Department of Physics and Astronomy

University of Heidelberg

Master thesis

in Physics

submitted by

Theo Heimel

born in Frankfurt

2021

Measuring QCD Splittings

with Invertible Neural Networks

This Master thesis has been carried out by Theo Heimel

at the

Institute for Theoretical Physics in Heidelberg

under the supervision of

Prof. Tilman Plehn

Abstract

Parton showers are ubiquitous objects in collider events. In the soft-collinear limit, they are described by QCD splitting functions. In this thesis, we introduce parameters for the divergent, finite and p_T-suppressed rest terms of these splitting functions, the latter accounting for corrections beyond the soft-collinear approximation. We show that conditional invertible neural networks can be used to extract posterior distributions for these parameters from low-level sub-jet observables. We start with jets from a toy shower generator and show that our approach based on low-level observables improves the inference performance compared to established high-level jet observables. Furthermore, we demonstrate the correct inverse square root scaling of the estimated measurement uncertainties with the number of measured jets. Finally, we discuss the effects of hadronization and detectors.

Zusammenfassung

Parton Showers sind allgegenwärtige Objekte in Kollisionen an Teilchenbeschleunigern. Im Grenzfall kleiner Impulsüberträge und kleiner Abstrahlungswinkel werden diese durch QCD-Splittingfunktionen beschrieben. In dieser Arbeit führen wir eine Parametrisierung für die divergenten, endlichen und p_T-unterdrückten Terme dieser Splittingfunktionen ein, wobei letztere Korrekturen jenseits der kollinearen Näherung entsprechen. Wir zeigen, dass invertierbare neuronale Netze genutzt werden können, um a-posteriori-Wahrscheinlichkeiten für diese Parameter aus hochdimensionalen Sub-Jet-Observablen zu bestimmen. Desweiteren demonstrieren wir, dass die Standardabweichungen der gemessenen Parameter das korrekte Skalierungsverhalten mit der inversen Quadratwurzel der Anzahl gemessener Jets zeigen. Abschließend diskutieren wir die Effekte von Hadronisierung und Detektoren.

Contents

1 Introduction

In recent years, machine learning has been tremendously successful inside and outside of physics since large data sets have become common both in commercial and scientific applications and more and more computational resources are available. While simple machine learning techniques like boosted decision trees for event classification have been used in particle physics for a long time, the success of neural networks and deep learning has led to various new applications for machine learning. This includes event classification, where especially top tagging is studied extensively [1], and anomaly detection [2] for searches for new physics. Futhermore, the third run of the LHC and the High Luminosity LHC will provide vast amounts of new data and excellent tools to perform precision measurements on this data can be constructed using machine learning [3]. Also, faster event generation methods will be required to keep up with the experimental data. Neural networks can help to accelerate various parts of the event generation process [4].

Parton showers are ubiquitous objects in collider events, therefore understanding their behavior and the underlying QCD splitting functions is an important task. These splitting functions describe the kinematics of parton showers in the soft-collinear limit. At leading order, the splittings are determined by the QCD casimirs C_A and C_F that were measured at LEP [5, 6]. In this thesis, we want to expand this LEP analysis by using a more general parameterization of the QCD splittings. Using machine learning techniques, we want to base our analysis on low-level sub-jet observables. Furthermore, more accurate describtions of parton showers were studied in recent years [7–12]. By including a general term accounting for corrections from these improved models in our parameterization, we want to examine whether our method is capable of measuring such corrections.

Our approach to achieve these goals is based on normalizing flows. Normalizing flows are a family of machine learning models that learn an invertible mapping between probability distributions [13–15]. Invertible neural networks (INNs) are one realization of such normalizing flows [14, 16]. Conditional INNs (cINNs) are an extension of these networks where this transformation is conditioned on additional network inputs [17]. Normalizing flows and invertible networks have been used for various particle physics applications, including event and phase space generation [18–22], detector and parton shower unfolding [23], density estimation [24] and anomaly detection [25]. In this thesis, we will use the BayesFlow method [26] to extract posterior distributions for QCD model parameters from low-level sub-jet observables.

Chapter 2 contains a short introduction to QCD and the Standard Model, and a description of the physics behind parton showers. In chapter 3, we give an overview over machine learning, neural networks and Bayesian statistics, followed by a description of normalizing flows, cINNs and the BayesFlow method. We then introduce a parameterization of the QCD splitting functions and describe how we implemented a toy parton shower generator based on this parameterization in chapter 4, and discuss our results extracting these parameters using BayesFlow. In chapter 5, we describe a more realistic simulation setup based on Sherpa [27] and Delphes [28] that takes hadronization and detector effects in account, and present our QCD parameter inference results.

The research in this thesis was carried out in close collaboration with Sebastian Bieringer, so many of the figures and results can be also found in his master's thesis as well as in our paper published on the project [29].

2 QCD, parton showers and jet physics

In this chapter, we will give a short overview over the Standard Model and quantum chromodynamics (QCD). We will define the QCD color factors and review their measurement at LEP. We will then derive the splitting functions and how they can be used to construct a Monte Carlo parton shower generator. After briefly discussing hadronization, detectors in collider experiments and jets, we will define six high-level jet observables that we will use later to benchmark the performance of our inference method.

2.1 The Standard Model of particle physics

The Standard Model of particle physics is a quantum field theory that describes our current knowledge about the fundamental particles and their interactions except for gravity. Extensive information on the Standard model and quantum field theory can be found in the literature (e.g. [30] or [31]), so the aim of the following sections is to give a brief overview over the topic, focusing on QCD.

The fundamental particles in the Standard Model are split into two groups according to their spin: fermions with half-integer spin and bosons with integer spin. Bosons act as mediators for the interactions. The massless photon mediates the electromagnetic interaction, the massive $W^{\pm}$ and Z bosons mediate the weak interaction and the massless gluon is the mediator of the strong interaction. Finally, there is the Higgs boson that gives mass to the other bosons and the fermions. The fermions are divided into quarks, charged leptons and neutrinos. The quarks can interact through all three Standard Model interactions, the charged leptons interact electromagnetically and weakly, and the neutrinos only take part in the weak interaction. There are three up-type quarks – up, charm and top – and three down-type quarks – down, strange and bottom. With masses below 100 MeV, the up, down and strange quark are referred to as the light quarks. Their masses can be neglected in many cases, e.g. cross section calculations for processes at colliders. The three charged leptons are the electron, muon and tau lepton, and there is a corresponding neutrino for each of the charged leptons.

As the Standard Model is a quantum field theory, the particles listed above are understood as excitations of a quantum field. Their dynamic is specified by the Lagrangian $\mathcal{L}$ of the theory. Since it is not a classical field theory, it is not enough to solve the Euler-Lagrange equation for $\mathcal{L}$, instead quantum fluctuations have to be taken into account, for example using the path integral method. Calculating

observables like cross-sections can often be done perturbatively by drawing Feynman diagrams of the studied processes and translating these into the corresponding matrix element equations. The Standard Model is formulated in terms of *gauge theories*. In a gauge theory, the Lagrangian is invariant under local transformations from a Lie group, called the *gauge group* of the theory. These transformations act on the fermion fields, but also on the gauge boson fields corresponding to the group. These are necessary to keep the Lagrangian invariant. The gauge group of the Standard Model is $\mathrm{SU}(3)_C \times \mathrm{SU}(2)_L \times \mathrm{U}(1)_Y$. The group $\mathrm{SU}(3)_C$ is responsible for the strong interaction. $\mathrm{SU}(2)_L \times \mathrm{U}(1)_Y$ are the gauge groups of the electroweak interaction, the symmetry $\mathrm{U}(1)_Y$ is called hypercharge. A more detailed description of gauge theories will be given in the following section, using QCD as an example.

It is not possible to make gauge bosons massive in a gauge-invariant way without adding an additional field. Moreover, $\mathrm{SU}(2)_L$ acts only on left-handed particles and therefore prevents gauge invariant fermion mass terms in the Lagrangian. This problem can be solved by introducing the Higgs field as a complex hypercharge doublet that gets a non-zero vacuum expectation value through spontaneous symmetry breaking. This results in new mass eigenstates of the electroweak gauge fields, the massive $W^{\pm}$ and Z bosons and the massless photon, where the latter two are linear combinations of the $\mathrm{U}(1)_Y$ gauge field and one component of the $\mathrm{SU}(2)_L$ gauge field. Moreover, the Higgs boson arises as a new excitation after spontaneous symmetry breaking. Fermion masses can be added by introducing Yukawa interactions with the Higgs field.

The Standard Model was completed by the experimental confirmation of the Higgs boson's existence in 2012 at the LHC [32, 33] and it has been very successful in explaining the observations from collider experiments. However, there is cosmological evidence for physics beyond the Standard Model since the behavior of dark matter cannot be explained using only the Standard Model particles and interactions. Also, since the Standard Model conserves the Baryon number it is not able to explain the asymmetry between matter and anti-matter in our universe.

2.2 Quantum chromodynamics

2.2.1 Gauge theories

Gauge groups are continuous groups, called *Lie groups*. A representation R of a Lie group is a mapping from the group to matrices acting on a vector space. In physics, this vector space is often also reffered to as the representation. The dimension of a representation is the dimension of this vector space. Each representation has a set of *generators* $\{T^a\}$ that can be understood as a basis of infinitesimal group transformations. Finite transformations can be obtained using

$$U = \exp(i\alpha^a T_R^a) \tag{2.1}$$

with parameters α. All representations of a Lie group have the same commutation structure, given by

$$[T_R^a, T_R^b] = i f^{abc} T_R^c \,, \tag{2.2}$$

where f^{abc} are group-specific *structure constants*. This is called the *Lie algebra* of the group. The smallest non-trivial representation of a group is called the *fundamental representation* F. We can also define the *adjoint representation* A with the generators

$$(T_A^a)^{bc} = -i f^{abc} \,. \tag{2.3}$$

The most important family of Lie groups are the special unitary groups SU(N). Their fundamental representations are the SU(N) matrices with dimension N and the adjoint representations have the dimension $N^2 - 1$.

Quantum chromodynamics (QCD) is the quantum field theory that describes the strong interaction. It is part of the Standard Model and is a SU(3) Yang-Mills theory. Yang-Mills theories are a generalization of quantum electrodynamics (QED), allowing for non-Abelian gauge groups. The elements of such groups do not commute. Physically, this leads to tree-level interactions between gauge bosons. This is the main difference to QED, where interactions between photons only occur in suppressed loop processes. The QCD Lagrangian for a single quark flavor is

$$\mathcal{L} = i\bar{\psi}(\gamma^\mu D_\mu - m)\psi - \frac{1}{4} G^a_{\mu\nu} G^{a,\mu\nu} \tag{2.4}$$

with the gauge covariant derivative

$$D_\mu = \partial_\mu - i g_s A^a_\mu T^a \tag{2.5}$$

the gluon field strength tensor

$$G^a_{\mu\nu} = \partial_\mu A^a_\nu - \partial_\nu A^a_\mu + g_s f^{abc} A^b_\mu A^c_\nu \,, \tag{2.6}$$

and the adjoint Dirac spinor

$$\bar{\psi} = \psi^\dagger \gamma^0 \,. \tag{2.7}$$

The four Dirac matrices γ^μ are defined by the *Clifford algebra*

$$\{\gamma^\mu, \gamma^\nu\} = 2\eta^{\mu\nu} \mathbb{1} \,. \tag{2.8}$$

Their numerical values depend on the choice of basis. The quark spinors ψ are from the fundamental representation, hence there are three different *colors* of quarks. The generators of the fundamental representation are called the *Gell-Mann matrices* λ^a. The gluon fields live in the adjoint representation, resulting in eight gluons. Gauge transformations act on the quark fields according to

$$\psi \to U\psi \qquad \text{and} \qquad \bar{\psi} \to \bar{\psi} U^\dagger \tag{2.9}$$

with the local transformation

$$U(x) = \exp\left(i\alpha^a(x)\lambda^a\right) \tag{2.10}$$

and the gluon fields transform as

$$A^a_\mu(x) \to A^a_\mu(x) + \frac{1}{g_s}\partial_\mu\alpha^a(x) - f^{abc}\alpha^b(x)A^c_\mu(x)\ . \tag{2.11}$$

The third term in the gluon field strength tensor gives rise to three- and four-gluon interaction vertices that are not present in Abelian gauge theories like QED. This also means that the gluons themselves are not color-neutral in contrast to the charge-neutral photons of QED. Hence, in contrast to the spread out field lines in QED, the field of the strong interaction is concentrated to a narrow flux tube, resulting in a linear potential between quarks with a slope on the order of 1 GeV/fm. Consequently, we do not observe free quarks and gluons, instead we only find them in color-neutral bound states. This is called *confinement*.

2.2.2 Color factors

To get a basis-independent characterization of a Lie group representation, we can define the *quadratic Casimir* and the *index*. For a representation R, the quadratic Casimir $C(R)$ is defined as

$$T^a_R T^a_R = C(R)\mathbb{1} \tag{2.12}$$

and the index $T(R)$ is defined as

$$\mathrm{tr}\left\{T^a_R T^b_R\right\} = T(R)\delta^{ab}\ . \tag{2.13}$$

The most important representations are the fundamental (F) and the adjoint (A) representations. For those, we give the quadratic Casimirs the names C_F and C_A and we name the indices T_F and T_A. For the SU(N) groups, their values are:

$$C_F = \frac{N^2+1}{2N}\ , \qquad C_A = N\ , \tag{2.14}$$

$$T_F = \frac{1}{2}\ , \qquad T_A = N\ . \tag{2.15}$$

For the SU(3) gauge group of QCD, we get $C_F = 4/3$, $C_A = 3$ and $C_A/C_F = 2.25$. These constants are also called the *color factors*. For C_A, we can derive the identity

$$f^{acd}f^{bcd} = C_A\delta^{ab} \tag{2.16}$$

by substituting the generators of the adjoint group with the structure constants f in Eq. 2.12. This relation and the two relations

$$\mathrm{tr}\left\{T^a T^b\right\} = C_F\delta^{ab} \tag{2.17}$$

$$(T^a T^a)_{ij} = T_F\delta_{ij} \tag{2.18}$$

for the generators of the fundamental representation are used in almost every QCD calculations, hence the color factors appear in most results in QCD [31].

2.2.3 Measurements of the color factors

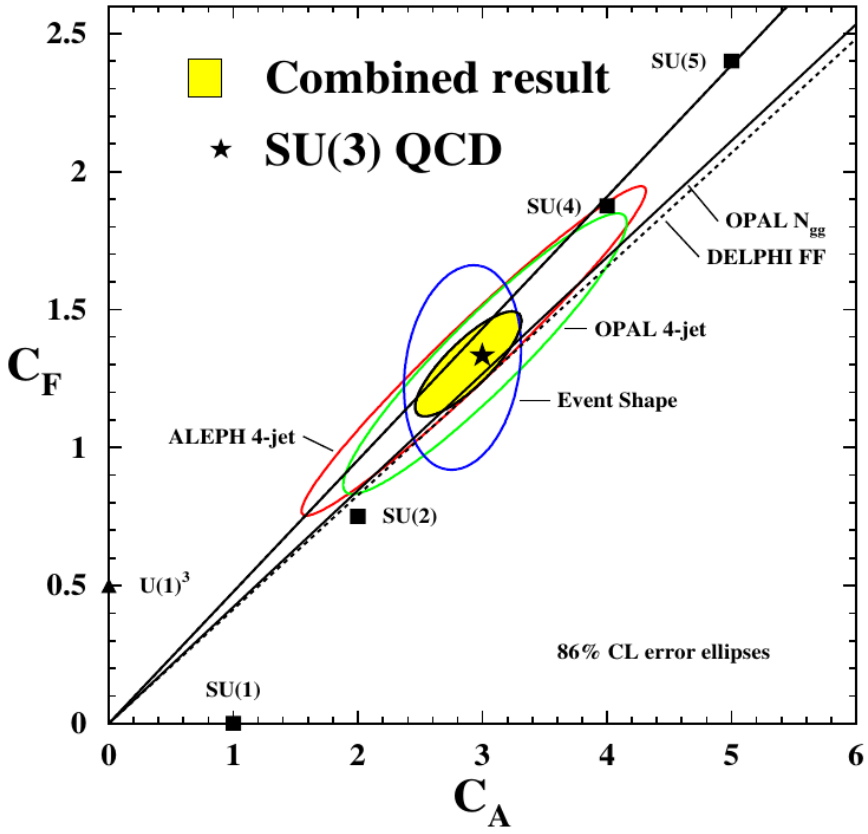


Figure 2.1: Results of the measurements of C_A, C_F and C_A/C_F at LEP [34–38] and the combined result. The figure was taken from [5].

Because the color factors are group-specific, measuring them is a way to check whether the gauge group of QCD is indeed SU(3). This was extensively studied at LEP. Because of the omnipresence of the color factors in QCD, there are multiple different ways to measure them.

At the OPAL detector, the charged particle multiplicities of two- and three-jet events were measured and the ratio C_A/C_F was calculated from the energy dependence of the multiplicity for quark- and gluon-jets. The result was $C_A/C_F = 2.23 \pm 0.14$ [34]. Another measurement of the ratio of the color factors using 3-jet events was performed by DELPHI where the fragmentation functions were measured and the result $C_A/C_F = 2.26 \pm 0.27$ was calculated from their scale-dependence using the DGLAP equations [35]. At ALEPH and OPAL, the color factors were also measured using the rate and angular corrections of four-jet events. The result from ALEPH is $C_A = 2.93 \pm 0.60$ and $C_F = 1.35 \pm 0.27$ [36] and the result from OPAL is $C_A = 3.02 \pm 0.56$ and $C_F = 1.34 \pm 0.30$ [37]. Using fits to the distributions of the event shape observables [39–43], $C_A = 2.84 \pm 0.24$ and $C_F = 1.29 \pm 0.18$ was measured [38].

Combining all these measurements yields the following results for the color factors:[5, 6]

$$C_A = 2.89 \pm 0.21 \qquad \text{and} \qquad C_F = 1.30 \pm 0.09 \; . \tag{2.19}$$

A plot summarizing the measurements mentioned above and the combined measurement can be seen in Fig. 2.1. The results are in excellent agreement with the predictions of SU(3) QCD. Also, it can be seen that the results for C_A and C_F are strongly correlated. The uncertainties are dominated by systematics.

2.3 Splitting functions and parton showers

In hard processes with quarks or gluons in the initial or final state, infrared divergences occur. This is caused by the collinear radiation of soft quarks and gluons from the hard partons. These radiated partons can again radiate more soft quarks and gluons, leading to higher-order processes with respect to α_s. This results in so-called *parton showers*. In this section, we will introduce the formalism behind parton showers following the derivation given in [44], and describe how to generate them using Monte Carlo methods.

2.3.1 Factorizing the phase space

To understand the infrared divergences, it is useful to factorize the $(n+1)$-particle phase space into the n-particle phase space and an universal *splitting kernel* or *splitting function* that depends only one the kinematics of the splitting vertex. Consider a mother parton, a, splitting into two daughter partons, b and c. One example is the vertex $q \to qg$. We define the energy fraction z of particle b and $1-z$ of particle c with respect to the mother parton as

$$z = \frac{E_b}{E_a} \; , \qquad 1 - z = \frac{E_c}{E_a} \; . \tag{2.20}$$

We assume a collinear splitting, i.e. the opening angle θ between $\vec{p}_b$ and $\vec{p}_c$ is small. Furthermore, we assume that the daughter partons are close to the mass shell, while the mother parton has to have a finite mass due to momentum conservation. As our convention for the momentum conservation, we will use

$$p_a = -p_b - p_c \; . \tag{2.21}$$

Consequently, we have to take absolute values to extract the energies, $E_i = |p_{i,0}|$. Using the momentum conservation equation and assuming small angles, we get

$$p_a^2 = z(1-z)E_a^2\theta^2 + \mathcal{O}(\theta^4) \quad \Longrightarrow \quad \theta \simeq \frac{1}{E_a}\sqrt{\frac{p_a^2}{z(1-z)}} \tag{2.22}$$

By introducing a free factor β, a unit four-vector n and a four-vector p_T that is orthogonal to p_a and n, we can write

$$-p_a = p_b + p_c = (-zp_a + \beta n + p_T) + (-(1-z)p_a\beta n - p_T) \; . \tag{2.23}$$

This is called the *Sudakov decomposition*. In general, the $(n+1)$-particle phase space is given by

$$\mathrm{d}\Phi_{n+1} = \prod_{i=1}^{n+1} \frac{\mathrm{d}^3\vec{p}_i}{2(2\pi)^3 E_i} \tag{2.24}$$

By setting $p_n = p_b$ and $p_{n+1} = p_c$ and choosing a frame of reference where the momentum $\vec{p}_a$ of the mother parton is oriented in the z direction, this can be rewritten in terms of the n-particle phase space and the momentum of the third particle p_c, the splitting transverse momentum p_T from the Sudakov decomposition and the azimuthal splitting angle ϕ:

$$\mathrm{d}\Phi_{n+1} = \mathrm{d}\Phi_n \, \frac{\mathrm{d}p_{c,3}\, \mathrm{d}p_T^2\, \mathrm{d}\phi}{4(2\pi)^3\, E_c\, z} \; . \tag{2.25}$$

In the collinear limit, it can be shown that

$$p_T^2 = z(1-z)p_a^2 \quad \Longrightarrow \quad \frac{\mathrm{d}p_T^2}{p_T^2} = \frac{\mathrm{d}p_a^2}{p_a^2} \tag{2.26}$$

and

$$\frac{\mathrm{d}p_{c,3}}{\mathrm{d}z} = \frac{E_c}{1-z} + \mathcal{O}(\theta) \; . \tag{2.27}$$

Those two results allow us to rewrite $\mathrm{d}\Phi_{n+1}$ again, this time in terms of z and p_a, as

$$\mathrm{d}\Phi_{n+1} = \mathrm{d}\Phi_n \, \frac{\mathrm{d}z\, \mathrm{d}p_a^2}{4(2\pi)^2} \; , \tag{2.28}$$

where we additionally assumed an azimuthal symmetry. As our final steps, we assume that the matrix element for the $(n+1)$-particle process factorizes into the matrix element of the n-particle process and a splitting kernel that only depends on z,

$$\overline{|\mathcal{M}_{n+1}|^2} \simeq \frac{8\pi\alpha_s}{p_a^2} P(z) \overline{|\mathcal{M}_n|^2} \; , \tag{2.29}$$

and use that assumption to write down the cross section of the process in the collinear limit:

$$\sigma_{n+1} \simeq \int \sigma_n \frac{\mathrm{d}p_a^2}{p_a^2} \mathrm{d}z \frac{\alpha_s}{2\pi} P(z) \tag{2.30}$$

We can also write this expression as an integral over the splitting transverse momentum p_T using Eq. 2.26. This will be useful later as it allows us to use p_T^2 as an ordering parameter when defining the parton shower algorithm.

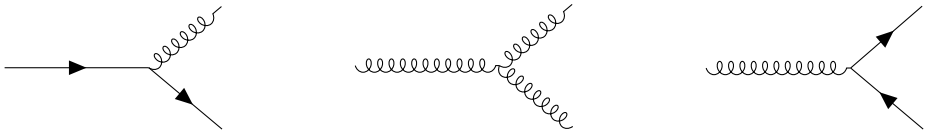


Figure 2.2: Feynman diagrams corresponding to the three QCD splitting functions

2.3.2 Splitting kernels

In the previous section, we introduced the splitting kernels $P(z)$ and assumed that they could be used to factorize the phase space. The specific form of the splitting kernels depends on the splitting vertices. In this section, we will sketch the derivation of the splitting function $P_{qq}(z)$ for a quark radiating a gluon and will then give the results for the other two splitting functions. Again, we follow the derivation given in [44].

