However, it should again be stressed that perturbative uncertainties are not plotted for clarity. Taking them into account, all curves would agree with each other within in the respective error bands.

It is essential for Monte Carlo event generators to describe the data as good as possible. Measurements of the production of a Z boson in association with jets in proton-proton collisions at a centre-of-mass energy of 13 TeV using 3.16 fb^{-1} of data, collected by the ATLAS experiment at the LHC [63], are compared to SHERPA in Figure 5.4. The Monte Carlo predictions are in good agreement with the data, while again reducing the negative fraction significantly by employing the above discussed methods.

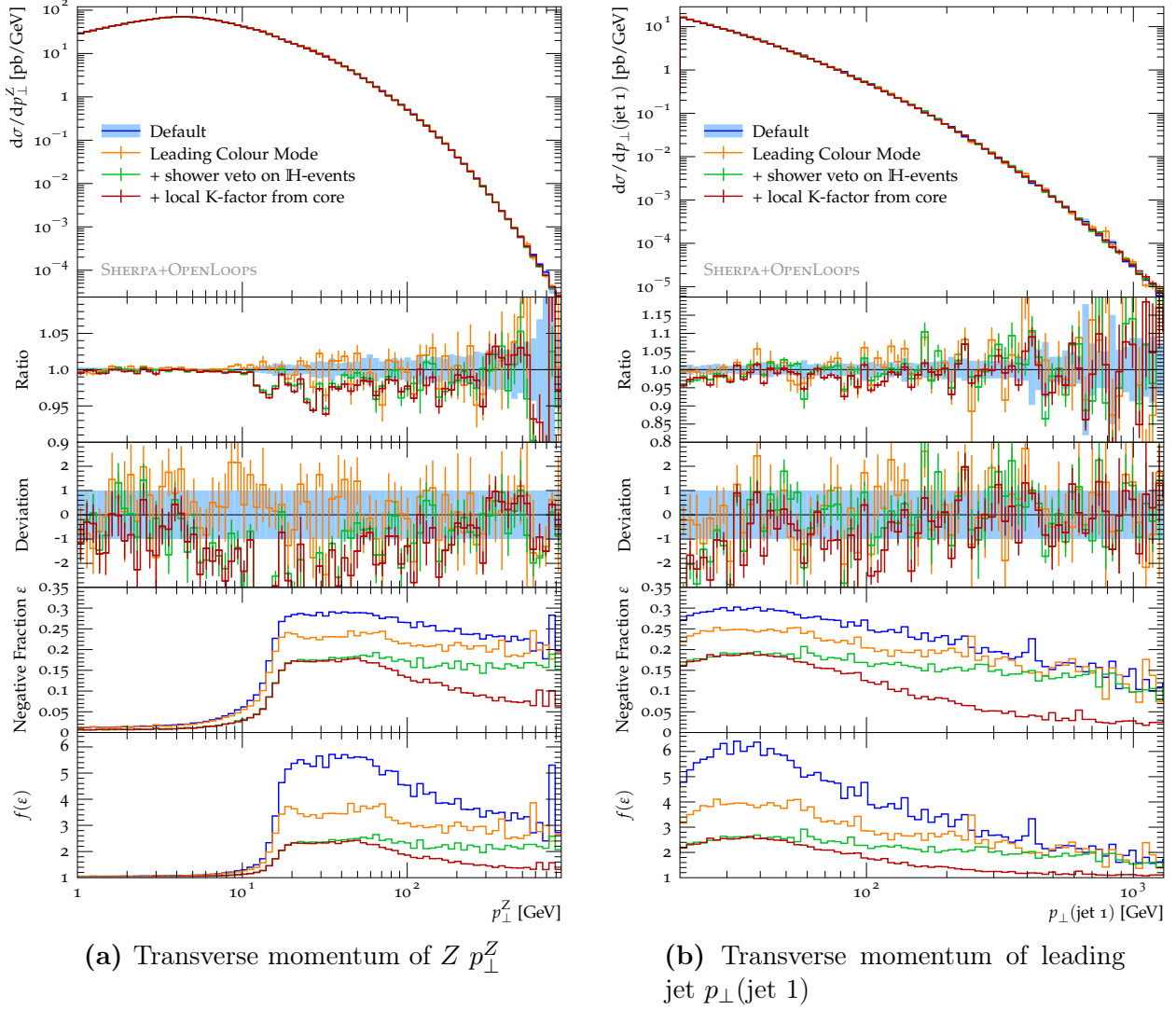


Figure 5.2: Monte Carlo validation observables comparing the different mechanisms to reduce the number of negative weighted events, as discussed in Section 5.2, for $pp \rightarrow Z + 0,1,2 \text{ jets@NLO} + 3 \text{ jets@LO}$. The lower two ratio plots display the negative weight fraction ε and the corresponding factor $f(\varepsilon)$, which indicates how much more events need to be generated on average in order to achieve the same statistical accuracy as compared to only positive weighted events, respectively.

