Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

Generative Modelling in High-Energy Physics

Dissertation

zur Erlangung des Doktorgrades an der Fakultät
für Mathematik, Informatik und Naturwissenschaften
Fachbereich Physik
der Universität Hamburg

vorgelegt von

Benno Kaech

Hamburg
2024

Declaration on oath

I hereby declare in lieu of oath that I have written this dissertation myself and that I have not used any auxiliary materials or sources apart from those indicated.

Hamburg, July 12, 2024 Benno Käch

The following machine learning-based writing tools were used in the preparation of this document: DeepL Write, Language Tool, ChatGPT, and GitHub Copilot.

Gutachter der Dissertation:	Dr. Isabell Melzer-Pellmann
	Prof. Dr. Gregor Kasieczka
Zusammensetzung der Prüfungskommission:	Dr. Isabell Melzer-Pellmann
	Prof. Dr. Gregor Kasieczka
	Dr. rer. nat. Olaf Behnke
	Prof. Dr. Sven-Olaf Moch
	Prof. Dr. Christian Schwanenberger
Vorsitzender der Prüfungskommission:	Prof. Dr. Sven-Olaf Moch
Datum der Disputation:	10.07.2024
Vorsitzender des Fach-Promotionsausschusses Physik:	Prof. Dr. Markus Drescher
Leiter des Fachbereichs Physik:	Prof. Dr. Wolfgang J. Parak
Dekan der Fakultät MIN:	Prof. Dr.-Ing. Norbert Ritter

Abstract

Monte Carlo simulations are an important tool in high-energy physics, e.g. to test the predictions of theory models or to infer a priori unknown parameters of the models. However, these simulations demand a substantial amount of computational resources. Thus, this thesis explores the viability of neural-network-based generative models for the CMS experiment at the LHC. The upcoming upgrades of the LHC further challenge the available computing budget of the collaboration in the near future, as these upgrades are expected to substantially increase the number of recorded collisions, necessitating a corresponding expansion in Monte Carlo simulation. For the CMS experiment, the simulation of a single event currently requires approximately two minutes. However, the required time is further expected to at least double, due to upgrades of the CMS detector. This increase is mainly owed to the upgrade of the endcap calorimeters, where the resulting number of channels that need to be simulated will be significantly higher. First, studies on the high-energy physics community `JetNet` dataset are extensively discussed, and the performance of different generative models is compared. An attention-based information aggregation, which scales linearly with the number of particles in terms of computational complexity, is proposed. Not only does this lead to state-of-the-art results on the `JetNet` datasets, but also promising results on the CaloChallenge. Finally, the viability of an end-to-end generation approach is studied in a search for Supersymmetry. The semi-leptonic decay of gluinos, produced via pair production, to neutralinos with an intermediate chargino in the decay chain is investigated. In this search, three a priori unknown parameters need to be scanned, which correspond to the masses of the superpartners. Typically, the mass of the intermediate particle is not scanned, since it is not feasible to generate Monte Carlo simulated data for all parameter combinations. The consequences of not scanning the chargino mass when a neural-network-based classifier is used to identify a signal pure region of the phase space for the statistical inference are investigated. Instead of fixing the mass to an arbitrary value, it is explored whether synthetic data from a generative model, which transforms the distributions from different values of the chargino mass into another, improves the statistical significance of the search. In this study, the required integrated luminosity to reach a similar statistical significance is reduced by $20 \pm 12\%$.

Zusammenfassung

Monte-Carlo-Simulationen sind in der Hochenergiephysik weitverbreitet, benötigen aber viele Rechenressourcen. In dieser Arbeit wird untersucht, ob generative Modelle auf Basis neuronaler Netze für das CMS-Experiment am LHC verwendet werden können, um die Rechenzeit für die Simulationen zu reduzieren. Die bevorstehenden Erweiterungen des LHC erhöhen die Anforderungen bezüglich Rechenressourcen und Datenmenge. Aktuell dauert die Simulation einer Kollision etwa zwei Minuten, aber es wird erwartet, dass sich diese Zeit durch die geplanten Aktualisierungen verdoppeln oder verdreifachen wird. Eine schnellere Simulationstechnik wird daher benötigt. Die Arbeit konzentriert sich zunächst auf die Untersuchung des `JetNet`-Datensatzes und vergleicht die Qualität und Geschwindigkeit verschiedener generativer Modelle. Eine Informationsaggregationstechnik, die auf Attention basiert und linear mit der Anzahl der Teilchen skaliert, bildet das Herzstück des Modells. Weiterhin kann das Modell mit minimaler Veränderung dazu verwendet werden, die Energieeinträgen aufgrund elektromagnetischer Schauer in Kalorimetern konditioniert zu modellieren. Abschließend wird die direkte Generierung von rekonstruierten Analysedaten am Beispiel einer Suche nach Supersymmetrie untersucht. Hier wird der Zerfall von Gluinos, produziert via Paarproduktion, mit einem intermediären Chargino in der Zerfallskette untersucht. Dabei müssen drei unbekannte Massenparameter gescannt werden. Normalerweise würde in diesem Fall nur Monte-Carlo Datensätze für verschiedene Kombinationen von zwei der drei Parametern generiert, während der Dritte auf die Hälfte der Summe der anderen beiden gesetzt wird. Es wird geprüft, ob ein generatives Modell die Abhängigkeit der Daten auf den letzteren Parameter lernen kann, und ob Samples von dem Modell die statistische Signifikanz der Suche verbessern können. Diese Studie zeigt, dass die notwendige integrierte Luminosität um $20 \pm 12\%$ reduziert werden kann durch die Verwendung der synthetischen Daten.

Acknowledgements

This PhD has been an extraordinary journey, and it is challenging to express my gratitude succinctly to all who have contributed to its success. Firstly, I extend my heartfelt appreciation to my office mate, Gabriele Milella. Gabriele became a close friend swiftly and made the distinction between work and leisure almost indistinguishable, enriching nearly every day of this journey. Despite our different research focuses, Gabriele was always there to lend an ear when I needed to vocalise my thoughts to better understand complex concepts. Special thanks go to Lucas Wiens and Frederic Engelke, who adeptly guided me through the analysis discussed in this thesis and always offered assistance when needed. Without them, the last chapter of my thesis would have taken longer than the rest of the thesis itself. Lucas's skill in designing beautiful and well-structured frameworks and Frederic's choice of user-friendly tools significantly enhanced my research experience. I also wish to express my gratitude to Olaf Behnke for his invaluable contributions to my research. His unique approach of elucidating complex ideas through thought-provoking questions has been truly inspiring. I am grateful to my supervisor, Isabell Melzer-Pellmann, who provided me with the opportunity to delve into these fascinating topics and entrusted me with the freedom to pursue my own research interests. I am deeply appreciative to observe and learn from her remarkable professionalism and exceptional interpersonal skills. I also want to thank the machine learning workgroup I was involved in together with Moritz Scham, Simon Schnake, and Dirk Krücker. Moritz, in particular, deserves special recognition for his relentless support in setting up various frameworks and maintaining a structured work environment. Further, I want to thank Jeremi Niedziela for his kind and educational way of discussing different topics. Similarly, I would like to express my gratitude to Erik Buhmann and Raghav Kansal. Despite being part of different research groups, they were always open to insightful discussions and continually inspired me with their own research. I also want to mention Gregor Kasieczka, who kindly offered to supervise me as a professor from the university of Hamburg. The sheer amount of insight and guidance he provided in relation to the limited interaction time is truly remarkable. I am very much inspired by his concise explanation style and profound expertise in the quickly evolving intersection of machine learning and physics. I am thankful for

the engaging discussions during coffee breaks with Freya Blekman and Matthias Komm, which were always enlightening. I also want to thank Andreas Hinzmann, who was always up to answer my naive physics questions. I also need to mention Oleg Filatov and Stepan Zakharov, who shared my love for coffee, and have introduced me to the beauty of filtered coffee. I want to thank Mykyta Shchedrolosiev, who was always a pleasure to meet in the corridor or the gym for a quick and uplifting chat. I will also remember the second-coolest office at DESY with Beatriz Lopez and Federica Colombina who always gave important advice for the PhD life Lastly, I acknowledge the collective effort and support of all my colleagues, the administrative staff, and everyone at DESY who played a role in my PhD journey. Your support was instrumental in my success. Last, but certainly not least, I want to thank my family and friends for their unwavering support and for always being there for me when I needed them most! Thank you all for making this PhD journey not only possible, but also a memorable and enriching experience.

Contents

Glossary

Adam Adaptive moment estimation
AUC Area Under Curve
BatchNorm Batch Normalisation
BSM Beyond Standard Model
CERN Conseil Européen pour la Recherche Nucléaire
CFM Conditional Flow Matching
CMS Compact Muon Solenoid
CNF Continuous Normalising Flow
CP Charge Parity
DDPM Denoising Diffusion Probabilistic Models
DESY Deutsches Elektronen-Synchrotron
ECAL Electromagnetic Calorimeter
EFP Energy Flow Polynomials
ELBO Evidence Lower BOund
FLOPS Floating-Point Operations Per Second
FPD Fréchet Physics Distance
GAN Generative Adversarial Network
GEANT4 GEometry ANd Tracking version 4
GP Gradient Penalty
GPU Graphical Processor Unit
GUT Grand Unified Theory
HCAL Hadronic Calorimeter

HEP High-Energy Physics
HGCAL High Granularity Calorimeter
KL divergence Kullback-Leibler divergence
KPD Kernel Physics Distance
LHC Large Hadron Collider
LSGAN Least-Squares Generative Adversarial Network
LSP Lightest Supersymmetric Particle
MC Monte Carlo
MLP Multi-Layer Perceptron
MSE Mean Squared Error
MSSM Minimal Supersymmetric Standard Model
NF Normalising Flow
NF(c) mass-conditioned Normalising Flow
NF(cc) mass-conditioned and constrained Normalising Flow
NN Neural Network
NSGAN Non-Saturating Generative Adversarial Network
ODE Ordinary Differential Equation
OT-CFM Optimal Transport-based Conditional Flow Matching
pMSSM phenomenological Minimal Supersymmetric Standard Model
PS Preshower System
ReLU Rectified Linear Unit
ROC Receiver Operator Statistic
RQS Rational Quadratic Spline
QCD Quantum Chromodynamics
SGD Stochastic Gradient Descent
SM Standard Model
SUSY Supersymmetry
TGAN Transformer Generative Adversarial Network
TNF Transformer Normalising Flow
VAE Variational Auto Encoder
VNF Vanilla Normalising Flow
WGAN Wasserstein Generative Adversarial Network

Introduction

The Standard Model of particle physics tries to explain and predict the interactions between the fundamental building blocks of the Universe, and is arguably the most precise theoretical framework that science has come up with so far. Nevertheless, there are still phenomena in the Universe, which the Standard Model cannot explain. Supersymmetry tries to extend the Standard Model to address multiple of its shortcomings simultaneously. However, independently of how plausible and aesthetically pleasing a theory is, it is fiction unless it can accurately predict experimental outcomes. The Large Hadron Collider (LHC) together with the Compact Muon Solenoid (CMS) experiment allows experimentalists to investigate and test such extensions of the Standard Model. In this thesis, signatures left by gluino pair production are investigated and simultaneously the importance of the computational methods associated with high-energy physics are highlighted. In particular, the upcoming technical upgrades of the LHC and the CMS experiment pose challenges for the available computing budget. Not only is the number of recorded collisions expected to increase significantly, but also, the amount of recorded data per event will grow, leading to a rise of the computational costs due to the simulation-based inference, since the increase in measured data should be mirrored in the simulation. To replace or assist the simulation, machine learning-based generative models provide a promising candidate. Recently, these methods have proven to be capable of solving tasks, which were unimaginable previously, e.g. the generation of images based on a provided query or the generalising capabilities of large language models. Thus, in this thesis, both these fascinating topics, high-energy physics and generative modelling,

are discussed.

First, an overview of the underlying fundamental physics investigated at the LHC is provided by discussing the SM, its shortcomings and possible extensions. A supersymmetric extension of the SM is discussed in more detail. Subsequently, the experimental apparatus of the LHC and the CMS are briefly discussed, before diving into the fundamental principles of machine learning. In the following chapter, the main classes of different generative models are discussed, and their performance on a two-dimensional toy example is compared. Then, a more particle physics-related problem is studied concerning the generation of highly energetic sprays of particles, referred to as jets, which are abundant in hadron colliders. These jets are represented as point clouds, which are unordered sets that yield complex correlations between the different constituents. This higher-dimensional problem draws attention to the limits of the previously best performing discrete normalising flows. To overcome these issues, the training paradigm is switched to the interplay of two models that are in a constant competition. To generate point clouds with hundreds of constituents, the computation, and memory requirements of these models need to linearly scale with the number of points in the cloud, as a fivefold increase in the point cloud cardinality makes the previously working models virtually untrainable. Thus, a linearly scaling information aggregation is introduced, motivated by a physics inspired approximation. Although, the model employing this aggregation yields competitive performance, its training is unstable and thus impractical for a more general application. Even worse, effects observed previously in the study in low dimensions hint at worse problems. Luckily, continuous normalising flows that are trained with conditional flow matching prove to perform even better than the previous models. Still, the `JetNet` datasets are extended toy examples, as little interest lies in the replacement of PYTHIA with a generative model. Thus, the previously mentioned problem associated with the computing budget is approached from a more practical perspective, namely the computationally expensive modelling of energy deposits left by electromagnetic showers in a calorimeter is investigated. In this example, the generalising capability of the previously studied information aggregation proves its value, as with only minor modifications, the model that was designed for the unconditional generation of point clouds, also performs well in the conditional generation of energy deposits left by electromagnetic showers. Notably, the latter task carries another ~ 80-fold increase

in the number of constituents. Finally, another light is shed on generative modelling. Instead of seeing the generative models as black-boxes that generate samples from a desired distribution from noise, they can be considered models that reshape a standard Gaussian distribution into the unknown data distribution. A problem present in searches for supersymmetric signatures in experiments is the dependence of the distribution of the traces left in the detector on a priori unknown parameters. As it is not possible to generate Monte Carlo simulations for all parameter combinations, a potential alternative is given by continuous normalising flows, that are trained to transform one distribution into the other with minimal transport cost. Thus, it is investigated whether synthetic data drawn from such a continuous normalising flow can benefit the search for gluino production in data simulated for the CMS experiment.

Notational Convention In this thesis, vectors are denoted by bold symbols $\boldsymbol{v}$, whereas scalars s are represented as small letters and matrices W as capital letters.

CHAPTER 1

The Standard Model and its Extensions

In this chapter, a minimal theoretical background for the physical processes that are investigated in the LHC experiments is discussed. First, a brief description of the Standard Model (SM) of particle physics is given before highlighting some of its shortcomings and discussing possible extensions. Finally, a promising candidate called Supersymmetry (SUSY) is discussed in more detail.

1.1 Standard Model of Particle Physics

The SM of particle physics is a theory describing the interactions between the fundamental particles, which are the smallest building blocks of the universe with no in structure. The SM postulates that the Universe is made up of spin-$\frac{1}{2}$ *fermions* that interact through the exchange of integer spin *bosons*.
For fermions, the free Lagrangian density (abbreviated in the following as Lagrangian) is given as:

$$\mathcal{L}_{\text{fermion}} = \bar{\psi}(i\gamma^{\mu}\partial_{\mu} - m)\psi, \tag{1.1}$$

where ψ is the fermion field, m is the mass of the fermion and γ^{μ} are the Dirac matrices that encode the anticommuting behaviour of fermions, i.e. the Pauli exclusion

principle. For bosons with no spin, the Lagrangian is given by:

$$\mathcal{L}_{\text{boson},0} = \frac{1}{2}\partial^\mu\phi\partial_\mu\phi - \frac{1}{2}m^2\phi^2, \tag{1.2}$$

where ϕ is a scalar field.
For spin-1 bosonic vector fields A^μ, the Lagrangian is given by:

$$\mathcal{L}_{\text{boson},1} = -\frac{1}{4}F^{\mu\nu}F_{\mu\nu} + \frac{1}{2}m^2A^\mu A_\mu, \tag{1.3}$$

where $F^{\mu\nu} = \partial^\mu A^\nu - \partial^\nu A^\mu$ is the *field strength tensor*.
In nature, four types of interactions are known that need to be included in the field theory:

1. The electromagnetic field; charged particles interact via the electromagnetic force, which is mediated by spin-1 *photons*.

2. The strong field; the strong force is mediated via spin-1 *gluons*, which cannot be directly observed due to the colour confinement, as further discussed in Section 1.5. The strong force is the reason the nuclei of atoms hold together, although they consist of multiple positively electrically charged protons that repel each other via the electromagnetic force. The first evidence for the existence of the mediator of the strong force, the gluon, has been discovered by events with 3 jets observed at DESY in 1978 [1].

3. The weak field; the mediators of the weak interaction between fermions are spin-1 bosons that were discovered at CERN in 1983 [2, 3]. The weak force was first proposed as an explanation for the beta decay by Enrico Fermi [4] in 1933.

4. The gravitational field; the interaction is postulated to be mediated by spin-2 particles, referred to as *gravitons*. However, as of now, there exists no quantum field theory of gravity that is compatible with the framework of the SM. Fortunately, on the small scales of particle physics, the effects of gravitational forces are insignificant. Thus, for the description of the processes taking place in a collider, excluding gravity does not deteriorate the predictive quality of the theory.

The elementary fermions of the SM are further grouped into *leptons* and *quarks*. The former interact through the weak force and also the electromagnetic interaction if they carry an electromagnetic charge. The latter additionally interact via the strong force.
The beauty of the SM lies in its construction, which is guided by principles of symmetry, and its completion by the *symmetry breaking* which is further discussed in Section 1.6.1.

1.1.1 Natural Units

In particle physics, natural units are used to simplify equations by using units in which the speed of light c and the reduced Planck constant $\hbar$ are set to one[1]. The basic unit, the *electron Volt* (eV) and its derivatives —MeV, GeV, and GeV — are commonly used. Setting $c = 1$ also has the consequence that the units of mass, energy, and momentum are equivalent, which is illustrated by Einstein's famous equation relating mass and energy:

$$E^2 = p^2c^2 + m^2c^4 \underbrace{=}_{c=1} p^2 + m^2. \tag{1.4}$$

1.2 Leptons

Three families, also referred to as *flavours*, of leptons are recognised in the SM, each member consisting of an electrically charged particle and a massless, electrically neutral neutrino[2]. The Dirac equation predicts the existence of an *antiparticle* of the same mass but opposite electrical charge for every fermion. Of the charged leptons, only the electron e^- and its antiparticle, the positron e^+, are stable. The other known leptons are the muon μ^- and the tau lepton τ^-, which only differ from the electron in their masses and lifetimes. The same holds for the corresponding anti-leptons, which are the μ^+ and τ^+. The associated electrically neutral neutrinos

[1] Additionally, the relative permittivity ϵ_0 can be set to 1, leading to the electric charge of an electron to be 1. Rarely, the Boltzmann constant can also be set to one such that temperature is also in units of eV.

[2] In the SM the neutrinos are massless. However, observations of neutrino oscillations [5, 6], imply that neutrinos have mass and that lepton flavour is not conserved. These findings make an extension of the SM necessary, as discussed in Section 1.8.2.

ν_e, ν_μ, ν_τ are similarly matched by their antiparticles $\bar{\nu}_e, \bar{\nu}_\mu, \bar{\nu}_\tau$. The electric charge is conserved in all interactions in the SM and similarly, the lepton flavour is conserved, exemplified by the decay:

$$\mu^- \rightarrow \nu_\mu + e^- + \bar{\nu}_e. \tag{1.5}$$

The leptons and their most important properties are summarised in Tab. 1.1.

Table 1.1: Masses, charges, and generations of leptons, organized by generation, taken from the fit in Ref. [7]. Neutrino masses are given as upper limits at 90%. For the masses of electrically charged leptons, the mass is given up to the first two significant digits of the associated uncertainty.

Lepton (Flavour)	**Mass (GeV)**	**Electric Charge**	**Generation**
Electron (e)	$0.51099895000 \times 10^{-3}$	-1	1^{st}
Electron Neutrino (ν_e)	$< 0.8 \times 10^{-9}$	0	
Muon (μ)	0.1056583755	-1	2^{nd}
Muon Neutrino (ν_μ)	$< 0.19 \times 10^{-3}$	0	
Tau (τ)	1.77686	-1	3^{rd}
Tau Neutrino (ν_τ)	$< 18.2 \times 10^{-3}$	0	

1.3 Quarks

The quarks are grouped into three generations as well, each consisting of two spin-$\frac{1}{2}$ fermions. They are different from leptons as both family members carry an electrical charge of either $2/3e$ or $-1/3e$, where e is the absolute value of the electric charge of a lepton. The different flavours of quarks are not individually conserved and can be changed under weak interactions. Quarks carry a colour charge, but connected to this is *colour confinement*, which is further discussed in Sec. 1.5. Colour confinement makes the observation of individual quarks at low energies impossible and only bound, colourless states are observable.

These bound states are referred to as *hadrons* and are further grouped in *baryons*, which are systems of 3 quarks, and *mesons* containing a quark and an anti-quark. While there exists no stable meson, the lightest baryon with an invariant mass of ~ 938 MeV is the only stable hadron. This is the proton, comprising two *up* quarks and one *down* quark. The neutron is ~ 1.3 MeV heavier and decays in free space

through the β-decay: $n \to p + e^- + \bar{\nu}_e$ with a lifetime of about 978 s. The lightest meson with 135 MeV is the pion π^0. The electrically charged pions π^+ and π^- are made of an up and a down quark and are 4.5 MeV heavier. The neutral π^0 is a superposition of the $u\bar{u}$ and $d\bar{d}$ states. Due to colour confinement, the quark masses depend on the energy and the $\bar{M}S$ scheme [8] is used to define the mass[3]. Instead, they are inferred from various indirect measurements and theoretical calculations. The top quark is an exception because it decays before it can form bound states and, hence, its mass can be inferred from its decay products. The quarks and their properties are listed in Table 1.2.

Table 1.2: Masses, charges, and generations of quarks, organized by generation. Quark masses are given within current experimental constraints and uncertainties, and the values together with the confidence intervals from the fit given in Ref. [7] are quoted.

Quark (Flavour)	**Mass**	**Electric Charge**	**Generation**
Up (u)	$2.16^{+0.49}_{-0.26}$ MeV	$+2/3e$	1^{st}
Down (d)	$4.67^{+0.48}_{-0.17}$ MeV	$-1/3$e	
Charm (c)	1.27 ± 0.02 GeV	$+2/3e$	2^{nd}
Strange (s)	$93.4^{+8.6}_{-3.4}$ MeV	$-1/3e$	
Top (t)	172.69 ± 0.30 GeV	$+2/3e$	3^{rd}
Bottom (b)	$4.18^{+0.03}_{-0.02}$ GeV	$-1/3e$	

1.4 Gauge Symmetries and Interactions

The interaction terms of the Lagrangian are derived by promoting the global gauge symmetries θ of the Lagrangian to local symmetries $\theta(\boldsymbol{x})$. The derivative present in the Lagrangian then demands the introduction of the *covariant derivative* based on a *gauge* field A^k_μ to maintain local invariance:

$$D_\mu = \partial_\mu - igA^k_\mu, \tag{1.6}$$

[3] An exception is the top quark, for which in the following the pole mass is given. For the others, the masses are given im the $\bar{M}S$ scheme.

where g is the coupling constant and the gauge field A_μ^k, which transforms as:

$$A_\mu^k \to A_\mu^k + \frac{1}{g}\partial_\mu \theta(x). \tag{1.7}$$

For a Lagrangian, which is invariant under a symmetry group, the fields transform as:

$$\psi \to e^{ig\theta^k \tau^k}\psi. \tag{1.8}$$

Here, τ^k are the generators of the symmetry group satisfying the following commutations relations:

$$[\tau^i, \tau^j] = if^{ijk}\tau^k. \tag{1.9}$$

For every gauge field A_μ^k the covariant derivative introduces, a free Lagrangian as in Eq. 1.3, but with a field strength tensor $F_{\mu\nu}^k$, arises:

$$F_{\mu\nu}^k = \partial_\mu A_\nu^k - \partial_\nu A_\mu^k + f^{klm} A_\mu^l A_\nu^m \tag{1.10}$$

Equation 1.10 reveals the fundamental connection between physics and the underlying symmetries, since it relates the field strength tensor to the structure constants of the symmetry group.
The symmetry group of the SM is $SU(3)_C \times SU(2)_L \times U(1)_Y$, where the indices indicate the charge that is conserved under each symmetry. Demanding local invariance of $SU(3)_C$ leads to the mediators of the strong interaction, and the breaking of $SU(2)_L \times U(1)_Y$ symmetry leads to the electromagnetic and weak interaction. Without the symmetry breaking, which is discussed in Section 1.6.1, there would be one unified electroweak force with four massless bosons.

Boson	**Mass (GeV)**	**Electric Charge**	**Role/Mediated Force**
Photon (γ)	0 (massless)	0	Electromagnetic force
W boson ($W^\pm$)	80.377 ± 0.012	$\pm e$	Weak nuclear force
Z boson (Z^0)	91.1876 ± 0.0021	0	Weak nuclear force
Gluon (g)	0 (massless)	0	Strong nuclear force
Higgs boson (H)	125.25 ± 0.17	0	Higgs mechanism/field

Table 1.3: Masses, charges, and roles of the gauge bosons and the Higgs boson in the Standard Model as given in Ref. [7]. The W mass fit excludes the Tevatron [9] results of the W mass measurement from 2022.

The gauge fields that emerge from the local gauge invariance requirement are referred to as *gauge* eigenstates. Note that the observable states are in a different basis and referred to as *mass* eigenstates. For the electroweak interaction, the gauge eigenstates are denoted as (B^μ) and $(W^{i\mu}, i = 1, 2, 3)$. The B^μ and $W^{3\mu}$ are mixed to form the physical Z^0 boson and the photon γ. This mixing is parameterised via the Weinberg angle θ_W:

$$\begin{pmatrix} A_\mu \\ Z_\mu \end{pmatrix} = \begin{pmatrix} \cos\theta_W & \sin\theta_W \\ -\sin\theta_W & \cos\theta_W \end{pmatrix} \begin{pmatrix} B_\mu \\ W^3_\mu \end{pmatrix}. \tag{1.11}$$

The remaining $(W^{i\mu}, i = 1, 2)$ gauge eigenstates form the $W^\pm_\mu$ bosons:

$$W^\pm_\mu = \frac{1}{\sqrt{2}}(W^1_\mu \mp iW^2_\mu) \tag{1.12}$$

1.5 Quantum Chromodynamics

During the late 1960s, particle physicists coined the term *particle zoo*, due to measurements of hundreds of strongly interacting particles that were believed to be elementary particles. The introduction of light quarks with flavours (up, down, strange) provided a framework to classify these particles and even foresaw the existence of the Ω^- resonance.
However, the Δ^{++} particle posed a problem. Its configuration $|uuu\rangle$ of three fermions in the same state is forbidden by the Pauli principle[4]. To resolve this, *colour* charge was introduced, and similarly to the electromagnetic charge, is the quantum number that is preserved in strong interactions.

Colour Conservation and Confinement

The colour charge, which is conserved under $SU(3)_C$, can take three different values, denoted by red, green, and blue. Quarks carry one colour (R, G, B), while antiquarks carry an anti-colour $(\bar{R}, \bar{G}, \bar{B})$. Since the underlying symmetry is non-Abelian, gauge

[4]Note that for particles in the form of two up-type quarks and one down-type quark or equivalently two down-type quarks and one up-type quark, the individual quarks can be in different spin states.

bosons carry charge[5].
To conserve the charge at a quark-quark-gluon vertex, the gluons carry a colour and anti-colour. Note that the charged gauge bosons also lead to self-interaction between gluons.
After thorough experimental searches, *colour confinement* was proposed to explain why no individual quarks were ever observed. Colour confinement postulates that free particles must be colour-neutral, i.e. they must be colour *singlets*. This is a consequence of the renormalisation of the gauge theory, which for QCD implies that the coupling constant of the strong force α_S increases at larger distances or more commonly[6]; at a low-energy scale Q^2:

$$\alpha_S(Q^2) = \frac{12\pi}{(11N_c - 2n_f)\log\frac{Q^2}{\Lambda^2_{QCD}}}, \tag{1.13}$$

where N_c is the number of colours, n_f the number of quark flavours and Λ^2_{QCD} the infrared cut-off that sets the validity of the perturbative approximation. This results in two effects at the different ends of the energy scale Q^2:

1. At low Q^2: quarks are not observed as isolated particles, and extracting them from a proton is not possible. This is referred to as *confinement*.

2. At high Q^2: in colliders, the running coupling implies that the strength of the strong force goes to zero when $Q \to \infty$. This leads to *asymptotic freedom*, where quarks (and gluons) behave as free particles and scatter as free particles, leading to deep inelastic scattering.

Another consequence is *hadronisation*, which describes a process that occurs when two quarks are pulled apart. In contrast to an electromagnetic potential, the potential associated with the strong interaction between two quarks increases with distance. It increases to the point where the potential is high enough to create a

[5]From the requirement of invariance under local gauge transformations, the gauge fields given in Equation 1.10 are introduced. The non-zero structure constants f^{ijk} lead to terms in the associated free Lagrangian for the gauge boson field, where only the gauge bosons are involved. This implies that gluons must carry charge.

[6]Note that the energy scale Q is related to the spatial distance λ via the de Broglie relation $\lambda = \frac{h}{p}$, where λ is the wavelength of the particle which is used as probe and p is the momentum of this particle.

new quark anti-quark pair, which is favourable in terms of energy. This leads to sprays of particles, which are abundant in hadron colliders. These sprays of particles are clustered into *jets*, typically with the anti-k_{T} [10] algorithm, simplifying their analysis. The anti-k_{T} algorithm iteratively constructs the jet, favouring the addition of hard particles.

Confinement also implies that from the 9 possible colour/anti-colour combinations for gluons, only eight are physical. This is because the colour singlet $(\frac{(R\bar{R}+G\bar{G}+B\bar{B})}{\sqrt{3}})$ is colourless and hence not confined. The existence of this gluon would result in a long range of the strong force, which does not hold from an experimental viewpoint. Since the $SU(3)_C$ symmetry is exact, gluons are massless[7].

1.6 Electroweak Interaction

Every fermion in the SM interacts via the weak interaction and as such carries a weak charge, which is referred to as weak isospin I_3. Weak interactions are characterised by long lifetimes (10^{-8}– 10^{-6} s), which are significantly longer than for the strong ($\sim 10^{-23}$ s) or electromagnetic ($\sim 10^{-19}$– $\sim 10^{-18}$ s) force[8]. The effects of the weak interaction are particularly visible in processes where the other interactions are either suppressed or forbidden, e.g. when neutrinos are involved since they carry neither electric nor colour charge. The weak force only couples to so-called left-handed particles (and right-handed anti-particles). The handedness describes the behaviour under parity transformations and is a purely quantum mechanical property referred to as *chirality*.

From demanding local gauge invariance, three gauge bosons emerge for the weak interaction[9] and were predicted in the 1960s by Glashow [11], Salam [12] and Weinberg [13]. These gauge bosons comprise the $W^{\pm}$ and Z bosons, which were measured in the 1980s at CERN [2, 3]. However, these measured gauge bosons are not

[7] For an exact symmetry the bosons arising from requiring local symmetry are massless, as mass terms induce terms, which are not cancelled by the covariant derivative in the Lagrangian.

[8] But note that the lifetime is also inversely proportional to the fifth power of the mass, which explains the short lifetime of the top quark.

[9] Note that the covariant derivative regarding the $SU(2)_L \times U(1)_Y$ symmetry is given by $D_\mu = \partial_m u + igT^i W^i_\mu + i\frac{1}{2}g' B_\mu$, where W^i_μ and B_μ are the gauge bosons arising from $SU(2)_L$ and $U(1)_Y$ symmetry and g, g' are the corresponding coupling constants.

massless, as expected from demanding local gauge invariance[10]. This inconsistency is resolved by the electroweak symmetry breaking, discussed in the following Section. The electroweak theory unifies two of the four fundamental forces of nature, namely the weak and electromagnetic forces. The unification is based on the $SU(2)_L \times U(1)_Y$, where L refers to the left-handed and Y to the hypercharge. The hypercharge Y is conserved in electromagnetic and weak interaction and is defined as $Y = 2(Q - I_3)$. In the electroweak theory, the electromagnetic and weak force are considered two different aspects of a single force. However, this unification is only valid at energies above ~ 100 GeV.

1.6.1 Electroweak Symmetry Breaking

At low energies, the $SU(2)_L \times U(1)_Y$ symmetry is broken via the *Spontaneous Symmetry Breaking* (SSB), which is explained by the Brout-Englert-Higgs [14, 15] mechanism. This mechanism introduces a complex field, referred to as the Higgs field $\mathbf{\Phi}$, that permeates all space and is a doublet $\mathbf{\Phi} = \begin{pmatrix} \phi^+ \\ \phi^0 \end{pmatrix}$ under $SU(2)$ symmetry. The doublet has hypercharge $Y = 2$ and isospin $I_3 = \pm\frac{1}{2}$. The first component is electrically charged, whereas the second one is electrically neutral. The Lagrangian for this field is given by:

$$\mathcal{L}_{\text{Higgs}} = (D_\mu \mathbf{\Phi})^\dagger (D^\mu \mathbf{\Phi}) - V(\mathbf{\Phi}). \tag{1.14}$$

The potential $V(\mathbf{\Phi})$ is given by:

$$V(\mathbf{\Phi}) = -\mu^2 \mathbf{\Phi}^\dagger \mathbf{\Phi} + \lambda (\mathbf{\Phi}^\dagger \mathbf{\Phi})^2. \tag{1.15}$$

Depending on the sign of the parameters μ^2 and λ the Higgs potential has different shapes:

- If $\lambda < 0$, the potential has no stable minimum and as such is not physical.
- If $\lambda > 0$, and $\mu^2 > 0$ the only solution is given by $\mathbf{\Phi} = 0$. This is referred to as the zero vacuum expectation value.

[10] However, them being massive explains the short range/long lifetimes of the weak force.

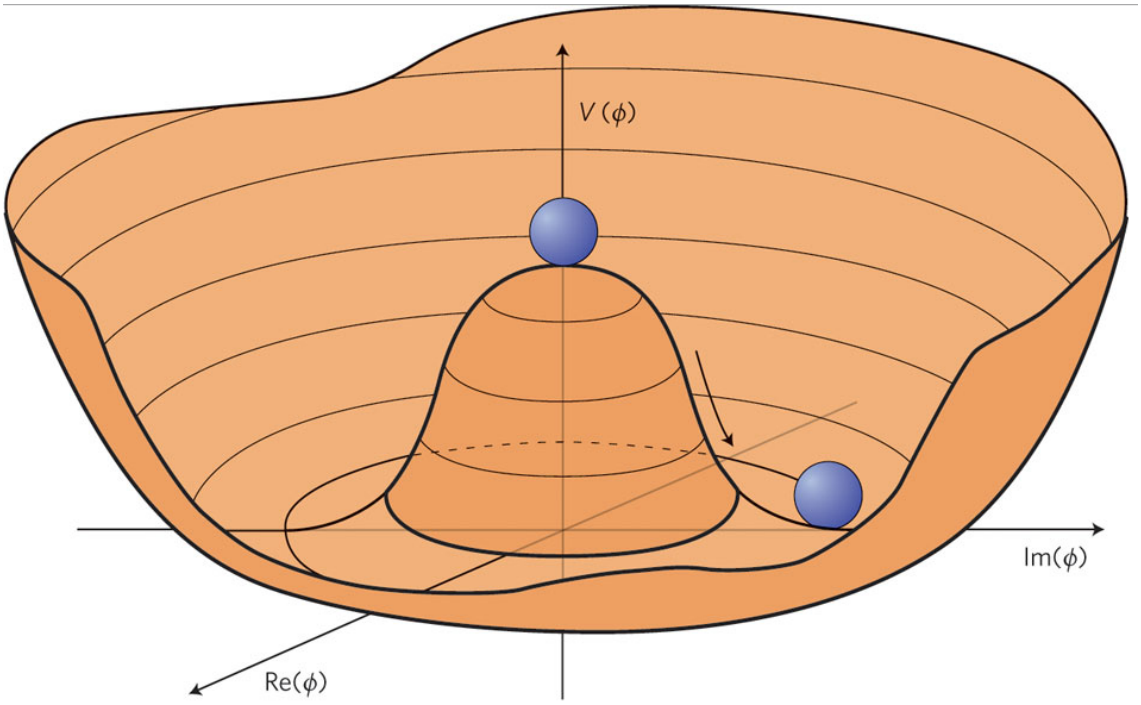


Figure 1.1: Visualisation of the Higgs potential for the case where $\lambda > 0$ and $\mu^2 < 0$ taken from Ref. [16]. There are infinitely many possible minima for the Higgs potential. The symmetry breaking refers to that only one specific point on the circle is chosen as a minimum.

- If $\lambda > 0$ and $\mu^2 < 0$, there is a whole circle of minima at non-zero values of the field $\mathbf{\Phi}$ where $\mathbf{\Phi}^\dagger\mathbf{\Phi} = \frac{-\mu^2}{2\lambda}$. This case is illustrated in Fig. 1.1.

The symmetry is broken in the sense that from a set of infinitely possible minima, one specific point is chosen as a minimum and the Higgs field is approximated through a taylor expansion as $\phi = \frac{v}{\sqrt{2}} + h(x)$, where v is the vacuum expectation value and h the Higgs boson. The Goldstone theorem [17] states that the breaking of a continuous symmetry leads to the appearance of new massless scalar particles, one for every generator of the broken symmetry. These are referred to as Goldstone bosons. In the breaking of the electro-weak $SU(2)_L \times U(1)_Y$ symmetry and reducing it to a $U(1)_{\text{em}}$ symmetry, it is more nuanced. From the breaking of $SU(2)_L$, three Goldstone bosons arise. These three degrees of freedom are absorbed by the existing gauge bosons of the electroweak interaction, granting them mass. This leaves a scalar Higgs field

with the following Lagrangian:

$$\mathcal{L}_{\text{Higgs}} = (D_\mu \phi)^\dagger (D^\mu \phi) - (\mu^2 \phi^\dagger \phi - \lambda (\phi^\dagger \phi)^2). \tag{1.16}$$

Excitations of the Higgs field manifest as particles, and as such the existence of the Higgs field can be indirectly confirmed by measuring a particle that is compatible with the properties of the Higgs boson. The existence of the Higgs boson was postulated in 1964, yet its discovery took place 48 years later at the Large Hadron Collider at CERN. The discovery took its time because although the theory predicts the mass of the Higgs to be

$$m_H = \sqrt{-2\mu^2}, \tag{1.17}$$

the mass is a priori not known, since μ^2 is a free parameter of the theory. Although the Higgs mechanism allows the gauge bosons to acquire mass, an additional mechanism is necessary to give masses to fermions. This mechanism is referred to as the *Yukawa* interaction between the scalar Higgs field ϕ and the fermion field ψ:

$$\mathcal{L}_{\text{Yukawa}} = -y \bar{\psi}_L \phi \psi_R, \tag{1.18}$$

where y is the Yukawa coupling constant.

1.7 Lagrangian of the Standard Model

The Lagrangian of the SM is fairly complicated and for brevity, the simplified, more memorable form is given in the following[11]. The interested reader can consult Ref. [18] for its explicit form and full derivation.

$$\mathcal{L}_{\text{SM}} = \mathcal{L}_{\text{gauge}} + \mathcal{L}_{\text{fermion}} + \mathcal{L}_{\text{Yukawa}} - V(\phi), \tag{1.19}$$

where:

- $\mathcal{L}_{\text{gauge}} = -\frac{1}{4} F_{\mu\nu} F^{\mu\nu}$ includes the kinetic terms of the gauge bosons and their self-interactions, responsible for the electromagnetic, weak, and strong forces. Note

[11] Which is conveniently available on at least one of all the coffee mugs available at the local HEP institute.

that the field strength $F^{\mu\nu}$ here takes all gauge bosons from the electroweak and strong forces arising from the demand of local gauge invariance with respect to $SU(3)_C \times SU(2)_L \times U(1)_Y$.

- $\mathcal{L}_{\text{fermion}} = i\bar{\psi}\gamma^\mu D_\mu \psi$ includes the kinetic terms for the fermions (quarks and leptons) and their interactions with gauge bosons.
- $\mathcal{L}_{\text{Yukawa}} = -y_{ij}\bar{\psi}_i \phi \psi_j + \text{h.c.}$, where $\boldsymbol{y}$ are the Yukawa coupling constants. Note that the *h.c.* stands for the hermitian conjugate of the term. This term describes the interactions between fermions and the Higgs field, leading to fermion masses after symmetry breaking.
- $V(\phi)$ describes the Higgs potential.

1.8 Open Questions in the Standard Model

The Standard Model is arguably the most precise and successful theory of fundamental physics. A prime example is the anomalous magnetic dipole moment of the electron, which is precisely predicted by the SM and measurements agree to an accuracy of 10^{-10}. Nevertheless, there are still open questions the SM cannot answer. A prime example is that gravity is not included in the SM, since quantising gravitation leads to a non-renormalisable theory. In the following, other shortcomings of the SM are highlighted and briefly discussed.

1.8.1 Charge Parity Violation

Baryons are significantly more abundant in the observable universe than antibaryons, this circumstance is referred to as baryon asymmetry. Neither the SM nor general relativity provides an explanation for this. Note, however, that this problem follows from the natural assumption that the universe is neutral to begin with. Based on this assumption, a possible cause comes from a charge parity (CP)-violating process[12], as otherwise an equal number of left-handed baryons and right-handed antibaryons are produced by this process. The SM contains a CP-violating process due to the

[12] A CP symmetric theory remains unchanged if particles are swapped with their antiparticles, and left-handed and right-handed particles are interchanged.

complex phase present in the Cabibbo-Kobayashi-Mashkawa (CKM) flavour mixing matrix. The CKM matrix describes the mixing between the mass eigenstates of quarks under the weak interaction, which allows quark flavour to change under weak interactions[13]. The CP-violating effects observed in SM experiments involving neutral kaons and B mesons are explained by this complex phase. However, the amount of CP violation predicted by the SM is not enough to account for the observed baryon asymmetry in the universe.

1.8.2 Neutrino Oscillations

Neutrinos are postulated to be massless in the SM; however, neutrino oscillations have been measured, in e.g. Ref. [19], where the flavour of a neutrino changed during its propagation. If neutrinos were massless like photons, no shift in the mass eigenstates leading to oscillations of the flavour would be measurable[14].

1.8.3 Dark Matter & Dark Energy

Results from astrophysics and cosmology imply that only around 5% of the energy density of the universe is accounted by ordinary matter. Around five times more of the matter in the universe is referred to as *dark* matter, as it does not interact with electromagnetic radiation and hence can only be observed astronomically by indirect methods. Dark matter was first postulated in 1937 [20] to explain deviations of measured rotation curves of galaxies. Further confirmation is given by studies of the microwave background [21]. But even more, the dark matter with the baryonic matter combined only accounts for $\sim$ 32% of the energy density of the universe needed to explain the accelerated expansion of the universe. The remaining discrepancy is termed *dark energy*, for which the first evidence came from the measurement of supernovae [22]. While some extensions of the SM, as Supersymmetry introduced in Section 1.10, provide a candidate for dark matter, no extension provides a compelling explanation of dark energy at the moment.

[13] Similarly, the Pontecorvo-Maki-Nakagawa-Sakata matrix is constructed for neutrinos to describe neutrino oscillations.

[14] Note that the neutrino oscillations would also not be present if the masses were degenerate.

1.8.4 Aesthetic Shortcomings

Another argument for the expansion of the SM is based on some aesthetic shortcomings of the SM, which are based on the concept of naturalness. Naturalness imposes that the parameters of the theory should all be in the same order of magnitude, or differently put, that the dimensionless ratio between free parameters should take values around 1. Furthermore, the parameters should not need an extensive fine-tuning, which refers to the need to adjust the parameters precisely to fit certain observations. A prime example comes from the neutron electric dipole moment, which is measured to be below 0.18^{-25} e cm [23]. In principle, QCD can violate CP symmetry by a term in the QCD Lagrangian ($\bar{\theta}$), which contributes to a non-zero electric dipole moment for the neutron. But to fit the small experimental observations of the neutron electric dipole moment, this requires to tune the $\bar{\theta}$ parameter of the theory to a value smaller than 10^{-10}, which is unnatural.

Hierarchy Problem

In Equation 1.17, only tree-level contributions to the Higgs mass are accounted for. Every particle that couples to the Higgs boson (i.e. every massive particle) will contribute with a radiative contribution. These terms produce a correction to the Higgs mass with a divergent integral that can be regularised with a cut-off Λ_{UV}^2 where the validity of the SM breaks down. If one assumes that no Beyond Standard Model (BSM) theory is present up to Λ_{Planck}, the corrections require a fine-tuning of the order of 10^{-17} to explain the low Higgs mass.

Unification of Coupling Constants

All coupling constants of the underlying symmetry groups of the SM have a different evolution, but the extrapolation at very high energies suggests that they unify at an energy scale of $M_{\text{GUT}} \approx 10^{15} - 10^{16}$ GeV. However, if no BSM particles are involved, there is no exact crossing point of the three trajectories.

1.9 Beyond Standard Model Extensions

Before discussing supersymmetry, some other extensions of the SM are briefly discussed here. These extensions usually introduce new symmetries, particles, or degrees of freedom to account for some shortcomings of the standard model.

- Extra Dimensions: to unify General Relativity and Quantum Field Theories, more than four space-time dimensions are required, as otherwise the theory is non-renormalisable. A non-renormalisable theory cannot make precise predictions, as infinitely many parameters are needed to account for divergences arising from quantum loop corrections. Many signatures from extra dimensions were proposed that could be found in the LHC experiments. One of them is to discover heavier versions of SM particles, referred to as Kaluza-Klein recurrences [24], that must exist in higher dimensions. Another possibility arises from the missing energy signature left by gravitons, since gravity can probe these extra dimensions and hence, the graviton disappears into them.

- Little Higgs: this extension is based on the idea that the Higgs boson is a pseudo-Goldstone boson arising from a breaking of a global symmetry at a TeV scale. The mechanism then serves as an explanation of the low mass of the Higgs boson of the SM.

1.10 Supersymmetry

Supersymmetry (SUSY) introduces a novel symmetry between fermions and bosons. For every fermion and boson, there exists a bosonic respectively fermionic superpartner, referred to as bosinos and sfermion. The superpartners are denoted by the same symbol as the particles (and antiparticles) but have a $\sim$ on top of them.
The conservation of R-parity is introduced add-hoc to keep the baryon and lepton number conserved[15]. The new quantum number R is defined as:

$$R = (-1)^{3B+L+2s}, \tag{1.20}$$

[15] Additionally, this prohibits rapid proton decay.

where B is the baryon number, s the spin, and L the lepton number. If R-parity is conserved, the lightest supersymmetric particle is stable and thus a good dark matter candidate. If SUSY was an exact symmetry, the SM particles and their corresponding superpartners would be mass degenerate. Since none of such particles have been observed, SUSY needs to be a broken symmetry. Multiple mechanisms have been proposed for the breaking of supersymmetry. For completeness, the three most common mechanisms are listed in the following:

1. Gravity-mediated breaking: this method was first proposed and postulates that SUSY breaks through gravitational interactions. The gravitino acquires its mass through the supersymmetric version of the Higgs mechanism.