Since we are in the collinear limit and have small scattering angles, we can write the spinors for the two spin states in the Dirac representation as

$$u_+(p) = \sqrt{E}\begin{pmatrix} 1 \\ \theta/2 \\ 1 \\ \theta/2 \end{pmatrix} \qquad \text{and} \qquad u_-(p) = \sqrt{E}\begin{pmatrix} -\theta/2 \\ 1 \\ \theta/2 \\ -1 \end{pmatrix} . \tag{2.31}$$

In this representation, the Dirac matrices have the form

$$\gamma^0 = \begin{pmatrix} \mathbb{1} & 0 \\ 0 & \mathbb{1} \end{pmatrix} \qquad \text{and} \qquad \gamma^j = \begin{pmatrix} 0 & \sigma^j \\ -\sigma^j & 0 \end{pmatrix} \tag{2.32}$$

where σ^j are the Pauli matrices. Using Feynman rules, we can write down the matrix element

$$\mathcal{M}_{qq} = -ig_s T^a \bar{u}_\pm(p_b)\gamma_\mu \epsilon_a^\mu u_\pm(p_a) \tag{2.33}$$

where p_a and p_b are the four-momenta of the incoming and outgoing quarks and p_c is the gluon four-momentum. The angles θ_a and θ_b in the Dirac spinors from Eq. 2.31 are given with respect to the gluon momentum $\vec{p}_c$ such that θ_b is equal to the scattering angle θ and in the the collinear limit, we can write $\theta_a = -z\theta$. The spin-, color- and polarization-averaged matrix element can be calculated by going through the matrix algebra, leaving us with the following result for the factorization of the $(n+1)$-particle matrix element:

$$\overline{\mathcal{M}_{n+1}^2} = \left(\frac{1}{p_a^2}\right)^2 g_s^2 \frac{\mathrm{tr}\{T^a T^a\}}{N_c N_a}\left(2(1-z) + 2\frac{(1+z)^2}{1-z}\right)\overline{\mathcal{M}_n^2} . \tag{2.34}$$

Here, $N_c = 3$ is the number of colors and $N_a = 2$ is the number of quark spin states. This equation can be further simplified and the definition of C_F for SU(N_c) from Eq. 2.14 can be used. The simplified equation has the form

$$\overline{\mathcal{M}_{n+1}^2} = \frac{2g_s}{p_a^2} C_F \frac{1+z^2}{1-z}\overline{\mathcal{M}_n^2} \tag{2.35}$$

Comparing this with our assumption from Eq. 2.29 gives us the result

$$P_{qq}(z) = C_F \frac{1+z^2}{1-z} \tag{2.36}$$

for the gluon-radiation splitting kernel. Hence, we have shown that the assumption was correct in the collinear limit. This is also the case for the two other splitting functions for the splitting of a gluon into two gluons and into two quarks. Figure 2.2 shows the vertices for all three splitting kernels. Evaluating the matrix elements and deriving the splitting functions yield

$$P_{gg}(z) = C_A \left(\frac{z}{1-z} + \frac{1}{z} + z(1-z) \right) \tag{2.37}$$

for the three-gluon splitting and

$$P_{gq}(z) = T_R \left(z^2 + (1-z)^2 \right) \tag{2.38}$$

for the gluon splitting into quarks. These are the Altarelli-Parisi splitting kernels [45].

2.3.3 DGLAP equation and regularization

Beside parton showers for outgoing partons at colliders, splitting kernels have a second important application. They can also be used to describe the scale dependence of the parton distribution functions of the incoming hadrons via the DGLAP equation [45–47]. To get the DGLAP equation, the evolution of a parton confined in a hadron with an energy fraction x that is initially on the mass shell is examined. In the case of a quark radiating gluons, the splittings decrease x while simultaneously increasing the virtuality t of the quark. This leads to an integro-differential equation for the parton density as a function of x and t that contains the splitting functions.

For use in the DGLAP equation and in cross section calculations where the splitting kernels have to be integrated, the splitting kernels must be regularized for the integrals to be finite. An example for such a regularization is the *plus subtraction scheme* which is defined as

$$F(z)_+ = F(z) - \delta(1-z) \int_0^1 F(y) \mathrm{d}y \ . \tag{2.39}$$

It can be shown that the poles of splitting kernels regularized this way are in agreement with results from dimensional regularization [44]. However, in parton shower generation algorithms we do not need such a regularization scheme. Instead, the integrals are finite because of the parton shower cut-off scale. This will be explained in greater detail in section 2.4.4.

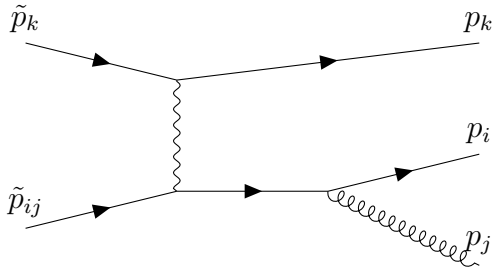


Figure 2.3: Feynman diagram for a $q \to qg$ Catani-Seymour dipole splitting. The upper incoming quark is the spectator parton and the lower incoming quark is the emitter parton.

2.4 Parton shower generators

So far, we looked at a single splitting process and derived the splitting kernels and phase space factorization for that case. However, to build a Monte Carlo parton shower generator, several splittings have to be generated in a randomized way, resulting in a splitting tree. This poses three problems. The first problem is that stricly speaking, we cannot look at the splittings in this tree as separate processes. Instead we have to take interference terms between them into account. However, it can be shown that using an ordered emission determined by the splitting kernels is a reasonable approximation and the interference can be neglected [44]. Secondly, in the derivation of the splitting kernels we assumed that the mother parton does not have to be on the mass shell, while the daughter partons are. This is no longer possible for multiple subsequent splittings. We can solve this problem with *Catani-Seymour dipoles*. Lastly, we need a way to sample which splitting occurs next and decide on the kinematics of that splitting. This problem is solved using *Sudakov form factors* and the *veto algorithm*.

2.4.1 Catani-Seymour dipoles

The problem of keeping all the partons in a parton shower on the mass shell and therefore allowing for multiple splittings can be solved by introducting a *spectator parton*. For each splitting of an *emitter parton*, such a spectator parton is selected and momentum is exchanged between the spectator and the emitter [48]. Fig. 2.3 shows a Feynman diagram for a $q \to qg$ splitting. The photon line denotes the momentum exchange and is not to be understood as an electromagnetic interaction. Here $\tilde{p}_k$ and p_k are the momenta of the spectator before and after the splitting. $\tilde{p}_{ij}$ is the emitter momentum before the splitting and p_i and p_j are the momenta of the two partons created in the splitting. These three momenta correspond to the momenta p_a, p_b and p_c in the derivation of the splitting kernels in the sections above. Then, the momentum conservation equation is

$$\tilde{p}_{ij} + \tilde{p}_k = p_i + p_j + p_k \,. \tag{2.40}$$

It is now kinetically possible to impose on-shell conditions for all the incoming and outgoing partons:

$$\tilde{p}_{ij}^2 = \tilde{p}_k^2 = p_i^2 = p_j^2 = p_k^2 = 0 \ . \tag{2.41}$$

As a new degree of freedom in the splitting process, we define the exchanged momentum fraction y such that

$$p_k = (1-y)\tilde{p}_k \ . \tag{2.42}$$

Using Eqs. 2.40 and 2.41, it can be shown that y is given by

$$y = \frac{p_i p_j}{p_i p_j + p_j p_k + p_k p_i} \ . \tag{2.43}$$

In analogy to the energy fraction z in the Altarelli-Parisi splitting functions (Eqs. 2.36 - 2.38), we define $\tilde{z}_i$ as the fraction of momentum transferred from the emitter parton to the first parton created in the splitting, but with both momenta projected on the incoming spectator momentum:

$$\tilde{z}_i = \frac{p_i \tilde{p}_k}{\tilde{p}_{ij} \tilde{p}_k} \ . \tag{2.44}$$

The momentum fraction for the second parton is $\tilde{z}_j = 1 - \tilde{z}_i$.

By replacing z with $\tilde{z}_i(1-y)$, the divergent part of the gluon-radiation splitting kernel has the form

$$\frac{1}{1-\tilde{z}_i(1-y)} = \frac{p_i p_j + p_j p_k + p_k p_i}{(p_i + p_k) p_j} \ . \tag{2.45}$$

It can be shown that both in the soft and the collinear limit, this description is in agreement with the results for splittings without Catani-Seymour dipoles [48]. Even at LHC energies, the predictions from Catani-Seymour dipoles are still sufficiently accurate, making them an ideal tool to describe parton showers at the LHC [44].

For the sake of readability, we will use $z := \tilde{z}_i$ for the rest of this thesis. In the Catani-Seymour dipole form, the three splitting kernels are as follows:

$$P_{qq}(z,y) = C_F \left[\frac{2z(1-y)}{1-z(1-y)} + (1-z) \right] \tag{2.46}$$

$$P_{gg}(z,y) = 2C_A \left[\frac{z(1-y)}{1-z(1-y)} + \frac{(1-z)(1-y)}{1-(1-z)(1-y)} + z(1-z) \right] \tag{2.47}$$

$$P_{gq}(z,y) = T_R \left[z^2 + (1-z)^2 \right] \tag{2.48}$$

Here, we have written them in a form where both the divergent and the finite parts are positive. This will become useful later when we introduce parameters for those different parts of the splitting functions

2.4.2 Sudakov form factor

Because ordered emission is used in parton shower generators, an ordering parameter t has to be defined*. A natural choice is the squared transverse momentum of the splitting

$$t := p_T^2 = 2\tilde{p}_{ij}\tilde{p}_k y z(1-z) \tag{2.49}$$

because the phase space integration for the total cross section in Eq. 2.30 can be expressed in terms of this variable. Also, it allows us to use a cut-off on the order of Λ_{QCD} because we cannot resolve multiple partons with lower transverse momenta towards each other and instead find them only in bound hadronic states [49].

Using Eqs. 2.30 and 2.26, we can write the differential probability for a splitting as a function of t as

$$\mathrm{d}\mathcal{P}_{ij}(t) = \frac{\mathrm{d}t}{t}\int \mathrm{d}z \frac{\alpha_s}{2\pi} P_{ij}(z,y) \ . \tag{2.50}$$

This equation can be used to derive an expression for the probability that no splitting occurs when the ordering parameter goes from t to t'. Iteratively applying the integral of the differential probability and summing up all the contributions results in an exponential function. The non-splitting probability, also called the *Sudakov form factor*, is then given by [44]

$$\Delta_i(t',t) = \exp\left[-\sum_j \int_{t_1}^{t_2} \frac{\mathrm{d}\tilde{t}}{\tilde{t}} \int \mathrm{d}z \frac{\alpha_s}{2\pi} P_{ij}(z,y)\right] \ , \tag{2.51}$$

where the sum runs over all the possible splittings for the parton $i \in \{q, g\}$.

For the parton shower algorithm, we need to sample the t' at which the next splitting is generated, given the current t. That means that we need the probability $\mathcal{P}_i^{(1)}(t',t)$ for a single splitting to occur at t'. It can be obtained from the probability that no splitting occured until t' (the Sudakov factor) by taking the t'-derivative [50], giving us the result

$$\mathcal{P}_i^{(1)}(t',t) = -\frac{\mathrm{d}\Delta_i(t',t)}{\mathrm{d}t'} \ . \tag{2.52}$$

*This t is not to be confused with the Mandelstam variable t, although the latter would be a suitable ordering parameter as well.

2.4.3 Rejection sampling

Typical pseudo-random number generators sample from a uniform distribution. For many Monte Carlo algorithms however, we have to draw random numbers from more complicated distributions. Consider a probability distribution $p(x)$ in an interval $[x_{\min}, x_{\max}]$. The cumulative distribution function is then

$$P(x) = \int_{x_{\min}}^{x} dx' p(x') \tag{2.53}$$

with $P(x_{\min}) = 0$ and $P(x_{\max}) = 1$. If $p(x) > 0$, we can invert P. By applying P^{-1} to random numbers from the unit interval, we can draw random numbers distributed according to $p(x)$. While this approach theoretically works for all positive probability distributions, it requires us to know P^{-1} for which we are not able to give an analytic expression in many cases.

This problem can be solved using *rejection sampling*. Consider a distribution $q(x)$ with $q(x) \geq p(x)$ for which we know the inverse cumulative distribution Q^{-1}. Let u_1 and u_2 be uniformly distributed random numbers between 0 and 1. Then $x_r = Q^{-1}(u_1)$ is distributed according to $q(x_r)$. We now reject x_r when $p(x_r)/q(x_r) < u_2$. The accepted random numbers are now distributed as $p(x)$. To make the algorithm efficient, $p(x)/q(x)$ should be as close to 1 as possible to keep the rejection rates low.

2.4.4 Sudakov veto algorithm

After collecting all the ingredients of parton shower generators, we will now discuss how a simple parton shower algorithm can be constructed. Starting from the center-of-mass energy of the whole process, continuously softer splittings are generated until a cut-off scale is reached. A typical value for the splitting transverse momentum cut-off is 1 GeV. For splittings with z close to 0 or 1, p_T goes to 0, so by introducing the p_T cut-off scale, we have also introduced upper and lower limits $z_{\min}$ and $z_{\max}$ for the energy fraction. Hence, we can use the unregularized splitting kernels in the parton shower algorithm [49].

Loosely speaking, in each evolution step in the parton shower algorithm we have to choose a spectator parton, an emitter parton and a splitting function and sample the transverse momentum for that splitting from the distribution given in Eq. 2.52. As the second step, we need to sample z from the probability distribution given by the splitting function. The form of the splitting kernels and the derivative of the Sudakov factor is too complicated to sample from directly. Instead, we have to make use of rejection sampling. For each splitting function $P_{ab}(z, y)$, an over-estimate $\tilde{P}_{ab}(z)$ has to be found, that can be analytically integrated and then inverted. It is possible to choose the overestimate such that it depends only on z. Let $I_{ab}(z_1, z_2)$ be the integral of $\tilde{P}_{ab}(z)$ from z_1 to z_2.

The algorithm starts by setting $t = p_T^2 = E_{\text{CMS}}^2$ where E_{CMS} is the center-of-mass energy of the hard process and therefore the maximal transverse momentum. It then iterates over all possible combinations of emitter and spectator partons and all possible splittings for that emitter. In the case of the $g \to q\bar{q}$ splitting, this also involves iterating over the quark flavors. By demanding t to be larger than the cut-off t_0,

$$t = 2\tilde{p}_{ij}\tilde{p}_k yz(1-z) \overset{!}{>} t_0 \ , \tag{2.54}$$

we get

$$z_{\text{max,min}} = \frac{1 \pm \sqrt{1 - \frac{2t_0}{\tilde{p}_{ij}\tilde{p}_k}}}{2} \ . \tag{2.55}$$

The t' at which the new splitting will happen has to be drawn from the distribution given in Eq. 2.52. As this distribution is too complicated to sample from directly, we use the over-estimated splitting instead. Then, the estimated Sudakov form factor can be written as

$$\tilde{\Delta}_{ab}(t', t) = \exp\left(- \int_t^{t'} \frac{\mathrm{d}\tilde{t}}{\tilde{t}} \frac{\alpha_s}{2\pi} I_{ab}(z_{\min}, z_{\max}) \right) \ . \tag{2.56}$$

Because the estimated splitting functions are y-independent, the integral inside the exponential function is easy to evaluate, resulting in

$$\tilde{\Delta}_{ab}(t', t) = \exp(g_{ab} \log t - g_{ab} \log t') \qquad \text{with } g_{ab} = \frac{\alpha_s}{2\pi} I_{ab}(z_{\min}, z_{\max}) \ . \tag{2.57}$$

Since the Sudakov form factor is the cumulative distribution of the probability for a splitting happening at t', to sample t' we have to invert $\tilde{\Delta}_{ab}$ and apply it to a random number. With a random number u_1 from a uniform distribution between 0 and 1, we get

$$t' = t \, \exp\left(\frac{1}{g_{ab}} \log u_1 \right) \ . \tag{2.58}$$

After calculating t' for all combinations of spectators, emitters and splittings, the largest t' is chosen. For that splitting, z can be sampled using the inverse of $\tilde{I}_{ab}$. y can then be calculated with Eq. 2.49. The azimuthal angle ϕ of the splitting with respect to the mother parton is the last kinematic degree of freedom of the splitting and is drawn from a uniform distribution. For the parton shower to be consistent with the splitting kernels even though we used the over-estimate $\tilde{P}_{ab}$ in the steps above [50], another uniformly distributed random number u_2 between 0 and 1 is drawn and the generated splitting is only accepted if

$$u_2 < \frac{P_{ab}(z, y)}{\tilde{P}_{ab}(z)} \ . \tag{2.59}$$

Because of this step, the algorithm is called the *Sudakov veto algorithm*. Then, t' is used as the new t and the steps above are repeated until the cut-off scale t_0 is reached.

2.5 Jets, hadronization and detectors

So far, we looked at parton showers and how to generate them using Monte Carlo methods. However, in an experimental setting, we cannot observe free quarks and gluons due to confinement. Instead, the partons undergo hadronization such that all quarks and gluons are in bound states. Moreover, detectors are needed to measure those particles and add additional uncertainties.

2.5.1 Hadronization

Hadronization is a complex QCD process and is not yet fully understood, so one has to resort to phenomenological models to describe and simulate it in event generators. As mentioned in section 2.2.1, for large distances there is a linearly increasing potential with a slope of about 1 GeV/fm between free quarks. This can be used to define a simple model of the hadronization process, the *string model*. In the following, this model will be explained for the simple example of a quark and an anti-quark propagating back-to-back. The linear potential is understood as a string between the quarks, corresponding to the narrow flux-tube between them that is formed because of the gluon self-interactions. As they move further apart the string expands and at some point it "snaps", creating a quark anti-quark pair. Now, there are four quarks connected by two strings. This process repeats, leading to more and more fragmentation of the string until the field energy is low enough that the quarks form bound hadronic states. These hadrons may be unstable and decay [51]. After hadronization, we are left with mesons like pions and kaons, baryons like protons and neutrons, and decay products like photons from π^0 decay and charged leptons and neutrinos from weak decays.

2.5.2 Detectors

A typical "general-purpose" detector like ATLAS and CMS at the LHC is built from four main components. Closest to the collision point is the *tracker* where the curved trajectory of charged particles in a magnetic field is measured. The *electromagnetic calorimeter* surrounds the tracker and measures the energy deposited by electrons, photons, but also hadrons. It has enough stopping power to absorb the energy of electrons and photons, but a *hadronic calorimeter* is needed to also absorb the energy of the hadrons. Since muons are able to penetrate both calorimeters, there is the *muon spectrometer* as a fourth layer. It is another tracking system to measure the muon trajectories.

Using the information from these components, the types and charges but most importantly, the four-momenta of the detected particles can be reconstructed. An example for such a reconstruction method is the CMS particle flow algorithm [52]. Detectors introduce multiple kinds of uncertainties. Trackers and calorimeters have

a limited spatial resolution. Furthermore, the measurement of the energy deposit in the calorimeter has an energy-dependent uncertainty. For example, the energy resolution of the ATLAS electromagnetic calorimeter is given by the expression

$$\frac{\sigma(E)}{E} \simeq \frac{10.1\sqrt{\text{GeV}}}{\sqrt{E}} \oplus 0.17 \tag{2.60}$$

where $\oplus$ stands for quadratic addition [53]. Hence, for a complete Monte Carlo event simulation setup, detector effects have to be simulated as well.

2.5.3 Jets

Starting from a hard quark or gluon, we saw that showering and hadronization create a cone of particles. This cone is called a *jet* and is a common object in most LHC analyses. However, in general the particles cannot be traced back to a single hard parton unambiguously because the angular separation between two or more hard partons can be small. Also, there are more sources for particles in collider events like initial-state radiation. Instead, jets are defined through *jet algorithms* that take a list of particles and assign them to jets. Important examples for jet algorithms are the k_T algorithm and the anti-k_T algorithm. Both will be used later in this thesis. The k_T algorithm follows a bottom-up approach by combining soft and collinear pairs of particles into sub-jets first. The algorithm aims to reconstruct the splitting tree [54] and will be discussed in greater detail in section 4.3. In contrast, the anti-k_T algorithm has a top-down approach and starts with hard and well-separated particles [55]. Typically, jet algorithms have a parameter R that determines the angular jet radius.

2.6 Jet observables

To get a better understanding about jets, it is useful to define jet observables. Given a list of four-momenta of the jet constituents, this allows us to get a low-dimensional view on the high-dimensional data. This not only helps humans to get insight into the data, it is also necessary for many inference methods to have a sufficiently low-dimensional representation of the data. Here, we will use a set of six established jet observables originally used for discriminating quark and gluon jets [56].

To define those observables, we have to first go from four-momenta in cartesian coordinates to a better suited coordinate system. The beam axis is usually defined to be the z axis. Given a four-vector (E, p_x, p_y, p_z), the transverse momentum (with respect to the beam axis) is given by

$$p_T = \sqrt{p_x^2 + p_y^2}\,. \tag{2.61}$$

Analogously, we can define the transverse energy

$$E_T = \sqrt{m^2 + p_x^2 + p_y^2} = \sqrt{m^2 + p_T^2} \tag{2.62}$$

for a particle with mass m. As a measure for the angle between the momentum and the beam axis, we define the pseudorapidity η as

$$\eta = \frac{1}{2} \log \left(\frac{E + p_z}{E - p_z} \right) \ . \tag{2.63}$$

As the third coordinate, the angle of the spatial momentum projected to the (x, y)-plane is used:

$$\phi = \text{arctan2}(y, x) \ . \tag{2.64}$$

The azimuthal angle ϕ is a symmetry of collider events. Therefore, only differences $\Delta\phi$ of the angles for multiple particles in one event are used. Using the last two definitions, we can define the angular separation

$$\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2} \ . \tag{2.65}$$

We can now define the six jet-observables. Let ΔR_{ij} be the angular separation between two particles and $\Delta R_{i,jet}$ between one particle and the total jet momentum. Also, we calculate the transverse momentum and energy for each individual particle and the total jet momentum. The jet observables are then given as

$$n_{PF} = \sum_i 1 \tag{2.66}$$

$$C_{0.2} = \frac{\sum_{i,j} E_{T,i} E_{T,j} (\Delta R_{ij})^{0.2}}{\sum_i E_{T,i}} \tag{2.67}$$

$$p_T D = \frac{\sqrt{\sum_i p_{T,i}^2}}{\sum_i p_{T,i}} \tag{2.68}$$

$$w_{PF} = \frac{\sum_i p_{T,i} \Delta R_{i,\text{jet}}}{\sum_i p_{T,i}} \tag{2.69}$$

$$x_{\max} = \max_i \left[\frac{E_{T,i}}{E_{T,\text{jet}}} \right] \tag{2.70}$$

$$N_{95} = \arg\min_n \left\{ p \;\middle|\; p = \sum_{i=1}^{n} p_{T,i}^{(\text{sorted})}, p \geq 0.95 p_{T,\text{jet}} \right\} \tag{2.71}$$

where all sums run over the jet constituents.

n_{PF} is the particle multiplicity [57]. In the case of measuring splitting functions, it is especially useful as it tracks the splitting probability. $C_{0.2}$ is a two-point energy correlator. The power 0.2 of ΔR_{ij} is optimized for the separation for quark and gluon jets [58]. The observable $p_T D$ measures whether the momentum is carried

by few hard particles or is spread among many soft particles [59]. w_{PF} is the girth (width of the radiation distribution) of the jet [60]. $x_{\max}$ is the maximal fraction of transverse energy contained in one jet constituent and N_{95} is the minimal number of jet constituents whose transverse momenta sum up to 95% of the total jet transverse momentum [61].

