Department of Physics and Astronomy

University of Heidelberg

Master thesis

in Physics

submitted by

Sebastian Guido Bieringer

born in Dieburg

2021

Using Invertible Networks to Measure QCD Splittings from Parton Showers

This Master thesis has been carried out by Sebastian Guido Bieringer

at the

Institute of Theoretical Physics

under the supervision of

Prof. Tilman Plehn

Abstract

Hadronic parton showers are a fundamental property of QCD and are omnipresent at collider experiments. For collinear and soft showers, the probability for the splitting of a mother parton into two partons can be described in terms of splitting kernels. In this work, we show, how we can use invertible neural networks, a special realization of normalizing flows, to examine these splitting kernels from low-level jet information.
We introduce a general parametrization of the splitting kernels, including parameters for collinearly suppressed rest terms, and present how a probability distribution for the parameters can be obtained using machine learning inference. We examine the effects of hadronization and detector simulations and show that the uncertainty of the inferred distribution scales with the inverse square root of the number of measurements for two orders of magnitude.

Zusammenfassung

Eine Auffächerung von Hadronen durch sukzessive Spaltungen is ein fundametale Ergebnis von Quanten Chromodynamik. Diese "shower" werden an allen Teilchenbeschleunigern beobachtet und können für kleine Abstrahlungswinkel and geringe Energieüberträge durch wenige Wahrscheinlichkeitsverteilungen der Spaltungen beschrieben werden. In dieser Arbeit zeigen wir, wie invertierbare Neuronale Netze zur Analyse dieser Wahrscheinlichkeitsverteilungen genutzt werden können.
Wir führen eine Parametrisierung für die Verteilungen ein, die auch einen Term beinhaltet für Effekte, die bei kleinen Abstrahlungswinkeln unterdrückt sind. Für diese Parameter zeigen wir, wie ihre bedingte Wahrscheinlichkeit durch neuronale Netze gewonnen werden kann. Wir untersuchen den Einfluss von Hadronisierung und Detektorsimulationen auf die Messung und zeigen, dass die Unsicherheit der Messung für zwei Größenordnungen invers zur Wurzel der Messungsgröße skaliert.

Contents

1 Introduction

With the continuous improvement of computational resources, modern machine learning methods and especially deep learning methods have gained a lot of momentum in the last decade. The techniques have had much success in commercial applications such as self-driving cars, image processing software, search algorithms, language translation tools and more. With respect to particle physics, with its big and well understood data sets, these computational developments have also found a strong foothold in modern physics research. A multitude of successful routines have been developed for example for classification tasks, such as jet classification [1], or generative approaches, for example to accelerate Monte Carlo event generators [2].

One central flaw of these techniques is the "black box"-nature of neural networks. They bring technological advance, that is better performance, however they usually do not help with the understanding of the fundamental processes. Quite the contrary is true, machine learning techniques often do not rely on a broad understanding of the underlying physics and it is hard, seemingly impossible, to interpret them this way. In this work we want to utilize an architecture, that uses neural networks to improve the understanding of the studied process.

In contrast, normalizing flows are a rather new architecture of networks. They allow an invertible mapping between a simple latent distribution and a complicated target distribution. Therefore, they have been examined for both generative and evaluation tasks. In [3] it was proposed how to use this class of networks for inference tasks. Utilizing the efficiently invertible architecture from [4], this method has been tested for different, small problems, mainly in sociology. The question arises: How can we employ it in the context of contemporary particle physics? What quantities can be inferred?

Hadronic parton showers in collider experiments, can be described in terms of only three fundamental splitting kernels, derived from only the quark-gluon and triple gluon interaction for collinear, soft radiation. They do not require an understanding of electroweak corrections or parton densities. As such they are a natural candidate for a LHC object to examine using normalizing flows.

The aim of this work is to introduce a new technique to analyze parton showers from low-level observables, that is particle 4-momenta. To this end, we propose a new parametrization for the splitting kernels including a small rest term not included in leading-order quantum chromodynamics (QCD). We vary the parameters beyond their Standard Model value and generate parton shower data using both a simplified, as well as an established parton shower generator. Starting from a leptonic electron-positron scattering, we try to estimate the possibility of using a similar analysis in LHC context.

In chapter 2 we give a short introduction to the physics basics of our analysis. We then introduce the description of parton showers in QCD, as well as the setups we use to generate them in chapter 3. An introduction to normalizing flows starting from neural network basics is given in chapter 4 and in chapter 5 we explain how we can leverage this architecture as an inference technique. Chapter 6 and 7 present the results for our toy setup and for events generated with SHERPA [5] respectively. We conclude in chapter 8

The results of this work, including most of the figures, have been published in [6]. The project was a joint effort of Theo Heimel and me and results can be found in both theses.

2 Important concepts from particle physics

Finding the most fundamental building blocks of matter, is a central goal of particle physics. High-energy collisions at particle colliders, such as the LHC, are used to generate so far unobserved particles. As yet, the constituents found are well described by the Standard Model of particle physics [7, 8, 9] and formulated in the framework of Quantum Field Theory (QFT). There are a lot of textbooks on QFT [10, 11] as well as the Standard Model [12], so we will only introduce a few key concepts here. The Standard Model describes the elementary particles and the strong and weak nuclear force, as well as the electromagnetic force. A unification with gravity has not yet been successful. In the following we will introduce the full particle content of the Standard Model.

2.1 Particle content of the Standard Model

The interactions between the elementary particles are described by the exchange of gauge bosons, elementary particles of integer spin. The gauge bosons mediating the elementary forces are spin-1 particles. Specifically, photons mediate the electromagnetic force, W^+, W^- and Z boson the weak interaction and gluons the strong interaction. The last boson in the Standard Model is the spin-0 Higgs boson. It is found to be essential in giving particles their masses through the Higgs mechanism and electroweak symmetry breaking.

The fermionic particles carry a spin of $\frac{1}{2}$ and can be grouped into two categories. The first category are the quarks. They carry a color charge and therefore participate in the strong interaction. The quarks themselves can also be put into two groups, three generations of down-like quarks, called down, strange and bottom, with an electric charge of $-\frac{1}{3}$ and up-like quarks, called up, charm and top, with an electric charge of $\frac{2}{3}$. Hadrons then emerge as combinations of quarks, held together by the strong interaction, that is by the exchange of gluons. A proton for example contains two up- and one down-quark. As a matter of fact, quarks are confined, meaning they only appear as hadrons.

The second category of fermions in the Standard Model are the leptons. They do not have a color charge and can therefore only interact electromagnetically, if they have an electric charge, or via the weak interaction. There are three generations of leptons, each containing an electron-like particle with a full negative charge and a electrically neutral neutrino. The three generations are electrons, muons and tauons.

Every particle in this picture has an antiparticle or, in case of the photons, gluons

and Z boson, are their own antiparticles. This is only possible for neutral particles, as antiparticles carry opposite charges but the same mass as the particle. One example for an well known antiparticle relevant for this thesis is the positron, the positively charged antiparticle of the electron.

The dynamics of a fermion ψ are determined by the Dirac-equation. It can be understood as the Schrödinger-equation for a particle obeying a linear energy-momentum relation. Usually, it is expressed using the four γ-martices

$$(i\gamma^\mu \partial_\mu - m)\,\psi = 0\,. \tag{2.1}$$

The solutions of Dirac-equation are the Dirac-spinors. Their properties are defined by the algebra of the γ-matrices. As a result of this, the solutions of the Dirac-equations describe spin-$\frac{1}{2}$ particles. We will write a fermion with positive spin as u_+ and with negative spin u_-.

To form Lorentz-invariant combinations of such spinors, we need to define the adjoint spinor

$$\overline{\psi} = \psi^\dagger \gamma^0\,. \tag{2.2}$$

To derive the Lagrangian of the Standard Model, lets start at the Lagrangian of fermions and introduce the bosonic content, except the Higgs, by constructing the Lagrangian to be invariant under certain symmetry transformations. The Standard Model Lagrangian can be defined as the minimal Lagrangian invariant under transformations of the symmetry group $SU(3) \times SU(2)_L \times U(1)_Y$. To demonstrate the generation of the bosons from symmetries, let us start at a Lagrangian of fermions. Using the adjoint spinor, we can rewrite (2.1) as the Lorentz-invariant Lagrangian

$$\mathcal{L} = \overline{\psi}(i\gamma^\mu \partial_\mu - m_\psi)\psi. \tag{2.3}$$

We see that this Lagrangian is not symmetric under a local symmetry of the form

$$\psi(x) \to M\psi(x) = e^{i\theta_a(x)t^a}\psi(x), \tag{2.4}$$

where M is part of the symmetry group and t^a are the generators of the group. For $M \in SU(N)$, there are $N^2 - 1$ such generators, fulfilling the condition of a Lie algebra

$$[t^a, t^b] = i f^{abc} t^c. \tag{2.5}$$

In case of $SU(3)$ and $SU(2)$, the structure constants f^{abc} are given by a totally antisymmetric Levi-Civita symbol. As $U(1)$ only has one generator, its structure constant is zero.

Because the derivative in the Lagrangian (2.3) acts on the factors $\theta(x)$ of the local symmetry transformation, it is not invariant under such transformations. To write an invariant Lagrangian, we have to introduce a gauge field A. The transformation properties, which render the Lagrangian invariant, are found to be

$$A_\mu \to M A_\mu M^\dagger + \frac{i}{g}(\partial_\mu M) M^\dagger, \tag{2.6}$$

in order to restore the invariance of the Lagrangian, using the covariant derivative

$$D_\mu = \partial_\mu + igA_\mu = \partial_\mu + igA_{\mu a} t^a \tag{2.7}$$

instead of the regular derivative. It transforms as

$$D_\mu \psi \to M D_\mu \psi. \tag{2.8}$$

The factor g determines the coupling strength between fermions and the bosons corresponding to the symmetry. For the strong interaction, we will write the coupling strength g_s.

To determine the dynamics of the new gauge field, we also have to include a kinetic term for them into our Lagrangian. We can write a Lorentz and gauge invariant kinetic term using the field strength tensor $F_{\mu\nu} = -\frac{i}{g}[D_\mu, D_\nu]$ as

$$\mathcal{L} = -\frac{1}{2} \operatorname{Tr} F_{\mu\nu} F^{\mu\nu}. \tag{2.9}$$

Using the covariant derivative (2.7), the whole fermionic content of the Standard Model can be written in one Lagrangian as a sum of Dirac-Lagrangians similar to (2.3) and kinetic terms similar to (2.9). We have to take into account the chirality of the particles. This is important since only left-handed fermions interact weakly [13, 14] and thus only left-handed fermions can be doublets under $SU(2)$. As sketched above, the interaction between fermions and gauge bosons is expressed by the covariant derivative. To make the derivative term invariant under transformations of the whole symmetry group $SU(3) \times SU(2)_L \times U(1)_Y$, the covariant derivative has to include gauge fields for all three symmetries.

To include masses for the per se massless $SU(2)_L$ gauge fields, we also have to include coupling terms to the Higgs field, that is a kinetic term for the Higgs. If we assume this field to have a non-vanishing vacuum expectation value, the coupling to it has an effect similar to a mass term of the Dirac-Lagrangian. The mass eigenstates of the $SU(2)_L$ and $U(1)_Y$ gauge fields can then be written in terms of the interaction eigenstates. These linear combinations, are the three massive bosons W^+, W^- and Z and the massless photon. This formalism is commonly referred to as electroweak symmetry breaking. The eight gauge bosons of the $SU(3)$ symmetry, the gluons, remain massless.

Couplings between fermions and the Higgs field also lead to mass terms and mixing between the quark families via exchange of a Higgs. The masses, as well as the mixing amplitudes, are determined by Yukawa-couplings.

Bringing all these different terms together, yields the full Lagrangian of the Standard Model. By restricting ourselves to the various gauge sectors, we distinguish processes involving only the $U(1)$ photon as quantum electrodynamics (QED) and processes involving only the $SU(3)$ gluons as quantum chromodynamics (QCD).

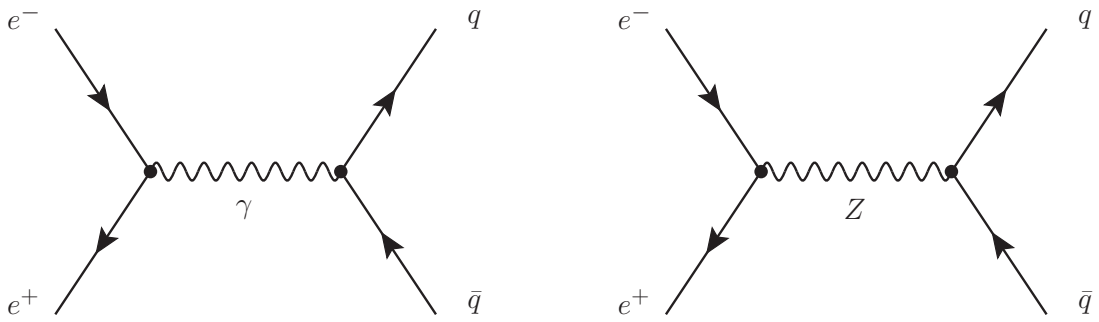

Figure 2.1: Leading order Feynman diagrams for $e^+e^- \to q\bar{q}$

2.2 Electron-positron annihilation

In this work we are interested in measuring contributions to hadronic parton showers. These arise as a sequence of quark-gluon and triple gluon interactions of final-state quarks or gluons. A thorough discussion of parton showers will be done in Chapter 3.

As the parton showering happens with rather small momentum transfers, we will distinguish between these soft splittings and the underlying hard process. To better examine the QCD processes, we want to choose a rather simple hard process. Generating hadronic showers from quarks generated in electron-positron scattering

$$e^+e^- \to q\bar{q} \tag{2.10}$$

at the Z-boson mass $m_Z = 91.18$ GeV circumvents having to distinguish between hadronic initial-state radiation and the showers from final-state quarks. Here $q\bar{q}$ denotes a quark/antiquark pair. For simplicity, we will only consider the production of an up-, down- or or strange-quark. and assume the quarks to be massless. With respect to the large Z-boson mass this is a good approximation.

In experiments, for example at the LEP, collisions of leptons also have the advantage that the energy of the process is known. In hadron colliders, such as the LHC, the momentum of the initial-state particles is always distributed following parton distribution functions and only the momentum of the full hadron is known.

Figure 2.1 shows both leading-order Feynman diagrams for this process. Using the Feynman rules for QED and for the weak interaction, we calculate the total matrix element as the sum of both matrix elements

$$|\mathcal{M}|^2 = |\mathcal{M}_\gamma + \mathcal{M}_Z|^2 = |\mathcal{M}_\gamma|^2 + |\mathcal{M}_Z|^2 + 2\,\mathrm{Re}\,\mathcal{M}_Z\mathcal{M}_\gamma \tag{2.11}$$

and determine the differential cross section of the process. The value of the cross section can then be used to unweight calculated events, that is to scale the frequency of the events with their cross section value.

Due to the mass of the propagating Z-boson, the matrix element squared for the weak interacting process has the shape of a Breit-Wigner distribution

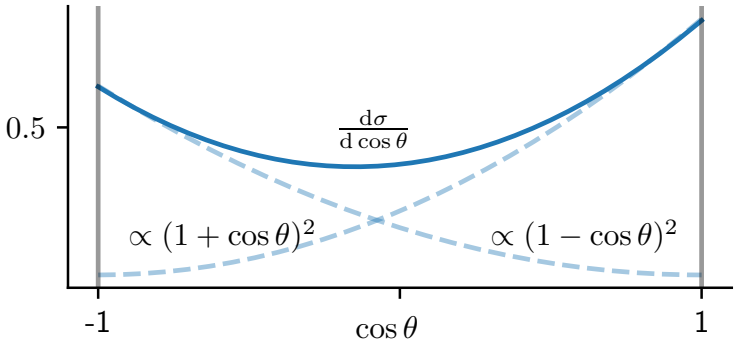

Figure 2.2: Normalized differential cross section for $e^+e^- \to q\bar{q}$ at $\sqrt{s} = 91.18$ GeV. The data has been generated using the matrix element generator from [16], which has been used in our toy model from Section 6.

$$|\mathcal{M}_Z|^2 \propto \frac{1}{\left(s - m_Z^2\right)^2 + m_Z^2 \Gamma_Z^2}, \tag{2.12}$$

with s the invariant mass square of the sum of the initial-state particles and Γ_Z the total decay width of the Z-boson. It peaks at m_Z, where it dominates the total matrix element [15]. As we examine collisions at $\sqrt{s} = m_Z$ we can neglect the QED and mixed contributions.

There are two degrees of freedom to the final states

$$p_q = \frac{\sqrt{s}}{2} \begin{pmatrix} 1 \\ \sqrt{(1-\cos\theta)^2}\cos\phi \\ \sqrt{(1-\cos\theta)^2}\sin\phi \\ \cos\theta \end{pmatrix} \quad \text{and} \quad p_{\bar{q}} = \begin{pmatrix} \sqrt{s}/2 \\ -\vec{p}_q \end{pmatrix} \tag{2.13}$$

of this process, the azimuthal angle $\phi \in \{0, 2\pi\}$ and the polar scattering angle relative to the initial beam direction $\theta \in \{0, \pi\}$. The cross section is independent of the azimuthal angle ϕ. Thus, we can pick a random value for this angle for every event we generate.

The matrix element however depends on the chirality of the initial- and final-state particles. For vanishing mass the helicity of a particle corresponds to its chirality and therefore the matrix element depends on the polar scattering angle θ. Due to the chirality of the weak interaction, the resulting matrix elements depend on the chirality of the outgoing relative to the ingoing particles. The spin-averaged matrix element squared therefore has contributions proportional to

$$\propto (1+\cos\theta)^2 \quad \text{and} \quad \propto (1-\cos\theta)^2 .$$

This can be seen in Figure 2.2. All these properties of the differential cross section, including the exact shape of the θ-dependence, are rather unimportant in the toy

example in Chapter 6, as we work with weighted data in the beginning. This means our events are created with a uniform distribution over $\cos\theta$. Since the neural network we want to use does not have to learn the angular distribution as an artefact of the hard process, this should simplify the learning process.

However, in Chapter 7 we generate data using SHERPA [5] for a more realistic setting. As such, we work with unweighted data like it would appear in experiment.

3 The data set: QCD parton showers

Quarks and gluons from hard scattering are not observed directly in collider experiments. For one, they cause hadronic showers by radiating quarks and gluons via quark-gluon and triple gluon vertices. These can then split again as long as the energy is sufficient. Including a splitting to the process yields a process next-to-leading order (NLO) in the strong coupling constant. As we will encounter in the discussion of splitting kernels in Section 3.1, processes of this nature exhibit infrared (IR) divergences. In Section 3.2, we discuss the machinery to calculate such parton showers and implement it in a first toy setup.

There are also, NLO terms with loop diagrams, exhibiting ultraviolet (UV) divergences that can be dealt with by renormalization [10]. UV divergent terms are well understood and will not be part of this thesis.

Another important effect, due to which the final states of the hard process are not observed directly, is the color confinement of QCD. At low energies, quarks only appear in bound states,which are singlets under the color $SU(3)$ group transformation, meaning that quarks are only observed indirectly through hadrons. The process forming hadrons from quarks at the end of our simulation chain is called hadronization. More details on additional mechanisms after parton showering and a summary of our more realistic simulation setup can be found in Section 3.3.

In Section 3.4, observables suitable for examining parton showers are introduced. For an introduction to QCD events at colliders, we refer to [17], whilst an even more in depth discussion is presented in [18].

3.1 From splitting kernels to the parameter space

As just introduced, IR divergent terms arise (amongst others) from radiation of soft quarks or gluons from initial or final state particles, that is radiation with a small momentum transfer. As the initial-state particles of our hard process are electrons and positrons (see Section 2.2), they can only radiate photons. Radiation that can be traced back to one final-state parton, is referred to as a jet. Please note that jets are commonly defined by the reconstruction of the radiation history from the the measured energy depositions, as we will see in Section 3.3.3, and not by the construction from the final-state partons of the hard scattering. The discussion of splitting kernels and jets in this chapter is largely adapted from [17].

As the splittings can happen multiple times, a description in a factorized phase space will be very useful. Let us introduce some notation, to describe how the $(n+1)$-particle phase space can be factorized into the n-particle phase space and a splitting event.

For a splitting of a single, arbitrary mother parton, denoted with an index a, into two daughter partons, b and c, we define the energy fraction of the final states as

$$z = \frac{|E_b|}{|E_a|} = 1 - \frac{|E_c|}{|E_a|}. \tag{3.1}$$

The opening angle between the momenta of the outgoing particles θ, can be related to the energy fraction as

$$p_a^2 = z(1-z)E_a^2\theta^2 + \mathcal{O}(\theta^4) \quad \overset{\theta \to 0}{\Longrightarrow} \quad \theta \simeq \frac{1}{|E_a|}\sqrt{\frac{p_a^2}{z(1-z)}}. \tag{3.2}$$

The limit $\theta \to 0$, that is the limit in which the sister parton is radiated in beam direction, is called the collinear limit. The whole description of parton showers through splitting kernels is done is this limit. For an initial state with finite positive invariant mass much higher than that of the final states, it allows us to write the kinematics of the splitting in Sudakov-decomposition

$$p_b = (-zp_a + \beta n + p_T) \tag{3.3}$$
$$p_c = (-(1-z)p_a - \beta n - p_T)\,. \tag{3.4}$$

Here n is an arbitrary unit vector, p_T is the momentum orthogonal to the momentum of the original state p_a and the unit vector n and β is a free factor. Using the Sudakov-decomposition, the $(n+1)$-particle phase space can be divided into the n-particle phase space and the splitting process. Neglecting terms of linear or higher order in the opening angle θ in the collinear limit, we can derive the separated collinear phase space

$$\mathrm{d}\Phi_{n+1} = \mathrm{d}\Phi_n \frac{\mathrm{d}p_{c,3}\,\mathrm{d}p_T^2\,\mathrm{d}\phi}{4(2\pi)^3\,|E_c|}\frac{1}{z} = \mathrm{d}\Phi_n \frac{\mathrm{d}z\,\mathrm{d}p_a^2\,\mathrm{d}\phi}{4(2\pi)^3}(1+\mathcal{O}(\theta)) \quad \overset{\theta \to 0}{=} \quad \mathrm{d}\Phi_n \frac{\mathrm{d}z\,\mathrm{d}p_a^2}{4(2\pi)^2}. \tag{3.5}$$

In the last step, we also assumed azimuthal symmetry and get a factor of 2π. A step-by-step calculation of this factorization with all neglected terms can be found in the literature [17].

The (unregularized) splitting kernel $\hat{P}(z)$ now arises from the fact that we can factorize the matrix element of the $(n+1)$-particle process to get the matrix element of the n-particle process and from the assumption that this part can be expressed depending on z only

$$\boxed{\overline{|\mathcal{M}_{n+1}|^2} \simeq \frac{2g_s^2}{p_a^2}\hat{P}(z)\overline{|\mathcal{M}_n|^2}\,.} \tag{3.6}$$

With this assumption the total cross section can be calculated as

$$\begin{aligned} \mathrm{d}\sigma_{n+1} &= \overline{|\mathcal{M}_{n+1}|^2}\mathrm{d}\Phi_{n+1} \\ &= \overline{|\mathcal{M}_{n+1}|^2}\mathrm{d}\Phi_n \frac{\mathrm{d}p_a^2\,\mathrm{d}z}{4(2\pi)^2} \\ &\simeq \frac{2g_s^2}{p_a^2}\hat{P}(z)\overline{|\mathcal{M}_n|^2}\mathrm{d}\Phi_n \frac{\mathrm{d}p_a^2\,\mathrm{d}z}{16\pi^2} \end{aligned} \tag{3.7}$$

$$\Longrightarrow \boxed{\sigma_{n+1} \simeq \int \sigma_n \frac{\mathrm{d}p_a^2}{p_a^2}\mathrm{d}z\frac{\alpha_s}{2\pi}\hat{P}(z)} \quad \text{using} \quad g_s^2 = 4\pi\alpha_s\,. \tag{3.8}$$

It is helpful to write the integration in terms of a more accessible variable. In (3.5) we have implicitly used that

$$p_a^2 = \frac{p_T^2}{z(1-z)}\,. \tag{3.9}$$

For initial state radiation, p_b often is the momentum of a particle entering a consecutive splitting event. In this context, it can interpret it as the Mandelstam-variable of a t-channel diagram. From the Sudakov-decomposition of the emitted momentum (3.3), we can express the Mandelstamm-variable as

$$t \equiv p_b^2 = \frac{p_T^2}{1-z}. \tag{3.10}$$

According to our problem, we can therefore choose our integration variable in (3.8) from

$$\frac{\mathrm{d}p_a^2}{p_a^2} = \frac{\mathrm{d}p_T^2}{p_T^2} = \frac{\mathrm{d}t}{t}\,. \tag{3.11}$$

3.1.1 Altarelli-Parisi splitting kernels

The definition of the splitting kernel in (3.6) relies on the assumption that the matrix element factorizes at least in the collinear limit. One can prove this assumption by direct calculation of the total matrix element in this limit. In Appendix A.1 we have done this calculation at leading order for the splitting $q(p_a) \to q(p_b) + g(p_c)$.

The splitting of a quark into a quark and a gluon is fundamental for our hard process, as we only discuss jets produced from quarks. For the first discussion in Chapter 6, we even limit our shower to this splitting exclusively.