2. Gauge-mediated breaking: here, the symmetry is broken via gauge interactions.

3. Anomaly-mediated breaking: this is again a special case of gravity-mediated SUSY breaking through the super-Weyl anomaly, which means that quantum corrections break SUSY.

Many Grand Unified Theories (GUTs) yield supersymmetry as a property, further motivating it as an extension of the SM. Introducing SUSY requires doubling the number of particles, which leads to many new possible signatures. In practice, experimentalists often test a minimal supersymmetric expansion of the SM.

1.10.1 Minimal Supersymmetric Standard Model

The Minimal Supersymmetric Standard Model (MSSM) is an extension of the SM, that introduces only the minimum number of new particles and new interactions. However, supersymmetric theories require at least two Higgs SU(2) doublets to give mass to up-type (down-type) quarks via H_u (H_d). This leads to a total of eight degrees of freedom, where three of them are needed to give masses to the gauge bosons of the weak interaction. The other five become observable particles:

- Two charged Higgs bosons $H^{\pm}$,
- Two CP-even neutral Higgs bosons h^0, H^0,
- A CP-odd neutral Higgs boson A^0,

Table 1.4: Summary of the postulated supersymmetric particles (sparticles)

Sparticle	**R-parity**	**Spin**
Slepton ($\tilde{\ell}$)	-1	0
Squark ($\tilde{q}$)	-1	0
Gluino ($\tilde{g}$)	-1	1/2
Neutralinos ($\tilde{\chi}_1^0, ..., \tilde{\chi}_4^0$)	-1	1/2
Charginos ($\tilde{\chi}_1^\pm, \tilde{\chi}_2^\pm$)	-1	1/2

In Tab. 1.4 the superparticles postulated in this framework are listed, note that the charginos (neutralinos) are mixtures of the superpartners of the Higgs bosons, referred to as higgsinos, and the $W^\pm$ (Z^0, B^0) bosons, referred to as winos (binos)[16]. One of the original motivations for supersymmetry comes from the hierarchy problem mentioned in Sec. 1.8.4, as for every radiative correction of a particle from BSM physics an identical but opposite signed term arises from the superpartner[17]. Similarly, if the superpartners of the SM are near the TeV scale, the coupling constants unify at around 10^{16} GeV. Finally, if R-parity is preserved in the MSSM, the Lightest Supersymmetric Particle (LSP) of the MSSM is stable and weakly interacting. This makes the LSP a good dark matter candidate. However, the MSSM model still yields over 100 free parameters, making it difficult to test in an experiment. The phenomenological MSSM (pMSSM) [25] reduces the number of free parameters even further to 19 through the following experiment-inspired assumptions:

1. No flavour changing neutral currents
2. No new CP-violating processes
3. Mass degenerate first and second-generation squarks.

Thus, 15 a priori unknown parameters are the masses of the newly postulated particles. Additionally, there is $\tan\beta$ which is the ratio of the vacuum expectation values of the two Higgs doublets and three trilinear couplings of the third generation (A_t, A_b, A_τ).

[16] Note that for simplicity the graviton and its superpartner, the gravitino, have not been included.

[17] The opposite sign arises from Fermi-statistics

1.10.2 Simplified Model for Gluino Pair Production

The many free parameters of supersymmetric extensions of the SM pose a problem for experimentalists. To infer parameters from observed data, particle physicists often scan different parameter ranges. But for the MSSM, where there are over 100 free parameters, it is not feasible due to the curse of dimensionality. Even in the pMSSM, where the number of free parameters is significantly reduced, this remains impossible. To illustrate this, consider a search, where only 3 possible values per parameter are scanned. This leads to over a billion possible parameter combinations, each leading to individual kinematics signatures. Thus, experimental searches investigate *simplified models* [26, 27], where only the existence of even fewer sparticles is considered. Usually, the allowed decays of the newly postulated particles to already existing particles is limited, and a specific signature is investigated. This simplification also benefits theorists, as the results from multiple simplified models can be scaled to check whether a more comprehensive model can be excluded or not. In the simplified model `T5qqqqWW` studied in this thesis, where the naming follows the CMS convention, two gluinos are produced via pair production and then decay to a pair of quark anti-quark of different flavours ($q\bar{q}'$) together with a chargino ($\tilde{\chi}_1^{\pm}$). The chargino then further decays to a $W^{\pm}$ boson and a neutralino ($\tilde{\chi}_1^0$), which is the LSP. The diagram corresponding to this process is shown in Figure 1.2. An experimental search for the signatures of this model in Run 2 data of CMS has recently been published in Ref. [28]. In the analysis, the gluino and neutralino masses are scanned, excluding gluino masses up to 2050 GeV and neutralino masses up to 1070 GeV. But note, that the chargino mass is set to the neutralino mass plus half of the mass difference between the gluino and neutralino mass. This is an arbitrary choice, and its impact will be further illustrated and discussed in Section 7.2.

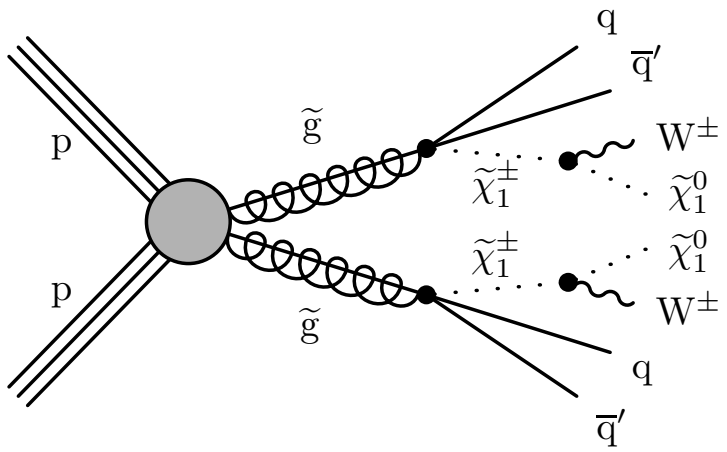


Figure 1.2: Diagram of the simplified model spectra for gluino pair production (`T5qqqqWW`). Due to R-parity, the gluinos are produced in pairs only. The LSP escapes detection, which is a key signature of the signal.

CHAPTER 2

Experimental Apparatus

The LHC [29] is a 26.7 km long particle collider ring for protons and heavy ions located at the "Conseil Européen pour la Recherche Nucléaire" (CERN), located in between 45 and 170 m below the surface of Switzerland and France. The colliding particles travel in opposite direction in two beam pipes. Each beam consists of up to 2808 bunches spaced by 25 ns in time. The two beams cross at four interaction points[1]. In its first run from 2010 to 2013, which already led to the discovery of the Higgs boson, the center-of-mass energy was $\sqrt{s} = 7$ TeV. In the following Run 2, it was then raised to $\sqrt{s} = 13$ TeV. In Run 3, a center-of-mass energy of $\sqrt{s} = 13.6$ TeV is reached. To bend the beams, over 1200 dipole magnets in cryostats that host both beam pipes are used. The required field strength is over 8 T, which is not reachable with conventional conductors. Thus, the magnet coils consist of niobium-titanium (NbTi) that is cooled to 1.9 K, which is below their critical temperature to acquire superconductivity, allowing for a significantly stronger magnetic field. The LHC is the final step in a chain of pre-accelerators to prepare the necessary beam quality and energy. At the LHC, the protons are accelerated by superconducting radio frequency cavities to their final energy, where each bunch contains more than 10^{11} protons. The LHC had been designed for a luminosity of 10^{34} cm^{-2}s^{-1} but in 2018 twice the design luminosity was reached [30]. Due to the high proton density in each bunch crossing, multiple proton collisions take place simultaneously, but typically

[1]The four bigger experiments (CMS, Alice, Atlas, LHCb) of the LHC are located at these points.

only the most energetic ones are of interest. The additional collisions are referred to as pile-up.
The expected luminosity specifies the number of collisions per unit of time together with the cross-section of the underlying processes, can be computed at the interaction points as:

$$L = \frac{N^2 n_b f}{4\pi\sigma_x\sigma_y} \cdot F, \tag{2.1}$$

where N is the number of particles per bunch, n_b is the number of bunches, f the revolution frequency, σ_x, σ_y the horizontal, and vertical beam sizes and F accounts for a reduction in luminosity due to the slight tilting of the beams at the crossing.

Experiments at the LHC There are four larger experiments located at the four interaction points. ATLAS [31] and CMS [32] are the general purpose experiments which allow precision measurements of the SM as well as searches for various kinds of new physics. ALICE [33] and LHCb [34] are more specialised experiments, for which the luminosity is intentionally lowered to reduce pile-up and prevent detector damage by slightly displacing the beams.
ALICE focuses on heavy ion physics to study the quark-gluon plasma. LHCb was designed to study the difference between matter and antimatter by examining particles that contain the bottom quark.

2.1 Compact Muon Solenoid

The CMS apparatus [32] is a multipurpose, nearly hermetic detector, designed to trigger on [35] and identify [36] electrons, muons, photons, and (charged and neutral) hadrons [37, 38, 39]. A global *Particle-Flow* (PF) algorithm [40] aims to reconstruct all individual particles in an event. It combines information provided by the all-silicon inner tracker and by the lead tungstate electromagnetic and brass-scintillator hadron calorimeters, operating inside a 3.8 T superconducting solenoid. Additionally, the algorithm includes data from the gas-ionisation muon detectors embedded in the flux-return yoke outside the solenoid. The strong magnetic field of the solenoid is fundamentally important for the operation of the detector because it allows to accurately and precisely measure the transverse momentum of charged particles. As

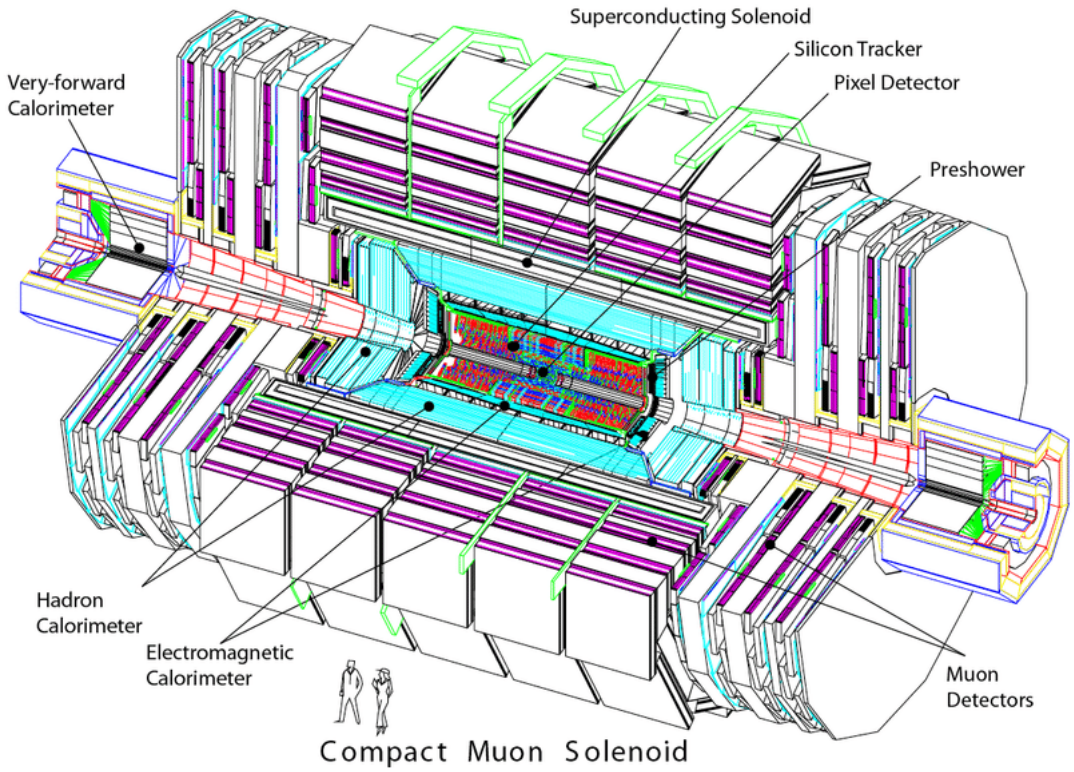


Figure 2.1: Schematic view of the detector taken from Ref. [41]: The innermost layer is composed of the silicon tracker and pixel detectors. The subsequent layer consists of the electromagnetic and hadron calorimeters, each partitioned into barrel and endcap regions. All detectors are enclosed within the superconducting solenoid. The outermost layer is composed of the muon system.

the name suggests, the detector was designed to efficiently detect muons and to accurately and precisely measure their transverse momentum. The resolution of the momentum when using combined information from the tracker and muon system is below 1.5% for a muon of approximately ~ 100 GeV. In Figure 2.1 a schematic overview of the detector is shown.

2.1.1 Coordinate System

The origin of the CMS coordinate system is centred at the nominal collision point inside the detector. The z-axis points in the beam direction, the y-axis points upwards, and the x-axis points inwards to the LHC detector ring. The azimuthal

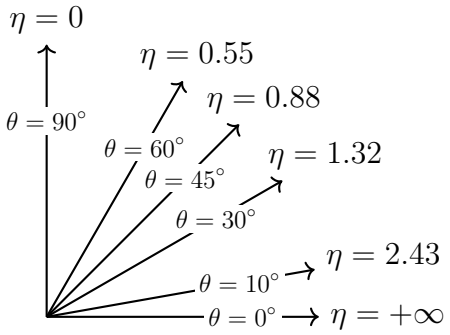


Figure 2.2: Correspondence of η to different values of θ. The forward region of the detector usually refers to $\eta > 1.4$, whereas the very forward region refers to $\eta > 3$.

angle $\phi \in [0, 2\pi]$ is defined in the x-y plane and measured from the x-axis. The polar angle $\theta \in [0, \pi]$ is measured from the z-axis. The pseudorapidity, defined as $\eta = -\ln \tan \theta/2$, is more practical to work with in HEP, as differences in η are Lorentz invariant. Figure 2.2 illustrates the correspondence between η and θ. The momentum transverse to the beam direction is denoted by p_{T}. Another important variable is the missing transverse momentum $p_{\mathrm{T}}^{\mathrm{miss}}$, which is the negative of the vectorial sum of the reconstructed particles $\boldsymbol{p}_{\mathrm{T}}$:

$$\boldsymbol{p}_{\mathrm{T}}^{\mathrm{miss}} = -\sum_{\mathrm{part.}} \boldsymbol{p}_{\mathrm{T}}. \tag{2.2}$$

2.1.2 Tracker

The CMS tracker, the largest silicon tracker ever built, spans over 200 m^2 of active sensing area, it is 5.8 m long and has a diameter of 2.6 m. The tracker can be conceptually divided into two main components:

- **Pixel Detector**: Closest to the interaction point, the pixel detector consists of 124 million silicon pixels in four layers since its upgrade in 2018. Given its proximity to the collision point, it is designed to handle the highest particle densities and provides precise hit information crucial for determining particle trajectories.

- **Strip Detector**: Surrounding the pixel detector, the strip detector comprises microstrip modules arranged in ten layers. There are also four endcaps on

every side, consisting of two inner endcaps and two outer ones. The inner endcaps contain 3 concentric layers, the outer endcaps have their silicon modules optimised differently for their place within the detector. The strip detector extends the tracking coverage and assists in the complete reconstruction of charged particle paths.

The tracking system was designed to accurately measure the trajectories of charged particles. The high granularity of the tracker ensures efficient and precise tracking, which is indispensable for many physics analyses. From the curvature of the trajectory and the magnetic field, the momentum, and charge of the incident particle can be inferred. The precision of the CMS tracker allows for an excellent momentum resolution. Moreover, the tracker plays a crucial role in the reconstruction of the primary and secondary vertices, aiding in the identification of long-lived particles, such as those resulting from the decay of b hadrons.
The tracker faces high levels of radiation, especially the innermost layers of the pixel detector. To ensure longevity and functionality, the materials and electronics of the tracker are chosen for radiation hardness. Additionally, an efficient cooling system is in place to manage the heat produced by the electronics and radiation exposure.

2.1.3 Electromagnetic Calorimeter

The Electromagnetic CALorimeter (ECAL) uses $\sim 760'00$ lead tungstate ($PbWO_4$) crystals to measure the energy of incoming electrically interacting particles. It can measure the pseudorapidity up to $|\eta| < 3$. Figure 2.3 depicts a schematic overview of its design. It consists of the Barrel (EB) in the central region and two Endcaps (EE). The EB covers up to $|\eta| < 1.4$ and comprises $\sim 61'000$ crystals. The EE cover from $1.4 < |\eta| < 3$ and both contain ~ 7500 crystals each.
When an electron or photon enters the ECAL, it interacts with the dense material. Highly energetic photons can convert into electron-positron pairs, whereas electrons emit photons through bremsstrahlung. These processes are responsible for the showering of electromagnetic particles.
The lead tungstate crystals act as scintillators and emit light proportional to the energy of the incident particle. This scintillation light is converted by a photodetector into an electrical signal. Apart from measuring the energy of the incoming particle,

it also measures the position where the particle hit, which is vital information when reconstructing the kinematics of the event. The ECAL thickness is chosen to be larger than 25 *radiation lengths*. The radiation length describes a material characteristic related to the energy loss of highly energetic particles interacting electromagnetically with the material. It is defined as the mean length at which the energy of an electron is reduced by a factor $1/e$. The energy resolution has been measured in test beams for energies between 20 and 250 GeV and the resolution is parameterised in Ref. [42] as:

$$\left(\frac{\sigma}{E}\right)^2 = \left(\frac{2.8\%}{\sqrt{E/GeV}}\right)^2 + \left(\frac{0.12\%}{E/GeV}\right)^2 + (0.3\%)^2 \,. \tag{2.3}$$

This means that for an electron of about 100 GeV the resolution is $\sim 0.4\%$. For higher energies, shower leakage becomes significant and the resolution degrades.

Preshower System

The Preshower System (PS) is located in front of the endcaps of the ECAL, consisting of two layers of silicon strip detectors interleaved with a lead radiator. The primary role of the PS is the identification and distinction between neutral pions π^0 and photons γ. This is crucial because most of the time the neutral pions decay to two photons in proximity. The finer granularity of the PS allows distinguishing between these two signatures.

2.1.4 Hadronic Calorimeter

The ECAL is surrounded by the Hadronic CALorimeter (HCAL) with coverage up to $|\eta| < 3$, comprising the Hadron Barrel (HB), Hadron Endcap (HE), Hadron Outer (HO) and Hadron Forward (HF). The HCAL is a sampling calorimeter made of alternating layers of brass and plastic scintillator. As the particles traverse the detector, they interact with the brass, creating showers. The particles in these showers then interact with the scintillators, causing them to produce light, which is converted by wavelength-shifting wires and channelled to photodetectors via optical fibres to create an electrical signal. Outside the magnet in the barrel region, there is the HO, which extends the covering of ~ 5 interaction lengths by the HCAL to 11 hadronic interaction lengths λ. The hadronic interaction length, similarly to the

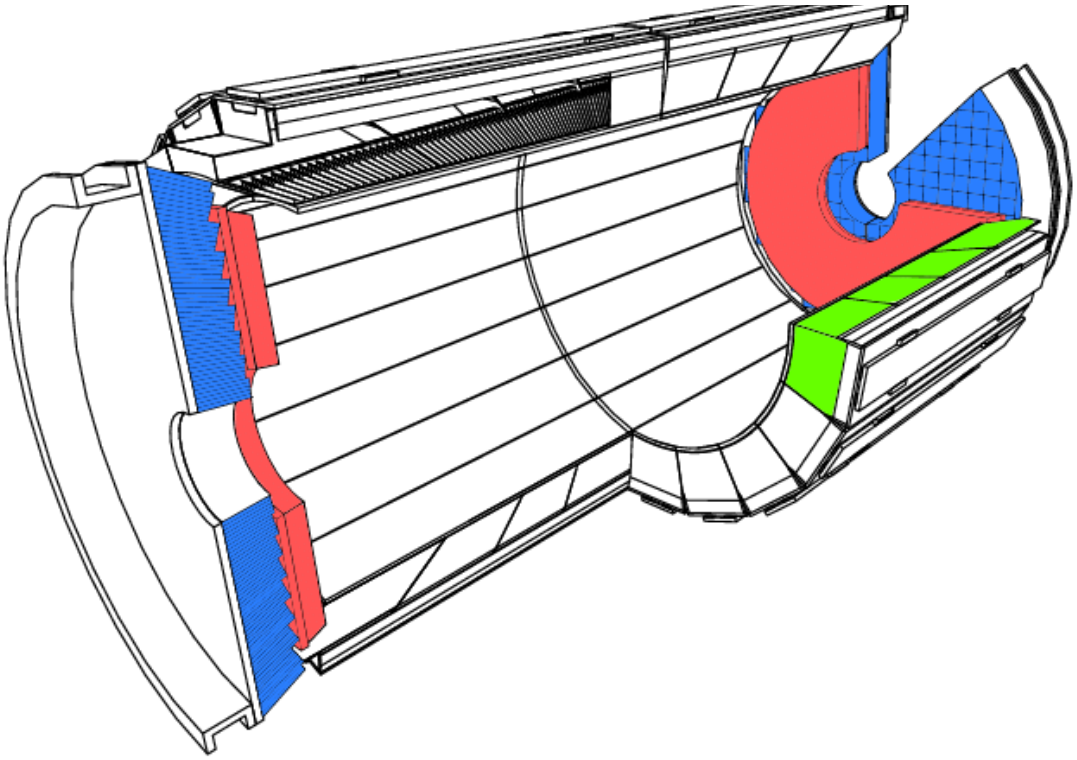


Figure 2.3: The ECAL barrel comprises 36 Super modules (one highlighted in green). The ECAL endcaps (blue) are divided into four dees, two on every end. The preshower systems (red) are located before the dees on each side. Figure from Ref. [43].

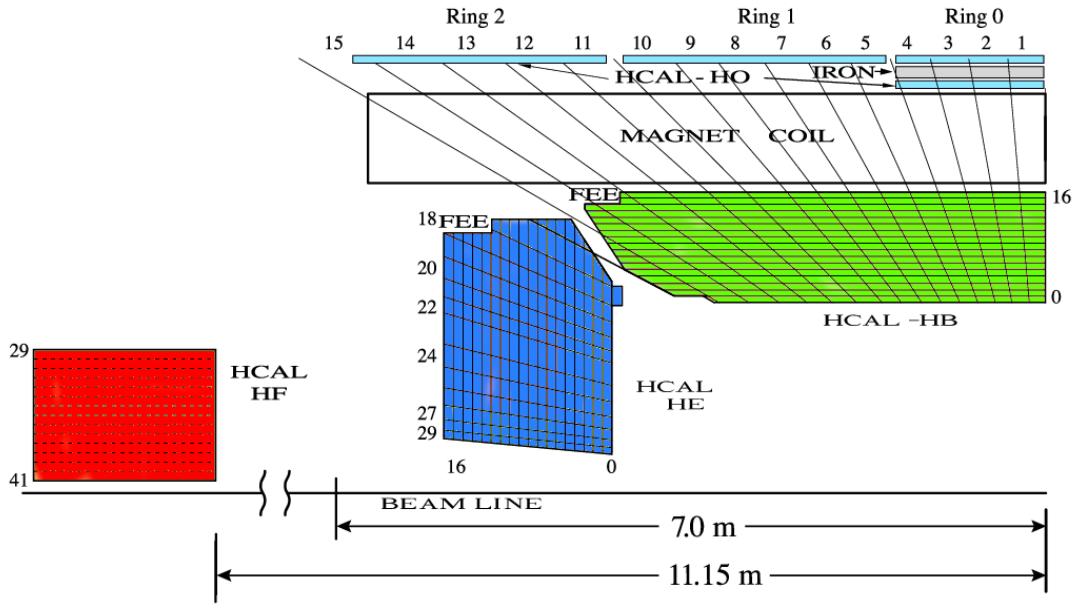


Figure 2.4: Schematic of the HCAL adapted from Ref. [45]. The HCAL barrel (green) comprises 36 identical azimuthal wedges. The HCAL endcaps (blue) contain 17 alternating layers of brass absorbers and plastic scintillator on each side. Shown in light blue is the outer hadron calorimeter placed outside the solenoid. This makes the HCAL cover a region $|\eta| < 3$ with a minimum of 11.8 interaction lengths, except for the transition region between barrel and endcap located at $|\eta| \sim 1.3$. An additional forward calorimeter is located 11.15 m away from the nominal interaction point to extend the coverage to a range of $|\eta| < 5.2$.

radiation length, defines the distance a hadron travels before it interacts with a nucleus, and is significantly larger than the radiation length. To cover the forward region ($|\eta| < 5$) an iron/quartz detector is placed at the end of the HE. The energy resolution of HCAL modules was measured in Ref. [44] with pion beams with an energy of $20 - 300$ GeV as

$$\left(\frac{\sigma_{HB+HE}}{E}\right)^2 = \left(\frac{115\%}{\sqrt{E/\text{ GeV}}}\right)^2 + (5.5\%)^2, \tag{2.4}$$

which is significantly lower compared to the ECAL and leads to a resolution of 12% on a 100 GeV pion.

2.1.5 Superconducting Magnet

The heart of the CMS is a superconducting magnet which was designed to reach a 4 T field resulting in a stored energy of 2.6 GJ at full current[2]. Due to the large number of turns required to generate such a strong magnetic field, the winding is composed of 4 layers. As mentioned before, the strong magnetic field bends the trajectories of charged particles providing an accurate measurement of the momentum of charged particles, since the curvature of the track together with the magnetic field strength and charge of the particle is related to the transverse momentum. The magnet coils also consist of NbTi, which is cooled to 4 K to acquire superconductivity. The magnetic flux is returned through a $10'000$ t steel yoke, primarily consisting of iron because of its high magnetic permeability. Additionally, this heavy construction serves as structural support for the detector and provides shielding against background radiation. Outside the yoke, the field strength is still 2 T, which is necessary for precise momentum measurement of the muon systems.

2.1.6 Muon Systems

As the name Compact Muon Solenoid suggests, the detector is optimised to accurately and precisely measure muons, which is crucial for the identification of various SM processes. The muon chambers are located as the outermost detector of CMS. In the barrel region drift tubes are used, as the muon flux is relatively low, in the endcaps cathode strip chambers are thus used. Complementary, resistive plate chambers are installed in the barrel and endcaps. Generally, the working principle of these detector parts is based on the ionisation of gases, where the freed electrons leave an electric pulse behind. This pulse is then read out to localise the trajectory of the muon. The target dimuon mass resolution was $\sim 1\%$ at 100 GeV and measured to be $\leq 1\%$ in the barrel region and $\leq 3\%$ in the endcap for a muon of ~ 100 GeV [38] when the measurement is combined with the measurements from the tracker. As depicted in Figure 2.5, the tracker contributes essentially to the precision of the muon transverse momentum measurement, especially for low and medium transverse momentum. This is due to the significantly higher spatial resolution of the tracker compared to the muon chambers.

[2]In practice the field strength was then reduced to a central magnetic flux density of 3.8T.

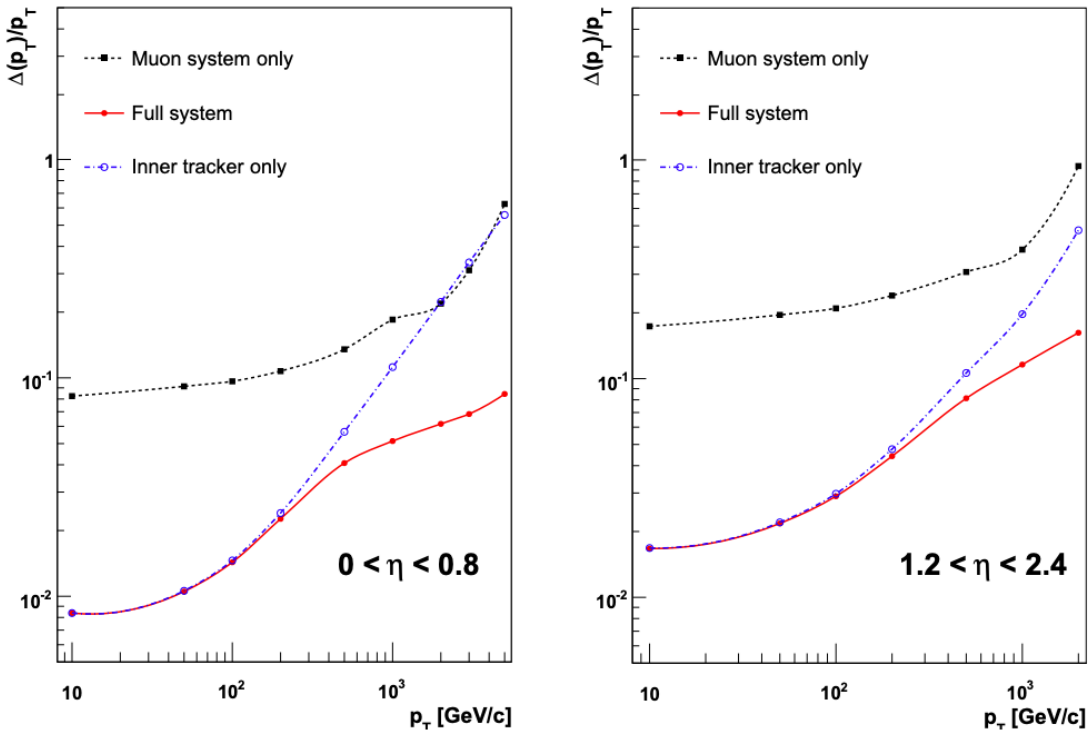


Figure 2.5: Muon transverse momentum resolution from Ref. [32] as a function of the transverse momentum. The left panel is for $|\eta < 0.8|$, the right panel for $1.2 < |\eta| < 2.4$. The resolution of the tracker is significantly higher than the one from the muon detector, however, the combination of the two measurements has a high importance to the resolution of highly energetic muons.

2.2 Data Acquisition System

To handle, process, and store the vast amount of data measured in the CMS experiment, a sophisticated data acquisition system is used. A two-tiered trigger system is employed to select potentially interesting collisions. The first level is called Level-1 (L1) Trigger and is a hardware-based system with a latency of ~ 3.2 μs and reduces the rate from 40 MHz to 100 kHz. The software-based high-level trigger (HLT) then further reduces the rate to around 1 kHz. This means that per year, multiple Petabytes of data are stored.

2.3 Monte Carlo Simulation Chain

Although there is a solid theoretical basis to understand the physics processes that take place in a collider like the LHC, it is not possible to directly infer theory parameters from measurements. Not only are higher orders of the matrix element getting rapidly more complex, but there are also non-perturbative effects, which are extremely challenging to calculate. Some phenomena, such as hadronisation, are not even calculable from first principles. Monte Carlo (MC) simulations provide a means to compare theoretical predictions with measured data. The underlying physical processes and detector responses are simulated, which allows interpreting experimental results.
The MC simulation chain is grouped in the following steps:

1. *Matrix Elements & Parton Shower*: The MC simulation chain starts with modelling the physical process of interest, including parton density, hard matrix element, initial and final state showers, matching between higher order calculations if required, and finally hadronisation and decay of short-lived particles. This comprises the GEN step of the simulation chain.

2. *Detector Simulation*: from here onwards, these final state particles are propagated through the different detector layers, which is referred to as the SIM step. The interactions between detector and particles is usually simulated with the GEometry ANd Tracking version 4 [46] (GEANT4) package in CMS. Note that this also simulates the hardware triggers and the digitisation, referred

to as DIGI, of the measured signal. The CMS GEANT4 simulation includes the shape and position of the detector materials. It also considers details like the atomic composition and radiation length of the materials through which the particles propagate. In the jargon of CMS, this is referred to as the *full simulation*, which is centrally produced for all SM processes. Since this full simulation takes $\sim \mathcal{O}(100\ s)$ per event, searches usually rely on the CMS internal `FastSim` [47] package for the simulation of the signal. The `FastSim` package speeds up the simulation and reconstruction[3] by a factor ~ 20, by using a simplified geometry with infinitely thin material layers. The interactions with the detector materials are described with simple analytical models.
After the SIM and DIGI step, additional low-energetic secondary collisions are overlaid with the signals of the primary collision. This serves to model the pile-up.

In Fig. 2.6, the distribution of the CPU usage for the major processes of the CMS experiment is depicted.

2.3.1 Event Weights

Another important feature of the MC simulation are event weights. In the simplest approximation, they are defined as the ratio of the expected number of events divided by the number of generated MC events. The weights can be used to upweight extreme regions of the phase space or to implement higher order corrections where even negative weights may occur. Thus, instead of being interested in the number of events from the MC simulation, the sum of the weights is considered. The associated uncertainty is then given as:

$$\sigma = \sqrt{\sum_{\text{Events}} w_i^2} \tag{2.5}$$

2.3.2 Computational Costs of the MC Simulation Chain

Although these simulations played a key role in the successful history of HEP, their computational need has become a relevant issue. As described in Ref. [48], the

[3]The reconstruction is almost the same, but there are some differences in the tracking. Not only is a simplified geometry used, but also the track reconstruction is sped up by mixing already reconstructed tracks from the primary collision and minimum-bias events.

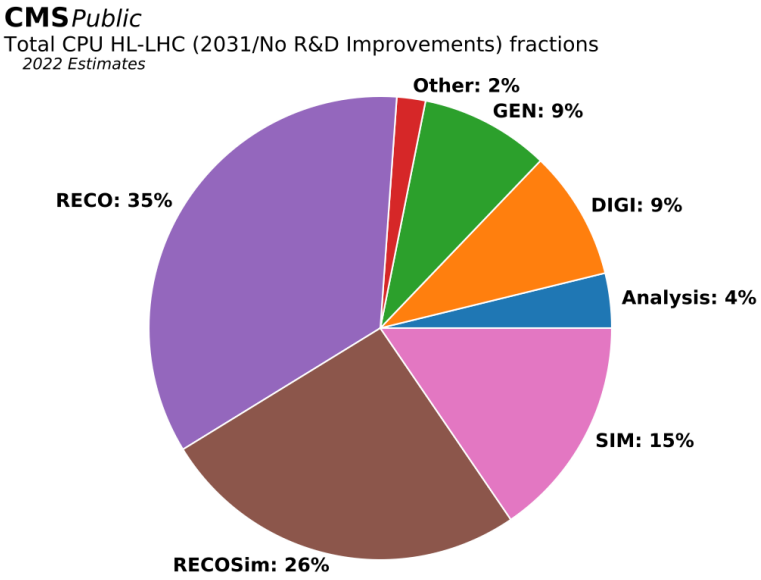


Figure 2.6: Pie chart from Ref. [48] of the fractional CPU resources needed, split up for the different processes. The MC simulation (GEN+SIM+DIGI+RECOSim) uses over 50% of the available resources.

projected computing needs for the coming years could exceed the available resources without major improvements from R&D side. Figure. 2.7 illustrates this potentially detrimental problem for the experiment.

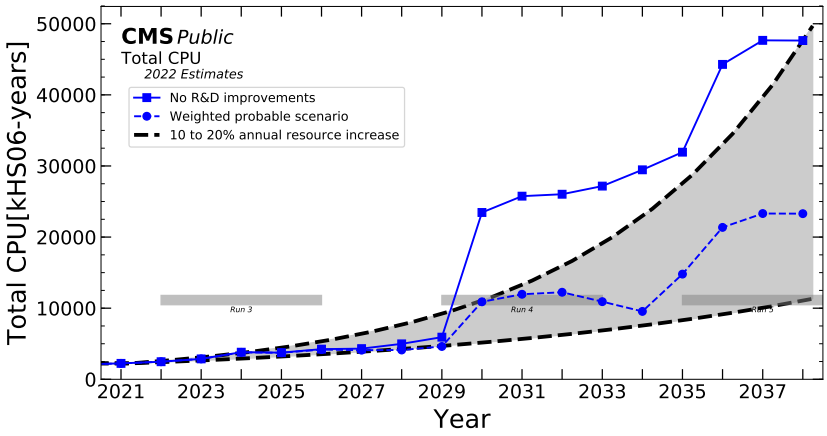


Figure 2.7: Projection of the needed CPU into the era of the HL-LHC, taken from Ref. [48]. The grey band represents the projected availability of the CPU resource with no significant budget increases. The solid line shows the projected computational needs with the current setup, while the dashed line depicts them with significant R&D improvements, like a double as fast detector simulation.

2.4 Upcoming Upgrades

This is increased demand for computing resources is mainly caused by the detector and collider upgrades during the Long Shutdown 3, starting after the end of Run 3. The upgrades are conducted to prepare the detector and collider for the High-Luminosity LHC (HL-LHC) phase, during which the luminosity of the LHC will be increased by one magnitude [49] over the design value of the LHC.

This not only demands an extremely radiation-tolerant detector, but also the pile-up in the detector is increased significantly. To account for this, the High-Granularity Calorimeter (HGCAL) [50] was proposed and is partly being constructed at DESY during the time of writing this thesis. The HGCAL is foreseen to read out more

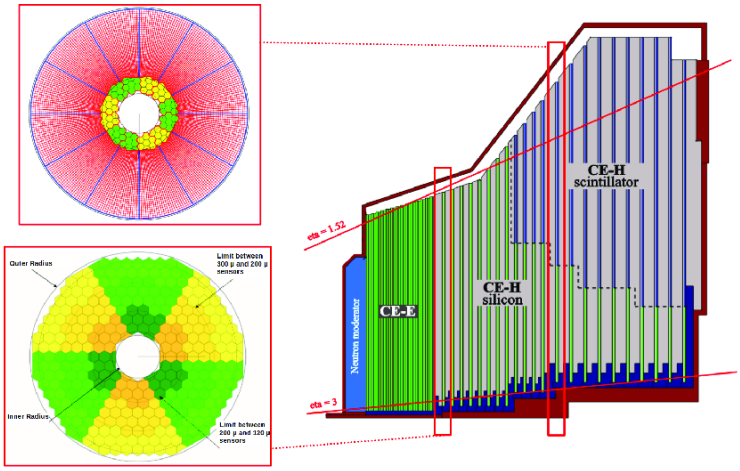


Figure 2.8: Schematic of the HGCAL, taken from Ref. [52]. The right shows the longitudinal cross-section of one half of the HGCAL endcap, the top left shows a layer of the fully hadronic calorimeter and the bottom left a layer of the electromagnetic calorimeter.

than six million channels, which increases the cost of simulation significantly. A schematic of the HGCAL detector is shown in Fig. 2.8. Currently, the simulation of a single collision requires approximately two minutes. With the incoming upgrades after Run 3, the simulation is expected to be two to three times slower [51]. The HGCAL contains two compartments:

1. The electromagnetic compartment (CE-E) comprises 28 silicon-based layers, covering over ~ 25 radiation lengths X_0.

2. The hadronic compartment consists of 22 silicon-based layers and 22 scintillator-based layers covering about 8.5 hadronic interaction lengths λ, which together with the electromagnetic compartment (1.3λ) covers almost 10λ.

The Silicon-based layers consist of hexagonal sensor cells, leading to a complicated detector geometry. In Sec. 6, a point cloud-based model for calorimeter simulation is introduced, and its performance on a highly granular detector is discussed. The irregularity of the HGCAL is another reason for using a generative model based on point clouds. The reconstruction of the detector simulation also contributes significantly to the CPU usage with 24% of the total CPU resources. This means that even if a generative model is used to speed up the detector simulation, the CPU

costs of the synthetic events are still significant. In this thesis, an even more efficient approach is investigated. Instead of replacing parts of the classical generation chain, the viability of directly generating analysis level data with an end-to-end surrogate model is explored. This is discussed in Chapter 7.

2.5 Statistical Analysis

In the following, a highly simplified outline of a *search* is used to highlight the importance of such MC simulation. In a search, the presence of a new physical effect, e.g. a new particle, is investigated. The search is usually guided by a theoretical framework which predicts the existence and properties of the effect in the form of a *cross-section* σ for the new process. The cross-section σ is directly related to the number of collisions N that are produced by this process in the experiment, together with the integrated luminosity $\mathcal{L}_{\text{int}}$, which is a specific property of the collider:

$$N = \mathcal{L}_{\text{int}}\sigma. \tag{2.6}$$

To confirm the existence of a new process, a hypothesis test is conducted where the null hypothesis only contains processes from the SM, associated with the cross-section σ_{SM}. The alternative hypothesis additionally includes the cross-section of the signal process σ_{S}. For both hypotheses, the experimental outcomes are simulated. With these simulated samples, the region of phase space is identified, where the predictions of these two hypotheses differ the most. This is the region, which is most pure in signal events and contains as few background events as possible. In this region, the measured data is then compared to the simulation.

The MC simulation is not only used to set up the experimental analysis, but it also serves to determine the expected statistical significance, which is referred to as the *sensitivity*, of an analysis. The sensitivity is computed by using the Asimov dataset [53], which is an idealised dataset. On the Asimov dataset, the observed outcomes are set to the expected ones, e.g. in a counting experiment the observed number of events n is set to the expected ones. In the case of discovery, where the goal is to reject the null hypothesis of background only, the Asimov data set is given

as:

$$n = s + b. \tag{2.7}$$

In Chapter 7 this will be used and discussed in more detail.

CHAPTER 3

Machine & Deep Learning

Machine Learning (ML), a key pillar of artificial intelligence, has revolutionised many fields, including computer vision, voice recognition, Natural Language Processing (NLP) and many more. Recently, ML has become an essential tool in the domain of experimental particle physics, outperforming classical approaches in many tasks, e.g. event reconstruction [54], particle identification [55], and the search for new particles [56]. A comprehensive review of the application of ML in HEP is given in Ref. [57]. Deep learning, a subset of machine learning, has emerged as a potent tool for handling complex, high-dimensional data. It relies on Neural Network (NN)-based algorithms, which have been remarkably successful in various tasks, surpassing conventional machine learning algorithms in numerous instances [58]. The strength of deep learning lies in its capability to learn from unstructured low-level data and extract discriminative high-level features. This alleviates the manual, time-consuming design and selection of high-level features present in conventional machine learning. This thesis places a primary emphasis on employing NN-based architectures within the realm of high-energy particle physics.
The landscape of NN-based algorithms is vast and rapidly evolving. It comprises a wide variety of model classes, e.g.

- Multi-Layer Perceptrons (MLPs)
- Convolutional NNs (CNNs) [59]
- Recurrent NNs (RNNs) [60]

- Transformer models [61]

These *architectures* have unique strengths and weaknesses and were designed to match particular symmetries present in the corresponding tasks. Given the breadth and depth of this field, this chapter focuses on the most essential foundational knowledge and research in NNs that have had a direct impact on the studies presented here. This includes the following fundamental concepts of NNs:

- the architecture of a basic NN,
- activation functions,
- loss functions,
- backpropagation,
- optimisation algorithms,
- regularisation techniques,
- and weight initialisation.

While striving to provide a comprehensive overview of the relevant topics, the provided coverage is by no means exhaustive. Interested readers are encouraged to explore this exciting field beyond what is covered in this thesis, some useful starting points are given in Refs. [58, 62], or Ref. [63] especially for physicists.

Among the first proposed architectures are Multi-Layer Perceptrons; they comprise a class of feed-forward NNs that includes at least three *layers*:

- An *input layer*, containing as many *nodes* as input features.
- One or multiple *hidden* layers, containing an arbitrary number of nodes.
- One *output* layer, comprising as many nodes as desired outputs, e.g. one single node in a binary classification task.

In a *fully connected* NN all nodes between neighbouring layers are connected via learnable weight matrices $(W_{ij})_{(1<i<n,1<j<m)}$, where the former index corresponds to the n features of the previous layer and the latter to the m features of the following

layer. Additionally, a *bias* vector $\boldsymbol{b} = (b_j)_{(1 \leq j \leq m)}$ is used for each neuron in a layer. The value of a node in a hidden layer is determined by the values of the nodes in the previous layer and the weights connecting the nodes:

$$y_i = W_{ij} x_j + b_j \tag{3.1}$$

For notational convenience, this is written as a matrix multiplication:

$$\boldsymbol{y} = W\boldsymbol{x} + \boldsymbol{b}. \tag{3.2}$$

The resulting vector $\boldsymbol{y}$ is then passed through an *activation* function. There are various activations to choose from[1], e.g.:

- Sigmoid $\sigma_S = (1 + e^{-x})^{-1}$ function,
- Rectified Linear Unit $\mathrm{ReLU}(x) = \max(0, x)$,
- Leaky Rectified Linear Unit $\mathrm{LeakyReLU}(x|\alpha)) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{otherwise.} \end{cases}$

Thus, for an MLP comprising l layers, the output is given by:

$$F(x) = \sigma_l \left(\boldsymbol{b}_l + W_l \left(\sigma_{l-1} \left(... \left(\sigma_1 \left(\boldsymbol{b}_1 + W_1 \boldsymbol{x} \right) \right) \right) \right) \right), \tag{3.3}$$

where the input is a vector $\boldsymbol{x} \in \mathbb{R}^n$, the weight matrices are given by $W_k \in \mathbb{R}^{i,j},\ k \in \{1, .., l\}$ for the l layers and $\boldsymbol{b}_k \in \mathbb{R}^i$ are the biases corresponding to the layer k. The biases and weight matrices are referred to as *weights* or *parameters* in common NN literature. This creates a complex web of relationships and processing pathways that give MLPs their notable computational power. In the infinite capacity limit [64], i.e. number of hidden nodes $N \to \infty$, even a basic MLP with one hidden layer is a universal function approximator, which means that the MLP can approximate any imaginable function[2].
Note that the choice of the weight dimensions already offers significant flexibility in designing an architecture, as even a simple MLP with l layers has $l-2$ *hyperparameters*

[1] Note that these functions are applied element-wise, hence no vector formalism.
[2] To be precise: every **continuous** on a **compact** subset of $\mathbb{R}^n$

from its number of nodes per layer only[3]. It is worth noting that over-optimising these hyperparameters is a common pitfall in the development of machine learning solutions, as such optimisations can be extremely time- and computation-consuming.

3.1 Training of Neural Networks

Neural networks *learn* through iterative adjustments of their weights. In *supervised* training, labelled training data $(\boldsymbol{x}, \boldsymbol{y})$ is available, and the weights are adjusted such that the discrepancy between the prediction $\hat{\boldsymbol{y}} = NN(\boldsymbol{x})$ and the target values $\boldsymbol{y}$ is minimised. The function which quantifies this discrepancy between target and prediction is referred to as the *loss* function. To determine the adjustments for the weights to reduce the loss, an algorithm called *backpropagation* [65] is employed. Backpropagation computes the gradients of the loss function concerning the parameters of the model. This is a significant benefit of NNs, which allows training them with gradient descent.