3 Machine learning and statistics

The aim of this thesis is to construct a machine-learning-based inference method to extract the form of the QCD splitting functions from (simulated) collider data. Machine learning is the process in which an algorithm builds a model from a data set (the training dataset) which it can then use to generate new data or to analyze other data, for instance by performing classification, inference or making predictions. After giving short introductions to machine learning with neural networks and Bayesian statistics, we will describe normalizing flows, how they can be implemented with invertible neural networks and how these can be used to perform Bayesian inference.

3.1 Neural networks

Artificial neural networks are one of the most important classes of machine learning models. They are built from neurons, simple units that perform mathematical operations on their inputs and have outputs that can be connected to other neurons. These neurons are often arranged in layers, where the neurons in one layer receive their input from the previous layer and send their output to neurons in the following layer. There are no connections between neurons of the same layer. That strongly simplifies the structure of the neural networks. The layers between the input and output layer are called hidden layers. The operations performed by the neurons typically have parameters that are varied during the training process to fit the model to the desired function.

Fully connected layers (also called dense layers) are a simple but important type of layer. Given an input vector $\boldsymbol{x}$ with dimension N, their output vector $\boldsymbol{y}$ with dimension M is given by

$$\boldsymbol{y} = f(W\boldsymbol{x} + \boldsymbol{b}). \tag{3.1}$$

Here, W is the $M \times N$ *weight matrix* and $\boldsymbol{b}$ is the *bias vector* with dimension N. Both are trainable parameters. Often, all trainable parameters are referred to as weights of the network. f is called *activation function*. Without the activation function, networks built from fully connected layers would only be linear functions and therefore not expressive enough to approximate arbitrary functions. By introducing non-linear activation functions and chaining multiple layers, approximating more complex functions becomes possible.

There are various commonly used activation functions. Most are applied element-wise to their input vector, but there are exceptions like *Softmax activation*. The following ones will be used in this thesis (if not stated otherwise, the function is to be applied element-wise):

- *Linear activation*: The linear activation function is the identity function $f(x) = x$. As discussed above, it cannot be used as the only activation function in expressive neural networks. However, it can be useful in some situations, for example as activation function for the last layer of a network.

- *ReLU activation*: The Rectified Linear Unit (ReLU) activation function is defined as $f(x) = \max\{0, x\}$. It is computationally cheap, yet usable in expressive networks. However, layers with ReLU activation often have a sparse output, i.e., many components of the output vector are zero. This is not always desirable, especially not for the output layer of a network.

- *ELU activation*: The Exponential Linear Unit (ELU) activation function is defined as

$$f(x) = \begin{cases} \alpha(e^x - 1) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}. \tag{3.2}$$

  For positive x, it behaves similar to ReLU, but it saturates at $-\alpha$ for large negative x. Therefore, contrary to ReLU, its output is not sparse. A common choice for α is 1.

- *Softmax activation*: For an input vector with M components x_i, the components of the output vector are given by

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{M} e^{x_j}}, \tag{3.3}$$

  such that $\sum_i y_i = 1$. Common uses for this activation function include final layers of classification networks, where the output components are interpreted as probabilities for different categories, and calculating weights for weighted averages.

3.2 Optimization

The training of the network is performed by an optimizer. The optimizer varies the trainable parameters of a model to minimize a *loss function* that specifies the objective of the training. Given a loss function $L(X; \boldsymbol{\alpha})$ where X is the training data and $\boldsymbol{\alpha}$ are the trainable parameters of the network, the optimizer in theory performs the operation

$$\boldsymbol{\alpha}_{\text{opt}} = \arg\min_{\boldsymbol{\alpha}} L(X; \boldsymbol{\alpha}) \tag{3.4}$$

In practice however, the $\boldsymbol{\alpha}$ found by the optimizer can differ from $\boldsymbol{\alpha}_{\text{opt}}$ due to insufficient convergence or local minima of the loss function. Most modern optimizers for neural networks are based the gradient descent method which updates the weights along the gradient of the loss function with respect to those weights. First, we express the total loss function $L(X;\boldsymbol{\alpha})$ as an average over the individual loss functions for the N points in the training data set:

$$L(X,\boldsymbol{\alpha}) = \frac{1}{N}\sum_{i=1}^{N} \ell(x_i;\boldsymbol{\alpha}) \ . \tag{3.5}$$

We can then define one gradient descent step as

$$\boldsymbol{\alpha}_{\text{new}} = \boldsymbol{\alpha} - \eta \nabla_{\boldsymbol{\alpha}} L(X;\boldsymbol{\alpha}) = \boldsymbol{\alpha} - \frac{\eta}{N}\sum_{i=1}^{N} \nabla_{\boldsymbol{\alpha}} \ell(x_i;\boldsymbol{\alpha}) \tag{3.6}$$

where the parameter η is the learning rate. For large data sets and complex models like deep neural networks where the evaluation of the gradients is expensive, doing a full gradient descent update is no longer feasible. Instead, the updates are performed for smaller portions of the data set, so-called mini-batches. After going through all mini-batches (this is called one *epoch*), the data set is shuffled and the optimization is continued. This modification of the method is called stochastic gradient descent (SGD). The backpropagation algorithm, a form of automatic differentiation, is used to compute the weight gradients of neural networks.

The plain stochastic gradient descent method suffers from problems like the risk to get stuck in local minima different from the global minimum and a slow rate of convergence. These problems are addressed by more sophisticated optimizers. One notable example is AdaGrad [62], which introduces individual learning rates for the weights of the model and adapts them during training depending on their gradients in previous iterations. In this thesis, we use the Adam [63] optimizer which takes the gradients and second moments of gradients from previous iterations into account with two decay constants (one for the gradients and one for the second moments) governing their influence on the current update. Adding momentum in this way makes it easier for the optimizer to escape local minima.

In addition, it is often advisable to introduce learning rate scheduling, i.e., decreasing the learning rate over time or adapting it based on the current value of the loss function. This way, fast convergence in the beginning of the training is possible without introducing large amounts of noise when the network is already well-converged. Another important aspect of a successful training is the initialization of the weights. Throughout this thesis, Glorot uniform initialization [64] will be used as the initialization method. In this initialization scheme, the weights of each layer are drawn from a uniform distribution over the interval $[-A, A]$, where $A = \sqrt{6/(n_{\text{in}} + n_{\text{out}})}$ takes into account the numbers of inputs n_{in} and outputs n_{out} of the layers.

3.3 Bayesian statistics

The Bayesian approach to statistics gets its name from Bayes' theorem. In contrast to the frequentist approach which focuses on repeatable experiments to test hypotheses, Bayesian statistics is formulated in terms of probability distributions modeling our knowledge and how we can update this knowledge given new experimental results.

Let $\boldsymbol{\theta}$ and $\boldsymbol{x}$ be vectors of random variables, where $\boldsymbol{\theta}$ stands for the parameters of some theory and $\boldsymbol{x}$ stands for data that was measured to infer these parameters. Using these definitions, we can write down Bayes' theorem:

$$p(\boldsymbol{\theta}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{x})} \tag{3.7}$$

The probability distributions in this equation are given names in Bayesian statistics. $p(\boldsymbol{\theta})$ is called the *prior* probability distribution, as it encodes our knowledge about the theory parameters before conducting the measurement. $p(\boldsymbol{x}|\boldsymbol{\theta})$ is called the *likelihood*. It defines the probability of a measurement outcome given a vector of parameters. $p(\boldsymbol{\theta}|\boldsymbol{x})$ is called *posterior* probability distribution. It tells us our new knowledge about the parameters θ after we updated our prior knowledge with the information gained from the measurement. Lastly, $p(\boldsymbol{x})$ can be understood as a factor that ensures the proper normalization of the posterior probability distribution. It can be calculated from the likelihood and the prior:

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}|\boldsymbol{\theta}')p(\boldsymbol{\theta}')\mathrm{d}\boldsymbol{\theta}' \tag{3.8}$$

We can see that Bayes' theorem enables us to update our knowledge about the theory parameters $\boldsymbol{\theta}$ if we know the likelihood for the theory. In practice however, the analytic expression for the likelihood is often not tractable, but it can be sampled from the likelihood using a Monte Carlo algorithm. In this case, the inference process is also called *likelihood-free inference* or *simulation-based inference*. The measurement of theory parameters in collider experiments is a good example for such a situation as we do not have a tractable likelihood, but we can use event generators to sample from it.

3.3.1 Approximate bayesian computation

Approximate Bayesian Computation (ABC) is a classical method for simulation-based inference [65, 66]. The ABC method is based on a problem-specific distance measure $\rho(x_{\mathrm{sim}}, x_{\mathrm{obs}})$ between simulated and observed data. A set of parameters $\{\theta^i\}$ is sampled from the prior and the simulation is performed with these parameters, yielding $\{x^i_{\mathrm{sim}}\}$. Only when the distance measure between a simulated point x^i_{sim} is

smaller than the threshold ϵ, the corresponding parameter θ^i is accepted. The set of accepted parameters approximately is a sample from the posterior. The choice of ϵ is a trade-off between the rate of acceptance and the accuracy of the posterior estimate. For a small ϵ, the posterior is estimated very accurately but the inference procedure is inefficient due to a high number of rejected samples.

ABC has two major problems. Firstly, whenever new data is processed, simulations have to be performed. This makes it a costly operation. In contrast, there are also *amortized* methods where the computation-heavy steps have to be performed only once and adding new data becomes a cheaper operation. Secondly, to define the distance measure ρ, problem-specific low-dimensional summary statistics are typically required when dealing with high-dimensional data. A poor choice of summary statistics can be detrimental to the quality of inference. These problems can be addressed using machine-learning techniques. In this thesis we will use the BayesFlow method that will be discussed in section 3.6 and solves these two problems [26]. A review of different methods for simulation-based inference and their advantages and disadvantages can be found in [67].

3.4 Normalizing flows

Normalizing flow models are a family of generative models that transform samples from one distribution into samples from another distribution in an invertible way. They were popularized by Rezende and Mohamed [13] and Dinh et al. [14]. The review by Kobyzev et al. [15] provides a comprehensive overview over the current landscape of normalizing flow models. Normalizing flows are built from smooth and bijective mappings $f : X \to Y$ between the spaces X and Y, where smoothness means that the Jacobian of the mapping is sufficiently well-behaved. When f is evaluated with values x drawn from the probability distribution $f_X(x)$, the resulting distribution can be obtained using the change of variables formula

$$p_Y(y) = p_X(x = f(y)) \left| \det \frac{\partial f^{-1}}{\partial x} \right| . \tag{3.9}$$

It is possible to chain multiple simple mappings $f_i : X_i \to X_{i+1}$ to get a more complicated mapping $f = f_n \circ \cdots \circ f_1$ between the spaces $X_1 \to X_2 \to \ldots \to X_{n+1}$. The resulting probability distribution $p_{X_{n+1}}(x_{n+1})$ given $p_{X_1}(x_1)$ can be calculated by iteratively applying equation 3.9.

To be used as a machine learning model, the mappings have to have trainable weights. The invertibility implies that the transformations must have the same input and output dimensions. Furthermore, there has to be an efficient way to calculate the Jabobian determinant for the model to be tractable. Given a data set with points x distributed according to a typically complicated probability distribution p_X, the model is evaluated in forward direction and trained to yield a simple output distribution p_Y like a (multivariate) Gaussian. This transformation towards

a simpler distribution is the reason why these models are named "normalizing flows". It is then possible to sample from p_X by drawing samples y from p_Y and evaluating the model in backwards direction: $x = f^{-1}(y)$.

3.5 Invertible neural networks

Invertible neural networks (INNs) are a realization of normalizing flow models introduced by Dinh et al. [14, 16]. They are built from a sequence of affine coupling blocks (ACB). The coupling blocks split their input vector into two halves $\boldsymbol{u} = (\boldsymbol{u}_1, \boldsymbol{u}_2)$. The minimal dimension of the input and output vectors is therefore two. The output vector $\boldsymbol{v} = (\boldsymbol{v}_1, \boldsymbol{v}_2)$ is then calculated with

$$\boldsymbol{v_1} = \boldsymbol{u}_1 \odot \exp(s_1(\boldsymbol{u}_2)) + t_1(\boldsymbol{u}_2) \tag{3.10}$$

$$\boldsymbol{v_2} = \boldsymbol{u}_2 \odot \exp(s_2(\boldsymbol{v}_1)) + t_2(\boldsymbol{v}_1)\ . \tag{3.11}$$

Here s_1, s_2, t_1 and t_2 are trainable sub-networks that do not have to be invertible, and $\odot$ denotes element-wise multiplication. Typically, they are fully-connected networks. The transformation given above can be easily inverted, resulting in the equations

$$\boldsymbol{u_2} = (\boldsymbol{v_2} - t_2(\boldsymbol{v}_1)) \odot \exp(-s_2(\boldsymbol{v}_1)) \tag{3.12}$$

$$\boldsymbol{u_1} = (\boldsymbol{v_1} - t_1(\boldsymbol{u}_2)) \odot \exp(-s_1(\boldsymbol{u}_2)) \tag{3.13}$$

for $\boldsymbol{u}$ as a function of $\boldsymbol{v}$. Furthermore, by writing the transformation in Eq. 3.10 as a vector operation performed in two steps, $(\boldsymbol{u}_1, \boldsymbol{u}_2) \to (\boldsymbol{v}_1, \boldsymbol{u}_2) \to (\boldsymbol{v}_1, \boldsymbol{v}_2)$, we can see that the resulting Jacobian has to be a product of two triangular matrices:

$$\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{u}} = \begin{pmatrix} \mathbb{1} & 0 \\ \text{finite} & \operatorname{diag} e^{s_1(\boldsymbol{u}_2)} \end{pmatrix} \begin{pmatrix} \operatorname{diag} e^{s_2(\boldsymbol{v}_1)} & \text{finite} \\ 0 & \mathbb{1} \end{pmatrix}\ . \tag{3.14}$$

As the determinant of a triangular matrix is the product of its diagonal elements and the determinant of the product of two matrices is equal to the product of their determinants, calculating the Jacobian of the ACB is computationally cheap. Because of the fixed splitting of the vector, one coupling block by itself is not expressive enough. To make the network more expressive, multiple coupling blocks have to be chained together, each with individually trainable weights of the sub-networks. Between the coupling blocks, a random but fixed permutation is applied to allow every INN input to have an effect on every output. An extension of this mechanism with trainable permutations was proposed in [68]. With the Jacobian determinant of the permutation layers being 1, the total Jacobian determinant is still easy to compute. This property together with its expressivity due to the sub-networks and the variable number of coupling blocks makes INNs a powerful realization of normalizing flows.

To improve the training stability by preventing diverging outputs, a bijective soft clamping function is applied in the coupling blocks after the exponential function in Eq. 3.10 is calculated [69]:

$$\text{clamp}(x) = \frac{2\alpha}{\pi} \arctan\left(\frac{x}{\alpha}\right) \tag{3.15}$$

Here, α is the limit for the absolute values of the outputs.

Conditional invertible neural networks (cINNs) are an extension of INNs, where the sub-networks s_1, s_2, t_1 and t_2 have an additional input $\boldsymbol{x}$ that is used to condition the operation of the coupling blocks [17]. The transformation in Eq. 3.10 now has the form

$$\boldsymbol{v}_1 = \boldsymbol{u}_1 \odot \exp(s_1(\boldsymbol{u}_2; \boldsymbol{x})) + t_1(\boldsymbol{u}_2; \boldsymbol{x}) \tag{3.16}$$
$$\boldsymbol{v}_2 = \boldsymbol{u}_2 \odot \exp(s_2(\boldsymbol{v}_1; \boldsymbol{x})) + t_1(\boldsymbol{v}_1; \boldsymbol{x}) \ . \tag{3.17}$$

The inverted equations change analogously.

3.6 BayesFlow method

BayesFlow is a method for simulation-based inference using normalizing flow networks [26]. It is based on conditional INNs and uses them to transform a Gaussian latent distribution $p(\boldsymbol{z})$ into the posterior distribution $p(\boldsymbol{m}|\{\boldsymbol{x}\})$ for theory parameters $\boldsymbol{m}$ given a set of measurements $\{\boldsymbol{x}\}$. A notable feature of the BayesFlow method is that it infers the parameters for the whole set of measurements at once. This is achieved with a *summary network*, an additional neural network that is trained along with the cINN, and that reduces the set of measurements to a fixed-size vector of summary statistics $\boldsymbol{h}$. This vector is then used to condition the cINN. Given a suitable architecture of the summary network, this makes BayesFlow able to cope with variable numbers of measurements, and it is able to learn the resulting contraction of the posterior. Also, the summary network makes it possible to work on high-dimensional data without having to construct summary statistics by hand. A second interesting feature of BayesFlow is that in contrast to most other methods for simulation-based inference [67], after the network is trained, applying it to new measurements is a very cheap operation and involves no further simulations. This compensates for the costly training process, making the method amortized.

3.6.1 Network architecture

As described above, the BayesFlow network is built from a cINN and a summary network. We write the forward direction of the cINN as $f_\phi(\boldsymbol{m}; \boldsymbol{h})$ where the second parameter is the condition. In this thesis, we will be only working with independent and identically distributed (i.i.d.) data, hence we will only discuss summary network

architectures for permutation-invariant measurements. However, BayesFlow can also be used for other types of data like time-series measurements, where a long short-term memory network (LSTM) is an example for a suitable architecture [70]. For permutation-invariant data, the summary network consists of a first sub-network $f_{\psi_1}(x_i)$ operating on the individual measurements, a pooling layer that reduces the result of the first sub-network to a single vector of summary statistics, and a second sub-network f_{ψ_2} operating on the pooled vector. When the pooling is done by averaging, the summary network can be written as

$$h = f_\psi(\{\boldsymbol{x}\}) = f_{\psi_2}\left(\frac{1}{M}\sum_{i=1}^{M} f_{\psi_1}(\boldsymbol{x}_i)\right) \tag{3.18}$$

with M being the number of measurements in $\{x\}$. An alternative for pooling by averaging is to use an attention mechanism where a third sub-network calculates the pooling weights for the individual measurements:

$$\tilde{w}_i = f_{\psi_3}(\boldsymbol{x}_i)\ . \tag{3.19}$$

An element-wise softmax operation is then performed to get normalized weights:

$$w_i = \exp(\tilde{w}_i) \oslash \sum_{i=1}^{M} \exp(\tilde{w}_i)\ . \tag{3.20}$$

The summary network can then be written as

$$h = f_\psi(\{\boldsymbol{x}\}) = f_{\psi_2}\left(\sum_{i=1}^{M} w_i \odot f_{\psi_1}(\boldsymbol{x}_i)\right)\ . \tag{3.21}$$

3.6.2 Loss function

The training objective for BayesFlow is to learn a mapping between the posterior distribution $p(\boldsymbol{m}|\{\boldsymbol{x}\})$ and a multivariate standard normal distribution $p(\boldsymbol{z})$ given a set of measurements $\{\boldsymbol{x}\}$. To implement this objective, we need a measure for the difference between two probability distributions. The Kullback-Leibler divergence is such a measure. For probability distributions $p(x)$ and $q(x)$, we can define it as

$$\mathbb{KL}(p\|q) = \mathbb{E}_{x\sim p(x)}\left[\log p(x) - \log q(x)\right] \tag{3.22}$$

where $\mathbb{E}_{x\sim p(x)}$ denotes the expectation value with respect to the random variable x distributed according to $p(x)$. For the sake of a more readable notation, we will notate sets of measurements $\{\boldsymbol{x}\}$ as $\underline{\boldsymbol{x}}$ in this section. The probability distribution $p(\underline{\boldsymbol{x}})$ is to be understood as a joint probability distribution over the number of observations and the probability distributions for the individual measurements $\boldsymbol{x}$ (that are all the same because we assume independent and identically distributed observations here).

With the KL divergence, we can write the training objective to find network weights ϕ and ψ for the cINN and the summary network that minimize the difference between the true posterior $p(\boldsymbol{m}|\underline{\boldsymbol{x}})$ and the estimated posterior $p_{\phi,\psi}(\boldsymbol{m}|\underline{\boldsymbol{x}})$ as

$$\hat{\phi}, \hat{\psi} = \arg\min_{\phi,\psi} \mathbb{E}_{\underline{\boldsymbol{x}}\sim p(\underline{\boldsymbol{x}})}[\mathbb{KL}(p(\boldsymbol{m}|\underline{\boldsymbol{x}})\|p_{\phi,\psi}(\boldsymbol{m}|\underline{\boldsymbol{x}}))] \tag{3.23}$$

$$= \arg\min_{\phi,\psi} \mathbb{E}_{\underline{\boldsymbol{x}}\sim p(\underline{\boldsymbol{x}})} \left[\mathbb{E}_{\boldsymbol{m}\sim p(\boldsymbol{m}|\underline{\boldsymbol{x}})} \left[\log p(\boldsymbol{m}|\underline{\boldsymbol{x}}) - \log p_{\phi,\psi}(\boldsymbol{m}|\underline{\boldsymbol{x}})\right]\right] \tag{3.24}$$

$$= \arg\min_{\phi,\psi} \mathbb{E}_{\underline{\boldsymbol{x}}\sim p(\underline{\boldsymbol{x}})} \left[\mathbb{E}_{\boldsymbol{m}\sim p(\boldsymbol{m}|\underline{\boldsymbol{x}})} \left[-\log p_{\phi,\psi}(\boldsymbol{m}|\underline{\boldsymbol{x}})\right]\right] \ . \tag{3.25}$$

In the last step, we used that $\log p(\boldsymbol{m}|\underline{\boldsymbol{x}})$ is an additive term that does not depend on ϕ or ψ and can therefore be left out of the optimization. Using the change of variables equation for normalizing flows (see Eq. 3.9) and writing the expectation values as integrals, we can rewrite the equation as

$$\hat{\phi}, \hat{\psi} = \arg\min_{\phi,\psi} \int\int \left(-\log p(f_\phi(\boldsymbol{m}; f_\psi(\underline{\boldsymbol{x}}))) - \log\left|\det \boldsymbol{J}_{f_\phi}\right|\right) \mathrm{d}\underline{\boldsymbol{x}}\, \mathrm{d}\boldsymbol{m} \tag{3.26}$$

As we train on finite training data sets, we go from the exact equation to a Monte Carlo estimate for a batch $\{\underline{\boldsymbol{x}}^i\}_{i=1}^N$ of N sets of observations:

$$\hat{\phi}, \hat{\psi} = \arg\min_{\phi,\psi} \frac{1}{N} \sum_{i=1}^{N} \left(-\log p(f_\phi(\boldsymbol{m}^i; f_\psi(\underline{\boldsymbol{x}}^i))) - \log\left|\det \boldsymbol{J}^i_{f_\phi}\right|\right) \ . \tag{3.27}$$

Finally we use that we want the distribution $p(f_\phi(\boldsymbol{m}^i; f_\psi(\underline{\boldsymbol{x}}^i)))$ to be Gaussian and hence write the equation above as

$$\hat{\phi}, \hat{\psi} = \arg\min_{\phi,\psi} \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{2}\left\|f_\phi(\boldsymbol{m}^i; f_\psi(\underline{\boldsymbol{x}}^i))\right\|_2^2 - \log\left|\det \boldsymbol{J}^i_{f_\phi}\right|\right) \tag{3.28}$$

$$=: \arg\min_{\phi,\psi} \frac{1}{N} \sum_{i=1}^{N} \ell((\boldsymbol{m}^i, \underline{\boldsymbol{x}}^i); (\phi, \psi)) \tag{3.29}$$

and use ℓ as a loss function as defined in Eq. 3.6.