For a gluon radiating of a massless quark, we get in the collinear limit (see Appendix A.1)

$$\overline{|\mathcal{M}_{n+1}|^2} = \frac{2g_s^2}{p_a^2}\frac{N_c^2-1}{2N_c}\frac{1+z^2}{1-z}\overline{|\mathcal{M}_n|^2}\,. \tag{3.12}$$

With (3.6) we identify the splitting kernel of this splitting as

$$\boxed{\hat{P}_{qq}(z) = C_F \frac{1+z^2}{1-z} \text{ with } C_F = (N_c^2 - 1)/2N_c = \frac{4}{3}\,.} \tag{3.13}$$

As we will see later, it is useful to rewrite this by expanding the numerator

$$\hat{P}_{qq} = C_F \frac{1+z^2}{1-z} = C_F \left(\frac{2z}{1-z} + \frac{z^2 - 2z + 1}{1-z} \right) \tag{3.14}$$

$$= C_F \left(\frac{2z}{1-z} + \frac{(1-z)^2}{1-z} \right) = C_F \left(\frac{2z}{1-z} + (1-z) \right) \tag{3.15}$$

The same quark-gluon vertex, that describes the splitting 3.16, describes the the splitting of an incoming gluon into on outgoing quark and antiquark. As in the derivation of $\hat{P}_{qq}$, the contribution of the unphysical scalar and longitudinal gluon polarizations, γ^0 and γ^3 respectively, cancel. For this splitting, only terms from diagonal spin combinations contribute. In the case of equal spin, the terms vanish due to the fact that gluons only couple to fermions with a spin-difference of 1. This leaves us with the unregularized splitting kernel

$$\boxed{\hat{P}_{gq}(z) = T_R \left[z^2 + (1-z)^2 \right] \text{ with } T_R = 1/2\,.} \tag{3.16}$$

Naturally, it is symmetric under the exchange of the outgoing quark and antiquark $z \Leftrightarrow (1-z)$.

A similar calculation can be done for the triple gluon vertex with special attention to the different possible polarizations of the gluons. The unregularized splitting kernel for the $g \to gg$ splitting event is

$$\boxed{\hat{P}_{gg}(z) = C_A \left[\frac{z}{1-z} + \frac{1-z}{z} + z(1+z) \right] \text{ with } C_A = N_c = 3\,.} \tag{3.17}$$

Again, it is symmetric under the exchange of the produced gluons and diverges in the case of a soft gluon.

These splitting kernels are often called *Altarelli-Parisi splitting kernels* [19].

Regularization

If we want to calculate the total cross section using (3.8), we have to integrate the splitting kernels over z. This is important for us, as we have to evaluate the probability of a splitting happening during the simulation of a parton shower. Because of the divergence of $\hat{P}_{qq}$ and $\hat{P}_{gg}$ for soft outgoing particles, $z \to (0,)1$, we need to regularize this integral. In practise this is often done by a simple cutoff, although the +-Regularization scheme using dimensional regularisation is analytically more appealing. It is shortly discussed in Appendix A.2.

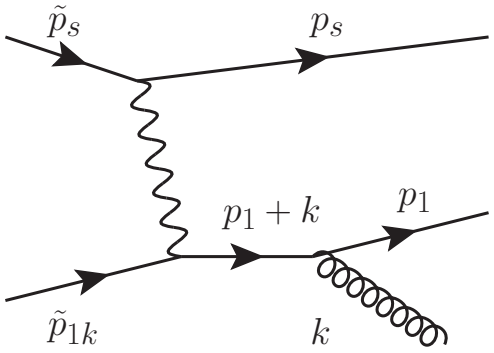

Figure 3.1: Catani-Seymour dipole splitting vertex for $q \to q + g$ splitting

As our parton shower simulations use a cutoff as opposed to dimensional regularisation, we do not need to circumvent the divergences our splitting kernels analytically. We will thus drop the hat in the following $P_{qq}(z) = \hat{P}_{qq}(z)$. Section 3.2.2 explains in more detail how the divergences are avoided by defining a transverse momentum cutoff. The IR divergences of the splitting kernel are simply not in the interval in which z is simulated.

3.1.2 Catani-Seymour dipoles

One problem of the above description is that it is missing the necessary degrees of freedom to include on-shell conditions for the partons in the splitting. This is no problem in theory, however, if one wants to compute a parton shower splitting after splitting, it is. The problem can be solved by considering a spectator quark [20]. A diagrammatic representation of the spectator is given in Figure 3.1. Here, $\tilde{p}_{1k} = p_1 + k$ is the momentum of the so-called emitter and $\tilde{p}_s$ the momentum of the spectator. Momentum conservation for this diagram reads

$$\tilde{p}^\mu_{1k} + \tilde{p}^\mu_s = p^\mu_1 + k^\mu + p^\mu_s \,. \tag{3.18}$$

And we simultaneously impose the on-shell conditions

$$\tilde{p}^2_{1k} = \tilde{p}^2_s = p^2_1 = k^2 = p^2_s = 0 \tag{3.19}$$

for massless partons, for the emitter particle, as well as for all others. Defining the transferred momentum fraction y to the spectator, we can make sure the on-shell condition $\tilde{p}^2_s = 0$ follows from the one for the outgoing spectator

$$p_s = (1 - y)\tilde{p}_s \,. \tag{3.20}$$

We can calculate the momentum fraction y from $\tilde{p}_{1k}^2 = 0$ using momentum conservation

$$0 \overset{!}{=} \tilde{p}_{1k}^2 = \left(p_1 + k + p_s - \frac{p_s}{1-y}\right)^2 \Longleftrightarrow y = \frac{(p_1 k)}{(p_1 k) + (p_1 p_s) + (p_s k)}\,. \tag{3.21}$$

The momentum fraction transmitted to p_1 from the emitter now is defined for the projection onto $\tilde{p}_s$, which gives the collinear direction. From momentum conservation we calculate

$$(p_1 \tilde{p}_s) = \tilde{z}_1\,(\tilde{p}_{1k} \tilde{p}_s) \Longleftrightarrow\ \tilde{z}_1 = \frac{(p_1 p_s)}{(p_1 p_s) + (p_s k)}\,. \tag{3.22}$$

The momentum transfer to the gluon is simply given as

$$\tilde{z}_k = 1 - \tilde{z}_1. \tag{3.23}$$

Using (3.21) and (3.22) we find the leading pole

$$\boxed{\frac{1}{1 - \bar{z}_1(1-y)} = \frac{(p_1 k) + (p_1 p_s) + (p_s k)}{(p_1 k) + (p_s k)}\,.} \tag{3.24}$$

We want to make sure that this pole shows the right divergence for soft gluon radiation, $z \to 0, 1$. In the collinear limit we also want to recover the results from the naive, collinear derivation in Appendix A.1. Again, we consider the case of a soft gluon being emitted of a hard quark, thus $\tilde{z}_1 \to 1$.

Soft radiation limit: This far, in Appendix A.1 we have only discussed the splitting in the collinear limit to derive the pole structure for soft radiation. For the case of a soft gluon being emitted of a quark leg, that is neglecting the gluon momentum, we can calculate the factorization of the matrix element without the collinear limit as

$$\mathcal{M}_{n+1} = g_s \epsilon_\mu^*(k) \left(\sum_j \hat{T}_j^a \frac{p_j^\mu}{(p_j k)}\right) \bar{u}(p) \mathcal{M}_n\,, \tag{3.25}$$

with k and p_j as before. The sum includes the possibility of the gluon splitting of one of the two quarks of our hard process. For massless partons, we then calculate the matrix element squared

$$\overline{|\mathcal{M}_{n+1}|^2} = -2 g_s^2 \sum_{i<j} \hat{T}_i^a \hat{T}_j^a \frac{(p_i p_j)}{(p_i k) + (p_j k)} \left(\frac{1}{(p_i k)} + \frac{1}{(p_j k)}\right) \overline{|\mathcal{M}_n|^2}\,. \tag{3.26}$$

If we use this to calculate the total cross section, we get a logarithmic divergence in the soft gluon limit, as well as in the collinear limit. The IR divergences however cancel due to the Kinoshita–Lee–Nauenberg theorem [21, 22].

In the soft limit, the momentum transfer to the emitter vanishes, $y \to 0$. The Feynman diagram 3.1 thus reduces to a triple vertex without a spectator and the leading pole of the Catani-Seymour leading pole (3.24) becomes

$$\frac{1}{1-\bar{z}_1(1-y)} = \frac{1}{\lambda}\frac{p_1 p_s + \mathcal{O}(\lambda)}{(p_1 p)+(p_s p)} \tag{3.27}$$

As k vanishes, we need to define a new reference momentum through $k =: \lambda p$, with $\lambda \to 0$ a small parameter characterizing the soft limit. This pole shows the same structure as (3.26). We can thus use this pole to write the soft splitting kernel.

Collinear radiation limit: For the collinear limit of the splitting we can write the outgoing daughter partons in a Sudakov-decomposition

$$p_1 = zp + p_T - \frac{p_T^2}{z}\frac{n}{2(pn)} \quad \text{and} \quad k = (1-z)p - p_T - \frac{p_T^2}{1-z}\frac{n}{2(pn)} \tag{3.28}$$

In the collinear limit where $p_T \to 0$ the Diagram 3.1 again reduces to a triple vertex, not connected to the spectator, and

$$y \to 0 \quad \text{and} \quad \tilde{z_1} \to z + \mathcal{O}(p_T^2)\,. \tag{3.29}$$

Therefore, in the collinear limit, the divergence (3.24) is exactly the same as the divergence of the splitting kernel in the naive derivation

$$\frac{1}{1-\tilde{z}_1(1-y)} = \frac{1}{1-z}\left(1+\mathcal{O}\left(p_T^2\right)\right)\,. \tag{3.30}$$

The Sudakov-decomposition (3.28) is also used for the explicit kinematic description of the particles in our parton shower. In our shower generator, as in many parton shower generators, n is identified with the spectator momentum.

We conclude that the Catani–Seymour description with a spectator parton keeps all particles on-shell and the divergences exhibit the expected behaviour in the soft and collinear limit.

In Section 6 we will see that the divergent terms dominate the behaviour of a shower. This description is therefore very effective.

3.1.3 Parametrization of the splitting kernels

The first step towards the splitting kernels we actually use in this work is to rewrite (3.13), (3.16) and (3.17) in terms of Canani-Seymour dipoles. This simply corresponds to the transformation $z \to z(1-y)$ in the divergent terms, as motivated in the previous discussion. Here we simplify writing $\tilde{z}_1$ as z for clarity, as it corresponds to z in the collinear and in the soft limit. The Cantani-Seymour splitting kernels are

$$\hat{P}_{qq} = C_F \left(\frac{2z(1-y)}{1-z(1-y)} + (1-z) \right) \tag{3.31}$$

$$\hat{P}_{gg} = 2C_A \left(\frac{z(1-y)}{1-z(1-y)} + \frac{(1-z)(1-y)}{1-(1-z)(1-y)} + z(1-z) \right) \tag{3.32}$$

$$\hat{P}_{gq} = T_R \left(z^2 + (1-z)^2 \right) , \tag{3.33}$$

where we used the rewritten version of $\hat{P}_{qq}$. This way, the divergent terms of $\hat{P}_{qq}$ and $\hat{P}_{gg}$ have the same shape and the finite terms of both only differ by a factor of z. Furthermore, all parts of the splitting kernels are positive and can be interpreted as probabilities themselves.

We now want to parameterize the divergent, finite and constant contributions to these splitting kernels. Actually, we want to parameterize the contributions to the cross section (3.8)

$$\sigma_{n+1} \simeq \int \sigma_n \frac{\mathrm{d}p_T^2}{p_T^2} \mathrm{d}z \frac{\alpha_s}{2\pi} \hat{P}(z) \, .$$

A constant term in the cross section thus needs to be suppressed with $p_T^2 \propto yz(1-z)$. Scaling the constant term with p_T^2 also ensures that it vanishes in the collinear limit, where our calculation of the splitting kernels is exact.

As the finite terms of the $\hat{P}_{qq}$ and $\hat{P}_{gq}$ kernel originate from the same vertex, the parameter factorizing both should also coincide in a consistent theory. Including all of these considerations into our parameterization, we get

$$P_{qq}(z,y) = C_F \left[D_{qq} \frac{2z(1-y)}{1-z(1-y)} + F_{qq}(1-z) + C_{qq}\, yz(1-z) \right] \tag{3.34}$$

$$\begin{aligned} P_{gg}(z,y) = 2C_A \Bigg[& D_{gg} \left(\frac{z(1-y)}{1-z(1-y)} + \frac{(1-z)(1-y)}{1-(1-z)(1-y)} \right) \\ & + F_{gg}\left(z(1-z)\right) + C_{gg}\, yz(1-z) \Bigg] \end{aligned} \tag{3.35}$$

$$P_{gq}(z,y) = T_R \left[F_{qq} \left(z^2 + (1-z)^2 \right) + C_{gq}\, yz(1-z) \right] . \tag{3.36}$$

In the following, we will ignore the hats on the splitting kernels, as we do not need to regularize them for the parton shower code including a transverse momentum cutoff.

From our previous considerations it is clear, that to leading order in QCD

$$D_{qq,gg} = 1 \qquad F_{qq,gg} = 1 \qquad C_{qq,gg,gq} = 0\,. \tag{3.37}$$

We will refer to this set of values as the Standard Model (SM) point of the parameter space examined in this work. In more evolved parton showers, D is often modified to include effects of higher-order contributions on the normalization [23]. For our toy model in Chapter 6, we ignore these contributions. In SHERPA they are included in the running of the strong coupling.

As we vary the parameters, we can encounter negative splitting kernel values. To make sure the splitting probabilities stay positive, we set the splitting kernels to zero for negative kernel values.

Figure 3.5 shows the effect of varying the parameters in terms of the high-level observables defined in Section 3.4for the example of $D_{qq} \in \{0.5, 2\}$. Generally, higher splitting kernel values lead to higher splitting probabilities and thus to higher particle numbers in the shower. This result is further discussed at the end of this chapter, once parton showers have been fully introduced.

The parameterization introduced in this section was developed in cooperation with Stefan Höche [6].

3.1.4 Experimental measurements on the splitting kernels

Because the divergent terms by far dominate the the other contributions (see Section 6.1), our measurements of D_{qq} and D_{gg} can be compared to experimental measurements of C_F and C_A, for example at LEP. The color factors also appear in our splitting kernels (3.13)-(3.16) and for QCD have the values $C_F = 4/3$, $C_A = 3$ and $C_A/C_F = 2.25$.

- In 3-jet events from electron-proton annihilation via a Z-Boson, OPAL measured the multiplicity of quark and gluon jets. From their energy dependence, they determine $C_A/C_F = 2.23 \pm 0.14$ [25] (solid line in Figure 3.2).
- DELPHI measured parton densities for quarks and gluons for similar 3-jet events. From the ratio of both, they measure $C_A/C_F = 2.26 \pm 0.16$ [26] (dashed line).
- ALEPH examined 4-jet events from the same process. From of a fit to the rate and angular correlations of the events, they get $C_A = 2.93 \pm 0.60$ and $C_F = 1.35 \pm 0.27$ [27] (red ellipse).
- In a similar analysis, OPAL measured $C_A = 3.02 \pm 0.56$ and $C_F = 1.34 \pm 0.30$ [28] (green ellipse).
- The authors of [29] used fits to event shape observables and get $C_A = 2.84 \pm 0.24$ and $C_F = 1.29 \pm 0.18$ (blue ellipse).

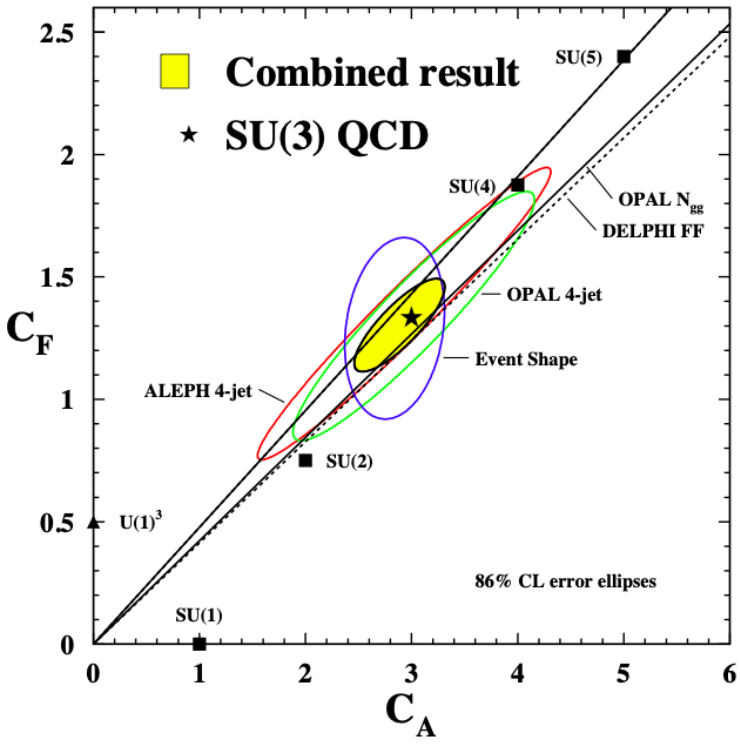

Figure 3.2: Measurements on the color factors C_F and C_A as discussed in section 3.1.4. The 4-jet measurements and the combined measurement of both factors are highly correlated. This figure was taken from [24].

The combination of the results [24, 30] (yellow shades in Figure 3.2) gives

$$C_A = 2.89 \pm 0.21 \quad \text{and} \quad C_F = 1.30 \pm 0.09\,. \tag{3.38}$$

For all presented analysis, both measurements are highly correlated, as can be see in Figure 3.2. The uncertainty of the results are dominated by systematic errors. and the relative error on the measurements are $\sim 7.3\%$ and $\sim 6.9\%$ respectively.

3.2 Generating parton showers

Up to this point, we have discussed the collinear factorization of phase space and the step-by-step derivation of our splitting kernels without paying attention to the regularization techniques to deal with the IR divergences emerging from multiple QCD splittings. Many of those techniques, such as resummation of parton splittings in the initial state and the associated scales or the running of the parton densities via the DGLAP-equation [19, 31, 32], are mostly important for hadronic initial states and are thus not relevant to this work. Furthermore, in this work we are not interested in the weights of the events, but in the events themselves. As such,

we are not investigating the inclusive cross section of our hard process, including QCD splittings of final-state partons. Instead, we start at the hard process and add partons trough splittings according to the probability of the splitting.

A more detailed review of the technicalities of parton shower generators can be found in [33] or in the literature belonging to the different event generators, for example [34].

3.2.1 Sudakov-factors

In order to calculate the probability of a splitting according to the factorization of the cross section (3.8), we need to integrate the splitting kernels in both, z and the evolution variable t.

The distribution of the splittings is given as

$$\mathrm{d}\mathcal{P}_{ij}(t) = \frac{\mathrm{d}t}{t}\int_{z_{\min}}^{z_{\max}} \mathrm{d}z \frac{\alpha_s}{2\pi} P_{ij}(z, y(z,t)) =: \mathrm{d}t f_{ij}(t)\,. \tag{3.39}$$

This has to be understood as the differential probability for one splitting with a specific $i, j \in \{q, g\}$ at one point in the evolution variable t.

It is the nature of this "radioactive decay"-like process, that a splitting can only happen, if it did not happen before. Technically, we are therefore interested in the probability for a single splitting to occur at t without any splittings prior to this point. The probability for no splitting to happen between a the scale t and an upper evolution scale t' can be calculated as

$$\Delta_i(t, t') := \mathcal{P}_{0,i}(t, t') = \exp\left(-\sum_j \int_t^{t'} \mathrm{d}\tilde{t} f_{ij}(\tilde{t})\right)\,, \tag{3.40}$$

with the sum going over all possible splittings. Δ_i is commonly referred to as a Sudakov-factor and an evolution equation can be derived for these factors. The exponential arises from the summation of all virtual corrections and unresolved emissions between the two scales [33]. It is related to the probability for a single branching to occur at t when starting at a scale t' by Poisson-statistics

$$\mathcal{P}_{1,i}(t, t') = \frac{\mathrm{d}}{\mathrm{d}t}\Delta_i(t, t') = \prod_j f_{ij}(t) \exp\left(-\int_t^{t'} \mathrm{d}\tilde{t} f_{ij}(\tilde{t})\right)\,. \tag{3.41}$$

To generate splittings in our parton shower generator, we now want to sample values of t according to this probability. For simplicity, assume only one integrated splitting kernel $f(t)$. If this splitting kernels has an invertible primitive function $F(t)$, it is easy to sample from $\mathcal{P}_1$.

3.2.2 Sudakov veto algorithm

To understand this, we review the basics of sampling from a distribution. A pseudo-random number generator typically generates uniform distributed numbers between 0 and 1. We can sample from a random distribution $p(x)$ in an interval $[x_{min}, x_{max}]$, by applying P^{-1} to the random numbers. Here,

$$P(x) = \int_{x_{min}}^{x} dx' p(x')$$

is the cumulative distribution function which for a probability distribution is continuously increasing and therefore invertible.

The first key difference to our problem is, that after integrating (3.41) we are left with only the exponential Sudakov-factor. We would therefore sample by applying the inverse primitive function $F^{-1}(t)$ to the logarithm of a random number. As motivated above, this is easy if the primitive function is known and invertible.

Our splitting kernels are not this nicely behaved. The Sudakov veto algorithm still allows us to sample from (3.41) by introducing a function $g(t) > f(t)$ that bounds the integrated splitting kernel from above [34]. This overestimate $g(t)$ must have an invertible primitive function $G(t)$. To sample according to the probability distribution (3.41), we now

1. start at the lower bound of the evolution scale integral $t_0 = t$ for $i = 0$,
2. sample a point t_i by applying $G^{-1}(G(t_{i-1}) - \log R)$ to the logarithm of a random number R,
3. reject this point with the probability $g(t_i)/f(t_i)$
4. and sample another point with the condition $t_{i+1} > t_i$ until a point is accepted [34].

One can see that this way the total probability to sample a value t is given by the sum over the probabilities to sample t with n intermediate steps

$$\begin{aligned}
\mathcal{P}_1^{(n)}(t, t') =& \frac{f(t)}{g(t)} g(t) \exp\left\{ - \int_t^{t_1} \mathrm{d}\tilde{t}\, g(\tilde{t}) \right\} \\
& \times \prod_{i=1}^{n} \left[\int_{t_{i-1}}^{t_{i+1}} \mathrm{d}t_i \left(1 - \frac{f(t_i)}{g(t_i)} \right) g(t_i) \exp\left\{ - \int_{t_i}^{t_{i+1}} \mathrm{d}\tilde{t}\, g(\tilde{t}) \right\} \right] \\
=& \mathcal{P}_1^{(0)}(t, t') \frac{1}{n!} \left(\int_t^{t'} \mathrm{d}\tilde{t} \left[g(\tilde{t}) - f(\tilde{t}) \right] \right)^n ,
\end{aligned} \tag{3.42}$$

where $t_{n+1} = t'$. The sum over all n thus gives an exponential and the term $g(t)$ in the new exponential cancels the one in the prefactor $\mathcal{P}_1^{(0)}$. The total probability

to sample a point t then corresponds to (3.41) [34]. It is easy to see that for this algorithm to be efficient, $f(t)/g(t)$ should be as close to 1 as possible.

In (3.8) we have seen that we can choose p_T as an evolution variable. This has the advantage that sub-leading order corrections which force decreasing scattering angles for subsequent emissions are implemented by design [35]. Our toy shower generator, as well as recent SHERPA and PYTHIA implementations use the transverse momentum as the evolution scale. While the t increases with the number of splittings, the maximum possible transverse momentum p_T decreases like the parton energy from splitting to splitting. This means, we want to start the veto algorithm at the maximum possible transverse momentum, given by the momentum of the final-state partons of the hard scattering. We then generate new points by $G^{-1}(G(t_{i-1}) + \log R)$ and enforce the condition that subsequently generated points are smaller than their predecessors.

We also have to introduce a lower bound on the evolution scale where confinement becomes relevant, that is a bound of order Λ_{QCD}. Partons differing less than 1 GeV in transverse momentum can not be be resolved. This introduces a cutoff scale $p_{T,0}$ for the IR. For lower values the veto algorithm terminates. As p_T is related to the energy fraction z and y as $p_T^2/Q^2 = yz(1-z)$, with Q the momentum sum of spectator and emitter, the cutoff directly introduces a $z_{\min}$ and $z_{\max}$. This means, we can use unregularized splitting kernels with the veto algorithm.

We can now build the parton shower generator by repeatedly using the algorithm to generate p_T values at which a splitting occurs. For every value, we also sample z from $g(p_T) = g(p_T, z)$ and reject the pair with the probability $g(p_T, z)/f(p_T, z)$. For an accepted point, this already fixes all degrees of freedom, except the azimuthal angle. However, in the derivation of (3.8) we assumed azimuthal symmetry, so we can sample the angle from a uniform distribution.

Multiple splitting kernels and splittings from different partons can be included in this algorithm, by using the second step of the algorithm to generate a value of p_T for every kernel and emitter and choose the one with the highest value. We then go on with the third step of the algorithm as before.

Multiple splittings are created by running the algorithm until it reaches the cutoff scale.This can be thought of as the factorization of the probability for multiple splittings into a chain of probabilities for single splittings.

To use the splitting kernels $P_{ij}(z, y(z, p_T))$ in the veto algorithm, we need to find an overestimate $\tilde{P}_{ij}(z)$, denoted with a tilde. We choose the overestimate to only depend on z, making the primitive function of the integral over the evolution variable a linear function in p_T^2. It only depends on the momentum of the emitter and spectator and on the cutoff via the integral boundaries. Our choice for the overestimates and their integrals for the splitting kernels (3.34) - (3.36) can be found in Appendix A.3.