3.1.1 Common Loss Functions

First, some frequently used loss functions are discussed. The loss functions are usually averaged over a *batch* of predictions containing n training samples.

Mean Squared Error The Mean Squared Error (MSE) is often used in regressions, where the target y is a real number:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2. \tag{3.4}$$

Binary Cross-Entropy Binary Cross-Entropy (BCE) is used in binary classification tasks, where the target are two possible classes $y = \{0, 1\}$ and the prediction[4]

[3]Generally, hyperparameters denote parameters that are chosen during the design of the NN and remain unchanged.

[4]Note that a Sigmoid activation is used after the output layer to obtain a number between 0 and 1.

$\hat{y} \in [0, 1]$:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]. \tag{3.5}$$

The BCE is used for classification, as it punishes confident wrong predictions, i.e. $y = 0, \hat{y} = 0.9$, significantly stronger than less confident ones, i.e. $y = 0$, $\hat{y} = 0.01$, due to the properties of the negative logarithm[5].

Cross-Entropy For multiclass classification tasks between M different classes, the cross-entropy loss is used:

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{n}\sum_{c=1}^{M} y_{ic} \log(\hat{y}_{ic}). \tag{3.6}$$

Note, now the prediction of the model and the target are vectors $\hat{\boldsymbol{y}} \in \mathbb{R}^M, \boldsymbol{y} \in \mathbb{R}^M$. The target label y_{ic} of class i is zero everywhere except for coordinate i, where it is one. For two classes, this is identical to BCE, since for a prediction $\hat{y}$ on class 0, the prediction on class 1 is given by $1 - \hat{y}$.

Focal Loss For notational convenience only BCE is considered in the following and the prediction on the true class p_t is defined as:

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise.} \end{cases} \tag{3.7}$$

Binary cross-entropy has two weak spots:

1. Class imbalance: if there is a large imbalance between the number of samples from different classes, pure cross-entropy will lead to a model that predicts the majority class more accurately, and disregards its accuracy on the minority class. This can be fixed by adding class weights to the loss terms arising from different class samples.

[5]Note the subtlety, although the prediction $\hat{y}$ can be arbitrarily close to zero, it is always a wrong prediction! A common misconception is to assume that everything below 0.5 would count as a correct prediction for the label $y = 0$.

2. No extra emphasis on hard examples: since BCE leads to a non-zero loss for terms that are easily classified, i.e. $p_t \gg 0.5$, simple examples lead to a significant contribution to the training loss. Thus, the optimisation can focus too much on easy examples, which comes as a trade-off for the performance on harder example. This can lead to a model that makes confident incorrect predictions, i.e. $p_t \ll 0.5$.

Lin et al. [66] propose the *Focal* loss to account for both of these weaknesses at the same time. Their proposed loss takes the form:

$$FL(p_t) = -\alpha_t(1-p_t)^\gamma \log p_t, \tag{3.8}$$

where $\alpha_t \in [0,1]$ and $\gamma \geq 0$ are parameters to account for the two problems mentioned previously. The former weights both classes such that over all samples they contribute on average the same loss at initialisation. It is defined similarly to p_t as:

$$\alpha_t = \begin{cases} \alpha & \text{if } y = 1 \\ 1-\alpha & \text{otherwise,} \end{cases} \tag{3.9}$$

where $\alpha = \frac{n_0}{n_0+n_1}$ with $n_{0/1}$ being the number of samples per class. The latter parameter γ down-weights the loss arising from confident correct predictions ($p_t \gg 0.5$). Figure 3.1 illustrates the effect on different choices of γ. This modification of the BCE loss is especially useful when used in HEP searches, where the goal is to identify a region of phase space pure in signal. For this, a binary classifier discriminating between signal and background events can be used. After its training, the signal region is defined as the region where the classifier prediction is high, e.g. $\hat{y} > 0.9$. Thus, there should be as little background as possible in this region, implying that there should be no confident incorrect predictions. If BCE is used, it can happen that the classifier enriches this supposedly signal pure region with background events. Note that in this thesis, the balancing parameter is set to $\alpha_t = 0.5$ everywhere. If there is an imbalance between the two classes, this is accounted for by oversampling the minority class with repetition.

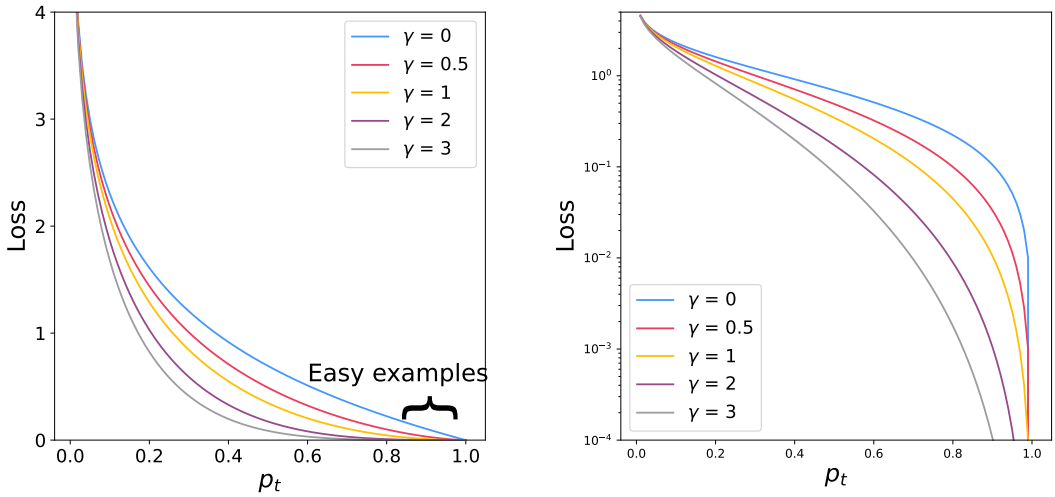


Figure 3.1: Comparison of the Focal loss for different choices of the parameter γ. Note that for $\gamma = 0$, Focal loss is equal to BCE. The left of the figure shows that confident correct predictions contribute a non-zero contribution to the loss. On the right, the loss is visualised on a log-scale to allow a comparison of different choices of γ.

3.1.2 Backpropagation

At its essence, backpropagation employs the chain rule to compute gradients of the loss function regarding the weights. The simple building blocks allow computing the derivatives analytically for each node, making this process feasible. The calculation starts from the output layer and propagates backwards through the network, hence the name backpropagation.

In the following, back propagation is described for a simple NN with a single hidden layer. The input is represented by $\boldsymbol{x}$, the hidden layer activations by $\boldsymbol{h}$, and the output[6] $\hat{y}$. The weight matrix connecting the input to the hidden layer is represented as W and the weight matrix connecting the hidden layer to the output as V. To abbreviate, the loss $\mathcal{L}(\hat{y}, y)$ is referred to as $\mathcal{L}$. The NN can be expressed as follows:

[6] $\hat{y}$ can also be a vector in general, but for simplicity, a scalar is chosen here.

$$\boldsymbol{h} = \sigma(W\boldsymbol{x} + \boldsymbol{b}) \tag{3.10}$$

$$\hat{y} = V\boldsymbol{h} + c \tag{3.11}$$

In this context, $\boldsymbol{b}$ and c are bias vectors[7], and σ is an element-wise activation function. Backpropagation computes the gradients of the loss function $\mathcal{L}(y, \hat{y})$ regarding the NN parameters (W, V, $\boldsymbol{b}$, and c) by using the chain rule starting from the output layer. The gradients of the loss function regarding the output layer parameters are given as:

$$\frac{\partial \mathcal{L}}{\partial V} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \boldsymbol{h}^T \tag{3.12}$$

$$\frac{\partial \mathcal{L}}{\partial c} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \tag{3.13}$$

To compute the gradient with respect to the parameters W of the hidden layer, it is necessary to compute the derivative of the loss with respect to $\boldsymbol{h}$:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}} = V^T \cdot \frac{\partial L}{\partial \hat{y}} \tag{3.14}$$

Finally, the partial derivatives of $\mathcal{L}$ with respect to W and $\boldsymbol{b}$ are determined.

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}} \frac{\partial \boldsymbol{h}}{\partial W} = (\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}} \odot \sigma'(W\boldsymbol{x} + \boldsymbol{b})) \cdot \boldsymbol{x}^T \tag{3.15}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{b}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}} \frac{\partial \boldsymbol{h}}{\partial b} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}} \odot \sigma'(W\boldsymbol{x} + \boldsymbol{b}) \tag{3.16}$$

Here, the symbol $\odot$ represents element-wise multiplication and σ' denotes the derivative of the activation function. This basic process is extended for an NN with L layers by repeating the computations for each layer, starting from the output layer and working backwards towards the input layer. In the following, the layer index is denoted with a superscript l, the forward pass is given by:

[7] Note that c is a scalar because the output was chosen to be one-dimensional.

$$\boldsymbol{h}^{[l]} = \sigma(W^{[l]}\boldsymbol{h}^{[l-1]} + \boldsymbol{b}^{[l]}) \tag{3.17}$$

$$\hat{y} = W^{[L]}\boldsymbol{h}^{[L-1]} + c \tag{3.18}$$

During the backward pass, the gradient of the loss function with respect to the weights of each layer is then given by:

$$\frac{\partial \mathcal{L}}{\partial W^{[l]}} = (\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}^{[l]}} \odot \sigma'(W^{[l]}\boldsymbol{h}^{[l-1]} + \boldsymbol{b}^{[l]})) \cdot (\boldsymbol{h}^{[l-1]})^T \tag{3.19}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{b}^{[l]}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}^{[l]}} \odot \sigma'(W^{[l]}\boldsymbol{h}^{[l-1]} + \boldsymbol{b}^{[l]}) \tag{3.20}$$

This process is recursively performed, where $\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}^{[l]}}$ is computed for the hidden layers as:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}^{[l]}} = (W^{[l+1]})^T \cdot \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}^{[l+1]}} \tag{3.21}$$

As this only outlines the case for a simple fully connected MLP, interested readers should consult the previously mentioned textbooks [58, 62] on for the general case.

3.1.3 Automatic Differentiation Frameworks

The elementary operations used in NNs are simple operations with known derivatives, making *Automatic Differentiation* frameworks valuable. Since only the PyTorch [67] framework is used in this thesis, the focus lies on this framework in the following. In PyTorch, the computation of gradients is separated into two *passes*. First, during the *forward* pass, each operation that is performed, dynamically creates a computational graph. The computational graph represents how the input moves through the NN. It contains two types of *nodes*:

1. *operational* nodes representing the mathematical operations,

2. *variable* nodes representing the inputs and weights on which the operations

are performed.

The other components of the computational graph are *edges*, which represent the dependencies between the operational and variable nodes. During the forward pass, the intermediate results are stored, as Eq. 3.12 shows that these are needed to calculate the gradients. The *backward* pass computes calculates the gradients of the loss with respect to the parameters of the NN as described above. The value of these frameworks is precisely that the computations for the gradients are done automatically and efficiently, which eases the optimisation of NN significantly.

3.1.4 Optimisation

After computing the gradients of the loss function, the optimisation process begins. In principle, adjusting the weights using a small step size, referred to as the *learning rate* η, in the opposite direction of gradients reduces the loss. A widely used algorithm is Stochastic Gradient Descent (SGD), for which the basic idea has been proposed already in the 1950s [68]. In SGD, the optimisation is done in batches, over which the gradients are averaged. This not only accelerates the optimisation, but also reduces the variance on the effective gradients. Note that NNs became especially popular due to the emergence of Graphical Processor Units (GPUs), allowing for parallelisation of matrix multiplications, significantly reducing the training and inference time. SGD exhibits several limitations, such as slow convergence and a strong susceptibility to the learning rate. To overcome some of these issues, momentum [65], RMSprop [69] and Adagrad [70] were proposed. RMSprop, despite its emergence from machine learning lecture notes by Geoffrey Hinton[8], became one of the most popular used optimisation algorithms. Hinton's proposal involves adjusting the learning rate based on the square root of the moving average of the gradients. AdaGrad adjusts the learning rate for each model parameter individually. It does this by using a factor that depends on the square root of the accumulated squared gradients[9]. In 2015, Knigma et al. [72] proposed the combination of both under the name of Adam, which stands for Adaptive Moment estimation. There are some additional subtle

[8]Note that the algorithm corrects the RPROP [71] rendering it suited for SGD.

[9]Note that this factor goes to zero, as the squared sum of gradients goes to zero. This also improved in the Adam [72] algorithm, mentioned in the following.

differences between the individual parts (e.g. Adam uses a bias-corrected estimate of the moments of the gradients, but these are highlighted in detail in Ref. [72]). Adam comes with two important hyperparameters additional to the usual learning rate: β_1 and β_2, where the former controls the exponential decay of the first moment of the gradients, and the latter controls the decay of the second moment. The full algorithm is given in Algorithm 1 for completeness. Most notably, Adam uses the running mean and variance of the gradients to update the weights. An appealing interpretation, especially for physicists, of the Adam optimisation is given by the interpretation of a rolling ball running down a slope with friction; for more details consult Ref. [73], where the former parameter represents the momentum, and the latter parameter is the friction of the ball. Crucially, Adam defines different friction parameters for the different weights. For weights receiving large gradients, the value of the friction is increased.

Algorithm 1 Adam: Adaptive Moment Estimation

Require: α: Step size
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

Linear Warm-up Cosine Annealing Scheduling

Something that proved to improve all models in this thesis was the use of a learning rate scheduler, which changes the learning rate during the training according to a

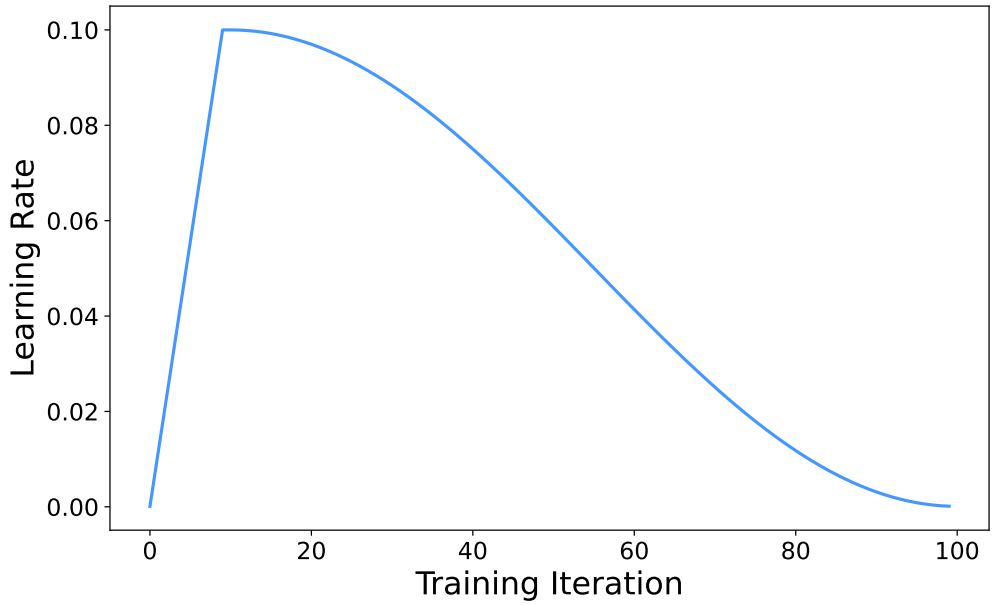


Figure 3.2: The learning rate as a function of the training iterations defined by the linear warm-up cosine annealing learning rate scheduler.

selected schedule. The linear warm-up cosine annealing schedule [74], which is shown in Fig. 3.2, is used for every training, unless stated otherwise.

Regularisation

Deep NNs often have a high number of parameters compared to the size of the available training data. From classical machine learning, one could expect that the high parameter-to-data ratio results in *overfitting*, where the model fits the target function to the training data too excessively and thereby captures noise and outliers, leading to poor generalisation performance on unseen data. For NNs it is not exactly as problematic, since some regularising effects arise from the training and NN structure. Especially interesting is the *double descent* phenomenon [75], which demonstrates that the generalisation error first decreases with the model size, then increases, showing overfitting, and then decreases to lower values than before. However, to actively counteract overfitting, regularisation techniques can be employed. Regularisation introduces a penalty for NN complexity into the loss

function, effectively reducing the number of free parameters in the model. Two common forms of regularisation are L1 and L2 regularisation. Both add a term to the loss function that is proportional to the sum of the absolute values (L1) or squares (L2) of the model weights. The Adam optimiser, a popular choice for training deep learning models, includes a form of L2 regularisation. Nevertheless, Loshchilov et al. [76] argue that the regularisation of Adam is not entirely effective, thus proposing a revised version named AdamW. In the work performed in this thesis, no significant differences were found between different optimisers, thus, if not stated otherwise, AdamW is used.

dropout [77] is another common regularisation strategy. During training, dropout randomly nullifies the output of the nodes in a layer with a probability p at each update. This forces the NN to encode information redundantly. During evaluation, all nodes are usually used, and their outputs are scaled by $\frac{1}{1-p}$ to account for the missing contributions during training. Goodfellow et al. [62] offer an intuitive explanation of dropout, describing it as an inexpensive approximation to training and evaluating a bagged ensemble of exponentially many NNs.

To assess whether a model is overfitting, a common practice is to split the available data into *training*, *validation*, and *test* sets. The training set serves to train the model, the validation set to tune hyperparameters and select the top-performing model, and the test set, often also referred to *hold-out* set, to measure the performance of the model chosen from the hyperparameter optimisation. Since the training is not always stable, the validation set is also used to select the best *checkpoint* during the training, i.e. the best configuration of weights, which is used for testing.

If a model exhibits good performance on the training set but performs poorly on the validation set, it is likely overfitting the training data. This performance discrepancy suggests that the capacity of the model is excessive, and the regularisation strength needs to be increased. Conversely, if the model performs poorly on both the training and validation sets, it is under-fitting the data, indicating that the capacity of the model is insufficient, or the regularisation strength is too large. Apart from these techniques, additional strategies for managing overfitting include gathering more training data, using data augmentation techniques or employing simpler architectures with fewer parameters.

3.1.5 Weight Initialisation

While the study of weight initialisation for NNs boasts an extensive history, the current discussion will primarily focus on what impacted or came up in the work of this thesis. Fundamentally, a critical objective in weight initialisation is to maintain the mean at zero and variance around unity across the different layers in the NN. The variance $\mathbb{V}$ of a random variable X is given by:

$$\mathbb{V} = \mathbb{E}(X - \mathbb{E}(X))^2,$$

where $\mathbb{E}$ is the expectation value. The variance should be around unity to prevent two potential failure scenarios:

1. If the variance consistently diminishes across the layers, a *vanishing gradient* problem emerges. In this scenario, the variance of the gradients progressively approaches zero as the input propagates backwards through the NN.

2. If the variance significantly exceeds 1, its magnitude may escalate uncontrollably, potentially triggering overflow errors on the computation backend.

The LeCun initialisation [78] was proposed in the late 1990s, where weights are drawn from a Gaussian distribution with a mean of zero and a standard deviation:

$$\sigma = n^{-1}, \tag{3.22}$$

where n represents the number of nodes in a layer. The LeCun initialisation ensures that the variance of the input remains around one during the forward propagation, but it does not address the issue of potential gradient vanishing during backpropagation. To resolve this, Glorot et al. proposed the Xavier initialisation [79] in 2010, which initializes the weights from a normal distribution with a standard deviation of:

$$\sigma = \frac{2}{n_{\text{in}} + n_{\text{out}}}, \tag{3.23}$$

where n_{in} and n_{out} denote the number of incoming and outgoing connections[10]. Tra-

[10] Although initially proposed for a uniform distribution U, the normal distribution is now commonly used. For the uniform distribution, the weights are drawn from $U(\left[-\sqrt{\frac{6}{n_{\text{in}}+n_{\text{out}}}}, \sqrt{\frac{6}{n_{\text{in}}+n_{\text{out}}}}\right])$

ditionally, sigmoid and tanh functions have been the preferred choice for activations, but they suffer from *saturation*. When subjected to high absolute input values, they saturate and their derivatives approach zero, which also causes the gradient to vanish. To further address the saturation problem, ReLUs were introduced. However, considering the strictly positive nature of ReLU, the mean can no longer be assumed to be zero. Consequently, in 2015, He et al. proposed an initialisation method similar to Xavier initialisation in Ref. [80], where the weights of each layer are drawn from a Uniform distribution $U(-\frac{1}{n}, \frac{1}{n})$: This is the default that is used in the PyTorch library [67]. Thus, if not stated otherwise, this initialisation is used.

3.1.6 Preprocessing

Note that a crucial detail that was assumed in the previous discussion is that the input in the first layer of the NN is normally distributed. This is achieved through *preprocessing*, where the data is transformed to follow a Normal distribution. But note, that preprocessing typically transforms the features independently, hence their input x and the parameters of the transformation are denoted as scalars. Two transformations that are commonly used are:

1. *Standard Scaling*: The mean is subtracted from the data and divided by its standard deviation:
$$x \to \frac{x - \mu}{\sigma}, \tag{3.24}$$
where the mean and the standard deviation is estimated from the training data. This is an invertible transformation with the only requirement that the standard deviation is not zero. It is a good choice if the data already has a Gaussian distribution.

2. *Box-Cox Scaling* [81]: this transformation is especially suited if the data is right-skewed. It is given as a parametric transformation
$$x \to \begin{cases} \frac{x^\lambda - 1}{\lambda} & \text{if } \lambda > 0 \\ \log(x) & \text{else,} \end{cases} \tag{3.25}$$
where λ is determined on the training dataset. It is also invertible, but requires

that $x > 0$.

3.1.7 Batch Normalisation

Rather than just applying the preprocessing in the input layer, batch normalisation, often referred to as *BatchNorm*, aims to standardise the latent features in the hidden layers. Note again that this normalisation is done independently for every feature over a training batch. The hidden representation z of the data is normalised by the mean μ_b and the standard deviation σ_b of the batch. BatchNorm introduces two trainable parameters, a scale γ and a shift β:

$$\boldsymbol{x} \to \boldsymbol{\gamma}\left(\frac{\boldsymbol{x}-\boldsymbol{\mu}_b}{\sqrt{\boldsymbol{\sigma}_b^2+\epsilon}}\right)+\boldsymbol{\beta}, \tag{3.26}$$

and ϵ is a small number for numerical stability. For inference, the mean and standard deviation are set to the running mean that is accumulated during training.

3.1.8 Residual Connections

There is plenty of evidence to suggest [82, 83, 84, 85] why *deeper* and *narrow*, i.e. NNs with a larger number of hidden layers and few nodes per layer, outperform *shallow* but *wide* ones, i.e. with few layers and many nodes per layer. Two possible explanations are:

1. Hierarchical feature learning: deeper architectures can learn hierarchical representations of the input data, e.g. image classification, see Ref. [86].

2. Parameter efficiency: from a theoretical perspective, it can be shown that deep architectures can provide functions which cannot be approximated by shallow but wide architectures, given the same number of parameters [85].

As previously mentioned in Section 3.1.5, deeper architectures bring problems of exploding or vanishing gradients. The initialisation solutions are only valid for the special case of a fully connected MLP. In general, architectures can be significantly more complex. To resolve this, He et al. [84]. proposed a deep learning architecture

known as Deep Residual Networks (ResNets) in 2015. In their study, they demonstrated that deeper architectures do not necessarily outperform shallower ones, even when additional techniques are employed to mitigate the vanishing gradient problem (e.g. intermediate normalisation layers). They demonstrate a *degradation* property of deep architectures that is independent of overfitting, as also the training error increases compared to a more shallow architecture. They conclude that deeper architectures are more difficult to optimise than more shallow ones. To address this issue, they introduce residual connections[11], which enable the gradients to backpropagate directly through several layers. Instead of learning the underlying mapping for the output $y = H(x)$, for a group of layers directly, the *residual* between the input and the output, $F(x) = H(x) - x$, is learned. Therefore, the original function becomes $H(x) = F(x) + x$. For a 2-layer block with weight matrices W_1, W_2, and activation σ the corresponding equation of the block is given by[12]:

$$y = \sigma(\boldsymbol{x} + W_2(\sigma(W_1\boldsymbol{x}))). \tag{3.27}$$

The authors illustrate the benefit of the residual connection with Fig. 3.3 on the ImageNet [88] dataset, a visual database containing over 14 million labelled images. In the left figure, they do not use residual connections, resulting in a higher training error for the deeper architecture. This is counterintuitive because in principle the shallow architecture is contained in the deeper architecture, and thus they should perform at least equally. They conclude that the difference must lie in the optimisation. On the right, the authors show that the performance significantly improves when the residual connections are included. Note that residual connections are non-parametric, and thus no additional parameters are introduced by this method. Starting from the argument that the residual is easier to learn, it seems counterintuitive that the activation is applied after the addition. Correspondingly, He et al. [89] then later proposed to reformulate the residual connection to facilitate the propagation of information. Apart from finding that an identity shortcut outperforms other gating mechanisms, their main conclusion is that applying the activation before W_1 and

[11] Residual connections are a specific type of *skip connections*. In a residual *block*, the input to the block is added to the output of the block. Skip connection, on the other hand, can be arbitrary shortcuts through the NN (e.g. as in the popular architecture U-Net [87].

[12] The biases are omitted for simplicity

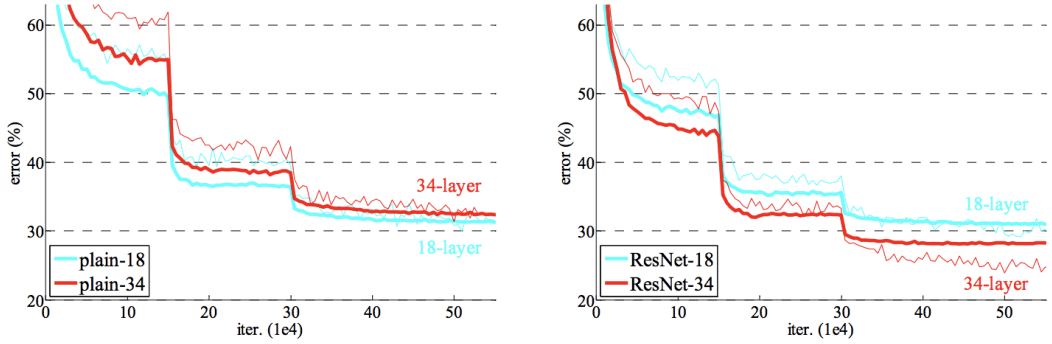


Figure 3.3: Training (thin lines) and validation (bold lines) error when comparing architectures of different depth (18 (blue) vs 34 layers (red). (left) Deeper architectures have a higher training and validation loss, which is not expected. (right) Adding residual connections, the performance of the deeper architectures excels the performance of the more shallow NN. Figure taken from Ref. [84].

not in the main "stream" improves the performance. So in Eq. 3.27 the activation σ is moved such that the equation yields:

$$\boldsymbol{y} = \boldsymbol{x} + W_2(\sigma(W_1\sigma(\boldsymbol{x}))) \tag{3.28}$$

Note, that during backpropagation the additional identity cures the problem of vanishing of gradients, as its derivative is always at least 1. They illustrate the supremacy of the new approach with Fig. 3.4, where they study the performance on CIFAR-10 [90], another image-based dataset. Note that although the references provided primarily focus on image-based problems, the same conclusions are drawn in many other fields (e.g. NLP [61], where also a preactivation residual connection is employed; also empirical findings during this thesis). Thus, in the following, unless stated otherwise, this setup is used for residual connections.

3.2 Symmetries

In the development of machine learning architectures, a crucial principle is integrating prior knowledge. Such knowledge is usually manifested as symmetries and invariances that should be mirrored in the learned representations. The complexity can be significantly reduced by identifying and utilising such symmetries, which in turn

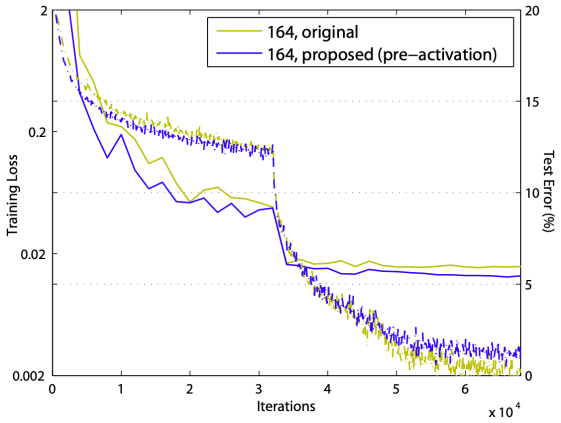


Figure 3.4: Performance comparison of a *post-activation* (green) residual connection, to the previously proposed *pre-activation* (blue) setup. Dashed lines represent the training loss, and the solid lanes represent the performance during testing. They conclude that it overfits less, and consistently outperforms the initially proposed setup. Figure taken from Ref. [89]

improves the generalisation performance of the model [91]. Weight sharing is a method commonly used to introduce symmetries into NN architectures. Convolutional neural networks (CNNs) provide a primary example of this, wherein the same filter is applied across all different spatial locations in the input image, resulting in translational invariance [86]. Intuitively, this is assuming that informative features extracted from a specific spatial position should also be recognized at other spatial positions, e.g. a pair of glasses at position x, y in an image, should be treated equally at position x', y'.

3.2.1 Invariances and Equivariances

Symmetries can be included in an NN, described as a function $\boldsymbol{f}$ in the following, by making the NN obey certain invariances or equivariances. A function is considered *invariant* under a symmetry transformation when the application of the transformation to the input of the function leaves its output unchanged. More formally, let $\boldsymbol{f}$ be a function and T the representation of a symmetry transformation g, then $\boldsymbol{f}$ is

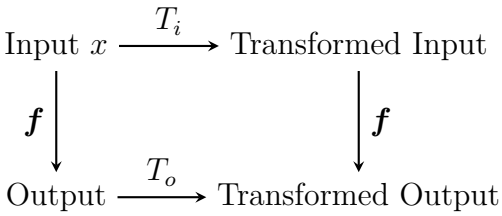


Figure 3.5: This flow chart demonstrates the principle of equivariance. When a function $\boldsymbol{f}$ is equivariant to a symmetry transformation $T_{i/o}$, the order of operations is interchangeable without affecting the outcome. Note that the index denotes the representation of the symmetry transformation on the input and output space. One can either apply T_i to the input and then $\boldsymbol{f}$, or first map the input to the output using $\boldsymbol{f}$ and subsequently apply T_o. Both paths yield the same result.

said to be invariant under T if

$$\boldsymbol{f}(T(\boldsymbol{x})) = \boldsymbol{f}(\boldsymbol{x}), \quad \forall \boldsymbol{x}. \tag{3.29}$$

An example from physics is the invariant mass of a jet, which remains the unchanged under Lorentz transformations. Conversely, *equivariance* refers to the output of a function transforming in the same manner as the input under the given transformation. Formally, $\boldsymbol{f}$ is equivariant to T if

$$\boldsymbol{f}(T_i(\boldsymbol{x})) = T_o(\boldsymbol{f}(\boldsymbol{x})), \quad \forall \boldsymbol{x}. \tag{3.30}$$

This concept is illustrated in Fig. 3.5.
Permutation equivariance is a particularly relevant symmetry for HEP. For instance, jets are naturally represented as point clouds, which are effectively sets of unordered elements. Given that the underlying physics is independent of the order, permutational symmetry is crucial. If a permutation equivariant n-dimensional function is approximated with a NN respecting permutation symmetry, the effective variance the model needs to explain, is a factor $n!$ smaller. This reduction occurs because the model already satisfies the requirement of being invariant to the $n!$ possible permutations of the input by construction.

3.2.2 DeepSets

For permutation-invariant tasks, i.e. where the input is a set, a simple yet powerful architecture is introduced in Ref. [92], where the authors propose the *DeepSets* aggregation. For the DeepSets aggregation illustrated in Fig. 3.6, every element of the input set is mapped to a latent representation with the same MLP. These latent representations, of every element of the input are summed[13]. The authors of *DeepSets* claim that this aggregation makes the NN fulfil a universal approximation theorem for function on sets. Later, Wagstaff et al. [93], provide a necessary and sufficient condition such that the universal approximation holds. Namely, the dimension of the latent space, where the sum pooling is carried out, is equal to the maximum set cardinality times the features per set element, which for HEP can go to $\sim \mathcal{O}(10^5)$, as discussed in Chapter 6. However, this is a theoretical argument, and empirically models that make use of such a permutation-equivariant information aggregation method have shown state-of-the-art results [94, 95, 96], even when the latent space has a smaller dimension than required.

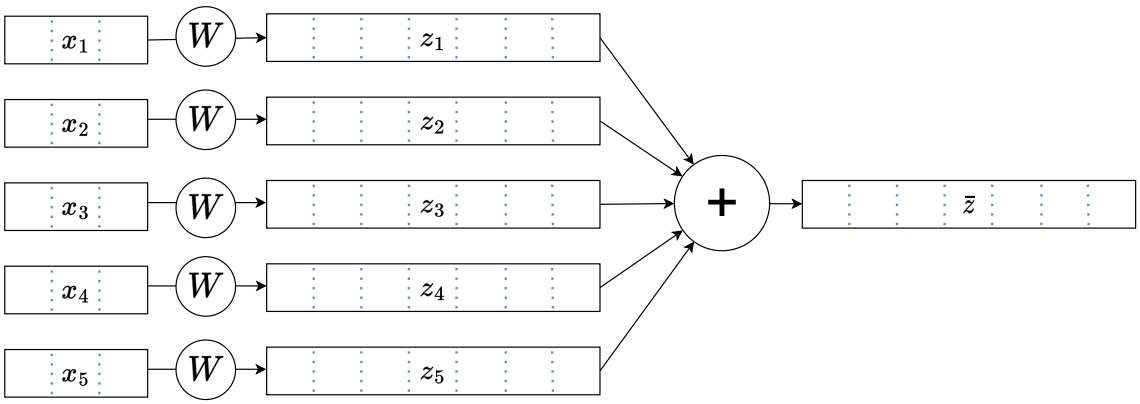


Figure 3.6: Visualisation of the DeepSets approach. Every set element is embedded in a latent space, where a sum pooling is applied. Note that this is permutation invariant, since the sum is a permutation invariant operation.

[13]This is referred to as *sum-pooling*. Generally, *pooling* operations are a useful tool to achieve invariances as they abstract away from specific details.

3.3 Attention

(Self-)attention was popularised in 2017 by Vaswani et al. [61], who demonstrated that their attention-based *Transformer* model outperforms the state-of-the-art with significantly lower training costs. The attention aggregation is primarily inspired by human cognitive processes, where the brain selectively focuses on certain parts of its incoming information to process it more efficiently. In the context of machine learning models, attention enables the model to consider inputs non-uniformly. This is best illustrated in NLP tasks; in a sequence-to-sequence model featuring attention, the model uses the attention mechanism to evaluate the relevance of each word in the input sentence when creating an output, e.g a translated word. Attention allows the model to focus on words that are more important in the context. Typically, the input of attention is denoted as *query* Q, *key* K and *value* V[14]. In practice, the query, key, and value are a sequence of *embedded tokens*. The tokenisation refers to breaking the input down into its smallest building blocks, e.g. the phrase "I like lollipops." can be tokenised as ["I", "*like*", "*lollipops*", "."], but note that the tokenisation is not unique. Each of these tokens is then transformed into a numerical representation in some high-dimensional space, a process known as *embedding*. *Self-attention* refers to the special case where the three embedded sequences x, y, z are the same, and cross-attention to the case with two different sources. The query comes from one sequence, and the key and value are extracted from another sequence.

3.3.1 Softmax

Before describing the attention mechanism with formulas, the *softmax* function needs to be introduced. It takes as input a vector of K real numbers and normalises them to add up to one. This function is widely used in machine learning algorithms, especially in the output layers of multiclass classification models, where it serves as a means for converting raw model outputs, which are real numbers, into values that can be interpreted as the probability of the classes. The softmax function σ for an

[14] These names originate from the field of information retrieval, specifically in database indexing and searches. When data is searched in a database, a query is usually submitted. The system uses this query to look up a key in its index. The key then maps to a value which contains the data which is looked for.

input vector $\boldsymbol{x} \in \mathbb{R}^N$ is defined as follows:

$$\sigma_i(\boldsymbol{x}) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{3.31}$$

Note that for two classes, the output of the softmax is equivalent to the output of a sigmoid (respective 1-sigmoid).

3.3.2 Scaled-Dot Product Attention

Formally the query, key, and value are given by different f-dimensional embeddings $x = (\boldsymbol{x}_i)_{i=1}^n, y = (\boldsymbol{y}_i)_{i=1}^l, z = (\boldsymbol{z}_i)_{i=1}^l$ are first mapped to hidden representation[15]:

$$Q = xW_Q \quad x \in \mathbb{R}^{n \times f}, W_Q \in \mathbb{R}^{f \times d} \tag{3.32}$$

$$K = yW_K \quad y \in \mathbb{R}^{l \times f}, W_K \in \mathbb{R}^{f \times d} \tag{3.33}$$

$$V = zW_V \quad z \in \mathbb{R}^{l \times f}, W_V \in \mathbb{R}^{f \times d} \tag{3.34}$$

where n/l is the number of sequence elements, d the number of features of the latent representation and f the number of features of the embedding. The formula for the attention aggregation is then given by[16]:

$$\text{Attention}(Q, K, V) = \boldsymbol{\sigma}\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{3.35}$$

The scaling with $\sqrt{d}$ is introduced to treat the saturation of the softmax as the variance of the dot product of two d-dimensional normally distributed random vectors is proportional to d. An illustration of the most general scaled-dot product attention is given in Fig. 3.7

3.3.3 Multi-Headed Attention

In the same paper [61], the advantages of computing attention across multiple independent linear representations of the input are demonstrated. Each representation

[15] Note that the key and value sequence are required to have the same length.

[16] Note that the latent dimension of the key and query are required to match, but the dimension of the value can be different but is usually chosen to be the same.

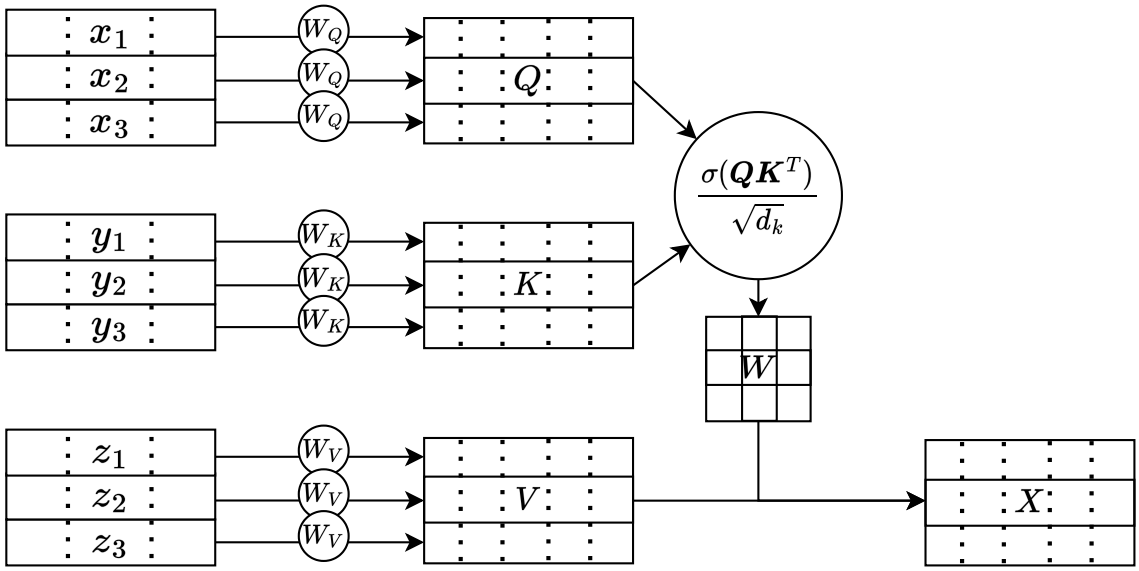


Figure 3.7: The attention mechanism; given 3 embedded input sequences x, y, z (note that y, z, need to have the same number of tokens, but x can in principle be of different length), each sequence is mapped to a (usually) higher dimension and stacked to form the matrices Q, K, V. Then, the softmax is applied to the scaled cross product of Q, K resulting in a square weight matrix W where all columns add up to 1. Multiplying this weight matrix W with V yields the output X of the attention aggregation.

is referred to as *head*. Specifically, rather than applying attention aggregation over a dimension d, attention is computed over h heads, each with a dimension of[17] $\frac{d}{h}$. This approach maintains the total number of trainable weights unchanged due to the reduced dimension per head, and the computational cost remains comparable. The outputs from the different heads are then concatenated:

$$\text{MultiHeadedAttention}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, ..., \text{head}_h)W^O \tag{3.36}$$

$$\text{where head}_i = \text{Attention}(xW_i^Q, yKW_i^K, zW_i^V), \tag{3.37}$$

where W^O is a projection matrix that unifies the output of the different heads. Notably, all well-known Large Language Models (LLMs), such as ChatGPT [97] and LLama [98], are based on the transformer architecture. Recently, attention mechanisms have expanded into a broader array of applications, including but not limited to computer vision [99] and reinforcement learning tasks [100].

3.3.4 Permutation Equivariance of Self-Attention

Note that in the previous section the input of the self-attention was always referred to as a sequence, which was historically motivated. But attention itself is permutation-equivariant as can be shown as follows. Given a permutation matrix P, applying it to the input sequence it follows::

$$\text{Attention}(PQ, PK, PV) = \boldsymbol{\sigma}\left(\frac{PQK^TP^T}{\sqrt{d}}\right)PV \tag{3.38}$$

Since the softmax is applied row-wise, the (row-)permutation matrices can be pulled out of the softmax. And since the permutation matrices fulfil $P^T = P^{-1}$, it follows $PP^T = \mathcal{I}$, and thus:

[17] This requires $d \bmod h = 0$.

$$\text{Attention}(PQ, PK, PV) = \boldsymbol{\sigma}\left(\frac{PQK^T P^T}{\sqrt{d_k}}\right) PV \tag{3.39}$$

$$= P \cdot \boldsymbol{\sigma}\left(\frac{QK^T}{\sqrt{d_k}}\right) P^T PV \tag{3.40}$$

$$= P \cdot \text{Attention}(Q, K, V) \tag{3.41}$$

□

Thus, the input might as well be a set. This is what makes self-attention well-suited for tasks in HEP, where the data can be represented as variable-sized sets of particles[18]. Furthermore, some elements of the sets are expected to have significantly more impact in the context of the underlying physics, e.g. high-energetic decay products carry more information about the parent particle than low-energetic ones.

3.3.5 Masking

Dealing with variable-sized inputs can be difficult with NN, as the matrices require a fixed input dimension. In the attention mechanism, this is alleviated by a clever use of the NN. A subtlety in the attention mechanism is that the trainable parameters of attention layers are applied to the tokens independently, which can be seen by the dimension of the matrices W_Q, W_V, W_K.
The interaction between tokens in the attention mechanism is introduced by the dot product in the softmax and the following multiplication with the value matrix. Thus, the inputs are padded with zeros to the same length, which allows calculating the attention weights efficiently. The added zeros are *masked* by adding a $-M \cdot \infty$ term in the softmax calculation. The mask is 1 if a token is padded and not present in the input. This will result in a zero as the output of the softmax and, as such, not have an impact on the output of the attention aggregation.

[18] Indeed in NLP tasks a *positional encoding* is explicitly introduced to account for the ordering in the input.

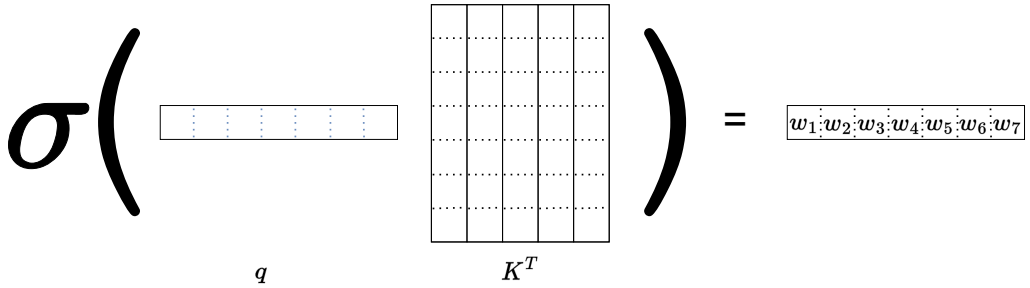


Figure 3.8: Connection between attention and DeepSets: considering only one element of the query (for more than one row in the query, the output is just stacked, but the principle remains the same). In the figure, the operation inside the softmax is illustrated: a $1 \times f$ matrix times a $f \times n$ matrix results in a $1 \times n$ matrix. Applying the softmax normalises the result to sum up to one $\sum_{i=1}^{f} w_i = 1$. Now when multiplying the output of the softmax together with the embedded input set V, it is equivalent to $x'_i = \sum_{j=1}^{n} w_j \boldsymbol{v}_i$, which is just a weighted sum of the latent representation of the input.

3.3.6 Relation to DeepSets

There is a subtle similarity between attention and the DeepSets approach when the query q contains only one token. This is illustrated in Fig. 3.8, as it is easier to understand this connection visually. The self-attention aggregation can be identified as a weighted sum of the individual representations of the value tokens. The weights are determined by the cosine similarity, i.e. the dot product, normalised by the magnitude of both vectors, between the query and the keys. Note that the softmax brings an important stabilisation to the whole aggregation. Since the weights add up to one, the scale of the output is independent of the number of inputs. Assuming the input is normally distributed, naive summing of these latent representations leads to larger variance for sets with high cardinality. This is explained by the Central Limit Theorem, stating that the variance of the resulting distribution is increased by a factor of n, where n is the number of elements in the sum. Normalising the weights to unity counteracts this. However, the normalisation of the weights leads to attention being agnostic to the cardinality of its input. This is a crucial realisation that was found during the study of point cloud datasets, which will be discussed in Chapter 5.

3.3.7 Linearised Attention in NLP

Since in the attention aggregation, the pairwise interaction between all tokens are computed, the computational complexity of attention scales quadratically $\mathcal{O}(n^2)$ with the number of inputs, as well as the required memory. This quadratic scaling of attention is also an issue for NLP, where the long-range dependence between the different words is of large importance for the context. There have been various propositions to deal with this, e.g. Refs. [101, 102, 103, 104, 105]. In the following, only `FlashAttention` [104, 105] is briefly discussed as it raises an important point, which is easily overlooked. The authors highlight that the quadratic scaling of attention is not problematic due to the number of Floating-Point Operations Per Second (FLOPS), which measure the complexity of an algorithm, but rather by loading and saving intermediate results. The authors clarify the difference between *compute*-bound operations and *memory*-bound operations and conclude that attention is memory-bound[19]. In other words, this means that the operations limiting the speed are simple operations, which are applied element-wise. The authors propose to fuse these operations to reduce memory use. However, as of the time of writing this, no ready-to-use implementation is available for the more common GPUs at DESY, but this certainly is an interesting direction for future work.