3.6.3 Training and inference procedure

In the BayesFlow method, we distinguish between the training phase and the inference phase. In the training phase, we repeat the following steps until the network weights are sufficiently converged: First, we generate a batch of N parameter points by drawing the number of measurements M and the values of the parameters $\boldsymbol{m}$ from their respective prior distributions. For each $(\boldsymbol{m}^i, M^i)$, we then run the simulator for the theory and get a set of measurements $\underline{\boldsymbol{x}}^i$. Using the $\{(\boldsymbol{m}^i, \underline{\boldsymbol{x}}^i)\}_{i=1}^N$ as inputs, we calculate the loss function given in Eq. 3.28 and then use backpropagation and optimization to update the weights ϕ and ψ for the cINN and the summary

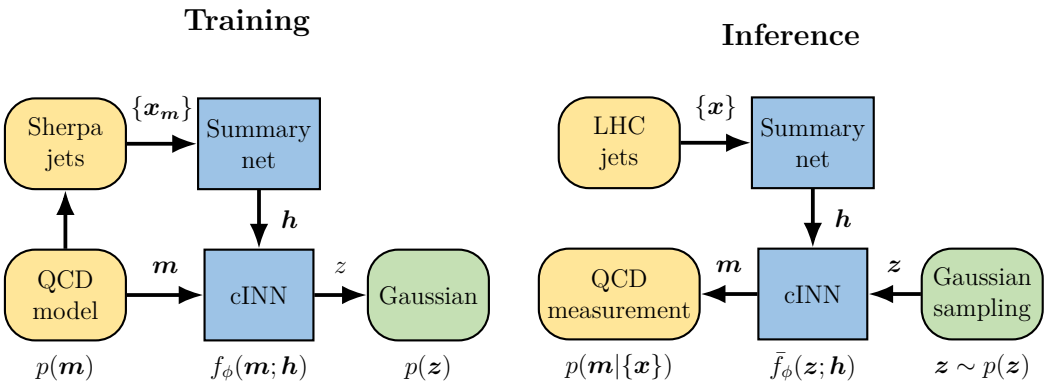


Figure 3.1: Overview of the flow of information in the BayesFlow method in our use case of learning QCD theory parameters from jets.

network. The sampling from the prior and the simulation step can be done in advance, but also during training. Optionally, the steps above except for updating the weights can be also repeated for a batch of testing data to get the loss function for data that is statistically independent from the training data. This is a way to detect over-fitting of the network. Over-fitting occurs when due to a lack of training data, the network learns features of the training data that are not features of the underlying model.

In the inference phase, we give the BayesFlow network a set of measurements $\underline{\boldsymbol{x}}$ and receive an estimate for the posterior distribution $p(\boldsymbol{m}|\underline{\boldsymbol{x}})$ by sampling from it. First, we draw G samples $\{\boldsymbol{z}^i\}_{i=1}^G$ from a multivariate normal distribution with the same number of dimensions as the number of theory parameters. For each of these samples, we evaluate the network in backward direction and get a parameter vector $\boldsymbol{m}^i$:

$$\boldsymbol{m}^i = \bar{f}_\phi(\boldsymbol{z}^i; f_\psi(\underline{\boldsymbol{x}}^i)) \; . \tag{3.30}$$

The resulting list of points $\{\boldsymbol{m}^i\}_{i=1}^G$ is a sample from the approximated posterior distribution. As mentioned before, the inference procedure is computationally very cheap compared to the traiing procedure and can be easily performed for multiple sets of measurements.

Fig. 3.1 gives an overview of the BayesFlow network architecture and how it is used during the training and inference phase.

3.7 Simulation-based calibration

Simulation-based calibration (SBC) is a tool to check the self-consistency of results from methods for simulation-based Bayesian inference [71]. It is based on the fact that for a posterior distribution $p(\theta|x)$ that is consistent with the likelihood $p(x|\theta)$, the following identity must hold:

$$p(\theta) = \int p(\theta|x)p(x)\mathrm{d}x = \int\int p(\theta|x)p(x|\tilde{\theta})p(\tilde{\theta})\mathrm{d}x\,\mathrm{d}\tilde{\theta}\ . \tag{3.31}$$

The rank statistic R is introduced to measure the similarity between the prior distribution and the posterior distribution averaged over the prior. Let $\tilde{\boldsymbol{\theta}}$ be a vector of parameters sampled from $p(\boldsymbol{\theta})$ and let x be the result of a simulation for these parameters, i. e., a sample from the likelihood $p(x|\tilde{\boldsymbol{\theta}})$. Typical methods for simulation based inference – with BayesFlow being no exception – give their result in the form of a set of samples $\{\boldsymbol{\theta}^i\}_{i=1}^G$ from the posterior $p(\tilde{\boldsymbol{\theta}}|x)$. In the following, $\boldsymbol{\theta}^i_j$ will denote the j-th component of the i-th vector of parameters. The rank statistic $R(\tilde{\boldsymbol{\theta}}_j; \{\boldsymbol{\theta}^1_j, \ldots, \boldsymbol{\theta}^G_j\})$ is defined as the position of $\tilde{\boldsymbol{\theta}}_j$ in a sorted list that contains $\boldsymbol{\theta}^i_j$ for all $i = 1, \ldots, n$ and $\tilde{\boldsymbol{\theta}}_j$ in ascending direction. If Eq. 3.31 holds, then the $\boldsymbol{\theta}^i_j$ must follow the same distribution as the $\tilde{\boldsymbol{\theta}}_j$ for a given j and therefore the latter cannot have a preferred position in the sorted list. As a consequence, R is uniformly distributed over the range of integers from 1 to G.

By drawing multiple parameter vectors from the prior and calculating the rank statistic for all the components of the vector and plotting a histogram of the rank statistic for each component, we can use this result to check the self-consistency of the inference method. If the histograms significantly differ from a uniform distribution, we can conclude that the inference result is not well-calibrated. In addition, we can get further information from the shape of the histograms in the case of non-uniform histograms. The histograms have a $\cap$ shape when the approximate posteriors on average over-estimating the width of the true posterior. Conversely, if the posteriors on average under-estimate the width of the true posterior, the histogram will have a $\cup$ shape. Asymmetric histograms are a sign for systematic deviations of the estimated posteriors.

However, a uniform histogram does not allow us to draw the conclusion that the inference method has found an optimal posterior estimator. Firstly, systematic deviations in one part of the prior can be compensated by deviations in the opposite directions in another part of the prior. The same applies to local over- and under-estimations of the posterior width. Secondly, in Eq. 3.31 we make no assumptions about x except for the consistency of the posterior and likelihood for that x. This is especially relevant if the inference method involves calculating summary statistics that could be not sufficiently expressive. In this case, the calculation of these statistics cannot be part of a consistent posterior estimation and would therefore be wrongly considered as part of the likelihood by SBC. For BayesFlow, that means that SBC is not able to detect poorly converged or not sufficiently expressive summary networks.

4 Extracting QCD splitting parameters from toy parton showers

In this chapter, we will first introduce a parameterization of the QCD splitting kernels and describe our implementation of a toy parton shower generator that allows us to generate events for different values of the parameters. After discussing different ways to order the four-momenta of a jet to make it easier for neural networks to extract the relevant information, we will give a detailed description of our BayesFlow training setup and network architecture and the motivation behind it. Finally, we will present our results for measurements of the QCD splitting parameters from the toy shower data.

4.1 Parameterizing the splitting kernels

The splitting kernels in the Catani-Seymour dipole formalism were introduced in section 2.4.1. In their Standard Model form, these contain the QCD color factors C_A, C_F and T_R. To get a model for measuring different contributions to the splitting kernels, we introduce seperate parameters for the divergent and finite parts of the splitting functions. This is done by first writing the splitting kernels in a form in which both the divergent and finite parts of the splitting functions are non-negative by themselves such that they can be separately understood as probability distributions. Furthermore, we want to test if our method is capable of measuring corrections to the splitting functions beyond the soft-collinear approximation. To model these, we add a general rest term proportional to p_T^2, resulting in constant corrections to the splitting probability according to Eq. 2.50. To get a dimensionless quantity, we use $yz(1-z) \sim p_T^2$ (see Eq. 2.49). Our parameterization of the splitting kernels is as follows:

$$P_{qq}(z,y) = C_F \left[D_{qq} \frac{2z(1-y)}{1-z(1-y)} + F_{qq}(1-z) + C_{qq} yz(1-z) \right] \tag{4.1}$$

$$P_{gg}(z,y) = 2C_A \Bigg[D_{gg} \left(\frac{z(1-y)}{1-z(1-y)} + \frac{(1-z)(1-y)}{1-(1-z)(1-y)} \right) + F_{gg}\left(z(1-z)\right) + C_{gg} yz(1-z) \Bigg] \tag{4.2}$$

$$P_{gq}(z,y) = T_R \left[F_{qq} \left(z^2 + (1-z)^2 \right) + C_{gq} yz(1-z) \right] \tag{4.3}$$

The parameters $D_{qq,gg}$ parameterize the divergent part of the splitting kernels, the $F_{qq,gg}$ parameters give the magnitude of the finite part and the $C_{qq,gg,gq}$ parameters

stand for higher-order contributions. As the divergent parts are the leading contribution, measuring them approximates the measurement of the color factors C_A and C_F. The F_{qq} parameter appears in both the P_{qq} and P_{gq} kernel because both are derived from the gqq vertex. The parameters for leading order Standard Model QCD are given by

$$D_{qq,gg} = 1, \qquad F_{qq,gg} = 1, \qquad C_{qq,gg,gq} = 0. \tag{4.4}$$

To measure the C parameters, the prior distributions for them have to be chosen such that the value 0 for the leading order Standard Model result is not at the boundary of the distributions. To prevent negative splitting kernels for negative C parameters, we set negative values to 0. For the D and F parameters, only non-negative values should be included in the prior distribution.

The veto algorithm is used to generate parton showers with our modified splitting kernels. Therefore, we need over-estimates of the splitting kernels that can be analytically integrated and then inverted. Furthermore, the estimates should be as close as possible to the actual splitting kernel to maximize the performance. We use the estimates

$$\tilde{P}_{qq}(z) = 2C_F \left(D_{qq} + \frac{F_{qq}}{2} + \left(\frac{C_{qq}}{8} \right)^+ \right) \frac{1}{1-z} \,, \tag{4.5}$$

$$\tilde{P}_{gg}(z) = 2C_A \left(D_{gg} + \frac{F_{gg}}{8} + \left(\frac{C_{gg}}{8} \right)^+ \right) \left(\frac{1}{1-z} + \frac{1}{1-(1-z)} \right) \,, \tag{4.6}$$

$$\tilde{P}_{gq}(z) = T_R \left(F_{qq} + \left(\frac{C_{gq}}{4} \right)^+ \right) \tag{4.7}$$

with $(x)^+ = \max\{x, 0\}$. Their correctness can be easily proven using the inequalities

$$\frac{z(1-y)}{1-z(1-y)} \leq \frac{1}{1-z} \,, \tag{4.8}$$

$$\frac{(1-z)(1-y)}{1-(1-z)(1-y)} \leq \frac{1}{1-(1-z)} \,, \tag{4.9}$$

$$yz(1-z) \leq z(1-z) \leq 1 \leq \frac{1}{4(1-z)} \qquad \left(\text{and } \leq \frac{1}{4(1-(1-z))} \right) \,, \tag{4.10}$$

$$z^2 + (1-z)^2 \leq 1 \,. \tag{4.11}$$

In implementations of parton shower generators, the P_{gg} splitting kernel is usually split up into two parts that can be transformed into each other under the exchange $z \leftrightarrow 1-z$. The two parts are given by

$$P_{gg,1}(z,y) = C_A \left[D_{gg} \frac{2z(1-y)}{1-z(1-y)} + F_{gg} \left(z(1-z) \right) + C_{gg} yz(1-z) \right] \,, \tag{4.12}$$

$$P_{gg,2}(z,y) = C_A \left[D_{gg} \frac{2(1-z)(1-y)}{1-(1-z)(1-y)} + F_{gg} \left(z(1-z) \right) + C_{gg} yz(1-z) \right] \tag{4.13}$$

and their over-estimates are

$$\tilde{P}_{gg,1}(z) = 2C_A \left(D_{gg} + \frac{F_{gg}}{8} + \left(\frac{C_{gg}}{8} \right)^+ \right) \frac{1}{1-z} \,, \tag{4.14}$$

$$\tilde{P}_{gg,2}(z) = 2C_A \left(D_{gg} + \frac{F_{gg}}{8} + \left(\frac{C_{gg}}{8} \right)^+ \right) \frac{1}{1-(1-z)} \,. \tag{4.15}$$

Integrating the estimates from z_1 to z_2 gives us the following results:

$$I_{qq}(z_1, z_2) = 2C_F \left(D_{qq} + \frac{F_{qq}}{2} + \left(\frac{C_{qq}}{8} \right)^+ \right) \log \left(\frac{1-z_1}{1-z_2} \right) \,, \tag{4.16}$$

$$I_{gg,1}(z_1, z_2) = 2C_A \left(D_{gg} + \frac{F_{gg}}{8} + \left(\frac{C_{gg}}{8} \right)^+ \right) \log \left(\frac{1-z_1}{1-z_2} \right) \,, \tag{4.17}$$

$$I_{gg,2}(z_1, z_2) = 2C_A \left(D_{gg} + \frac{F_{gg}}{8} + \left(\frac{C_{gg}}{8} \right)^+ \right) \log \left(\frac{z_2}{z_1} \right) \,, \tag{4.18}$$

$$I_{gq}(z_1, z_2) = T_R \left(F_{qq} + \left(\frac{C_{gq}}{4} \right)^+ \right) (z_2 - z_1) \,. \tag{4.19}$$

From these integrals, we can derive functions that transform a uniformly distributed $u \in [0,1]$ into $z_{qq,gg,gq} \in [z_1, z_2]$ distributed according to the estimates $\tilde{P}_{qq,gg,gq}(z)$. We get

$$z_{qq}(u) = z_{gg,1}(u) = 1 + (z_2 - 1) \left(\frac{1-z_1}{1-z_2} \right)^u \,, \tag{4.20}$$

$$z_{gg,2}(u) = z_1 \left(\frac{z_2}{z_1} \right)^u \,, \tag{4.21}$$

$$z_{gq}(u) = z_1 + u(z_2 - z_1) \,. \tag{4.22}$$

4.2 Parton shower generator

As our first step to measure the splitting function parameters, we implemented a toy parton shower generator. It is based on a Python code provided by Stefan Höche [72], but was rewritten in C for performance reasons. As our benchmark scenario, we choose the process

$$e^+e^- \to q\bar{q} \qquad \text{at} \qquad E_{\text{CMS}} = m_Z \,. \tag{4.23}$$

All quarks are assumed to be massless. This leaves two degrees of freedom for the process, the polar and azimuthal angle. In our toy event generator, they are drawn from a uniform distribution over the unit sphere. Then, a parton shower with a 1 GeV lower p_T cutoff is simulated using the veto algorithm described in section

2.4.4. By only using the second quark from the hard process as a spectator in the first splitting and excluding it from the shower generation afterwards, we generate a single jet from the first quark. This simplifies the generation process because less potential splittings have to be taken into account and no jet clustering step is needed.

The output of the algorithm for each event is a list of the parton four-momenta of the shower (excluding the second quark from the hard process). This list is created by first initializing it with the hard quark. Each time a new splitting is generated, the momentum of the mother parton in the list is replaced with the momentum of one of the daughter partons, while the four-momentum of the other daughter parton is appended at the end of the list. For a gluon-radiation splitting, the appended parton is always the gluon. Therefore, the order of the four-momenta in the list contains information about the splitting sequence.

4.3 Sorting the constituents

While the summary network is permutation invariant on the level of jets inside a parameter point, i.e., the order of the jets has no effect on the network output, it is not invariant on the constituent level. As described in section 4.2, the order in which our toy parton shower generator emits the four-momenta of the jets contains information about the order of the splittings and is therefore an information backdoor. In the following, this order of constituents will be called *truth sorting*. We found that a simple sorting scheme like p_T sorting significantly decreased the inference performance, so it is necessary to find a sorting scheme that more closely resembles truth sorting.

We achieve that using the k_T algorithm [54] as it not only gives us a reconstruction of the splitting tree but also the order of the splittings. Because we are already dealing with four-momenta of a single jet, we can simplify the algorithm by removing the cut-off condition and we do not have to consider beam radiation.

Our simplified k_T algorithm works by initializing a list of sub-jets with the four-momenta of the jet constituents. Using ΔR as defined in Section 2.6, we calculate

$$y_{ij} = \Delta R_{ij} \min\{p_{T,i}, p_{T,j}\} \tag{4.24}$$

for all pairs (i, j) of sub-jets and find the pair with the minimal y_{ij}. These two sub-jets are removed from the list and merged into a new subjet with momentum $p_{\text{new}} = p_i + p_j$ that is added to the list. This is repeated until only one sub-jet is left in the list. It carries the entire momentum of the jet and is the root of the reconstructed splitting tree.

In addition to the binary tree of sub-jets, we can also extract the order in which they were found. With this information, we can now sort the four-momenta. In the

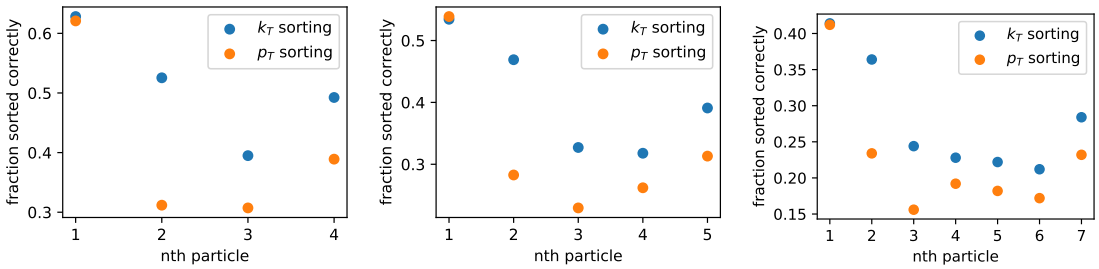


Figure 4.1: Fraction of toy shower events with 4, 6 and 7 constituents where the n-th particle was sorted correctly, for k_T- and p_T-sorted constituents.

reverse order that they were constructed in, we go through the sub-jets as follows: We determine which of the two child sub-jets carries the larger fraction of the parent sub-jet's energy. If this branch of the tree was not visited by the algorithm before, we go through the tree choosing following the subjets with the higher energy fraction until we reach a jet constituent. The four-momentum of the constituent is then appended to the sorted list of four-momenta. Going back to the starting point, we repeat the same steps for the sub-jet with the lower energy fraction.

If there is a limit for the number of output four-momenta, the reduction can be achieved by stopping the algorithm after a sufficient number of four-momenta is on the output list. We examined a second reduction method that conserved the total momentum of the jet by outputting sub-jet four-momenta instead of constituent four-momenta for the splittings found first during the reconstruction algorithm, effectively reducing the size of the tree. However, as the soft constituents found last during k_T sorting only have a small effect on the inference result, we found the first method to be simpler while resulting in a better inference performance. The order found by our k_T sorting algorithm is the same as truth sorting if

1. the tree was reconstructed correctly,

2. the order in which the subjets were merged is the reverse order of their production, and

3. for each splitting, the parton with the higher energy fraction was placed first in the list of partons.

From the description of truth sorting in Section 4.2 it is clear that especially the last point cannot be guaranteed for all splittings. In practice however, the reconstruction of the splitting tree and the order of the splittings is often not accurate as well. Still, the k_T sorting algorithm is better at reproducing truth sorting than simply sorting by p_T.

Fig. 4.1 shows a comparison between k_T sorting and p_T sorting for data generated by our toy shower generator. For each position in the sorted list of four-vectors, the fraction of constituents that are in the same position as in truth sorting is shown. The comparison is made for three different numbers of constituents. We observe

higher accuracies for lower numbers of constituents. Furthermore, k_T sorting (as well as p_T sorting) is more accurate for the first constituents in the list. A likely explanantion for this observation is that the tree reconstruction is less accurate for the soft jet constituents, affecting the assumptions (1) and (2) in the enumeration above. Furthermore, while assumption (3) holds for hard quarks emitting soft gluons, it becomes less likely to be true once triple-gluon vertices occur.

4.4 Network architecture and training setup

The BayesFlow network is implemented in TensorFlow 1.14 [73] and all trainings are performed on GPUs, as the training of neural networks strongly profits from parallelization. The code is based on the BayesFlow implementation by Stefan Radev [74]. The training data set consists of *parameter points* and each parameter point contains M jets generated with the same splitting kernel parameters. M can be either fixed or randomly drawn from a prior distribution to learn the measurement uncertainty as a function of the number of jets. Our training data sets either contain 10^5 parameter points or the training data is generated during the training. This choice will be discussed in more detail in the next section. We split the parameter points up into batches of 16 points, resulting in 6250 batches per epoch. For technical reasons, M has to be the same for every point in a batch. In addition to the training data, 10^4 parameter points of testing data are sent through the network in every epoch. The loss function for the testing data should be always similar or higher than the loss function of the training data, otherwise the setup suffers from over-fitting and a larger training data set is needed.

For trainings with a variable number M of jets per parameter point, we draw M from the interval $10^2 \dots 10^5$ with a probability distribution proportional to $1/M$ (reciprocal distribution) to compensate for the larger runtimes for points with a large M. Also, if M follows a reciprocal distribution, $\log M$ is uniformly distributed. In our case, that means that each order of magnitude of M constitutes approximately one third of the training data set. This makes the reciprocal distribution a very good choice when M spans multiple orders of magnitude. In our current BayesFlow implementation given our hardware limitations, we were not able to go beyond 10^5 jets per point.

The summary network is built from four fully connected layers with 64 units and two fully connected layers with 32 units. All layers except for the last one have ReLU activation. The last layer has a linear or ELU activation function to prevent sparse network outputs. It is followed by a pooling layer where the summary network outputs are averaged over all jets in the parameter point. We do not use a second stage of the summary network operating on the pooled values as we found it to be detrimental to the overall performance and especially the training stability of the network. The choice of the summary network architecture has a large effect on the inference performance. Making the summary network smaller in terms of depth, width and number of outputs leads to a worse performance. Further increasing its

size beyond the values given above does not lead to a better performance. Notably, we found the necessary number of summary statistics for a well-performing network and stable training to be much larger than the number of splitting kernel parameters that the network is trained on (32 compared to 2 or 3).

Because the summary statistics contain no information about the number of jets in a point, that information has to be added manually and has to be passed to the cINN. For trainings with a variable point size, we append $\sqrt{M}$ to the vector of summary statistics. Alternatively, summing instead of averaging could be used in the pooling layers. However, this would lead to the values of the summary statistics spanning several orders of magnitude, making it much harder for the cINN to process them.

The cINN part of the model consists of five affine coupling blocks. The sub-networks of the ACBs are fully connected networks with three layers with 64 units each and ELU activation functions. We observed that changes to the cINN architecture have much less effect on the inference performance compared to changes to the summary network architecture.