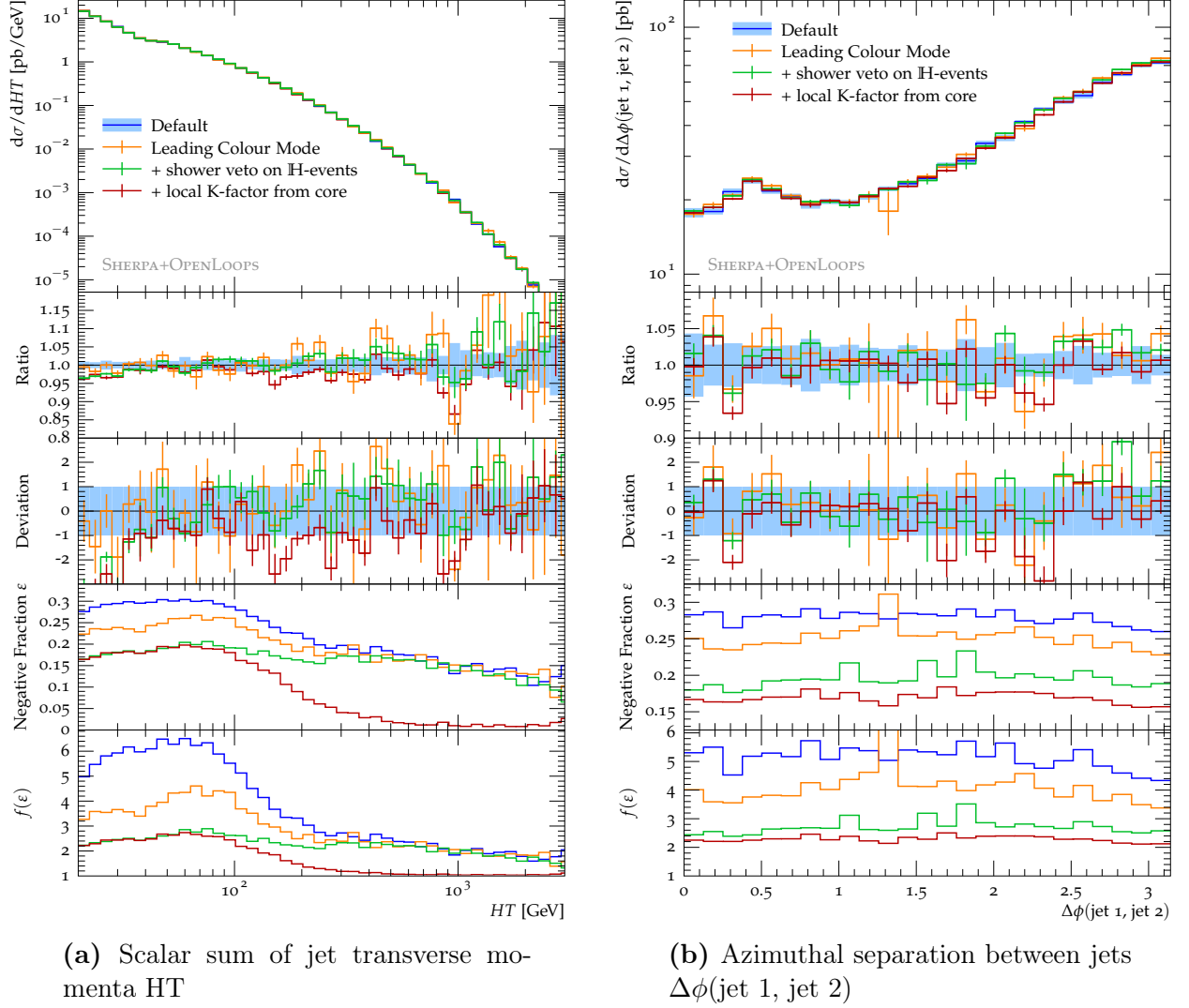
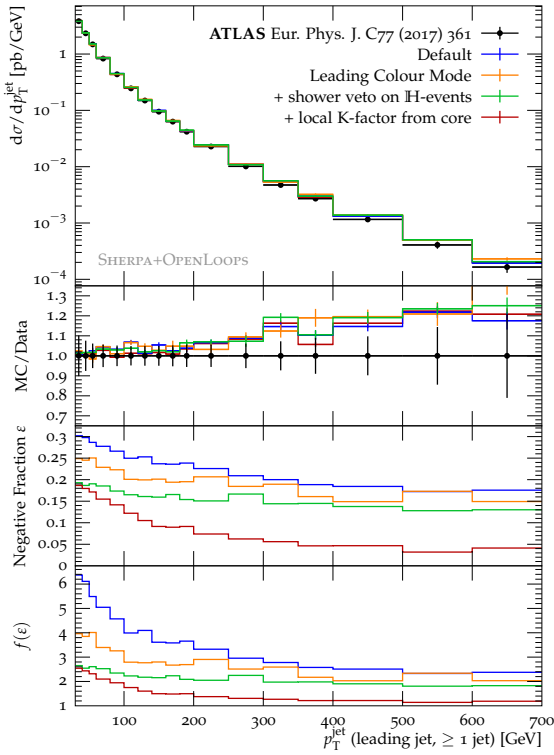
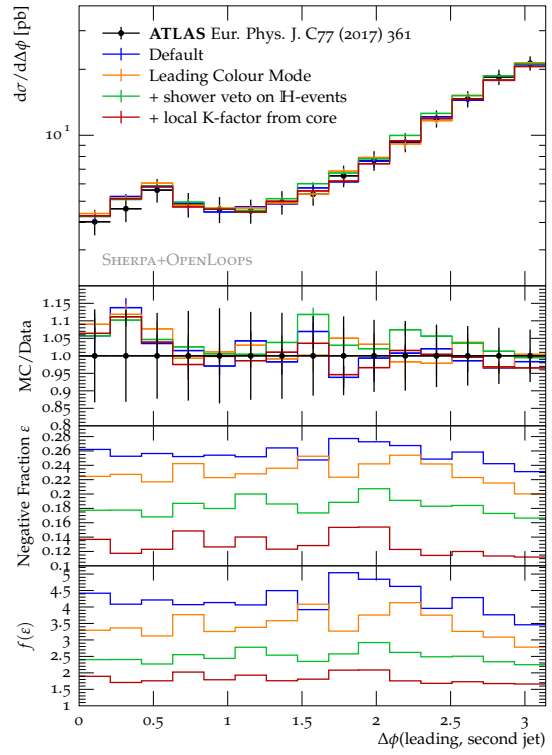


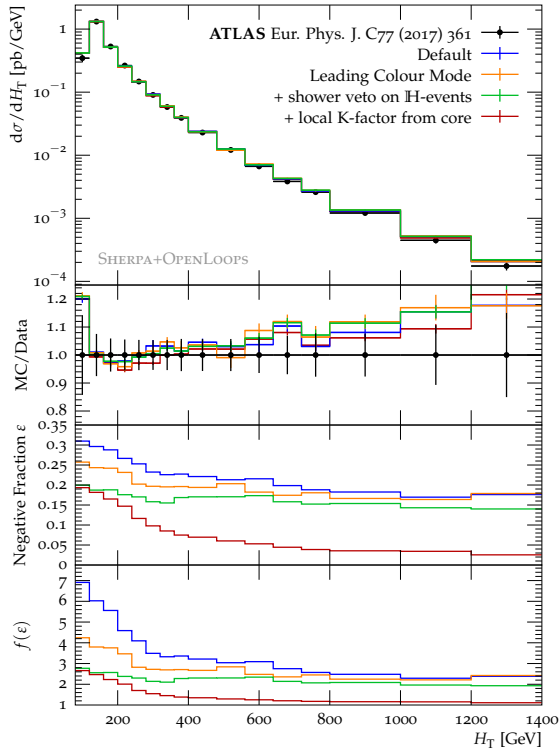
Figure 5.3: Monte Carlo validation observables comparing the different mechanisms to reduce the number of negative weighted events, as discussed in Section 5.2, for $pp \rightarrow Z + 0, 1, 2 \text{ jets@NLO} + 3 \text{ jets@LO}$. The lower two ratio plots display the negative weight fraction ϵ and the corresponding factor $f(\epsilon)$, which indicates how much more events need to be generated on average in order to achieve the same statistical accuracy as compared to only positive weighted events, respectively.



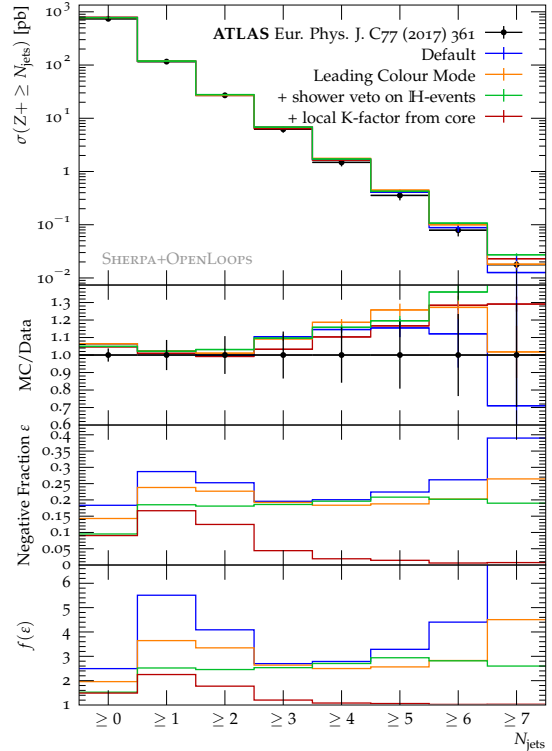
(a) Transverse momentum distribution of leading jet



(b) Azimuthal separation between jets $\Delta\phi(\text{leading, second jet})$



(c) H_T distribution for all jets



(d) Jet multiplicity

Figure 5.4: Comparison of the different mechanisms as described in Section 5.2 to experimental Z +jets data measured by ATLAS [63]. The lower two ratio plots display the negative weight fraction ε and the corresponding factor $f(\varepsilon)$, which indicates how much more events need to be generated on average in order to achieve the same statistical accuracy as compared to only positive weighted events, respectively.