[19]The interested reader should consult [104], where an illustrative explanation is provided.

CHAPTER 4

Generative Modelling

This chapter introduces various approaches to Generative Modelling, beginning with Variational Auto-Encoders (VAEs) and Generative Adversarial Networks (GANs), and then moving on to more recent methods such as Normalising Flows (NFs), Diffusion Models, and Continuous Normalising Flows (CNFs) trained with Conditional Flow Matching (CFM). A personal goal during my thesis was to experiment with every major class of generative models, to gain a comprehensive understanding. For each approach discussed, its performance on a simple toy problem, which is introduced in the following section, will also be presented. The repository to reproduce these results is available on GitHub [106], and is encouraged to be used as a template for generative modelling tasks. In the following, *real* data is used to refer to data from the target distribution that should be learned. In contrast, data drawn from the model is referred to as *synthetic* data.

4.1 Two Moons Dataset

The Two Moons Dataset, depicted in Fig. 4.1, is a two-dimensional dataset that was used in this thesis to briefly experiment with different approaches to generative modelling. It consists of two crescent-shaped components that are not connected. Converting a distribution over a connected support to a distribution over two disconnected components is not a simple task, as it requires the function to be

non-continuous[1]. The Two Moons are constructed as:

$$(x, y) = (\cos(\theta), \sin(\theta)) + \begin{cases} (0, 0) & \theta < \pi \\ (1, -0.5) & \pi < \theta < 2\pi \end{cases} \tag{4.1}$$

for θ in the range of $[0, 2\pi]$. Finally, Gaussian noise $\mathcal{N}(0, \sigma^2)$ is added to each point, where σ was chosen as 0.05 for the following experiments.
One benefit of choosing two-dimensional data is that the learned distribution can be visualised with a two-dimensional histogram, allowing a visual evaluation of the model performance, without sacrificing information due to marginalising. All models in the following are trained for 50 epochs and 1000 iterations per epoch[2]. The batch size is chosen to be 256 for all models. All models rely on the same NN architecture, which is embedded in the different generative models. This means that there is no architectural difference between a VAE Encoder and a GAN generator, the only potential difference lies in the output layer, which is defined by the specific requirements of the different models. All models are constructed such that they have approximately the same number of parameters[3]. Due to the unstable training, first a small hyperparameter optimisation for the GAN was conducted, where the number of layers and nodes per layer and the learning rate were optimised. These parameters were then also selected for the configuration of all NNs in the other models.

4.2 Variational Auto-Encoders

VAEs [107], proposed by Kingma and Welling in 2013, represent a significant advancement in deep generative modelling. To explain their workings, it is best to start with traditional Auto-Encoders (AEs). The training paradigm of an AE is simple: first, the data is encoded to a lower dimensional latent space with an NN $\boldsymbol{E}(\boldsymbol{x})$.

[1] Usually a simple connected distribution e.g. a standard Gaussian distribution is used as input for the generative model.

[2] Except for the Continuous Normalising Flow described in Section 4.5, which only used 100 iterations per epoch, as otherwise, the training takes more than 24 hours. For the CNF, a different NN architecture is chosen, as its construction is more complicated.

[3] This is slightly more difficult for NF-based on coupling layers, since they have one NN per coupling layer. To account for this, the number of hidden features was reduced such that in total they have the same number of parameters as the other models

Figure 4.1: Histogram of 100'000 samples of the Two Moons dataset.

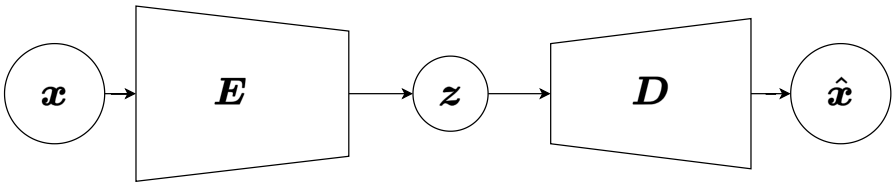


Figure 4.2: A VAE consists of an encoder $\boldsymbol{E}$ and a decoder $\boldsymbol{D}$. The encoder encodes the input data to a typically lower dimensional latent *code* $\boldsymbol{z}$. The decoder produces the reconstruction $\hat{\boldsymbol{x}}$ as a function of the latent code.

Then this representation is decoded with another NN $\hat{\boldsymbol{x}} = \boldsymbol{D}(\boldsymbol{z})$. To train the two networks the following loss, usually referred to as reconstruction error, is minimised:

$$\mathcal{L}_{AE} = ||\boldsymbol{x} - \boldsymbol{D}\left(\boldsymbol{E}(\boldsymbol{x})\right)||^2 \tag{4.2}$$

An AE can also act as a generative model by sampling the lower-dimensional encoding space and passing it through the decoder. In this case, the encoder is typically not used after training is complete[4]. To achieve this, the distribution of the input in the latent space should be easy to sample from. For an AE, there is no incentive to have a connected or continuous topology in the latent space. This is where the VAE comes into play; instead of deterministic encoding, the encoding becomes probabilistic and the encoder is optimised so that the encoded inputs follow a standard Gaussian

[4]Note that this is not always the case. For example, denoising images requires both models after training [108]

distribution.
The VAE provides a mathematically rigorous methodology for learning complex data distributions, introduced in the following: The encoder $\boldsymbol{E}$ encodes the input $\boldsymbol{x} \in \mathbb{R}^n$ into the latent *code* $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$[5] where $\boldsymbol{z} \in \mathbb{R}^m,\ m < n$. The index ϕ highlights that this is dependent on the parameters $\boldsymbol{\phi}$ of the encoder NN. In practice, this is done by using the output of the encoder $\boldsymbol{E}(\boldsymbol{x})$ as parameters of a probability distribution, typically a Gaussian with mean $\boldsymbol{E}_{\boldsymbol{\mu}}$ and diagonal covariance matrix[6], where the diagonal is given by ${\boldsymbol{E}_{\boldsymbol{\sigma}}}^2$:

$$\begin{bmatrix} \boldsymbol{E}_{\boldsymbol{\mu}} \\ \boldsymbol{E}_{\boldsymbol{\sigma}} \end{bmatrix} = \boldsymbol{E}(\mathbf{x}) \tag{4.3}$$

A point $\boldsymbol{z}$ is then sampled from this distribution using the reparametrisation trick:

$$\boldsymbol{z} = \boldsymbol{E}_{\boldsymbol{\mu}} + \boldsymbol{E}_{\boldsymbol{\sigma}} \odot \boldsymbol{\epsilon}, \tag{4.4}$$

where $\boldsymbol{\epsilon}$ is a random noise sampled from the standard Gaussian distribution. This reformulation makes the sampling process of the latent variable differentiable, and thus allows the encoder to be trained from the output of the decoder while remaining probabilistic. The decoder $\boldsymbol{D}$ distribution is given by $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$, where $\boldsymbol{\theta}$ again denote the parameters of the NN.

The loss function for VAEs is the Evidence Lower Bound (ELBO), which provides a lower bound on the marginal likelihood of the observed data[7]. Thus, maximising the ELBO implicitly maximises the likelihood of the generated data under the model. To derive the ELBO, first the joint distribution $p_\theta(\boldsymbol{x}, \boldsymbol{z})$ is multiplied with $1 = \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}$ and Jensen's inequality is used. Jensen's inequality states that if f is a concave function, it holds that $\mathbb{E}[f(X)] \geq f(\mathbb{E}[X])$. This is used to switch the order of the

[5] Describing the encoded input as a distribution emphasises the probabilistic nature of the encoder.

[6] A diagonal covariance for the distribution implies that all features in the latent space are maximally disentangled, i.e. independent. Note that this may be too strict a requirement, and thus may affect the performance of the model. Kingma et al. [107] also provide the framework for more general latent space distributions

[7] Note that marginal in this context refers to the marginalisation over the latent $\boldsymbol{z}$ and not the marginal distributions of the data.

logarithm and expectation value.:

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \log \int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})\, d\boldsymbol{z} \tag{4.5}$$

$$= \log \int \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\, d\boldsymbol{z} \quad \text{(introduce } q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\text{)} \tag{4.6}$$

$$= \log \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \left[\frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \right] \quad \text{(integral as expectation)} \tag{4.7}$$

$$\geq \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \right] \quad \text{(Jensen's inequality)} \tag{4.8}$$

$$= \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \right]}_{\text{Reconstruction Loss}} - \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \left[\log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{z})} \right]}_{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}||p_{\boldsymbol{\theta}})}, \tag{4.9}$$

where for the last equality it is used that $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) = p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{z})$ together with the rules for the logarithm.
The ELBO consists of two parts: a reconstruction term and a regularisation term. The reconstruction term is the expectation of the log-likelihood of the observed data given the latent code $\boldsymbol{z}$ and is denoted as $\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]$. This term encourages the VAE to accurately reconstruct the input data from the encoded input, and is identified with the mean squared error $E_{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}||\boldsymbol{x} - \boldsymbol{D}\boldsymbol{z}||^2$. The regularisation term is the Kullback-Leibler (KL) divergence between the encoder distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ and a target prior distribution $p(\boldsymbol{z})$, typically chosen as a standard Gaussian distribution:

$$D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) = \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z})} d\boldsymbol{z}. \tag{4.10}$$

Due to the reparametrisation trick, where the latent space distribution of the input was chosen as a Gaussian with diagonal covariance (i.e. $p(\boldsymbol{E}(\boldsymbol{x})) = N(0, \boldsymbol{\sigma})$), an analytical form for the KL divergence between the target $\mathcal{N}(0, 1)$ and the distribution of the latent codes is available:

$$D_{\mathrm{KL}}(q_{\boldsymbol{\theta}}|\mathcal{N}(0, 1)) = \frac{1}{2} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} (\mathbb{V}(\boldsymbol{z}) - \log \mathbb{V}(\boldsymbol{z}) - 1 + \mathbb{E}(\boldsymbol{z}_i^2)), \tag{4.11}$$

where $\mathbb{V}$ denotes the of variance of the argument along its feature dimension[8]. Note that this loss is equal to 0 if the encoder outputs a mean of 0 and a variance of 1 for each input. However, in this case, the latent representation does not contain any information about the input. Therefore, the KL term should only be seen as a regularisation, similar to the L1 or L2 regularisation mentioned in Section 3.1.4, which has its minimum when all model parameters are 0. However, in weight regularisation techniques, the regularisation term is weighted with a small value, whereas in vanilla VAE the weight is 1. A detail that is often omitted is that it can be crucial for the success of the optimisation that the weight of this loss term is ramped up during the start of the training [109]. In total, the following loss is minimised during training:[9]:

$$L(\theta, \phi; \boldsymbol{x}) = \\ \sum_{i=1}^{n} (\boldsymbol{D}(\boldsymbol{E}_{\mu,\sigma}(\boldsymbol{x}))_i - \boldsymbol{x}_i)^2 - \frac{1}{2} \sum_{i=1}^{m} \left(\boldsymbol{E}_\sigma(\boldsymbol{x})_i^2 - \log \boldsymbol{E}_\sigma(\boldsymbol{x})_i - 1 + \boldsymbol{E}_\mu(\boldsymbol{x})_i^2 \right), \quad (4.12)$$

where m is the dimension of the latent space and n is the dimension of the input space.

4.2.1 Two Moons Dataset

The Two Moons dataset is not the optimal toy example for VAEs because usually the encoding dimension is chosen to be smaller than the input dimension, as the input usually contains noise, which is not relevant for the perception of the input. However, if the latent space dimension is chosen to be 1 on the Two Moons dataset, the model has only one degree of freedom and is unable to model the spread perpendicular to the crescents as shown in the left of Fig. 4.3, even when considering only the reconstructed samples. If the latent dimension is chosen as 2, it is a test of how well the input variables can be decorrelated with the KL-divergence regularisation[10].

[8] Note that a frequently used extension is the β-VAE, where another hyperparameter β is introduced which balances the two loss terms. This hyperparameter allows controlling the disentangling of the latent representation, where a higher value of β leads to a more disentangled representation.

[9] Here the formalism is a bit sluggish, but to fit everything on one line, $\boldsymbol{E}_{\boldsymbol{\mu}}(\boldsymbol{x}) + \boldsymbol{E}_{\boldsymbol{\sigma}}(\boldsymbol{x}) \odot \boldsymbol{\epsilon}$ is abbreviated as $\boldsymbol{E}_{\mu,\sigma}(\boldsymbol{x})$. Furthermore, the derivation of the closed form of the KL divergence between two Gaussian distributions involves quite a bit of algebra and is therefore omitted.

[10] In this case the model should learn to transform the data distribution to a standard Gaussian distribution and then back into the data distribution while remaining in the same dimension. For a

Embedded Manifolds

The Two Moons dataset illustrate an important concept for synthetic data. If the dimension l of the latent space is chosen to be smaller than the dimension n of the training data, undesired consequences arise since the samples drawn from the model lie on a manifold with dimension l. A classifier that is trained to distinguish a signal, in this example corresponding to data drawn from the model, from a background, corresponding to data that does not lie on a smaller dimensional manifold, the classifier can perfectly distinguish the two classes by testing whether a sample lies on the lower-dimensional manifold[11]. When the model is used for inference, i.e. is used to find a signal in measured data, the measured data contains noise and as such does not lie on the smaller-dimensional manifold. Thus, the model will not be sensitive to the signal, even if it is present. To demonstrate that the classifier can use the lower dimensional manifold, an AE is trained on the Two Moons dataset, as shown on the left of Fig. 4.3. A discriminator is then trained to distinguish real inputs (target=1) from inputs the AE reconstructed (target=0). The right of Fig. 4.3 depicts the score distribution of this classifier on a testing set after training. Although the real and synthetic data have overlapping support, the model can still distinguish the two classes perfectly. This is only possible if the discriminative model makes use of the lower-dimensional manifold of one class. Thus, for all models in this thesis, the dimension of the noise that is used as input for the models is equal to the dimension of the modelled data.

Impact of Latent Space Regularisation

Figure 4.4 illustrates why regularisation of the latent space is necessary. In the middle, *reconstructed* samples, i.e. real inputs that are encoded and then decoded, are shown, and their distribution looks near perfect. However, the distribution of generated new samples, i.e. when latent codes are randomly sampled and then passed through the decoder, does not resemble the true distribution at all as shown on the left. The latent representation is shown on the right, which explains why

standard Gaussian distribution, both features are independent, hence the term decorrelate.

[11]This example may sound a bit too constructed, but this is a project of this thesis discussed in Chapter 7.

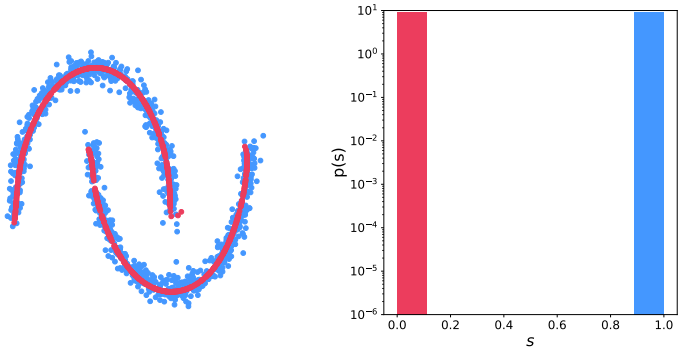


Figure 4.3: Results of an AE on the Two Moons dataset when the latent dimension is chosen as one. (left) Reconstructed samples if the encoding dimension is chosen as 1. The model is unable to capture the noise perpendicular to the crescents. (right) Score distribution of the classifier that is trained to distinguish real and reconstructed samples. Although the two distributions have overlapping support, a classifier can perfectly distinguish real and reconstructed samples.

the newly drawn samples do not resemble the target distribution. Introducing the

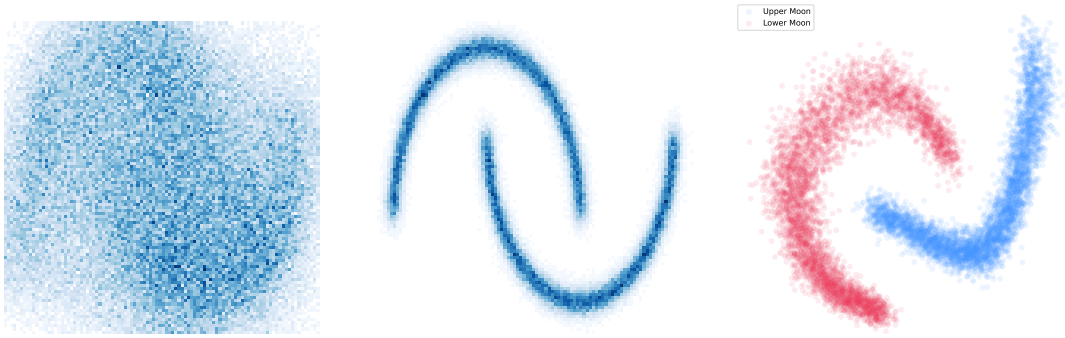


Figure 4.4: Results from training an AE illustrate why the regularisation of the latent space is crucial. (left) Samples drawn from the trained AE bear little similarity with the target distribution. (middle) The inputs reconstructed by the AE are near perfect. (right) The latent distribution of the AE is not connected and does not resemble a standard Gaussian distribution at all.

regularisation, equivalent to moving from an AE to a VAE, produces the results shown in Fig. 4.5. The reconstruction of the input is not as perfect any more, however, generated samples resemble the reconstructed ones.

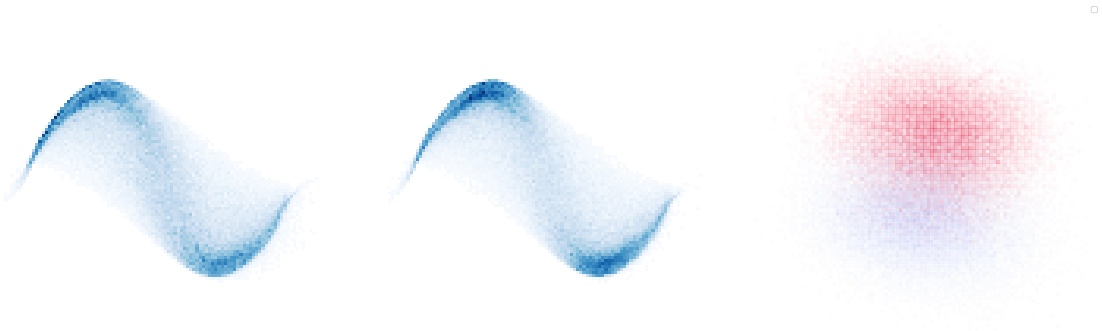


Figure 4.5: Results from training a VAE illustrate that the regularisation of the latent space closes the discrepancy between reconstructed and newly generated samples. (left) The samples drawn from VAE are different from the ones from the target distribution. (middle) The inputs that are reconstructed by the VAE are blurred between the two crescents. (right) The latent distribution of encoded inputs shows that the two crescents are encoded in different parts of the latent space, but there is a non-negligible overlap.

4.3 Generative Adversarial Networks

GANs, proposed by Goodfellow et al. [110] in 2014, train a generative model in competition with a discriminative model, as illustrated in Fig. 4.6. The discriminative model is optimised to distinguish real data $\boldsymbol{x}$ from synthetic data $\hat{\boldsymbol{x}}$ generated by the generative model. In contrast, the generative model is optimised such that the discriminative model is unable to synthetic data from real data. The training process is thus given as a *minimax* game[12]:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{real}}} L(D(\boldsymbol{x})) - \mathbb{E}_{\boldsymbol{z}\sim p_{\mathcal{N}(0,1)}} L(D(\boldsymbol{G}(\boldsymbol{z}))), \tag{4.13}$$

where D is the discriminator and G the generator. There is a Nash-Equilibrium when the generator produces synthetic data, which cannot be distinguished from real data by the discriminator. Interestingly, the generator learns to produce seemingly real samples without ever being directly exposed to real samples. Note that Equation 4.13 poses no specific requirements for G and D, which makes this training

[12]The terminology comes from game-theory; zero-sum games are called minimax games because their solution involves an inner loop maximisation and an outer loop minimisation

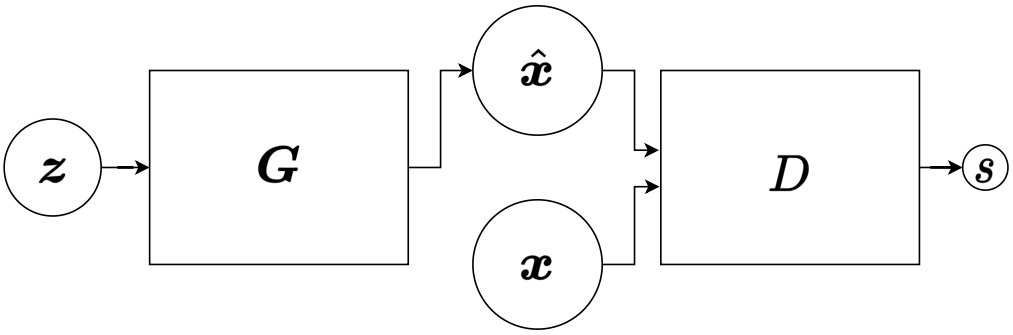


Figure 4.6: Schematic of a GAN comprising a generator $\boldsymbol{G}$ and a discriminator $\boldsymbol{D}$. The generator takes noise $\boldsymbol{z}$ as input and generates synthetic data $\hat{\boldsymbol{x}}$. The discriminator distinguishes real inputs $\boldsymbol{x}$ from synthetic ones by predicting a score s as output.

paradigm so versatile [111, 112, 113, 114, 115, 116]. Training GANs is notoriously challenging, the performance of the model is usually heavily dependent on the chosen hyperparameters [117]. The choice of loss function has a major impact on the training dynamics, further discussed in Section 4.3.1. While solutions have been proposed to further stabilise the training process, their effectiveness varies with the problem. Section 4.3.2 discusses some methods that have proven valuable in this thesis to stabilise the training.

4.3.1 Losses

An overview of different loss functions is presented in Table 4.1 and is discussed in more detail in the following.

Table 4.1: Summary of common GAN losses. Real data is represented with $\boldsymbol{x}$, and synthetic data is represented with $\boldsymbol{G}(\boldsymbol{z})$, where $\boldsymbol{z} \sim \mathcal{N}(0, 1)$ is the noise input for the generator. Note the subtle difference between discriminators $\boldsymbol{D}$ and critics $\boldsymbol{C}$, a model is referred to as a discriminator only if it outputs a value between 0 and 1.

Name	Discriminator/Critc Loss	Generator Loss
Minimax $\mathcal{L}$ [110]	$\log D(\boldsymbol{x}) + \log(1 - D(\boldsymbol{G}(\boldsymbol{z}))$	$\log(1 - D(\boldsymbol{G}(\boldsymbol{z}))$
Non-Saturating $\mathcal{L}_{\text{NS}}$ [110]	$\log D(\boldsymbol{x}) + \log(1 - D(\boldsymbol{G}(\boldsymbol{z}))$	$-\log(D(\boldsymbol{G}(\boldsymbol{z}))$
Least Squares $\mathcal{L}_2$ [118]	$(C(\boldsymbol{x}) - 1)^2 + C(\boldsymbol{G}(\boldsymbol{z}))^2$	$\frac{1}{2}(C(\boldsymbol{G}(\boldsymbol{z})) - 1)^2$
Wasserstein $\mathcal{L}_{\text{WGAN}}$ [119]	$C(\boldsymbol{G}(\boldsymbol{z})) - C(\boldsymbol{x})$	$-C(\boldsymbol{G}(\boldsymbol{z}))$
WGAN w/ Gradient Penalty $\mathcal{L}_{\text{GP}}$ [120]	$C(\boldsymbol{G}(\boldsymbol{z})) - C(\boldsymbol{x}) + \lambda(\|\|\nabla_{\boldsymbol{G}(\boldsymbol{z})} C(\boldsymbol{G}(\boldsymbol{z})\|\|_2 - 1)^2$	$-C(\boldsymbol{G}(\boldsymbol{z}))$

Minimax Loss

The original GAN publication is based on the following optimisation objective:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\psi}} V(D_{\boldsymbol{\psi}}, \boldsymbol{G}_{\boldsymbol{\theta}}) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D_{\psi}(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D_{\boldsymbol{\psi}}(\boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{z})))] \quad (4.14)$$

where ψ and θ are the parameters of the discriminator and generator respectively, $\boldsymbol{x}$ are data samples, $\boldsymbol{z}$ are noise samples, $D(\boldsymbol{x})$ is the estimate of the probability that a sample $\boldsymbol{x}$ comes from the real distribution by the discriminator, and $\boldsymbol{G}(\boldsymbol{z})$ is a sample drawn from the model.

In this framework, D is trained to maximise the probability of correctly identifying real and synthetic samples, hence the term $\max_{\boldsymbol{\psi}}$. Simultaneously, the generator $\boldsymbol{G}$ is trained to make the discriminator D misidentify synthetic samples, hence the term $\min_{\boldsymbol{\theta}}$. This results in a dynamic training process where $\boldsymbol{G}$ and D are constantly adapting to each other.

The minimax game has a global optimum when the generator replicates the data distribution perfectly and the discriminator outputs a probability of $\frac{1}{2}$ for all samples, indicating that real and synthetic samples are indistinguishable. In practice, this equilibrium is unlikely to be reached.
Already with this setup, the authors can generate new samples from complicated high-dimensional distributions (e.g. MNIST [121], CIFAR-10 [90])[13]. They further show that during training the Jensen-Shannon divergence between the data and generator distributions is reduced, and the global optimum of the training loss lies at[14] $-\log 4$.

Non-Saturating Loss

In the same paper, Goodfellow et al. [110] point out that the minimax loss function has a flaw. When the discriminator assigns a low probability to synthetic data,

[13] It is subtle to note that there is an important difference in priorities between HEP and the more common generative modelling literature in ML. In HEP, it is of high importance that the distribution of samples is similar to the true underlying distribution. However, e.g. for image generation, more focus is laid on individual samples.

[14] To see this, evaluate the discriminator loss function, if the discriminator outputs 0.5 for synthetic and real samples.

which is usually the case early in training, the loss for the generator saturates, as shown in Fig. 4.7. A reformulation of the generator loss can take this problem into account. Instead of minimising $\log(1-D(G(\boldsymbol{z})))$, the generator is trained to minimise $-\log D(G(\boldsymbol{z}))$. This can be understood as follows: in the minimax formulation, the generator tries to make the discriminator assign a high probability to synthetic samples as being real; in the latter formulation, it tries to prevent the discriminator from classifying synthetic samples correctly. This GAN is referred to as NSGAN in the following.

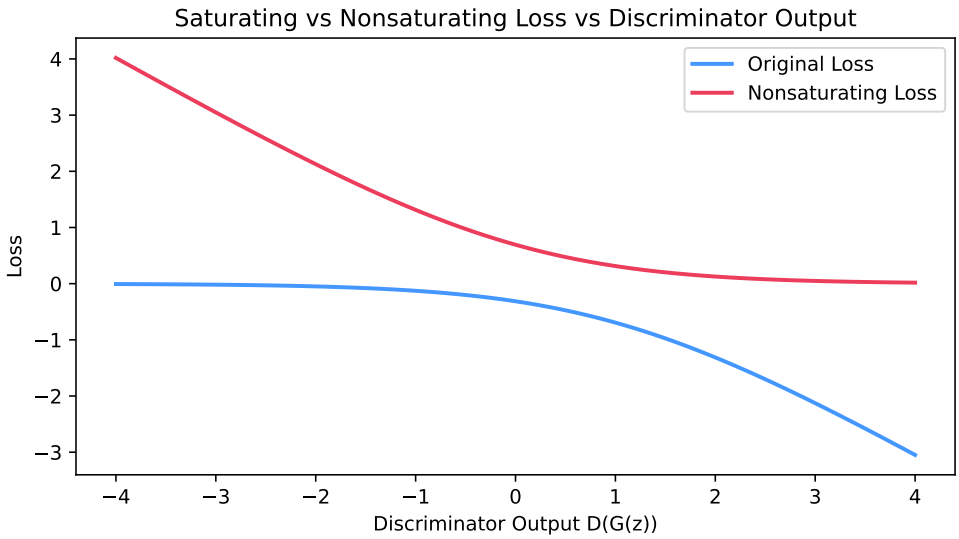


Figure 4.7: The difference between the two losses proposed by Goodfellow et al. in Ref. [110]. For the saturating loss, the gradient approaches zero when the discriminator is very confident that a sample is synthetic and therefore assigns a low value to it. Since this is usually the case at the beginning of the training, the generator gets little feedback to improve. The opposite is true for the non-saturating loss. The gradient for the generator is large if the discriminator assigns a low value to the scores and small if the discriminator assigns a high probability to it. Note, that it is no problem that the gradient for the generator saturates if the discriminator assigns a high probability to it, as the Nash equilibrium is at the point where the discriminator outputs 0.5 for synthetic and real samples.

Least-Squares GAN

Mao et al. [118] introduced the LSGAN, which uses an MSE to as loss function. They show that this GAN minimises the χ^2 divergence of between the real and generated distribution. The loss for both the generator and critic[15] are given as $|C(\boldsymbol{x}) - y|^2$. For the generator $y = 1$ for synthetic samples, and for the critic $y = 1$ for real samples and $y = 0$ for synthetic samples.

Wasserstein GAN Loss & Gradient Penalty

Arjovsky et al. show that the Jensen-Shannon (JS) or Kullback-Leibler (KL) divergence are a suboptimal objective to minimise for a generative model in Ref. [119]. They propose that the Wasserstein-1 distance given in Eq. 4.15 is a more suiting objective:

$$W(p_{\text{data}}, p_g) = \left(\inf_{\pi \in \Pi} \int_{\mathbb{R}^d \times \mathbb{R}^d} ||\boldsymbol{x} - \boldsymbol{y}|| d\pi(\boldsymbol{x}, \boldsymbol{y}) \right), \tag{4.15}$$

where p_{data} is the data distribution, p_g is the model density, and Π is the set of joint distributions whose marginals are p_{data} and p_g. Note that the expression is intractable due to the infimum over the set of joint distributions Π. The authors propose the Wasserstein GAN (WGAN) setup that minimises the Wasserstein-1 distance between the target and model distributions:

$$W(p_{\text{data}}, p_g) = \sup_{||C||_L \leq 1} \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[C(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x} \sim p_g}[C(\boldsymbol{x})]. \tag{4.16}$$

The supremum is taken over all Lipschitz-1 continuous functions, i.e. functions which fulfils $|f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)\| \leq L\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|, \forall \boldsymbol{x}_1, \boldsymbol{x}_2$. This is thus a requirement that the critic C must satisfy[16]. The Lipschitz constraint is enforced during training by weight clipping, i.e. limiting the range of values the weights of the NN can take to small values. They present their results on the image dataset *LSUN-Bedrooms* [122] and show that their loss function stabilises the training of a previously unstable model.

[15] Note that here the naming moves from the discriminator D to the name critic C because there will be no sigmoid activation at the end of the model. This means that it will no longer output a value that can be interpreted as probability, since it is no longer normalised to one.

[16] The reason this Lipschitz constraint is needed is that otherwise, a function $f' = \lambda f$ where λ is a scalar would have an λ times larger distance than when initial f is used. This can be problematic since the critic minimises the negative of this distance.

One major benefit of their approach is that the critic loss can be interpreted as a multiple of the Wasserstein distance[17]. Gulrajani et al. [120] later proposed an improved approach to make the critic Lipschitz-1 continuous, to which the authors refer as *Gradient Penalty* (GP). They add a term to the loss function of the critic, that penalises the critic if the norm of the gradient of the critic deviates from 1:

$$\mathbb{E}_{\hat{\boldsymbol{x}}\sim p_{\hat{\boldsymbol{x}}}}[(||\boldsymbol{\nabla}_{\hat{\boldsymbol{x}}}C(\hat{\boldsymbol{x}})||_2 - 1)^2], \tag{4.17}$$

where $\hat{\boldsymbol{x}}$ is sampled uniformly along a straight line between pairs of real and synthetic samples.

4.3.2 Other Approaches to Stabilise Training

Even though the losses discussed claimed to stabilise the training, in practice the models are still far from robust. In the following, some other measures that stabilised the training are briefly discussed.

Feature Matching Salimans et al. [123] propose additional ideas to improve the convergence of GAN training. In this thesis, *feature matching* proved to be useful. The objective of the generator is changed. Instead of maximising the output of the critic or discriminator, it should generate data that matches the expected values of statistics from real data. Specifically, the statistics are chosen as the activations from an intermediate layer of the critic.

Weight Normalisation & Spectral Normalisation Kingma et al. [124] proposed a reparametrisation to decouple the direction from the magnitude of weight vectors[18]:

$$\boldsymbol{w} = \frac{g}{||\boldsymbol{v}||}\boldsymbol{v}, \tag{4.18}$$

where g is a learnable scalar, $\boldsymbol{w}$ the weight vector used in NN and $\boldsymbol{v}$ is a new parameter with the same shape as the weight vector $\boldsymbol{w}$. Thus, the network learns

[17] Beforehand it was omitted that if the NN is not 1-Lipschitz but k-Lipschitz, Equation (4.16) results in the Wasserstein distance times a factor k.

[18] Weight vectors refer to the rows of the weight matrices

both g and $\boldsymbol{v}$ for each layer. This reparametrisation should stabilise the training, referred to as *weight norm*.
Xiang et al. [125] propose a modification to the initial weight normalisation which is more suited for GANs[19]. In their study, they show that their modification of the normalisation performs significantly better than BatchNorm, respective, the absence of any normalisation. However, their weight normalisation enforces the squared singular values of the weight matrix to add to d, where d is the minimum of the number of rows and columns of the matrix. Miyato et al. [126] show that this imposes a much stronger constraint on the matrix than intended, as this leads to a model that only uses one feature, and propose another normalisation *spectral normalisation* where the weight matrix is normalised by its maximum singular value. This cures the previously mentioned problem, as the singular values of the matrix are now decoupled[20]. However, in this thesis, GANs trained with spectral normalisation consistently performed worse than GANs trained with weight normalisation.

4.3.3 Two Moons Dataset

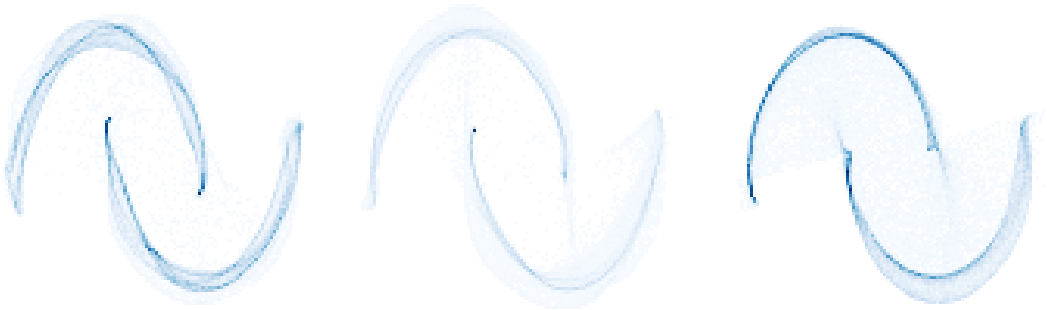
In this section, the performance of different GANs on the Two Moons dataset is evaluated and discussed. The NSGAN and LSGAN were both trained with standard momentum values for Adam. For the WGAN it is known [119] that momentum can break the training, and hence it is set to 0 or the RMSprop optimiser is used. In the studies in this thesis, using any momentum $\beta > 0$ for the training of a GAN resulted in a non-converging training.
As GANs are notoriously unstable, even in this small toy study a hyperparameter optimisation was conducted. The following architectural choice space[21] was searched:

- Batch Norm: [True, False]
- Spectral Normalisation: [True, False]
- Weight Normalisation: [True, False]

[19] They argue that in its simplest form, the original weight normalisation does not normalize the mean value of the input and adapt it accordingly.
[20] Previously they are coupled since their squares add up to d.
[21] The feature matching loss mentioned previously did not give any acceptable results on this toy dataset, and hence is left out.

- Gradient Penalty: [True, False]

In Fig. 4.8 samples drawn from the best-performing model from every training paradigm are shown. These results of the NSGAN and LSGAN nicely illustrate that GANs produce good sample quality, as almost all points lie on the two crescents. However, both models do not cover the entire support of the distribution and prefer specific modes. This is a known artefact often present in GANs, which is referred to as *mode collapse* [127]. For the WGAN, the distribution is slightly smoother, but the model is still unable to capture the target distribution accurately. This already points at a major problem with GANs for use in particle physics, since distributions are more important there than the individual sample quality. It is unusual that in searches only single events are studied[22]. Thus, the mode collapse could pose a detrimental property in HEP.



(a) Non-Saturating (b) Least-Squares (c) WGAN-GP

Figure 4.8: Histograms of samples from different GAN training paradigms. On the left, results for the non-saturating GAN are shown, in the middle results from the least-squares GAN and on the right samples from a WGAN that was trained with Gradient Penalty. For all models, mode collapse is visible, as the samples are concentrated on certain modes of the distribution..

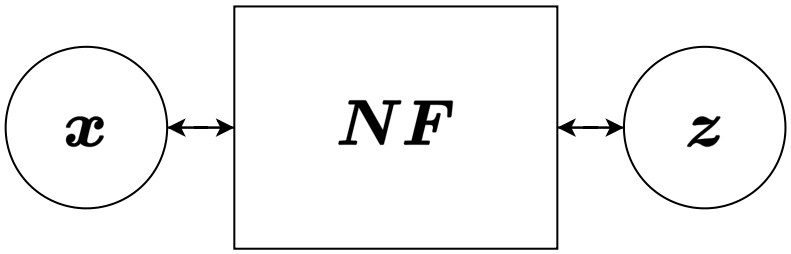


Figure 4.9: A NF learns the unknown distribution of the data $\boldsymbol{x}$ to a standard Gaussian distribution of the corresponding latent variables $\boldsymbol{z}$. The mapping functions of the NF are constructed such that they are invertible. Samples from the approximated data distribution can then be drawn by sampling the standard Gaussian distribution and applying the inverted transformations.

4.4 Discrete Normalising Flows

Although the concept of discrete Normalising Flows[23] (NFs) been previously used in a similar framework before in Refs. [129, 130], NFs have gained popularity mainly through the works of Rezende et al. [131] and Dinh et al. [132]. Typically, NFs are defined as a series of invertible and differentiable transformations to transform a simple probability distribution, e.g. a standard Gaussian, into the complex, unknown data distribution. The change of variables theorem is then used to compute the probability of the data, allowing the NNs hidden in the transformation functions to be trained. To serve as generative models, the invertibility property of NFs is exploited. New samples can be drawn from the model distribution of the data by sampling first from the base distribution and then applying the inverse transformations. An illustration of the training paradigm is shown in Fig. 4.9. Due to the bijectivity requirement, $\boldsymbol{x}$ and $\boldsymbol{z}$ are required to have the same dimension.
A more precise mathematical formulation is given in the following: Let $X, Z \in \mathbb{R}^n$ be two random variables. The goal is to find invertible and differentiable transformations $\boldsymbol{f}$ that map the known base distribution $p_Z(\boldsymbol{z})$ to the unknown training data distribution $p_X(\boldsymbol{x})$. To achieve this, the change of variables formula for probability distributions given in Equation 4.19 provides access to the density of the unknown

[22] Even when evidence for the gluon in form of a 3-pronged jet was found in 1979 [128] at PETRA, it was clear that a single event was not enough to claim discovery.
[23] For notational convenience the discrete is omitted in the following.

data distribution:

$$p_X(\boldsymbol{x}) = p_Z(\boldsymbol{f}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{x})) \left|\det\, D\boldsymbol{f}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{x}))\right|, \tag{4.19}$$

The functions $\boldsymbol{f}$ contain NNs, and their parameters $\boldsymbol{\theta}$ are optimised by maximising the likelihood of the training data:

$$\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}|X) = \max_{\boldsymbol{\theta}} \prod_{\boldsymbol{x}\in X} p(\boldsymbol{x}|\boldsymbol{\theta}) \tag{4.20}$$

In practice, the negative logarithm of the likelihood is minimised for computational reasons[24]. The loss function estimated over a batch of M samples is given by[25]:

$$L_{\text{nLL}}(\boldsymbol{\theta}) = -\sum_{\text{i}=1}^{M} \log\, p_X(\boldsymbol{x}^{(\text{i})}|\boldsymbol{\theta}) = -\sum_{\text{i}=1}^{M} \log p_Z(\boldsymbol{f}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{x}^{(\text{i})}|\boldsymbol{\theta})) + \log \left|\det\, D\boldsymbol{f}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{x}^{(\boldsymbol{i})}|\boldsymbol{\theta})\right|, \tag{4.21}$$

Note that direct maximisation of the likelihood in the input space is not possible, as there is no access to the data distribution $p_X(\boldsymbol{x})$ in the input space.
The functions attain their invertibility by being element-wise parametrised transformations, for which the inverse is analytically available. For the loss in Equation 4.21 to be tractable, the determinant of the Jacobian needs to be efficiently calculable. In general, the calculation of the determinant of a $D \times D$ matrix requires $O(D^3)$ computations. However, for triangular matrices, this is significantly faster, since the determinant is the product of its diagonal elements. Therefore, NFs usually rely on transformations that have a triangular Jacobian, which are further divided into two classes:

1. *Autoregressive* transformation-based model [133, 134]; the transformation of variable $z_k = f^{-1}(x_k|x_{k-1}, ...x_1)$ is only dependent on previous features $x_{k-1}, x_{k-2}, .., x_1$, for a chosen order of features. Autoregressive NFs are D times slower to evaluate in one direction[26], where D is the dimension of $\boldsymbol{x}$. This is because the NNs in the transformations are always evaluated in the same

[24] As every term in the product of the maximum likelihood is small, and the multiplication of many small terms reaches the floating-point precision of the machine.

[25] For simplicity, only one function to transform the data is used, but for multiple transformations, the individual terms can be summed to obtain the loss of the composite transformation.

[26] Note, that the direction which is quick to evaluate can be chosen.

direction. Thus, when the reverse transformation $x_k = f(z_k|x_1, ..., x_{k-1})$ is applied variables $x_1, ..., x_{k-1}$ need to be available, hence for the other direction in this case f needs to be evaluated D times. This approach is motivated by the decomposition of a joint distribution $p(x_1, ..., x_D)$ into conditional univariate distributions:

$$p(x_1, ..., x_D) = p(x_1) \prod_{i=2}^{D} p(x_i|x_1, ..., x_{i-1}). \tag{4.22}$$

2. *Coupling* layer-based models [135, 136, 137]; only half of the variables are transformed at a time, leading to an NF that transforms with the same number of function evaluations in both directions.

The interested reader can consult the comprehensive reviews about NFs in Refs. [138, 139] for further information. In this thesis, coupling layer-based NFs are used, and thus, they are discussed in the following in more detail.

4.4.1 Normalising Flows with Coupling Layers

To summarise, the NFs must meet the following requirements, in descending order of difficulty:

1. Invertibility,
2. tractable Jacobian determinant, especially beneficial if it is easy to calculate,
3. and differentiability.

Coupling layers solve the first and most difficult requirement by construction. This is best understood with the illustration given in Fig. 4.10. During the reverse transformation (from left to right), the input data X is split along its feature dimension into two disjoint sets A and B. The split is chosen randomly when the NF is initialised but remains fixed afterwards. The first set is mapped to itself via the identity function. The second set is element-wise transformed with an invertible parameterised function, which gets its parameters for the transformation θ_A from an NN. The variables in the latent space are denoted by z_i. To invert the coupling layer (from right to left), the same split is applied to z_i to obtain the sets A and B'. Since A has been transformed only by the identity, its inverse is trivial. Then

the parameters θ_A can be calculated again by passing the input through the NN. With their knowledge, the analytical transformation can be inverted and applied to B'. For illustration, the invertible transformations are connected by double-sided arrows. From this construction, it can also be seen that the non-trivial part of the Jacobian of this transformation (the Jacobian for A is unity) is not dependent on other elements from B, such that a triangular Jacobi matrix is obtained for which the determinant is the product of its diagonal. These coupling layers are chained together to obtain a more expressive transformation, which is necessary. Otherwise, when using only one coupling layer, half of the variables are not transformed at all.

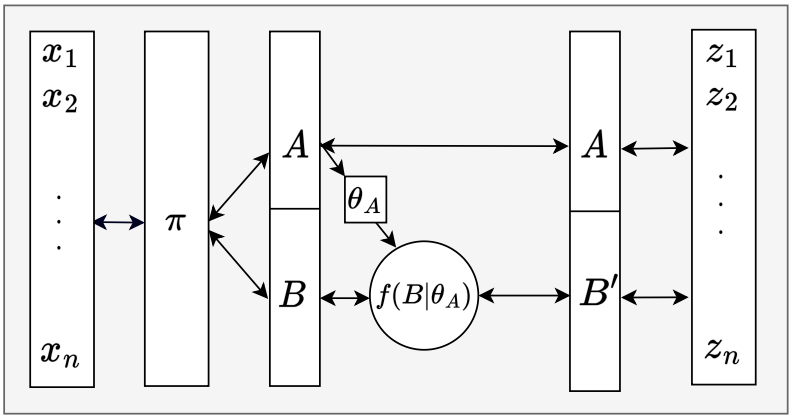


Figure 4.10: Schematic of a coupling layer, explained in the paragraph above.