3.2.3 Toy shower setup

The toy parton shower used to benchmark our inference technique is a direct implementation of the veto algorithm. Without any further modifications it reproduces the same results as SHERPA [5]. The toy setup allows us to generate only one jet by excluding the second quark from the shower. To minimize momentum transfer to the second quark, we only allow it to be the spectator of the first splitting. We then directly use the momenta generated by the parton shower generator without including hadronization or detector effects to train or network. They are part of the simulation chain for SHERPA data and are explained in more detail in Section 3.3.

We have to pay attention to the ordering of the momenta generated by the shower generator. Every time a splitting occurs, the code updates the momenta of the mother particle with one of the daughter particle and appends the other daughters momentum to the array of momenta. In a $q \to qg$ splitting, the daughter quark replaces the mother quark. In a $g \to gg$ splitting the symmetry in the splitting leads to a random choice. And, in a $g \to q\bar{q}$ splitting, the daughter quark again replaces the mother gluon. This way, we get an ordering, which includes information on the sequence of the splittings. We refer to this as *truth-sorting*. Using truth-sorted data for inference poses an information-backdoor, as this information is not available from experiment. Therefore, it is necessary to develop a sorting algorithm inspired by by the sequence of splittings that only uses the information of the momenta. This sorting, which we will refer to it as k_T*-sorting*, is introduced in Section 3.3.3.

For Standard Model (SM) parameters of the splitting kernels, Figure 3.3 shows the distribution in transverse momentum of a single jet for our toy parton shower. We see most jets are generated at the upper boundary given by the momentum of the quark after the hard scattering. Figure 3.3 also shows the distribution for showers generated for $e^+e^- \to q\bar{q}$ with SHERPA's CSSHOWER++ [36] shower generator. To get a single jet from SHERPA results we used an anti-k_T jet clustering algorithm with a cutoff at 20 GeV.

From Figure 3.5 we can see that the toy shower data is dominated by showers with a very low number of splittings $n_{PF} < 4$. Therefore, it poses no constraint that we limit the maximum number of constituents used for inference to $n = 13$. To achieve a constant input size for the inference network, we cut or pad the toy shower data to the correct size.

3.3 Augmenting the simulation chain

More evolved event generators, such as SHERPA [5], describe more physical scenarios than our simple toy shower generator. They have to apply different physic schemes for collinear and hard jets and account for color confinement of the jet products by hadronization. We also want to include a simulation of the restrictions in the experimental setup and an algorithm to determine the jets from the measured momenta.

As we are only examining single electron-positron collisions, effects, such as initial-

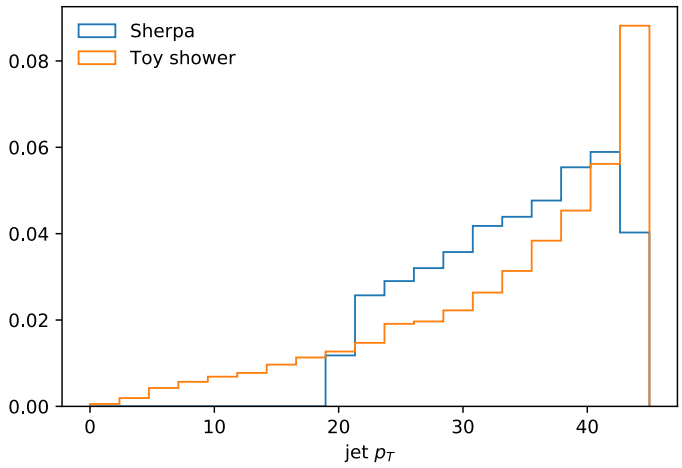

Figure 3.3: Spectrum of the total transverse momentum p_T of a single jet for $n = 10^4$ jets generated from our toy setup and from SHERPA. The single jet generated with SHERPA was defined using an anti-k_T clustering algorithm cutoff at 20 GeV.

state radiation, underlying events from particles not included in the hard process and pile-up from using bunches instead of single particles, are not relevant for this study.

It should also be mentioned that parton shower generators for initial-state radiation are fundamentally different from final-state ones. They often employ a backwards evolution scheme starting from the incoming particles of the hard interaction [34]. Luckily, we have chosen a hard process with leptonic initial states that do not exhibit initial-state parton showers.

Another fact worth mentioning is, that the parton shower as described above relies on a hard process, which is only described to leading order. There are several methods to include next-to-leading order terms in Monte Carlo methods. The two main algorithms are the POWHEG and the MC@NLO method [17].

3.3.1 Hadronization

The toy setup we use for benchmarking our inference process uses the momenta of the quarks after the parton shower generator terminates. However, at small energies, color confinement becomes relevant and we have to transition from single gluons and quarks to hadronic final states as they would be measured in a collider experiment. This transition is hard to describe, because at scales where confinement is relevant, QCD is a non-perturbative theory.

We will briefly discuss the two models currently used to describe hadronization in Monte Carlo event generators [33].

String model

The first model is the string model. It builds on the observation that the potential between a quark-antiquark pair rises linearly as

$$V(r) = \kappa r \quad \text{with} \quad \kappa \approx 1 \text{ GeV/fm}. \tag{3.43}$$

This corresponds to the picture where a flux tube with constant energy per length is stretched between the two quarks. The energy flux in the tube can be described as a massless string. When the original pair is separated further, the energy contained in the string leads to the creation of a new quark-antiquark pair.

Cluster model

The hadronization model used in SHERPA is called the cluster model [33]. It uses the preconfinement property of QCD: At every stage, the parton shower forms color-singlet combinations of partons (clusters). The invariant mass distribution of these clusters only depends on the cutoff scale and Λ_{QCD}, and is independent of the center-of-mass energy of the collision itself.

At the cutoff scale, final-state gluons are split into quarks non-perturbatively. And included into primary clusters. The mass distribution of the clusters is then closely connected to the hadron spectra [37]. This is known as local parton-hadron duality.

The clusters are then transformed into hadrons via two-body decays. Clusters of mass above $3 - 4$ GeV first have to split into two lighter clusters, while very low mass clusters can transition into hadrons directly.

The cluster model produces marginally more accurate results, whilst using fewer parameters than the string model.

3.3.2 Detector simulation

In our simulation chain we assume the hadrons to be stable enough to reach the detector without decaying. However, the measurement process itself introduces constraints on the measured data. They are given by the spatial and momentum resolution of the detector.

In our experimental setup we use DELPHES [38] to simulate the detector effects of the ATLAS detector. It includes a track propagation system embedded in a magnetic field, electromagnetic and hadron calorimeters, and a muon identification system [38]. Tracks and calorimeter deposits are then used to reconstruct the particles, as is done in experiment. A common algorithm for this reconstruction is the CMS particle flow event-reconstruction algorithm [39]. To profit from these insights, we use particle flow (PF) objects when including detector simulations in our setup.

3.3.3 Jet clustering

From experiment or in our case from simulation, we obtain a large set of particle momenta. Unlike at the start of this chapter, we do not know to splitting history. As such, we have to employ recombination algorithms to assign the particles to a shower. Although the term *jet* has been mentioned already, the proper definition of a jet is via a recombination algorithm. It groups the particle momenta according to the initial hard process by trying to recreate the splitting history.

The idea is to find collinear partners for identified subjets. Generally, this is done by constructing a distance measure y_{ij} for the collinearity of two particles or subjets i and j and combining two subjets if this distance meausrure falls below a threshold value. To construct the concrete measures, we need the azimuthal angle ϕ and the scattering angle θ in form of the pseudorapidity

$$\eta = -\log\tan\frac{\theta}{2}\,. \tag{3.44}$$

More concretely, we are interested in the distance of the two particles in the η-ϕ plane

$$\Delta R_{ij} = \sqrt{\Delta\phi_{ij}^2 + \Delta\eta_{ij}^2}\,. \tag{3.45}$$

where Δ denotes the distance in the respective dimension. Using this distance scaled with some power of the transverse momentum as the distance measure, corresponds to

- the k_T algorithm [40]

$$y_{ij} = \frac{\Delta R_{ij}}{R}\min(p_{T,i}, p_{T,j}) \quad \text{and} \quad y_{iB} = p_{T,i}\,,$$

- the Cambridge/Aachen algorithm [41]

$$y_{ij} = \frac{\Delta R_{ij}}{R} \quad \text{and} \quad y_{iB} = 1$$

- or the anti-k_T algorithm [42]

$$y_{ij} = \frac{\Delta R_{ij}}{R}\min(p_{T,i}^{-1}, p_{T,j}^{-1}) \quad \text{and} \quad y_{iB} = p_{T,i}^{-1}\,.$$

Here R is a value which has to be chosen appropriately for the "size" of a jet and to distinguish jets from beam radiation. A jet recombination algorithm then finds the combination of particles or subjets, i and j with the minimal distance, and combines the two if $\min(y_{ij}, y_{iB}) = y_{ij}$ or removes them if they are beam radiation, $\min(y_{ij}, y_{iB}) = y_{iB}$, until a distance cutoff or a minimal number of jets is reached. Recombination algorithms like these are infrared save, meaning they are independent on the IR cutoff of the showering.

The distance measure of the k_T algorithm scales with the transverse momentum. It therefore starts from the softest constituent, while the anti-k_T algorithm does the opposite and starts the sorting with the hard constituents. As the radiation of our parton shower is ordered in p_T, the k_T algorithm approximates the splitting history of the shower. The tree generated by the anti-k_T algorithm has no simple physical interpretation. It is however more compact and easier to interpret geometrically, making it easier to subtract background effects depending on the geometry of the jet. The Cambridge/Aachen algorithm can be interpreted geometrically as well. It produces jets circular in the η-ϕ plane.

In this work we need recombination algorithms to perform two different tasks:

Finding a single jet

In our simulation chain we need to identify a single jet from the $e^+e^- \to q\bar{q}$ simulation, as SHERPA we use sherpa to generate two showers, one each from the quarks exiting the hard process. For this we use the implementation of an anti-k_T algorithm in FASTJET [43] with $R = 1.2$ and a lower cutoff at $p_T = 20$ GeV. We then choose the harder one of the two jets, as it contains more splittings. Figure 3.3 shows the transverse jet momentum of jets recombined this way.

We note that FASTJET uses an anti-k_T algorithm where every entry in the distance measure above is squared.

k_T-sorting

As explained in Section 3.2.3, we also need a clustering algorithm to construct a sorting for the particle momenta, because the toy parton shower generator produces an array including information on the shower history. We have found that the results using this information backdoor are significantly better than training on shuffled data or data sorted by p_T. We therefore want to find an ordering that includes an estimated shower history.

To reconstruct the shower history, we use a k_T algorithm. We can choose $R = 1$ since we do not have to discriminate between different jets. Starting from the first splitting in the reconstructed shower history, we follow the hardest constituent in every splitting, that is the daughter particle with the highest energy fraction z, to produce the first entry of the array. We then go to the second constituent of this splitting and again follow the hardest constituent. We do so for every branching in the shower history.

This sorting, in the following referred to *k_T-sorting*, reproduces the truth-sorted data for a correctly identified shower history with the difference that the daughter partons of a splitting are ordered in their energy fraction and not depending on the splitting.

For low numbers of splittings, our quark jets are dominated by the radiation of a soft gluon of a hard quark. Therefore, we expect the k_T-sorting to produce similar results to the truth-sorting for low numbers of emissions. Even assuming the

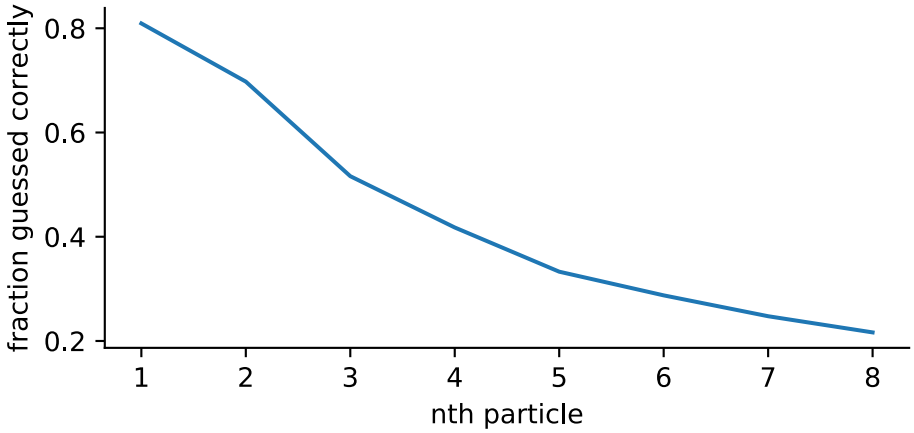

Figure 3.4: Ratio of momenta remaining on the same position in k_T-sorting and in truth-sorting depending on the position in the sorted array ($n = 10^5$).

k_T algorithm produced the right sorting history, for higher numbers of branchings in the shower, the symmetry in the gluon emission leads to significant differences between both sortings. This can be seen in Figure 3.4. For the first entries of the array, that is the left side of the graph, the ratio is dominated by the high number of parton showers with few splittings (see Figure 3.5). Only for these showers with low n_{PF} the sorting gives similar results to the truth-sorting.

For SHERPA generated data or data from experiment, there exists no truth-sorting. It still is sensible to include the reconstructed splitting history in the sorting. We have tried different ordering schemes and have found the k_T-sorting to give the best results.

The results presented in Section 6.1 indicate that further improvements to the sorting strategies will have an overall positive effect on the inference performance.

3.3.4 Sherpa setup

In the following we summarize the data generation setup for the realistic scenario including hadronization. We use a modified version SHERPA 2.2.10 [5] that includes our modified splitting kernels (3.34)-(3.36) to simulate the hard process

$$e^+e^- \to q\bar{q}\,. \tag{3.46}$$

We limit the quark flavours to up, down and strange and assume them to be massless. Like in the toy setup, we terminate the parton shower generation at 1 GeV. The resulting quarks are then hadronized.

To examine the effect of hadronization, we save the momenta of the hadrons, photons and charged leptons at this point and use an anti-k_T algorithm from FASTJET 3.3.4 [43] with $R = 1.2$ and a lower cutoff at 20 GeV to select a single jet.

In Section 7, we also want to examine the effect of the detector simulation on the inference. We apply DELPHES 3.4.2 [38] with the default ATLAS card to the simulation after hadronization. Again, we use an anti-k_T algorithm to determine a single jet and then save all particle flow (PF) objects [39].

In both cases, like in the toy setup, we use the k_T-sorting to sort the momenta. We then crop the data to $n = 13$ constituents to keep the input dimension constant. As can be seen in Figure 3.5, the parton showers in the more realistic setup can contain more particles than this. This should not be problematic, as only very soft constituents get cropped due to the k_T-sorting. We have found that limiting the number of constituents further can decrease the quality of inference.

3.4 Jet observables

To analyze parton showers with a variable number of constituents, as well as to benchmark the effect of training directly on the particle momenta, we use six standard jet observables [44] without regard to their infrared and collinear safety. We introduce

- the particle multiplicity [45]

$$n_{PF} = \sum_i 1\,, \tag{3.47}$$

- the girth of the distributed radiation [46]

$$w_{PF} = \frac{\sum_i p_{T,i} \Delta R_{i,jet}}{\sum_i p_{T,i}} \tag{3.48}$$

  with $R_{i,jet}$ being the angular separation between momentum p_i and the total jet momentum,

- a measure

$$p_T D = \frac{\sqrt{\sum_i p_{T,i}^2}}{\sum_i p_{T,i}} \tag{3.49}$$

  for the effect of the "softness" of the radiation [47]

- and the two point energy correlator [48]

$$C_{0.2} = \frac{\sum_{i,j} E_{T,i} E_{T,j} (\Delta R_{ij})^{0.2}}{\left(\sum_i E_{T,i}\right)^2} \tag{3.50}$$

  with 0.2 a tuned parameter to discriminate between quarks and gluons and $E_T = \sqrt{m^2 + p_T^2}$ the transversal energy.

The remaining two jet observables are the highest fraction of $p_{T,jet}$ carried by a single constituent $x_{\max}$ and the smallest number N_{95} of constituents carrying 95% of $p_{T,jet}$ [49].

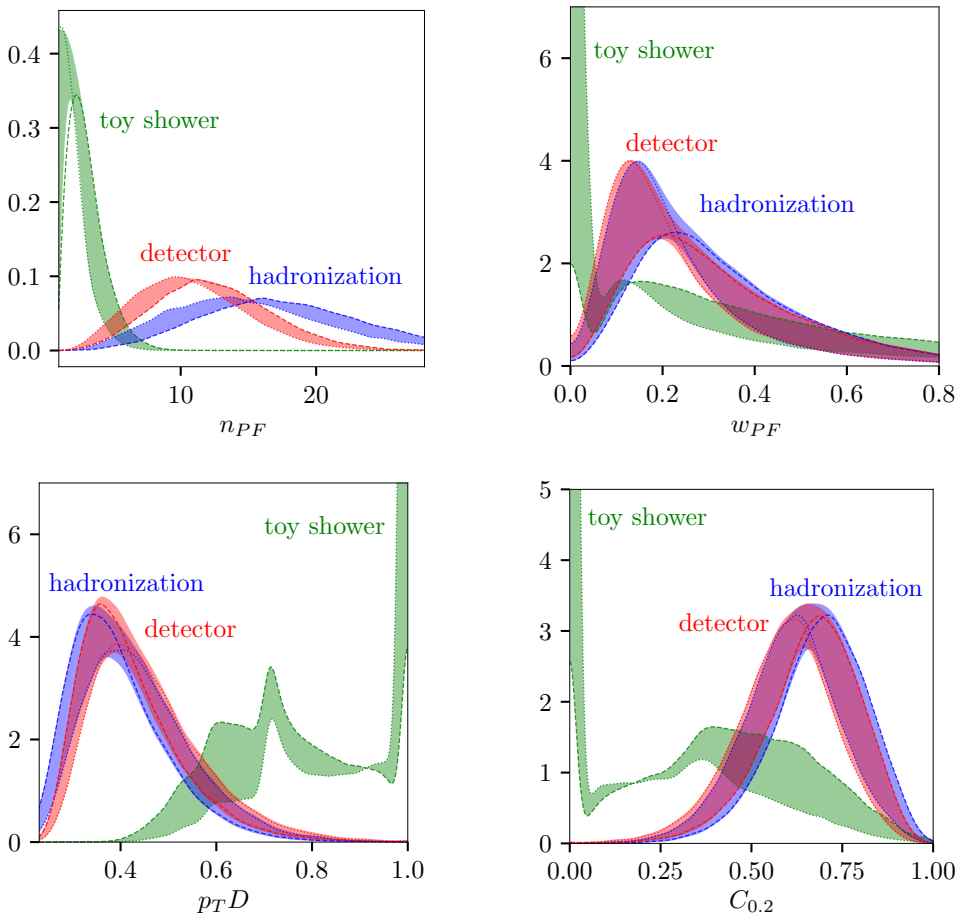

Figure 3.5: Distribution of the high-level jet observables n_{PF}, w_{PF}, $p_T D$, and $C_{0.2}$ for 10^5 jets generated with the toy setup and the SHERPA setup with and without detector simulation. Exemplary, the bands show the variation of the leading term of quark splitting between $D_{qq} = 0.5$ (dotted) and 2 (dashed).

Effect of the parametrization on the jet observables

Figure 3.5 shows the distribution of the first four observables for the toy shower setup and for the SHERPA setup directly after hadronization and including detector simulations. To show the effect of varying a parameter in our splitting kernels, bands delimited by $D_{qq} = 0.5$ and 2 are shown. The edge defined by the lower parameter is dotted, while the edge for the higher parameter is dashed.

In the upper left panel, we can clearly see the effect of hadronization and detector resolution on the particle number. As expected, the we observe a higher number of particles after hadronization. This number decreases during the detector simulation, because very soft particles can not be resolved. We also see the effect of an increased splitting kernel, which leads to a higher splitting probability. Thus, the distributions

for higher D_{qq} are shifted to the right.

Because the toy shower generates just a few splittings, its girth w_{PF} in the upper right panel accumulates at the lower limit. A slight peak can be found for the showers with enough splittings. For hadronization and detector level data, w_{PF} shows a clear peak at $w_{PF} \approx 0.2$. As for the particle number, increasing D_{qq} and thus the splitting kernel shifts the distribution to higher values.

For the toy parton shower generator, the distribution of $p_T D$ in the lower left again is dominated by the low number of splittings. A single parton has $p_T D = 1$. Adding soft radiation shifts the distribution to lower values. For events with a higher number of splittings, a second peak forms at $p_T D \approx 0.7$. For the full simulation we then see a broad peak at $p_T D \approx 0.4$. Like for w_{PF}, the distribution only changes slightly when detector effects are included. As expected, a higher splitting kernel value leads to more soft particles.

Just like the distributions of the other three observables, the toy setup distribution for the correlator $C_{0.2}$ is again dominated by $C_{0.2} = 0$ for a single parton. Hadronization then moves the broad maximum of the toy-level distribution to higher values. The distribution peaks at $C_{0.2} \approx 0.7$ and again detector effects contribute only marginally.

From this discussion, we expect big changes in the inference when going to hadronized data and only little changes when including detector effects. This is confirmed by our results in Chapter 7.

4 Introducing invertible neural networks

The aim of this work is, to use the BayesFlow algorithm [50] to infer the splitting kernel parameters, introduced in Chapter 3.1.3, from the particles calculated in our simulation chain. This inference technique is based on conditional invertible neural networks (cINN). In this chapter, we derive this architecture from the very basics of neural networks.

A neural network is a chain of linear functions, called neurons, and non-linear activation functions. This chain forms a composite function with a high dimensional parameter space. The parameters are then optimized to minimize a loss function by some form of gradient descent. The learning objective usually is for the network to transform data with a given input data distribution $p_Z(z)$ into output data with a distribution $p_Y(y)$ resembling a specific target distribution. In simple supervised models or in generative adversarial networks, this distribution is given in form of true data. And, in our case, it is simply a Gaussian distribution.

Denoting the vector of parameters of the network as ϕ, we can write the network as a function

$$f_\phi : Z \to Y, \quad z \mapsto f_\phi(z) = y \tag{4.1}$$

where z is a vector in the original space Z and y in the target space Y. In terms of probability densities of input and output we can write

$$z \sim p_Z(z) \quad \text{and} \quad y \sim p_Y(f_\phi(z)) = p_Y(y)) \,. \tag{4.2}$$

4.1 Neurons

A neuron is the basic building block of a neural network. It has n inputs z_i, which each are weighted by multiplying with a trainable weight w_i. All weighted inputs are then summed over and an additional trainable bias b can be added. Finally, the output is processed with an activation function f. We will discuss different activation functions in Section 4.3. The output y of a single neuron can be written as

$$y = f(w_i z_i + b) \,, \tag{4.3}$$

where summation over identical indices is implied.

This architecture was chosen to resemble the function of a neuron in the brain of a mammal. In the biological prototype, the neuron gets information in form of electrical impulses from multiple dentrites. The electrical potential in the neuron

sums up over time and various inputs, until it reaches a threshold. If this level is reached, the neuron fires and sends out an electrical signal via its axion, resetting its internal potential. This firing behaviour is remodeled with the activation function. However, in the brain there are a several kinds of different neurons, where in the case of an artificial neuron mostly the activation function, the connections between them and the order of different networks are altered.

4.2 Layers

As mentioned above, the connections of the neurons between each other are decisive for the networks function. While there is no limit to ones imagination in ordering neurons, it is sensible to build networks from layers differing in activation functions and architecture. There are different kinds of layers for different uses. However, we will only discuss the basics which are important for the BayesFlow architecture.

4.2.1 Dense layers

The simplest way to connect neural networks, is to use every output of the preceding layer as an input for every neuron in the current layer. This is called a fully connected or dense layer. In (4.3) we wrote the weights of a single neuron as a vector with n entries, where n was the number of inputs to the neuron. In the case of a dense layer, n is the same as the dimension of the preceding layer. The weights of all m neurons in one layer can now be summarised as a $n \times m$ matrix and the biases as a m-dimensional vector

$$y_j = f(w_{ji} z_i + b_j)\,. \tag{4.4}$$

Dense layers are the basis for many more complicated layers. Since calculating a dense layer reduces to vector calculus, it can be highly parallelised and is thus best computed on a GPU.

4.2.2 Pooling layers

Just like convolutional layers, pooling layers are most commonly used to reduce the dimensions of 2-dimensional image-like tensors. The function of pooling layers is simple. If given a input tensor of 1 (2) dimension, we split the input into non-overlapping parts of size $a(\times b)$. We then calculate the average, sum, maximum or another kind of mathematical operation for all parts. This gives us the output of the pooling layer. It is not a trainable layer, but is only used to reduce the size of the input.

In our case, we work with sets of 1-dimensional vectors, that can be stacked to a 2-dimensional tensor. We use average-pooling or sum-pooling layers to reduce tensors to the size of one 1-dimensional vector.

4.3 Activation functions

The most simple activation function one can construct for a neuron, is a linear activation function. Usually, a *linear* activation function is just the identity.

However, activation functions are used to introduce non-linearity into the neural network. With a linear activation function, a network would simply be a big concatenation of linear functions, and as such linear again. As most problems are not linear, we should only use linear activation functions in special cases, such as output layers.

A common non-linear activation function is the *Rectified Linear Unit (ReLU)*

$$\mathrm{ReLU}(x) = \max(x, 0). \tag{4.5}$$

It is very fast to compute and is therefore often used in big architectures. The ReLU has some obvious problems. First, its derivative is undefined in 0. As this is only one point, it usually is not problem for the gradient based optimisation method. It however has a problem with the vanishing derivative for values < 0. As the output in this region always vanishes, it does not change between optimization steps and can therefore not be optimized with its output being 0 for the rest of the training. The neuron is dead in some sense and only takes up resources. In principle this can be solved by assigning a small slope for negative values. We use ReLU activation functions in the summary network of our architecture.