5.3.2 $t\bar{t}$ +jets

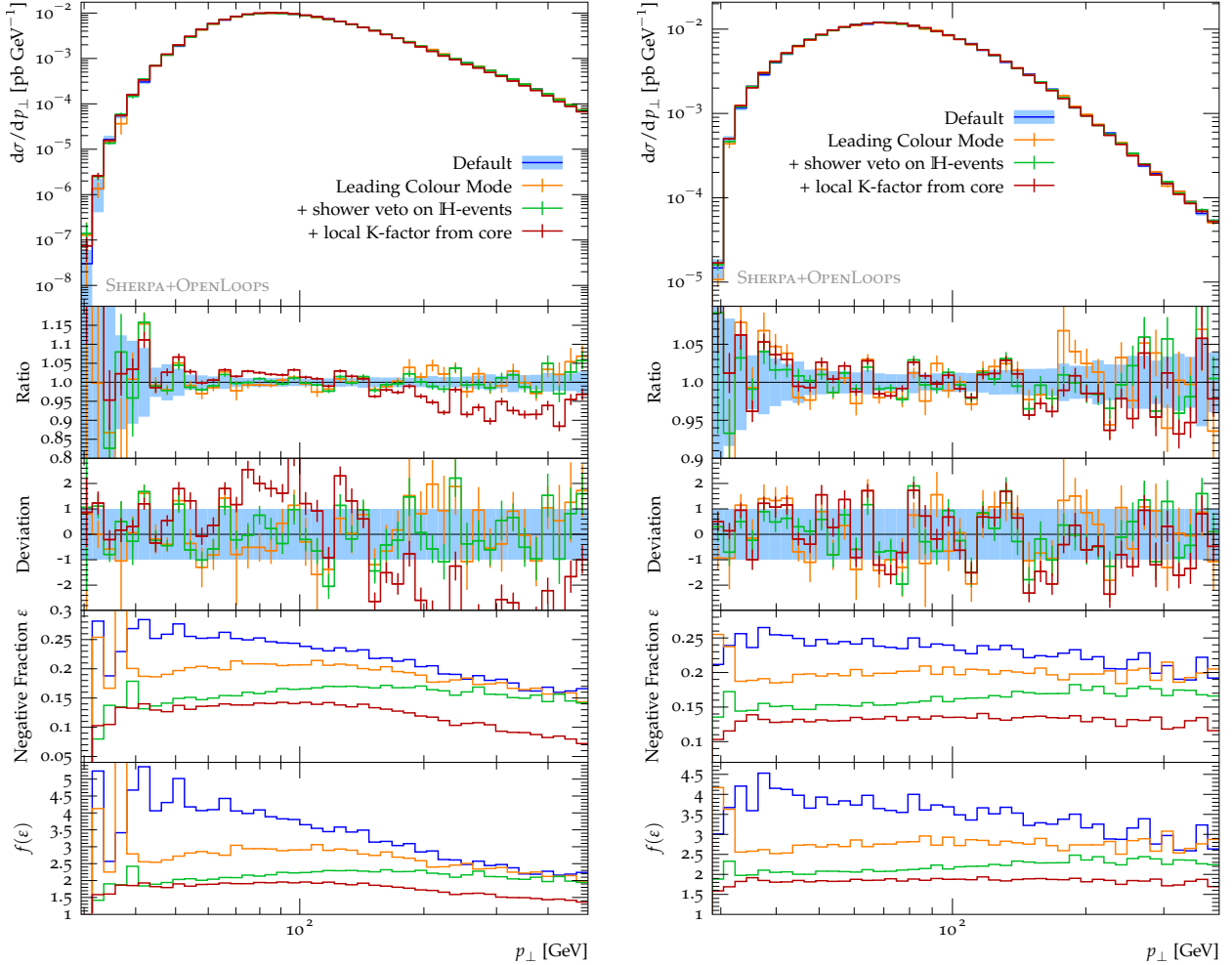
The investigation of top quark pair production and decays in association with jets is crucial in the context of LHC physics. These processes allow for a detailed examination of fundamental parameters of the Standard Model, such as the mass of the top quark. In addition, they represent an essential background for potential physics beyond the Standard Model, making the reduction of negative weights for them desirable. However, the study of these processes is very challenging as they involve heavy quarks carrying colour charges.

	Negative Weight Fraction
Default	24.8%
Leading Colour Mode	18.7%
+ shower veto on $\mathbb{H}$ -events	14.5%
+ local K-factor from core	12.6%

Table 5.3: *Evolution of the mean negative weight fraction for $pp \rightarrow t\bar{t}+0,1 \text{ jets@NLO} +2,3 \text{ jets@LO}$.*

As can be seen from Table 5.3, $t\bar{t}$ +jets samples, as investigated in the scope of this thesis, contain approximately a quarter of events with negative weights. This fraction can roughly be halved using the mechanisms as discussed in Section 5.2. In Figures 5.5 to 5.7, the different approaches aiming for the reduction of ε are compared for various Monte Carlo validation observables. The methods are compatible with the default for all examined observables within the respective errors. Again, the leading colour approximation reduces ε in the soft region, whereas the shower veto on $\mathbb{H}$ -events decreases the negative weighted events in this region even further. The fraction ε can be lowered in the high p_T sector by defining the local K-factor, as needed for smooth merging, of the lowest multiplicity process at NLO.

Figure 5.8 compares the Monte Carlo results to measurements of top quark pair production in association with jets for pp collisions at a centre-of-mass energy of 13 TeV collected by the ATLAS detector, corresponding to an integrated luminosity of 3.2 fb^{-1} [64]. In particular, the transverse momentum of hadronically decaying top quarks as well as of the $t\bar{t}$ -system are investigated, where all tested methods are in agreement with the data.



(a) Transverse momentum distribution for jet 1

(b) Transverse momentum distribution for b -jet 1

Figure 5.5: Monte Carlo validation observables comparing the different mechanisms to reduce the number of negative weighted events, as discussed in Section 5.2, for $pp \rightarrow t\bar{t} + 0,1 \text{ jets@NLO} + 2,3 \text{ jets@LO}$. The lower two ratio plots display the negative weight fraction ε and the corresponding factor $f(\varepsilon)$, which indicates how much more events need to be generated on average in order to achieve the same statistical accuracy as compared to only positive weighted events, respectively.