An illustrative example is *affine* coupling layers [136], for which the transformation $\boldsymbol{f_\theta}$ is given as[27]:

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \begin{cases} \boldsymbol{x}^A & \text{if } x_i \in \boldsymbol{x}^A \\ \boldsymbol{x}^B \odot \boldsymbol{s}_{\boldsymbol{\theta}(\boldsymbol{x}^{\mathrm{A}})} + \boldsymbol{t}_{\boldsymbol{\theta}(\boldsymbol{x}^{\mathrm{A}})} & \text{else} \end{cases} \tag{4.23}$$

To see why this construction is invertible, consider the inverse direction: starting from $\boldsymbol{z}$ the same split is applied, and $\boldsymbol{x}^A$ is obtained via identity. Then, $\boldsymbol{s}_\theta, \boldsymbol{t}_\theta$ are obtained by evaluating the NN on $\boldsymbol{x}^A$. Finally, $\boldsymbol{z}^B$ is found by subtracting $\boldsymbol{t}_\theta$ and dividing by $\boldsymbol{s}_\theta$. The only condition here is that $\boldsymbol{t}_\theta$ must be non-zero, which is satisfied by exponentiation. The determinant for the transformation is simply the product of

[27] Note the slight misuse of notation; $x_i \in \boldsymbol{x}^A$ is a condition whether feature x_i is part of the split $\boldsymbol{x}^A$

the diagonal of the Jacobian, as shown in Equation 4.24:

$$D\boldsymbol{f} = \begin{bmatrix} \mathbb{I} & 0 \\ \frac{\partial \boldsymbol{z}_{\mathrm{D/2:D}}}{\partial \boldsymbol{x}_{\mathrm{1:D/2}}} & \mathrm{diag}(s_{\boldsymbol{\theta}}(\boldsymbol{x}_{\mathrm{1:D/2}})) \end{bmatrix} \Rightarrow \det|D\boldsymbol{f}| = \prod_{\mathrm{i}=1}^{\mathrm{D/2}} s_{\boldsymbol{\theta}(\boldsymbol{x}_{\mathrm{1:D/2}})_{\mathrm{i}}} \tag{4.24}$$

4.4.2 Monotonic Rational Quadratic Spline Coupling Layers

The affine transformation is illustrative, but in this thesis, more expressive coupling transformations called piecewise invertible Rational Quadratic Splines (RQS) [140] proved to be performing best. These coupling layers are conveniently implemented in `nflows` [141], a PyTorch framework for NFs. The RQS is defined on an interval $[-B, B]$, and as the identity function outside this interval. They contain K different rational-quadratic functions in K bins, of which the boundaries are given by $K+1$ knots $\left\{\left(x^{(k)}, y^{(k)}\right)\right\}_{k=0}^{K}$. The knots are constrained to increase monotonically and each is paired with a derivative δ_k, which is constrained to be positive. The derivatives at the first and last knot are an exception and are set to $\delta_0 = \delta_K = 1$ to match the linear extrapolation. This means that the RQS coupling layers have $3k-1$ parameters per input dimension, which is significantly larger than the 2 parameters per dimension of the affine NFs. However, this only affects the dimension of the last layer of the parameter NNs, and since most of the weights come from the hidden layers, this is not a major issue. Figure 4.11 from Ref. [140] shows an RQS and its derivative. Since Equation 4.19 shows that the derivative of the distribution has a direct contribution to the expressiveness of the mapping, this highlights the expressiveness of the RQS. A detailed construction of these transformations is given in Ref. [140].

4.4.3 Conditioning for Normalising Flows

Conditional generation refers to the task of drawing samples from the conditional distribution $p_{\mathrm{data}}(\boldsymbol{x}|c)$, where c is a certain condition, e.g. $c = \mathrm{cat}$, where the training data distribution $p(\boldsymbol{x})$ is over images of animals, so a sample of $p(\boldsymbol{x}|c)$ is an image of a cat. For coupling layers, conditioning is implemented as illustrated by the red dashed lines in Fig. 4.10. The only difference is that the NN predicting the parameters $\boldsymbol{\theta}_A$ of the invertible transformation f has an additional arbitrary input $c \in \mathbb{R}^k$. This input can increase the expressivity of the transformation but breaks the full invertibility of

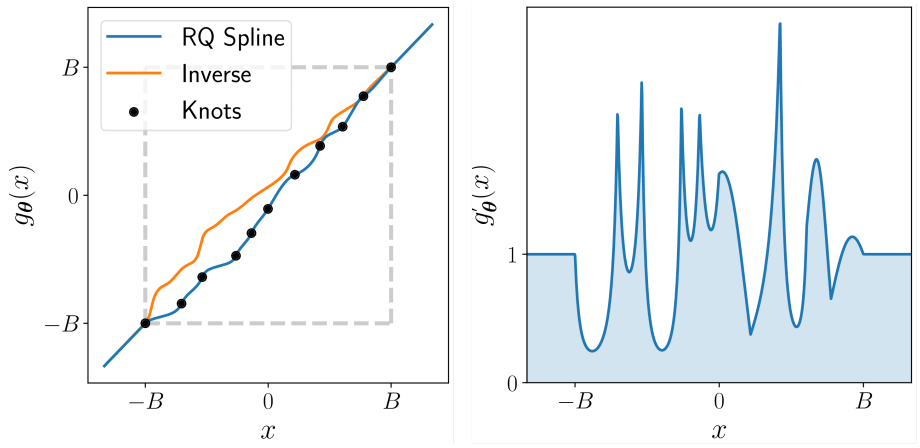


Figure 4.11: Illustration motivating the expressivity of RQS. (left) A Monotonic RQS with $K = 10$ bins (equivalent to 11 Knots) and linear tails outside the $[B, -B]$ interval. (right) The derivative of the RQS. Together with Equation 4.19 this nicely illustrates the multimodality of the RQS transformation, as the derivative appears in the equation defining the transformation of the density of the random variable. Both figures are taken from Ref. [140].

the NF since the condition needs to be supplied also in the inverted direction. In the `nflows` framework, conditioning is implemented by mapping it to the same hidden dimension as the A split and then a Gated Linear Unit [142] is used to inject the condition in a residual connection[28]. This is formalised as follows:

$$\begin{aligned}\boldsymbol{x} &= \boldsymbol{x} + \text{Block}(\boldsymbol{x}, \text{c}) \\ \text{Block}(\boldsymbol{x}, \text{c}) &= \text{W}_2(\sigma_2(\text{W}_1(\sigma_1(\boldsymbol{x})))) \odot \sigma_\text{S}(\text{W}_\text{c}\text{c}),\end{aligned}$$

where σ_S is a Sigmoid activation, σ_1, σ_2 are arbitrary activations, $W_{1,2,c}$ are weight matrices and $\odot$ is an element-wise multiplication.

4.4.4 Two Moons Dataset

Here, the performance of the affine and RQS coupling NFs is compared on the Two Moons dataset. In Fig. 4.13, samples from the RQS and affine NF are compared together with their training loss. For this dataset, NFs with RQS coupling layers

[28] This is mentioned here because, as will be shown in Section 5.1 this conditioning is quite effective.

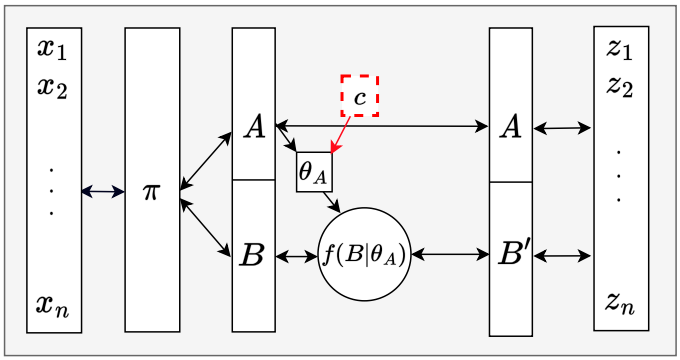
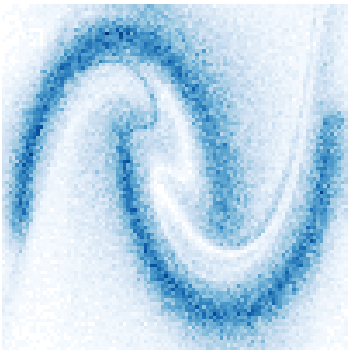
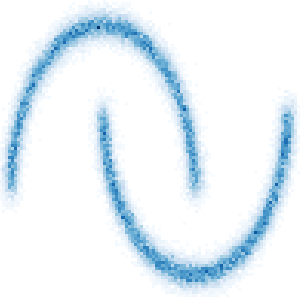


Figure 4.12: Schematic for conditioned coupling layers. The difference to the standard coupling layers is highlighted in red.

perform significantly better than affine coupling layer-based ones[29]. This difference in performance was also seen in other problems studied during the work of this thesis, so in the following, all discrete NF results are from RQS NFs unless stated otherwise.



(a) Affine



(b) RQS

Figure 4.13: Comparison of Affine and RQS Flows. (left) Samples from the affine NF, there are artefacts visible where the distribution is not correctly modelled. (right) Samples drawn from the RQS NF, here the samples are nearly indistinguishable from samples from the underlying distribution.

In Fig. 4.14 more results of the RQS-based NF are shown. A benefit of NFs is that they provide access to the probability density function, which the model is sampling.

[29]Note that this cannot be due to overfitting, as new training samples are generated for each training step.

This is visualised in Fig. 4.14a, which illustrates the outstanding performance of the NF. Since there is no direct access to the probability density function of the data, it cannot be compared, but judging by eye, it seems excellent. In Fig. 4.14b the logarithm of the probability density function is visualised, which shows that the NF is not perfectly modelling the density where it has not seen data, but this is to be expected. Figure 4.15 shows a histogram of the latent representation of the input, red represents the image of the upper moon and blue represents the image of the lower moon when the NF is evaluated in the inverse direction. The figure indicates that the NF can map the two crescents to non-overlapping regions, which the VAE discussed earlier was unable to do. If the encoding dimension for a VAE is 2, the goals of the NF and the VAE are indistinguishable on the Two Moons dataset. Both models try to transform the data distribution to a standard Gaussian distribution, but there are two major differences between VAEs and NFs:

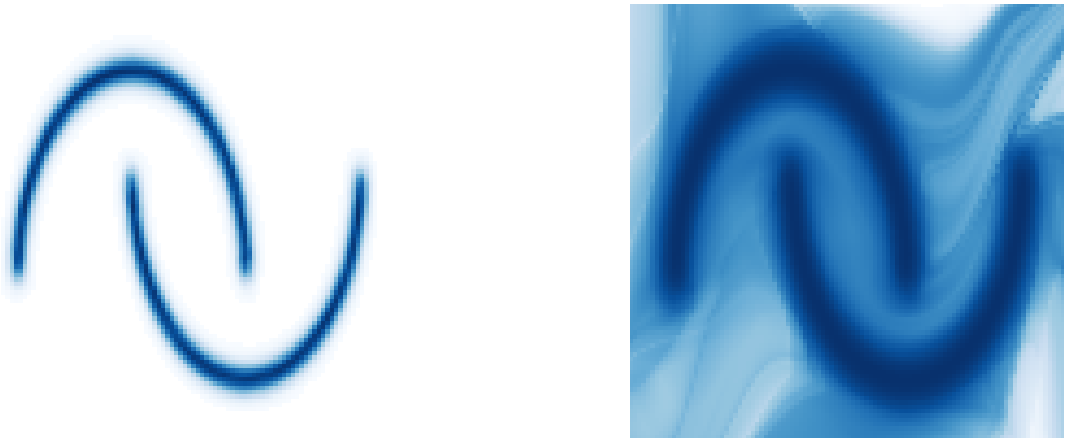
1. For the NF, the "decoder" is by design the exact inverse of the "encoder", which maps to the latent space. This makes the optimisation significantly easier.

2. For the VAE, the loss function which should decorrelate the two latent dimensions is the KL divergence and only acts as a regulariser. For the NF, maximising the Jacobi determinant of the transformation acts as a more intuitive objective. The determinant describes how a unit of space stretches, or respectively, is compressed. Note that the largest growth is attained when the features are orthogonal to each other and hence are not correlated.

The results on the Two Moons dataset might suggest that RQS NFs are all that are needed to model and sample unknown probability distributions. However, in Section 5.4.1 it is shown that the performance deteriorates quickly when it comes to higher dimensional distributions.

Conditioning on the Two Moons Dataset

NFs are constructed to be differentiable, which implies that the connectivity of the image will be equal to the support of the base density. Thus, since the support of a Gaussian distribution only contains one connected area, the transformed density,



(a) $p_X(\boldsymbol{x})$ (b) $p_X(\boldsymbol{x})$ on log scale

Figure 4.14: Visualisation of the NF probability density. (left) Probability density obtained from the NFs with RQS layers. (right) The connection between the two crescents is only visible on a log scale of the density ($nLL_{\text{min}} = 0.0001$).

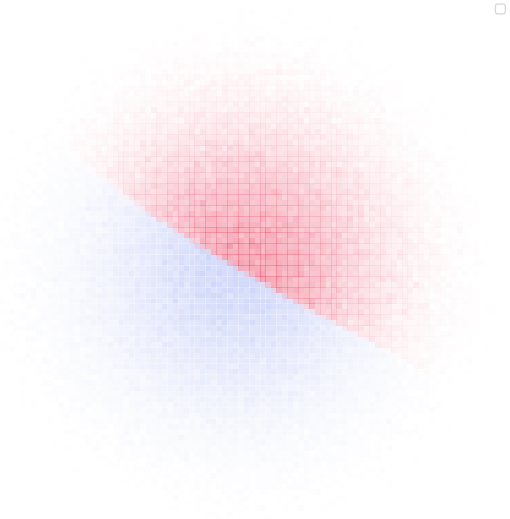


Figure 4.15: Visualisation of the input distribution in the latent space $p_Z(\boldsymbol{f}^{-1}(\boldsymbol{x}))$ for the Two Moons dataset. The red points correspond to the upper moon, while the blue points correspond to the lower moon. The model encodes the two crescents in two regions with almost no overlap.

referred to as *push-forward*, also contains one connected area. This is visualised

in Fig. 4.14b and becomes only visible if the density is visualised on a log scale. Introducing conditioning resolves this. Since the condition ($[0, 1]$) is not connected, the image can also be disconnected. The condition then fully determines on which moon a corresponding $\boldsymbol{z}$ in the latent space lies, implying that the latent space is fully decorrelated from the choice of the moon as shown. This feature of conditioned NFs can also be used to decorrelate features, which has been proposed by Klein et al. [143]. Since the latent space is not required to be partitioned any more as previously shown in Fig. 4.15, the image of the whole latent space can be transformed to on one of the crescents by choosing the respective condition. This is illustrated in Fig. 4.16, which shows the latent representation of both crescents, the conditional probability density function of the lower moon together with samples from the conditioned NF.

(a) $p_Z(\boldsymbol{f}^{-1}(\boldsymbol{x}))$ (b) $p_X(\boldsymbol{x}|c)$ (c) Sample from $pX(\boldsymbol{x}|c)$

Figure 4.16: Results from conditioned Normalising Flow. (left) The latent distributions of the upper moon, shown in red, and the lower moon are shown, in blue, reveal that the model is not required to map the two crescents to disjoint regions. (middle) Probability density obtained from the conditional NF when the condition is set as the lower moon. (right) Sample drawn from the model if the condition is chosen as the lower crescent.

4.5 Continuous Normalising Flows

Although physicists generally like to discretise things, Continuous Normalising Flows (CNFs), proposed by Chen et al. [144] in 2018, are an interesting counterexample[30]. For CNFs, the infinitesimal dynamics of the probability density are of interest. Thus, now the CNF $\boldsymbol{\phi}_t$ is dependent on time, denoted in its index. The dynamics are described with an Ordinary Differential Equation (ODE):

$$\frac{\partial \boldsymbol{\phi}_t(\boldsymbol{x})}{\partial t} = \dot{\boldsymbol{\phi}}_t = \boldsymbol{u}_t(\boldsymbol{\phi}_t(\boldsymbol{x})) \tag{4.25}$$

$$\boldsymbol{\phi}_{t=0}(\boldsymbol{x}) = \boldsymbol{x}, \tag{4.26}$$

where $\boldsymbol{u}_t$ is a time-dependent vector field. The CNF $\boldsymbol{\phi}$ describes the change of the data distribution as:

$$\frac{\partial p_t(\boldsymbol{\phi}_t(\boldsymbol{x}))}{\partial t} = -\text{div}(\boldsymbol{u}_t(\boldsymbol{\phi}_t(\boldsymbol{x}))), \tag{4.27}$$

where p_t is the *probability path*, a time-dependent probability distribution:

$$\int p_t(\boldsymbol{x}) d\boldsymbol{x} = 1 \quad \forall t \in [0, 1] \tag{4.28}$$

The CNF reshapes a simple distribution $p = p_{t=0}$, e.g. a standard Gaussian, to the data distribution $q = p_{t=1}$. However, this is done continuously and as such is defined for every time $t \in [0, 1]$ as:

$$p_t(\boldsymbol{x}) = [\boldsymbol{\phi}_t]_* p_0(\boldsymbol{x}) = p_0(\boldsymbol{\phi}_t^{-1}(\boldsymbol{x})) \det[\boldsymbol{\nabla}_{\boldsymbol{x}} \boldsymbol{\phi}_t^{-1}(\boldsymbol{x})] \tag{4.29}$$

To train the CNF, the same Maximum Likelihood-based training as for NFs is used. But for this, access to the density of the data distribution $p_{t=1}(\boldsymbol{x})$ is needed:

$$\log p_1(\boldsymbol{x}) = \log p_0(\boldsymbol{x}) - \int_0^1 \boldsymbol{\nabla} \cdot u_t(\boldsymbol{\phi}_t(\boldsymbol{x})) dt \tag{4.30}$$

In practice, an ODE solver is used to compute this integral, which makes the training of CNFs much slower than other models.

[30] Note, that CNFs are in practice discretised as well, since the Ordinary Differential Equation that is being solved is discretised.

Solving Ordinary Differential Equations

In this thesis, the Midpoint Euler method, which is also one of the simplest approaches, proved to perform best. It starts at the boundary condition $(\boldsymbol{x}_0, \boldsymbol{y}_0)$ and iteratively solves the ODE, as described in Alg. 2. The accuracy of this solver is proportional to $\mathcal{O}(h^2)$, where h is the step size.

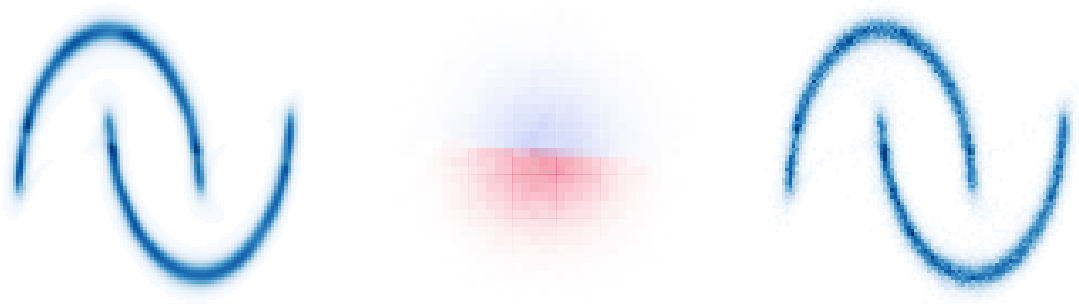
Algorithm 2 Midpoint Method for Solving ODEs

Require: Initial value (x_0, y_0), step size h, function $f(x, y)$ representing the derivative y', and number of steps N.

$x \leftarrow x_0$
$y \leftarrow y_0$
for $i = 1$ to N **do**
    $k_1 \leftarrow f(x, y)$ ▷ Slope at the start of the interval
    $y_{\text{mid}} \leftarrow y + \frac{h}{2} \cdot k_1$ ▷ Estimate the value at the midpoint
    $k_2 \leftarrow f\left(x + \frac{h}{2}, y_{\text{mid}}\right)$ ▷ Slope at the midpoint
    $y \leftarrow y + h \cdot k_2$ ▷ Update the value of y using the midpoint slope
    $x \leftarrow x + h$ ▷ Move to the next point
end for

4.5.1 Two Moons Dataset

The results obtained from the continuous NF are shown in Fig. 4.17. Since the model trains significantly slower compared to previous models, it was ruled out without further investigations. However, with a modification to the training objective, CNFs become a viable option as discussed later in Sec. 4.7.



(a) $p_Z(\boldsymbol{f}^{-1}(\boldsymbol{z}))$ (b) Latent Codes (c) Sample from $p_Z(\boldsymbol{f}^{-1}(\boldsymbol{z}))$

Figure 4.17: (left) Probability density obtained from a CNF. (middle) Input encoded in latent space. (right) Sample drawn from CNF.

4.6 Diffusion Models

Although, diffusion models are not employed for any projects mentioned in the following, this thesis would be incomplete if this class of generative models was not briefly discussed. However, only denoising diffusion probabilistic models (DDPMs) are discussed, and score-based generative models are left out. Diffusion models [145] gained a lot of attention in the 2020s, as they were capable of beating GANs on image synthesis [146] when considering sample quality.

These models work by adding iteratively a small amount of noise to a data sample, and an NN is then trained to revert these noise additions. Noise is added until the transformed data becomes indistinguishable from pure noise[31]. To draw samples from the model, standard Gaussian noise is sampled and the NN is iteratively applied to denoise the sample. Note that this iterative approach, makes sampling significantly slower, as typically the NN is evaluated up to 200× in this thesis.

More formally, the *forward* process is described as a Markov Chain:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t\boldsymbol{I}), \tag{4.31}$$

[31] Note that a sample itself cannot become noise, it is actually the data distribution that becomes indistinguishable from a standard Gaussian distribution.

where $\{\beta_t \in (0,1)\}_{t=1}^{T}$ defines the *variance schedule* over T time steps. The goal of training the model is to approximate the reverse process $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$, which for small β_t is also Gaussian. However, it is not possible to just apply Bayes theorem to obtain this posterior[32]. Thus, an NN should learn to approximate:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \mu_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t), \Sigma_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)). \tag{4.32}$$

To reverse the noising process, the joint distribution of the backward Markov Chain $p_{\boldsymbol{\theta}}(\boldsymbol{x}_0, \cdots, \boldsymbol{x}_T)$ should match the joint distribution of the forward Markov Chain $q(\boldsymbol{x}_0, \cdots, \boldsymbol{x}_T)$. The ELBO, which is maximised during training is given as[33]:

$$\mathbb{E}[-\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0)] \leq \mathbb{E}_{q(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)} \left[-\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_T) - \sum_{t=1}^{T} \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})} \right]. \tag{4.33}$$

The ELBO can be rewritten in terms of KL-Divergences as:

$$\mathbb{E}_q \left[\underbrace{D_{\mathrm{KL}}(q(\boldsymbol{x}_T|\boldsymbol{x}_0)||p(\boldsymbol{x}_T))}_{L_0} + \sum_{t=1}^{T} \underbrace{D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0, \boldsymbol{x}_T)||p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))}_{L_t} - \underbrace{\log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)}_{L_T} \right].$$

The L_T term is not dependent on the NN parameter and hence can be left out during the optimisation. The authors of Ref. [145] found that omitting the L_0 term benefits the optimisation. Thus, the only terms that remain are the KL-divergences between Gaussian distributions, for which an analytical form is available under an assumption. If the variance of the reverse distribution is assumed to be fixed, minimising this KL-divergence and as such, maximising the ELBO can be done by minimising the distance between the means of the two Gaussian distributions q and p_θ. However, the authors also found that predicting the noise, instead of the means leads to better

[32] As for the VAE, the denominator is not tractable because it would require the whole dataset for every timestep.

[33] respectively usually the negative ELBO is minimized for computational reasons as before.

results[34]. Thus, the loss is given as:

$$L_{\boldsymbol{\theta}} := \mathbb{E}_{t\sim U(0,1),\boldsymbol{x}_0\sim q(\boldsymbol{x}_0,\boldsymbol{\epsilon}\sim\mathcal{N}(0,1)}\left[||\epsilon - \epsilon_{\boldsymbol{\theta}}(\underbrace{\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}}_{\boldsymbol{x_t}}, t)||^2\right], \tag{4.34}$$

where $\epsilon \sim \mathcal{N}(0,1)$, $\epsilon_{\boldsymbol{\theta}}$ is the output of the NN, $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=0}^{t} 1 - \beta_s$ is defined by the chosen variance schedule, which defines how the noise is added. Usually, the variance of the noise is increased monotonically: $\beta_1 < \beta_2 < .. < \beta_T$. Here, it is worth noting that no constraints are required for the NN, except that the output dimension of the NN needs to match the input dimension of the NN, similar to the NF shown in Fig. 4.9. To draw new samples, the algorithm given in Alg. 3 is used, where σ_t are time-dependent constants defined by the variance schedule.

Algorithm 3 Sampling the DDPM

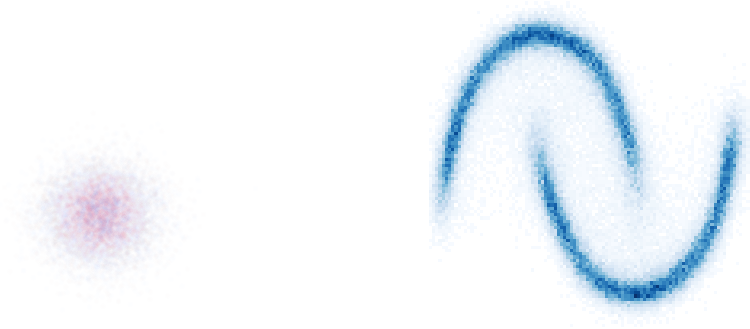
1: $\boldsymbol{x}_T \sim \mathcal{N}(0, I)$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \boldsymbol{z} \sim \mathcal{N}(0, I)$ **if** $t > 1$ **else** $\boldsymbol{z} = 0$
4: $\quad \boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\alpha_t}}\epsilon_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\right) + \sigma_t \boldsymbol{z}$
5: **end for**
6: **return** $\boldsymbol{x}_0$

4.6.1 Two Moons Dataset

The left of Fig. 4.18 depicts samples drawn from the trained model. Similar to the samples from the NFs, the samples are nearly indistinguishable from the real samples, although for this model, there are some samples between the two crescents. The right of Fig. 4.18 shows the noised data distribution. In red, points corresponding to the upper moon are shown, in blue the points corresponding to the lower moon. This illustrates that for a DDPM the latent space, unlike previous models, does not contain any encoding of the input. Figure 4.19a depicts the reverse diffusion chain $p_{\boldsymbol{\theta}}$ at different time steps $t = [0, 20, 40, 60, 80, 100]$. This time evolution shows that the

[34]The informed reader might notice here that the part that the losses at different time steps have a different weight which was not denoted in the loss formulation. However, Ho et al. also found that ignoring these terms leads to better results.

model starts forming the two crescents after 80 steps, before the latent distribution is indistinguishable from noise.



(a) Diffused inputs (b) Samples

Figure 4.18: Results from the DDPM. (left) The diffused inputs; blue points represent the lower crescent, red points the upper crescent. (right) Samples drawn from the trained diffusion model.

4.7 CNFs & Flow Matching

As described in Section 4.5, due to the inefficient simulation-based training requiring the numerical solution of an ODE, CNFs were not a viable option. Lipman et al. [147] addressed this inefficiency by introducing a novel method for the training of CNFs, which they refer to as *Flow Matching*. They propose to train a NN $\boldsymbol{v}_t^{\boldsymbol{\theta}}$ to learn to approximate the vector field $\boldsymbol{u}_t$, given before[35] in Eq. 4.25. To train this NN, an MSE is used:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t,p_t(\boldsymbol{x})}||\boldsymbol{v}_t(\boldsymbol{x}) - \boldsymbol{u}_t(\boldsymbol{x})||^2. \tag{4.35}$$

But there is no direct access to p_t and $\boldsymbol{u}_t$. Thus, the *conditional probability path* $p_t(\boldsymbol{x}|\boldsymbol{x}_1)$ is introduced, which is conditioned on a particular data sample $\boldsymbol{x}_1$, and

[35]Note that in the following the superscript for the parameter θ will be omitted, however $\boldsymbol{v}_t$ always refers to the NN.

fulfils:

$$p_{t=0}(\boldsymbol{x}|\boldsymbol{x}_1) = p(\boldsymbol{x}) \tag{4.36}$$

$$p_{t=1}(\boldsymbol{x}|\boldsymbol{x}_1) = \mathcal{N}(0, \sigma_{\min}\mathbb{I}), \quad \sigma_{\min} << 1 \tag{4.37}$$

Thus, at the start the conditional probability path is the base distribution p and at $t = 1$ it is a Gaussian concentrated around $\boldsymbol{x}_1$. The conditional probability path is then marginalised over the distribution of all samples $q(\boldsymbol{x}_1)$:

$$p_t(\boldsymbol{x}) = \int p_t(\boldsymbol{x}|\boldsymbol{x}_1)q(\boldsymbol{x}_1)d\boldsymbol{x}_1, \tag{4.38}$$

which closely approximates that data distribution q at $t = 1$. Corresponding to this, a *marginal* vector field is defined, which generates the marginal probability path $p_t(\boldsymbol{x})$:

$$\boldsymbol{u}_t(\boldsymbol{x}) = \int \boldsymbol{u}_t(\boldsymbol{x}|\boldsymbol{x}_1)\frac{p_t(\boldsymbol{x}|\boldsymbol{x}_1)q(\boldsymbol{x}_1)}{p_t(\boldsymbol{x})}d\boldsymbol{x}_1, \tag{4.39}$$

where the conditional vector field $\boldsymbol{u}_t(\cdot|\boldsymbol{x}_1)$ corresponds to the conditional probability path $p_t(\cdot|\boldsymbol{x}_1)$. Now, the unknown and intractable marginal vector field $\boldsymbol{u}_t$ can be broken down into simpler conditional vector fields $u_t(\boldsymbol{x}|\boldsymbol{x}_1)$ that depend solely on a single data sample.
However, the integrals in equation 4.38 and 4.39 are still intractable due to the integral over $d\boldsymbol{x}_1$ and hence there is no access to $\boldsymbol{u}_t$ to train the NN $\boldsymbol{v}_t$. Luckily, the Conditional Flow Matching objective:

$$\mathcal{L}_{CFM}(\theta) = \mathbb{E}_{t,\boldsymbol{q}(\boldsymbol{x}_1),p_t(\boldsymbol{x}|\boldsymbol{x}_1)}||\boldsymbol{v}_t(\boldsymbol{x}) - \boldsymbol{u}_t(\boldsymbol{x}|\boldsymbol{x}_1)||^2, \tag{4.40}$$

has the same gradients as $\mathcal{L}_{\text{FM}}$ given in Eq. 4.35. This means that the CNF can be trained to generate the marginal probability path p_t without having access to either the marginal probability path or the marginal vector field.

4.7.1 Conditional Vector Field & Probability Path

The conditional probability path is chosen to be Gaussian:

$$p_t(\boldsymbol{x}|\boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_t(\boldsymbol{x}_1), \sigma_t(\boldsymbol{x}_1)^2\boldsymbol{I}), \tag{4.41}$$

where $\boldsymbol{\mu}_{t=0}(\boldsymbol{x}_1) = 0$ and $\sigma_{t=0}(\boldsymbol{x}_1) = 1$, meaning that all conditional probability paths converge to the same standard Gaussian distribution at $t = 0$. As mentioned previously, at $t = 1$ the conditional probability path is chosen to be a Gaussian concentrated around $\boldsymbol{x}_1$ with a tiny variance σ_{min}:

$$p_1(\boldsymbol{x}|\boldsymbol{x}_1) = N(\boldsymbol{x}|\boldsymbol{x}_1, \sigma_{\text{min}}^2\mathbb{I}). \tag{4.42}$$

Next, the functional form of the CNF is chosen as:

$$\boldsymbol{\phi}_t(\boldsymbol{x}) = \sigma_t(\boldsymbol{x}_1)\boldsymbol{x} + \boldsymbol{\mu}_t(\boldsymbol{x}_1), \tag{4.43}$$

and its corresponding conditional vector field $\boldsymbol{w}(\boldsymbol{x}) = \boldsymbol{u}_t(\boldsymbol{x}|\boldsymbol{x}_1)$ is derived as follows starting from Eq. 4.25:

$$\dot{\boldsymbol{\phi}}_t = \boldsymbol{w}_t(\boldsymbol{\phi}_t(\boldsymbol{x})) \qquad \text{where } \boldsymbol{w}_t(\boldsymbol{x}) = \boldsymbol{u}_t(\boldsymbol{x}|\boldsymbol{x}_1) \tag{4.44}$$

$$\dot{\boldsymbol{\phi}}_t(\boldsymbol{\phi}_t^{-1}(\boldsymbol{y})) = \boldsymbol{w}_t(\boldsymbol{y}) \qquad \boldsymbol{\phi}_t \text{ invertible, } \boldsymbol{x} = \boldsymbol{\phi}_t^{-1}(\boldsymbol{y}) \tag{4.45}$$

$$\boldsymbol{\phi}_t^{-1}(\boldsymbol{y}) = \frac{\boldsymbol{y} - \boldsymbol{\mu}_t(\boldsymbol{x}_1)}{\sigma_t(\boldsymbol{x}_1)} \qquad \text{inverting } \boldsymbol{y} = \sigma_t(\boldsymbol{x})\boldsymbol{x} + \boldsymbol{\mu}_t(\boldsymbol{x}_1) \tag{4.46}$$

$$\dot{\boldsymbol{\phi}}_t(\boldsymbol{x}) = \dot{\sigma}_t(\boldsymbol{x}_1)\boldsymbol{x} + \dot{\boldsymbol{\mu}}_t(\boldsymbol{x}_1) \qquad \text{time derivative of Eq. 4.43} \tag{4.47}$$

$$\boldsymbol{w}_t(\boldsymbol{y}) = \frac{\dot{\sigma}_t(\boldsymbol{x}_1)}{\sigma_t(\boldsymbol{x}_1)}(\boldsymbol{y} - \boldsymbol{\mu}_t(\boldsymbol{x}_1)) + \dot{\boldsymbol{\mu}}_t(\boldsymbol{x}_1) \qquad 4.46 \text{ and } 4.47 \text{ into } 4.45. \tag{4.48}$$

Finally, a choice for the functions $\boldsymbol{\mu}_t(\boldsymbol{x}_1)$ and $\sigma_t(\boldsymbol{x}_1)$ needs to be made. These functions can be chosen such that the deterministic probability flows from diffusion [148] are obtained[36], highlighting that CNFs can be considered a generalisation of diffusion models. But the benefit of the CFM framework is that these functions can be chosen

[36]Note, that for this equivalence the *variance exploding path*, which defines how the noise is added, was chosen. The choice of probability was skipped in the previous section about DDPMs. The interested reader can consult [145, 148, 147] for further information on the effect of different probability paths.

as:

$$\boldsymbol{\mu}_t(\boldsymbol{x}) = t\boldsymbol{x}_1, \text{ and } \sigma_t(\boldsymbol{x}) = 1 - (1 - \sigma_{min})t, \tag{4.49}$$

where the mean and the standard deviation change linearly in time. The resulting vector field is then given by 4.48:

$$\boldsymbol{u}_t(\boldsymbol{x}|\boldsymbol{x}_1) = \frac{\boldsymbol{x}_1 - (1 - \sigma_{min})\boldsymbol{x}}{1 - (1 - \sigma_{min})t}. \tag{4.50}$$

From this, the resulting CNF can be derived with Eq. 4.43 as:

$$\boldsymbol{\phi}_t(\boldsymbol{x}) = (1 - (1 - \sigma_{min})t)\boldsymbol{x} + t\boldsymbol{x}_1 \tag{4.51}$$

This CNF is, in fact, the Optimal Transport [149] (OT) displacement map between the two Gaussian $p_0(\boldsymbol{x}|\boldsymbol{x}_1) = \mathcal{N}(0, 1)$ and $p_1(\boldsymbol{x}|\boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x}_1, \sigma^2_{min})$. The advantage of choosing these OT paths over the diffusion paths is that they are straight and not curved as the latter. The curvature matters to the ODE solvers as more steps are needed to approximate the solution of the ODE well enough, signifying that the OT probability paths can lead to a more efficient generation.

4.7.2 Extending to Different Sources

Tong et al. [150] provide a framework such that the source distribution can be arbitrary and, more importantly, does not need to be tractable. This is done by the following modification, where they identify the former condition $\boldsymbol{x}_1$ as an independent pair $\boldsymbol{z} = (\boldsymbol{x}_0, \boldsymbol{x}_1)$. They assume once again that the conditionals are Gaussian flows between $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$, with standard deviation σ_{min}:

$$p_t(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|t\boldsymbol{x}_1 + (1 - t)\boldsymbol{x}_0, \sigma^2_{min}) \qquad \text{where } q(\boldsymbol{z}) = q(\boldsymbol{x}_0)q(\boldsymbol{x}_1) \tag{4.52}$$

$$u_t(\boldsymbol{x}|\boldsymbol{z}) = \boldsymbol{x}_1 - \boldsymbol{x}_0 \qquad \text{this follows from 4.48.} \tag{4.53}$$

Note, that now the conditional probability path at $t = 0$ is not a standard Gaussian any more, but a Gaussian with a small variance around $\boldsymbol{x}_0$. Also note that the vector field now is independent of the time. To be more precise, insert $\boldsymbol{\mu}_t = t\boldsymbol{x}_1 + (1 - t)\boldsymbol{x}_0$ and $\sigma_t = \sigma_{min}$ into 4.48 to obtain the last equality.

4.7.3 Optimal Transport CFM

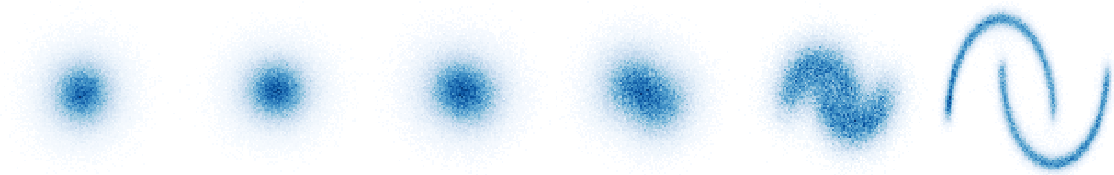
In the same publication [150], Tong et al. also emphasize that although the **conditional** probability path $p_t(\boldsymbol{x}|\boldsymbol{x}_1)$ is an optimal transport path, the **marginal** probability path is **not** in general an OT path.

To resolve this, the condition $\boldsymbol{z}$ is not sampled independently from $q(\boldsymbol{z}) = q(\boldsymbol{x}_0)q(\boldsymbol{x}_1)$ but from the optimal transport plan $\pi(\boldsymbol{x}_0, \boldsymbol{x}_1)$. For this choice, the marginal path p_t and vector field u_t solve the optimal transport problem. The construction of this OT transport plan is out of the scope of this thesis, the interested reader should consult Ref. [151].

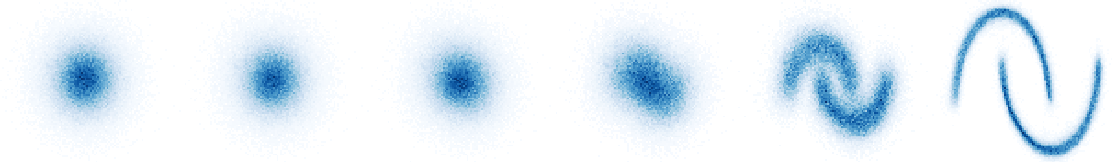
4.7.4 Two Moons Dataset

On the Two Moons dataset, the differences of DDPMs, and CNFs trained with Flow Matching and OT Flow Matching, can be nicely illustrated. The ODE is solved with 100 time steps. Note that similarly to the DDPM, this iterative sampling approach makes CNF slow at generating data[37]. In Fig. 4.19, the time evolution of the base density is shown at times $t = k \times 20,\ k \in [0, 5]$. For the Diffusion model, the distribution of the samples seems indistinguishable from a standard Gaussian. Similarly, for the CNF trained with CFM model, not a big difference between the first time steps can be recognized. There is a slight difference visible between these two models, as there seems to be more structure in step 80 of the CNF trained with CFM. But, the biggest contrast can be seen for the OT-CFM. Already from the 20^{th} time step, the samples are already split into two separate sections. This hints at the *interpolating* capability of CNFs trained with OT-CFM that will be further investigated in Chapter 7.

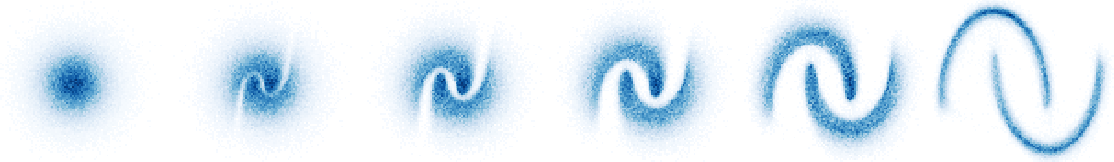
[37] Or in the context of the DDPM: the diffusion process has 100 noising steps.



(a) Diffusion Model DDPM



(b) Conditional Flow Matching



(c) Optimal Transport Conditional Flow Matching

Figure 4.19: Time evolution of samples from the diffusion model, respective continuous normalising flow during the solution of the ODE over 100 steps. Going from left to right, the sample state at $t = k \times 20,\ k \in [0, 5]$ is shown.

4.8 Evaluation Metrics

A major problem in generative modelling is the absence of a general metric to evaluate the performance of the generative model. Here, some options are briefly discussed.

4.8.1 Wasserstein Distances

The Wasserstein distance is a promising candidate to quantify the distance between two distributions. Given two probability densities p_X, p_Y, the p-Wasserstein distance is defined as:

$$W_p(p_X, p_Y) = \left(\inf_{\pi \in \Pi} \int_{\mathbb{R}^d \times \mathbb{R}^d} |\boldsymbol{x} - \boldsymbol{y}|^p d\pi(x, y) \right)^{\frac{1}{p}}, \tag{4.54}$$

where Π denotes the set of all joint distributions on $\mathbb{R}^d \times \mathbb{R}^d$ whose marginals are p_X and p_Y. Unfortunately, it is not tractable due to the infimum over Π. The

WGAN introduced in Section 4.3.1 estimates the Wasserstein distance by utilising an NN to distinguish the two distributions. However, enforcing the classifier to be a Lipschitz-continuous function is not straightforward. Furthermore, it is only a precise estimate of the Wasserstein distance if the classifier has enough capacity and is trained to optimality. There are two special cases for the tractability of the Wasserstein distance:

1. For distributions over a single dimension, the W_1 distance between p_X and p_Y is given by:
$$W_1(p_X, p_Y) = \int_{-\infty}^{\infty} |F_X(x) - F_Y(x)| dx, \tag{4.55}$$
where F_X and F_Y are the cumulative distributions of p_X and p_Y, which are in practice approximated by the empirical cumulative distribution functions. This is especially useful if there are summary statistics available, that allow testing whether the model generates correlations between the features correctly. However, it should still be noted that information is lost if a high-dimensional distribution is marginalised to a single dimension.

2. The Fréchet Inception Distance [73] (FID) gained popularity as an evaluation metric in image generation, and it is currently used as a standard metric to quantify the performance of image generation models. The FID is motivated by the tractability of the Wasserstein-2-distance W_2 if the distributions under consideration are Gaussian:
$$W_2^2(\mathcal{N}_1, \mathcal{N}_2) = \|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|_2^2 + \mathrm{Tr}(\Sigma_1 + \Sigma_2 - 2\sqrt{\Sigma_1 \Sigma_2}), \tag{4.56}$$
where $\boldsymbol{\mu}_i, \Sigma_i$ are the means and covariance matrices of the Gaussian distributions. The mean μ and variance Σ of the latent distribution inside a NN define a multivariate Gaussian distribution. Thus, the Wasserstein distance can be estimated by fitting a multivariate Gaussian to the features of a final layer of a pre-trained image classifier [152] for real and generated data[38]. Note, that this does not influence the training of the generative model at all and only serves

[38]The classifier which is used is called Inception-v3, hence the name of the metric. Furthermore, note that this classifier is not trained to distinguish synthetic and real images, but to infer information about the objects in the image.

as an evaluation metric.
Chong et al. [153] show that the FID is positively biased for a finite sample size N. They find that the bias is in a linear relationship with $\frac{1}{N}$ and thus propose the FGD_∞ metric, which is computed by linearly fitting the FID as a function of $\frac{1}{N}$. The value of a FID for an infinite sample size is given by the intercept of the y-axis and the linear regression fit.

4.8.2 Classifier-based Metrics

NN-based classifiers can also be used to quantify the performance of generative models. A straightforward approach is to use a binary classifier. The classifier is trained to distinguish real and synthetic samples and then evaluated by considering its Receiver Operator Curve (ROC) [154]. The ROC shows the dependence of the true positive rate vs the false positive rate. For a perfect classifier, the ROC rises vertically to the top-left corner. In contrast, a classifier that is random guessing is represented by the diagonal. The Area-Under-Curve (AUC), which is the integral of the ROC, can then be used to quantify the performance of a model. This method will be used in Chapter 6 and Chapter 7.
Another option to compare different models is proposed in Ref. [155]. First, a multiclass classifier, i.e. a classifier that is trained to distinguish multiple different classes, is trained to distinguish data from multiple generative models.
After training, the multiclass classifier is evaluated on ground truth data. This assumes that the classifier will assign the highest output to the one model, which produces the most realistic data. Thus, the best model is then determined by the model to which the multiclass classifier assigns the highest output $\hat{y}$ on average over all samples. This method is applied to the models introduced in this chapter and will be used in Chapter 5.

Two Moons Dataset

Figure 4.20 visualises the distribution of the different models that are compared in the multiclass classifier test. Note that evaluation with the multiclass classifier on these models is not a prime example, since the support of the learned density of all models is similar. This makes it difficult to obtain an accurate multiclass classifier, which

impacts the quality of the evaluation significantly. The multiclass classifier is trained on $100'000$ samples of every model. On the left of Fig 4.21 the confusion matrix of the multiclass classifier is shown and highlights that the multiclass classifier can recognize some models to a certain extent[39]. Each row of the confusion matrix represents the ground truth class, each column represents the predicted class. The diagonal gives the correctly predicted classes, and all off-diagonal values are the misclassifications. On the right of Fig. 4.21, the evaluation of the multiclass classifier on real data is shown. Although the multiclass classifier can identify the worse-performing models, it is unable to recognise the mode-collapse present in GANs. In Appendix A, the same test is evaluated on only three classes to rule out that the multiclass classifier is unable to recognise mode collapse due to the many classes. An issue of the latter method is that there are generally no guarantees for the performance of an NN-based model on data from a distribution unseen during training.

[39]For a random guessing multiclass classifier, the prediction on all classes would be equally often, since the sample size is the same for every model.