We use the Adam optimizer [63] to train the network. The optimizer starts with a learning rate of 10^{-3}, followed by a stepwise exponential decay:

$$\eta_t = 10^{-3} \cdot 0.99^{\lfloor t/n_s \rfloor} \tag{4.25}$$

The learning rate decay step size n_s turned out to be one of the most important hyper-parameters. A high learning rate decay results in large fluctuations of the network weights and this in turn results in systematic measurements errors. For example, the posterior means for one parameter might be systematically larger than the true value at one time, but after resuming the training for one more epoch, they might be all systematically lower when the performance is tested the next time. Low learning rate decays lead to systematic errors as well because the network is not fully converged. The optimal n_s depends on the training data, i.e., the choice of shower generator, splitting kernel parameters and sorting strategy, so it has to be tuned individually.

A summary of the network architecture and training parameters can be found in Table 4.1.

4.4.1 Methods to handle very large data sets

One major issue of the BayesFlow architecture is the large number of jets needed to train it. We need a sufficient amount of statistics per parameter point to get low measurement uncertainties of the splitting kernel parameters. Since each parameter point is reduced into a single vector of summary statistics that is passed to the cINN, we also need a sufficient amount of parameter points to train the network without over-fitting. For instance, for our experiments with a fixed number of jets per parameter point, the total number of jets in our training data set was $10^4 \cdot 10^5 = 10^9$,

resulting in file sizes exceeding 100 GB even for compressed data with a limited number of stored jet constituents. As the trainings were performed on a GPU cluster where the training data was not necessarily stored on the same machine that the training was running on, loading times posed a considerable performance problem. We developed two solutions to counter this problem.

As each generated event is statistically independent from all other events, the process of generating events is "embarassingly parallel". Therefore, it lends itself to be executed in parallel on a GPU. Because GPUs cluster the parallel threads they are running into *warps* and the threads in each warp have to run the same instruction at any time but operate on different data, efficient parallelization on GPUs requires programs with only a small amount of data-dependent branching. We found that this requirement is fulfilled in the case of the gluon radiation shower where only one splitting function is active. Hence, we are able to generate the training data during the training (*online learning*). Instead of repeating the same data in each epoch, new data is generated continuously, slightly improving the training stability. Especially for trainings on variable numbers of jets per data points, this proved to be advantageous. However, for the shower simulation including all the splitting functions, the simulation code is not efficient enough for online learning to be a viable option. We use online learning for all the trainings on gluon radiation showers in this thesis.

Our second solution of the data loading performance problem is to introduce "super-batches". Classically, one batch of parameter points would be loaded into the memory, and the network would be trained on it. Then, this would be repeated for every batch. Instead, we load multiple batches into memory at a time and then train the network on these batches multiple times. The number of batches in such a super-batch is limited by the available memory and should be as large as possible. Choosing the number of repetitions is a trade-off between a fast and stable training.

	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 layers	n_{layers}	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 4.1: BayesFlow network architecture and training hyper-parameters

We found that for 50 batches per super-batch with 25 repetitions per super-batch had a large benefit in terms of training time while not significantly reducing the training stability. We use the super-batch training method for all trainings on showers where all three splitting functions are enabled.

4.4.2 Attempts to improve the summary network architecture

Our largest improvements in inference performance were achieved by tuning the summary network architecture. Before settling for the architecture described above in combination with the k_T sorting algorithm, we tried out multiple other options. With these, we were mainly trying to tackle two problems of the summary network. Firstly, the summary network as it is used in the final training setup is not invariant under a change of the order of the input four-momenta. Therefore, the four-momenta have to be in a meaningful order. Also, the varying number of jet constituents has to be encoded in some way. We do this by zero-padding the four-momenta in our final setup. However, we also tried out order-invariant architectures that did not rely on zero-padding. Secondly, we observed that the summary statistics as a function of the splitting parameters were often very similar to each other, so we looked for ways to increase the expressiveness of the summary statistics. While none of these attempts ultimately led to better results, we will discuss them briefly in the following.

Radev et al. found that an attention mechanism used in the summary network as described in section 3.6.1 led to faster convergence and better performance [26]*. In our implementation of the attention mechanism, we used a fully connected network with ReLU activation functions that started with a layer with 64 nodes, followed by four to six layers with decreasing numbers of nodes. The final layer had one node and linear activation. We observed no beneficial effects of the attention mechanism. Instead, we found that it had a negative effect on the training stability and made the convergence slower (in terms of training time) because of the added complexity of the network.

In [75], an autoencoder was trained on zero-padded jet constituent four-momenta. The network performance significantly increased when the four-momenta were Lorentz-boosted into the jet frame as a pre-processing step. However, that was not the case for the BayesFlow network.

Looking for an architecture that would be invariant under the order of the input four-momenta, we tried to extend the idea behind the permutation-invariant summary networks to the four-momenta. In this three-stage summary network, a first network acts on each four-momentum individually. It is followed by a pooling layer, that takes the averages of the outputs of the first network over all four-momenta of the jets. We are then left with a fixed-size vector for each jet and these are used as inputs for the rest of the summary network which is the same as in the final architecture. Again,

*The discussion of the attention mechanism and its effects was removed in the fourth version of the BayesFlow paper [26] on arXiv.

we did not observe a positive effect on the network performance for this model. This was still true when the tree-stage summary network was used in conjunction with the attention mechanism in the first and/or second stage of the network.

Louppe et al. [76] proposed a network architecture acting on jet constituent four-momenta that takes the splitting tree into account by recursively applying a sub-network, resulting in a fixed-size vector of summary statistics. Thus, this architecture is an interesting candidate for the first stage of the summary network. We implemented this network for truth-sorted gluon-radiation showers where all gluons split off one quark. The recursive network had a similar performance to our normal network. We did not try this architecture for more complicated splitting trees where it would have to be used together with the k_T algorithm.

We tried two different methods to reduce the similarity between the summary statistics. First, we tried to pre-train the summary network as the encoder part of an autoencoder as proposed in [23]. However, we found that the effect of the pre-training on the summary statistics was only visible in the first epochs, and for the fully converged network, no difference between the results with and without pre-training was observed at all. Our second attempt was to add a term to the loss function that punishes correlations between the summary statistics. While it led to less correlations of the summary statistics to some extent, this method had a negative effect on the overall performance.

4.5 Results

4.5.1 Gluon-radiation shower

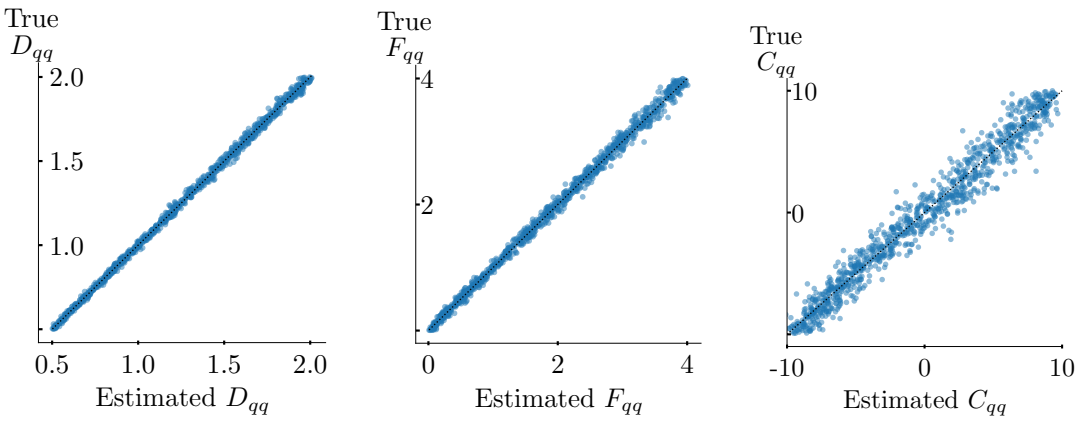
As a first benchmark of the inference, we will look at the performance of our network for showers with only gluon radiation from our toy shower generator. In this configuration, the splittings starting from a gluon are disabled and only the splitting kernel P_{qq} is taken into account in the generation process. We vary the parameters $\{D_{qq}, F_{qq}, C_{qq}\}$ and choose the prior to be a uniform distribution over the intervals

$$D_{qq} \in [0.5, 2], \qquad F_{qq} \in [0, 4], \qquad C_{qq} \in [-10, 10] \; . \tag{4.26}$$

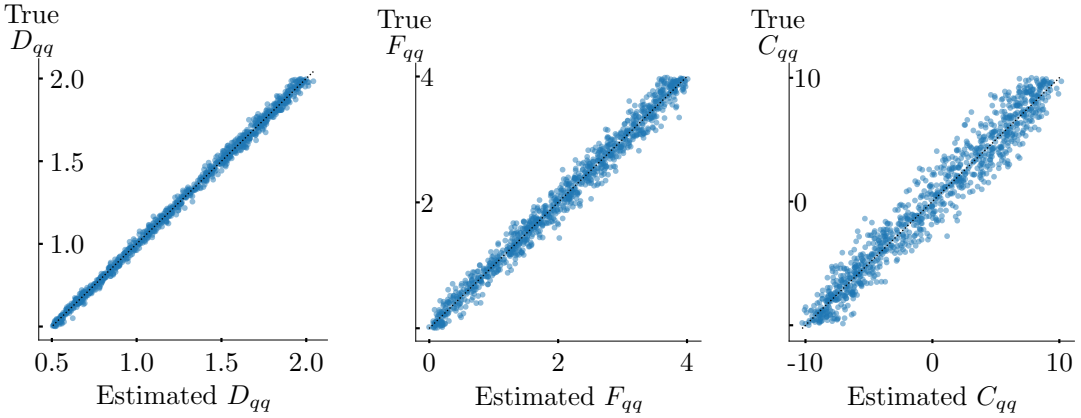
As the parton shower generation is computationally comparatively cheap for the gluon-radiation shower, we choose this model to test the scaling of the posterior width for trainings with a variable number M of jets per parameter point. M is sampled from a reciprocal distribution over the interval $[100, 100000]$. In the following, we will discuss the results of one training on truth-sorted data and one training on k_T-sorted data.

To evaluate the success of a training run and its quality of inference, we take multiple criteria into account. First, we look at the loss as a function of training epochs to

check whether the network weights are sufficiently converged. Next, we draw 1000 parameter points from the prior and generate a fixed number of jets per parameter point, $M = 10000$. For each point, we run the network in inference mode to sample 2000 points from the posterior. Using the means of the posterior sample as estimates for each of the three parameters, we can plot the estimated parameters against the true parameters to see the quality of inference over the whole prior. In Fig. 4.2, this is shown for both truth sorting and k_T sorting. It can be seen that there is no region of the prior with clear systematic deviations except for the boundaries of the prior intervals, where we can see some deviations of the points towards the inner of the prior interval. This is of course the expected outcome as the prior encodes our knowledge that the parameters should not be outside of these intervals.



(a) Truth sorting

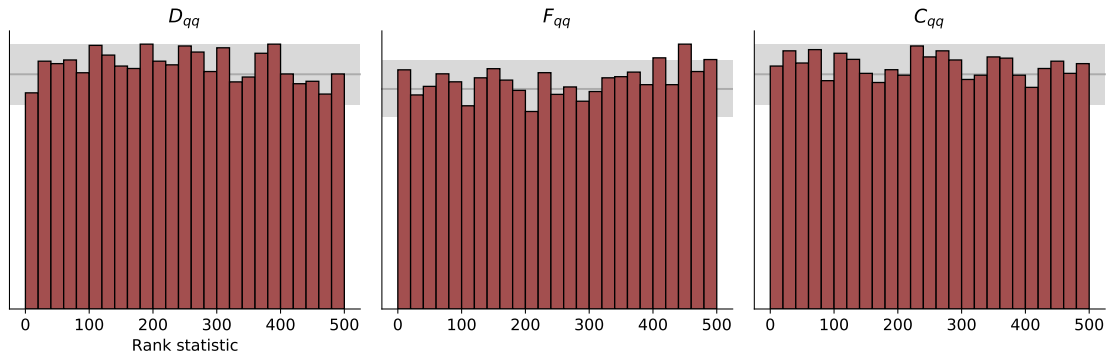


(b) k_T sorting

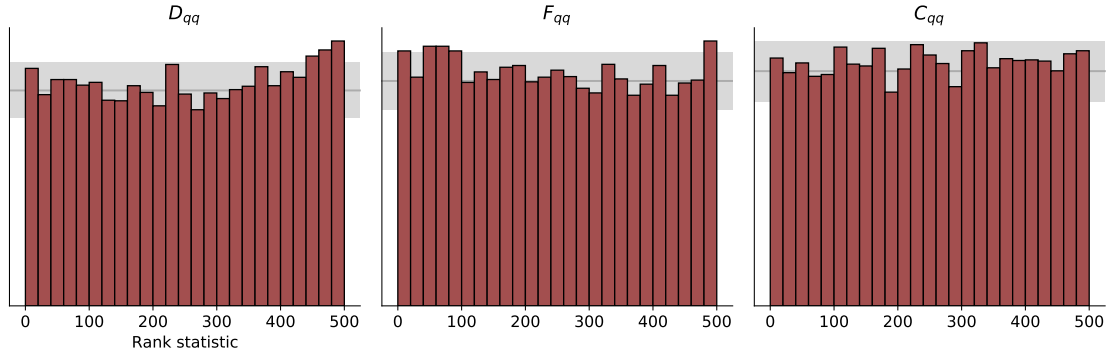
Figure 4.2: The estimated parameters plotted against the true parameters $\{D_{qq}, F_{qq}, C_{qq}\}$ for 1000 points randomly drawn from the prior for trainings on gluon-radiation showers with two different sortings of the four-momenta. The closer the points are to the dotted line, the better was the recovery of the true parameters.

Another way to assess the quality of inference over the whole prior is to use the simulation-based calibration method described in section 3.7. Again using 1000

points from the prior, 10000 jets per parameter point, but this time 500 samples from the posterior, we calculate the rank statistic for all the parameters for each parameter point. The histograms of the rank statistics for the two trainings with truth- and k_T-sorted data can be seen in Fig. 4.3. Both networks are well-calibrated. We only show the calibration plots for these two trainings, but all other trainings discussed in the following were checked to be well-calibrated too.



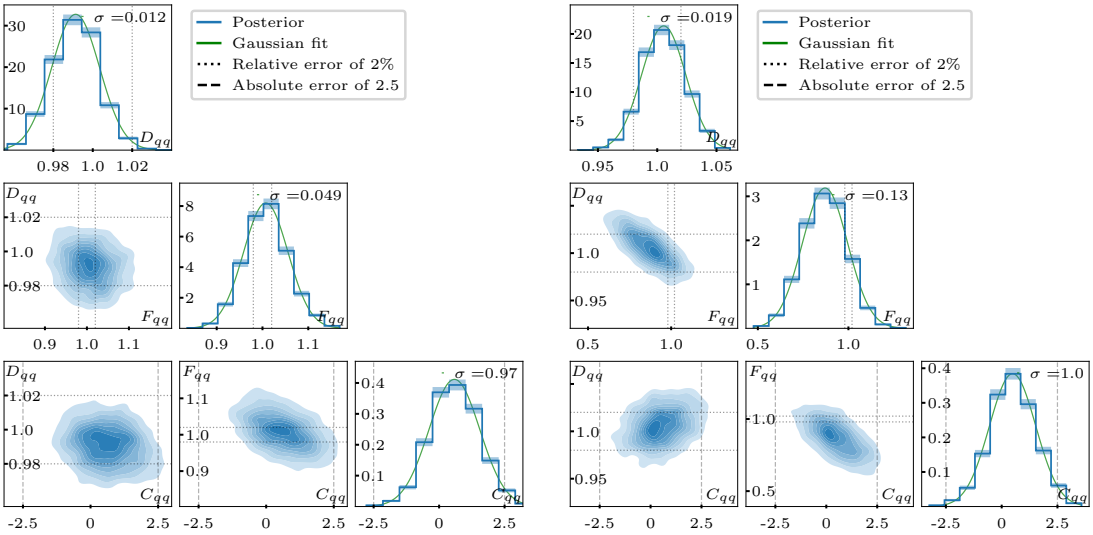
(a) Truth sorting



(b) k_T sorting

Figure 4.3: Histograms of the rank statistics for simulation-based calibration for the trainings on gluon-radiation showers, where the parameters $\{D_{qq}, F_{qq}, C_{qq}\}$ were varied.

The most interesting result from a physics standpoint is the performance of the network measuring jets with leading-order Standard Model QCD parameters. We will call them Standard Model parameters or speak about SM-like jets for brevity. In the case of the gluon radiation shower, the Standard Model parameters are $D_{qq} = 1$, $F_{qq} = 1$ and $C_{qq} = 0$. 10000 SM-like jets are generated and 2000 points are sampled from the posterior. Fig. 4.4 shows the marginal and bivariate posterior distributions for truth- and k_T-sorted training data. In addition, a fit to a Gaussian distribution is shown for the marginal posteriors. We generate multiple sets of jets at Standard Model parameters and choose a point for the plot where the distributions are centered around the true value to make the figure easier to interpret. In practice, the position of the maximum of the distributions fluctuates around the true value according to their width. It can be seen that the fitted Gaussians are in good agreement with the histograms for all the parameters and both truth and



(a) Truth sorting (b) k_T sorting

Figure 4.4: Posterior probabilities of the gluon radiation parameters $\{D_{qq}, F_{qq}, C_{qq}\}$ for 10000 SM-like toy shower jets for truth-sorted and k_T-sorted constituents.

k_T-sorted four-momenta. From the Gaussian standard deviations of $\sigma(D_{qq}) = 0.012$ and $\sigma(F_{qq}) = 0.049$ for truth sorting compared to $\sigma(D_{qq}) = 0.019$ and $\sigma(F_{qq}) = 0.13$ for k_T sorting, we can see that the performance for truth-sorted data is indeed much better due to the information backdoor. For the rest term parameter, the difference between the two different sorting strategies is only small with $\sigma(C_{qq}) = 0.97$ for truth sorting and $\sigma(C_{qq}) = 1.00$ for k_T sorting. The stronger correlations between D_{qq} and F_{qq} as well as F_{qq} and C_{qq} for the k_T-sorted constituents are another sign that infering from k_T-sorted data is the greater challenge for the network. These correlations also contribute to the increased width of the marginal distributions. In both cases, we can clearly see the hierarchical structure of the three parameters. The parameter D_{qq} for the regularized divergence has the largest effect on the splitting probability and is therefore the easiest to infer for the BayesFlow network. The parameter F_{qq} for the finite terms is slightly harder to infer and the measurement error is the largest for the p_T-suppressed rest terms.

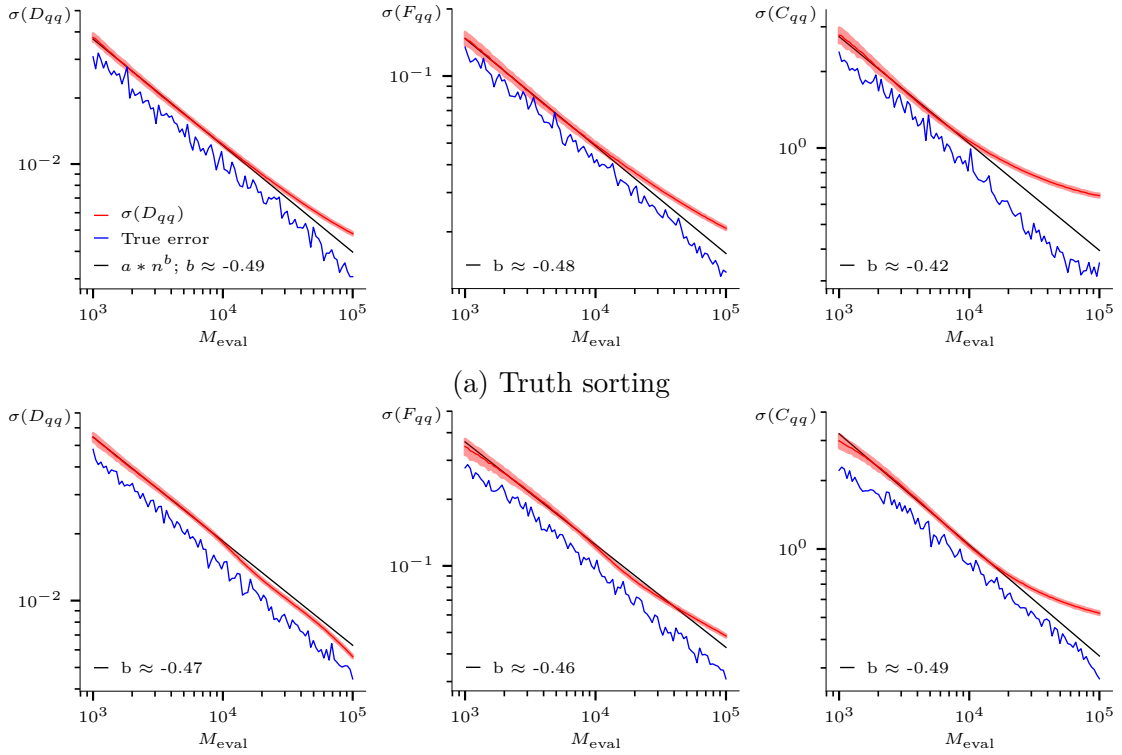
The results discussed above were all obtained for a fixed number of 10000 jets. But as we train the network with a variable number of jets per parameter point, we need to evaluate the network performance for different sizes M_{eval} of the evaluation data point. We look at 100 different M_{eval} spaced logarithmically between 10^3 and 10^5. For each M_{eval}, we generate 200 sets of jets of the respective size. For each set i with $i = 1, \ldots, 200$, we sample 2000 points from the posterior and calculate the means $\mu^i(T)$ and standard deviations $\sigma^i(T)$ for each theory parameter $T = D_{qq}, F_{qq}, C_{qq}$.

$$\sigma(T) = \frac{1}{200} \sum_{i=0}^{200} \sigma^i(T) \tag{4.27}$$

is the mean estimated error and we call

$$\sigma_{\text{true}}(T) = \frac{1}{200} \sum_{i=0}^{200} |\mu^i(T) - T| \tag{4.28}$$

the true error. The numbers 200 and 2000 are chosen to offer sufficient statistics for this analysis and are otherwise arbitrary.



(a) Truth sorting

(b) k_T sorting

Figure 4.5: Measurement error as a function of the number of jets for truth- and k_T-sorted gluon radiation showers. The red curve shows the mean and spread of the estimated posterior standard deviation for 200 sets of M_{eval} SM-like jets. The blue curve shows the true error, defined by the mean over the absolute differences between the true and estimated parameters. The black curve is a power law fit to the red curve.

The results for the estimated and true error along with a fit to a power law $a \cdot M^b$ can be seen in Fig. 4.5. For all three parameters, the network is slightly overestimating the measurement error. This is more prominent for the training on k_T-sorted data. As the comparison to the fit curve shows, the scaling of the estimated error as a function of M_{eval} is close to the expected $1/\sqrt{M_{\text{eval}}}$ power law. However, for the k_T-sorted data, we can see deviations from the expected behaviour for large M_{eval}, where there is a bend in the estimated error curves for all three parameters. While the true errors of D_{qq} and F_{qq} have the expected $1/\sqrt{M_{\text{eval}}}$ scaling, this is not the case for C_{qq}. This is caused by systematic deviations and influences the error estimation of all three parameters. These systematic deviations are the reason why it is necessary

to tune the learning rate decay as detailed in section 4.4. While the learning rate decay was tuned for both of the trainings, it is not always possible to completely prevent the systematic uncertainties from occuring. However, for a poorly tuned decay constant, the deviations from the expected behaviour would be much larger. Another way to improve the performance and stability for high M at the cost of longer training times, is to choose a uniform distribution for M in the training data set instead of a reciprocal distribution.