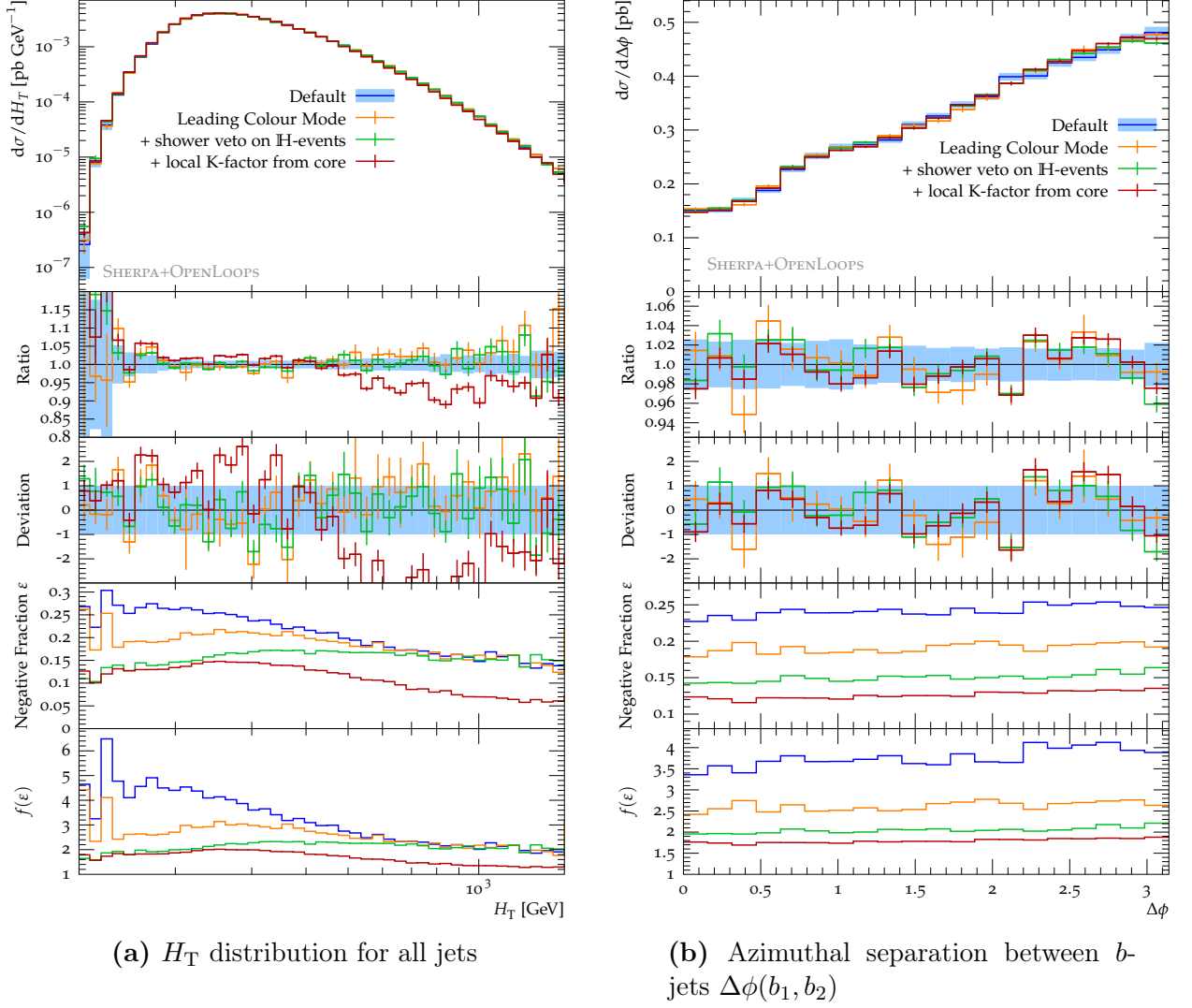
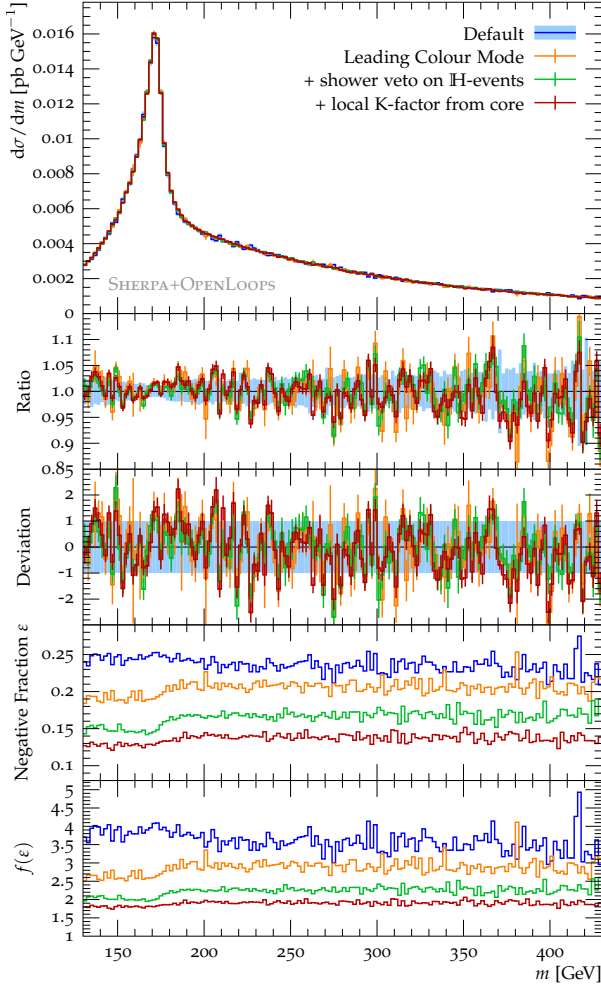
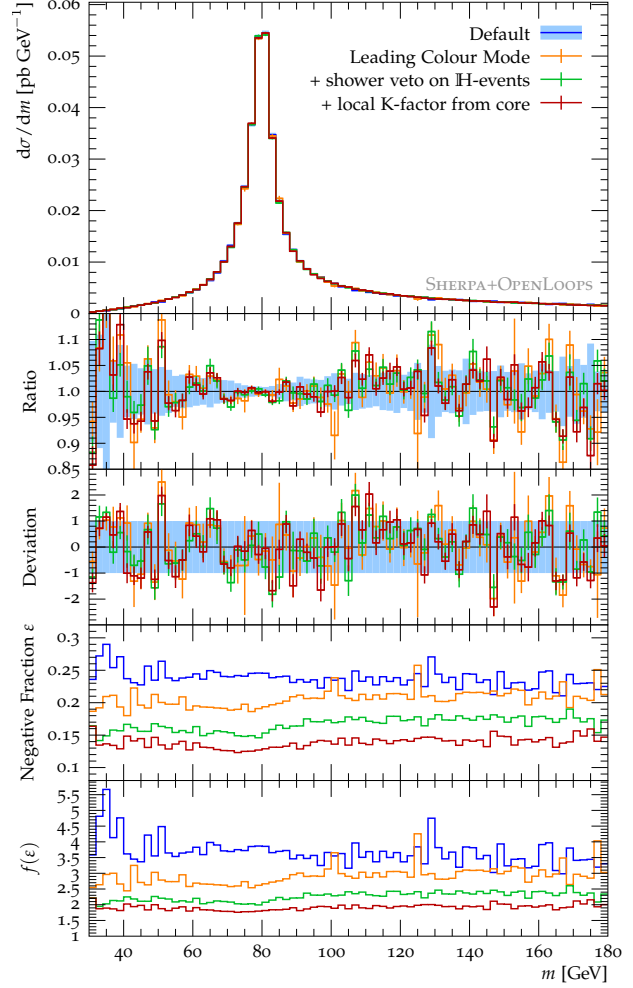


Figure 5.6: Monte Carlo validation observables comparing the different mechanisms to reduce the number of negative weighted events, as discussed in Section 5.2, for $pp \rightarrow t\bar{t} + 0, 1 \text{ jets@NLO} + 2, 3 \text{ jets@LO}$. The lower two ratio plots display the negative weight fraction ϵ and the corresponding factor $f(\epsilon)$, which indicates how much more events need to be generated on average in order to achieve the same statistical accuracy as compared to only positive weighted events, respectively.



(a) Mass distribution for reconstructed top quark



(b) Mass distribution for W bosons

Figure 5.7: Monte Carlo validation observables comparing the different mechanisms to reduce the number of negative weighted events, as discussed in Section 5.2, for $pp \rightarrow t\bar{t} + 0,1 \text{ jets@NLO} + 2,3 \text{ jets@LO}$. The lower two ratio plots display the negative weight fraction ε and the corresponding factor $f(\varepsilon)$, which indicates how much more events need to be generated on average in order to achieve the same statistical accuracy as compared to only positive weighted events, respectively.