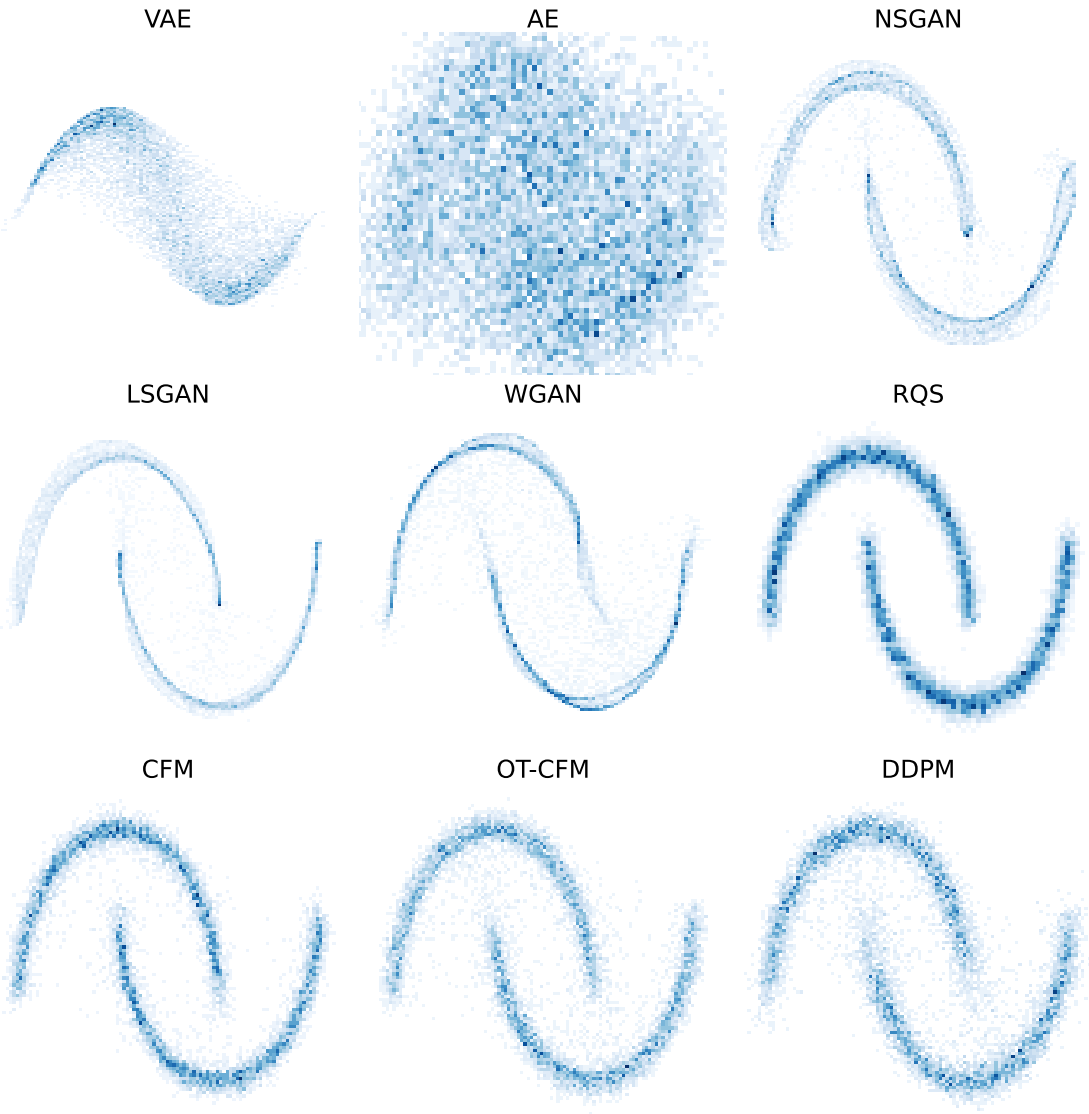


Figure 4.20: Comparison of samples drawn from all models discussed in this chapter. These models are then further used to train a multiclass classifier to distinguish them. From visual evaluation, the RQS model is expected to perform best.

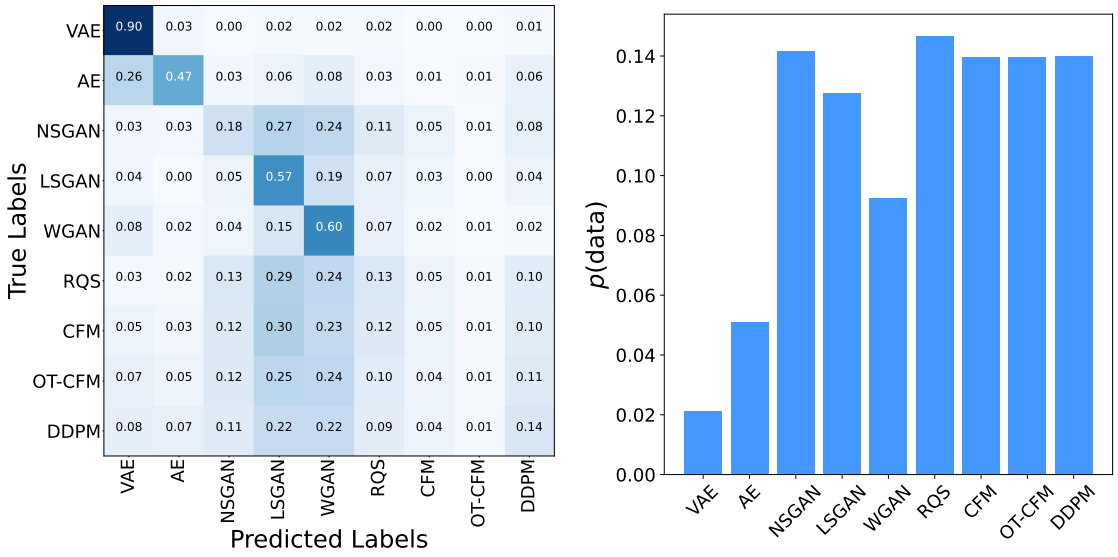


Figure 4.21: Results of the multiclass classifier test. (left) Confusion matrix of the classifier, showing that it can distinguish between some classes. (right) The average probability to come from the different models under test, which the multiclass classifier assigns to real data.

CHAPTER 5

Generative Modelling on Point Clouds I: `JetNet`

This chapter describes the studies that were conducted on the `JetNet` datasets during this thesis in chronological order. It starts with the introduction of the `JetNet` [156] datasets, which had a significant impact on this thesis. They allowed the incremental growth of the models and their adaptation to increasingly challenging problems. I was fortunate to start investigating deep-generative modelling of HEP data with NFs. While their workings are not as intuitively understandable as GANs or VAEs, their stability offers a strong basis for further investigations.
Since the modelling of variable-sized data is not straightforward, two methods used in this thesis are briefly discussed in Section 5.2. In Section 5.3, the evaluation methods, which are employed to compare and quantify the performance of different generative models, are briefly introduced.
Then, in Section 5.4 follows a larger part containing all studies that were conducted on the `JetNet30` datasets. Mass conditioned and constrained NFs were explored in the first project of this thesis [157]. In the following, permutation equivariant extensions of NFs are briefly discussed and the encountered failure modes are highlighted. Finally, the NF-based model is fused with a GAN[1], which led to the second major result [158]. In Section 5.4.5, the results from the different models are quantitatively compared on the `JetNet30` datasets, together with, the at the time of study, State-

[1] At this point I was unaware of the mode collapse problems highlighted in the previous chapter.

of-the-Art (SotA) Message Passing GAN (MPGAN).
The natural next step to move to the higher-dimensional JetNet150 datasets is discussed in Section 5.5. These datasets include up to five times as many particles, causing significant difficulties for the previously used models, for which the memory use and computational complexity scaled quadratically with the number of particles. Eventually, the stability provided by the NF was abandoned in favour of a solely GAN-based solution that employed the mean-field aggregation mechanism that scales linearly with the number of particles. Together with some more minor improvements, this led to the third major result [159]. The proposed model turned out to be a versatile, as it is not only performing well on JetNet150, but also in calorimeter simulation, which will be discussed in Chapter 6.
Finally, instead of a GAN, a CNF trained with CFM is used. It also relies on the mean-field aggregation as the GAN, but further improves the results on all considered metrics.

5.1 JetNet Datasets

A point cloud representation, i.e. an unordered set of points, is arguably the most natural representation of a jet. In the context of the JetNet datasets, a point refers to a final state particle. These two expressions will be used interchangeably in the following. The JetNet datasets were first published by Kansal et al. [156] along with metrics, which are further discussed in Section 5.3.1, to compare the performance of different models. The authors compared their MPGAN, to different SotA generative point cloud models. They demonstrated that none of the existing point-cloud-based models were able to capture the complex correlation between the individual particles.
The JetNet datasets consist of five datasets of the decay of different partons produced at leading order with MADGRAPH5_aMCATNLO 2.3.1 [160]. These partons have a high transverse momentum of $p_{\mathrm{T}} = 1$ TeV with a spread of $\frac{\Delta p_{\mathrm{T}}}{p_{\mathrm{T}}} = 0.01$. The parton-level objects are decayed and showered with PYTHIA 8.212 [161]. The final-state particles obtained from PYTHIA are then clustered into jets using the anti-k_T algorithm with a distance parameter of $R = 0.8$.
The jet-initiating particle can be either a gluon, a light/top quark or a W/Z boson and the number of jet constituents can be selected to be up to 150. However, models

that explore these datasets are typically trained on the reduced datasets (JetNet30), which contain up to 30 particles, or the full datasets, which contain up to 150 particles per jet (JetNet150).
The final state particles comprising the jet are assumed to be massless, which is a valid assumption given that the initiating parton has an energy of 1 TeV. Thus, the particles are fully described by their 3-momenta or, equivalently, by their transverse momentum p_T, pseudorapidity η and azimuthal angle ϕ. The jets are centred in η, ϕ and the coordinates of particle i in the jet are given relative to the jet axis[2]:

$$\eta_i^{\mathrm{rel}} := \eta_i^{\mathrm{particle}} - \eta^{\mathrm{jet}}, \tag{5.1}$$

$$\phi_i^{\mathrm{rel}} := (\phi_i^{\mathrm{particle}} - \phi^{\mathrm{jet}}) \bmod 2\pi, \tag{5.2}$$

$$\tag{5.3}$$

The transverse momentum of the particles is given relative to the jet momentum:

$$p_{\mathrm{T,i}}^{\mathrm{rel}} := \frac{p_{\mathrm{T,i}}^{\mathrm{particle}}}{p_\mathrm{T}^{\mathrm{jet}}}. \tag{5.4}$$

The particles are ordered by decreasing relative transverse momentum $p_{\mathrm{T,i}}^{\mathrm{rel}}$.
An essential high-level feature for many physical analyses is the invariant mass m_{jet} of a jet, which contains important physical information about the parent particle. It is a global variable that depends on the correlations between all individual constituents of the jet. Therefore, it is an important statistic for evaluating the performance of different models on these datasets. As it is a one-dimensional variable, the use of the Wasserstein distance is particularly motivated to quantify the performance, as discussed in Section 4.8.1. For the relative quantities given above in Eq. 5.1, the relative invariant jet mass is defined as[3]:

$$\left(m^{\mathrm{rel}}\right)^2 = \frac{m_{\mathrm{jet}}^2}{p_{\mathrm{T,jet}}^2} = \left(\sum_i E_i^{\mathrm{rel}}\right)^2 - \left(\sum_i \boldsymbol{p}_i^{\mathrm{rel}}\right)^2. \tag{5.5}$$

[2]However, note that this centring is done on the PYTHIA level, where the true jet axis is calculated from all final state particles. The jet axis that comes out of the anti-k_T clustering only considers 150 particles, and as such does not exactly overlap with the PYTHIA-level jet axis. This means that adding up the η^{rel} of all particles in a jet does not result in exactly 0.

[3]In the following, the relative invariant jet mass is abbreviated as the mass.

For brevity, the JetNet studies presented in this thesis, solely focus on the top quark dataset [162]. This dataset is chosen due to the complex substructure of top quark jets. The favoured decay $t \to bW$ leads to a 3-pronged jet, for which the clustering can lead to artificial artefacts, as it can miss the decay products related to the b quark in the clustering. This leads to a two-peaked mass distribution, that is particularly difficult to model accurately.
Note that the results presented in this thesis are not identical to those in the corresponding preprints [157, 158, 163, 159], since all models were combined in the same framework and retrained from scratch to ensure a fair comparison.
Before training, the data is further pre-processed in two possible ways:

1. For the NF-based models, it proved to be beneficial to use a Box-Cox scaling [81] for $p_{\mathrm{T}}^{\mathrm{rel}}$, since the $p_{\mathrm{T}}^{\mathrm{rel}}$ distribution is strongly skewed and follows an approximately exponential distribution. The coordinates $\eta^{\mathrm{rel}}, \phi^{\mathrm{rel}}$ are standard scaled, i.e. the mean of the variable is subtracted and divided by its standard deviation.

2. The GAN-based models performed worse when a Box-Cox scaling is used; hence, standard scaling is used on $\eta^{\mathrm{rel}}, \phi^{\mathrm{rel}}, p_{\mathrm{T}}^{\mathrm{rel}}$ to preprocess the data.

Note that the parameters for these preprocessing transformations are calculated from the set of all particles and all jets in the training set.

5.2 Modelling Variable Sized Data

Jets can have a variable number of constituents, which means that the generative model also needs to output a variable number of particles. This is difficult to implement for most generative models, since NNs are inherently based on matrix multiplication, which expects a regular shaped input. In this thesis, two strategies were used to address this.

1. *Padding* and *flattening*: to every point cloud **0**s are added until they reach the cardinality of the biggest point cloud. Then the input is reshaped from $\mathbb{R}^{n \times p \times f}$ to $\mathbb{R}^{n \times (p \cdot f)}$, where n is the number of particles, p is the maximum number of particles and f is the number of features per point. The latter step is referred

to as flattening. However, the former step is problematic if the variance in the number of constituents is high, as the added zeros consume a lot of memory without adding any information. The flattening removes the particle structure present in the input[4].

2. Pointwise NNs and permutation equivariant aggregations: instead of flattening the tensor, the NNs are designed to be evaluated on the points independently. To include interactions, permutation equivariant aggregation operations are used. Two possible candidates for this are:

   (a) The DeepSets aggregation, discussed in Section 3.2.2,

   (b) attention-based aggregations, discussed in Section 3.3.4.

   A subtlety of this approach is that the input to the generative model must already have the same cardinality as the desired output. Therefore, the number of jet constituents must be provided implicitly by the dimension of the noise, which is given as input to the generative model. Thus, the model does not sample $p(\boldsymbol{x})$ directly, but $p(\boldsymbol{x}|n)$ instead. To still sample $p(\boldsymbol{x})$, first, the number of constituents $p(n)$ is sampled and then a sample of the conditional probability density $p(\boldsymbol{x}|n)$ is drawn. Fortunately, sampling a one-dimensional distribution is fairly simple, as discussed in the following.

5.2.1 Modelling the Number of Jet Constituents

One-dimensional probability densities can be sampled with the use of the *Probability Integral Transform*. It states that if X is a continuous random variable with a cumulative distribution function (CDF) F_X, then the random variable $Y = F_X(X)$ follows a uniform distribution on the interval $[0, 1]$. Since the CDF is a monotone one-dimensional function, it is fit with a monotonically increasing Piecewise Cubic Hermite Interpolating Polynomial [164] (PCHIP) provided by *SciPy* [165], which is invertible since it is strictly monotone. To draw new samples from the one-dimensional distribution $p(n)$, samples from the uniform distribution are drawn and evaluated

[4] An intuitive illustration why this can make it more difficult is provided by images. Flattening can also be done by concatenating all pixels from left to right, top to bottom. This results in a representation of the image as a line instead of a rectangle, which loses all the spatial structure.

with the inverted PCHIP. Note that this method is only usable if the random variable is continuous, which is not the case for the number of jet constituents. This is resolved by adding uniform noise $U(0, 1)$ the number of jet constituents becomes continuous.

Another approach, which is much easier to implement, was proposed in a later publication by Buhmann et al. [96], where a Kernel Density Estimate (KDE) is fit and sampled instead[5]. This approach is also adopted in the JetNet experiments shown in this thesis. KDE is closely related to histograms, but instead of choosing a fixed bin width and a step function to calculate the y value, in KDE non-negative functions, referred to as *kernels* are used:

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right). \tag{5.6}$$

The bandwidth h is a smoothing parameter that affects the width of the kernel. The kernel is chosen as a Gaussian in this thesis, and the bandwidth is chosen with Scott's Rule [167] :

$$K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2}. \tag{5.7}$$

Sampling from the KDE is straightforward; a sample used during the fit of the KDE is randomly sampled and noise is drawn from the Kernel and added to the sample. A comparison of samples from the KDE distribution to samples of the underlying PYTHIA distribution is shown in Fig. 5.1[6].

5.3 Evaluation

Evaluating the performance of the generative models on this dataset is not as straightforward as for the Two Moons dataset in Chapter 4, since it is not possible to visualise a density over more than two dimensions. Still, the marginal histograms over a single variable of PYTHIA data and data drawn from the generative model

[5]There is no single definite reference or paper from which the Kernel Density Estimation algorithm can be cited. However, the interested reader should consult [166] for an in-depth explanation of the algorithm.

[6]Note that to obtain the number of constituents for the JetNet30 dataset, the sampled number is clamped at 30.

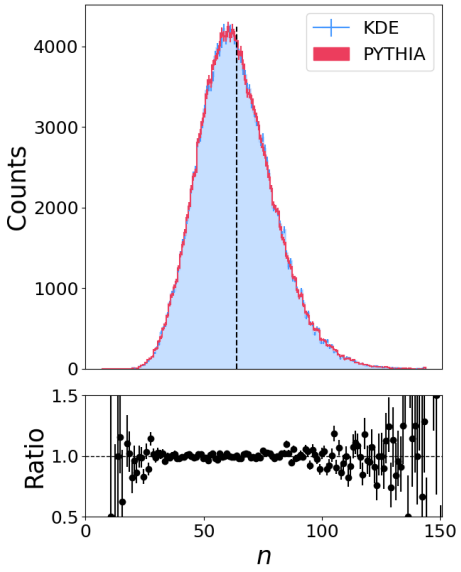


Figure 5.1: Comparison of the KDE (red) and PYTHIA (blue) generated distribution of the number of jet constituents n on the `JetNet150` top quark dataset. The dashed line marks the mean of the distribution, which is around sixty particles for the top quark dataset.

are compared. Good candidates are histograms of the features $(\eta^{rel}, \phi^{\text{rel}}, p_{\text{T}}^{\text{rel}})$ of all generated particles. As mentioned before, a more sensitive test is provided by the distribution of the invariant jet mass, since the jet mass is a high-level summary statistic that depends on all particles in the jet. Further visual tests are provided by heatmaps of the correlation matrices between the particle features $(\eta^{\text{rel}}, \phi^{\text{rel}}, p_{\text{T}}^{\text{rel}})$ of the $p_{\text{T}}^{\text{rel}}$ sorted particles in a jet. In those plots, the x- and y-axes denote the index of the i^{th} hardest particle. The correlation coefficients are computed with the linear Pearson correlation, which gives the covariance of two variables normalised by their individual standard deviations.

5.3.1 Metrics

Judging the performance of the model based on plots is not only tedious when conducting a hyperparameter scan, but also it is certainly not sufficient to judge whether the joint distribution is modelled correctly. Thus, Kansal et al. [156] also provide several metrics to quantify the performance of the generative models in the

publication of the JetNet datasets. In a later publication [168], Kansal et al. further discuss different metrics in-depth and conclude that a set of Wasserstein-1 distances together with the *Fréchet Particle Distance* (FPD) and *Kernel Physics Distance* (KPD) provides a comprehensive set of metrics for the analysis of the generated data. The Wasserstein-1 variables are calculated on the following three sets of variables:

1. The particle features $(\eta^{\text{rel}}, \phi^{\text{rel}}, p_{\text{T}})$,
2. the invariant relative mass m^{rel},
3. the Energy Flow Polynomials [169] (EFPs), which comprise a discrete linear basis for all infrared- and collinear-safe observables. EFPs are used to investigate the jet substructure.

The FPD is inspired by the FGD_∞ metric mentioned in 4.8.1 but adapted for HEP. To obtain an approximatively Gaussian joint distribution for which the Wasserstein distance can be calculated analytically, 36 EFPs (all EFPs of degree less than 5) are computed for every jet. The KPD is the Maximum Mean Discrepancy [170] calculated on the same EFPs with a polynomial kernel.
Other metrics have also been proposed by Buhmann et al. [171], where the use of KL-divergences is proposed. The authors argue that the benefit of Wasserstein distances is that they are bounded if the densities have non-overlapping support, which should not matter for a well-trained generative model. The authors highlight a failure mode of the Wasserstein metric and argue that for HEP, the compatibility of the learned and real density is more important. Leigh et al. [172] proposed to measure the Wasserstein-1 distance on additional jet substructure variables. Nevertheless, since the metrics proposed by Kansal et al. are well established, the models in this thesis will be evaluated on only this subset of metrics.
Note that to obtain a single number, the Wasserstein distance on the particle features (W_1^{P}) and the EFPs (W_1^{EFP}) are given as the inverse variance weighted mean given in Equation 5.8 of the individual Wasserstein distances x_i per feature i.

$$\bar{x}_w = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i}, \quad w_i = \frac{1}{\sigma_i^2}, \tag{5.8}$$

where σ_i^2 is the variance of the metric on feature $i \in (\eta^{\text{rel}}, \phi^{\text{rel}}, p_{\text{T}}^{\text{rel}})$. Similarly, the uncertainty associated with this weighted mean is calculated as:

$$\sigma_{\bar{x}_w} = \sqrt{\frac{1}{\sum_{i=1}^{n} w_i}}, \quad w_i = \frac{1}{\sigma_i^2}. \tag{5.9}$$

For all models, the checkpoint for evaluation is selected according to the lowest Wasserstein-1 distance computed between the mass distribution of $50'000$ samples from PYTHIA, which were also used during training, and $50'000$ samples drawn from the generative model[7].

[7]Due to the limited number of statistics and the susceptibility of the metrics on the sample size, it was decided not to use a validation set to select the best training checkpoint, but to use the training dataset. This allows for an independent testing set of $50'000$ samples.

5.4 Studies on JetNet30

In this section, the performance of different models on the JetNet30 datasets is discussed.

5.4.1 Normalising Flows on JetNet30

For the JetNet30 dataset, the data is padded and flattened, which results in a total number of 90 dimensions. Given that the generation of seemingly real images of faces is possible [137, 173, 174], which needs orders of magnitudes more dimensions, one might expect that this is an easy task[8]. However, a notable difference is that these models are usually more concerned with the sample quality rather than how well they capture the underlying true distribution. The sensitive summary statistics like the mass make it significantly easier to spot such mismodeling, as will be shown in the following.

Additional Preprocessing Steps for NFs

A subtlety of NFs is that padding of the clouds causes issues. Since all padded points lie on the same value, it is impossible to spread these singular values to a standard Gaussian distribution, which is the training objective of the NF. This is resolved by adding Gaussian noise with a variance of 10^{-8} to the padded particles. During sampling, particles with a $p_{\mathrm{T}}^{\mathrm{rel}} < 0.0001$ are set to 0.

Vanilla Normalising Flow

Autoregressive NFs are motivated by the decomposition of a joint probability distribution into the product of conditional ones. However, either the forward or the reverse direction of the autoregressive NF is D times slower, where D is the total dimension of the input space. This means that either sampling or training is 90 times slower on the JetNet30 datasets. From personal experience, the autoregressive NFs never outperformed the coupling NFs on this dataset. Therefore, the following results

[8]The dimension of an image is given by its number of pixel times the number of channels. A 1-megapixel RGB image, which current SotA models like DALL-E3[174] generate, hence has 3 Million features.

only concern coupling layer-based NFs. The most naive approach is to use RQS NFs on the flattened point cloud. This model is referred to as the Vanilla Normalising Flow (VNF) in the following.

Mass Conditioned Normalising Flows

As discussed in Section 4.4.3, NFs are conditioned by including additional variables for the NNs that predict the parameters of the coupling transformation. Conditioning increases the expressiveness of the transformation, as more information is given to the NNs predicting the parameters of the coupling layers. It also allows disentangling the latent space, as discussed in Section 4.4.4. For this study, the relative invariant jet mass m^{rel} is used to condition the NF. Note that this makes the NF not fully invertible any more, as the mass must be supplied when the NF is evaluated in both directions. However, since the NF is trained only in the forward direction, where the mass can be calculated from the input, it only needed to draw samples from the NF. To sample the mass distribution, the first approach from Section 5.2.1 is used, except the mass distribution is already continuous and hence no dequantisation is necessary. The mass conditioned NF is referred to as NF(c) in the following.

Mass Conditioned and Constrained Normalising Flows

To further improve the mass modelling, a *mass constraint* is introduced. This is implemented by introducing another loss term, which is minimised concurrently to the maximum likelihood training. The loss is constructed by sampling the NF and calculating the MSE between the condition $m^{\text{rel}}_{\text{cond}}$ supplied to the NF and the mass that is calculated from the generated jet:

$$L_{\text{mse}} = \left| m^{\text{rel}}_{\text{cond}} - m^{\text{rel}}(\boldsymbol{x}_{\text{gen}}(m^{\text{rel}}_{\text{cond}})) \right|^2 \tag{5.10}$$

This loss term is added to the negative log-likelihood loss of the NF with an additional parameter λ_m, which allows determining the importance of the constraint during the optimisation:

$$L_{\text{tot}} = L_{\text{nLL}} + \lambda_m L_{\text{mse}} \tag{5.11}$$

This means that the NF is evaluated in both directions during one training step, further ruling out the use of autoregressive NFs.

Training

All the three proposed models are trained for 2000 epochs using the AdamW optimiser with momentum coefficients $(\beta_1, \beta_2) = (0.9, 0.999)$. The hyperparameters for the different models are given in Table 5.1. The NF has 25 coupling layers, the RQS is constructed over $[-5, 5]$ with 6 bins. The parameters NNs in the coupling layers contain 3 residual blocks with 128 hidden features. For the NF and NF(c) λ_m is set to zero. For the NF(cc), the best results were obtained with $\lambda_m = 10$.

Parameter	NF
λ_m	0/10
`optimiser`	AdamW
`coupling_layers`	25
`residual_blocks`	3
`hidden_features`	128
`tail_bound`	5
`num_bins`	6

Table 5.1: Hyperparameters for different configurations

Results

The previously discussed NF approaches are compared in Fig. 5.2, which displays histograms of PYTHIA samples and synthetic samples. Going from left to right and top to bottom, the figure first depicts the three marginal distributions $\eta^{\text{rel}}, \phi^{\text{rel}}, p_{\text{T}}^{\text{rel}}$ of all generated points. These histograms demonstrate the effectiveness of NFs as already with the simplest approach (red), there is no strong mismodelling directly apparent in the marginal distributions. Some artefacts are visible when examining the ratio of yields from PYTHIA samples divided by the yields of samples from the generative model, shown below the histograms. These discrepancies are especially visible in the tails of the distributions. The histogram on the bottom-right raises more problematic issues for the VNF. It indicates that the VNF is unable to learn even the linear correlations between particles accurately. Conditioning the NF (yellow)

significantly improves the mass modelling, as now the mass distribution is not of Gaussian shape any more. It also removes most of the artefacts in $(\eta^{\text{rel}}, \phi^{\text{rel}}, p_{\text{T}}^{\text{rel}})$. As expected, the mass constraint (violet) further improves the modelling of the mass distribution.

Figure 5.3 depicts the Pearson correlation between the $p_{\text{T}}^{\text{rel}}$ ordered particles for PYTHIA samples and synthetic samples. While in general the correlations seem comparable to the one in PYTHIA-generated data, there are some spurious correlations for the VNF in Fig. 5.3b, e.g. in ϕ^{rel} close to the diagonal. For the conditioned NF, the linear correlations mostly remain the same when compared to the VNF, but the spurious strong negative correlations in ϕ^{rel} disappeared, as shown in Fig. 5.3c. However, in η^{rel} there are still some artefacts visible, especially for the 6^{th} hardest particle. When using the mass constraint, the linear correlations in η^{rel} and ϕ^{rel} again slightly improve. As a reference, the results whne not using Box-Cox preprocessing are given in Appendix B.

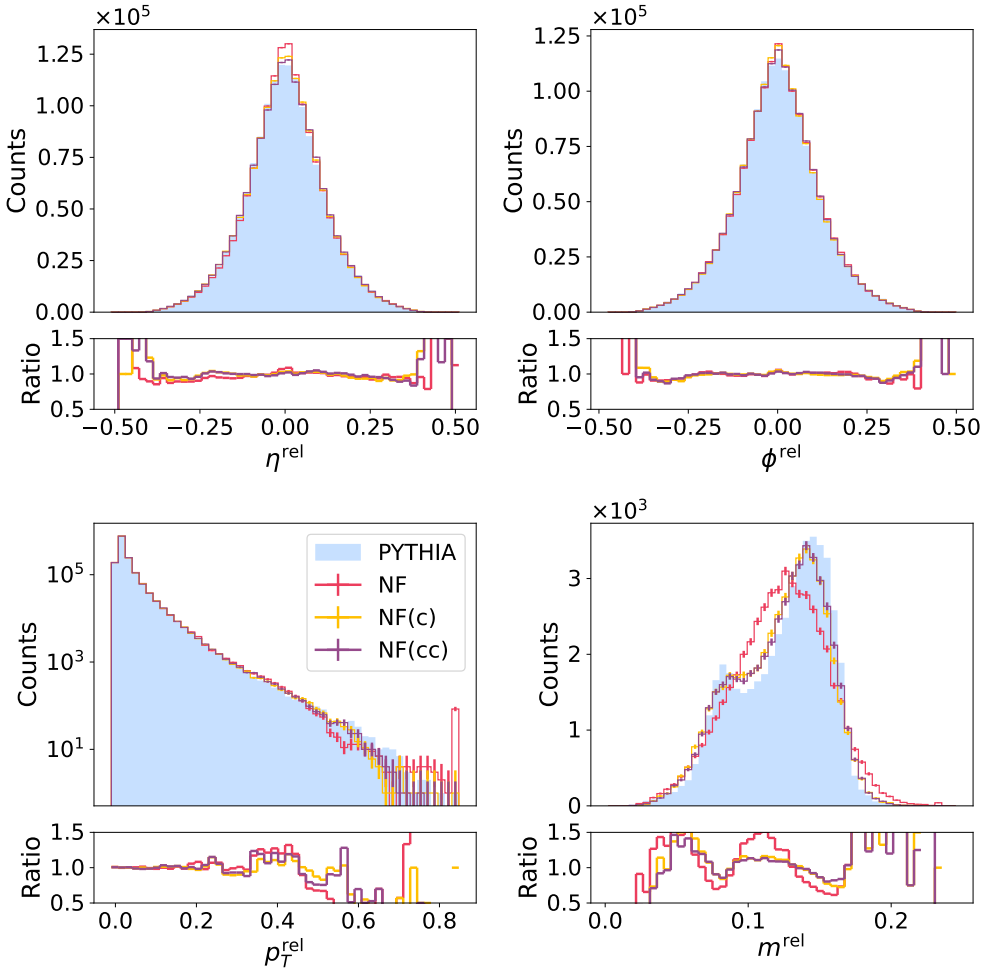
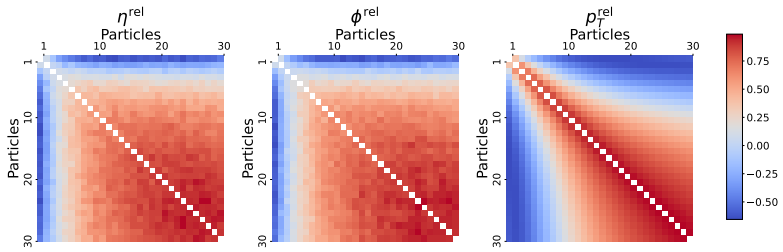
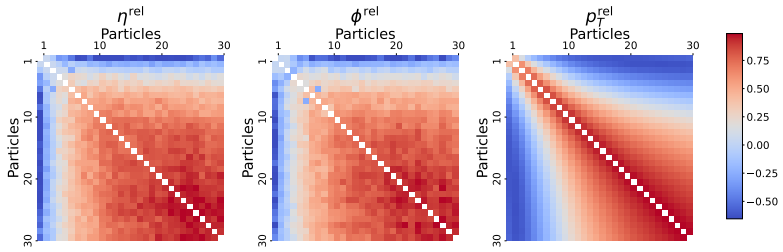


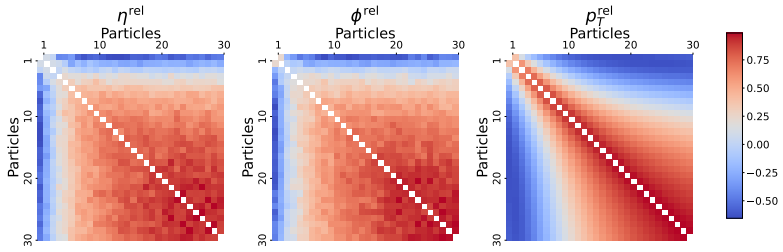
Figure 5.2: Comparison of synthetic samples from the different NF approaches to PYTHIA samples (blue). Below each plot, the yield ratio of PYTHIA data to synthetic data is shown. Going from left to right and top to bottom, the first three histograms depict the relative pseudorapidity, azimuthal angle and transverse momentum of all particles drawn from the model. The first three figures might indicate that the VNF (red) is sampling the underlying data distribution accurately. However, the mass distribution on the bottom right reveals that the VNF is unable to model the correlations between the particles accurately. Including the mass as a condition (yellow) improves the mass modelling significantly. Training the model with the mass constraint (violet) further improves the mass modelling.



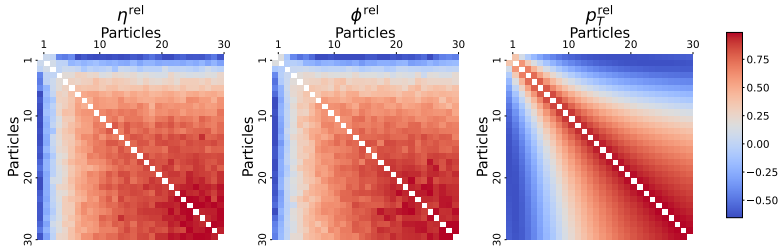
(a) PYTHIA



(b) VNF



(c) NF(c)

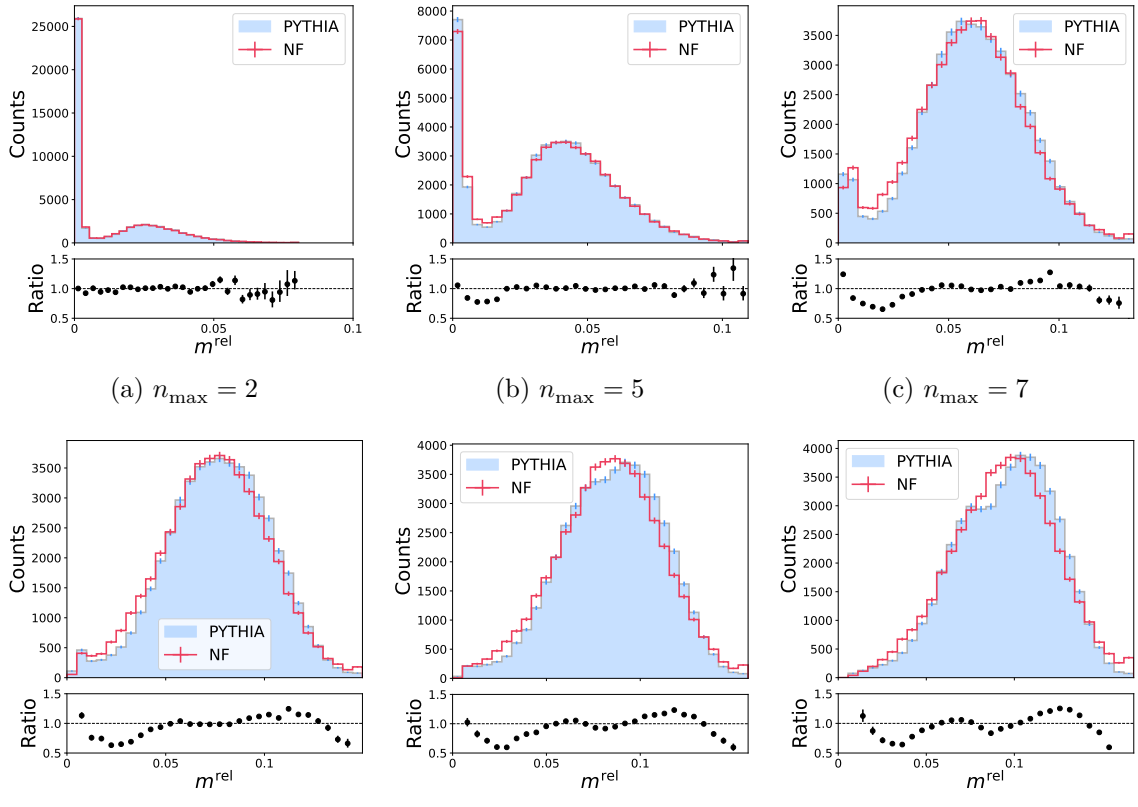


(d) NF(cc)

Figure 5.3: Heatmaps of the linear correlations between the individual features of the p_T^{rel} ordered particles of the proposed NF-based models. Every square in the figure represents the correlation between two particles. As a reference, the correlations calculated in the PYTHIA data are visualised in the top row.

Critical Dimension for Mass Modelling

In Section 4.4.4 the RQS NFs showed promising results on a fairly complicated density. To investigate whether the unsatisfying performance is only due to the higher dimension of the space over which the distribution is modelled, the number of jet constituents is reduced to n_{max} and the VNF is retrained. Figure 5.4 depicts the mass distribution for different choices of n_{max}. The figure indicates that the NFs



(a) $n_{\text{max}} = 2$ (b) $n_{\text{max}} = 5$ (c) $n_{\text{max}} = 7$

(d) $n_{\text{max}} = 10$ (e) $n_{\text{max}} = 12$ (f) $n_{\text{max}} = 15$

Figure 5.4: Mass distribution of jets, where only the hardest n_{max} particles per jet are considered in the clustering. For a low number of constituents, the VNF can model the mass distribution correctly. However, more than 7 particles in the jet lead to the model being unable to capture the mass distribution accurately.

can model the sharper edges of the mass distribution until 7 particles. Once more particles are added to the jet, the Gaussian shape, which is also visible as the red distribution in Fig. 5.2, emerges. This suggests that the higher number of dimensions

is causing the unsatisfying performance of the NF.

5.4.2 Permutation Equivariant NFs

The NFs that were used previously have no inductive bias designed to work well with point clouds. Respecting the permutation symmetry would be a beneficial property for the model. However, with the NF approach discussed here, all structure in the input, i.e. that there are 30 particles, is lost at the moment where the input is padded and flattened. Making the NNs in the coupling layer permutation equivariant did not lead to improved results. This can be understood since when the data is randomly split in the coupling layer, all structure is lost, even before the NNs are applied. Splitting the sets particle-wise also did not lead to improved results.
Thus, the permutation equivariance needs to be applied in the construction of the NF. Three different ideas to introduce permutation equivariance into the NFs were explored:

1. The most simplistic idea is to apply the NF independently to each particle. However, treating all points independently results in losing all correlations present between the points, which are strongly connected to the underlying physics of interest. Nevertheless, the results of this model are also shown in the following to give a baseline for comparison. This model is referred to as Independent Point Flow (IPF). A schematic of the model is given on the left of Fig. 5.5.

2. The second approach is based on PointFlow [175]. The authors argue that for a permutation invariant density, there exists a condition $\boldsymbol{z}$ with which the points are conditionally independent, referring to Finetti's theorem with the following lines:

   > De Finetti's representation theorem states that any exchangeable distribution can be written as a factored distribution, conditioned on a latent variable:

   $$P(X) = \int_z p_{\boldsymbol{\phi}}(\boldsymbol{z}) \prod_{\boldsymbol{x} \in X} p_\theta(\boldsymbol{x}|\boldsymbol{z}), \tag{5.12}$$

where $p(X = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\})$ is a distribution that fulfils $p(X = \{\boldsymbol{x}_{\pi(1)}, ..., \boldsymbol{x}_{\pi(n)}\}) = p(X = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\})$ for all permutations π.

To construct such a *shape* NN that takes a point cloud as input and encodes it to a lower dimensional $\boldsymbol{z}$, which is given as a condition to the NF[9].

However, while Finetti's theorem asserts that there exists a $\boldsymbol{z}$ for which Equation 5.12 holds, it does not make a statement of how to construct such a condition. Thus, it is left over to the optimisation algorithm to find such a condition. But it is unclear whether the shape NN can get a gradient from the loss that is optimised, since the loss is concerned with individual points only. This model is referred to as Point Flow (PF) in the following. A schematic of the model is given on the right of Fig. 5.5.

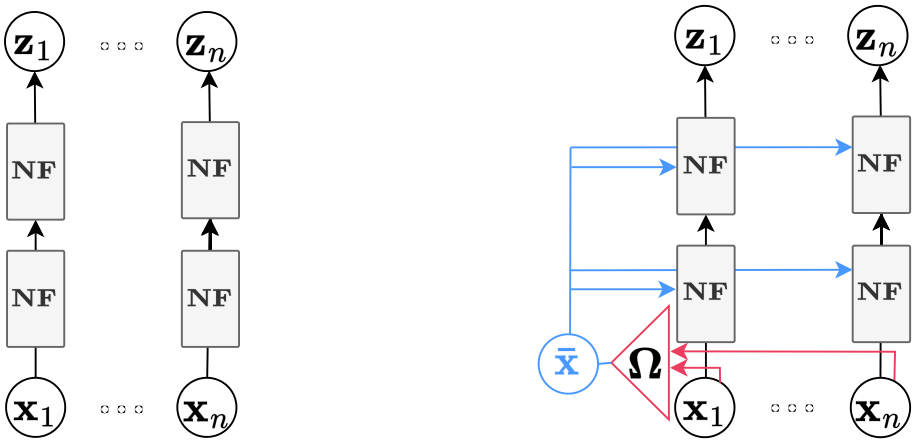
3. To resolve the problem that the shape NN does not get any feedback from the loss of the joint distribution of all points in the cloud, a natural next step is to introduce a mechanism to account for that. Since there is no analytical way to build a loss function for this, one option is to introduce an adversarial loss term. Hence, a discriminator that distinguishes PYTHIA samples from samples drawn from the model is introduced. Note that this discriminator is applied to the whole cloud. The objective of the shape NN is to maximise the loss of the discriminator. The only way the shape NN can fool the discriminator is if the provided condition renders the joint distribution conditionally independent. This approach is referred to as Adversarial Point Flow (APF) in the following[10].

Results

In Fig. 5.6, the marginal features and the mass for samples drawn from the IPF (red) are compared to PYTHIA-generated samples (blue). Note that the first and last

[9]The name "shape" derives from Ref. [176], where 3D point clouds with certain s shapes e.g. a chair, are generated. Note that the shape NN in this case has the same architecture as the discriminator of the TGAN, introduced in the following Section, with the only difference that its output is 10-dimensional.

[10]Note that the discriminator that was used has the same architecture as the discriminator of the TGAN, which is introduced in the Section 5.4.4.



(a) IPF (b) PF

Figure 5.5: Schematic of the permutation equivariant NFs, for illustrative purposes only two NF layers are used. (left) For the IPF, the only difference to the VNF is that the NF is applied to all particles independently. (right) For the PF, a permutation-invariant encoder $\mathbf{\Omega}$ (red) is additionally used. It encodes all points to a conditional variable $\bar{\boldsymbol{x}}$ (blue), which is used as a condition for all NFs.

bins show the under- and overflow. Although the marginal variables are modelled accurately, the mass distribution is slightly worse than for the VNF. It appears that conditioning with the shape NN used in the PF (yellow) slightly worsens the modelling of the marginal features, which becomes only visible in the ratio below the histograms. However, the mass distribution remains of the same Gaussian shape. This finding agrees with Refs. [177, 178], where the authors found that their model improves when the shape encoder is left away. Unfortunately, introducing an adversarial loss term to give the shape NN an incentive to encode the points such that Finetti's theorem is fulfilled, does not improve the modelling of the mass distribution either. The histograms of the APF (violet) reveal that the modelling of the marginal features got even worse.

But when considering the Pearson correlation matrices, shown in Fig. 5.7, even worse issues become apparent. For the IPF, the linear correlations between the individual particles are not modelled at all, as shown in Fig 5.7b[11]. This is problematic

[11]Note that the correlation structure visible in $p_{\mathrm{T}}^{\mathrm{rel}}$ comes from ordering of the particles in descending $p_{\mathrm{T}}^{\mathrm{rel}}$.

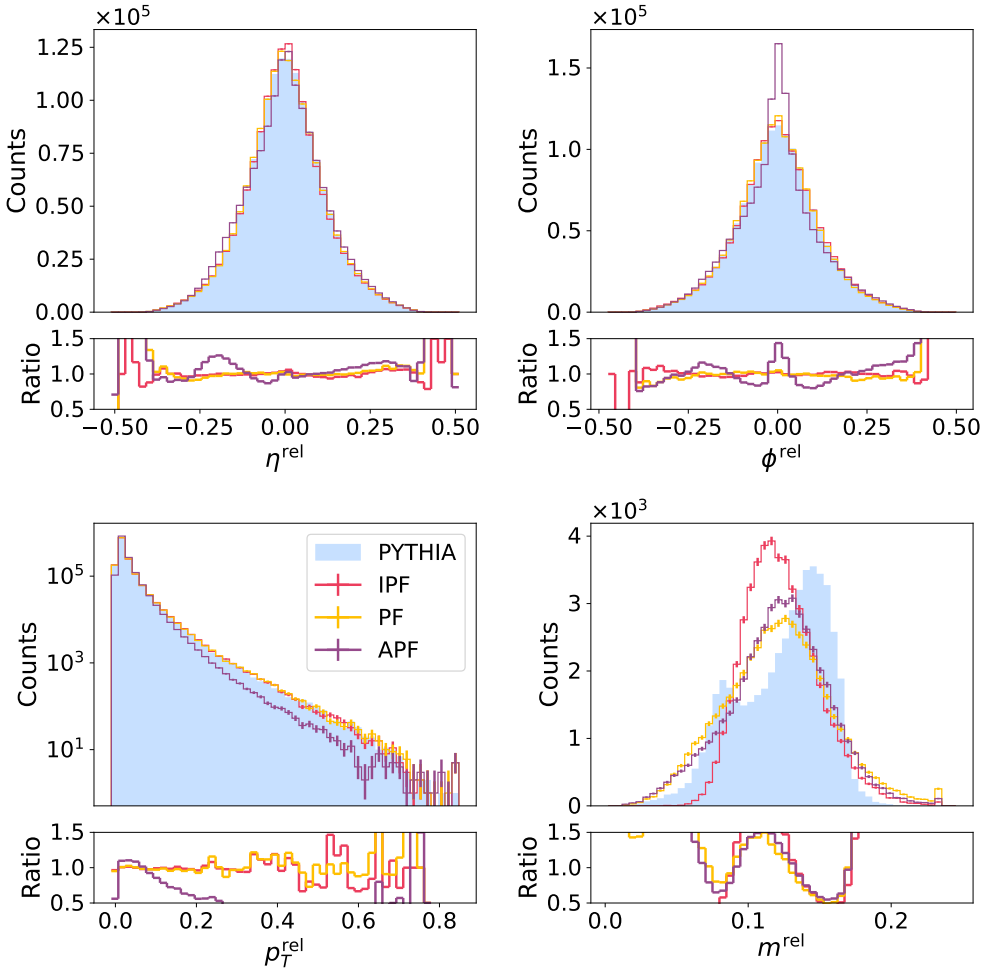
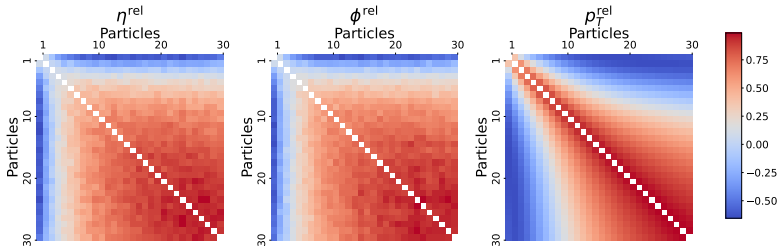
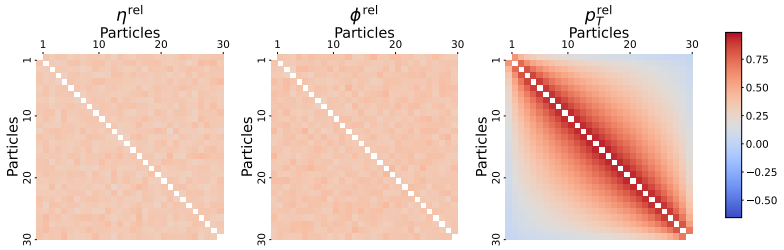


Figure 5.6: Comparison of synthetic data drawn from the permutation equivariant NFs to PYTHIA samples (blue). Below each plot, the yield ratio of PYTHIA data to synthetic data is shown. Going from left to right and top to bottom, the first three histograms depict the relative pseudorapidity, azimuthal angle and transverse momentum of all particles drawn from the model. The figure on the bottom right depicts the invariant mass distribution calculated from the particles in every jet. Similar to the VNF, the permutation equivariant NFs excel at modelling the marginal features. However, the mass distribution reveals that the correlations between the particles are not modelled correctly for neither the IPF (red), PF (yellow) nor APF (violet).