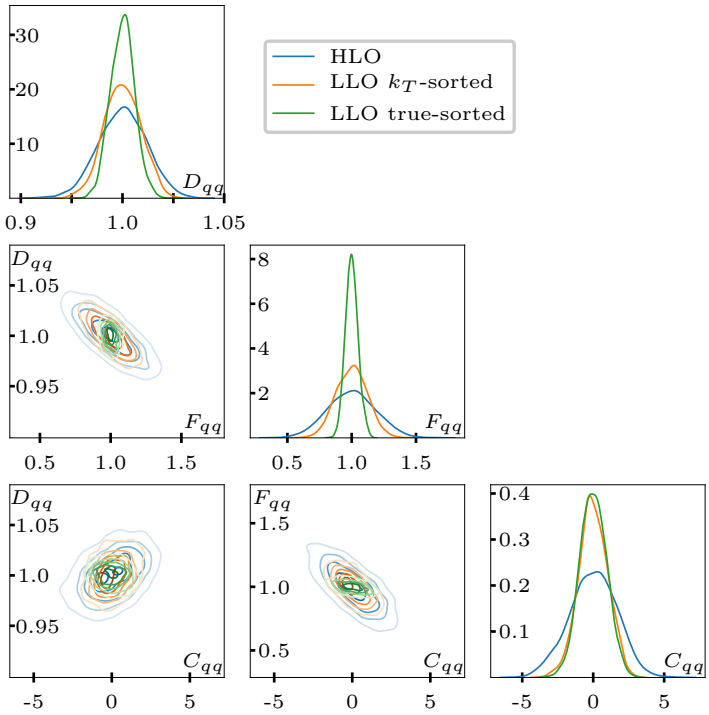


Figure 4.6: Posterior probabilities of the gluon-radiation parameters $\{D_{qq}, F_{qq}, C_{qq}\}$ for 10000 SM-like toy shower jets for a network trained on high-level observables and the networks trained on k_T- and truth-sorted low-level observables.

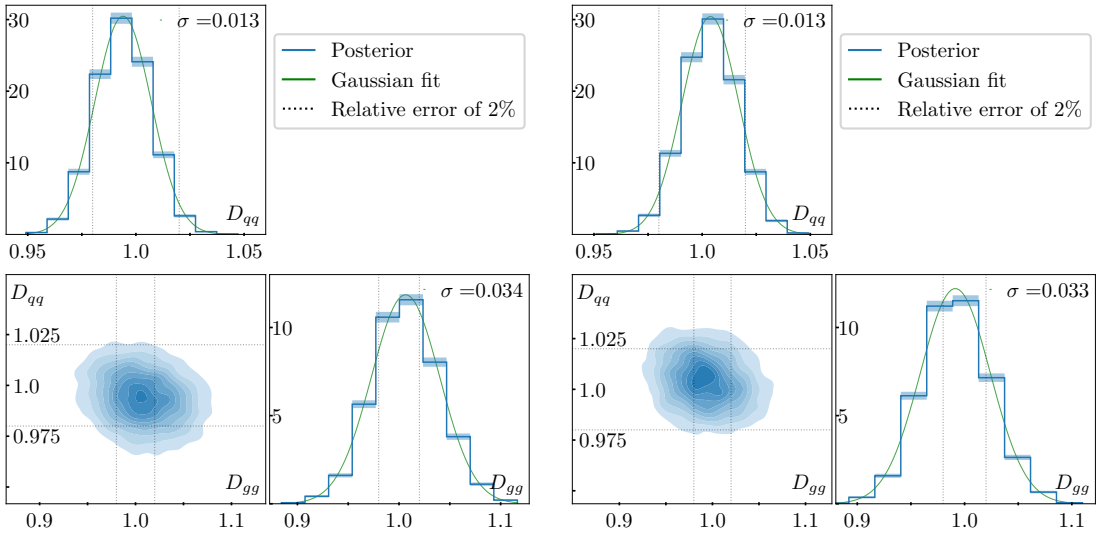
To show that the training on high-dimensional low-level observables performs better than the training on low-dimensional high-level observables, as it would be done in more traditional methods for Bayesian inference, we also trained a BayesFlow network on the six high-level observables introduced in section 2.6. The network architecture and training procedure is otherwise kept the same as for the low-level observables since we found this architecture to work the best also in the case of high-level observables. The posterior distributions for SM-like jets estimated for truth- and k_T-sorted low-level observables and high-level observables are shown in Fig. 4.6. From the posterior widths and the reduced correlations, it is clear that the network is able to extract more information from the k_T-sorted four-momenta than from the high-level observables. The training performance in terms of widths and correlations is the best for the truth-sorted data because of the additional information contained in the order of the four-momenta.

4.5.2 Measuring the leading terms

Having discussed the gluon-radiation shower in detail, we can proceed to the full QCD shower where all three splitting functions are enabled in the shower simulation. First, we will look at the extraction of the parameters for the soft-collinear divergences, D_{qq} and D_{gg}. This approximately corresponds to the measurement of the QCD color factors C_F and C_A under the assumption that the splittings are dominated by the leading terms. We choose a uniform prior again. The parameters are drawn from the intervals

$$D_{qq} \in [0.5, 2], \qquad D_{gg} \in [0, 3] \; . \tag{4.29}$$

Since we are generating quark jets, D_{qq} should not go down to zero because then, there would be no shower at all. This is not a problem for D_{gg}, so we can choose a larger prior interval. Again, we train the network with truth- and k_T-sorted data.



(a) Truth sorting (b) k_T sorting

Figure 4.7: Posterior probabilities of the leading parameters $\{D_{qq}, D_{gg}\}$ for 10000 SM-like toy shower jets for truth-sorted and k_T-sorted constituents.

Fig. 4.7 shows the posterior distributions for SM-like jets with $D_{qq} = D_{gg} = 1$. [†] The estimated error of D_{gg} is more than twice as large as the error of D_{qq}. This is the expected result, as we are only generating showers starting from a quark and we expect less gluon splittings as a consequence. It can be seen that there is no significant difference between the truth-sorted and k_T-sorted terms, indicating that the summary network might be learning simpler observables here that are less dependent on additional information about the splitting sequence. Moreover, in contrast to the LEP measurements, there is almost no correlation between D_{qq} and D_{gg}.

[†]The figures of the estimated parameters plotted against the true parameters for these trainings and other trainings where they are not referenced in the text, can be found in Appendix A.

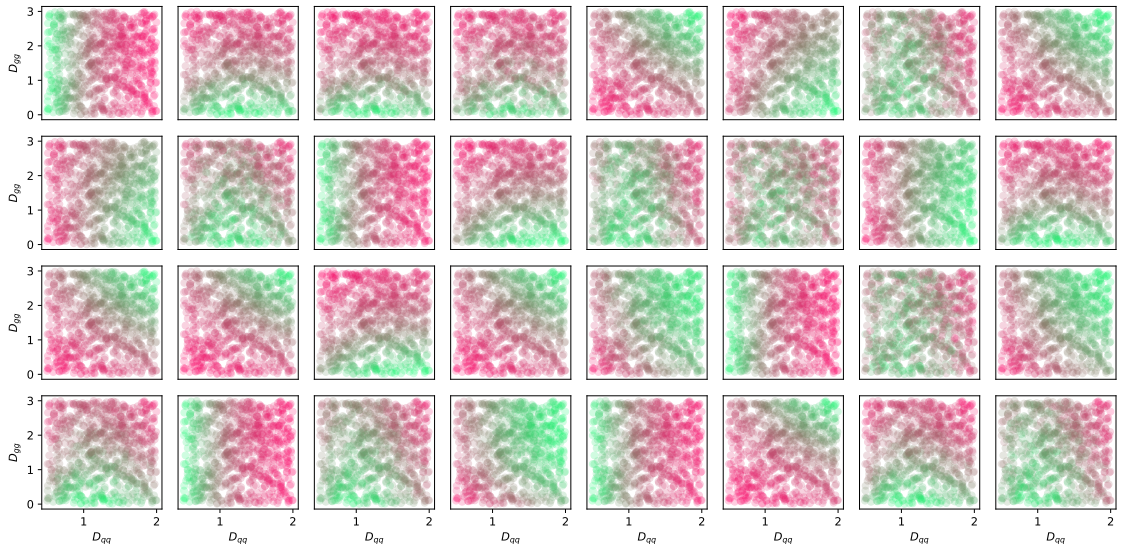


Figure 4.8: Summary statistics for 1000 parameter points with 10000 k_T-sorted jets each. The positions of the point are given by (D_{qq}, D_{gg}). The color indicates the value of the summary statistic from red (minimal) to green (maximal). Plots for all 32 output nodes of the summary network are shown.

The training on only two splitting parameter allows us to visualize the learned summary statistics as a function of D_{qq} and D_{gg} for the test data set with 1000 parameter points drawn from the prior. The result is shown in Fig. 4.8. It can be seen that there is a variety of different summary statistics. None of the summary statistics are dominated by random noise, instead they all show a clear dependence on at least one of the two parameters. Some focus mostly on D_{qq} while others focus on D_{gg} or linear combinations of the two observables. Also, they have the region of their largest sensitivity in different parts of the prior. As expected for the large number of 32 summary statistics, some of the network outputs are very similar. Still, this amount of summary network outputs proved to be beneficial for the training stability and inference performance.

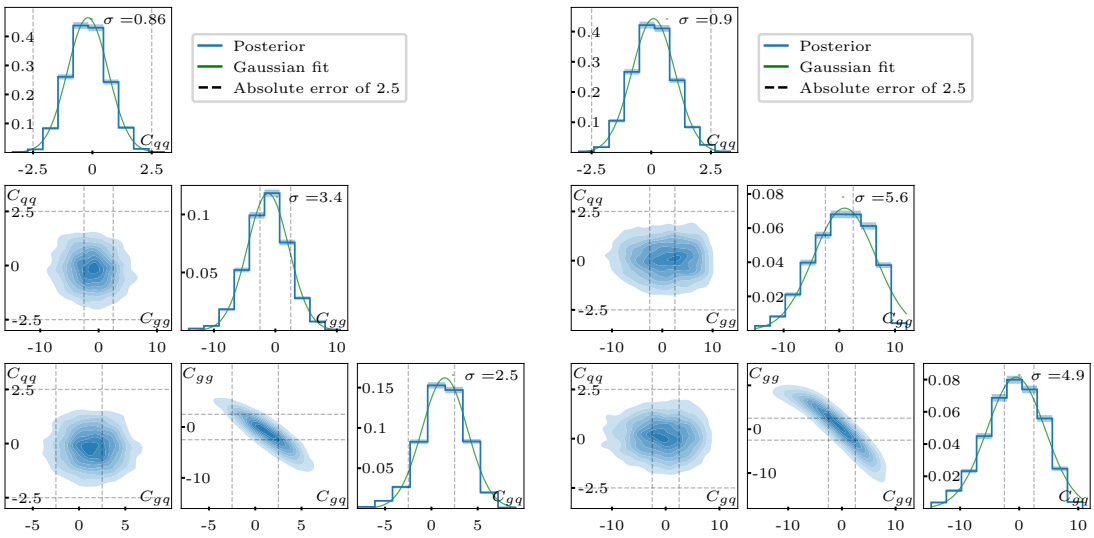
4.5.3 Measuring the rest terms

As the next step, we examine whether we are able to measure p_T-suppressed corrections to all three splitting kernels with our approach. The network is trained with a uniform prior over the intervals

$$C_{qq} \in [-10, 10] \,, \qquad C_{gg} \in [-15, 15] \,, \qquad C_{gq} \in [-15, 15] \,. \tag{4.30}$$

The larger intervals for the second and third parameters are again chosen because the corresponding splittings are less likely to occur than the first splitting. The posteriors for truth- and k_T-sorted data with vanishing rest terms $C_{qq} = C_{gg} = C_{gq} = 0$ are shown in Fig. 4.9. For both sortings, $\sigma(C_{qq})$ is smaller than the corresponding

errors from the trainings with gluon-radiation showers, with $\sigma(C_{qq}) = 0.90$ (0.86) for k_T- (truth-)sorted data in this run and $\sigma(C_{qq}) = 1.00$ (0.97) in the previous run. This could be because of the hierarchical structure of the parameters in the $\{D_{qq}, F_{qq}, C_{qq}\}$ training that the network has to separate, in contrast to the more similar magnitudes of the effects of $\{C_{qq}, C_{gg}, C_{gq}\}$ on their respective splitting functions. As expected, $\sigma(C_{gg})$ and $\sigma(C_{gq})$ are larger than $\sigma(C_{qq})$. However, the large errors in the marginal distributions are partially caused by the strong negative correlation between C_{gg} and C_{gq}. Other than for the gluon-radiation shower where the correlation was caused by the choice of sorting, we observe this correlation for both sorting methods, making it less likely that the correlation is an artifact of our inference method.



(a) Truth sorting (b) k_T sorting

Figure 4.9: Posterior probabilities of the rest term parameters $\{C_{qq}, C_{gg}, C_{gq}\}$ for 10000 SM-like toy shower jets for truth-sorted and k_T-sorted constituents.

Like for the gluon-radiation shower, we also train the network on high-level observables with the same architecture and settings. The comparison between the posteriors for high-level observables and k_T-sorted and truth-sorted low-level observables is shown in Fig. 4.10. Again, the results are clearly the best for the truth-sorted data and the k_T-sorted data performs better than the high-level observables. However, the performance gain from using the k_T sorting over the high-level observables is not as high as for the gluon-radiation shower. The high-level observables posterior for C_{gg} deviates from a Gaussian, indicating that the trainings are not as stable as for low-level observables.

4.5.4 Further results

We have only presented results for data sets where two or three splitting parameters were varied. We also tried to vary more parameters in a single training, for

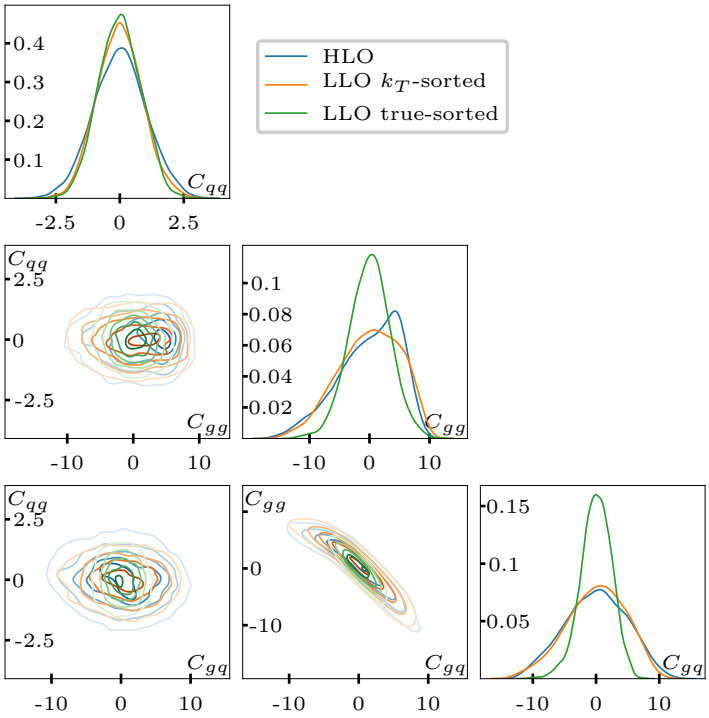


Figure 4.10: Posterior probabilities of the rest term parameters $\{C_{qq}, C_{gg}, C_{gq}\}$ for 10000 SM-like toy shower jets for a network trained on high-level observables and the networks trained on k_T- and truth-sorted low-level observables.

example by combining the finite parameters with the rest terms, with five parameters $\{C_{qq}, F_{qq}, C_{gg}, F_{gg}, C_{gq}\}$ being varied. However, the training stability decreased tremendously and the network was not even able to infer a sub-set of the parameters with the same quality that we saw in runs with lower numbers of parameters. While there are examples of BayesFlow being used with higher-dimensional model parameters [77], in these cases, the model predictions for the inferred parameters are often more important than the values of the parameters themselves. This does not apply to our physics use-case.

Another limitation of the BayesFlow method is that the maximal number of jets included in one measurement is limited by the maximal number of jets per parameter point during the training. We examined if we could do a Bayesian update by using the posterior extracted from one set of jets as the prior distribution in a second training, such that the inference could be repeated for a second set of jets, giving us the combined posterior. While it is in principle possible to use BayesFlow that way, the training is no longer amortized, making the inference method computationally very expensive and rendering it unusable for practical applications.

5 Hadronization and detector effects

So far, we discussed the results for idealized data from our toy shower generator and found them to be promising. However, this data is not realistic in multiple ways. Firstly, instead of simulating parton showers starting from the two quarks generated in the hard process, we only let one of those quarks undergo splittings. In a more realistic setup, we would get two jets that we would need to reconstruct using a jet algorithm. Secondly, we trained the network directly on the partons of the parton shower and in the case of truth-sorted data, the network even had additional information about the order of the splittings. Of course, we cannot measure this order in reality and because of confinement, we cannot directly measure the four-momenta of the partons either. Instead, we have to take hadronization effects into account. Lastly, to actually measure events we need a detector. This means that add a detector simulation step to our Monte Carlo event generation chain. Our more realistic generator setup and the resulting trainings will be discussed in this chapter.

5.1 Event generation and training setup

For our more realistic event generation setup, we use a modified version of Sherpa 2.2.10 [27] that includes our parameterized splitting functions introduced in section 4.1. Sherpa does not make use of the symmetry of the gluon splitting, so all four splitting kernels P_{qq}, $P_{gg,1}$, $P_{gg,2}$, P_{gq} have to be implemented. The hard process is

$$e^+e^- \to q\bar{q}\,, \qquad q = u, d, s \tag{5.1}$$

at $E_{\text{CMS}} = m_Z$ again. Because of the leptonic initial state, we can ignore initial state radiation, and we do not simulate pile-up. The weakly-decaying heavy c and b quarks are not only disabled in the hard process but also in the parton shower generator. In contrast to the toy shower generator, the D parameters are modified with an universal factor accounting for next-to-leading-order corrections to the soft gluon emission that is typically included in standard parton shower generators [78]. The parton shower has a cut-off at 1 GeV after which hadronization is simulated.

Because we want to examine the effects of hadronization alone and of hadronization combined with detector effects, we either use the four-momenta of hadrons, charged leptons and photons, or we pass the Sherpa output on to Delphes 3.4.2 [28] with the default ATLAS card and use the four-momenta of the particle flow objects [52]

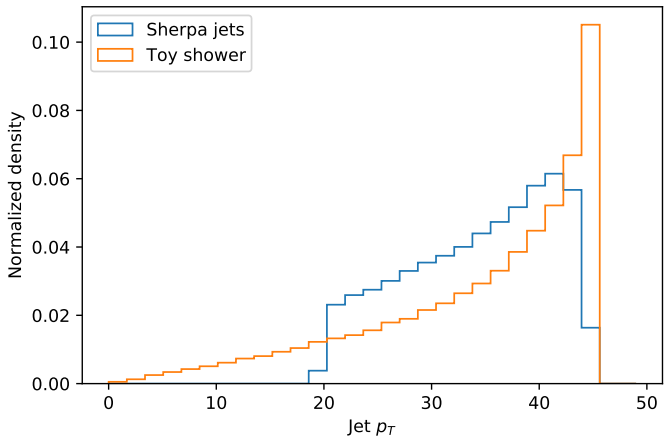


Figure 5.1: Jet p_T distributions of 100000 jets generated with SHERPA and our toy shower at Standard Model parameters.

found by DELPHES. In both cases, we use FASTJET 3.3.4 [79] to find anti-k_T jets [55] with $R = 1.2$ and a lower p_T cut-off of 20 GeV. The constituents of the jet with the largest total p_T are then sorted using our k_T sorting algorithm. We only save the first 13 constituents since the effect of the following softer constituents on the inference performance is small.

Fig. 5.1 shows the p_T distributions of the hadronized SHERPA jets and the jets from our toy shower generator. The maximal p_T is $m_Z/2$ in both distributions. As no lower cut-off is applied to the toy shower jets, the p_T can be close to zero. The toy shower p_T distributions peaks at $m_Z/2$ while the SHERPA jet p_T distribution peaks earlier because not the entire momentum of the hard quark is necessarily included in the jet. These jets are much softer than typical LHC jets and going to higher p_T would likely result in more splitting information being contained in the sub-jet observables. However, in this regime we would also expect a larger effect of the calorimeter energy resolution.

To check the consistency of the SHERPA shower with our toy shower generator, we compared the distributions of high-level observables for full showers without hadronization and jet clustering in SHERPA, and with showering from both hard quarks in the toy shower. This was done for different values of the splitting parameters and we found the distributions to be very similar. Even after comparing jets instead of full events by using FASTJET on the SHERPA events, and starting the shower from one hard quark in the toy shower, the differences between the distributions were still small. This means that our method to efficiently generate single jets in the toy shower generator proved to be a reasonable approximation.

The BAYESFLOW training setup is the same as for the toy shower data. We will again vary two or three parameters at a time for the three different combinations of parameters $\{D_{qq}, F_{qq}, C_{qq}\}$, $\{D_{qq}, D_{gg}\}$ and $\{C_{qq}, C_{gg}, C_{gq}\}$. We use the same priors as for the toy shower. Unlike for the toy shower, we also use a fixed number of jets

per parameter point for the gluon-radiation parameters and we use the full QCD shower instead of deactivating the P_{gg} and P_{gq} splittings. We found that the training setup for variable numbers of jets per parameter point is still working for the SHERPA showers, but due to the computationally costlier generation and training process, we will not use it in the following analyses.

5.2 Effects of hadronization and detector

To show the effects of our splitting parameters on the parton showers and what changes after hadronization and detector effects are taken into account, we pick the parameter for the leading term of the gluon-radiation splitting function D_{qq} and vary it over the interval $[0.5, 2]$. This is the same as the prior interval that we use for the D_{qq} parameter. Fig. 5.2 shows histograms for four of the high-level observables introduced in section 2.6, excluding N_{95} and $x_{\max}$ due to their high correlation with the number of constituents n_{PF}.

The numbers of constituents n_{PF} for the toy shower are typically very low, and in the case of $D_{qq} = 0.5$, the histogram peaks at a single constituent. For higher D_{qq}, the splitting probability increases and therefore the n_{PF} histogram shifts towards higher values. Still, almost all of the showers have less than 10 constituents. The hadronization process strongly increases the number of constituents, resulting in a broad n_{PF} distribution peaking between 14 and 16 depending on the value of D_{qq}. This number is reduced again by the detector because of the limited resolution and very soft constituents not being detected. The detector-level n_{PF} distributions peak between 9 and 11.

The distribution of the girth w_{PF} for the toy shower has a large peak at 0 from jets with a single constituent with a second peak around 0.15 from narrow jets with a higher number of splittings. The distribution then slowly falls towards higher w_{PF}. Showers with a high girth are more likely for higher D_{qq}. For the hadronized jets and after detector simulation, there is no peak from one-constituent jets, so they peak between 0.15 and 0.2 depending on D_{qq}. The difference between the latter two curves are small with a slight shift towards lower w_{PF} for the detector.

The $p_T D$ distribution again looks very different for the toy shower compared to the results after hadronization. This is again caused by the low numbers of splittings. A single hard constituent results in $p_T D = 1$. The values for two, three and four constituents with equal p_T are $\sqrt{2}/2 \approx 0.71$, $\sqrt{3}/3 \approx 0.58$ and $\sqrt{4}/4 = 0.5$. $p_T D$ increases the less uniform the p_T is distributed among the constituents. This manifests itself in the peak structure of the histogram. The histograms for hadronized jets with and without detector effects are again very similar with a broad maximum around 0.4.