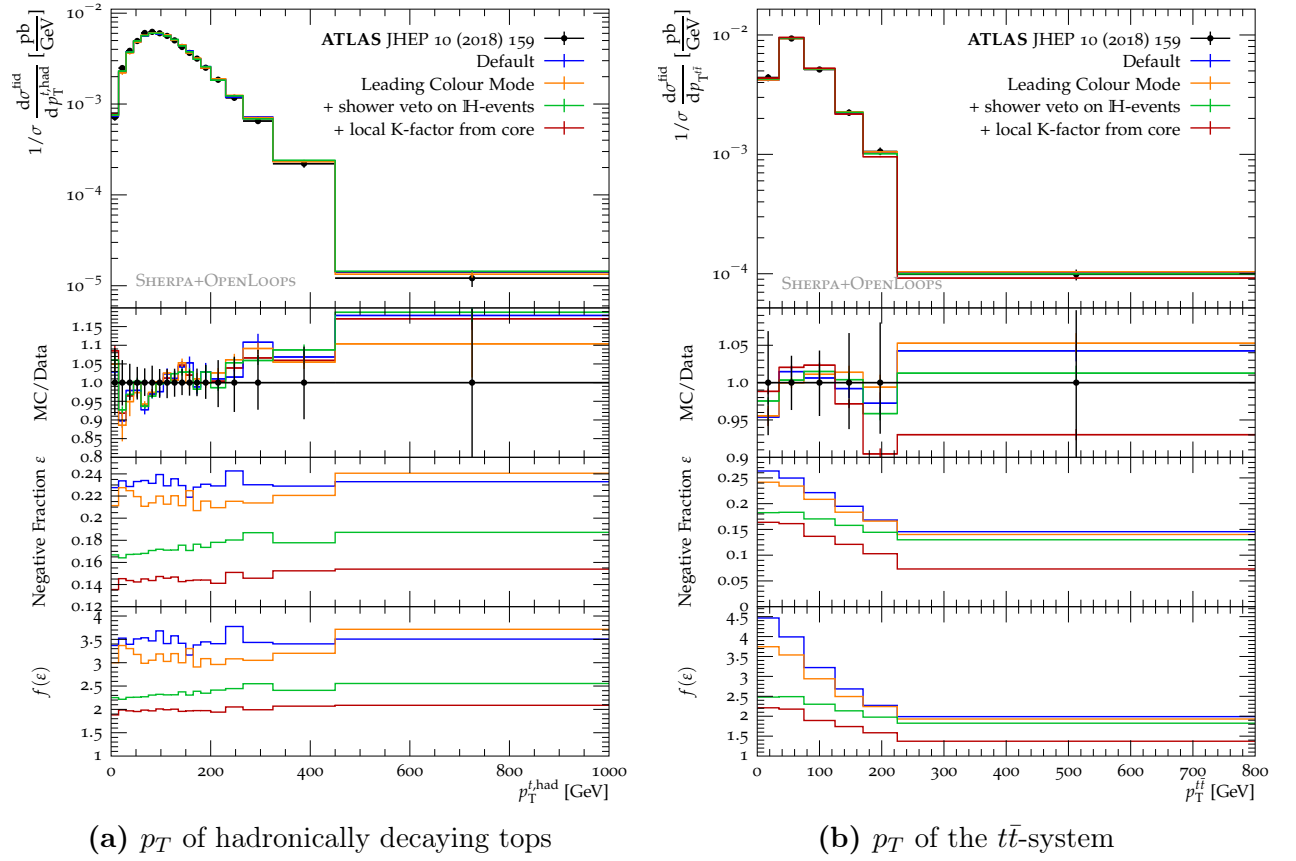


Figure 5.8: Comparison of the different mechanisms as described in Section 5.2 to experimental $t\bar{t}$ +jets data measured by ATLAS [64]. The lower two ratio plots display the negative weight fraction ϵ and the corresponding factor $f(\epsilon)$, which indicates how much more events need to be generated on average in order to achieve the same statistical accuracy as compared to only positive weighted events, respectively.

6 Conclusion and Outlook

Monte Carlo event generators are crucial tools needed for the confrontation of theoretical calculations and experimental measurements. Due to their essential role in particle physics phenomenology, it is of utmost importance that they provide precise predictions while maintaining reasonable computational costs. However, the continual efforts to increase the precision of Monte Carlo event generators comes at the expense of higher CPU consumption.

In this thesis, performance improvements concerning the general-purpose event generator SHERPA, especially for high-multiplicity processes, were investigated. In particular, the focus was on improvements regarding the unweighted event generation speed as well as the reduction of negative weighted events, which naturally arise when going beyond the leading order in the perturbative expansion.

For the former, the so-called Machine Learning based unweighting prescription was implemented in SHERPA and its performance in comparison to the “classic” approach investigated. This method presents itself as an extension of the usual unweighting technique by incorporating Machine Learning models in order to increase the unweighting speed. In this context, various Machine Learning algorithms were tested for their utility, whereas Neural Networks turned out to be most suitable. The optimisation procedure of the employed Neural Network was discussed in detail and its prediction capability investigated. The unweighting performance could be improved for all examined processes, namely processes belonging to the $Z+4$ jets, $W+4$ jets as well as the $t\bar{t}+3$ jets groups. Using the Machine Learning approach, CPU time savings between 12% and a factor of 23.6 could be achieved. Multiple Monte Carlo validation observables were inspected, comparing the physics description of SHERPA with and without the Machine Learning approach. Both unweighting variants are in perfect agreement with each other, validating the correctness of the Machine Learning based unweighting prescription.

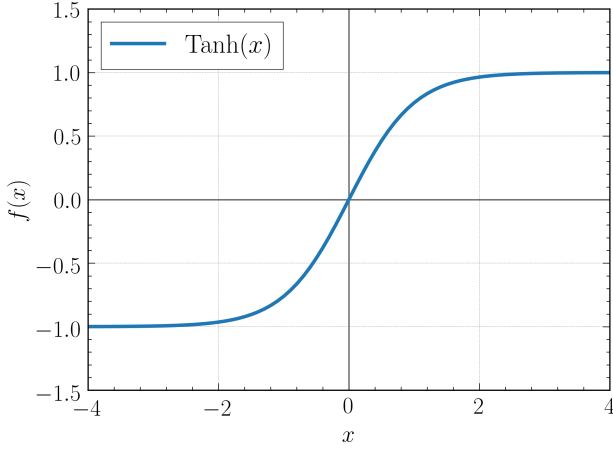
Within the scope of this thesis, all matrix elements were generated and performance improvements assessed using AMEGIC in the context of unweighted event generation. The COMIX matrix element generator enables the production of even higher multiplicity final-states and could thus be subject of further studies. Currently, a lot of effort is put into improvements of the Monte Carlo sampling utilising Neural Networks [65, 66, 67], aiming to increase the unweighting efficiencies. The interplay of these approaches with the Machine Learning based unweighting could be an interesting aspect of future work. Furthermore, only processes at

leading order were tested so far. The Machine Learning based unweighting could be extended to higher perturbative orders in the future.

A second subject of this thesis was the study and reduction of the negative weight fraction ε for next-to-leading order accurate samples. In particular, Z +jets and $t\bar{t}$ +jets were investigated. A large number of negative weights spoils the statistical convergence of event samples and is hence undesirable. In order to reduce the amount of negative weighted events, a leading colour approximation in the matching terms of the S-MC@NLO method was established and validated, effectively decreasing the negative weight fraction in the soft domain. Together with a Sudakov suppression on the first emission of $\mathbb{H}$ -events as well as a local K-factor modification, ε could roughly be halved for both investigated processes, from 18.1% to 9.1% for Z production in association with jets and from 24.8% down to 12.6% in the case of the $t\bar{t}$ +jets setup. The examined approaches significantly reduce the negative weight fraction while still being in good agreement with the SHERPA default and data.