An activation function that is differentiable in 0 and solves the vanishing gradient problem is the *Exponential Linear Unit (ELU)*

$$\mathrm{ELU}(x) = \begin{cases} \alpha\,(e^x - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}, \tag{4.6}$$

with $\alpha > 0$. As an ELU flattens for small values, it helps the network converge by reducing the outputs and thus gradients during training. We use ELU activation in the inference part of our network, because of their stability.

4.4 Loss function

Following the notion of (4.1) and the preceding sections, a neural network is concatenation of multi-parameter functions mapping from a initial probability distribution to a target probability distribution. In order to optimize the parameters of the function, we first have to introduce a metric to determine how well the network output is resembling the target distribution. This is usually formulated as the sum of a distance measure for the output point from the target point over all data points in one training step. For classical classification purposes this is often a mean-squared error (MSE) distance or cross entropy. The learning incentive of BayesFlow architectures is to transform a given distribution into a Gaussian latent distribution. In Section 5.3.2 we explain in more detail how to derive the loss function explicitly.

4.5 Gradient descent

If a loss function $L(f_\phi(z))$ for the network f_ϕ is specified, we can try to find the global minimum of this function in the high-dimensional parameter space by updating our parameter point iteratively into the direction where the loss decreases. We therefore have to calculate the derivative ∇_ϕ of the loss function with respect to every parameter in ϕ. This results in long chains of derivatives, but is in principle elementary analysis. We use this gradient to define a step in our iterative process

$$\phi_{t+1} = \phi_t - v_t := \phi_t - \eta_t \nabla_{\phi_t} L(f_{\phi_t}(z)) \,. \tag{4.7}$$

The parameter η is called learning rate and has to be tuned for every problem. A low learning rate leads to very slow convergence and a high learning rate might hinder convergence if the step size is bigger than the minimum we search for.

This simple gradient descent method also can be unstable, since the loss function might exhibit local minima or saddle points where the gradient vanishes and the optimization process stagnates.

A first improvement on this descent method, is to evaluate the loss function for batches instead of the whole data set. This introduces stochastic variations leading to a more stable routine and speeds up the calculation. It is often referred to as stochastic gradient descent [51].

A more stable algorithm can be built by adding a momentum term

$$v_t = \gamma v_{t-1} + \eta_t \nabla_{\phi_t} L(f_{\phi_t}(z)) \,, \tag{4.8}$$

where γ should be smaller than 1. This allows the network to carry on if the gradient vanishes, making it more stable against saddle points and local minima. It also allows the algorithm to pick up speed over multiple optimization steps, if the gradients are constantly small.

One way to further improve this optimization algorithm is by dynamically changing the learning rate. It improves convergence for the training to decrease the learning rate further into the training, when the parameters are close to the global minimum. This can be done by a preset exponential decay or depending on the changes of the loss function over the last optimization steps.

A more evolved algorithm is implemented in the Adam optimizer [52]. It scales the learning rate dynamically, to achieve higher learning rates into directions with less curvature. As calculating the second order derivative, that is the Hessian matrix, requires a lot of resources, the algorithm instead uses the mean and the uncentered variance of the gradient to determine the curvature. These are calculated as moving averages during training. Parameters β_1 and β_2 control the exponential decay of the moving averages

$$g_t := \nabla_\phi \left(L(f_{\phi_t}(z)) \right) \tag{4.9}$$

$$m_t = \frac{\beta_1 m_{t-1} + (1-\beta_1)\, g_t}{1-\beta_1^t} \tag{4.10}$$

$$s_t = \frac{\beta_2 s_{t-1} + (1-\beta_2)\, g_t^2}{1-\beta_2^t}. \tag{4.11}$$

As both averages are initiated as zeros, the denominator has to be included to counteract the resulting initialization bias. The update step of the parameters is then constructed according to

$$\phi_{t+1} = \phi_t - \eta_t \frac{m_t}{\sqrt{s_t} + \epsilon}. \tag{4.12}$$

The defaults for the decay parameters are $\beta_1 = 0.9$, $\beta_2 = 0.999$ and for the divergence prohibiting constant $\epsilon = 10^{-8}$. Since its introduction, ADAM has established itself as the default optimization technique for neural network purposes. We also use ADAM in this work without changing the default values, except the starting learning rate.

Since we use a stochastic gradient descent method, an optimization step is done after evaluating a batch, that is set of data points with a fixed size. In our application the batch size coincides with the amount of points processed by the network at once. One full run through all points in one data set will be referred to as an epoch.

4.6 Initialization

As mentioned for activation functions we want to avoid vanishing gradients during training. We also want to avoid gradients that are too high, as they can hinder convergence of the training. It is thus important not to set your network weights to high or to low at the beginning of your training.

One way to circumvent these problems is using a Glorot initialization, also called Xavier initialization [53]. It is a good idea to construct the initialization such that its mean vanishes. To ensure proper forward and backward (training) properties, we also want the variance over the inputs and the variance of the derivative of the loss function with respect to the outputs of one layer to be the same for all layers. These conditions can be reformulated as

$$\mathrm{Var}[W^i] = 1/n_i \tag{4.13}$$

$$\mathrm{Var}[W^i] = 1/n_{i+1}, \tag{4.14}$$

with W^i the weights in the i-th layer, n_i the number of neurons in that layer and n_{i+1} the number of neurons in the subsequent layer. As both conditions exclude each other, we have to find a compromise.

Glorot and Bengio proposed [53]

$$\mathrm{Var}[W^i] = \frac{2}{n_i + n_{i+1}}. \tag{4.15}$$

This can be satisfied by sampling the individual weights w_i from a uniform distribution $\mathcal{U}$

$$w_i \sim \mathcal{U}\left[\sqrt{\frac{-6}{n_i + n_{i+1}}}, \sqrt{\frac{-6}{n_i + n_{i+1}}}\right] \tag{4.16}$$

or from a normal distribution $\mathcal{N}$

$$w_i \sim \mathcal{N}\left[0, \sqrt{\frac{2}{n_i + n_{i+1}}}\right]. \tag{4.17}$$

The biases are initialized as 0. In our work we use a Glorot uniform initialization.

4.7 Normalizing flows

Now that we have discussed all the building blocks of neural networks, we will go into more detail on our specific application. One common task for neural networks is to model a probability density function given data from the distribution. Normalizing flows are a class of techniques designed for this purpose. Comprehensive reviews on the topic can be found in [54] and [55]. As BayesFlow in principle uses a normalizing flow to learn the probability distribution of the parameters we want to infer, we will also give a short summary in the following.

The statistical interpretation (4.1) of neural networks is the basis of describing normalizing flows. Normalizing flows are invertible and differentiable mappings of elements from a usually simple base distribution to elements with a more complicated target distribution. As the concatenation of invertible mappings is invertible again and as the same is true for differentiable mappings, multiple invertible and differentiable elements can be linked together to form a normalizing flow.

Because it is invertible by design, a normalizing flow can be used in two directions. The first direction maps the simple, often Gaussian base distribution $p_Z(z)$ to the complicated target distribution $p_Y(y)$.

$$g_\phi : Z \to Y, \quad z \sim p_Z(z) \mapsto y \sim p_Y(y) \tag{4.18}$$

We refer to it as the generative direction, because points in the target distribution can be sampled by generating points in the base distribution and using the normalizing flow on the points. The inverse direction

$$f_\phi = g_\phi^{-1} : Y \to Z, \quad y \sim p_Y(y) \mapsto z \sim p_Z(z) \tag{4.19}$$

normalizes the complicated distribution, giving this class of networks its name. We will refer to this direction as the normalizing direction. The target distribution is related to the original one as

$$\begin{aligned} p_Y(y) &= p_Z(f_\phi(y)) \left| \det \frac{\partial f_\phi}{\partial y} \right| \\ &= p_Z(f_\phi(y)) \left| \det \frac{\partial g_\phi}{\partial z} \right|^{-1}, \end{aligned} \tag{4.20}$$

where $\frac{\partial f}{\partial y}$ denotes the Jacobian of the function. In principle, it has been shown that normalizing flows can, within reasonable assumptions on both distributions, transform any base distribution into any target distribution [56]. However, to be practical in use and not only in principle, a normalizing flow should be

- efficiently invertible, to use both directions of the flow. Often training is done in normalizing direction or in both directions, while the generative direction is often of interest.
- Furthermore, for density evaluations according to (4.20) the Jacobian should be efficiently computable
- and while both of these conditions should be fulfilled, the network still needs to be expressive enough for the task at hand.

4.8 Invertible neural networks

One class of normalizing flows, which is by construction easily invertible and has an accessible Jacobian, can be constructed from multiple coupling blocks [57], each made up of two complementary affine coupling layers [58, 59]. We will refer to one block as an affine coupling block (ACB) and to the total architecture as an *invertible neural network* (INN), although all normalizing flows are constructed to be invertible. Normalizing flows built from coupling layers are called coupling flows.

The trick with our architecture is that the networks themselves do not have to be inverted, as they are only used in element-wise multiplication ($\odot$) and addition. To achieve this, one coupling block $f_\phi(Y)$ splits the input vector into two similarly sized vectors $y = (y^A, y^B)$. It then learns the representation in the base distribution from the on in the target distribution as

$$f_\phi(y) = \begin{pmatrix} z^A \\ z^B \end{pmatrix} = \begin{pmatrix} y^A \odot e^{s_2\left(y^B\right)} + t_2\left(y^B\right) \\ y^B \odot e^{s_1\left(z^A\right)} + t_1\left(z^A\right) \end{pmatrix}, \tag{4.21}$$

with the inversion

$$g_\phi(z) = \begin{pmatrix} y^A \\ y^B \end{pmatrix} = \begin{pmatrix} \left(z^A - t_2\left(y^B\right)\right) \odot e^{-s_2\left(y^B\right)} \\ \left(z^B - t_1\left(z^A\right)\right) \odot e^{-s_1\left(z^A\right)} \end{pmatrix}. \tag{4.22}$$

Here, $s_1\,, s_2\,, t_1$ and t_2 can be arbitrarily complicated functions, learned by a neural network. In principle there are no restrictions to their architecture, except that the output dimension must match the dimension of the element-wise multiplication. In practice, we choose a shallow network of equally sized, fully connected layers with ReLU activation. The exponential function is in principle not needed, but circumvents division by zero in the inverse direction. As both directions use the same neural networks, both directions are easily accessible.

The vector notation in (4.21) can be misleading, as one coupling block contains two concatenated layers. The first one only calculates z^A from y^A and y^B, the second one gives z^B in terms of z^A and y^B. The Jacobian of a coupling block can thus be calculated as the Jacobian of both functions. It is the product of two triangular matrices

$$\frac{\partial f_\phi(y)}{\partial y} = \begin{pmatrix} \mathbb{1} & 0 \\ \text{finite} & \operatorname{diag} e^{s_1\left(z^A\right)} \end{pmatrix} \begin{pmatrix} \operatorname{diag} e^{s_2\left(y^B\right)} & \text{finite} \\ 0 & \mathbb{1} \end{pmatrix}. \tag{4.23}$$

As the determinant of the product of two matrices is the product of the determinants, the Jacobian determinant of a coupling block can be computed very efficiently.

To increase the expressiveness of the neural network, we concatenate multiple coupling blocks. The architecture then fulfils all requirements of a normalizing flow by construction.

Improvements on the coupling blocks

The coupling flow architecture has been pioneered and improved by NICE [58], RealNVP [59] and Glow [60]. While the latter uses a 1×1-convolution between coupling layers to ensure the connection of every part of the input with every part of the output, we use fixed permutations between the coupling blocks, as done in NICE and RealNVP. Since our parameter space is low-dimensional compared to the image spaces investigated in the pioneering work, it contains at most 3 dimensions, using fixed permutations is expected not to limit the performance. It also guarantees a cheap inversion of the mixing.

We also apply a bijective soft clamping

$$s_i^{\text{clamp}} = \frac{2\alpha}{\pi} \arctan\left(\frac{s_i}{\alpha}\right) \quad \text{for } i = 1, 2\,, \tag{4.24}$$

with a clamping parameter of $\alpha = 1.9$ in every coupling layer. It prevents instabilities from divergent outputs and has been optimized in [4].

5 Constructing the inference method

The basic idea of inference using conditional normalizing flows is simple [3]: Train a flow to map from the inference parameters space to a simple latent space given a condition and using the generating direction with a fixed condition will give you the distribution in the parameter space. To make this more specific, we discuss the basics of Bayesian statistics in Section 5.1, as well as some established methods of likelihood-free inference in Section 5.2, before discussing the BayesFlow method in Section 5.3.

We introduce our detailed setup in Section 5.4 and showcase a method to ensure correct calibration of the trained model in Section 5.5.

5.1 Bayesian statistics

In statistics, two different approaches to probabilities are distinguished: The Bayesian and the frequentist approach.

As the name already implies, the frequentist approach treats probabilities as the relative frequency of events in an infinite repetition of the experiment. In a frequentist inference setup, usually confidence intervals for hypothesis tests are constructed, often using a (log-)likelihood ratio. Pure frequentist approaches can not account for dependencies on other variables that have a probability distributions of their own.

The Bayesian approach on the other hand interprets probabilities as the degree of belief in an event. It can be updated if information is gained using Bayes' theorem

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}. \tag{5.1}$$

Here y and x are events. In our case, y are the parameters of our simulation. The corresponding event would be "the true parameter combination is y". The second event x is a measurement of shower data.

The aim of our inference is the conditional probability of a parameter combination given a fixed measurement $p(y|x)$. It is called the *posterior*. It depends on the *likelihood* $p(x|y)$, the *marginal likelihood* $p(x)$ and the *prior* $p(y)$. The likelihood is the probability distribution of the measurement given our theory parameters and can therefore be obtained from our simulation chain numerically. The prior encodes our knowledge of the the parameters before the measurement. In standard Bayesian inference applications, the posterior of a past measurement can be used as the prior of the next measurement. This chain of *Bayesian updates* improves the posterior over consecutive measurements. The last piece of the puzzle can be calculated from

the likelihood and prior using the normalization property of a probability density fucntion

$$p(x) = \int \mathrm{d}y \, p(x|y) \, p(y) \, . \tag{5.2}$$

However, for problems with high-dimensional parameter spaces, this integral is numerically cost intensive to calculate.

While in frequentist approaches hypotheses are usually accepted or rejected within a specific confidence, Bayesian approaches can give a full probability distribution for a hypothesis under a certain condition.

5.2 Likelihood-free inference

In our case, as well as in many others, the likelihood $p(x|y)$ cannot be calculated from the the simulation parameters y analytically. Inference problems of this nature are dubbed *likelihood-free* or *simulation-based* inference problems, as the intractable likelihood has to be obtained from the simulation in a different way. In [61] the authors give an broad overview on different likelihood-free inference techniques.

As mentioned in the previous section, the simulation in principle generates a numeric approximation of the likelihood and can thus be directly used for the inference. To do so, if the output space of the simulation is high-dimensional, lower-dimensional *summary statistics* have to be derived. They have to be tailored to the inference problem at hand, as their distribution is compared between simulated and observed data. One can do so by using density estimation methods, such as histograms or kernel density approximations, to approximate the likelihood or by using Approximate Bayesian Computation (ABC).

In the past, ABC has been very popular for likelihood-free inference. The basic rejection ABC estimates the posterior distribution by running the simulation for different values of y. The value is saved as a point in the posterior, if the simulated data and the observed data are sufficiently close regarding a pre-defined distance measure. If the rejection criterion, that is the tolerance, is chosen sufficiently small, the resulting posterior tracks the posteriors well. For small tolerances ABC is ineffective and the whole simulation chain has to be repeated for new measurements. ABC therefore is an example for an inference technique that is not *amortized*. As particle physics produces many independent and identically distributed (i.i.d.) observations, amortization is an important criterion for an inference technique.

Density estimation based methods are one example for an amortized technique, because new measurements can directly be used with the estimated likelihood in both frequentist and Bayesian approaches. However, the technique still scales poorly with the dimension of the data, as the required number of simulations depends exponentially on it. Thus, effective summary statistics are very important in both approaches. Going from handmade to machine learned summary statistics is one way to improve upon these techniques using modern methods.

We have already discussed two areas in which traditional likelihood-free inference methods need to be improved.

- The first area was the *amortization* of the technique. We do not want to repeat the simulations if we collect new data.
- Also, we want to improve the *sample efficiency* of the techniques, as detectors at particle colliders produce high-dimensional data and calculating summary statistics from the observation bears the risk of losing information.
- Not losing information in summary statistics, is one possibility of improving *quality of inference* itself. The ultimate goal of the likelihood-free inference method is to approximate the true likelihood as good as possible.

Neural networks are one possibility to evaluate data that scales better with dimensionality. But, training a network directly to give point estimates of the simulation parameters is not very insightful in terms of a statistical interpretation. However, if we introduce conditions for normalizing flows, we can use them to generate a posterior parameter distribution from a latent distribution given the measurement as a condition [3].

5.3 BayesFlow networks

This is exactly what BayesFlow does using a conditional form of INNs [50]. The network will be trained for the number of measurements to be within a certain interval. As such, it is only amortized within this interval. Although, Bayesian updating can in principle be used with the BayesFlow method, it is very inefficient as the network has to be retrained.

As a neural network application, it should scale well with the dimensionality of the data, although we limit this analysis to 52-dimensional measurements.

The quality of the inference will be estimated in the chapters 6 and 7.

5.3.1 Conditional coupling blocks

To use INNs from Section 4.8 for inference purposes, we need a conditional normalizing flow [3]

$$f_\phi(y\,;x^*) : Y \to Z, \quad y \sim p_Y(y|x = x^*) \mapsto z \sim p_Z(z)\,. \tag{5.3}$$

The inverse direction $g_\phi(z\,;x^*)$ then transforms the latent distribution $p_Z(z)$ back to the posterior distribution $p_Y(y|x = x^*)$. Sampling from the latent distribution and applying the network in generating direction samples the posterior distribution as can be proven using of the change of variable formula (4.20) [50].

To derive this, let $p_\phi(y|x)$ be the learned and $p(x|y)$ the true posterior. We suppress writing the space of the probability distributions for brevity. Assuming a perfectly converged normalizing flow f_ϕ, the learned posterior distribution is

$$p_\phi(y|x^*) = p(f_\phi(y\,; x^*)) \left|\det \frac{\partial f_\phi}{\partial y}\right| \tag{5.4}$$

and in in normalizing direction the prior is

$$p(z) = p(y|x^*) \left|\det \frac{\partial g_\phi}{\partial z}\right| . \tag{5.5}$$

Using that the normalizing direction is the inverse of the generating direction and the multiplicativity of the determinant, we can show that the learned posterior indeed is the true one

$$\begin{aligned} p_\phi(y|x^*) &= p(z) \left|\det \frac{\partial f_\phi}{\partial y}\right| \\ &= p(y|x^*) \left|\det \frac{\partial g_\phi}{\partial z}\right| \left|\det \frac{\partial f_\phi}{\partial y}\right| \\ &= p(y|x^*) \left|\det \left(\frac{\partial f_\phi}{\partial z}\right)^{-1}\right| \left|\det \frac{\partial f_\phi}{\partial y}\right| \\ &= p(y|x^*) \left|\det \left(\frac{\partial f_\phi}{\partial z}\right)^{-1} \frac{\partial f_\phi}{\partial y}\right| \\ &= p(y|x^*)\,. \end{aligned} \tag{5.6}$$

The INN (4.21) can be easily redesigned to include conditions and to learn this function by concatenating every input with the condition

$$f_\phi(y\,; x^*) = \begin{pmatrix} z^A \\ z^B \end{pmatrix} = \begin{pmatrix} y^A \odot e^{s_2\left(y^B, x^*\right)} + t_2\left(y^B, x^*\right) \\ y^B \odot e^{s_1\left(z^A, x^*\right)} + t_1\left(z^A, x^*\right) \end{pmatrix} . \tag{5.7}$$

We will refer to this as a *conditional invertible neural network* (cINN).

Notation

As mentioned in the introduction to this section, we want our model to work with sets of measurements. We will therefore denote a set of simulated or measured data for parameters y as $\boldsymbol{x}_{1:M} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_M)$, where every entry is a scalar or vector. In our case, every entry is a 52-dimensional vector containing particle 4-momenta from the simulation. The parameters of our model build a L-dimensional vector $\boldsymbol{y} = (y_1, ..., y_L)$. Our model has 7 total parameters, however we only examine subsets with $D = 2$ or 3 at a time. The normalizing flow thus is a function

$$f_\phi : \mathbb{R}^L \to \mathbb{R}^L\,.$$

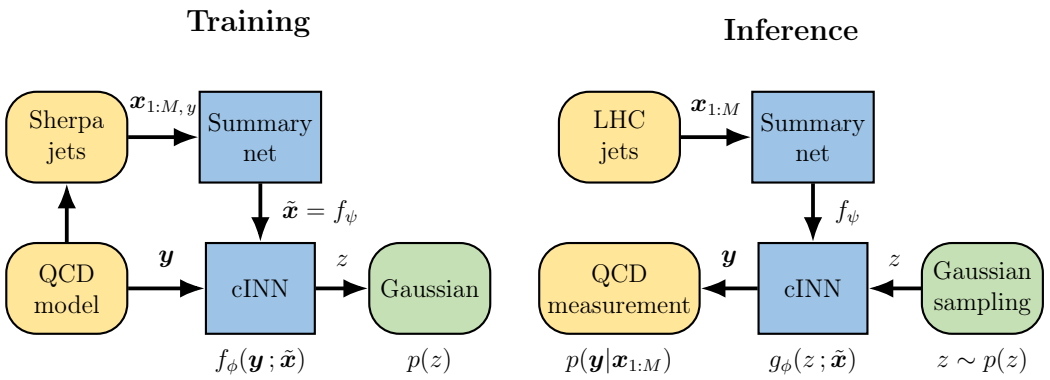

Figure 5.1: BayesFlow setup of the cINN for training and inference as in [50] adapted from [6]

If we assume the existence of a fixed sized summary statistic η containing all the information of $\boldsymbol{x}_{1:M}$ on the parameters $\boldsymbol{y}$, the same arguments as in (5.6) can be used to prove the convergence of the setup for sets of multiple measurements. However, the values of this vector or if it exists are a priori not clear. The best estimate is to use a neural network $f_\psi(\boldsymbol{x}_{1:M})$ to process the data and train both networks simultaneously to reproduce the true posterior distribution as good as possible. For perfect convergence, the summary network then learns the maximally informative summary statistic

$$\eta = f_\psi(\boldsymbol{x}_{1:M}) =: \tilde{\boldsymbol{x}}\,.$$

We will call this network the *summary network* as it generates summary statistics of the measurement. Figure 5.1 shows a graphic of both, the normalizing, training direction and the generative, inference direction.

5.3.2 Learning objective

From the discussion in the previous section it is clear, that we want to train our network to learn the true posterior as good as possible. To do so, we need a measure to compare the true and the estimated probability distribution with. One popular choice for fitting a flow-based model to a target distribution is the *Kullback-Leibler* (KL) *divergence* [55, 62].

The KL divergence (or relative entropy) is the expectation value of the logarithmic difference between two probability distributions. In more intuitive words, it measures how different two probability distributions $p(x)$ and $q(x)$ are or how much information is gained going from p to q. It is defined as

$$\mathbb{KL}(p\|q) = \int \mathrm{d}y\, p(y) \log \frac{p(y)}{q(y)}\,. \tag{5.8}$$

As can be readily seen, it is positive semi-definite and vanishes if, and only if both distributions are the same.

Thus perfect convergence of our inference setup is defined by

$$\mathbb{KL}(p(\boldsymbol{y}|\boldsymbol{x}_{1:M}) \,\|\, p_\phi(\boldsymbol{y}|\boldsymbol{x}_{1:M})) = 0\,. \tag{5.9}$$

Minimizing the KL divergence between both distributions can be used as a training objective. We get the optimal parameters of the cINN $\hat{\phi}$ and summary network $\hat{\psi}$ as

$$\begin{aligned}\hat{\phi}, \hat{\psi} &= \arg\min_{\phi,\psi} \left\langle \mathbb{KL}(p(\boldsymbol{y}|\boldsymbol{x}_{1:M}) \| p_\phi(\boldsymbol{y}|\boldsymbol{x}_{1:M})) \right\rangle_{\boldsymbol{x}} \\ &= \arg\min_{\phi,\psi} \left\langle \log p(\boldsymbol{y}|\boldsymbol{x}_{1:M}) - \log p_\phi(\boldsymbol{y}|\boldsymbol{x}_{1:M}) \right\rangle_{\boldsymbol{x},\boldsymbol{y}} \\ &= \arg\min_{\phi,\psi} \left\langle -\log p_\phi(\boldsymbol{y}|\boldsymbol{x}_{1:M}) \right\rangle_{\boldsymbol{x},\boldsymbol{y}}\,,\end{aligned} \tag{5.10}$$

where we neglected terms independent of ϕ or ψ. As we do minibatch optimization, we write this for one batch with N simulated data sets and data generating parameters

$$\hat{\phi}, \hat{\psi} = \arg\min_{\phi,\psi} -\frac{1}{N} \sum_{i=1}^{N} \log p_\phi(\boldsymbol{y}^i|\boldsymbol{x}^i_{1:M})\,. \tag{5.11}$$

Using the change of variables formula (4.20) to write the estimated posterior in terms of the latent space distribution and enforcing a normal distributed latent space with unit standard deviation, we finally arrive at

$$\begin{aligned}\hat{\phi}, \hat{\psi} &= \arg\min_{\phi,\psi} \frac{1}{N} \sum_{i=1}^{N} \left(\log p(f_\phi(\boldsymbol{y}^i; f_\psi(\boldsymbol{x}^i_{1:M}))) - \log \left| \det \boldsymbol{J}^i_{f_\phi} \right| \right) \\ &= \arg\min_{\phi,\psi} \frac{1}{N} \sum_{i=1}^{N} \left(\frac{\| f_\phi(\boldsymbol{y}^i; f_\psi(\boldsymbol{x}^i_{1:M})) \|_2^2}{2} - \log \left| \det \left(\boldsymbol{J}^i_{f_\phi} \right) \right| \right) \\ &=: \arg\min_{\phi,\psi} L(f_{\phi,\psi}(\boldsymbol{y}))\,,\end{aligned} \tag{5.12}$$

in terms of a batch-wise loss function $L(f_{\phi,\psi}(\boldsymbol{y}))$.