Finally, the two-point energy correlator $C_{0.2}$ also has a peak from single-constituent showers at 0 and very broad maximum around $C_{0.2} \approx 0.5$. Again, the shape of

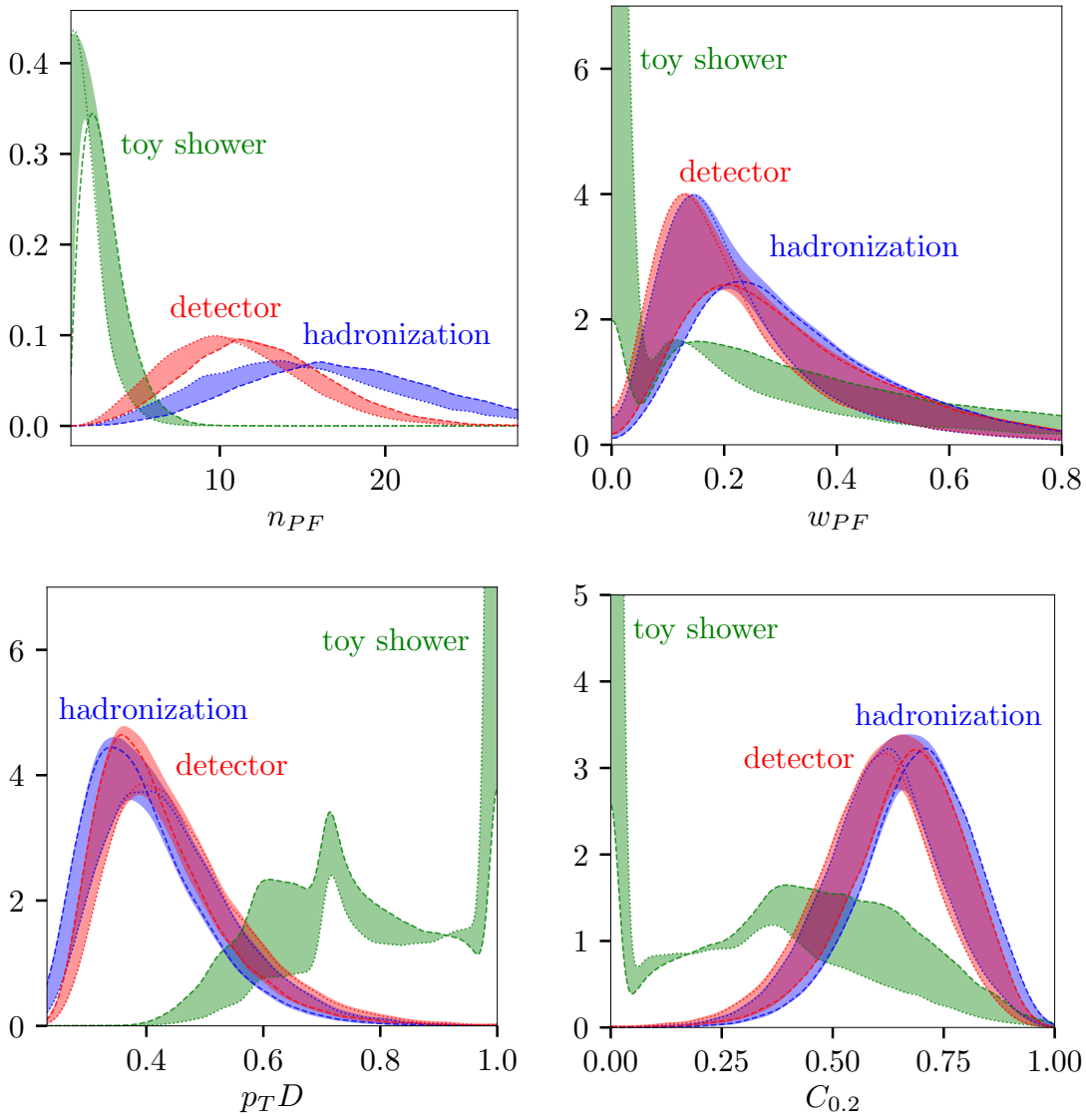


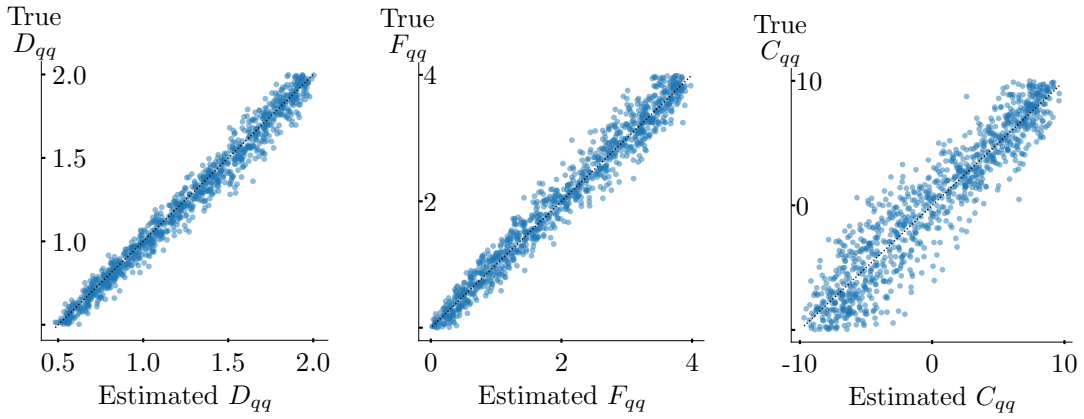
Figure 5.2: Effects of varying the leading gluon-radiation parameter D_{qq} on the high-level observables n_{PF}, w_{PF}, $p_T D$ and $C_{0.2}$ for toy shower jets and SHERPA jets with and without detector effects. The dotted lines show the results for $D_{qq} = 0.5$ and the dashed line show the results for $D_{qq} = 2$. The histograms for parameters between those two values are in the filled areas.

the distribution is less complex for the hadronized jets and the detector effects only cause a slight shift of the histogram.

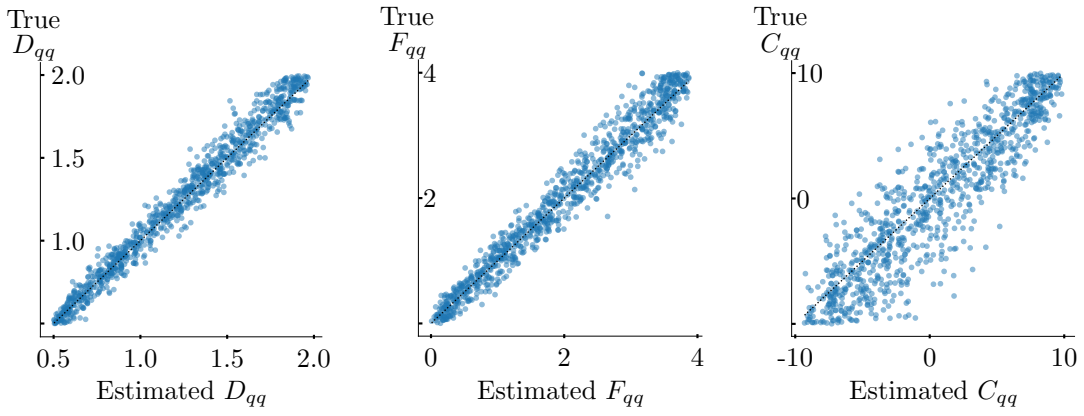
Even when the peaks from single-constituent jets in the toy shower histograms are neglected, it is still valid to say that the hadronization effects are much more important than the detector effects. This is why we first study the effects of hadronization on the inference of shower parameters and then include the detector effects.

5.3 Results

5.3.1 Measuring the gluon-radiation parameters



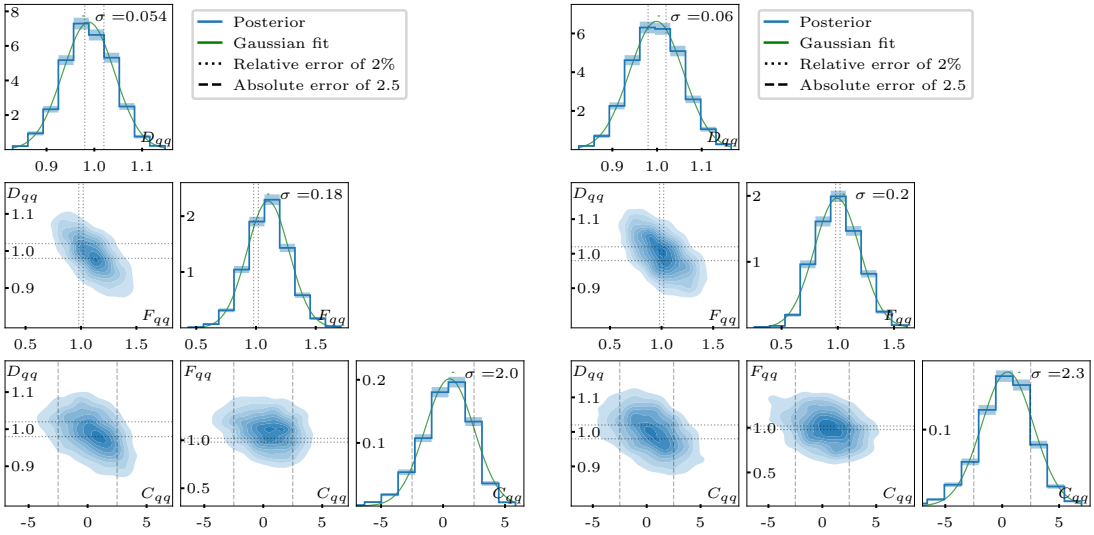
(a) Without detector effects



(b) Detector effects simulated with DELPHES

Figure 5.3: The estimated parameters plotted against the true parameters $\{D_{qq}, F_{qq}, C_{qq}\}$ for 1000 points randomly drawn from the prior for trainings on SHERPA jets with and without detector effects. The closer the points are to the dotted line, the better was the recovery of the true parameters.

For the measurement of the three hierarchical gluon-radiation parameters $\{D_{qq}, F_{qq}, C_{qq}\}$, we generate the full QCD shower in SHERPA instead of looking at a pure gluon-radiation shower like in the previous chapter. Fig. 5.3 shows the inference performance over the whole prior with and without detector effects. Already from this plot, it is apparent that the measurement uncertainties are much larger than for the toy shower. Moreover, the plots for C_{qq} show that there is a larger region around the boundaries of the prior interval where the parameter estimates are biased.



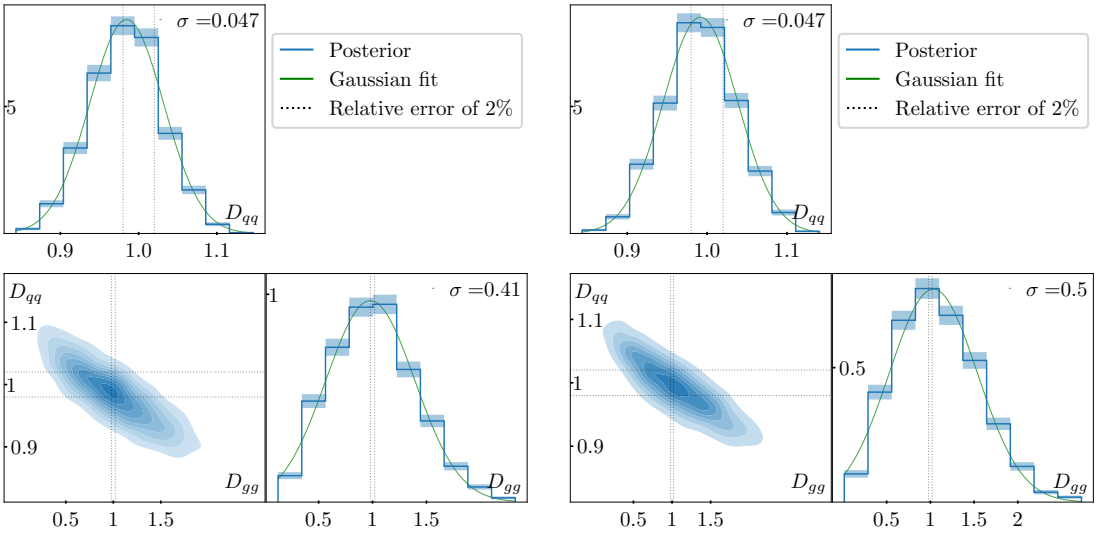
(a) Without detector effects (b) With detector effects

Figure 5.4: Posterior probabilities of the gluon-radiation parameters $\{D_{qq}, F_{qq}, C_{qq}\}$ for 10000 SM-like SHERPA jets with and without detector simulation.

Fig. 5.4 shows the posteriors for 10000 SM-like jets with and without detector effects. It can be seen that compared to the results for the k_T-sorted toy shower, the estimated uncertainty for the hadronized jets is slightly larger for the finite parameter F_{qq}, larger by a factor of ~ 1.9 for the rest term C_{qq} and larger by a factor of ~ 2.8 for the leading term. However, the correlations between the parameter are not stronger than in the toy shower. The uncertainties increase only slightly when detector effects are applied. This matches our expectations from the previous section where we stated that the hadronization effects are quantitatively more important than the detector effects.

5.3.2 Measuring the leading terms

Next, we look at the leading parameters D_{qq} and D_{gg}, approximately corresponding to a measurement of the QCD color factors C_F and C_A. Fig. 5.5 shows the posterior distributions for jets with and without detector effects. Like in the case of k_T-sorted toy showers, D_{qq} has a lower estimated uncertainty than in the training with the three gluon-radiation parameters. The detector effects make no significant



(a) Without detector effects

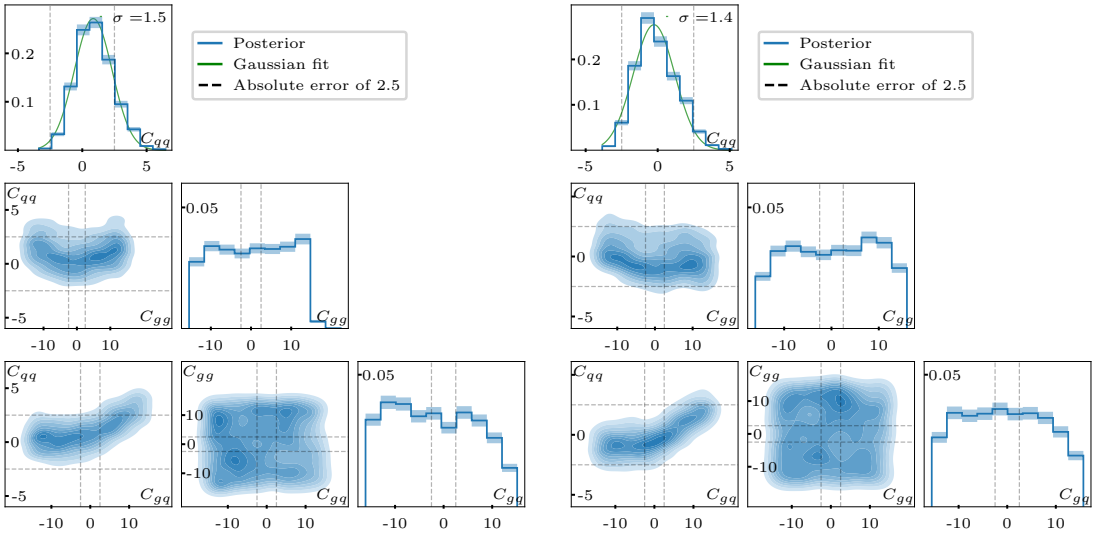
(b) With detector effects

Figure 5.5: Posterior probabilities of the leading parameters $\{D_{qq}, D_{gg}\}$ for 10000 SM-like SHERPA jets with and without detector simulation.

difference. However, the uncertainty of the leading term for the triple-gluon splitting explodes by a factor of more than 12 compared to the toy shower results. Also, the two parameters are now strongly anti-correlated, contributing to the larger width of the marginal distributions. As we are training on quark-initiated jets, the gluon splittings are already much more rare than the quark splittings. Consequently it is harder for the network to extract this information. The likely reason for the large drop in performance going to hadronized results is the much larger number of jet constituents (see Fig. 5.2). This makes it even harder for the network since the four-momenta containing relevant information about the gluon splittings can appear in many different positions in the list of input four-momenta. Comparing the results for the full simulation chain with the results for the QCD Casimirs from LEP (see section 2.2.3), we can see that the relative error of $\approx 5\%$ for D_{qq} for only 10000 jets is close to the error of the combined LEP result for C_F, while our relative error for D_{gg} is much larger than LEP measurement uncertainty for C_A.

5.3.3 Measuring the rest terms

Lastly, we examine the results for the three rest term parameters C_{qq}, C_{gg} and C_{gq}, shown in Fig. 5.6. The uncertainty of the parameter C_{qq} for the gluon-radiation is roughly a factor two larger than the result for the k_T-sorted toy shower. The uncertainty is lower than in the training on the gluon-radiation shower parameters, again indicating that separating the three hierarchical terms is difficult for the network. The distributions for C_{gg} and C_{gq} are very similar to their respective prior distributions. That means that the network was not able to extract any information about these parameters from the data. Therefore, we give no numerical value for



(a) Without detector effects (b) With detector effects

Figure 5.6: Posterior probabilities of the rest term parameters $\{C_{qq}, C_{gg}, C_{gq}\}$ for 10000 SM-like SHERPA jets with and without detector simulation.

the standard deviation because it would only reflect our choice of the prior distribution. Choosing a larger prior is not a reasonable option for the C parameters because the rest terms would no longer be small, and the hierarchical nature of the parameters D, F and C would be lost. The drop in performance is again caused by the training on quark-initiated jets. As the estimated errors were already very large for the leading parameter D_{gg}, it is not a surprising result.

6 Summary and outlook

In this thesis, we proposed a new method to measure the fundamental properties of QCD splitting functions using machine-learning techniques. First, we introduced a general parameterization of the splitting functions with parameters for the divergent and finite terms and introduced a p_T-suppressed rest term to account for contributions beyond the soft-collinear approximation. Since the splitting functions are dominated by the divergent terms, measuring the parameters for those terms approximately corresponds to measurements of the QCD color factors C_A and C_F. We implemented a toy shower generator that allows us to vary these parameters. The generated parton four-momenta were then used to train a conditional invertible neural network (cINN) and a summary network according to the BayesFlow method. This allowed us to estimate posterior distributions for the splitting parameters from sets of many jets.

We looked at two different sortings of the generated parton four-momenta as input for the summary network. Firstly, truth-sorted partons still contained information about the order in which the splittings were generated due to the way we implemented the toy shower generator. Secondly, we introduced the k_T sorting strategy based on the k_T algorithm to approximate the thruth sorting. As a baseline, we also trained the network on six established high-level jet observables. We looked at three combinations of parameters to test our method: The three parameters of a shower with only gluon-radiation, the two leading parameters of the gluon-radiation and triple-gluon splittings and the three rest term parameters. In all three cases, our network was able to measure the parameters. As expected, we got the best results for truth-sorted constituents, but the measurement uncertainties for k_T-sorted constituents were still significantly smaller than for trainings on high-level observables. Also, we were able to demonstrate that the estimated posterior standard deviations have the expected inverse square root scaling behaviour with the number of measured jets.

For a more realistic analysis including hadronization effects, we modified the Sherpa parton shower generator to include our splitting parameterization. Furthermore, we used Delphes with an ATLAS card to also take detector effects into account. We showed that our network was still able to extract the gluon-radiation parameters. The relative uncertainty of our result was in a similar order of magnitude as the error of the LEP measurement of C_F, even though we only used a very small number of jets in our measurement. Also, we were able to measure the rest term parameter of this splitting. These are very encouraging results, indicating that our method could be used to improve measurements of the QCD casimirs and that even testing higher-order contributions might be within reach. Estimating the parameters of the

gluon splitting function P_{gg} and P_{gq} proved to be more challenging as we trained the network on quark-initiated showers, making these splittings more rare. Our numerical results for the posterior standard deviations from all the trainings discussed in this thesis can be found in Table 6.1.

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 6.1: Estimated posterior standard deviations for 10000 SM-like jets for all the trainings discussed in this thesis. This includes the results for the gluon-radiation parameters, leading parameters and rest term parameters. For the toy shower, the results for truth-sorted and k_T-sorted low-level observables and for high-level observables are shown. For the hadronized Sherpa showers, the results with and without detector simulation are shown. The asterisks denotes non-Gaussian posterior distributions. The values in parentheses are the estimated errors when only one parameter is varied. They are calculated by taking a slice of the posterior distribution at the Standard Model value instead of using the marginal distribution.

Our approach proved to be a promising first step in the direction of a systematic study of QCD splitting properties, opening many avenues for future research. Using gluon-initiated jets instead of or mixed with quark-initiated jets could help to improve the inference performance for the parameters of the P_{gg} and P_{gq} splitting kernels where the splitting particle is a gluon. The next step would then be to go from the electron-positron scattering process at $E_{\text{CMS}} = m_Z$ to a LHC scenario with harder jets. This would increase the number of splittings and therefore also the effect of the splitting kernel parameters, but at the same time, the calorimeter energy resolution would have a larger influence on the results. Also, for a more realistic LHC simulation, effects of pile-up and initial state radiation would have to be included. After enhancing the simulations like that, the final step would be to test the performance of our method on actual experimental data.

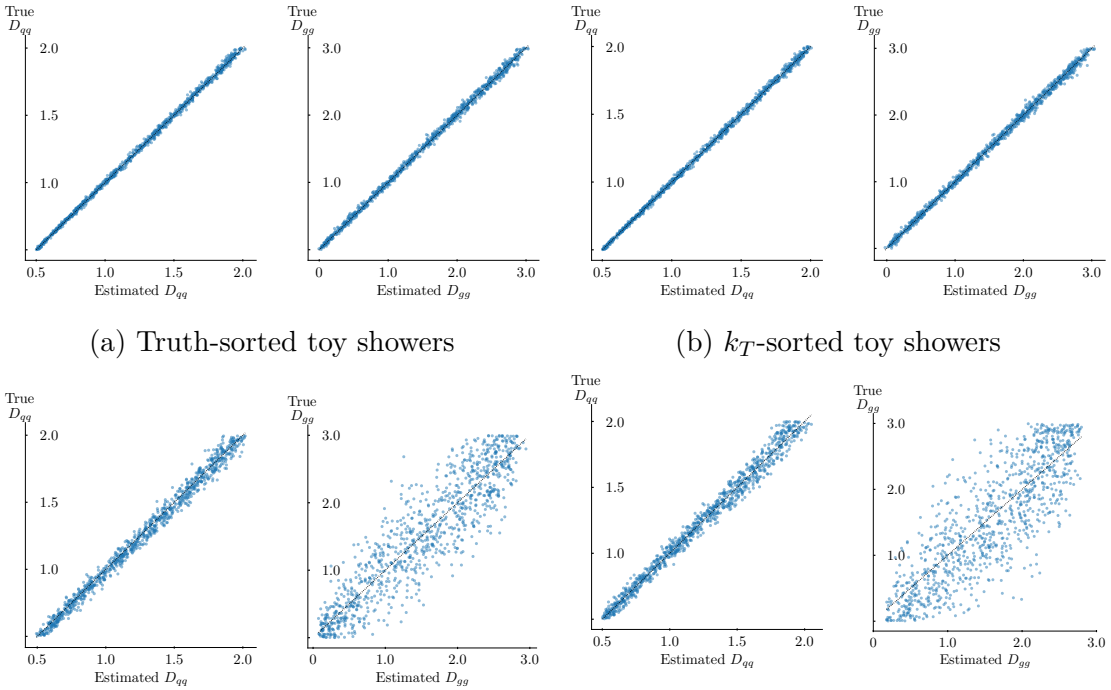
One of the main problems of the BayesFlow method is that it does not scale well for higher numbers of jets. This is not an issue when conventional likelihood methods are used. The maximum number of jets for which the posterior can be extracted is determined by the maximum number of jets per parameter point during the training. We used up to 10^5 jets per data point, however in a typical LHC analysis one would like to analyze millions of jets. Splitting up the data, doing the inference in multiple

batches and combining the results afterwards would require a better understanding of the involved systematic uncertainties.