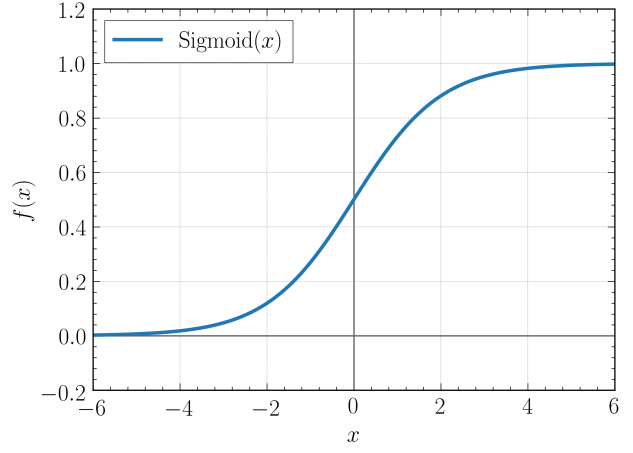
A Appendix

A.1 Activation Functions



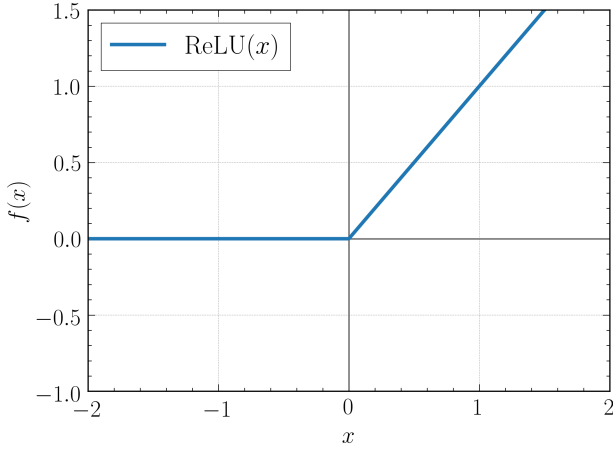
(a) Hyperbolic tangent

$$f(x) = \text{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



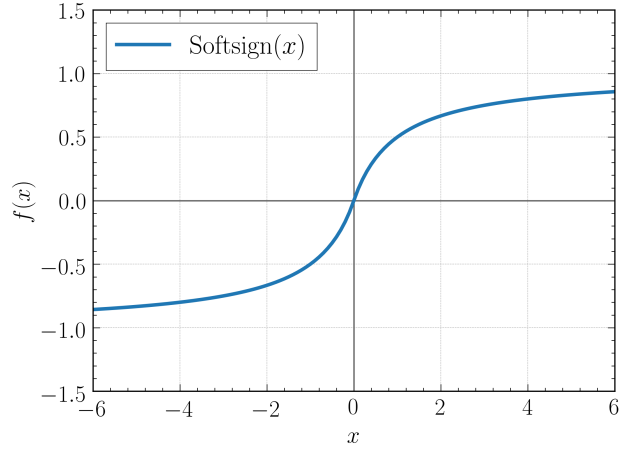
(b) Sigmoid activation function

$$f(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x + 1}$$



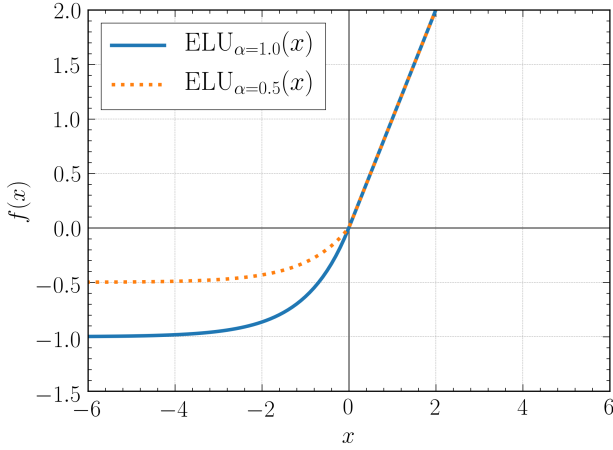
(c) Rectified Linear Unit (ReLU)

$$f(x) = \max(0, x)$$



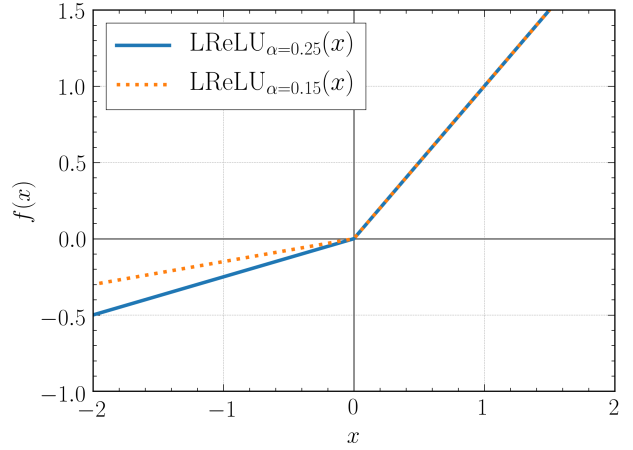
(d) Softsign activation function

$$f(x) = \frac{x}{1+|x|}$$



(e) Exponential Linear Unit (ELU)

$$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & x < 0 \\ x & x \geq 0 \end{cases}$$



(f) Leaky Rectified Linear Unit

$$f(\alpha, x) = \begin{cases} \alpha x & x < 0 \\ x & x \geq 0 \end{cases}$$

Figure A.1: Selection of activation functions commonly employed in neural networks and tested within this thesis.

List of Figures

2.1	Weight distributions for Z plus jets at leading order	16
2.2	Pictorial representation of one Monte Carlo event	18
3.1	Feynman diagrams quark-antiquark scattering at LO and NLO	21
3.2	Pictorial illustration of double-counting occurring during application of parton shower to NLO calculation	26
3.3	Pictorial representation of determination of a clustering history	28
4.1	Comparison between classic and Machine Learning based unweighting	31
4.2	Schematic representation of a single artificial neuron	35
4.3	Illustration of a simple fully connected feedforward neural network	36
4.4	Comparison of performance of different input features	40
4.5	Comparison of performance of different feature scalings	41
4.6	Comparison of performance of different hidden layer and nodes per layer configurations in terms of accuracy and speed	42
4.7	Comparison of performance of different activation functions	43
4.8	Comparison of Adam and RMSProp optimisers with differing learning rates	44
4.9	Comparison of differing numbers of phase-space points neural network training	46
4.10	Training and validation loss of optimised neural network	47
4.11	Distribution of weights $x = \frac{w}{a}$ for $gg \rightarrow e^-e^+ggd\bar{d}$	47
4.12	Evolution of second maximum x_{max} for $gg \rightarrow e^-e^+ggd\bar{d}$	48
4.13	Relative event weight distribution Machine Learning for $gg \rightarrow e^-e^+ggd\bar{d}$	50
4.14	Monte Carlo validation observables for $gg \rightarrow e^-e^+ggd\bar{d}$	52
4.15	Monte Carlo validation observables for $gg \rightarrow e^-e^+ggd\bar{d}$	53
4.16	Monte Carlo validation observables for $gg \rightarrow e^-\bar{\nu}_e gg u \bar{d}$	55
4.17	Monte Carlo validation observables for $gg \rightarrow e^-\bar{\nu}_e gg u \bar{d}$	56
4.18	Monte Carlo validation observables for $uu \rightarrow t\bar{t}guu$	58
4.19	Monte Carlo validation observables for $uu \rightarrow t\bar{t}guu$	59
5.1	Dependence of factor f on negative weight fraction ε	62
5.2	Monte Carlo validation observables comparing different mechanisms to reduce negative weight fraction for $pp \rightarrow Z+0,1,2 \text{ jets@NLO}+3 \text{ jets@LO}$	66

5.3	Monte Carlo validation observables comparing different mechanisms to reduce negative weight fraction for $pp \rightarrow Z+0,1,2 \text{ jets@NLO}+3 \text{ jets@LO}$	67
5.4	Comparison experimental $Z+\text{jets}$ ATLAS data to different mechanisms to reduce negative weight fraction	68
5.5	Monte Carlo validation observables comparing different mechanisms to reduce negative weight fraction for $pp \rightarrow t\bar{t}+0,1 \text{ jets@NLO}+2,3 \text{ jets@LO}$	70
5.6	Monte Carlo validation observables comparing different mechanisms to reduce negative weight fraction for $pp \rightarrow t\bar{t}+0,1 \text{ jets@NLO}+2,3 \text{ jets@LO}$	71
5.7	Monte Carlo validation observables comparing different mechanisms to reduce negative weight fraction for $pp \rightarrow t\bar{t}+0,1 \text{ jets@NLO}+2,3 \text{ jets@LO}$	72
5.8	Comparison experimental $t\bar{t}+\text{jets}$ ATLAS data to different mechanisms to reduce negative weight fraction	73
A.1	Selection of tested activation functions	78