5.3.3 Summary network

To learn maximally informative summary statistics of the data sets $\boldsymbol{x}_{1:M}$, we need a summary network that is tailored to the task at hand. In [50] the authors for example use long-short term memory (LSTM) recurrent networks for the inference analysis of time series data. This is not necessary for our purpose, as the shower events in our data sets are independent and identically distributed. For our events the summary statistic should be independent of the ordering of the shower events within $\boldsymbol{x}_{1:M}$. Ideally, they are also independent of the ordering of the single 4-vectors within one shower event. In practice however, the inference is highly dependent on this preprocessing of the data, as will be seen in the Section 6.

To obtain summary statistics of constant length independent of M and to be invariant under permutations of $\boldsymbol{x}_{1:M}$, our summary network needs to have a pooling layer. In the simplest case, the summary network uses a fully connected sub-network f_{ψ_2} before and another one f_{ψ_1} after pooling. The full summary network then acts on the input data $\boldsymbol{x}_{1:M} = (x_1, ...x_M)$ as

$$\tilde{\boldsymbol{x}} = f_\psi(\boldsymbol{x}) = f_{\psi_1}\left(\frac{1}{N}\sum_{i=1}^{N} f_{\psi_2}(\boldsymbol{x}_i)\right), \tag{5.13}$$

In our case, we use mean-pooling, however one can also use a sum-pooling here. Sum-pooling has the advantage that the summary statistic also contains information on M, but it may lead to less stable training, as the output of the pooling layer is not bounded from above. We solve this by concatenating $\sqrt{M}$ to the output of the pooling layer when training with varying M.

In practise we found that the second network f_{ψ_1} did not improve the performance of the inference. We therefore only used one fully connected network with subsequent mean-pooling. The summary network architecture we found to work best is six layers with $64, 64, 64, 64, 32$ and 32 nodes, ReLu activation in the first five layers and one layer with ELU or linear activation to avoid vanishing summary statistics. A further reduction of the output dimensions led to more unstable and over all worse performing networks.

Attention mechanism

In [50] an additional attention mechanism for the summary network is introduced. A third sub-network f_{ψ_3} calculates weights for the events of one data point before pooling

$$\tilde{\boldsymbol{x}} = f_\psi(\boldsymbol{x}) = f_{\psi_1}\left(\frac{1}{N}\sum_{i=1}^{N} \boldsymbol{w}_i \odot f_{\psi_2}(\boldsymbol{x}_i)\right) \tag{5.14}$$

$$\boldsymbol{w}_i = \exp(f_{\psi_3}(\boldsymbol{x}_i)) \oslash \sum_{j=1}^{N} \exp(f_{\psi_3}(\boldsymbol{x}_j))\,. \tag{5.15}$$

In theory the summary network should then be able to weight different events corresponding to one parameter combination, and for example neglect showers without splittings. Like the network after pooling, this did not improve our results.

Three stage summary network

We also used a similar attention mechanism to construct an architecture invariant under the sorting of the 4-momenta. To achieve this, we first used a network on the individual 4-momenta, weighted the outputs of the first network with the attention mechanism, pooled the results for individual shower events and used the previously

defined network architecture from here. This setup did not yield better results than a single fully convolutional network with a pooling layer.

Even when exchanging the weighting algorithm with a self-attention mechanism as described in [63] to allow the network to learn from pair-wise correlations of the momenta, the results did not improve.

Recursive Summary network

Following [64], we also experimented with a recursive network architecture. The architecture follows the tree-like structure of the parton shower by repeatably using a rather small fully convolutional network on the 4-momenta and the network outputs. One asset of this architecture is, that differing numbers of particles are naturally part of the tree structure and do not have to be implemented by 0's. We found the architecture to give equally good results to the simple summary network, even with much smaller numbers of notes. A further benefit from the invariance in the particle ordering could not be observed.

5.4 Inference setup

Information on the setup used in this work has been scattered between the discussions on the theoretical foundation throughout the preceding sections. For clarity, we summarize all important parameters of our setup in Table 5.1. With all relevant steps introduced, we present the full methodology of our approach, schematically presented in Figure 5.1.

We start at our QCD model from Section 3.1.3. It includes 7 parameters for the distributions of our splitting kernels. However, we only examine $L = 2$ or

	Symbol	Value
Number of parameters	L	$2, 3$
Maximum number of constituents	F	13
Jets per parameter point (variable/fixed)	M	10^2 ... 10^5 / 10^4
Batch size	N	16
Batches per epoch	E	6250
Output dimension summary network	S	32
Fully connected summary net architecture	S_i	64,64,64,64,32,32
Coupling blocks	n_{blocks}	5
Fully connected coupling layer architecture	s_i/t_i	64,64,64
Epochs	e	10 ... 40
Decay steps (toy shower/PF flow)	n_s	200 ... 500 / 500 ... 1000
Learning rate after t batches	η_t	$10^{-3} \cdot 0.99^{\lfloor t/n_s \rfloor}$
Training/testing points		100k / 10k

Table 5.1: Network and training hyperparameters.

3-dimensional subspaces. We concentrate on the inference of

- the **hierarchically ordered terms** of the quark-quark splitting D_{qq}, F_{qq} and C_{qq},
- the **soft-collinear leading terms** D_{qq} and D_{gg}
- and the p_T**-suppressed rest terms** C_{qq}, C_{gg} and C_{gq}.

When varying these parameter combinations away from their SM point, we keep all other 5 or 6 parameters at their SM values. Per batch we choose $N = 16$ parameter combinations from a completely uniform prior. Due to the cutoff for negative kernel values, we do not have to implement complicated boundary conditions for the prior.

For every parameter combination we generate M quark showers with the toy setup from Section 3.2.3 or with the SHERPA setup described in Section 3.3.4. We get M arrays of momenta of partons or particle flow objects, which we then crop or pad with zeros to contain F particles. More information on data generation and handling is given in Appendix B.

The summary network then pools this $N \times M \times F \cdot 4$ dimensional tensor over all showers generated for the same data point. In our case, the summary network consists of six fully connected layers with ReLU activation, ELU or linear activation in the last layer and the described mean pooling afterwards. The layers have the dimensions $64.64, 64, 64, 32$ and 32. Thus the summary network returns a tensor of $N \times S$ per batch with $S = 32$.

Alternatively, we can use arrays of the high-level observables from Section 3.4 instead of the four-momenta. This changes the input dimension of the summary network from $F \cdot 4$ to the number of high-level observables used. Although the input dimension is significantly smaller now, we have found the best results when leaving the summary network architecture unchanged to preserve the expressiveness of the network.

If we want to train the right scaling behaviour as well, we have to train varying M from batch to batch. As we use mean-pooling in the summary network, the summary statistics do not contain any information on M. We found that appending $\sqrt{M}$ to the output leads to the most stable results for the scaling with the measurement size. The summary network then transfers a tensor of size $N \times S + 1$ to the inference network.

The inference network itself consists of $n_{blocks} = 5$ coupling blocks. The four networks in each coupling block have three fully connected layers with 64 nodes and ELU activation.

To construct the whole architecture we use TENSORFLOW r1.14 [65]. We initialize the system with the built-in Glorot uniform initialization and train the network with the built-in ADAM optimizer to map to a Gaussian latent space. We implement an additional stepwise exponential decay function for the learning rate after t batches

$$\eta_t = 10^{-3} \cdot 0.99^{\lfloor t/n_s \rfloor} . \tag{5.16}$$

Adapting the decay rate of the learning rate is the most effective tool to manipulate convergence and thus quality of the inference.

After training, we use the network to infer a parameter distribution from a simulated or measured data set. L-dimensional points are sampled from the latent space and transformed to parameter points by the cINN. We can then use histograms or kernel-estimation techniques to relate the point cloud to a density.

Bayesian updating

Since we use the prior to sample from in the training stage, a Bayesian updating is very ineffective in this setup, as the network needs to be trained again for the new prior. However, we have found similar results when training with $M = 10^5$ using the posterior of an earlier training with $M = 10^5$ as the prior and when training the same setup for $M = 2 \cdot 10^5$ showers.

5.5 Simulation-based calibration

One problem with likelihood-free inference methods is estimating the correctness of the results. For Bayesian methods capable of generating posterior samples, such as BayesFlow, simulation-based calibration (SCB) [66] is one method of validating the inference. It uses the *data averaged posterior*

$$\int \mathrm{d}\tilde{x}\, \mathrm{d}\tilde{y}\, p(y|\tilde{x})\, p(\tilde{x}|\tilde{y})\, p(\tilde{y}) \overset{!}{=} p(y), \tag{5.17}$$

which analytically is equal to the prior of the model. A mismatch between both thus indicates an error in the Bayesian inference. This self-consistency check can be performed by examining histograms of rank statistics. For a well calibrated model they should be uniform.

Consider model parameters chosen from the prior $\tilde{y} \sim p(y)$, data simulated given these parameters $\tilde{x} \sim p(x|\tilde{y})$, a set of parameters generated by Bayesian inference $\{y_1, ..., y_L\} \sim p(y|\tilde{x})$ and an arbitrary one-dimensional random variable f. According to (5.17), the rank statistic of the prior sample with respect to the posterior sample regarding the random variable

$$\#\left\{y_i \text{ with } i = 1, ..., L \,|\, f(\tilde{y}) < f(y_i)\right\} \in [0, L] \tag{5.18}$$

has to be uniformly distributed. A proof can be done by calculating the probability distribution of the rank statistic directly and is given in [66].

Generating histograms of the rank statistic of single prior samples, with respect to a set of corresponding posterior samples for all dimensions separately, is therefore a self-consistency check of the system. Not only does the uniformity of the histograms confirm the consistency of the method, certain kind of variations also track specific problems:

- A uniform distribution with excess at the edges indicates correlated posterior samples, as the posterior samples are, due to the correlation, either higher or lower than the prior sample on average.
- A similar distribution is obtained for over-contracted posteriors. If the computed posterior is narrower than the true posterior, that is the prior, the samples from the prior are again biased towards the edges of the histogram. This time the shape of the histogram is convex.
- By the same logic, an overdisperse posterior leads to an excess in the middle of the histogram and therefore a concave shape.
- If the inference misidentifies the actual point, that is the computed posterior is biased towards one direction, the histogram will lean towards the same side.

As we use a summary network before the inference step, the results of the SBC have to be handled with care. Uniform histograms in SBC are necessary for a consistent inference, however they do not guarantee good results, as a badly trained summary network can hinder the inference without causing inconsistencies in the SBC. We check the SBC distribution for every run and only present well calibrated results.

6 Benchmarking inference quality with the toy shower

For a first benchmark of the networks performance, we generate data from our toy setup. As explained in Section 3.2.3, it calculates the simple hard process

$$e^+e^- \to q\bar{q}$$

and implements a parton shower for one of the two quarks, ignoring all sub-leading jets. To keep the momentum transfer to the second quark as small as possible, we only allow the second quark to be the spectator of the first scattering. The spectators for later scatterings are fixed to be products of the shower.

The calculations are done at 91.1786 GeV, the mass of the Z-boson, the running coupling is chosen such that $\alpha_s(m_Z) = 0.118$ and the parton-shower cutoff is set to 1 GeV.

6.1 Gluon-radiation shower

In the very first step, we restrict the QCD splittings to the P_{qq} kernel. This corresponds to successive radiation of soft gluons of the hard quark. Because the gluon shower can be parallelised to a high degree, we can use the GPU to generate data during training. As generating datasets for the full parton shower is numerically cost intensive, we use the gluon-radiation shower to benchmark the sorting and show the scaling behaviour of the posteriors.

During training, we choose the three parameters of the kernel to be uniformly distributed in

$$\{D_{qq}, F_{qq}, C_{qq}\} \in [0.5, 2] \times [0, 4] \times [-10, 10]\,. \tag{6.1}$$

Effects of sorting

As explained in Section 3.2.3, in the toy shower the ordering of the data is given by the appearance of the splittings in the shower generator. This truth-sorting constitutes an information backdoor, as such we expect it to give the best results. Figure 6.1 shows the posterior probabilities of the gluon shower parameters for truth- and k_T-sorting for training on $M = 10^2$ to 10^5 jets per parameter point distributed with $1/M$ and $M_{\text{eval}} = 10^4$ jets in the measurement. As can be seen, we find approximately Gaussian posteriors. Unsurprisingly, the sensitivity is higher for truth-sorting. This is partly caused by the correlations introduced by going from

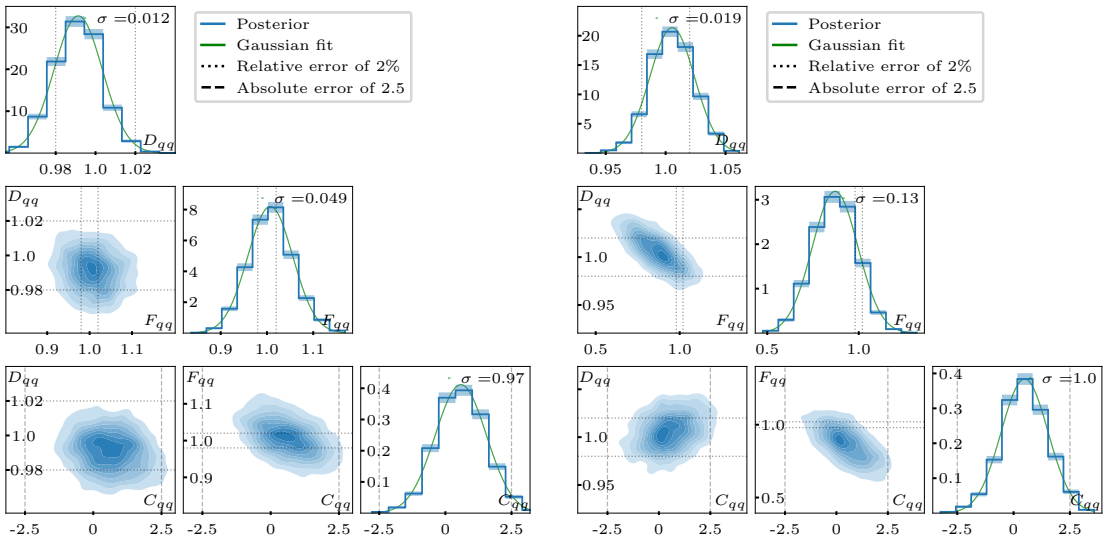

Figure 6.1: Posterior probabilities of the hierarchical parameters of the gluon-radiation shower , $\{D_{qq}, F_{qq}, C_{qq}\}$. We measure 10^4 SM-like jets and consider truth-sorting (left) and k_T-sorting (right).

truth- to k_T-sorting, because the marginal posteriors are generated as the projection of the whole prior onto the specific dimension.

The posteriors also depict the hierarchy of the different terms. The standard deviation of the leading term $\sigma(D_{qq}) = 0.019$, the finite term $\sigma(F_{qq}) = 0.13$ and the p_T-suppressed rest term $\sigma(C_{qq}) = 1.0$ are separated by almost a full order of magnitude each. A summary of all results, including the width of the 1-dimensional distribution obtained at the SM values of the other parameters to counter the effects of the correlations, can be found in Table 8.1.

Scaling behaviour

The training was done on on varying number of jets per parameter point. Therefore, the network should have also learned the right scaling behaviour of the measurement. To test this, we evaluate the SM measurement for different numbers of showers M_{eval}. For every value of M_{eval} we take 200 independent measurements in form of point clouds estimating the posterior. Every point cloud has 2000 points.

Figure 6.2 shows that the uncertainty of the measurement, that is the mean of the standard deviation of the 1-dimensional posterior over all measurements with the same M_{eval}, scales with $1/\sqrt{M_{\text{eval}}}$. This is exactly what we want for a statistically limited measurement. The standard deviation of the posterior also overestimates the true error slightly. The true error is generated as the average deviation between the point cloud mean and the SM value.

For large statistics, the scaling behaviour exhibits a small deviation from the expected behaviour, especially for the rest terms. It is caused by oscillations of the

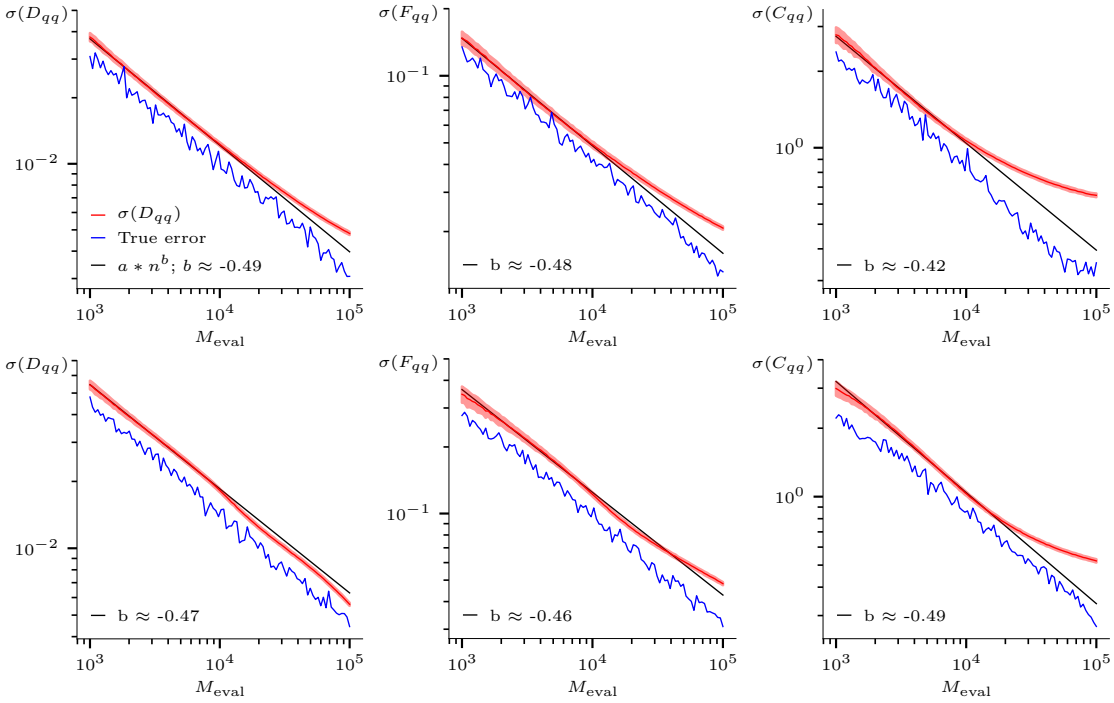

Figure 6.2: Dependence of the error on the measurement size, for the measurement of the hierarchical parameters of the gluon-radiation shower, $\{D_{qq}, F_{qq}, C_{qq}\}$. The standard deviation of the posterior (red), a fit to the standard deviation (black) and the absolute difference between the estimated and true parameters (blue) are shown. We consider truth-sorting (upper) and k_T-sorting (lower)

inferred value around the true value. As C_{qq} is has the smallest effect and thus is the hardest to learn, the inferred value for the rest term shows the biggest oscillations. For high statistics they are relevant and hinder the network from learning the right scaling behaviour as the oscillation exceed the statistical limitations of the measurement.

Usually, this can be solved by further tuning of the learning rate to suppress the oscillations in combination with choosing M from a uniform distribution to enlarge the number of training points at high statistics. However, this comes at a high computational cost.

Calibration

The good agreement between standard deviation and true error in Figure 6.2 already indicates a good calibration of the model. However, it only corresponds to a single point in parameter space. We want to make sure the network performs well on the whole prior.

Figure 6.3 shows how well the network approximates the true value of a measurement for the whole prior. We chose approximately 1000 parameter combinations

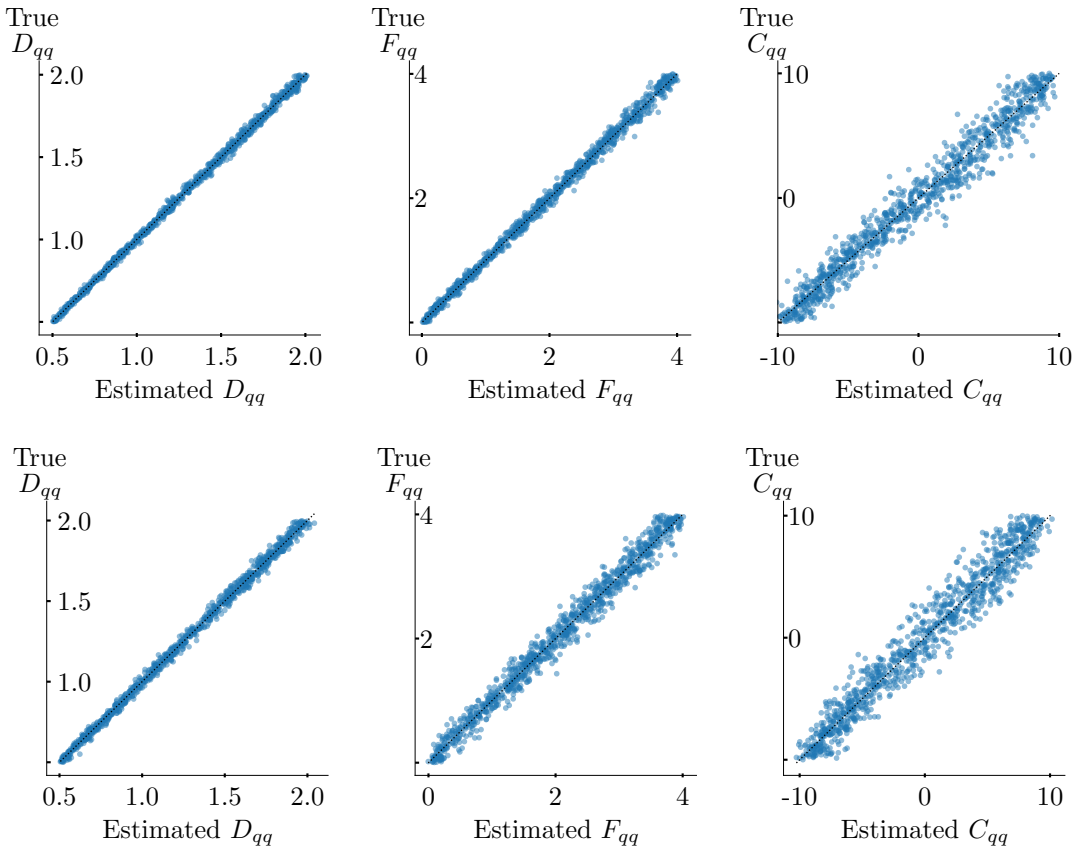

Figure 6.3: True values over posterior means for the hierarchical parameters of the gluon-radiation shower, D_{qq} (left), F_{qq} (middle) and C_{qq} (right), for parameter combinations drawn from the prior ($M_{\text{eval}} = 10^4$). We compare truth-sorting (upper) and k_T-sorting (lower).

from the prior and calculate $M_{\text{eval}} = 10^4$ showers for every point. For each measurement, we then sample 2000 points from latent space and transform them to a point cloud of the posterior using the generative direction of our network. We show the true parameter combination over the mean of the point clouds, that is the estimated parameter value. As can be seen, the network performs similarly well for every combination in the prior and shows no bias. The spread of the graph corresponds well to uncertainty obtained for the SM point in Figure 6.1. We check this for every run in the following, but exclude them from the main evaluation as they do not allow any more insights then the posteriors at the SM point. The evaluations for the whole prior for k_T-sorted toy setup and including hadronization and detector effects are attached in Appendix C.

To guarantee good calibration all over the prior, we also employ the simulation-based calibration method introduced in Section 5.5. Figure 6.4 shows the model is well calibrated.

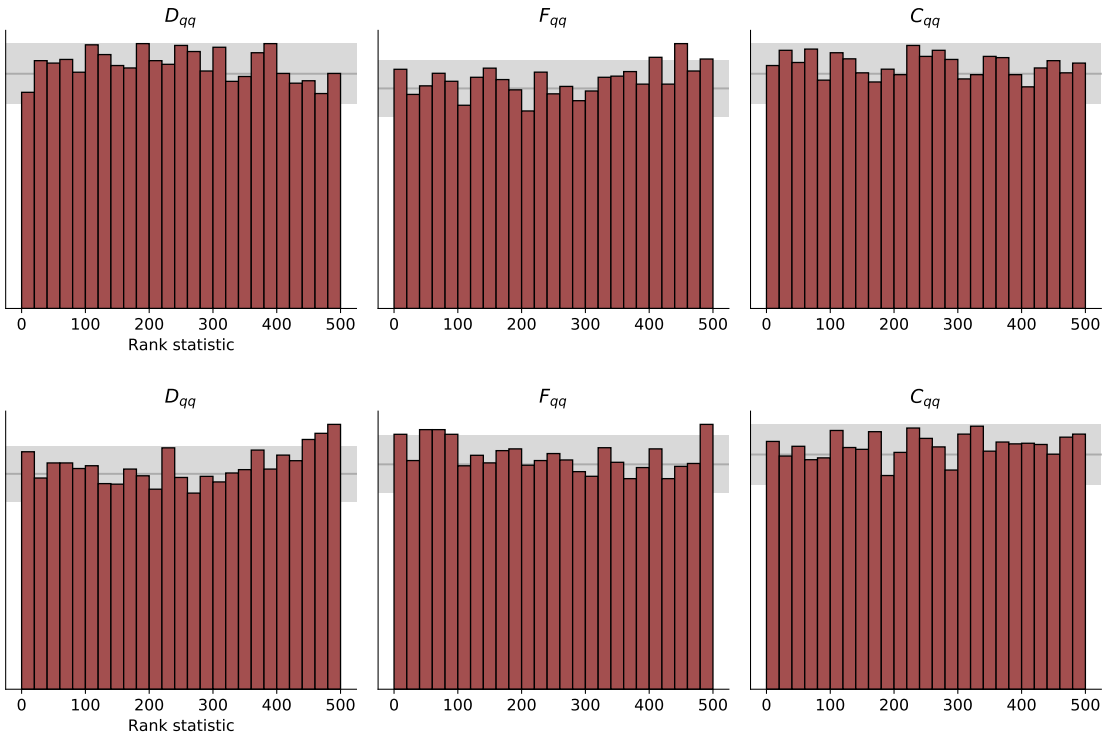

Figure 6.4: Simulation-based calibration plots for the Bayesian inference of the hierarchical parameters , $\{D_{qq}, F_{qq}, C_{qq}\}$, of the gluon-radiation shower in truth-sorting (upper) and k_T-sorting (lower).