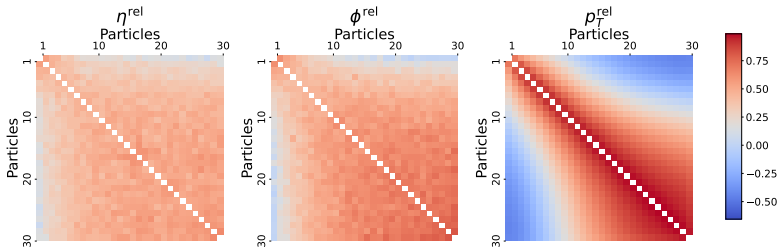
since most of the underlying physics is encoded in the correlations between particles. But nothing else is to be expected if the particles are drawn independently. For the PF, the linear correlations shown in Fig. 5.7c slightly improve, in the sense that the individual particles now exhibit a non-zero correlation. But, note that the correlations are still not comparable to the ones of the other NF models (VNF, NF(c), NF(cc)). For the APF, the linear correlations between the particles, shown in Fig. 5.7d, remain the same as for the PF. Note that for this evaluation, the latent distribution of the conditions used for the PF and APF was taken from the testing set, by encoding the jets in the testing set with the shape NN. In practice, the distribution of the encoded jets would need to be sampled as well, which is expected to worsen the performance even further.



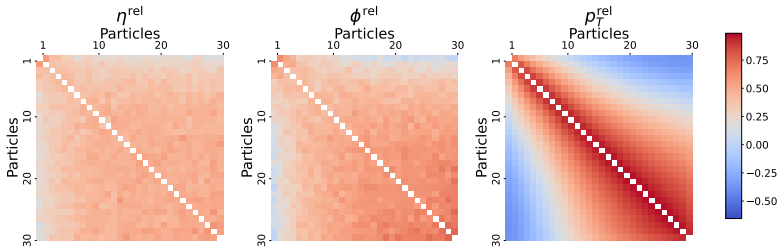
(a) PYTHIA



(b) IPF



(c) PF



(d) APF

Figure 5.7: Heatmaps of the linear correlations between the individual features of the p_T^{rel} ordered particles of the proposed permutation-equivariant NF-based models. As a reference, the correlations calculated from the PYTHIA data are visualised in the top row. The first two plots per row reveal the problem of sampling the points independently, as is done with the permutation equivariant NF models.

5.4.3 Discussion on Normalising Flows

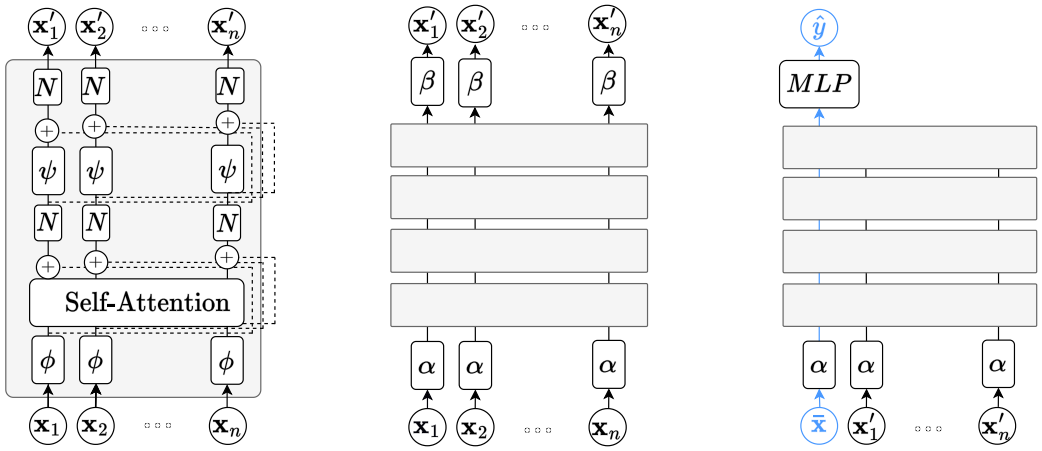
The results on the JetNet30 reveal that the RQS NFs are effective at modelling unknown densities, even in problems where they do not have an inductive bias tailored to the problem. However, they perform significantly worse than on the two-dimensional Two Moons dataset. This becomes evident when investigating the relative invariant jet mass distribution, which provides a sensitive candidate for measuring higher order and non-linear correlations between particles. It appears that the capability of NFs to model correlations quickly deteriorates, when the dimension of the data is increased, as discussed in Section 5.4.1. Conditioning the NF with the mass, and thus learning to model $p(\boldsymbol{x}|m)$, certainly improves the mass modelling. Especially if one makes use of bidirectional training. But the mass is a sensitive statistic that showed that the model is incapable of capturing the correlations entirely. By adding the mass as an input to the model, this test loses its sensitivity. It remains unclear, whether the conditioned model can model other correlations which are not as easily accessible as the mass. In Section 5.4.5, multiple metrics will reveal that this is indeed the case.
The results from Section 5.4.2 show that the three proposed ideas to make the NFs permutation-equivariant do not improve the performance, but rather significantly worsen it. The conditioning capability of NFs is further explored in Section 7.1.2 to investigate whether this conditioning can also be used to interpolate between distributions.

5.4.4 Flowing into a GAN

The VNF can capture most of the linear correlations between particles; however, it fails to model the high-level features accurately. To account for this, a post-processor NN is introduced, which modifies the output of the VNF such that these high-level correlations are modelled correctly. Since it is difficult to define a loss that enforces the modelling of those high-level features, the post-processor NN is trained adversarially with a discriminator.
When it came to designing the post-processing NN a transformer-based model was chosen because transformers were (and still are) the current SotA in NLP. Eventually, only the encoder part of a transformer is used, which was inspired by BERT [179],

a groundbreaking model from NLP[12]. The main block in the transformer encoder comprises the following sequential operations illustrated in Fig. 5.8[13]:

1. A particle-wise NN ϕ that maps the individual particles independently to a latent dimension to a higher-dimensional representation.
2. A self-attention layer that lets the individual representation of all particles interact with each other.
3. A residual connection, together with an independent normalisation layer (LayerNorm [180]) N for all particles.
4. Another particle-wise NN ψ.
5. A residual connection together with another normalisation layer N.



(a) Main Block (b) Post Processor (c) Critic

Figure 5.8: Schematics of the different parts of the TNF, a detailed description is given in Section 5.4.4. (left) Main information aggregation block of the post-processor NN. (middle) The post-processor NN consisting of 4 such blocks. (right) The critic also consists of 4 blocks. The classification token is marked in blue

[12]Note that although this has the same naming as the encoder mentioned in Section 4.2, this encoder does not encode the data to a lower dimension usually.

[13]As aforementioned in Section 3.3.4, the common positional encoding is left out to make the aggregation equivariant to permutations.

This block accepts a variable-sized input and produces an output with the same size. This means if n particles, or in NLP jargon n *tokens*, go into the block, its output also consists of n updated particles.

The output of the NF $[\boldsymbol{x}_1, ..., \boldsymbol{x}_m]$ is fed to the post-processor NN, which is depicted in Fig. 5.8. There are two additional layers:

1. The layer α maps every point from its three-dimensional representation to a higher-dimensional latent representation.

2. Vice versa, the layer β maps the point from the latent dimension down to the three-dimensional representation.

The critic, which is used to train the post-processor NN, has a nearly identical architecture. It also accepts a variable-sized input and is based on the same building blocks. However, since only a one-dimensional output is desired, a classification token/particle $\bar{\boldsymbol{x}}$ is introduced. The value of this token is learnable and treated the same as all other particles in the main blocks. After the main blocks, this artificial token is fed into a two-layered MLP that maps it to a one-dimensional output $\hat{y}$.
The self-attention aggregation is the core of this model that lets the representations of the different particles interact with each other. All other components in the model are applied to the particles independently.
Attention allows to adaptively determine the influence of the particles on each other during the information exchange. This is beneficial for jets, as it is expected that the harder particles carry more information. A more technical detail is how transformers allow a variable-sized input during batch-based training. This is done by padding missing particles, and excluding the influence in the information aggregation step as explained in Section 3.3.5. In the following, this model is referred to as the Transformer Normalising Flow (TNF).

Omitting the Normalising Flow

The NF that is used as a prior can be left out without worsening the performance of the model. This makes the training slightly more unstable, but the model becomes significantly smaller and faster. The resulting model is referred to as TGAN.

Training

The trained VNF from Section 5.4.1 is used to provide the prior for the post-processor NN. The weights of the VNF remain frozen, meaning they remain unchanged during the training of the post-processor NN. The LSGAN training objective performed best to train the post-processor NN. The AdamW optimiser is used, but if the momentum coefficient β_1 of AdamW is chosen to be greater than 0 for either post-processor NN or critic, the training does not converge.
The hyperparameters used for the training are given in Table 5.2. The `_gen` postfix denotes hyperparameters of the generator part of the GAN; otherwise, they correspond to the critic. The `heads` parameter denotes the number of heads used in the multi-headed attention, the `hidden` parameter denotes the dimension over which the attention aggregation takes place. This also determines the dimension of the output of the single layer ϕ, respectively input-dimension of the single layer ψ. The parameter `l_dim` stands for the latent dimension to which the input features are mapped by the embedding NN $\boldsymbol{\alpha}$. The `num_blocks` parameter determines how many blocks from the left of Fig. 5.8 are used in the model.

Parameter	Value
heads_gen	16
heads	16
hidden_gen	256
hidden	256
l_dim_gen	16
l_dim	16
lr	0.0001
num_blocks	4
num_blocks_gen	4
opt	AdamW
weightdecay	0.01
beta1	0
beta2	0.999

Table 5.2: Hyperparameters for the GAN in the TNF

Results

Both GAN-based approaches model the tail of the $p_{\mathrm{T}}^{\mathrm{rel}}$ distribution, even when no Box-Cox preprocessing is used. The results with Box-Cox preprocessing are given in Appendix B, which are slightly worse on most of the metrics discussed in Sec. 4.8. In Fig. 5.10 the histograms of the marginal features and the relative mass are shown for the TNF (red) and the TGAN (yellow). Both models are capable of generating samples that model the mass distribution as in data, without relying on the mass as a condition.

However, the linear correlations shown in Fig. 5.10b still exhibit some differences. The TNF appears to strongly correlate the features between neighbouring particles, which manifest as the darker shades around the diagonal.

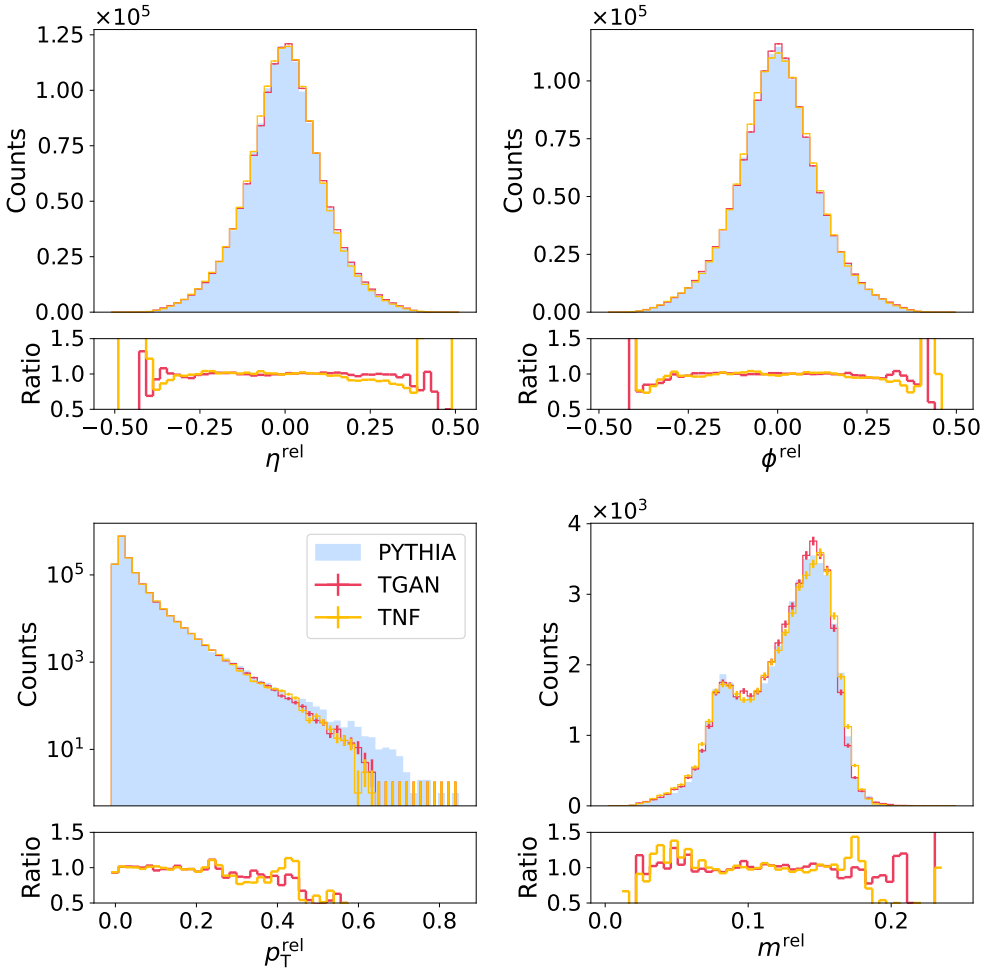
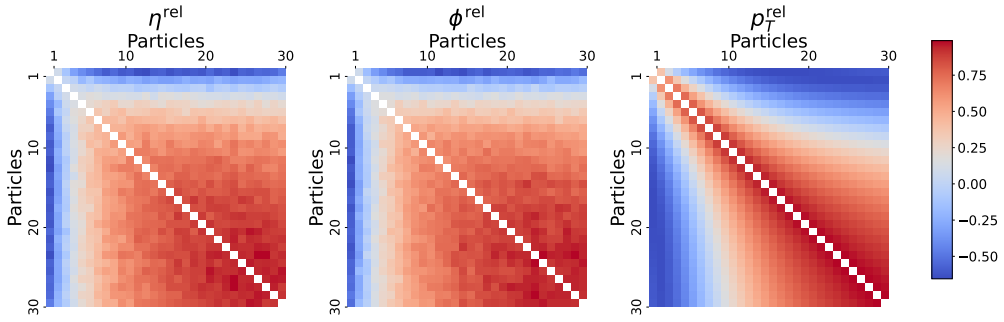
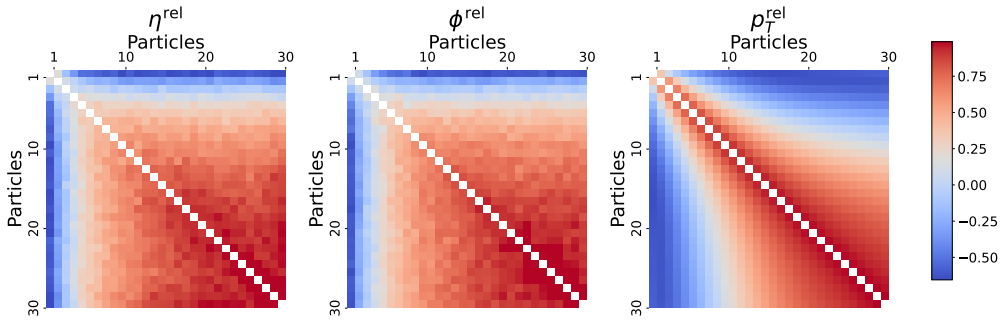


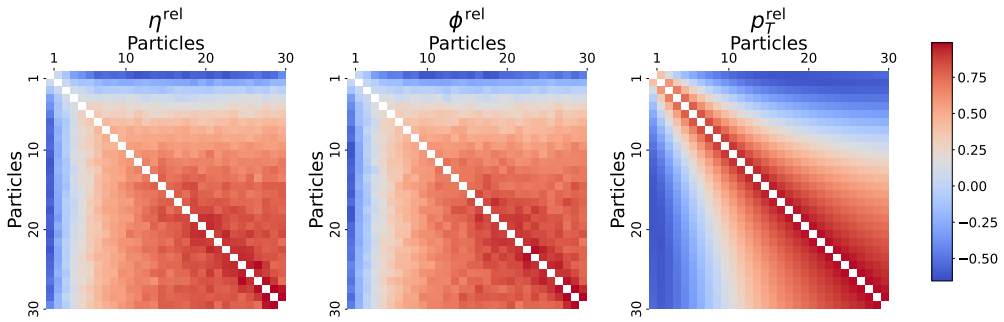
Figure 5.9: Comparison of synthetic data drawn from the GAN-based models to PYTHIA samples (blue). Below each plot, the yield ratio of PYTHIA data to synthetic data is shown. Going from left to right and top to bottom, the first three histograms depict the relative pseudorapidity, azimuthal angle and transverse momentum of all particles drawn from the model. The figure on the bottom right depicts the invariant mass distribution calculated from the particles in every jet. With the TNF (red) and TGAN (yellow), it is possible to model mass distribution accurately without supplying the mass as a condition.



(a) PYTHIA



(b) TNF



(c) TGAN

Figure 5.10: Heatmaps of the linear correlations between the individual features of the p_T^{rel} ordered particles of the GAN-based models. As a reference, the correlations calculated from the PYTHIA data are visualised in the top row.

5.4.5 Quantitative Comparison: JetNet30

To not only compare plots, the metrics, which were discussed in Sec. 4.8, are computed between samples from each model and an independent set of $50'000$ samples. The result is given in Table 5.3, where the value of the best-performing model per score and the scores of models which lie within uncertainties of the best-performing model are highlighted in bold font. As a reference, the in-sample distance and the performance of the, at the time of the TNF study, SotA model MPGAN [156] are shown additionally[14].

The permutation equivariant NF models (IPF, PF, APF) are performing significantly worse than the other models.

The NF models that remove the particle structure by flattening the input perform worse than the GAN-based models, which are permutation equivariant. When conditioned on the mass, the W_1^M decreases significantly, but the W_1^{EFP}, KPD & FPD metrics still signify a worse performance than GAN-based models. This supports the claim that conditioning is more of a cure for a symptom, rather than an actual solution to the problem.

Overall, the permutation equivariant GAN-based models (TNF, TGAN) perform best. The purely GAN-based model performs best on most metrics and is also the winner in terms of generation speed and model size.

[14] To calculate the distances for the MPGAN, the checkpoint provided in their repository was used to generate a new dataset.

Table 5.3: Comparison of the proposed models on the top quark JetNet30 dataset. Bold font is used to highlight the best-performing model. Multiple models are marked at the same time if the metrics are compatible within one standard deviation of the best performing model. The in-sample distance between the training and testing sample (IN) is also given for reference, together with the distances calculated from the state-of-the-art MPGAN [156]. Additionally, the number of parameters and the time needed to generate one jet is given.

Jet Class	Model	$W_1^M(\times 10^3)$	$W_1^P(\times 10^3)$	$W_1^{EFP}(\times 10^5)$	$KPD(\times 10^4)$	$FPD(\times 10^4)$	Time [μs]	#parameters
	PF	10.8 ± 0.2	1.95 ± 0.02	18.9 ± 0.1	32 ± 7	1990 ± 70	5.0	2.9 M
	IPF	9.7 ± 0.2	1.50 ± 0.03	18.98 ± 0.07	170 ± 20	3990 ± 60	4.2	2.8M
	APF	12.6 ± 0.2	3.09 ± 0.04	43.1 ± 0.3	5k $\pm$1k	24k $\pm$2k	5.0	3.1M
	NF	5.37 ± 0.04	2.08 ± 0.03	11.01 ± 0.07	2.9 ± 0.3	121 ± 3	3.8	6.9M
Top Quark	NF(c)	3.6 ± 0.2	1.87 ± 0.03	5.4 ± 0.1	2.6 ± 0.2	29.8 ± 0.7	4.0	6.9M
	NF(cc)	2.01 ± 0.06	1.82 ± 0.03	5.04 ± 0.07	4.3 ± 0.6	54.9 ± 0.8	4.0	6.9M
	TNF	0.48 ± 0.06	0.60 ± 0.03	0.95 ± 0.03	$\mathbf{-0.0 \pm 0.1}$	$\mathbf{1.9 \pm 0.2}$	3.8	7.1M
	TGAN	$\mathbf{0.42 \pm 0.03}$	$\mathbf{0.54 \pm 0.03}$	$\mathbf{0.87 \pm 0.06}$	$\mathbf{-0.01 \pm 0.07}$	2.2 ± 0.4	**0.1**	**80k**
	MPGAN	0.5 ± 0.1	0.97 ± 0.03	$\mathbf{0.9 \pm 0.2}$	$\mathbf{0.07 \pm 0.08}$	17.4 ± 0.8	35.7	716k
	IN	0.31 ± 0.07	0.20 ± 0.03	0.47 ± 0.09	-0.13 ± 0.06	0.7 ± 0.3	-	-

Multiclass Classifier Test

The multiclass classifier, which was discussed in Section 4.8.2, used for this has mostly the same architecture as the classifier used for the training of the TNF and TGAN. The only difference lies in the output layer, where the multiclass classifier has eight features, one for each model. In Fig. 5.11 the results of the multiclass classifier test are depicted. The confusion matrix on the left shows that the classifier can recognise the different model classes, but misidentifies some models within the individual model class. The evaluation of the classifier on PYTHIA data align with the claim that the GAN-based models perform best.

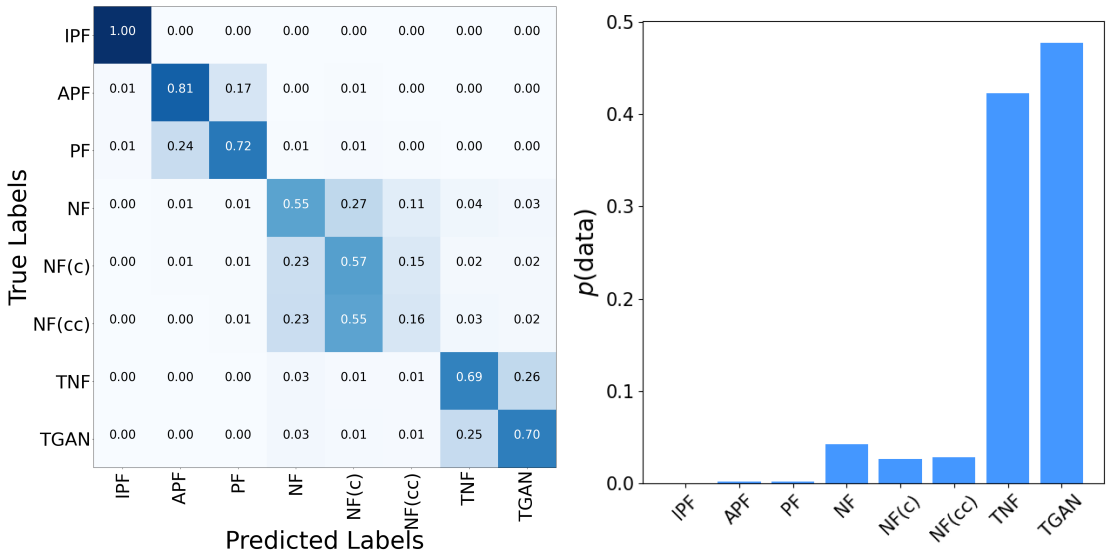


Figure 5.11: Evaluation of the multiclass classifier test on the JetNet30 datasets. (left) The confusion matrix shows that the classifier can recognise the different model classes. There are some uncertainties within the GAN-based, NF-based and conditioned permutation equivariant NF models. (right) Since the classifier mostly assigns the TGAN and TNF labels to PYTHIA data, this test supports the claim that the TNF and TGAN perform best.

5.5 Studies on JetNet150

After obtaining a model that is competitive on `JetNet30`, the logical next step was to scale it up to 150 particles. At the same conference where I presented the TNF model, promising results of the Equivariant Point Cloud (EPiC-GAN) by Buhmann et al. were first shown on the `JetNet150` gluon dataset. The authors later became the first to publish SotA results on the full gluon-, light- and top quark datasets [96]. Their model uses the DeepSets aggregation, along with a global z_{global} and local jet state z_{local}. The local states are representations of the individual particles and update the global state via sum- and mean-pooling. The global state of the jet updates the individual local states independently. This is an efficient information aggregation mechanism, as its computational complexity scales linearly with the number of particles, in difference to the self-attention used in the TNF and TGAN.

5.5.1 Curse of $\mathcal{O}(n^2)$

As described in Section 3.3, the attention weights are calculated by multiplying the query matrix $Q \in \mathbb{R}^{n \times d}$ with the transposed key matrix $K^T \in \mathbb{R}^{d \times n}$, where n is the number of points in the cloud and d the dimension of the latent representation. The computational complexity of this is $\mathcal{O}(n^2 \cdot d)$. This is because in self-attention, the interaction of every input with every other input is considered, as illustrated in Fig. 5.12. This scaling not only affects the generation speed of the model but is primarily restrictive due to its memory use, as discussed in Section 3.3.7. When trying to scale up the TNF and TGAN to the higher cloud cardinality, the large memory usage makes the training infeasible, since the batch size has to be reduced to $\mathcal{O}(1)$. Therefore, self-attention had to be replaced with an aggregation that scales linearly with the number of constituents.

5.5.2 Mean-field Approximation of Particle Interactions

When considering the constraints of matrix multiplication and the row-wise application of the softmax, it can be concluded that for a linearly scaling attention-based

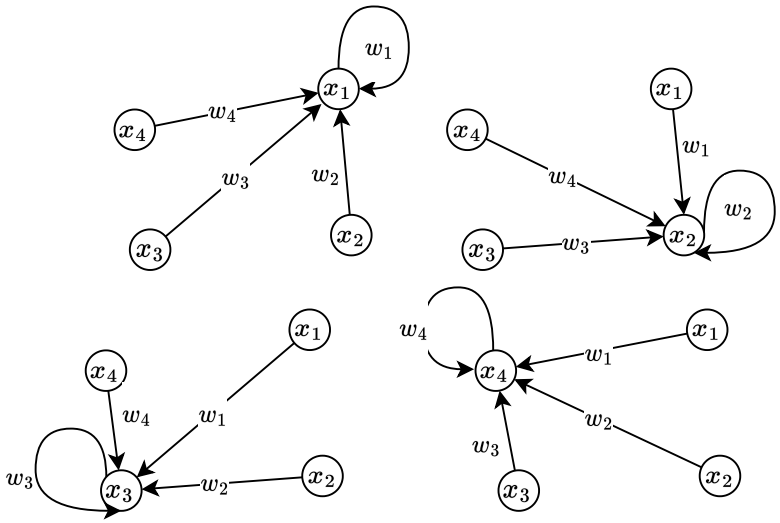


Figure 5.12: Illustration of the quadratic computational complexity of self-attention on Point Clouds. Every point $\boldsymbol{x}_i$ is updated with a dynamic contribution of all points with a weight w_i. Note that all four updates happen simultaneously during one self-attention aggregation step.

aggregation, the query Q should consist of a single token only[15]. Because when choosing the key K to have a shape of $(1, d)$, where d is the number of features of every embedded point, the multiplication of QK^T results in an output of shape $n \times 1$, where n is the number of particles. If the softmax is then applied row-wise, every row becomes equal to one. This leads to an over-engineered method to construct an identity mapping using cross-attention.

However, using only one token for the query means that the output of the aggregation also contains only one token. Therefore, cross-attention cannot directly be used to replace self-attention. Instead, a similar approach to EPiC-GAN is employed to transform the individual particles. A conditioned, particle-wise NN is applied to all particles. Its condition is the output of the cross-attention aggregation.

This approach also has an illustrative interpretation when recognising attention as a weighted sum, as described in Section 3.3. Instead of every particle interacting with every other particle, the interaction is mediated through a mean field. In the first step, the particles interact with the mean-field and update it, where the weight of

[15]Note that this can be considered as cross-attention to a single token only, and is not directly related to self-attention any more.

each contribution of the particles is determined dynamically. In the second step, the mean-field then updates the particles independently. This mechanism is illustrated in Fig. 5.13 and allows for a more complex information aggregation, compared to an arithmetic mean or sum-pooling. After the mean-field has been updated, it interacts

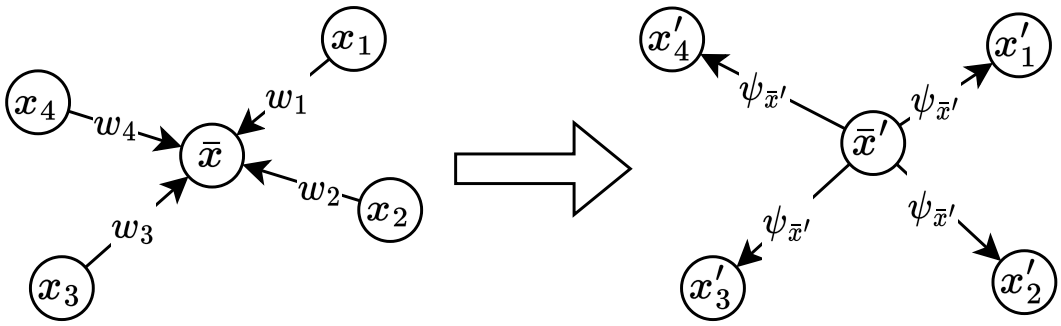


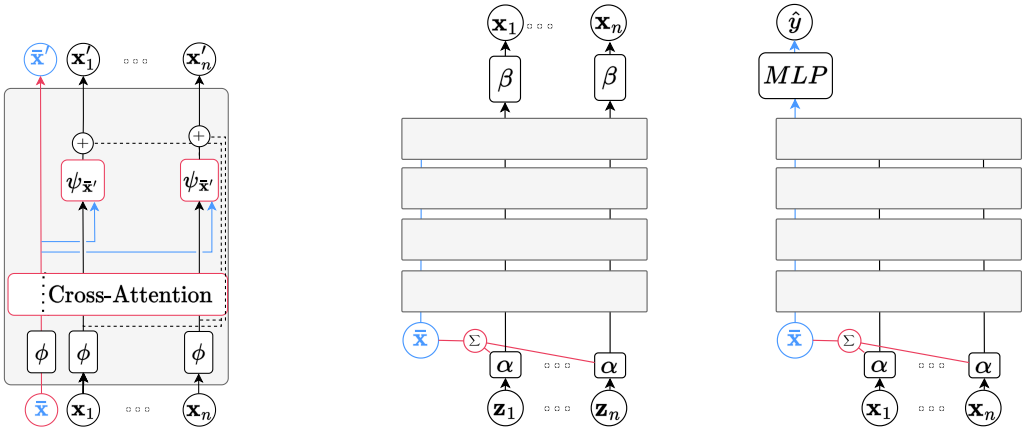
Figure 5.13: Schematic of the mean-field aggregation; the left side illustrates the cross-attention aggregation, where each particle contributes to the mean-field with a dynamic weight w_i. On the right side, the interaction of the updated mean-field with the particles is depicted, which is mediated by the same NN $\psi_{\bar{x}'}$ for every particle.

with every particle independently. A schematic of the linearly scaling model is given in Fig. 5.14, where on the left the main block of the architecture is depicted. The differences to the transformer encoder architecture, which was used before, are highlighted in red. The most significant modification is in the main building blocks of the architecture, where self-attention is replaced with the mean-field aggregation.

5.5.3 The Missing Piece

Although the model now scaled linearly with the number of particles, it did not produce competitive results on `JetNet150`. This was puzzling since it worked well on 30 particles[16]. It was puzzling, why EPiC-GAN can perform so well, although it uses a much simpler aggregation. What especially stood out is that mean- and sum-pooling are redundant as they are closely related, the only additional information is given by the number of constituents. But similarly to an arithmetic mean, the mean-field aggregation is agnostic to the number of constituents. Since the mean-field

[16]It even came so far that I scanned the number of jet constituents that are modelled, to see at which number of particles the model breaks down, which was around 50.



(a) Main Block (b) Generator (c) Critic

Figure 5.14: Schematic of the architecture making use of the mean-field aggregation. Parts of the architecture that were modified compared to previously used self-attention-based architecture are marked in red. Parts associated with the mean-field are marked in blue. (left) The main building block of the model where the mean-field is updated with cross-attention with respect to the particles. The mean-field then updates the particles independently with $\psi_{\bar{\mathbf{x}}'}$. There is a residual connection between the particles going into cross-attention and the updated particles, which is depicted by a dashed line. The output of the block is the updated mean-field and the updated particles. (middle) Schematic of the generator; standard Gaussian noise with three features for every point is mapped to a higher dimensional representation by $\boldsymbol{\alpha}$. Thereafter, the mean-field is initialised by sum pooling these representations. After multiple mean-field aggregation blocks, the particles are mapped independently to three dimensions with a fully connected layer $\boldsymbol{\beta}$. (right) Schematic of the Critic, the only difference to generator architecture is that there is a 2-layer MLP after the main body that takes as input the final mean-field $\bar{\boldsymbol{x}}$ and outputs a score s.

aggregation is the only one that allows the exchange of information between particles, the whole model was agnostic to the number of particles. The reason for this lies in the softmax normalisation of the attention weights. Introducing this simple piece of information into the model allowed to make it competitive with the SotA on the `JetNet150` datasets.

5.5.4 Mean-field Matching

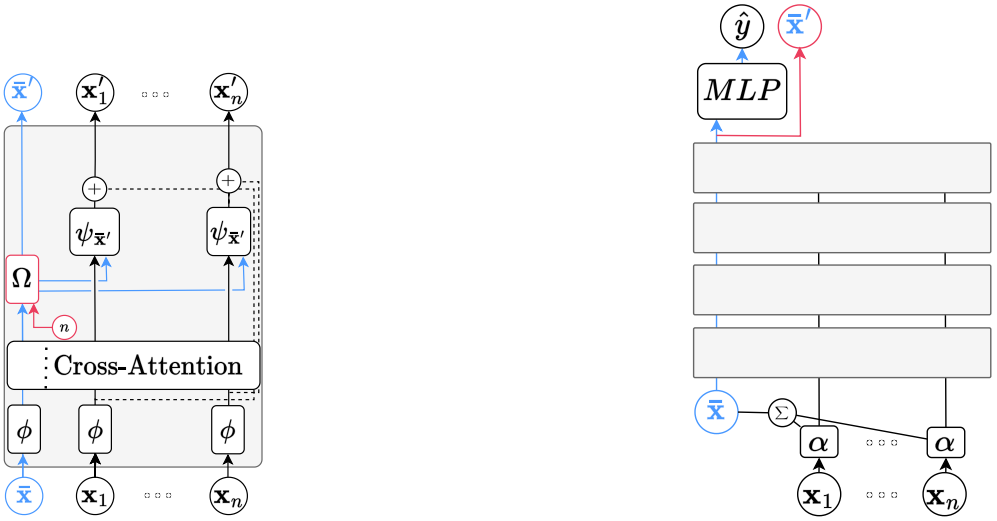
Since switching away from GANs made the training significantly more unstable, new ways to stabilise the training were needed. As mentioned in Section 4.3.2, one that stood out was the feature matching. In the case of the mean-field, this is also quite illustrative. The mean-field of a batch should be equal for PYTHIA and data drawn from the model. Hence, another loss term is introduced:

$$\mathcal{L}_{MF} = \left| \sum_{\text{batch}} (\bar{\boldsymbol{x}}_{\text{real}}) - \sum_{\text{batch}} (\bar{\boldsymbol{x}}_{\text{fake}}) \right|^2 , \tag{5.13}$$

where $\bar{x}$ is the mean-field after the main body in the critic and the index indicates whether the mean-field comes from PYTHIA (real) or the generative model (fake). Additionally, the weights of the critic are normalised, as proposed by Xiang et al. [125].

5.5.5 Matching Deep Mean-fields Attentive GAN

Implementing all these modifications, the results in a new model, referred to as Matching Deep Mean-fields Attentive (MDMA)-GAN [163]. The schematic of the final changes in architecture is shown in Fig. 5.15. While the macroscopic generator architecture remains the same, the main blocks are modified by introducing a fully connected layer Ω, which is conditioned on the number of particles. The critic is also modified to give the mean-field after the main blocks, hence the name Deep Mean-Field, as output.



(a) Main Block (b) Critic

Figure 5.15: Schematic describing the novel parts in the MDMA-GAN. Modifications to the previous model are marked in red. Parts associated with the mean-field are marked in blue. (left) Main building block of the model. Parts associated with the mean-field are marked in blue. Compared to the architecture before, the mean-field is now updated after the cross-attention layer with a fully connected layer which takes the number of points in the cloud as an additional input. (right) Compared to before, the critic also outputs the mean-field after the last block in addition to the score.

Differences to EPiC Aggregation

Since this setup can seem very similar to the predecessor EPiC-GAN the major differences are highlighted in the following:

- The MDMA generator does not use additional degrees of freedom for a global state. Furthermore, the mean-field is transformed by the same particle-wise NN as the other particles.
- Attention allows masking of particles, which makes the batch generation of different sized clouds possible.
- The information aggregation with attention is a generalised version of sum-pooling, where different contributions are weighted adaptively.
- The normalisation of the attention weights is beneficial when moving to problems, where the number of constituents distribution has a high variance.

5.5.6 Efficient Data Loading with Variable-sized Data

The variance in the number of jet constituents becomes relevant on the `JetNet150` dataset, since padding all jets to maximum size per batch becomes inefficient. This is illustrated in Fig. 5.1, which shows the distribution of the number of jet constituents. Since the mean of the distribution is around 60 particles, if the clouds are always zero-padded to 150 particles, the required memory increases by 250%. Note that this is especially expensive if quadratic scaling is present, since although the particles do not influence the result of the calculation, they still lead to significantly higher memory use. To resolve this, bucketing is introduced, where during training the batches contain similarly sized point clouds. This reduces padding and hence accelerates the training process.

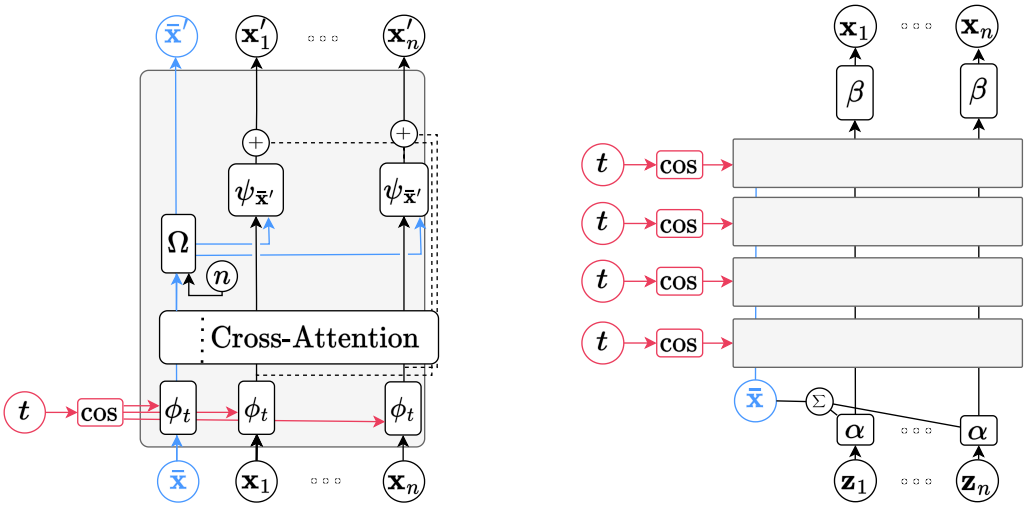
5.5.7 Matching Flows on MDMA

GANs were the only models that achieved competitive performance on the JetNet datasets for a long time. Although diffusion models were an upcoming model class, the early publications [181, 172] with diffusion models all relied on jet variables as

conditions. But the issues of conditioning as discussed in Section 5.1 remained, and I was convinced that diffusion models carry the same problems as NFs. However, Buhmann et al. [171] showed that CNFs trained with Flow Matching, which can be considered a more general version of diffusion models, also lead to promising results. The authors show that the main building block of the EPiC-GAN performs well when used in a CNF. Since the MDMA block can be considered as a more general version of the EPiC block, it is worth investigating how the model performs with the MDMA block. The schematic of the model adapted for the CNF training is shown in Fig. 5.16, where the changes compared to the MDMA-block are highlighted in red. Note that the cosine encoding for the time is computed as:

$$t_{\text{emb,i}} = \cos\left(t \cdot f_i \cdot \pi\right), \tag{5.14}$$

where the frequency $f_i \in [0, d]$, with d being the dimension of the latent space.



(a) Main Block (b) CNF

Figure 5.16: Schematic describing the CNF, new components are marked in red. Components associated with the mean-field are marked in blue. (left) The only change in the main block of the CNF is that it now additionally takes the time, which passes through a cosine embedding, as input. (right) For illustrative purposes, only four blocks are drawn in the CNF.

In the following, this model is referred to as MDMA-Flow and its results are

presented and compared to results of the MDMA-GAN. The training of the EPiC-FM and MDMA-Flow slightly differ. Buhmann et al. use the Flow Matching proposed by Lipman et al. [147]. For the MDMA-Flow the CFM training, as described in Section 4.7, is used. The training with OT-CFM is significantly slower, and does not lead to improved results on this dataset. The results from the CNF trained with OT-CFM are given in the Appendix B.6.

Training on `JetNet150`

The hyperparameters used to train the MDMA-GAN and MDMA-Flow are given in Table 5.4. The MDMA-GAN is trained with the LSGAN loss. Similarly to the GAN-based models before, the MDMA-GAN and MDMA-Flow perform better when no Box-Cox scaling is used to preprocess the training data. The results for Box-Cox preprocessing are given in the Appendix B.5.

Table 5.4: Hyperparameters of the MDMA-GAN

Parameter	**MDMA-GAN**	**MDMA-Flow**
`heads_gen`	16	-
`heads`	16	16
`hidden_gen`	48	-
`hidden`	64	256
`l_dim_gen`	16	-
`l_dim`	16	128
`lr`	0.0001	0.0001
`num_blocks`	2	8
`num_blocks_gen`	7	-
`opt`	AdamW	AdamW
`weightdecay`	0.01	0.01
`beta1`	0	0.9
`beta2`	0.999	0.999

Results

For both models, the marginal distributions are modelled accurately. Some discrepancies are visible in the tails of the distributions, where little data is available. The two-peaked mass distribution is modelled, although there are minor discrepancies

visible around the peaks of the mass distribution. To solve the ODE to generate data with the MDMA-Flow, the midpoint solver is used with 200 iterations. In Fig. 5.17 the histograms of the marginal features and the mass distribution are shown.

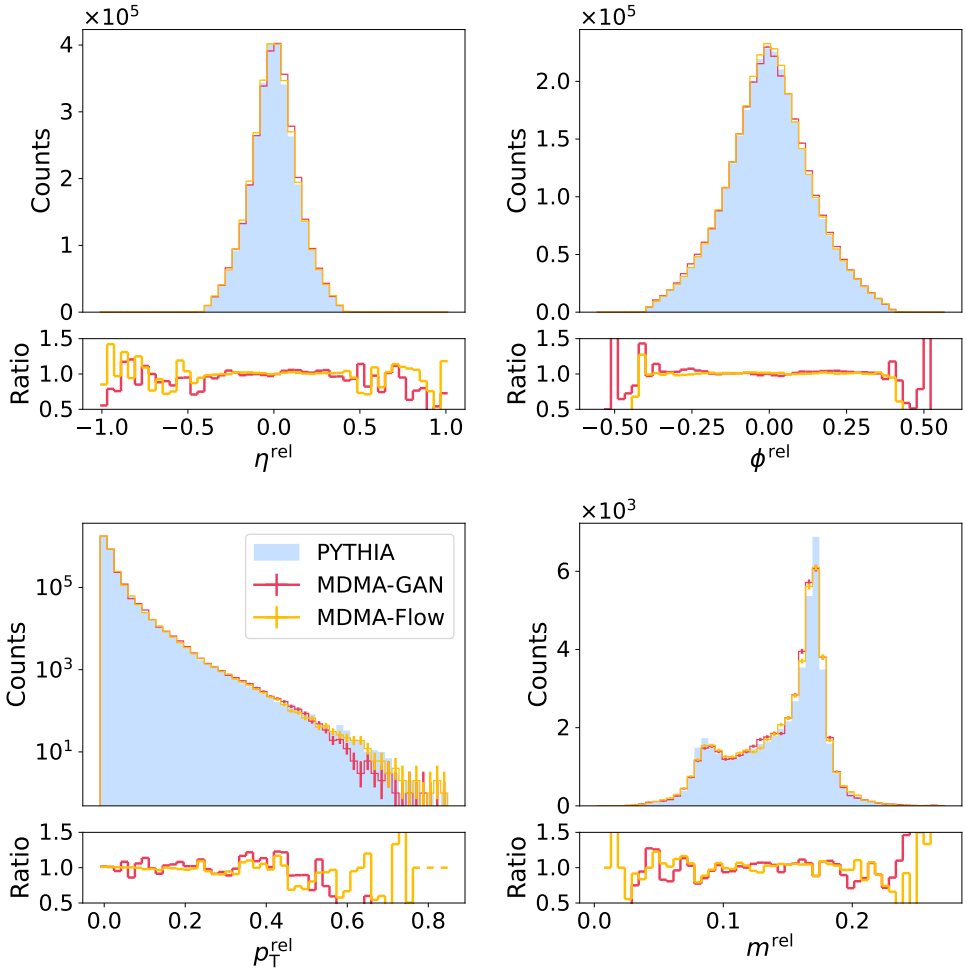


Figure 5.17: Comparison of synthetic data drawn from the MDMA-GAN (red) and MDMA-Flow(yellow to PYTHIA samples (blue). Below each plot, the yield ratio of PYTHIA data to synthetic data is shown. Going from left to right and top to bottom, the first three histograms depict the relative pseudorapidity, azimuthal angle and transverse momentum of all particles drawn from the model. The figure on the bottom right depicts the invariant mass distribution calculated from the particles in every jet. From the figure by itself, it is difficult to judge which model performs better.