List of Tables

2.1	Comparison of convergence of different numerical integration techniques	10
4.1	Investigated hyperparamters for optimisation of neural network	38
4.2	Average times needed to generate one (partially) unweighted event for $gg \rightarrow e^-e^+ggd\bar{d}$ with and without thr ML based unweighting	49
4.3	Results Machine Learning based unweighting Z +jets	51
4.4	Results Machine Learning based unweighting W +jets	54
4.5	Results Machine Learning based unweighting $t\bar{t}$ +jets	57
5.1	Percentage of positive and negative weighted $\mathbb{S}$ and $\mathbb{H}$ -events	63
5.2	Evolution negative weight fraction $pp \rightarrow Z+0,1,2$ jets@NLO+3 jets@LO	65
5.3	Evolution negative weight fraction $pp \rightarrow t\bar{t}+0,1$ jets@NLO+2,3 jets@LO	69

Bibliography

- [1] G. Aad et al. (The ATLAS collaboration). *Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC*. Phys.Lett., **B716**:1–29, 2012. [[arXiv:1207.7214 \[hep-ex\]](#)].
- [2] S. Chatrchyan et al. *Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC*. Phys.Lett., **B716**:30–61, 2012. [[arXiv:1207.7235 \[hep-ex\]](#)].
- [3] Q. R. Ahmad et al. *Direct evidence for neutrino flavor transformation from neutral current interactions in the Sudbury Neutrino Observatory*. Phys.Rev.Lett., **89**:011301, 2002. [[nucl-ex/0204008](#)].
- [4] Y. Fukuda et al. *Evidence for oscillation of atmospheric neutrinos*. Phys.Rev.Lett., **81**:1562–1567, 1998. [[hep-ex/9807003](#)].
- [5] L. Evans and P. Bryant. LHC machine. *Journal of Instrumentation*, 3(08):S08001–S08001, aug 2008. [[JINST 3 S08001](#)].
- [6] S. et al Ask. *An Introduction to PYTHIA 8.2*. Comput.Phys.Commun., **191**:159–177, 2015. [[arXiv:1410.3012 \[hep-ph\]](#)].
- [7] M. Bahr et al. *Herwig++ Physics and Manual*. Eur.Phys.J., **C58**:639–707, 2008. [[arXiv:0803.0883 \[hep-ph\]](#)].
- [8] J. Bellm et al. *Herwig 7.2 Release Note*. 2019. [[arXiv:1912.06509 \[hep-ph\]](#)].
- [9] T. Gleisberg et al. *Event generation with SHERPA 1.1*. JHEP, **02**:007, 2009. [[arXiv:0811.4622 \[hep-ph\]](#)].
- [10] E. Bothmann et al. *Event Generation with Sherpa 2.2*. SciPostPhys., **7**(3):034, 2019. [[arXiv:1905.09127 \[hep-ph\]](#)].
- [11] F. James. *Monte Carlo Theory and Practice*. Rept.Prog.Phys., **43**:1145, 1980.
- [12] J. Campbell, J. Huston, and F. Krauss. *The Black Book of Quantum Chromodynamics*. Oxford University Press, 1 edition, 2018.

- [13] R. Kleiss and R. Pittau. *Weight optimization in multichannel Monte Carlo*. Comput.Phys.Comm., **83**:141–146, 1994. [[hep-ph/9405257](#)].
- [14] S. Weinzierl. *Introduction to Monte Carlo methods*. 2000. [[hep-ph/0006269](#)].
- [15] F. A. Berends, R. Pittau, and R. Kleiss. *All electroweak four fermion processes in electron - positron collisions*. Nucl.Phys., **B424**:308–342, 1994. [[hep-ph/9404313](#)].
- [16] G. P. Lepage. *Vegas: An Adaptive Multidimensional Integration Program*. 1980. [[CLNS-80/447](#)].
- [17] T. Sjostrand, S. Mrenna, and P. Z. Skands. *PYTHIA 6.4 Physics and Manual*. JHEP, **05**:026, 2006. [[hep-ph/0603175](#)].
- [18] S. Höche, S. Schumann, and F. Siegert. *Hard photon production and matrix-element parton-shower merging*. Phys.Rev., **D81**:034026, 2010. [[arXiv:0912.3501 \[hep-ph\]](#)].
- [19] Geoffrey T. Bodwin. *Factorization of the Drell-Yan Cross-Section in Perturbation Theory*. Phys.Rev., **D31**:2616, 1985. [Erratum: Phys. Rev.D34,3932(1986)].
- [20] J. C. Collins, D. E. Soper, and G. F. Sterman. *Factorization for Short Distance Hadron - Hadron Scattering*. Nucl.Phys., **B261**:104–142, 1985.
- [21] J. C. Collins, D. E. Soper, and G. F. Sterman. *Soft Gluons and Factorization*. Nucl.Phys., **B308**:833–856, 1988.
- [22] B. Mistlberger. *Higgs boson production at hadron colliders at N^3LO in QCD*. JHEP, **05**:028, 2018. [[arXiv:1802.00833 \[hep-ph\]](#)].
- [23] Long-Bin Chen, Hai Tao Li, Hua-Sheng Shao, and Jian Wang. *Higgs boson pair production via gluon fusion at N^3LO in QCD*. Phys.Lett., **B803**:135292, 2020. [[arXiv:1909.06808 \[hep-ph\]](#)].
- [24] C. Duhr, F. Dulat, and B. Mistlberger. *The Drell-Yan cross section to third order in the strong coupling constant*. 2020. [[arXiv:2001.07717 \[hep-ph\]](#)].
- [25] M. Beneke, A. Maier, Th. Rauh, and P. Ruiz-Femenia. *Non-resonant and electroweak NNLO correction to the e^+e^- top anti-top threshold*. JHEP, **02**:125, 2018. [[arXiv:1711.10429 \[hep-ph\]](#)].
- [26] S. Höche. *Introduction to parton-shower event generators*. In *Proceedings, Theoretical Advanced Study Institute in Elementary Particle Physics: Journeys Through the Precision Frontier: Amplitudes for Colliders (TASI 2014): Boulder, Colorado, June 2-27, 2014*, pages 235–295, 2015. [[arXiv:1411.4085 \[hep-ph\]](#)].