For later results, we always ensure a good calibration using this method. However, for brevity sake we only present this for the current, representative example.

Comparison to high-level observables

We have already established the effect of the sorting on the inference quality. A natural continuation of this question is, whether using the permutation invariant high-level observables from Section 3.4 effects the results significantly. This is also interesting, as it allows us to judge the impact of using low-level 4-momenta in the first place.

We use the same setup with variable M and $M_{\text{eval}} = 10^4$ as before but train on all six previously defined jet observables. In Figure 6.5 we can see, that going from high-level observables to k_T-sorted momenta improves the results and reduces correlations. Clearly, the truth-sorting is superior and shows the room for improvement concerning the sorting algorithm.

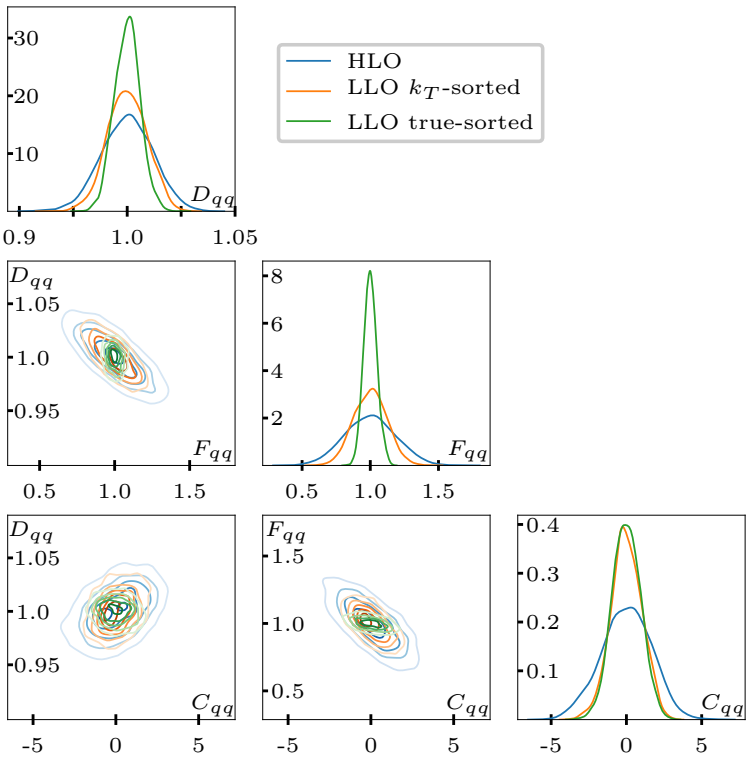

Figure 6.5: Comparing the posterior probabilities of the gluon-radiation shower parameters, $\{D_{qq}, F_{qq}, C_{qq}\}$, for training on low-level observables with both sortings (compare figure 6.1) and high-level jet observables.

6.2 Inference from a minimal number of high-level observables

An additional question, diagonal to the main goal of this work, is how many high-level observables are actually needed to resolve all three parameters and how expressive the individual jet observables actually are.

In principle the distribution of one high-level observable might already contain enough information to infer certain parameters. Again, we examine this in the most simple case, the gluon radiation shower, varying the three hierarchically ordered terms D_{qq}, F_{qq} and C_{qq} and training on a single observable at a time.

We find that the network is able to reconstruct the leading term for every observable. Reasonable posteriors for all three parameters are generated when using the distributions of $C_{0,2}$, x_{max}, $p_T D$ and with larger uncertainties also for the rest terms also w_{PF}. The particle number observables n_{PF} and N_{95} seem to be less expressive and the network has problems inferring more than the leading contribution.

The best results we find for $C_{0,2}$. This does not surprise as $C_{0,2}$ was optimized to discriminate between quarks and gluons and the varying the parameters manipulates

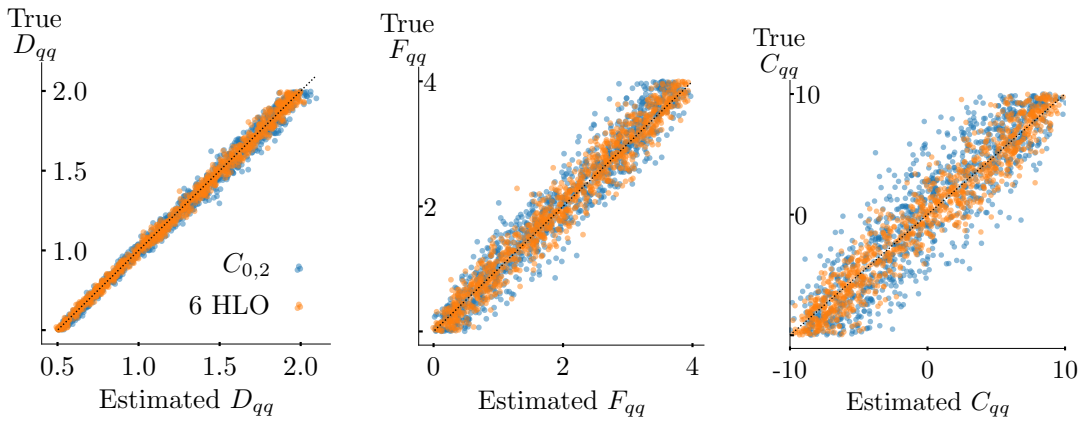

Figure 6.6: True values over posterior means for the hierarchical parameters of the gluon-radiation shower, D_{qq} (left), F_{qq} (middle) and C_{qq} (right), for parameter combinations drawn from the prior ($M_{\text{eval}} = 10^4$). We compare the results from training on 6 high-level observables (blue) and a single observable $C_{0,2}$ (orange).

the radiation of a single gluon. In Figure 6.6 we show that the performance on a single observable can be compared to that on all six. Both networks track the true values reasonable well and the spread using only $C_{0,2}$ is only slightly larger than for a training with all six jet observables.

It can therefore be instructive to compare the influence of the parameters on the distributions of the observables by hand, to judge the complexity of the inference task. Figure 6.7 shows the effect of varying every parameter in the splitting kernels on the distribution of $C_{0,2}$ for the toy setup without hadronization. We see the lower limit of the priors chosen in this work in blue, the SM value in orange and the upper limit in green. All values that were not altered remain at the SM value. The upper plot depicts the histograms over 10^4 showers, where we cut off the access at $C_{0,2} = 0$ caused by showers with no splittings, while the lower plot illustrates the difference between the histograms generated at the limit values and the one at the SM value. Statistical errors are shown as a shaded band.

In general, we find the same results as in Section 3.4. Increasing the splitting kernel, that is increasing a parameter value, shifts the distributions to the right. As can be seen, the influence of the quark splitting parameters is clearly visible and statistically relevant and the effect of the different terms decreases hierarchically. Small changes in the leading term lead to big changes in the distribution of $C_{0,2}$, while the effect of the finite and rest term are smaller.

We also find a relevant effect when altering the leading term of the gluon splitting. However, from the distributions we conclude, that inferring D_{gg} is significantly harder than inferring D_{qq}.

For the rest terms of both gluon splitting kernels, we find no significant changes in the distributions. Inferring these parameters from high-level observables seems

to be a very hard task.

Including hadronization and detector effects dramatically increases the number of particles and smears out the effects of the parameters even further.

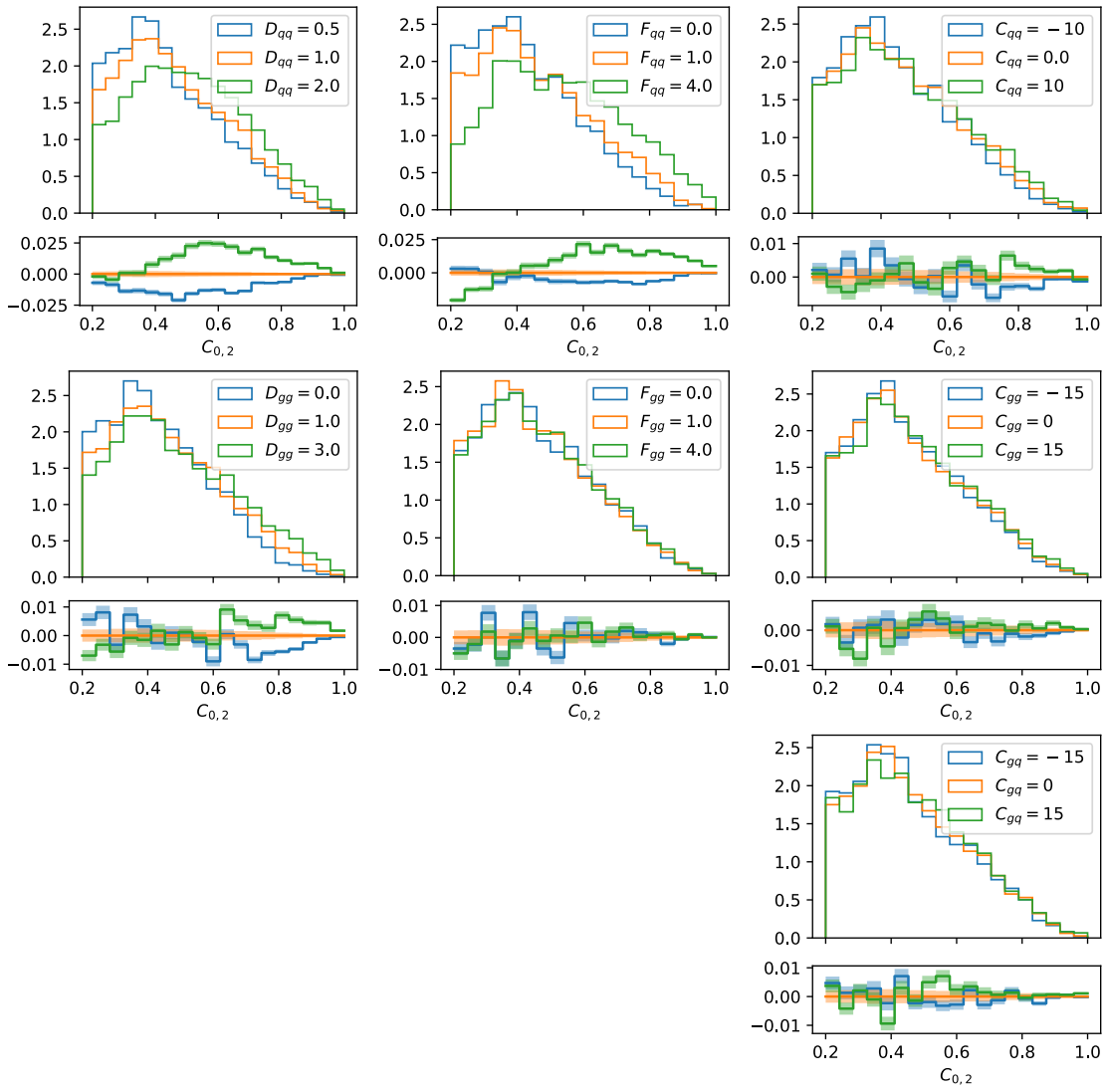

Figure 6.7: Effects of varying all parameters in the splitting kernels on the distribution of $C_{0,2}$ for 10^4 showers from the toy shower setup without hadronization. We show the distribution for the lower limit of the respective parameter (blue), the SM value (orange) and the upper limit (green).

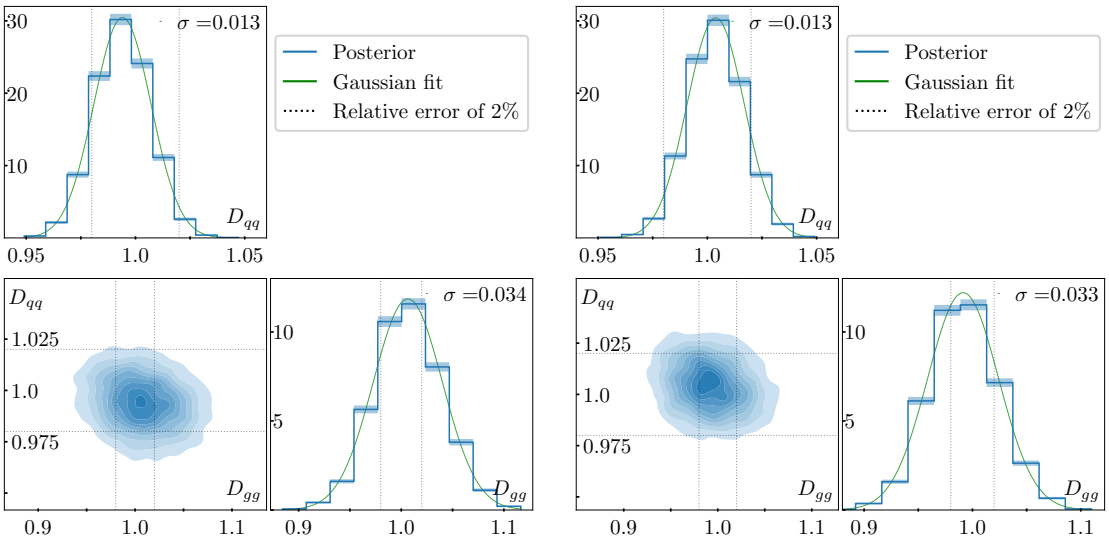

Figure 6.8: Posterior probabilities of the soft-collinear leading terms , $\{D_{qq}, D_{gg}\}$, for the full toy shower. We measure $M_{\text{eval}} = 10^4$ SM-like jets and consider truth-sorting (left) and k_T-sorting (right).

6.3 Full parton shower

In the next step, we include all QCD splittings in the toy shower, which slows down the generation of data considerably. We thus have to start training on a fixed dataset of 10^5 parameter points. As we have already proven the scaling behaviour, we generate $M = 10^4$ showers per parameter combination.

Leading terms

For this setup, we can now compare the contributions of the leading terms of quark-gluon and the triple quark interactions. We vary them within

$$\{D_{qq}, D_{gg}\} \in [0.5, 2] \times [0, 3]\,, \tag{6.2}$$

and find a significantly smaller width for the measurement of the D_{qq} than for D_{gg} in Figure 6.8. This makes sense. Since we evaluate showers with an initial quark, the hardest splitting has to be a quark-gluon splitting. The influence of the quark-gluon splitting, and thus of D_{qq}, has to be higher than the one of the triple gluon interaction. If this is the true reason of the difference, it can be corrected by including showers with initial gluons.

This measurement is especially interesting, as the measurement of the leading terms can be compared to the measurement of C_A and C_F in Section 3.1.4. The combined experimental measurement [24, 30] displayed relative errors of $\sim 7\%$ for both quantities, dominated by systematic errors and with high correlations between the two. The results presented in Figure 6.8 show significantly smaller errors than

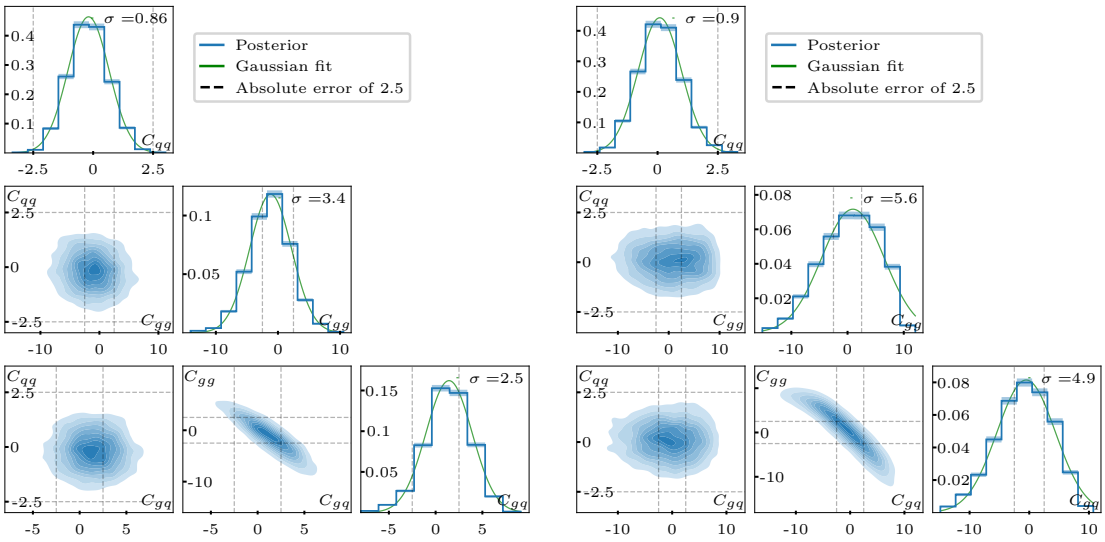

Figure 6.9: Posterior probabilities of the p_T-suppressed rest terms , $\{C_{qq}, C_{gg}, C_{gq}\}$, for the full toy shower. We measure $M_{\text{eval}} = 10^4$ SM-like jets and consider truth-sorting (left) and k_T-sorting (right).

the previous LEP measurements and no correlations for both, the truth- and the k_T-sorting. This is very promising and makes us optimistic for the runs including hadronization and detector effects.

Rest terms

The most interesting measurement for improvements on the splitting kernels , if the form of the leading order terms is true, is the measurement of the Rest terms

$$\{C_{qq}, C_{gg}, C_{gq}\} \in [-10, 10] \times [-15, 15] \times [-15, 15]\,. \tag{6.3}$$

Again, we employ the same architecture and get the results shown in Figure 6.9. Compared to the gluon-radiation shower, the posterior of C_{qq} is slightly more narrow, because the network does not have to disentangle effects of different magnitude. Again, due to the same logic, we obtain significantly better results for the splittings where the mother parton is a quark, that is C_{qq}, as for splittings of a gluon, C_{gg} and C_{gq}. The high (anti-)correlation between both gluon splittings then leads to the high uncertainty. If we estimate the variance of one parameter with the others at 0, the error on C_{gg} and C_{gq} is only slightly higher than the one on C_{qq} (see Table 8.1).

Going from truth- to k_T-sorting, the anti-correlation increases and the uncertainty in the gluon-parameters grows accordingly. There is no reason why the correlation should decrease if we include hadronization and detector effects.

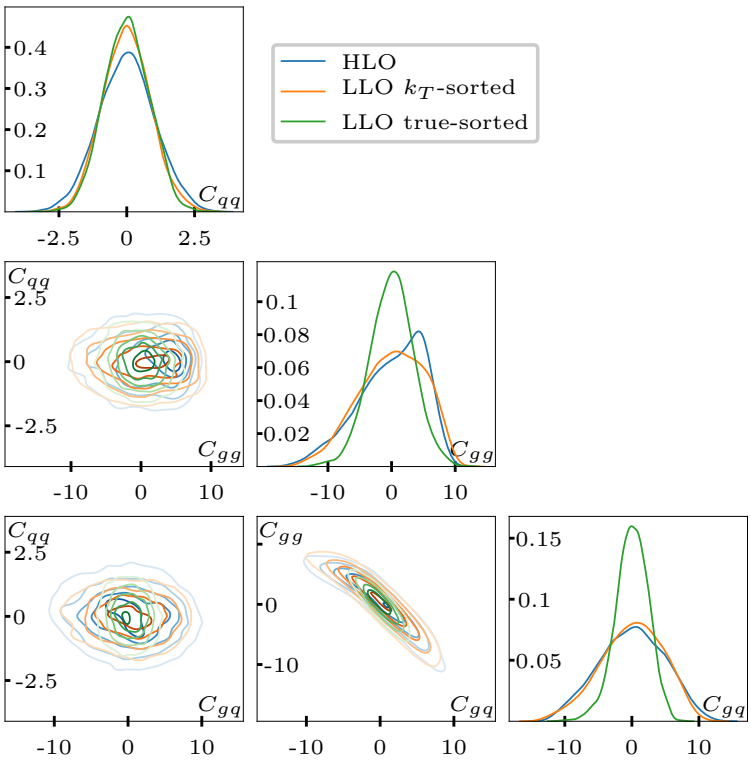

Figure 6.10: Comparing the posterior probabilities of the p_T-suppressed terms, $\{C_{qq}, C_{gg}, C_{gq}\}$, for training on low-level observables with both sortings (compare Figure 6.9) and high-level jet observables from the toy shower data.

Comparison to high-level observables

As for the pure gluon-radiation shower, we want to determine the effect of going from jet observables to jet momenta. No significant improvements were found for the leading terms. The uncertainty on the standard deviations between different training runs is exceeds the slight decrease of the standard deviations we detected in Table 8.1.

For the rest terms, the situation is different, as can be seen in Figure 6.10. Again, the performance of the truth-sorting is unmatched, although using k_T-sorting decreases correlations and prevents non-Gaussian structures compared to the training on high-level observables. As discussed in Section 6.1, this motivates developing a better sorting algorithm.

When discussing Figure 6.10, we have to keep in mind that it only shows one single training run. We have found similar behaviour for all runs, however different runs can have varying performance, as the convergence is not totally stable. It might be possible to find better trainings, with less non-Gaussianities.

7 Inference including hadronization and detector

The next step is to include hadronization and detector effects into our simulation. We use SHERPA [5] to generate

$$e^+e^- \to q\bar{q}$$

at m_Z, including parton showers and hadronization. For the results including detector effects, we also apply a simulation of the ATLAS detector. We then use an anti-k_T algorithm with cutoff at 20 GeV to extract a single jet. Without the detector, we train on k_T-sorted 4-momenta of hadrons, photons and charged leptons. They correspond to the particle flow objects we consider when including detector simulations. We presented a more detailed discussion of the simulation chain in Section 3.3.4.

We choose the same setup for our model, including the intervals of the uniform prior distributions, as in the toy example. However, we found that due to the increased complexity of the problem, a slightly slower learning rate decay gives better results (see Table 5.1).

Hierarchical terms of P_{qq}

In analogy to the examination of the toy setup, we start with varying the parameters of the the quark-gluon splitting kernel P_{qq}. A big difference to the toy setup is, that we do not exclude the other two splittings.

As explained in Section 3.4 with regard to the distributions of high-level jet observables for the different setups, we expect a significant difference when including hadronization, but only a small influence of the detector effects. This is confirmed by the results in presented in Figure 7.1 and Table 8.1. The uncertainties on the leading term and on the rest term increase by a factor of two or more, while the uncertainty of the finite term does not increase significantly. As expected, the detector simulation only slightly deteriorates results.

Leading terms

Comparing Figures 7.2 to Figures 6.8, we find the same behaviour. Hadronization deteriorates the results by a factor of four in D_{qq} and up to a factor ten in D_{gg}. The detector simulation then only diminishes the inference quality slightly. Table 8.1 shows that about half of this is due to the high correlation of both parameters after hadronization.

Again, we assume that the initial particle of the shower being a quark leads to the high correlation and including a gluon shower in the sample could pose a simple resolution to this problem. However, even if we account for the effect of the correlations, the uncertainty on D_{gg} still surpasses the uncertainty of LEP measurement by a factor of four.

The results for D_{gg} in this section have to be taken with a grain of salt. For a normal distribution centered at 1 with a standard deviation of 0.5, parts of the distribution are already outside of the lower threshold of the prior. Therefore, the boundary of the prior actually leads to an underestimation of the uncertainty. As negative values for D_{gg} can not be justified theoretically, increasing the prior cannot fix this. When including a gluon-shower to remove the correlation or when considering bigger statistics, this is expected not to be an issue.