Finally, one of the most interesting features of the BayesFlow network is that it learns to calculate jet summary statistics in an unsupervised way. A closer examination of those summary statistics regarding their interpretability might be a way to develop high-level observables that improve the extraction of the information of interest from the data.

A Additional plots

In the discussion of the inference results, we have focused on the network performance at a single parameter point. However, it is important to check that the estimated posterior distributions are unbiased over the whole prior of the trainings. This can be done using simulation-based calibration or by looking at the estimated posterior means against the true parameters for several points in the training dataset. For the measurements of $\{D_{qq}, D_{gg}\}$ and $\{C_{qq}, C_{gg}, C_{gq}\}$, these plots are shown in Fig. A.1 and A.2.



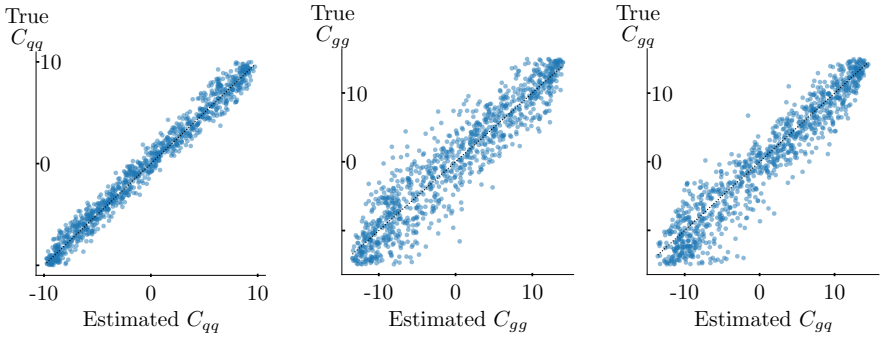
(a) Truth-sorted toy showers

(b) k_T-sorted toy showers

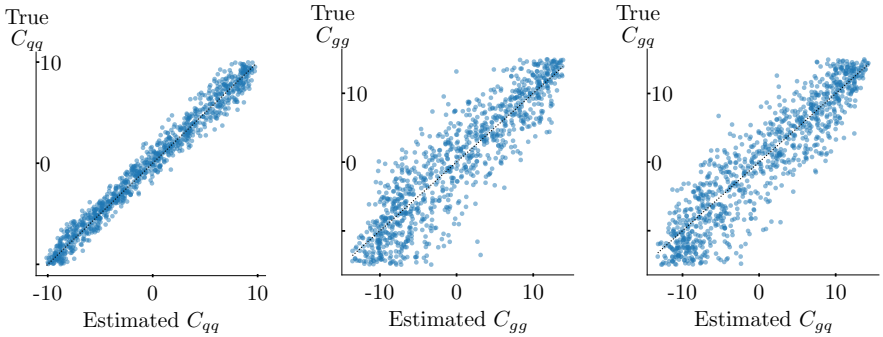
(c) Hadronized Sherpa showers

(d) Hadronized Sherpa showers with detector effects

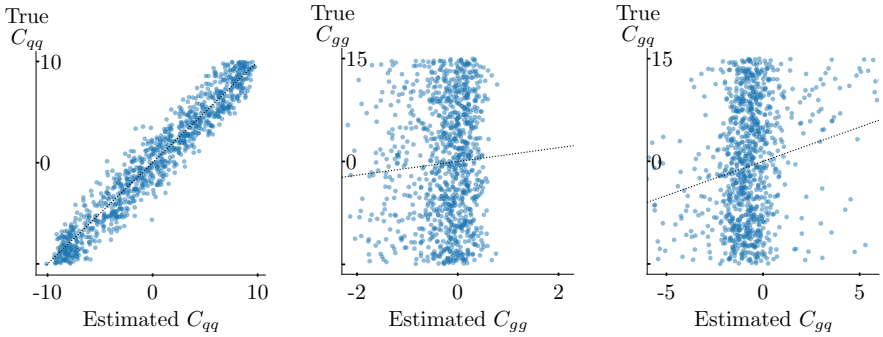
Figure A.1: The estimated parameters plotted against the true parameters $\{D_{qq}, D_{gg}\}$ for 1000 points randomly drawn from the prior for trainings on our four different data sets. The closer the points are to the dotted line, the better was the recovery of the true parameters.



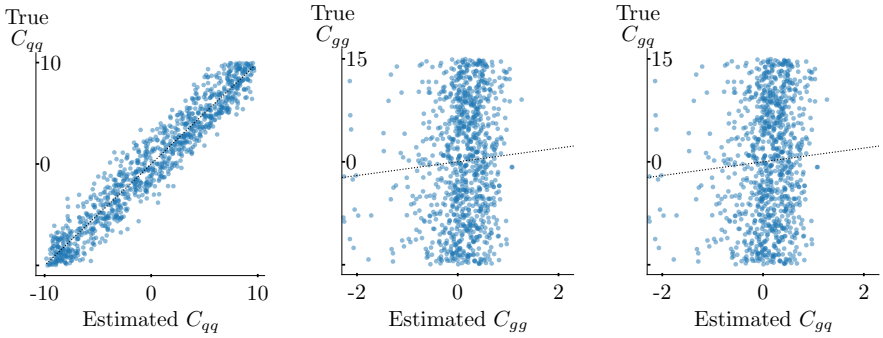
(a) Truth-sorted toy showers



(b) k_T-sorted toy showers



(c) Hadronized Sherpa showers



(d) Hadronized Sherpa showers with detector effects

Figure A.2: The estimated parameters plotted against the true parameters $\{C_{qq}, C_{gg}, C_{gq}\}$ for 1000 points randomly drawn from the prior for trainings on our four different data sets. The closer the points are to the dotted line, the better was the recovery of the true parameters.

B Lists

B.1 List of Figures

B.2 List of Tables

C Bibliography

[1] G. Kasieczka et al. *The machine learning landscape of top taggers*. 2019. DOI: 10.21468/scipostphys.7.1.014. arXiv: 1902.09914.

[2] Benjamin Nachman. "Anomaly Detection for Physics Analysis and Less than Supervised Learning." In: (2020). arXiv: 2010.14554.

[3] Johann Brehmer and Kyle Cranmer. *Simulation-based inference methods for particle physics*. 2020. arXiv: 2010.06439.

[4] Anja Butter and Tilman Plehn. *Generative networks for LHC events*. 2020. arXiv: 2008.08558.

[5] S. Kluth. "Jet physics in e+e- annihilation from 14 to 209 GeV." In: *Nuclear Physics B - Proceedings Supplements* 133.1-3 SUPPL. (2004), pp. 36–46. ISSN: 09205632. DOI: 10.1016/j.nuclphysbps.2004.04.134. arXiv: 0309070 [hep-ex].

[6] Stefan Kluth. "Tests of quantum chromo dynamics at e+e- colliders." In: *Reports on Progress in Physics* 69.6 (2006), pp. 1771–1846. ISSN: 00344885. DOI: 10.1088/0034-4885/69/6/R04. arXiv: 0603011v2 [arXiv:hep-ex].

[7] L. Hartgring, E. Laenen, and P. Skands. "Antenna Showers with One-Loop Matrix Elements." In: *JHEP* 10 (2013), p. 127. DOI: 10.1007/JHEP10(2013)127. arXiv: 1303.4974 [hep-ph].

[8] Hai Tao Li and Peter Skands. "A framework for second-order parton showers." In: *Phys. Lett. B* 771 (2017), pp. 59–66. DOI: 10.1016/j.physletb.2017.05.011. arXiv: 1611.00013 [hep-ph].

[9] Stefan Höche, Frank Krauss, and Stefan Prestel. "Implementing NLO DGLAP evolution in Parton Showers." In: *JHEP* 10 (2017), p. 093. DOI: 10.1007/JHEP10(2017)093. arXiv: 1705.00982 [hep-ph].

[10] Falko Dulat, Stefan Höche, and Stefan Prestel. "Leading-Color Fully Differential Two-Loop Soft Corrections to QCD Dipole Showers." In: *Phys. Rev. D* 98.7 (2018), p. 074013. DOI: 10.1103/PhysRevD.98.074013. arXiv: 1805.03757 [hep-ph].

[11] Mrinal Dasgupta et al. "Parton showers beyond leading logarithmic accuracy." In: *Phys. Rev. Lett.* 125.5 (2020), p. 052002. DOI: 10.1103/PhysRevLett.125.052002. arXiv: 2002.11114 [hep-ph].

[12] Mrinal Dasgupta et al. "Logarithmic accuracy of parton showers: a fixed-order study." In: *JHEP* 09 (2018). [Erratum: JHEP 03, 083 (2020)], p. 033. DOI: 10.1007/JHEP09(2018)033. arXiv: 1805.09327 [hep-ph].

[13] Danilo Jimenez Rezende and Shakir Mohamed. "Variational inference with normalizing flows." In: *32nd International Conference on Machine Learning, ICML 2015* 2 (2015), pp. 1530–1538. arXiv: 1505.05770.

[14] Laurent Dinh, David Krueger, and Yoshua Bengio. "NICE: Non-linear independent components estimation." In: *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings* 1.2 (2015), pp. 1–13. arXiv: 1410.8516.

[15] 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* 8828.c (2020), pp. 1–1. ISSN: 0162-8828. DOI: 10.1109/tpami.2020.2992934. arXiv: 1908.09257.

[16] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using real NVP." In: *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings* (2017). arXiv: 1605.08803.

[17] Lynton Ardizzone et al. "Guided Image Generation with Conditional Invertible Neural Networks." In: (2019). arXiv: 1907.02392.

[18] Bob Stienen and Rob Verheyen. *Phase space sampling and inference from weighted events with autoregressive flows*. 2020. arXiv: 2011.13445.

[19] I Chen, Matthew D Klimek, and M Perelstein. *Improved neural network Monte Carlo simulation*. 2020. arXiv: 2009.07819.

[20] Christina Gao et al. "Event generation with normalizing flows." In: *Physical Review D* 101.7 (2020), pp. 1–9. ISSN: 24700029. DOI: 10.1103/PhysRevD.101.076002. arXiv: 2001.10028.

[21] Christina Gao, Joshua Isaacson, and Claudius Krause. *i-flow: High-dimensional Integration and Sampling with Normalizing Flows*. 2020. DOI: 10.1088/2632-2153/abab62. arXiv: 2001.05486.

[22] Enrico Bothmann et al. "Exploring phase space with neural importance sampling." In: *SciPost Physics* 8.4 (2020), pp. 1–20. ISSN: 25424653. DOI: 10.21468/SciPostPhys.8.4.069. arXiv: 2001.05478.

[23] Marco Bellagente et al. "Invertible Networks or Partons to Detector and Back Again." In: (2020), pp. 1–24. arXiv: 2006.06685.

[24] Johann Brehmer and Kyle Cranmer. *Flows for simultaneous manifold learning and density estimation*. 2020. arXiv: 2003.13913.

[25] Benjamin Nachman and David Shih. "Anomaly detection with density estimation." In: *Physical Review D* 101.7 (2020). ISSN: 24700029. DOI: 10.1103/PhysRevD.101.075042. arXiv: 2001.04990.

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

[27] Enrico Bothmann et al. "Event generation with Sherpa 2.2." In: *SciPost Physics* 7.3 (2019). ISSN: 2542-4653. DOI: 10.21468/scipostphys.7.3.034. arXiv: 1905.09127.

[28] 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 (2014). ISSN: 10298479. DOI: 10.1007/JHEP02(2014)057. arXiv: 1307.6346.

[29] Sebastian Bieringer et al. “Measuring QCD Splittings with Invertible Networks.” In: (2020), pp. 1–24. arXiv: 2012.09873.

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

[31] Matthew D. Schwartz. *Quantum Field Theory and the Standard Model*. Cambridge University Press, Mar. 2014. ISBN: 978-1-107-03473-0, 978-1-107-03473-0.

[32] Georges Aad et al. “Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC.” In: *Phys. Lett.* B716 (2012), pp. 1–29. DOI: 10.1016/j.physletb.2012.08.020. arXiv: 1207.7214 [hep-ex].

[33] Serguei Chatrchyan et al. “Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC.” In: *Phys. Lett.* B716 (2012), pp. 30–61. DOI: 10.1016/j.physletb.2012.08.021. arXiv: 1207.7235 [hep-ex].

[34] G. Abbiendi et al. “Particle multiplicity of unbiased gluon jets from e+e - three-jet events.” In: *European Physical Journal C* 23.4 (2002), pp. 597–613. ISSN: 14346052. DOI: 10.1007/s100520200926.

[35] P. Abreu et al. “Measurement of the gluon fragmentation function and a comparison of the scaling violation in gluon and quark jets.” In: *The European Physical Journal C* 13.4 (2000), p. 573. ISSN: 14346044. DOI: 10.1007/s100520050719.

[36] A. Heister et al. “The ALEPH collaboration: Measurements of the strong coupling constant and the QCD colour factors using four-jet observables from hadronic Z decays.” In: *European Physical Journal C* 27.1 (2003), pp. 1–17. ISSN: 14346044. DOI: 10.1140/epjc/s2002-01114-2.

[37] The OPAL collaboration and G. Abbiendi. “A Simultaneous Measurement of the QCD Colour Factors and the Strong Coupling.” In: 25.3 (2001), pp. 1–29. DOI: 10.1007/s100520100699. arXiv: 0101044 [hep-ex].

[38] S. Kluth et al. “A measurement of the QCD colour factors using event shape distributions at $\sqrt{s} = 14$ to 189 GeV.” In: *European Physical Journal C* 21.2 (2001), pp. 199–210. ISSN: 14346044. DOI: 10.1007/s100520100742. arXiv: 0012044 [hep-ex].

[39] S. Brandt et al. “The principal axis of jets - an attempt to analyse high-energy collisions as two-body processes.” In: *Physics Letters* 12.1 (Sept. 1964), pp. 57–61. ISSN: 00319163. DOI: 10.1016/0031-9163(64)91176-X.

[40] Edward Farhi. “Quantum Chromodynamics Test for Jets.” In: *Physical Review Letters* 39.25 (1977), pp. 1587–1588. ISSN: 00319007. DOI: 10.1103/PhysRevLett.39.1587.

[41] G. Parisi. "Superinclusive cross sections." In: *Physics Letters B* 74.1-2 (1978), pp. 65–67. ISSN: 03702693. DOI: 10.1016/0370-2693(78)90061-8.

[42] John F. Donoghue, F. E. Low, and So-Young Pi. "Tensor analysis of hadronic jets in quantum chromodynamics." In: *Physical Review D* 20.11 (1979), pp. 2759–2762. DOI: 10.1103/PhysRevD.20.2759.

[43] S Catani, G Turnock, and B R Webber. "Jet broadening measures in e+e- annihilation." In: *Physics Letters B* 295.3-4 (1992), pp. 269–276. ISSN: 03702693. DOI: 10.1016/0370-2693(92)91565-Q.

[44] Tilman Plehn. "Lectures on LHC physics." In: *Lecture Notes in Physics* 886 (2015), pp. 1–340. ISSN: 00758450. DOI: 10.1007/978-3-642-24040-9. arXiv: 0910.4182.

[45] G. Altarelli and G. Parisi. "Asymptotic freedom in parton language." In: *Nuclear Physics, Section B* 126.2 (1977), pp. 298–318. ISSN: 05503213. DOI: 10.1016/0550-3213(77)90384-4.

[46] Yuri L. Dokshitzer. "Calculation of the Structure Functions for Deep Inelastic Scattering and e+ e- Annihilation by Perturbation Theory in Quantum Chromodynamics." In: *Sov. Phys. JETP* 46 (1977), pp. 641–653.

[47] V. N. Gribov and L. N. Lipatov. "Deep inelastic e p scattering in perturbation theory." In: *Sov. J. Nucl. Phys.* 15 (1972), pp. 438–450.

[48] 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. DOI: 10.1016/S0550-3213(96)00589-5. arXiv: 9605323 [hep-ph].

[49] Stefan Höche. "Introduction to Parton-Shower Event Generators." In: (2015), pp. 235–295. DOI: 10.1142/9789814678766_0005. arXiv: 1411.4085.

[50] Torbjörn Sjöstrand, Stephen Mrenna, and Peter Skands. *PYTHIA 6.4 physics and manual*. 2006. DOI: 10.1088/1126-6708/2006/05/026. arXiv: 0603175 [hep-ph].

[51] B. Andersson et al. "Parton fragmentation and string dynamics." In: *Physics Reports* 97.2-3 (1983), pp. 31–145. ISSN: 03701573. DOI: 10.1016/0370-1573(83)90080-7.

[52] Florian Beaudette. "The CMS Particle Flow Algorithm." In: *EPJ Web of Conferences* 191 (2018). ISSN: 2100014X. DOI: 10.1051/epjconf/201819102016. arXiv: 1401.8155.

[53] ATLAS Collaboration. "Energy Linearity and Resolution of the ATLAS Electromagnetic Barrel Calorimeter in an Electron Test-Beam." In: November (2006). DOI: 10.1016/j.nima.2006.07.053. arXiv: 0608012 [physics].

[54] Stephen D. Ellis and Davison E. Soper. "Successive combination jet algorithm for hadron collisions." In: *Physical Review D* 48.7 (1993), pp. 3160–3166. ISSN: 05562821. DOI: 10.1103/PhysRevD.48.3160. arXiv: 9305266 [hep-ph].

[55] Matteo Cacciari, Gavin P Salam, and Gregory Soyez. "The anti-k t jet clustering algorithm." In: *Journal of High Energy Physics* 2008.4 (2008), pp. 1–12. ISSN: 11266708. DOI: 10.1088/1126-6708/2008/04/063. arXiv: 0802.1189.

[56] 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.

[57] Christopher Frye et al. "Casimir meets Poisson: improved quark/gluon discrimination with counting observables." In: *Journal of High Energy Physics* 2017.9 (2017). ISSN: 10298479. DOI: 10.1007/JHEP09(2017)083. arXiv: 1704.06266.

[58] Andrew J. Larkoski, Gavin P. Salam, and Jesse Thaler. "Energy correlation functions for jet substructure." In: *Journal of High Energy Physics* 2013.6 (2013). ISSN: 11266708. DOI: 10.1007/JHEP06(2013)108. arXiv: 1305.0007.

[59] CMS Collaboration. "Performance of quark/gluon discrimination in 8 TeV pp data." In: (2013).

[60] Jason Gallicchio et al. "Multivariate discrimination and the Higgs+W/Z search." In: *Journal of High Energy Physics* 2011.4 (2011). ISSN: 10298479. DOI: 10.1007/JHEP04(2011)069. arXiv: 1010.3698.

[61] Jon Pumplin. "How to tell quark jets from gluon jets." In: *Physical Review D* 44.7 (1991), pp. 2025–2032. ISSN: 05562821. DOI: 10.1103/PhysRevD.44.2025.

[62] John C. Duchi, Peter L. Bartlett, and Martin J. Wainwright. "Randomized smoothing for (parallel) stochastic optimization." In: *Proceedings of the IEEE Conference on Decision and Control* 12 (2012), pp. 5442–5444. ISSN: 01912216. DOI: 10.1109/CDC.2012.6426698.

[63] 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), pp. 1–15. arXiv: 1412.6980.

[64] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." In: *Journal of Machine Learning Research* 9 (2010), pp. 249–256. ISSN: 15324435.

[65] Donald B. Rubin. "Bayesianly Justifiable and Relevant Frequency Calculations for the Applied Statistician." In: *Ann. Statist.* 12.4 (Dec. 1984), pp. 1151–1172. DOI: 10.1214/aos/1176346785.

[66] Mark A. Beaumont, Wenyang Zhang, and David J. Balding. "Approximate Bayesian computation in population genetics." In: *Genetics* 162.4 (2002), pp. 2025–2035. ISSN: 00166731. DOI: 10.1111/j.1937-2817.2010.tb01236.x.

[67] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. "The frontier of simulation-based inference." In: *Proceedings of the National Academy of Sciences* (2020), p. 201912789. ISSN: 0027-8424. DOI: 10.1073/pnas.1912789117. arXiv: 1911.01429.

[68] Diederik P. Kingma and Prafulla Dhariwal. "Glow: Generative flow with invertible 1×1 convolutions." In: *Advances in Neural Information Processing Systems* 2018-December (2018), pp. 10215–10224. ISSN: 10495258. arXiv: 1807.03039.

[69] 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.

[70] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with LSTM." In: *Neural Computation* 12.10 (2000), pp. 2451–2471. ISSN: 08997667. DOI: 10.1162/089976600300015015.

[71] Sean Talts et al. "Validating Bayesian Inference Algorithms with Simulation-Based Calibration." In: (2018), pp. 1–19. arXiv: arXiv:1804.06788v2.

[72] Stefan Höche. "Introduction to Parton Showers and Matching - Tutorial for Summer Schools." In.

[73] Ashish Agarwal et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems." In: (2015). arXiv: arXiv:1603.04467v2.

[74] Stefan Radev. *BayesFlow source code*. 2020. URL: https://github.com/stefanradev93/BayesFlow.

[75] Theo Heimel et al. "QCD or what?" In: *arXiv* (2018), pp. 1–21. ISSN: 23318422. DOI: 10.21468/scipostphys.6.3.030. arXiv: 1808.08979.

[76] Gilles Louppe et al. "QCD-aware recursive neural networks for jet physics." In: *Journal of High Energy Physics* 2019.1 (2019). ISSN: 10298479. DOI: 10.1007/JHEP01(2019)057. arXiv: arXiv:1702.00748v2.

[77] Stefan T Radev et al. "Model-based Bayesian inference of disease outbreak dynamics with invertible neural networks An application to the COVID-19 pandemics in Germany." In: (2020). arXiv: arXiv:2010.00300v3.

[78] S. Catani, B. R. Webber, and G. Marchesini. "QCD coherent branching and semiinclusive processes at large x." In: *Nucl. Phys. B* 349 (1991), pp. 635–654. DOI: 10.1016/0550-3213(91)90390-J.

[79] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. "FastJet user manual." In: *The European Physical Journal C* 72.3 (2012), pp. 1–69. ISSN: 1434-6044. DOI: 10.1140/epjc/s10052-012-1896-2. arXiv: 1111.6097.

Acknowledgements

First of all, I would like to thank Tilman Plehn for the opportunity to work on this project and for his continuous supervision that made this project a success in spite of the difficulties caused by the Corona pandemic, like social distancing and working from home.

I want to thank Sebastian Bieringer for the fun and productive collaboration. We were a great team and I really enjoyed our daily discussions!

Moreover, I want to thank Ullrich Köthe and especially Stefan Radev for their quick and helpful support on taming the BayesFlow networks. I am grateful to Stefan Höche for his help with all the parton-shower-related aspects of the project, including the parameterization of the splitting kernels, the parton shower simulators and the k_T sorting algorithm.

Finally, I would like to thank Lara Grabitz and Sebastian Bieringer for proof-reading this thesis, and everyone in the pheno group for providing a fun and interesting working environment, even though we could not see each other as much as I would have liked to.

Erklärung

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

Heidelberg, den 21.02.2021 Heimel