-
- [27] R. Keith Ellis, W. James Stirling, and B. R. Webber. *QCD and collider physics*. *Camb. Monogr. Part. Phys. Nucl. Phys. Cosmol.*, 8:1–435, 1996.
- [28] S. Frixione and B. R. Webber. *Matching NLO QCD computations and parton shower simulations*. *JHEP*, **06**:029, 2002. [[hep-ph/0204244](#)].
- [29] P. Nason. *A New method for combining NLO QCD with shower Monte Carlo algorithms*. *JHEP*, **11**:040, 2004. [[hep-ph/0409146](#)].
- [30] S. Höche, F. Krauss, M. Schonherr, and F. Siegert. *A critical appraisal of NLO+PS matching methods*. *JHEP*, **09**:049, 2012. [[arXiv:1111.1220 \[hep-ph\]](#)].
- [31] T. Kinoshita. *Mass singularities of Feynman amplitudes*. *J.Math.Phys.*, **3**:650–677, 1962.
- [32] T. D. Lee and M. Nauenberg. *Degenerate Systems and Mass Singularities*. *Phys.Rev.*, **133**:B1549–B1562, 1964. [,25(1964)].
- [33] S. Höche and M. Schonherr. *Uncertainties in next-to-leading order plus parton shower matched simulations of inclusive jet and dijet production*. *Phys.Rev.*, **D86**:094042, 2012. [[arXiv:1208.2815 \[hep-ph\]](#)].
- [34] S. Catani, F. Krauss, R. Kuhn, and B. R. Webber. *QCD matrix elements + parton showers*. *JHEP*, **11**:063, 2001. [[hep-ph/0109231](#)].
- [35] S. Höche, F. Krauss, S. Schumann, and Frank Siegert. *QCD matrix elements and truncated showers*. *JHEP*, **05**:053, 2009. [[arXiv:0903.1219 \[hep-ex\]](#)].
- [36] Th. Gehrmann, S. Höche, F. Krauss, M. Schonherr, and F. Siegert. *NLO QCD matrix elements + parton showers in $e^+e^- \rightarrow \text{hadrons}$* . *JHEP*, **01**:144, 2013. [[arXiv:1207.5031 \[hep-ph\]](#)].
- [37] S. Höche, F. Krauss, M. Schonherr, and F. Siegert. *QCD matrix elements + parton showers: The NLO case*. *JHEP*, **04**:027, 2013. [[arXiv:1207.5030 \[hep-ph\]](#)].
- [38] S. Höche, F. Krauss, M. Schonherr, and F. Siegert. *NLO matrix elements and truncated showers*. *JHEP*, **08**:123, 2011. [[arXiv:1009.1127 \[hep-ph\]](#)].
- [39] A. Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O’Reilly Media, 2 edition, 2019.
- [40] J. Krause. *Efficiency Improvements Using Machine Learning in Event Generators for the LHC*, 2015. [[masterthesis-2015-10-johannes-krause.pdf](#)].
- [41] S. Ruder. *An overview of gradient descent optimization algorithms*. *CoRR*, **abs/1609.04747**, 2016.

- [42] F. Chollet. keras. <https://github.com/fchollet/keras>, 2015.
- [43] M. Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*, 2016. [arXiv:1603.04467 [cs.DC]].
- [44] T. Hermann et al. *frugally-deep*. 2016.
- [45] F. Rosenblatt. *The perceptron: A probabilistic model for information storage and organization in the brain*. *Psychological Review*, **65**(6):386–408, 1958.
- [46] F. Chollet. *Deep Learning with Python*. Manning, November 2017.
- [47] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning Representations by Back-propagating Errors*. *Nature*, **323**(6088):533–536, 1986.
- [48] RH Hahnloser, R Sarpeshkar, MA Mahowald, RJ Douglas, and HS Seung. *Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit*. *Nature*, **405**(6789):947–951, June 2000.
- [49] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014.
- [50] Y. LeCun, L. Bottou, G. Orr, and K. Müller. *Efficient BackProp*. In *Neural Networks: Tricks of the Trade*, Lecture Notes in Computer Science, chapter 2, page 546. Springer Berlin / Heidelberg, 1998.
- [51] H. Mhaskar, Qianli Liao, and T. A. Poggio. *Learning Real and Boolean Functions: When Is Deep Better Than Shallow*. CoRR, **abs/1603.00988**, 2016. [arXiv:1603.00988 [cs.LG]].
- [52] D. Rolnick and M. Tegmark. *The power of deeper networks for expressing natural functions*. CoRR, **abs/1705.05502**, 2017. [arXiv:1705.05502 [cs.LG]].
- [53] J. Stier and M. Granitzer. *Structural Analysis of Sparse Neural Networks*. *ProcediaComputerScience*, **159**:107–116, 2019.
- [54] X. Glorot and Y. Bengio. *Understanding the difficulty of training deep feedforward neural networks*. In Yee Whye Teh and D. Mike Titterton, editors, *AISTATS*, volume **9** of *JMLR Proceedings*, pages 249–256. JMLR.org, 2010.
- [55] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. CoRR, **abs/1502.01852**, 2015.
- [56] G. Hinton. *Lecture 6e—RMSProp: Divide the gradient by a running average of its recent magnitude*. COURSE: Neural Networks for Machine Learning, 2012. [RMSProp: Divide the gradient by a running average of its recent magnitude].

-
- [57] 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**:1929–1958, 2014.
 - [58] F. Krauss, R. Kuhn, and G. Soff. *AMEGIC++ 1.0: A Matrix element generator in C++*. *JHEP*, **02**:044, 2002. [[hep-ph/0109036](#)].
 - [59] T. Gleisberg and S. Höche. *Comix, a new matrix element generator*. *JHEP*, **12**:039, 2008. [[arXiv:0808.3674](#) [[hep-ph](#)]].
 - [60] R. D. et al Ball. *Parton distributions for the LHC Run II*. *JHEP*, **04**:040, 2015. [[arXiv:1410.8849](#) [[hep-ph](#)]].
 - [61] A. Denner and M. Pellen. *NLO electroweak corrections to off-shell top-antitop production with leptonic decays at the LHC*. *JHEP*, **08**:155, 2016. [[arXiv:1607.05571](#) [[hep-ph](#)]].
 - [62] F. Cascioli, P. Maierhofer, and S. Pozzorini. *Scattering Amplitudes with Open Loops*. *Phys.Rev.Lett.*, **108**:111601, 2012. [[arXiv:1111.5206](#) [[hep-ph](#)]].
 - [63] M. Aaboud et al. *Measurements of the production cross section of a Z boson in association with jets in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector*. *Eur.Phys.J.*, **C77**(6):361, 2017. [[arXiv:1702.05725](#) [[hep-ex](#)]].
 - [64] M. Aaboud et al. *Measurements of differential cross sections of top quark pair production in association with jets in pp collisions at $\sqrt{s} = 13$ TeV using the ATLAS detector*. *JHEP*, **10**:159, 2018. [[arXiv:1802.06572](#) [[hep-ex](#)]].
 - [65] Ch. Gao, J. Isaacson, and C. Krause. *i-flow: High-Dimensional Integration and Sampling with Normalizing Flows*. 2020. [[arXiv:2001.05486](#) [[physics.comp-ph](#)]].
 - [66] Ch. Gao, S. Höche, J. Isaacson, C. Krause, and H. Schulz. *Event Generation with Normalizing Flows*. 2020. [[arXiv:2001.10028](#) [[hep-ph](#)]].
 - [67] E. Bothmann, T. Janßen, M. Knobbe, T. Schmale, and S. Schumann. *Exploring phase space with Neural Importance Sampling*. 2020. [[arXiv:2001.05478](#) [[hep-ph](#)]].

Acknowledgements

This thesis would not have been possible without the help and support of many people.

First of all, I want to thank Frank Siegert for not only supervising my bachelor, but now also my master thesis as well as giving me the opportunity to work on interesting projects in between. I appreciate your efforts and the possibilities you enabled for me like the attendance of the MCnet summer school and the annual Sherpa meeting.

A huge thank you to my parents and my brother Robert, for their support throughout my studies and always having my back. On this occasion, I want to thank Alexander, Ingola, Martin and Natalie Bismark for being such great people and making Freiburg im Breisgau my second home.

I want to thank all the other members of the “Emmys” group, namely Johannes Krause, Sebastian Liebschner, Herberth Torres and Christian Wiel. At this, special thanks go to Johannes for being the good soul of the office and our numerous discussions about nice bike and hiking tours.

Very special thanks is due to Alexander, for his endless love and support, the innumerable beautiful moments aside from physics and for proofreading this thesis.

Finally, I would like to thank COVID-19 for making the writing process of this thesis as unpleasant as possible.

Erklärung

Hiermit erkläre ich, Katharina Danziger, dass ich diese Arbeit im Rahmen der Betreuung am Institut für Kern- und Teilchenphysik ohne unzulässige Hilfe Dritter verfasst und alle Quellen als solche gekennzeichnet habe.

Katharina Danziger
Dresden, März 2020