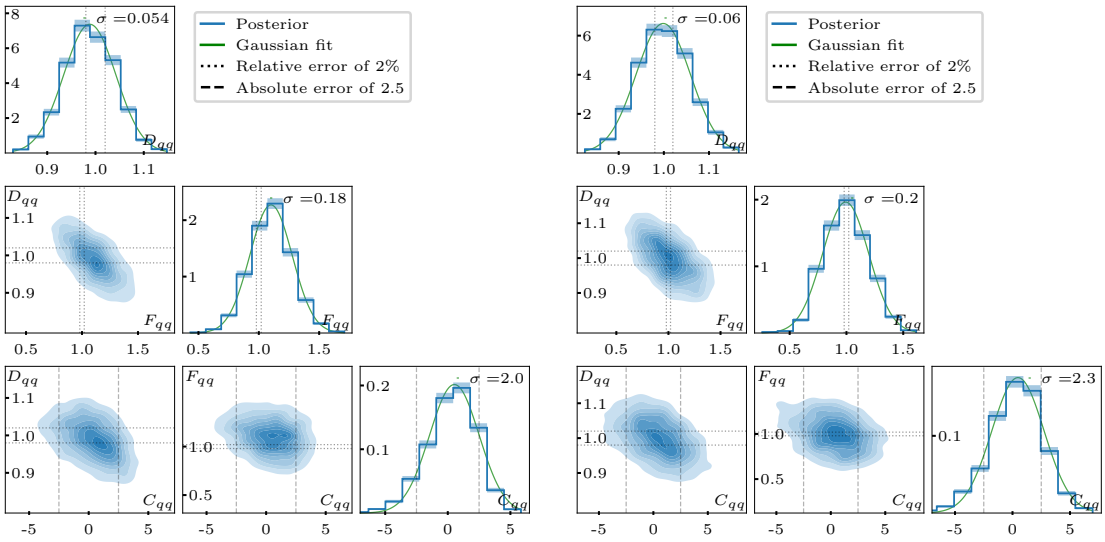

Figure 7.1: Posterior probabilities of the parameters of the quark-gluon splitting kernel , $\{D_{qq}, F_{qq}, C_{qq}\}$, for the SHERPA shower. We measure 10^4 SM-like jets and consider the shower immediately after hadronization (left) and including detector simulations (right).

Rest terms

The results form the toy shower, especially the strong correlation between C_{gg} and C_{gq}, already hinted at the problems occurring when including hadronization. After hadronization the effect of both rest terms is so small, the network can not distinguish it from statistical noise. Thus the network returns the prior, a uniform distribution between -15 and 15.

Although the rest terms of the gluon splittings cannot be resolved, the result for the quark splitting C_{qq} is more positive. The rest term can clearly be inferred and the uncertainty only drop by 50% compared to the toy setup. Therefore, it is conceivable that similar results for C_{gg} and C_{gq} are in reach, when including a gluon shower.

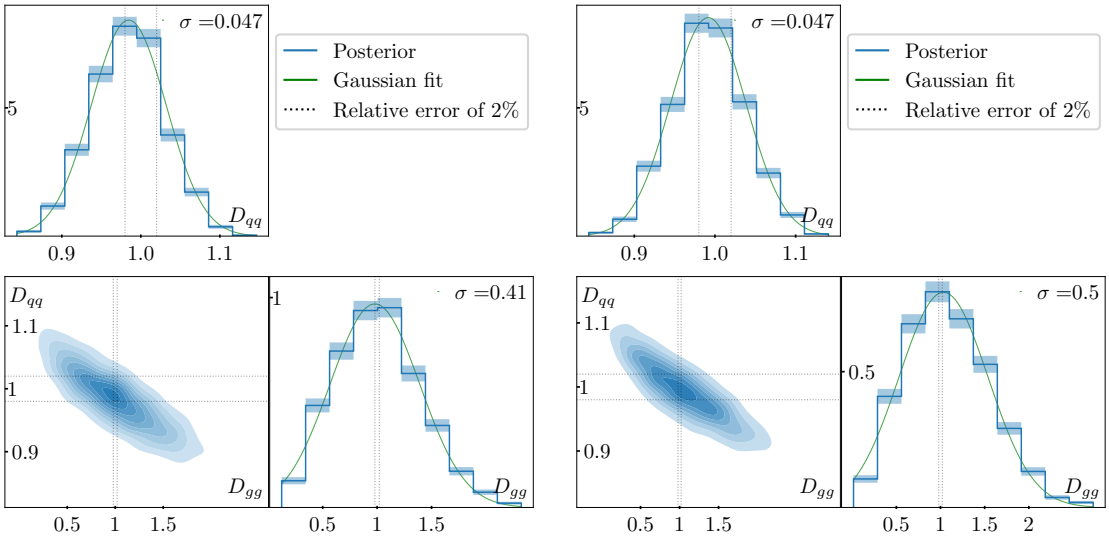

Figure 7.2: Posterior probabilities of the soft-collinear leading terms , $\{D_{qq}, D_{gg}\}$, for the SHERPA shower. We measure $M_{\text{eval}} = 10^4$ SM-like jets and consider the shower immediately after hadronization (left) and including detector simulations (right).

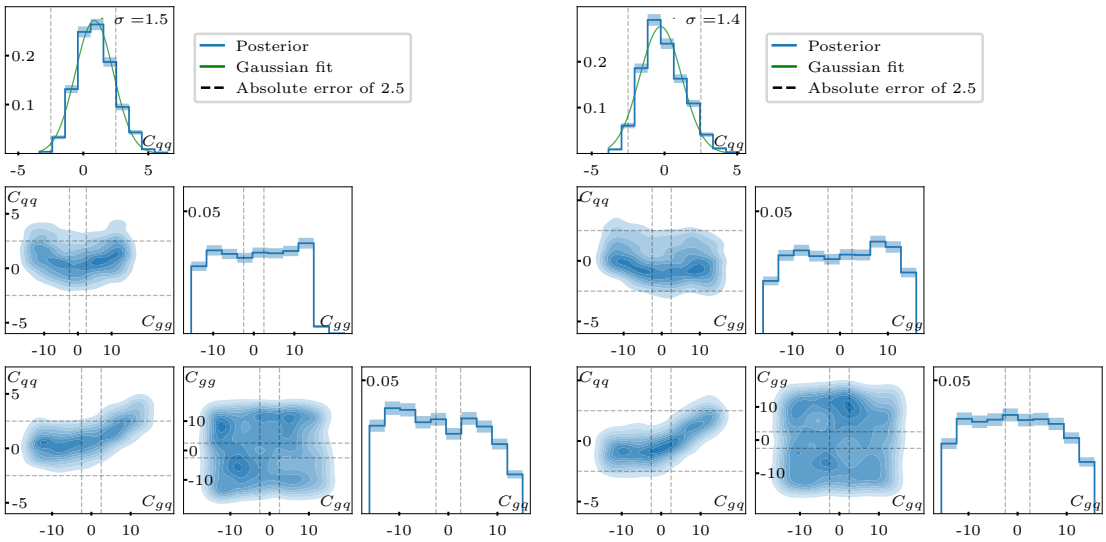

Figure 7.3: Posterior probabilities of the p_T-suppressed rest terms , $\{C_{qq}, C_{gg}, C_{gq}\}$, for the SHERPA shower. We measure $M_{\text{eval}} = 10^4$ SM-like jets and consider the shower immediately after hadronization (left) and including detector simulations (right).

8 Conclusions

In this work we estimated the usability of conditional invertible neural networks for inference of QCD parton shower contributions.

We started at the description of the Altarelli-Parisi splitting kernels in terms of Catani-Seymour dipoles, where we introduced factors for the leading divergent terms and the finite terms. We also added p_T-suppressed rest terms, which vanish in the collinear limit, where the derivation of the splitting kernels is exact.

These parameterized splitting kernels were then implemented in a simple toy setup, only including a hard scattering process and a parton shower. In a first step, we limited the splittings to the radiation of a soft gluon of a hard quark and found that the network was able to recover all three parameters, with minor correlation only between the finite and the rest term. The uncertainty in the individual measurements reflected the hierarchy in the terms. In this setting we were also able to show that the uncertainty of the measurement scales like a statistically limited measurement for two orders of magnitude. Including all QCD splittings in the parton shower generator, we were able to recover both divergent and all three rest terms at the same time.

Comparing results from training on jet 4-momenta and on jet observables, we found that the low-level data indeed contained more information on the QCD splittings.

Including hadronization we found similar results with and without detector effects. We were able to recover the hierarchical structure of the terms in the P_{qq} kernel. For the measurement of the leading terms including hadronization, we found a strong correlation between the leading term of the quark-gluon and the triple gluon splitting. After hadronization the rest terms of the gluon splittings could not be resolved anymore.

A necessary next step would be to include a parton shower originating from a gluon to avoid the correlations of the leading terms and improve the measurements on the splittings originating from gluons. As the results for the P_{qq} kernel were very promising and showed smaller errors than the LEP measurement of C_F even for our small amount oft jets in the measurement, we are very optimistic the method can prove useful in a LHC setting.

Our setup, using electron-positron scattering as a hard scattering, does not exhibit some of the key difficulties of hadron colliders such as intial state radiation, pile-up or additional interactions of the partons. Due to its low energy the effects of the thresholds of the detector are small as well. These difficulties have to be overcome when using BayesFlow for LHC analysis. However, a larger number of measurements, as well as harder jets including more splittings and thus more information

Setup & Parameter		Toy shower						SHERPA			
		Truth-sorted		k_T-sorted		HLO		Hadronized		Detector-level	
$\{D_{qq}, F_{qq}, C_{qq}\}$	$\sigma(D_{qq})$	0.012	(0.013)	0.019	(0.013)	0.024	(0.015)	0.054	(0.025)	0.060	(0.03)
	$\sigma(F_{qq})$	0.05	(0.04)	0.16	(0.07)	0.19	(0.08)	0.18	(0.09)	0.20	(0.1)
	$\sigma(C_{qq})$	0.97	(0.8)	1.04	(0.8)	1.7	(1.0)	2.0	(1.2)	2.3	(1.4)
$\{D_{qq}, D_{gg}\}$	$\sigma(D_{qq})$	0.013	(0.013)	0.013	(0.013)	0.013	(0.013)	0.047	(0.025)	0.047	(0.025)
	$\sigma(D_{gg})$	0.034	(0.034)	0.033	(0.033)	0.035	(0.035)	0.41	(0.23)	0.50	(0.25)
$\{C_{qq}, C_{gg}, C_{gq}\}$	$\sigma(C_{qq})$	0.86	(0.8)	0.90	(0.8)	1.0	(1.0)	1.5	(1.0)	1.4	(0.9)
	$\sigma(C_{gg})$	3.4	(1.4)	5.6	(1.7)	5.4∗	(1.7)	∗	∗	∗	∗
	$\sigma(C_{gq})$	2.7	(1.1)	4.9	(1.4)	5.2∗	(1.4)	∗	∗	∗	∗

Table 8.1: Error on the splitting kernel parameters ($M_{\text{eval}} = 10^4$) obtained as the standard deviation of a Gaussian fit on the projected posterior distribution and, in brackets, assuming only one variable splitting parameter at a time. We consider the different setups: gluon radiation only, soft-collinear leading contributions, and p_T-suppressed rest terms. The asterisk denotes non-Gaussian posteriors.

on the kernels, are expected to have a positive effect.

By design, an inference method using a normalizing flow to learn a posterior distribution is limited to the analysis of measurements not bigger then the sets trained on. The method does not scale well with additional, new data and Bayesian updating can not be incorporated without retraining the network. One could therefore say, that it is not amortized well. It will thus be computationally expensive to train a network to work with millions of jets. However, as the posteriors are normally distributed, breaking down the measurements into many small measurements and combining them might be a viable option.

A Explicit calculations on splitting kernels

A.1 Calculation of $\hat{P}_{qq}$

To confirm that the matrix element factorizes at least in the collinear limit in the definition of the splitting kernel in (3.6), we will explicitly calculate the matrix element of an additional $q(p_a) \to q(p_b) + g(p_c)$ branching. To do this, we first need to express the spinors of the massless quarks $u(p_a)$ and $u(p_b)$ in terms of two-component spinors

$$u_{\pm}(p) = \sqrt{E} \begin{pmatrix} \chi_{\pm} \\ \pm\chi_{\pm} \end{pmatrix} \tag{A.1}$$

with

$$\begin{aligned} \chi_{+} &= \begin{pmatrix} 1 \\ \theta/2 \end{pmatrix} \quad \text{(spin up)} \\ \chi_{-} &= \begin{pmatrix} -\theta/2 \\ 1 \end{pmatrix} \quad \text{(spin down)} \, . \end{aligned} \tag{A.2}$$

This simple form of the spinors is only true for massless quarks in the collinear limit, where the scattering angle θ goes to 0. We use this to calculate the matrix element of the splitting

$$V_{qqg} = -ig_s T^a \bar{u}_{\pm}(p_b) \, \gamma_\mu \epsilon_a^\mu u_{\pm}(p_c) =: -ig_s T^a \epsilon_q^\mu F_{\pm}^{(j)} \, , \tag{A.3}$$

with the γ-matrices in Dirac-representation. Doing so, we only need to do the calculations for the physical gluon polarizations γ^1 and γ^2, as the contributions of the unphysical scalar polarization γ^0 and the longitudinal polarization γ^3 cancel each other. Since the gluon carries away a spin of 1, we also only need to look at quark combinations with a spin flip. For the spin-up case this is

$$\begin{aligned} \frac{F_{+}^{(1)}}{\sqrt{E_b}\sqrt{E_a}} = \frac{\bar{u}_{+}(p_b) \, \gamma^1 u_{+}(p_a)}{\sqrt{E_b}\sqrt{E_a}} &= \left(1, \frac{\theta_b^*}{2}, 1, \frac{\theta_b^*}{2}\right) \begin{pmatrix} & & & 1 \\ & & 1 & \\ & 1 & & \\ 1 & & & \end{pmatrix} \begin{pmatrix} 1 \\ \theta_a^*/2 \\ 1 \\ \theta_a^*/2 \end{pmatrix} \\ &= \left(1, \frac{\theta_b^*}{2}, 1, \frac{\theta_b^*}{2}\right) \begin{pmatrix} \theta_a^*/2 \\ 1 \\ \theta_a^*/2 \\ 1 \end{pmatrix} = \theta_a^* + \theta_b^* \end{aligned} \tag{A.4}$$

$$\begin{aligned}\frac{F_+^{(2)}}{\sqrt{E_b}\sqrt{E_a}} = \frac{\bar{u}_+(p_b)\,\gamma^2 u_+(p_a)}{\sqrt{E_b}\sqrt{E_a}} &= \left(1, \frac{\theta_b^*}{2}, 1, \frac{\theta_b^*}{2}\right) \begin{pmatrix} & & & i \\ & & -i & \\ & -i & & \\ i & & & \end{pmatrix} \begin{pmatrix} 1 \\ \theta_a^*/2 \\ 1 \\ \theta_a^*/2 \end{pmatrix} \\ &= i \left(1, \frac{\theta_b^*}{2}, 1, \frac{\theta_b^*}{2}\right) \begin{pmatrix} -\theta_a^*/2 \\ 1 \\ -\theta_a^*/2 \\ 1 \end{pmatrix} = i\,(\theta_b^* - \theta_a^*)\,. \end{aligned} \tag{A.5}$$

The spin-down case can be calculated analogously

$$\begin{aligned}\frac{F_-^{(1)}}{\sqrt{E_b}\sqrt{E_a}} &= \frac{\bar{u}_-(p_b)\,\gamma^1 u_-(p_a)}{\sqrt{E_b}\sqrt{E_a}} = \theta_a^* + \theta_b^* \\ \frac{F_-^{(2)}}{\sqrt{E_b}\sqrt{E_a}} &= \frac{\bar{u}_-(p_b)\,\gamma^2 u_-(p_a)}{\sqrt{E_b}\sqrt{E_a}} = i\,(\theta_a^* - \theta_b^*)\,. \end{aligned} \tag{A.6}$$

The angles in the spinors are given relative to the gluon direction p_c. They are related to the total opening angle θ as

$$\theta_b^* = \theta \quad \text{and} \quad \theta_a^* = -\theta_c = -z\theta \tag{A.7}$$

$$\implies \quad \theta_a^* + \theta_b^* = \theta(1-z) \text{ and } \theta_a^* - \theta_b^* = \theta(-z-1)\,. \tag{A.8}$$

To calculate the $(n+1)$-particle matrix element from the n-particle one according to (3.6), we need to average over all spin combinations in (A.3), colors configurations and gluon polarizations. We then need to square the absolute value of the vertex. To make our life a little easier we observe, that $u_+\bar{u}_- = u_-\bar{u}_+ = 0$ for spinors with the same momentum and that

$$F_+^{(i)} \left(F_+^{(j)}\right)^* = -F_-^{(i)} \left(F_-^{(j)}\right)^*, \tag{A.9}$$

for $i \neq j$. Thus, when taking the absolute value squared of the vertex, the terms mixing u_+ and u_- vanish and the terms mixing γ^1 and γ^2 cancel each other respectively. We are therefore only left with the squares of (A.4) - (A.6). They compute to

$$\left|F_+^{(1)}\right|^2 = \left|F_+^{(1)}\right|^2 = E_a E_b\,(\theta_a^* + \theta_b^*)^2 = E_a^2 z(z-1)^2\theta^2 \tag{A.10}$$

and

$$\left|F_+^{(2)}\right|^2 = \left|F_+^{(2)}\right|^2 = E_a E_b\,(\theta_a^* - \theta_b^*)^2 = E_a^2 z(z+1)^2\theta^2. \tag{A.11}$$

As a last step, we have to include a internal quark propagator of the form $u\bar{u}/p_a^2$, because the incoming quark now is an internal particle of the overall diagram. We

need to express the factor p_a^2 in terms of E_a before putting everything together. From four momentum conservation and on-shell conditions for the outgoing particles we know

$$\begin{aligned} p_a^2 &= (p_b + p_c)^2 = 2p_b p_c = 2(E_b E_c - |\vec{p}_b||\vec{p}_c|\cos\theta) = 2E_b E_c(1-\cos\theta) \\ &= 2(zE_a)((1-z)E_a)(1-\cos\theta) = 2E_a^2(z(1-z))(\theta^2 + \mathcal{O}\left(\theta^4\right)). \end{aligned} \tag{A.12}$$

We can now calculate the $(n+1)$-particle matrix element by tracing out T^a and averaging over the color and spin configurations of the intermediate quark p_a.

- Since there are two possible spin states the latter leaves us with a factor of $N_a = 2$ and from the color average we get a factor N_c, that is the number of color charges $N_c = 3$.
- We know $\operatorname{Tr} T^2 = 1$ for every generator of $SU(n)$ and the dimension of the group is $n^2 - 1$. Therefore, we get $\operatorname{Tr} T^a T^a = N_c^2 - 1$.

Including all these factors, the calculation yields

$$\begin{aligned} \overline{|\mathcal{M}_{n+1}|^2} &= \left(\frac{1}{p_a^2}\right)^2 \frac{1}{N_c}\frac{1}{N_a}\left[-ig_s T^a \epsilon_a^j F_\pm^{(j)}\right]^2 \overline{|\mathcal{M}_n|^2} \\ &= \left(\frac{g_s}{p_a}\right)^2 \frac{1}{N_c}\frac{1}{N_a}\operatorname{Tr} T^a T^a \left[\epsilon_a^j F_\pm^{(j)}\right]^2 \overline{|\mathcal{M}_n|^2} \\ &= \frac{g_s^2}{p_a^2}\frac{1}{N_c}\frac{1}{N_a}\operatorname{Tr} T^a T^a \frac{2\left[E_a^2 z(z-1)^2\theta^2 + E_a^2 z(z+1)^2\theta^2\right]}{p_a^2}\overline{|\mathcal{M}_n|^2} \\ &= \frac{g_s^2}{p_a^2}\frac{N_c^2-1}{2N_c}\frac{2\left[E_a^2 z(z-1)^2\theta^2 + E_a^2 z(z+1)^2\theta^2\right]}{2E_a^2(z(1-z))}\overline{|\mathcal{M}_n|^2} \\ &= \frac{g_s^2}{p_a^2}\frac{N_c^2-1}{2N_c}\frac{(1+z)^2 + (1-z)^2}{1-z}\overline{|\mathcal{M}_n|^2} \\ &= \frac{2g_s^2}{p_a^2}\frac{N_c^2-1}{2N_c}\frac{1+z^2}{1-z}\overline{|\mathcal{M}_n|^2} \end{aligned} \tag{A.13}$$

With (3.6) we identify the splitting kernel as

$$\boxed{\hat{P}_{qq}(z) = C_F\frac{1+z^2}{1-z} \text{ with } C_F = (N_c^2-1)/2N_c = \frac{4}{3}.} \tag{A.14}$$

A.2 +-Regularization

We have to integrate over z, if we want to calculate the total cross section using (3.8). This is important to us, as we have to evaluate the probability that such a splitting happened during the simulation of a parton shower (see Section 3.2.2).

Because of the divergence of $\hat{P}_{qq}$ and $\hat{P}_{gg}$ for soft outgoing particles $z \to 1(,0)$, we need to regularize this integral. In practise this is often done by a simple cutoff, although using dimensional regularisation is analytically more appealing

$$\begin{aligned}
\int_0^1 dz \frac{f(z)}{(1-z)^{1-\epsilon}} &= \int_0^1 dz \frac{f(z)-f(1)}{(1-z)^{1-\epsilon}} + \int_0^1 dz f(1) \frac{1}{(1-z)^{1-\epsilon}} \\
&= \int_0^1 dz \frac{f(z)-f(1)}{(1-z)} (1+\mathcal{O}(\epsilon)) + \frac{f(1)}{\epsilon} \\
&= \int_0^1 dz \frac{f(z)}{(1-z)_+} (1+\mathcal{O}(\epsilon)) + \frac{f(1)}{\epsilon} \\
\Rightarrow \int_0^1 dz \frac{f(z)}{(1-z)^{1-\epsilon}} - \frac{f(1)}{\epsilon} &= \int_0^1 dz \frac{f(z)}{(1-z)_+} (1+\mathcal{O}(\epsilon)).
\end{aligned} \tag{A.15}$$

The implicitly defined plus subtraction scheme is

$$\begin{aligned}
& F(z)_+ \equiv F(z) - \delta(1-z) \int_0^1 dy F(y) \\
\iff \quad & \int_0^1 dz \frac{f(z)}{(1-z)_+} = \int_0^1 dz \left(\frac{f(z)}{1-z} - \frac{f(1)}{1-z} \right).
\end{aligned} \tag{A.16}$$

It regularizes the splitting kernel as the plus subtracted integral (with terms $\mathcal{O}(\epsilon)$), is the dimensionally regularized integral minus the pole.

Using this subtraction scheme on the splitting kernel $\hat{P}_{qq}$ from (3.13), we can calculate the regularized splitting kernel P_{qq}

$$\begin{aligned}
& \left(\frac{1+z^2}{1-z} \right)_+ - (1+z^2) \left(\frac{1}{1-z} \right)_+ \\
&= \frac{1+z^2}{1-z} - \delta(1-z) \int_0^1 dy \frac{1+y^2}{1-y} - \frac{1+z^2}{1-z} + \delta(1-z) \int_0^1 dy \frac{1+z^2}{1-y} \\
&= \delta(1-z) \int_0^1 dy \frac{y^2-1}{y-1} \\
&= \delta(1-z) \int_0^1 dy (y+1) = \frac{3}{2} \delta(1-z)
\end{aligned} \tag{A.17}$$

$$\Longrightarrow \boxed{P_{qq}(z) = C_F \left(\frac{1+z^2}{1-z} \right)_+ = C_F \left[\frac{1+z^2}{(1-z)_+} + \frac{3}{2} \delta(1-z) \right].} \tag{A.18}$$

In the calculation of parton density evolutions vie the DGLAP-equation terms appear that can be used to naturally +-regularize the splitting kernels. However, the scheme is rather unimportant for final-state parton-shower generators.

A.3 Splitting kernel overestimates

To use the veto algorithm in our parton shower generator as described in Section 3.2.2, we need overestimates of our splitting kernels (3.34) - (3.36). As explained before we choose them to be independent of p_T^2 and therefore independent of y. We use

$$\tilde{P}_{qq}(z) = 2C_F \left(D_{qq} + \left(\frac{F_{qq}}{2}\right)^+ + \left(\frac{C_{qq}}{8}\right)^+ \right) \frac{1}{1-z}\,, \tag{A.19}$$

$$\tilde{P}_{gg}(z) = 2C_A \left(D_{gg} + \left(\frac{F_{gg}}{8}\right)^+ + \left(\frac{C_{gg}}{8}\right)^+ \right) \left(\frac{1}{1-z} + \frac{1}{1-(1-z)} \right), \tag{A.20}$$

$$\tilde{P}_{gq}(z) = T_R \left((F_{qq})^+ + \left(\frac{C_{gq}}{4}\right)^+ \right). \tag{A.21}$$

As explained in Section 3.1.3 we cut off the kernels at 0 with $(x)^+ = \max\{x, 0\}$.

The algorithm uses the integral over z

$$I_{ij}(z_1, z_2) = \int_{z_1}^{z_2} \tilde{P}_{ij}(z) \mathrm{d}z \tag{A.22}$$

to generate new values for p_T^2 and to sample values of z The integrals of the overestimates are

$$I_{qq}(z_1, z_2) = 2C_F \left(D_{qq} + \left(\frac{F_{qq}}{2}\right)^+ + \left(\frac{C_{qq}}{8}\right)^+ \right) \log\left(\frac{1-z_1}{1-z_2}\right), \tag{A.23}$$

$$I_{gg}(z_1, z_2) = 2C_A \left(D_{gg} + \left(\frac{F_{gg}}{8}\right)^+ + \left(\frac{C_{gg}}{8}\right)^+ \right) \log\left(\frac{(1-z_1)z_2}{(1-z_2)z_1}\right), \tag{A.24}$$

$$I_{gq}(z_1, z_2) = T_R \left((F_{qq})^+ + \left(\frac{C_{gq}}{4}\right)^+ \right) (z_2 - z_1). \tag{A.25}$$

For the parton shower algorithm, we also need to sample z according to the overestimate. If we define

$$u(z) = \frac{I_{ij}(z_{\min}, z)}{I_{ij}(z_{\min}, z_{\max})}\,. \tag{A.26}$$

we can draw a $z \in [z_{\min}, z_{\max}]$ from the probability overestimate by drawing a random number $u \in [0, 1]$ from a uniform distribution and then calculating $z = z(u)$ using the inverse of the function $u(z)$.

From (A.23)-(A.25) we can calculate the sampling functions as

$$z_{qq}(u) = 1 + (z_{max} - 1)\left(\frac{1 - z_{\min}}{1 - z_{\max}}\right)^{u} \tag{A.27}$$

$$z_{gg}(u) = \frac{e^{\alpha u}}{e^{\alpha u} + \beta} \tag{A.28}$$

$$z_{gq}(u) = z_{\min} + u(z_{\max} - z_{\min})\,, \tag{A.29}$$

where we introduced the abbreviations

$$\alpha = \log\left(\frac{(1 - z_{\min})z_{\max}}{(1 - z_{\max})z_{\min}}\right) \tag{A.30}$$

$$\beta = \frac{1 - z_{\min}}{z_{\min}}\,. \tag{A.31}$$

B Data handling

Within this work we use two different approaches to the training data.

Online learning

As mentioned before, reducing the possible splittings to a gluon radiating of a quark, allows us to parallelize multiple showers. Running the shower on the parallelized code on the GPU during training is quick enough to calculate the data when needed without slowing down the training. This *online training* has the advantage that overfitting is impossible by design. Furthermore, changes to the theory can be included without constructing a new dataset.

Loading training data in big batches

When including more splittings or working with SHERPA data, the simulation is to slow to be done during the training. The datasets then contain 10^6 parameter points with 10^5 showers each. For each shower we save 52 momentum entries and 6 high-level observables in a HDF5 file with medium compression. Still the file easily is bigger ($\sim 150\,\text{GB}$) than our available working memory. Loading each batch from the the hard drive produces to much I/O between the machine the data is stored on and the one doing the training and jams the computer cluster for other users. To solve this problem we came up with a compromise.

- We reduce I/O by loading a multiple the size of one batches at once, colloquially referred to as *super-batches*.
- We then train on all batches in one super-batch for multiple iterations, shuffling the super-batch after every full pass. After a fixed amount of iterations, jokingly referred to as *quasi-epochs*, we load another super-batch.

This data loading routine introduces minor instabilities for the training. If the number of quasi-epochs before loading is two big compared to the size of a super-batch, the training is more prone to converge into a local minimum. This is most important at the beginning of the training.

We have found that using a super-batch size of 100 batches, each containing 16 datapoints, and reloading every 25 quasi-epochs leads to the most stable results. The loss still converges to the same value as in the classical training.

C Performance on the whole prior

In the main part of this work, we focus on the posterior distributions at one point in parameter space. After all, we assume the parameter values to be close to the SM values. However, the network should still perform well all over the prior.

To assert this, we calculate the mean of 1000 point clouds (2000 points each) sampled for measurements ($M_{\text{eval}} = 10^4$) generated with arbitrary parameter combinations.

Figures C.1 - C.3 show the estimated values track the true values without bias and the spread corresponds well to the standard deviations from Table 8.1. Only for parameters inferred with low certainty, effects of the boundary can be seen for very high or low values.

For the inference of the rest terms including hadronization and detector effects, the network is not able to estimate the parameters of the gluon splittings (see Figure 7.3). Therefore, the network correctly returns the prior for every parameter combination. As the prior is a uniform distribution symmetric around zero, the mean vanishes for all true parameters.

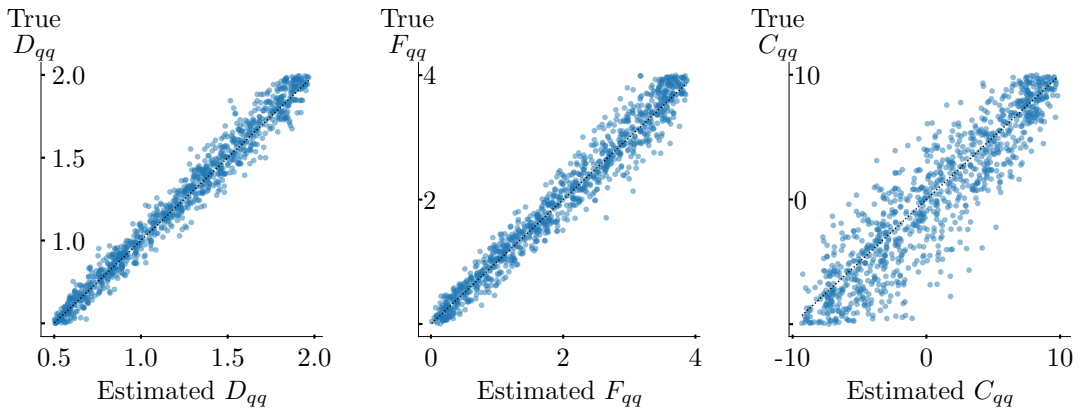

Figure C.1: True values over posterior means for the parameters of the gluon-radiation shower, D_{qq} (left), F_{qq} (middle) and C_{qq} (right), for parameter combinations drawn from the prior ($M_{\text{eval}} = 10^4$). We show the results for the k_T-sorted SHERPA shower including detector effects.

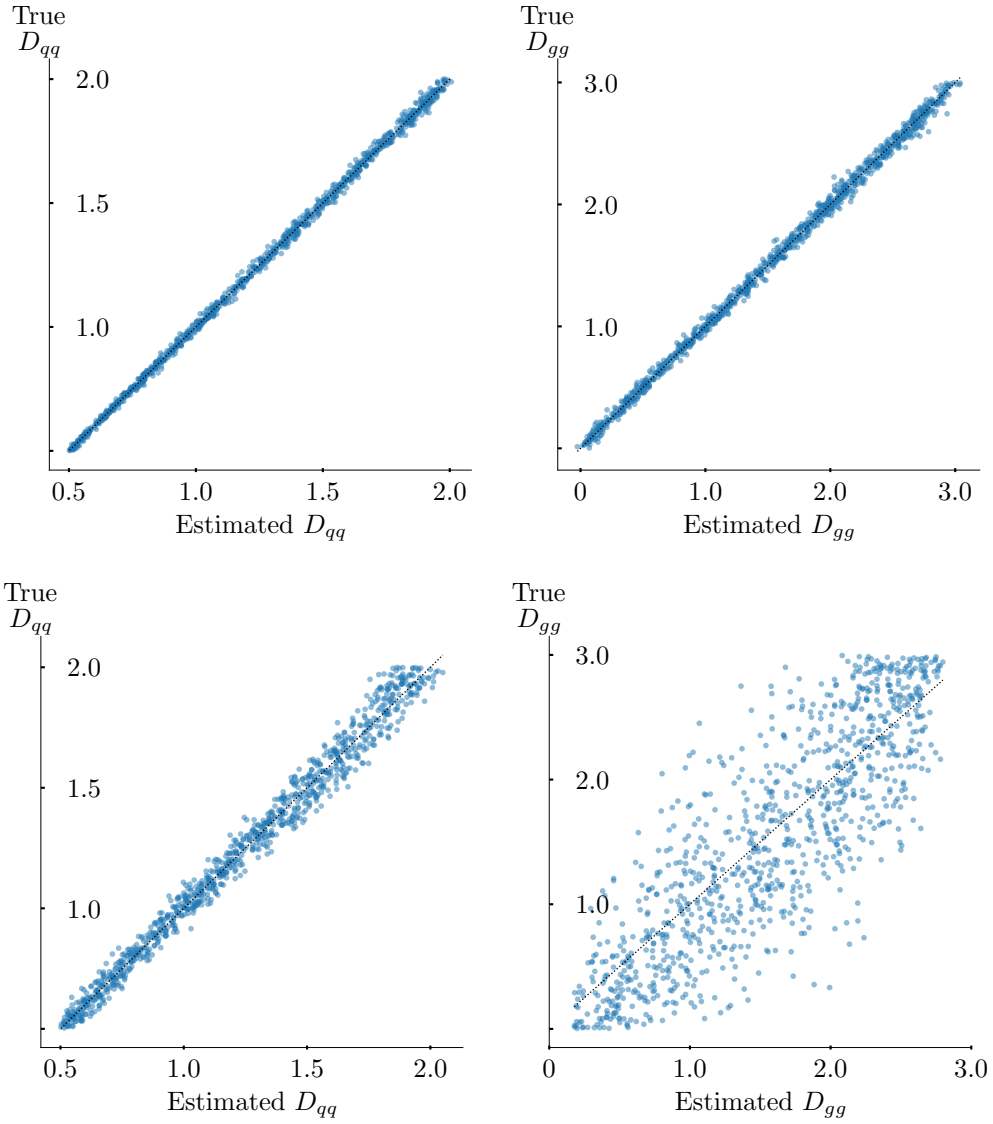

Figure C.2: True values over posterior means for the soft-collinear leading terms, D_{qq} (left), D_{gg} (right), for parameter combinations drawn from the prior ($M_{\text{eval}} = 10^4$). We show the results for the k_T-sorted toy shower (upper) and for the k_T-sorted SHERPA shower including detector effects (lower).

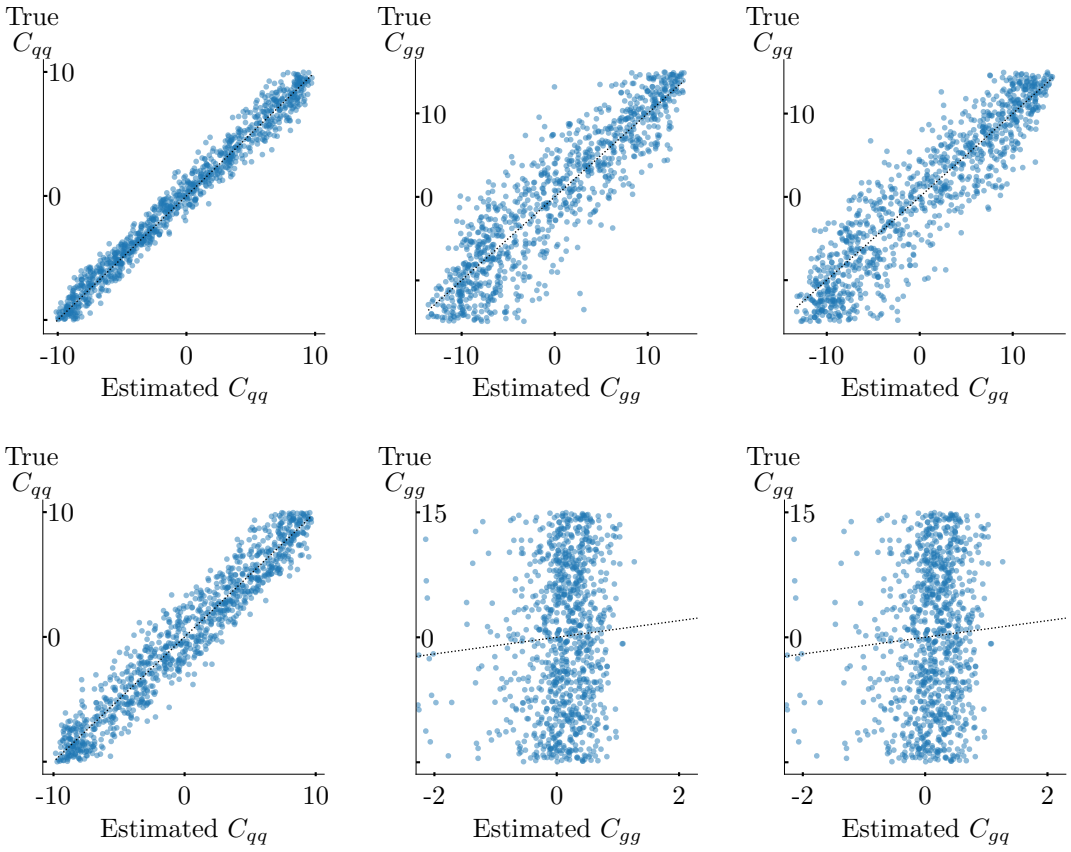

Figure C.3: True values over posterior means for the p_T-suppressed rest terms, C_{qq} (left), C_{gg} (middle) and C_{gq} (right), for parameter combinations drawn from the prior ($M_{\text{eval}} = 10^4$). We show the results for the k_T-sorted toy shower (upper) and for the k_T-sorted SHERPA shower including detector effects (lower).

D Bibliography

[1] Michael Kagan. *Image-Based Jet Analysis*. 2020. arXiv: 2012.09719.

[2] Anja Butter and Tilman Plehn. *Generative Networks for LHC events*. 2020. arXiv: 2008.08558 [hep-ph].

[3] Christina Winkler et al. *Learning Likelihoods with Conditional Normalizing Flows*. 2019. arXiv: 1912.00042 [cs.LG].

[4] Lynton Ardizzone et al. "Guided Image Generation with Conditional Invertible Neural Networks". In: *arXiv e-prints* (July 2019), arXiv:1907.02392. arXiv: 1907.02392 [cs.CV].

[5] Enrico Bothmann et al. "Event generation with Sherpa 2.2". In: *SciPost Physics* 7.3 (Sept. 2019). ISSN: 2542-4653. DOI: 10.21468/scipostphys.7.3.034.

[6] Sebastian Bieringer et al. *Measuring QCD Splittings with Invertible Networks*. 2020. arXiv: 2012.09873 [hep-ph].

[7] Sheldon L. Glashow. "Partial-symmetries of weak interactions". In: *Nuclear Physics* (1961). ISSN: 00295582.

[8] Steven Weinberg. "A model of leptons". In: *Physical Review Letters* (1967). ISSN: 00319007. DOI: 10.1103/PhysRevLett.19.1264.

[9] Abdus Salam and J. C. Ward. "Weak and electromagnetic interactions". In: *Il Nuovo Cimento Series 10* (1959). ISSN: 00296341. DOI: 10.1007/BF02726525.

[10] Michael E Peskin and Daniel V Schroeder. *An introduction to quantum field theory*. Boulder, CO: Westview, 1995.

[11] Steven Weinberg. *The Quantum Theory of Fields*. Vol. 3. Cambridge University Press, 2000. DOI: 10.1017/CBO9781139644198.

[12] Mark Thomson. *Modern Particle Physics*. Cambridge University Press, 2013. DOI: 10.1017/CBO9781139525367.

[13] C. S. Wu et al. *Experimental test of parity conservation in beta decay [5]*. 1957. DOI: 10.1103/PhysRev.105.1413.

[14] R. Bayes et al. "Experimental Constraints on Left-Right Symmetric Models from Muon Decay". In: *Phys. Rev. Lett.* 106 (4 Jan. 2011), p. 041804. DOI: 10.1103/PhysRevLett.106.041804.

[15] S. Schael et al. "Precision electroweak measurements on the Z resonance". In: *Phys. Rept.* 427 (2006), pp. 257–454. DOI: 10.1016/j.physrep.2005.12.006.

[16] Stefan Höche. "Introduction to Parton-Shower Event Generators - Tutorial for summer schools".

[17] Tilman Plehn. "Lectures on LHC physics". In: *Lecture Notes in Physics* 886 (2015), pp. 1–340. ISSN: 00758450. arXiv: 0910.4182.

[18] John Campbell, Joey Huston, and Frank Krauss. *The Black Book of Quantum Chromodynamics*. Vol. 1. 2018, pp. 1–11. ISBN: 9780199652747. DOI: 10.1093/oso/9780199652747.001.0001.

[19] G. Altarelli and G. Parisi. "Asymptotic freedom in parton language". In: *Nuclear Physics, Section B* 126.2 (1977), pp. 298–318. ISSN: 05503213.

[20] S. Catani and M. H. Seymour. "A general algorithm for calculating jet cross sections in NLO QCD". In: *Nuclear Physics B* 485.1-2 (1997), pp. 291–419. ISSN: 05503213. arXiv: 9605323.

[21] Toichiro Kinoshita. "Mass Singularities of Feynman Amplitudes". In: *Journal of Mathematical Physics* 3.4 (1962), pp. 650–677. DOI: 10.1063/1.1724268.

[22] T. D. Lee and M. Nauenberg. "Degenerate Systems and Mass Singularities". In: *Phys. Rev.* 133 (6B Mar. 1964), B1549–B1562. DOI: 10.1103/PhysRev.133.B1549.

[23] S. Catani, B.R. Webber, and G. Marchesini. "QCD coherent branching and semi-inclusive processes at large χ". In: *Nuclear Physics B* 349.3 (1991), pp. 635–654. ISSN: 0550-3213.

[24] S. Kluth. "Jet physics in e+e annihilation from 14 to 209 GeV". In: *Nuclear Physics B - Proceedings Supplements* 133 (July 2004), pp. 36–46. ISSN: 0920-5632. DOI: 10.1016/j.nuclphysbps.2004.04.134.

[25] G. Abbiendi et al. "Particle multiplicity of unbiased gluon jets from e+e three-jet events". In: *The European Physical Journal C* 23.4 (Apr. 2002), pp. 597–613. ISSN: 1434-6052. DOI: 10.1007/s100520200926.

[26] P. Abreu et al. and The DELPHI Collaboration. "Measurement of the gluon fragmentation function and a comparison of the scaling violation in gluon and quark jets". In: *The European Physical Journal C - Particles and Fields* 13.4 (2000), pp. 573–589. DOI: 10.1007/s100520000313.

[27] A. Heister et al. and The ALEPH Collaboration. "Measurements of the strong coupling constant and the QCD colour factors using four-jet observables from hadronic Z decays". In: *The European Physical Journal C - Particles and Fields* 27.1 (2003), pp. 1–17.

[28] G. Abbiendi et al. "A simultaneous measurement of the QCD colour factors and the strong coupling". In: *The European Physical Journal C* 20.4 (May 2001), pp. 601–615. ISSN: 1434-6052. DOI: 10.1007/s100520100699.

[29] S. Kluth et al. "A measurement of the QCD colour factors using event shape distributions at $\sqrt{s} = 14$ to 189 GeV". In: *The European Physical Journal C* 21.2 (June 2001), pp. 199–210. ISSN: 1434-6052. DOI: 10.1007/s100520100742.

[30] Stefan Kluth. "Tests of quantum chromo dynamics at e+e colliders". In: *Reports on Progress in Physics* 69.6 (May 2006), pp. 1771–1846. ISSN: 1361-6633.

[31] Yuri L Dokshitzer. "Calculation of the structure functions for deep inelastic scattering and e+ e- annihilation by perturbation theory in quantum chromodynamics". In: *Zh. Eksp. Teor. Fiz* 73 (1977), p. 1216.

[32] V.N. Gribov and L.N. Lipatov. In: *Sov.J.Nucl.Phys.* 15 (1972), p. 438.

[33] Stefan Höche. "Introduction to parton-shower event generators". In: *arXiv e-prints* (Nov. 2014), arXiv:1411.4085. arXiv: 1411.4085 [hep-ph].

[34] Torbjörn Sjöstrand, Stephen Mrenna, and Peter Skands. "PYTHIA 6.4 physics and manual". In: *Journal of High Energy Physics* 2006.05 (May 2006), pp. 026–026. ISSN: 1029-8479.

[35] Gösta Gustafson. "Dual description of a confined colour field". In: *Physics Letters B* 175.4 (1986), pp. 453–456.

[36] Steffen Schumann and Frank Krauss. "A Parton shower algorithm based on Catani-Seymour dipole factorisation". In: *JHEP* 03 (2008), p. 038. arXiv: 0709.1027 [hep-ph].

[37] Ya I Azimov et al. "Similarity of parton and hadron spectra in QCD jets". In: *Zeitschrift für Physik C Particles and Fields* 27.1 (1985), pp. 65–72.

[38] J. de Favereau et al. "DELPHES 3: a modular framework for fast simulation of a generic collider experiment". In: *Journal of High Energy Physics* 2014.2 (Feb. 2014). ISSN: 1029-8479. DOI: 10.1007/jhep02(2014)057.

[39] Florian Beaudette. *The CMS Particle Flow Algorithm*. 2014. arXiv: 1401.8155 [hep-ex].

[40] Stephen D. Ellis and Davison E. Soper. "Successive combination jet algorithm for hadron collisions". In: *Phys. Rev. D* 48 (1993), pp. 3160–3166. DOI: 10.1103/PhysRevD.48.3160.

[41] M. Wobisch and T. Wengler. "Hadronization corrections to jet cross-sections in deep inelastic scattering". In: *Workshop on Monte Carlo Generators for HERA Physics (Plenary Starting Meeting)*. Apr. 1998, pp. 270–279.

[42] Matteo Cacciari, Gavin P Salam, and Gregory Soyez. "The anti-ktjet clustering algorithm". In: *Journal of High Energy Physics* 2008.04 (Apr. 2008), pp. 063–063. ISSN: 1029-8479.

[43] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. "FastJet user manual". In: *The European Physical Journal C* 72.3 (Mar. 2012). ISSN: 1434-6052.

[44] Gregor Kasieczka et al. "Quark-gluon tagging: Machine learning vs detector". In: *SciPost Physics* 6.6 (2019), pp. 1–23. ISSN: 2542-4653. DOI: 10.21468/scipostphys.6.6.069. arXiv: 1812.09223.

[45] Christopher Frye et al. "Casimir meets Poisson: improved quark/gluon discrimination with counting observables". In: *Journal of High Energy Physics* 2017.9 (Sept. 2017). ISSN: 1029-8479. DOI: 10.1007/jhep09(2017)083.

[46] Jason Gallicchio et al. "Multivariate discrimination and the Higgs+W/Z search". In: *Journal of High Energy Physics* 2011.4 (Apr. 2011). ISSN: 1029-8479. DOI: 10.1007/jhep04(2011)069.

[47] *Performance of quark/gluon discrimination in 8 TeV pp data*. Tech. rep. CMS-PAS-JME-13-002. Geneva: CERN, 2013.

[48] Andrew J. Larkoski, Gavin P. Salam, and Jesse Thaler. "Energy correlation functions for jet substructure". In: *Journal of High Energy Physics* 2013.6 (June 2013). ISSN: 1029-8479. DOI: 10.1007/jhep06(2013)108.

[49] Jon Pumplin. "How to tell quark jets from gluon jets". In: *Phys. Rev. D* 44 (7 Oct. 1991), pp. 2025–2032. DOI: 10.1103/PhysRevD.44.2025.

[50] Stefan T. Radev et al. "BayesFlow: Learning complex stochastic models with invertible neural networks". In: (2020). arXiv: 2003.06281.

[51] Léon Bottou. "Stochastic Gradient Descent Tricks". In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 421–436. ISBN: 978-3-642-35289-8.

[52] Diederik P. Kingma and Jimmy Lei Ba. "Adam: A method for stochastic optimization". In: *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. 2015. arXiv: 1412.6980.

[53] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Journal of Machine Learning Research*. 2010.

[54] Ivan Kobyzev, Simon Prince, and Marcus Brubaker. "Normalizing Flows: An Introduction and Review of Current Methods". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020). ISSN: 0162-8828. DOI: 10.1109/tpami.2020.2992934. arXiv: 1908.09257.

[55] George Papamakarios et al. *Normalizing Flows for Probabilistic Modeling and Inference*. 2019. arXiv: 1912.02762.

[56] V I Bogachev, A V Kolesnikov, and K V Medvedev. "Triangular transformations of measures". In: *Sbornik: Mathematics* 196.3 (Apr. 2005), pp. 309–335. DOI: 10.1070/sm2005v196n03abeh000882.

[57] Lynton Ardizzone et al. "Analyzing inverse problems with invertible neural networks". In: *7th International Conference on Learning Representations, ICLR 2019* i (2019), pp. 1–20. arXiv: 1808.04730.

[58] Laurent Dinh, David Krueger, and Yoshua Bengio. *NICE: Non-linear Independent Components Estimation*. 2015. arXiv: 1410.8516 [cs.LG].

[59] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. *Density estimation using Real NVP*. 2017. arXiv: 1605.08803 [cs.LG].

[60] Diederik P. Kingma and Prafulla Dhariwal. *Glow: Generative Flow with Invertible 1x1 Convolutions*. 2018. arXiv: 1807.03039 [stat.ML].

[61] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. "The frontier of simulation-based inference". In: (2019), pp. 1–10. arXiv: 1911.01429.

[62] S. Kullback and R. A. Leibler. "On Information and Sufficiency". In: *Annals of Mathematical Statistics* 22.1 (Mar. 1951), pp. 79–86. DOI: 10.1214/aoms/1177729694.

[63] Ashish Vaswani et al. "Attention Is All You Need". In: *arXiv e-prints* (June 2017). arXiv: 1706.03762 [cs.CL].

[64] Gilles Louppe et al. "QCD-aware recursive neural networks for jet physics". In: *Journal of High Energy Physics* (2019). ISSN: 10298479. DOI: 10.1007/JHEP01(2019)057.

[65] Martın Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015.

[66] Sean Talts et al. *Validating bayesian inference algorithms with simulation-based calibration*. 2018. arXiv: 1804.06788.

Acknowledgements

2020 was not not an easy year for students. It was characterized by long stretches of social distancing, home office and online teaching. Loosing motivation and focus was a permanent threat.

As such I first and foremost want to thank Prof. Tilman Plehn, not only for the opportunity to work this project, but also for the continuous supervision, always making sure we are busy.

I also want to thank Theo Heimel for the easy, productive and fun cooperation. Many parts of this project, for example running a parton shower algorithm on a GPU, would not have been possible without his technical abilities. Thank you, for always being patient with a rather computer novice like me. I will miss our daily discussions.

Furthermore, I am grateful for Stefan Radev's advice on machine learning and statistics and for being available for questions always and all the time. My gratitude extends to Anja Butter for her help defining, setting up and working out technical problems of the project. It also extends to Stefan Höche for supplying the initial parton shower code and further assistance with SHERPA, constructing a sensible sorting algorithm and the final parametrization of the splitting kernels. And to Prof. Ulrich Köthe for general advice on the architecture.

Last, but not least, I want to thank Frederick del Pozo and again Theo Heimel for making sure this work is readable and all my friends and family for keeping me sane during the year!

Erklärung:

Ich versichere, dass ich diese Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den 13.01.2021 .....