5.5.8 Quantitative Comparison: JetNet150

The quantitive performance of the models is compared in Table 5.5. Additionally, the number of parameters of the generative model is given, together with the time that is needed for the generation of a single jet. The scores of the best-performing model and models with scores that lie within one standard deviation of the best model are marked in bold font. Note that for the KPD metric, all models which are compatible with zero have been marked in bold font. As a reference, the metrics computed on samples from the EPiC-GAN and the unconditional EPiC-FM are also listed[17]. These metrics strongly suggest that the cross-attention-based MDMA aggregation outperforms the DeepSets-based EPiC aggregation, since on all scores the MDMA-based models are compatible or better. While the MDMA-GAN is the winner in terms of model size and generation speed, it cannot keep up with the performance of the MDMA-Flow. Still, when compared to the EPiC-GAN, it performs better on every single metric. Since there is a factor of ~ 30 between the number of parameters, one might be tempted to conclude that the difference in performance is due to this. However, increasing the number of parameters of the GAN destabilised the training to the extent that no convergence was reached any more. The only drawback of the MDMA-Flow is the inference speed of the model. But there are methods, like knowledge distillation [182] to significantly increase the speed of Flow Matching based models. Concerning the high number of parameters of the MDMA-Flow, one could argue that due to the Double-Descent phenomena [75] the increased number of parameters makes the model generalise better. In HEP, it is not necessary to increase the generation speed by a factor of six orders of magnitude, already one or two orders of magnitude are sufficient. Especially, when considering that this is still a factor $\times 10$ faster than PYTHIA, as demonstrated in Ref. [172], for the generation ($46'200\ \mu s$).

[17] New samples were drawn from the model with the provided checkpoints in the repository corresponding to the publication.

Table 5.5: Quantitative results on the `JetNet150` top-quark dataset. The discussed models are compared to the SotA unconditional generative models (EPiC-GAN and EPiC-FM). The MDMA-Flow is outperforming, or at least compatible, with all other models on all considered metrics. Although it has significantly more parameters than the MDMA-GAN, this is not a real drawback. The only significant drawback comes in the inference speed of the model, which is around 1000× slower than for the MDMA-GAN.

Jet Class	Model	$W_1^M(\times 10^3)$	$W_1^P(\times 10^3)$	$W_1^{EFP}(\times 10^5)$	$KPD(\times 10^4)$	$FPD(\times 10^4)$	Time [μs]	#parameters
Top Quark	MDMA-GAN	$\mathbf{0.65 \pm 0.08}$	0.126 ± 0.003	2.7 ± 0.1	$\mathbf{0.1 \pm 0.2}$	1.9 ± 0.4	**7.1**	**111443**
	MDMA-Flow	$\mathbf{0.6 \pm 0.1}$	$\mathbf{0.045 \pm 0.006}$	$\mathbf{1.93 \pm 0.06}$	$\mathbf{-0.1 \pm 0.3}$	$\mathbf{1.4 \pm 0.3}$	6880.4	3119491
	EPiC-GAN	0.8 ± 0.1	0.67 ± 0.01	3.3 ± 0.1	2.8 ± 0.7	23 ± 1	62.5	424850
	EPiC-FM	1.3 ± 0.1	$\mathbf{0.05} \pm 0.01$	3.9 ± 0.1	$\mathbf{-0.1 \pm 0.1}$	1.9 ± 0.2	6075.0	561330
	IN	0.5 ± 0.1	0.08 ± 0.02	1.2 ± 0.1	-0.1 ± 0.2	0.8 ± 0.3	$46'200$	-

5.6 Conclusions from Modelling Jets

In this chapter, the NN-based generation of jets was extensively discussed with the JetNet datasets. For all models that are compared, there are still significant differences between the PYTHIA samples and the generated data. It is not certain how severe the impact of these differences is when a model would be employed in an actual analysis, as usually corrections need to be applied to the generated samples anyway. However, in practice, there is little interest in replacing PYTHIA showering with a generative model, since PYTHIA is not the bottleneck when it comes to the computational budget for MC data generation. Still, it is worth exploring and comparing different generative models on this dataset, as it is an open dataset that allows the benchmarking of different models. It also remains unclear whether the 100'000 training samples are too few to train a generative model. More data may reduce the remaining discrepancies between PYTHIA-generated data and the samples drawn from the model. The main conclusions are:

1. While normalising flows are not only stable to train, they also show very promising results on the marginal distributions. However, for high-dimensional data $d \gtrsim 20$, their performance deteriorates, especially on high-level summary statistics like the invariant jet mass.

2. GANs, although being unstable to train, perform significantly better than the discrete NFs on all considered metrics if they converge.

3. Continuous normalising flows are a promising candidate that brings the best of both worlds. While they are not only stable during training, they also model high-level correlations accurately. Their only drawback is the inference speed, since the ODE that is used to generate data is solver iteratively.

4. When modelling a point cloud, it is crucial that the information aggregation mechanism scales linearly with the number of points.

CHAPTER 6

Generative Modelling on Point Clouds II: CaloChallenge

After achieving performance competitive with the SotA on the `JetNet150` datasets, the model demonstrated its versatility on another point cloud generation task, i.e. the generation of energy deposits from a high-granularity calorimeter [159] for the CaloChallenge [183]. This is a good test the limits of the mean-field aggregation, as the dimension of the data increases by over 150× compared to `JetNet150`[1].

6.1 Dataset

The data comprises calorimeter images, which can be thought of as three-dimensional images. Corresponding to the pixel in an image, there are *voxels* in calorimeter images, yielding an energy deposit. For the datasets in the CaloChallenge, the energy deposits left by electromagnetic showers of electrons and pions are simulated in a toy detector. The more general layout of the toy detector is shown in Fig. 6.1. The CaloChallenge contains three different datasets with increasing granularity, corresponding to a difficulty ranging from low to high:

1. Dataset 1 [184] contains charged pion- and photon showers and is based on the ATLAS GEANT4 open datasets [185], previously used for fast-simulation

[1]There are up to 20000 hits on dataset 3 of the CaloChallenge, and every hit now has 4 coordinates.

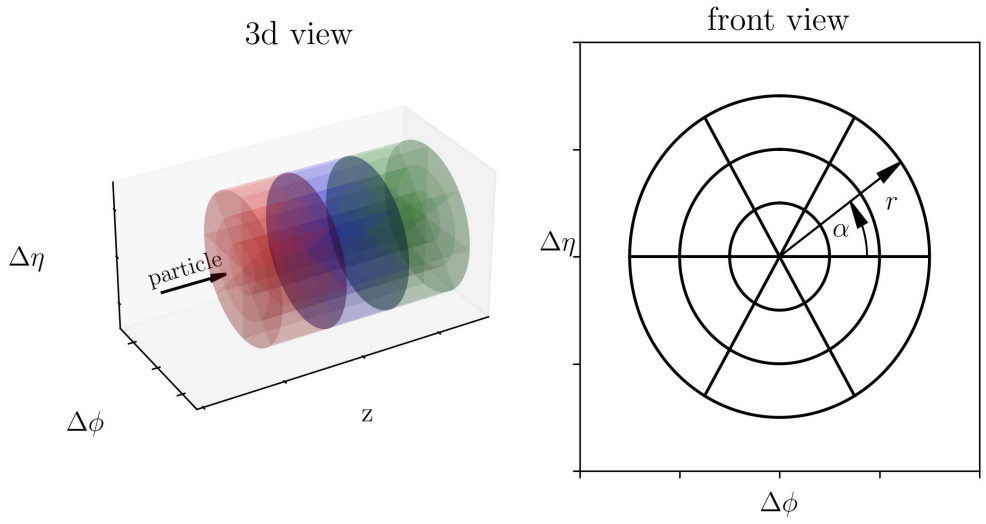


Figure 6.1: Illustration of the toy detector from the CaloChallenge and its coordinate system, taken from Ref. [183]. The detector consists of concentric cylinders. The particles propagate along the $z-$axis. The three datasets differ in the granularity of the detector, which corresponds to a difficulty level from low to high.

studies. The detector model for photon showers contains 368 voxels, and the one for electron showers contains 533 voxels. For this dataset, the incoming particles have one of 15 energy levels between 256 MeV to 4 TeV.

2. Dataset 2 [186] contains electron showers from electrons with an incoming energy from 1 GeV to 1 TeV sampled according to a log-uniform distribution. The detector consists of 45 layers in z-direction. Each layer has 144 readout cells, of which 9 are in the radial and 16 in the angular direction. This yields a total of 6480 voxels to simulate. This detector is simulated according to the Par04 GEANT4 [187] fast simulation example.

3. Dataset 3 [188] is simulated like dataset 2, but the granularity of the detector is different. It yields the same number of layers in z direction but has 18 bins in radial direction and 50 in angular direction leading to a total of 40600 voxels.

Compared to the previously discussed generation of jets, the generation process on the CaloChallenge datasets is concerned with *conditional* generation. Each shower comes with the associated energy of the incident particle. This not only means

that the model needs to generate seemingly real showers, but that these should also correspond to the incoming energy that is given as input to the generative model. This motivates the detector *response* as a statistic to consider during the evaluation of the model. The detector response is the ratio between energy deposited in the detector and the incoming energy of the particle:

$$\text{response} = \frac{\sum \mathrm{E_i}}{\mathrm{E_{inc}}}. \tag{6.1}$$

6.2 Calorimeter Data as Point Clouds

Energy deposits in a calorimeter can be converted to a point cloud by assigning the spatial coordinates of hits to the corresponding energy deposits. This representation of calorimeter data as point clouds is beneficial for multiple reasons:

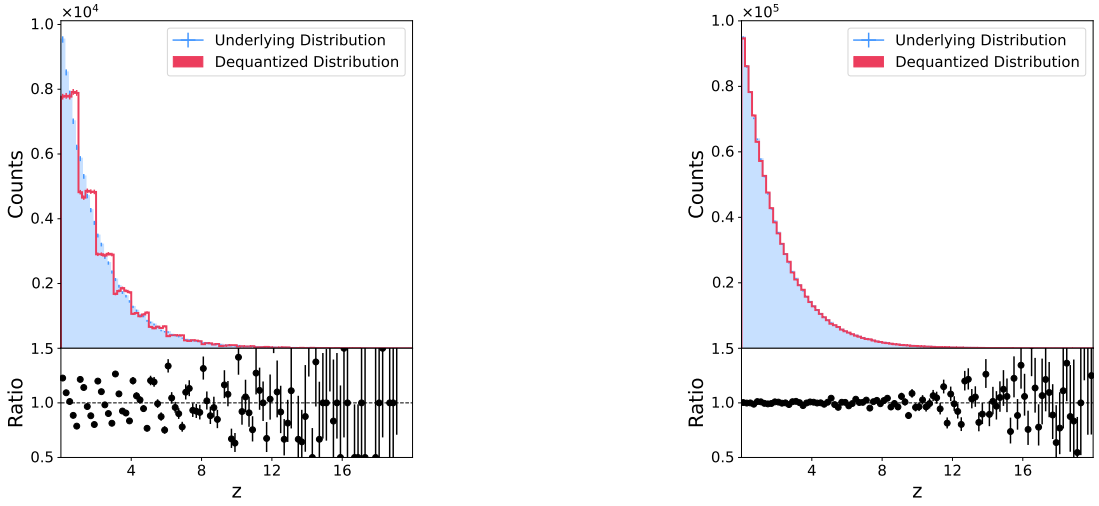
- Memory efficient: in highly granular calorimeters, like the HGCAL [50], the energy deposits are *sparse*. The means that many of the 6.5 million channels will be empty, i.e. have no energy deposits over the detector noise boundary. The benefit of using a point cloud representation of energy deposits is that these cells do not need to be simulated. But note that the point cloud representation only becomes memory efficient if the detector occupancy (e.g. the ratio of hit cells and total cells is below 25%, as 3 additional coordinates per energy deposit are needed for the point cloud representation.

- Irregular Geometry: traditional data formats based on regular grids are unrealistic, since a real detector rarely has a regular shape. A prime example is again the HGCAL, which not only has varying cell sizes, but also an irregular layout consisting of different hexagonal structures.

- More general: a model that is trained to generate a point cloud can be transferred more easily, as discussed in Sec. 6.3.4. This is not possible for a voxel-based model that is bound to generate the geometry it was designed for.

6.2.1 Dequantisation & Preprocessing

Since the data from the CaloChallenge is in a regular grid format, the data needs to be converted to a point cloud representation before starting the training of a point-cloud-based model.
The point cloud representation is obtained by assigning to every energy deposit its indices in the three coordinates (z_i, α_j, R_k). It is not straightforward for a generative model to have an integer output. Thus, the coordinates associated with the hits need to be smeared, which is referred to as *dequantisation*. To compare the effects



(a) Uniform Dequantisation

(b) Linear Dequantisation

Figure 6.2: Comparison of different dequantisation methods on a Gamma distribution with $k = 1$, $\theta = 2$. For the dequantised distribution shown in red, samples are drawn from the Gamma distribution. The sampled value is rounded down and then dequantised. (left) For uniform dequantisation, artificial steps appear in the dequantised distribution. (right) When using the linear dequantisation method, the underlying and dequantised distribution are much more similar.

of different dequantisation methods, first a toy example is studied. The underlying distribution is chosen to be a Gamma distribution with $k = 1$ and $\theta = 2$:

$$f(x; k, \theta) = \frac{\theta^k x^{k-1} e^{-\theta x}}{\Gamma(k)} \quad \text{for } x > 0,\ k > 0,\ \theta > 0. \tag{6.2}$$

The dequantised distribution is obtained by sampling the Gamma distribution, rounding the sampled value down and then applying the dequantisation. The simplest form of dequantisation is to add uniformly distributed noise $U(0,1)$ to the coordinates. But this leads to artificial steps in the distribution, as shown on the left of Fig. 6.2. Not only are these steps artificial, but they can also pose difficulties for the model to learn.
A more sophisticated approach is to linearly interpolate between neighbouring bins. For this, a linearly increasing or decreasing distribution between neighbouring bins is defined as:

$$p(x) = (y_{i+1} - y_i) \times x \quad x \in [0,1], \tag{6.3}$$

where y_i is the yield of the bin i of a histogram. This one-dimensional distribution is then sampled with the use of the Probability Integral Transform to obtain noise that linearly interpolates between neighbouring bins. The algorithm for this is given in Alg. 4.

Algorithm 4 Sampling from Piecewise Linear Distribution

for each bin pair i and $i+1$ **do**
  Define slope $s = y_1 - y_0$ ▷ y_0 and y_1 are values at bins i and $i+1$
  Define Normalisation $k = \frac{y_0+y_1}{2}$ ▷ $\int_0^1 p(x)dx = 1$
  Define linear interpolation distribution $p(x) = \frac{1}{k}(y_0 + sx)$
  Define inverted CDF $f(u) = \frac{\sqrt{y_0^2+2s\cdot k\cdot u}-y_0}{s}$
  Sample uniform $u \sim U(0,1)$
  Apply inverted CDF to obtain sample: $x_{\text{sample}} = f(u)$
end for

Note that the absolute value of the slope s between neighbouring bins can at max be two, due to the following two requirements for probability densities p:

1. The density needs to be normalised: $\int_0^1 p(x)dx = 1$,
2. The density must be non-negative $p(x) \geq 0$.

The right of Fig. 6.2 shows that the linear dequantisation approximates the underlying density significantly more closely[2].

[2]For the CaloChallenge a subtlety is what the underlying distribution is. One could argue that

Thus, for the CaloChallenge the linear dequantisation is used. Since there is already a uniform distribution in the polar angle α, only the coordinates z and R are dequantised. To compare, Figure 6.3 shows the results of linear and uniform dequantisation.

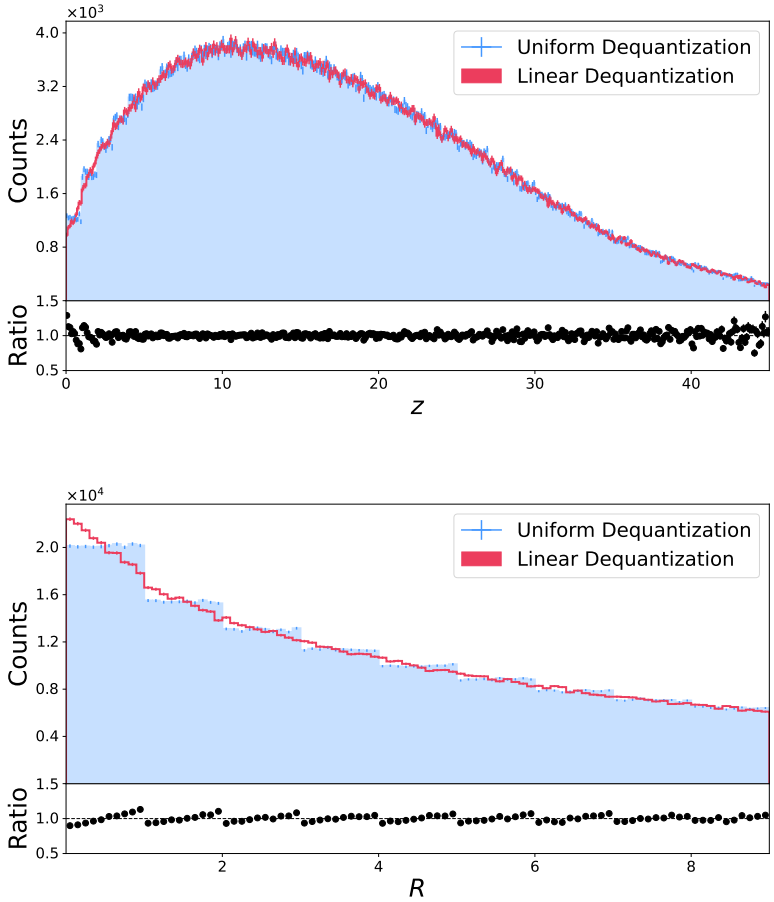


Figure 6.3: Comparison of uniform and linear dequantisation on the CaloChallenge data. (top) In the layer coordinate z the steps are not visible by eye due to the high granularity. (bottom) In the radial coordinate R the steps of the uniform dequantisation are visible, the results from the linear dequantisation are much smoother.

For the training, the energy of hits is Box-Cox transformed. The hit coordinates

the underlying distribution is the one that is measured, and hence is the quantised distribution. However, from a physical point of view, the assumption that the underlying distribution should be smooth between bins certainly holds.

are first scaled to $[0, 1]$ by subtracting the minimum and dividing by the difference between the maximum to the minimum (Min-Max scaling). Then the logit transformation, given in Eq. 6.4 is applied, such that the data lies in $(-\infty, \infty)$[3].

$$\text{logit}(\text{x}) = \log \frac{\text{x}}{1 - \text{x}} \tag{6.4}$$

Finally, the data is standard scaled, as with the `JetNet` datasets.

6.3 Mean-field Aggregation for the CaloChallenge

Since the goal is to sample the conditional distribution $p(x|E_{inc})$, the architecture has to be slightly modified. Instead of only conditioning the mean-field $\bar{\boldsymbol{x}}$ with the number of points, now additionally the incoming energy E_{inc} is used. The fully connected layer Ω used for `JetNet150`, now only takes the conditions (E_{inc}, n) as input and maps them to the same dimension as the mean-field $\bar{\boldsymbol{x}}$. Then, a gated linear unit is applied:

$$GLU(\bar{\boldsymbol{x}}, \boldsymbol{\Omega}(E_{\text{inc}}, n)) = \bar{\boldsymbol{x}} \otimes \sigma(\boldsymbol{\Omega}(E_{\text{inc}}, n)), \tag{6.5}$$

where the σ is a sigmoid function and $\otimes$ represents element-wise multiplication. The updated architecture is shown in Fig. 6.4. Other than that, the architecture remains the same as before. For the training on the CaloChallenge, the size of the MDMA-Flow was reduced to be in the same order of magnitude as for the MDMA-GAN, which is done to for two reasons. The former is that the comparison of the models more fair this way. The latter is that due to the large cloud cardinality on dataset 3, the training of the model would require a batch size of the order $\mathcal{O}(1)$. Similarly, as on the `JetNet150` dataset, the CNF trained with CFM is trained on the CaloChallenge datasets. The architecture is adapted to conditional generation identically to the GAN.

[3]To suppress infinities arising from the logit transformation, the output of the Min-Max scaling is clamped between $(10^{-5}, 1 - 10^{-5})$.

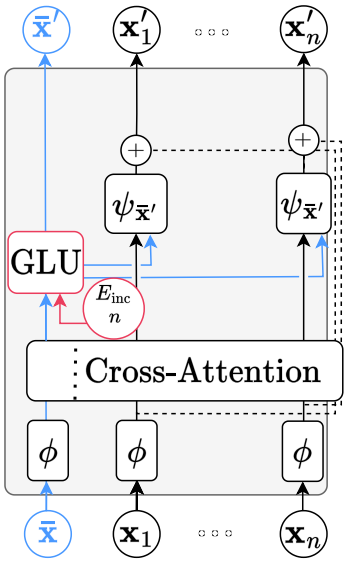


Figure 6.4: The main block of the MDMA architecture for the CaloChallenge, modifications of the previous block are marked in red. The fully connected layer Ω is replaced by a Gated Linear Unit, and the incoming energy of the electron is used as an additional condition. Note that the cosine embedding of the time is not shown in this figure, since it only concerns the MDMA-Flow. However, it is implemented the same way as shown in Fig. 5.16.

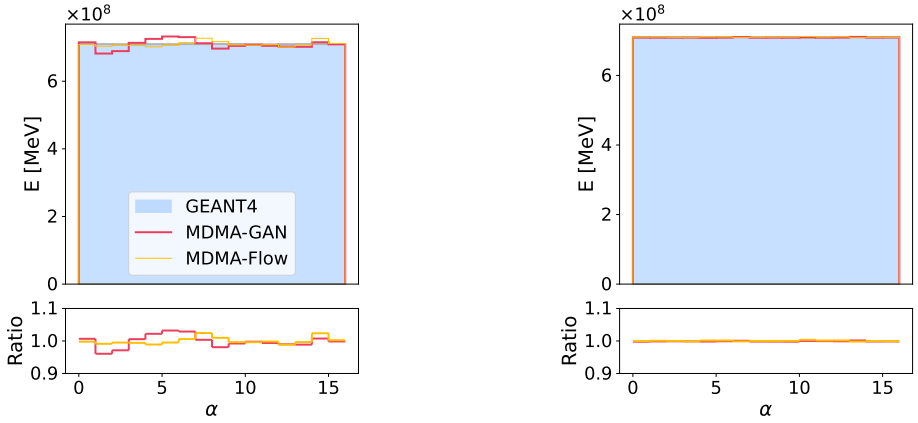
6.3.1 Sampling Synthetic Data

During the evaluation of the model, it was omitted that for the point-cloud-based model, not only the energy of the incoming electron is needed as a condition, but also the number of hits as an implicit input, since the model generates as many points as it receives noise inputs. In practice, another model would be needed that predicts the number of points as a function of the incoming energy of the electron. For the following testing, the number of hits associated with an incoming energy from the testing set was used. In Appendix C.2, the results are shown when the number of hits is deduced from the incoming energy of the electron.

6.3.2 Post-processing

When converting the voxel representation to the point cloud representation, an issue with the conversion in the angular coordinate α was overlooked. The indexing in angular direction starts from zero and ends in the number of voxels in angular direction α_{max}. Thus, in the point cloud representation, the maximum number of bins is closer in space than the third bin. The model is unable to learn this periodicity in α as shown in Fig. 6.5, which shows the energy-weighted angular distribution of samples drawn from the generative model.

A simple way to resolve this, is to rotate generated showers by a random angle, which is sampled from the uniform distribution between $U(0, \alpha_{\text{max}})$. Note that the rotation does not change anything about the structure of the shower, since all hits in a shower are rotated by the same angle. However, this makes use of the angular symmetry of the detector, which makes the model less general. But, here it is used to treat an issue most probably arising from the representation of the dataset.



(a) Raw (b) Post-processed

Figure 6.5: The effect of neglecting the periodicity in the angular coordinate α becomes apparent in the energy-weighted histograms of α, especially for the MDMA-GAN (red). (left) Output of the α coordinates of showers drawn from the model. The model is unable to learn the uniform distribution. (right) Rotating showers by a random angle resolves this.

6.3.3 Results on Dataset 2

In Fig. 6.6 the marginal distributions of samples simulated by GEANT4 and synthetic samples drawn from the MDMA-GAN and MDMA-Flow are shown. Although the results of the MDMA-Flow and MDMA-GAN seem similar, the ratio below histogram of the z coordinate reveals that the MDMA-Flow agrees much better with the GEANT4 distribution. Note that the energy E is given in units of MeV. Figure 6.7

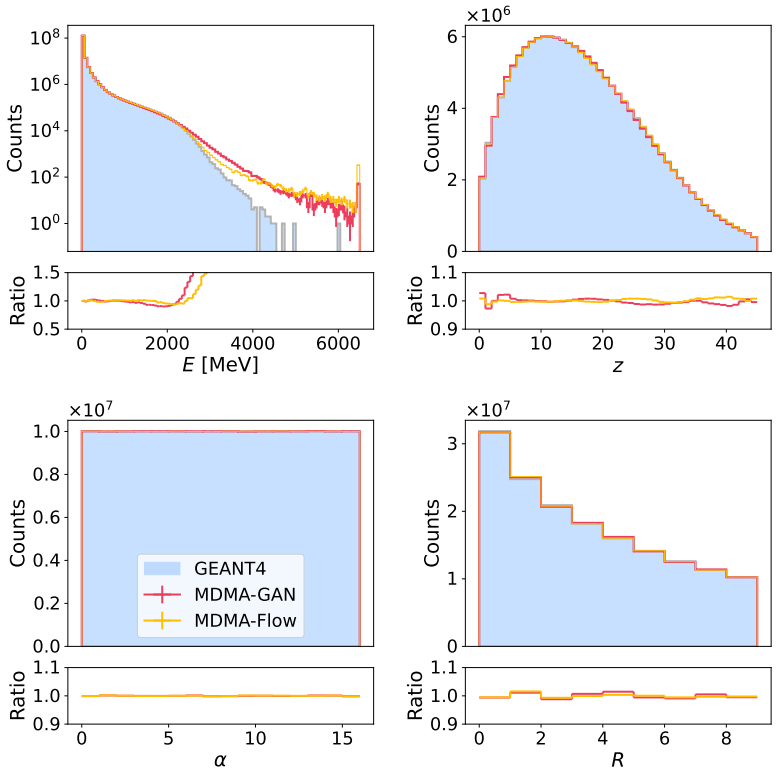


Figure 6.6: Comparison of the GEANT4 simulated showers (blue) to synthetic samples drawn from the MDMA-GAN and MDMA-Flow. The figure depicts the marginal histograms over all generated showers and all generated hits. From left to right, top to bottom, the distribution of the energy and the forward-, angular- and radial coordinates are shown. Below each figure, the ratio between the yields of GEANT4 and synthetic data generated hits is shown. The MDMA-GAN (red) models the z-coordinate significantly worse than the MDMA-flow (yellow).

shows the energy-weighted coordinate histograms, i.e. the energy deposited in the calorimeter per coordinate direction.

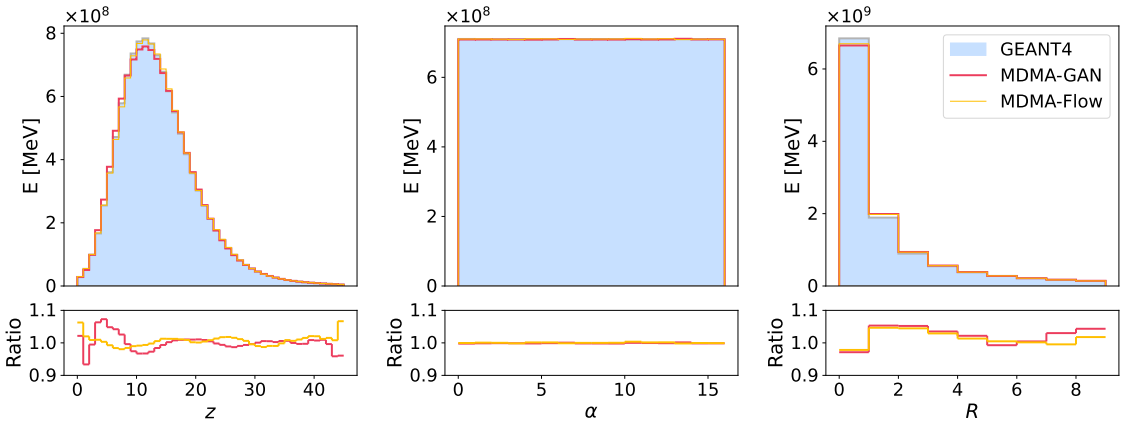
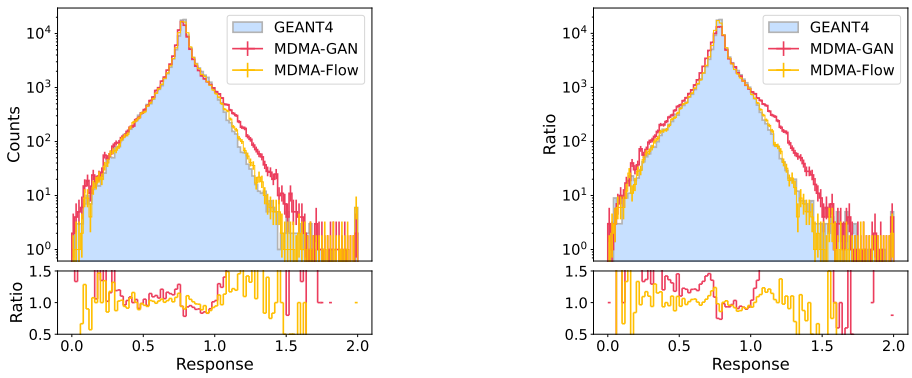


Figure 6.7: The energy-weighted histograms of the forward-, angular- and radial coordinates are shown, with the ratio of GEANT4 data yields divided by synthetic data yields below each histogram. The ratio reveals that there is significant mismodelling, especially in energy deposited per layer in the forward z and radial R direction. However, the MDMA-Flow (yellow) models all distributions significantly better than the MDMA-GAN (red).

A histogram of the detector response, given as the ratio of the deposited energy and the incoming energy $\frac{\sum E_i}{E_{\text{inc}}}$ is shown in Fig. 6.8. The MDMA-Flow also models the shape of the response more accurately than the MDMA-GAN.



(a) Dataset 2 (b) Dataset 3

Figure 6.8: Histograms of the detector response for GEANT4 simulated (blue) and synthetic showers from the MDMA-GAN (red) and MDMA-Flow (yellow). (left) The MDMA-Flow models the detector response more accurately on dataset 2. (right) The MDMA-Flow also models the detector response more accurately on dataset 3.

6.3.4 Results on Dataset 3

On dataset 2, there are up to ~ 4500 hits per shower, which already restricts the model size during training. But for dataset 3 there are up to $\sim 20'000$ hits. This results in a much slower training, since the batch size needs to be reduced by another factor of 5, making training from scratch not feasible. Luckily, the point-cloud-based models show an impressive generalising property. The model trained on dataset 2 can be fine-tuned on dataset 3, and after a single epoch it already produces promising results. The final model is trained for 200 epochs with a reduced learning rate of 10^{-6}. The marginal histograms are depicted in Fig. 6.9 and the energy-weighted histograms are shown in Fig. 6.10. The response of the detector is shown on the right of Fig. 6.8. In Fig. 6.10 the energy-weighted histograms are shown.

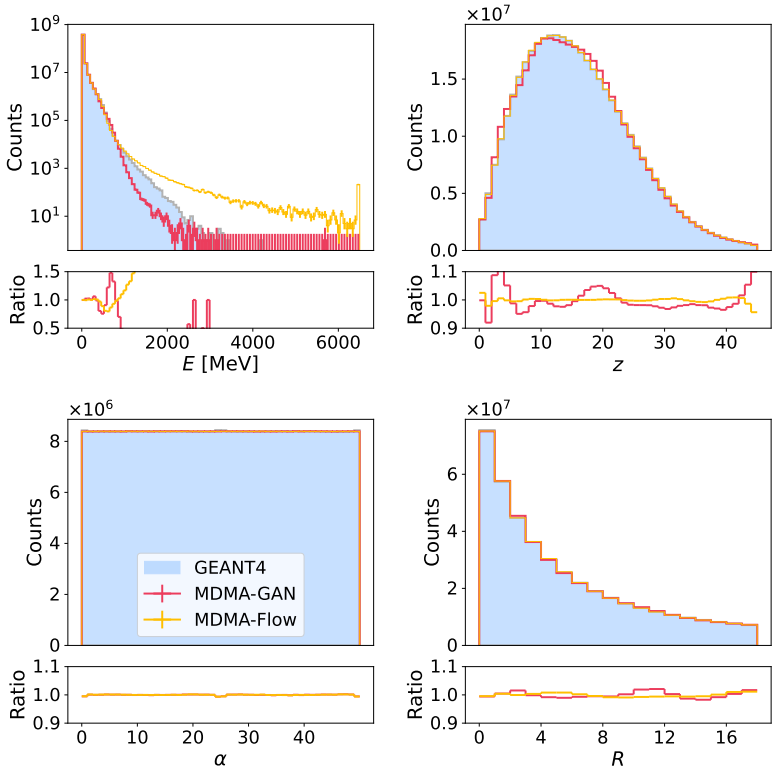


Figure 6.9: Results on dataset 3 of the CaloChallenge. The GEANT4 simulated showers (blue) are compared to samples drawn from the MDMA-GAN (red) and MDMA-Flow (yellow). The figure shows the marginal histograms over all generated hits from all generated showers. From left to right, top to bottom, the distribution of the energy and the forward-, angular- and radial coordinates are shown. Below each figure, the ratio between the yields of GEANT4 and synthetic data yields is shown.

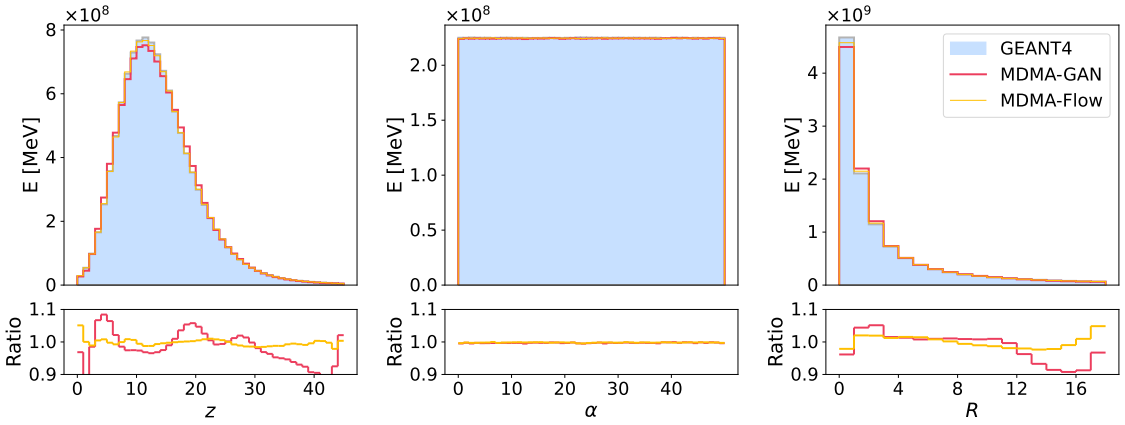
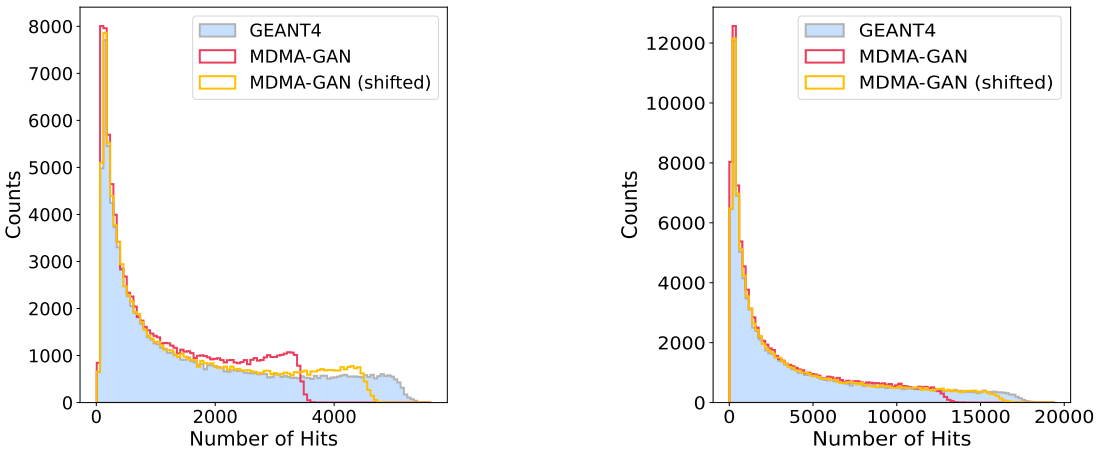


Figure 6.10: The energy-weighted histograms of the forward-, angular- and radial coordinates on dataset 3 are shown. Below each histogram, the ratio of GEANT4 data yields divided by synthetic data yields is shown. The ratio reveals that there is significant mismodelling, especially in energy deposited per layer z. The MDMA-Flow (yellow) models all distributions significantly better than the MDMA-GAN (red).

6.3.5 Mapping Back to the Voxel Representation

As a last step, the point cloud needs to be converted back to the voxel representation. But here an issue of the point cloud representation arises. Nothing restricts the model from placing multiple hits in the same detector voxel, especially, since the model only sees a continuous point cloud space after the coordinates have been dequantised. To account for this, hits with the same (quantised) coordinates are moved to neighbouring cells in a post-processing step. Note that the hardest hit per cell is not moved. Figure 6.11 shows the effect of this post-processing on dataset 2 and 3 for the MDMA-GAN, where histograms of the number of voxels with energy deposits per shower are depicted. The results for the MDMA-Flow are very similar and are shown in Appendix C.1.
During the work on the CaloChallenge, this post-processing and further utilities for dealing with point clouds representation of detectors were published in the `caloutils` python library [189] together with M. Scham.



(a) Dataset 2

(b) Dataset 3

Figure 6.11: Histograms of the number of hits created by the GEANT4 (blue) simulation and after the voxelisation of the samples drawn from the MDMA-GAN (red), together with the effect of the post-processing (yellow). (left) On dataset 2, the model is unable to generate showers, which hit more than 4000 cells, although there are showers with up to 5000 hits in the GEANT4 data. The post-processing effectively lowers the discrepancy between the GEANT4 simulation and MDMA-GAN (right) On dataset 3, the post-processing improves the agreement between the model and GEANT4 distribution even more.

6.4 Quantitative Comparison

The organizers of the CaloChallenge provide a classifier for the evaluation, similarly as it was used in this thesis in Section 4.8.2. There are two classifiers available:

1. High-level classifier: this classifier takes as input high-level features extracted from the voxel-based representation of the showers. The following statistics serve as high-level features:
   - Total energy of the shower E_{tot}
   - Energies per layer E_i
   - Centres of energy in η and ϕ per layer
   - Width of centre of energy in η and ϕ per layer
2. Low-level classifier: this classifier takes as input all voxels. It is a fairly simple fully connected 2-layer classified with 512 hidden features per layer with no strong inductive bias designed for the problem (i.e. rotational symmetry, strong correlation between neighbouring cells).

In the following, the AUC of both classifiers are given as a performance measure. Additionally, the average time to generate a shower is given for every model. As a reference, the average generation time of the GEANT4 simulation per shower is given as well. It might seem misleading to quote the average time because the generation time of GEANT4 scales linearly with the energy of the incoming electron, as shown in Fig. 6.12. As mentioned previously, the energy of the incoming electron is sampled from a log-uniform distribution between 1 Gev and 1 TeV for these datasets. But as Fig. 6.13 reveals, the same approximatively holds for the generative model. Thus, the mean generation time per shower is quoted[4]. The number of parameters of both models are given additionally.

[4]To estimate the mean generation time, the mean of a log-uniform distribution between 0.1 s and 100 s is used.

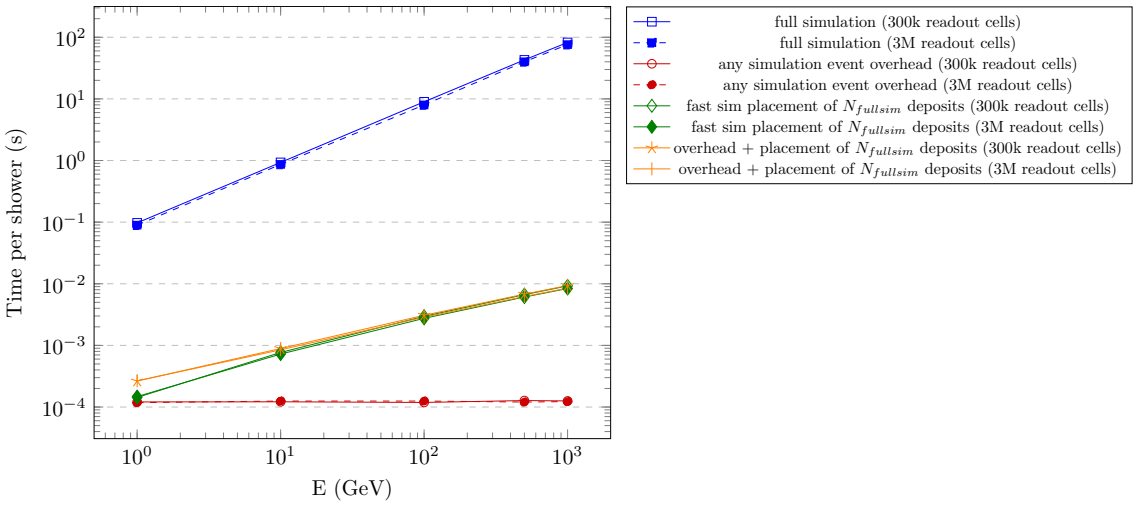
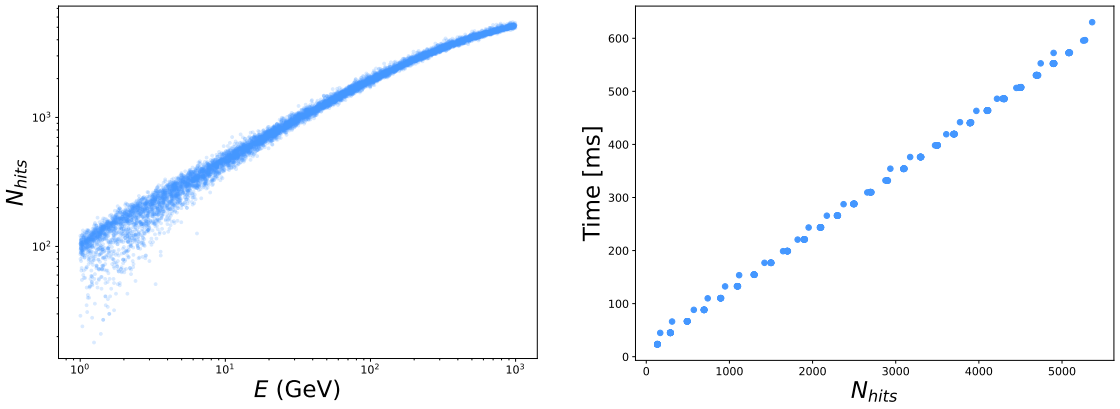


Figure 6.12: The generation time of GEANT4 (blue) from Ref. [190] for dataset 2 and dataset 3 is linearly dependent on the energy of the incoming electron. The figure also shows other contributions to the total generation time that are often neglected in generative models. The contribution related to loading and storing the generated data (red) is constant as a function of the energy. The mapping of the generated hits to the cells (green) is linearly dependent on the energy. The total of these two contributions (yellow) defines the minimum time any generative model would need in practice.



(a) Number of hits vs E

(b) Time vs number of hits

Figure 6.13: The generation time of the generative model also approximately scales linearly with the energy. The left figure shows, that the number of hits per shower is approximately linearly dependent on the energy of the incoming electron. The right figure reveals that the generation time depends linearly on the number of cells that are hit.

Table 6.1: Comparison of the proposed models on dataset 2 and dataset 3 of the CaloChallenge. For the average time for GEANT4, see text.

Dataset	Model	AUC (low-level)	AUC (high-level)	Time $[ms]$	#parameters
Dataset 2	MDMA-GAN	0.94	0.80	1.1	139 k
	MDMA-Flow	0.92	0.76	313.6	257 k
	GEANT4	-	-	$\sim 14 \times 10^3$	-
Dataset 3	MDMA-GAN	0.90	0.93	3.4	139 k
	MDMA-Flow	0.84	0.86	1005	257 k
	GEANT4	-	-	$\sim 14 \times 10^3$	-

6.5 Discussion

The quantitative results confirm what the qualitative results suggested. The distribution of the MDMA-Flow samples appears to be more compatible with the GEANT4 distribution, than its GAN counterpart, since a lower AUC means that the classifier is worse at separating the two distributions. Note, that here the MDMA-Flow was restricted to a smaller number of parameters, such that it is also trainable on dataset 3. The GAN is in terms of generation speed significantly faster, but in practice, e.g. to tackle most of the computing requirements of the HL-LHC, a generation time reduction of a factor 10^4 is not necessary, one order of magnitude would be sufficient. Furthermore, that the models are assigning multiple hits to the same coordinates could be caused by the attention-based aggregation, which allows the models to disregard less important points in the point cloud. From a physics perspective, these more important points are expected to be the more energetic hits in the shower. Therefore, the models could pay less attention to the lower energetic hits in the shower, and place them wherever in the detector, leading to the many double hits. However, Figure 6.9 indicates that both models do not learn the high-energy tail of the energy distribution accurately, which could contradict this argument. However, considering that the energy-weighted distributions agree much better with the GEANT4, as shown in Fig. 6.10, the inaccurate modelling of the higher-energy tail could arise from the low statistics in that region.
Different from the MDMA-GAN, scaling up the number of parameters of the MDMA-Flow improved the performance on the `JetNet150` datasets. Thus, an interesting direction for future work is to explore the dependence of the performance of model size and the amount of training data. The most interesting property of the point-cloud-based model is that it is transferable from dataset 2 to dataset 3 fairly easily. It is unexpected that there is such a large benefit of fine-tuning a pretrained model, where the number of hits is that different.
This motivates the stage-wise training of a point-cloud-based model, directly on the output of GEANT4, without matching the GEANT4 hits to cells. In this context, the stages refer to different clusterings of the GEANT4 point cloud, with a clustering radius that gets reduced for later stages. This would circumvent multiple issues, such as the need to dequantise or the treatment of periodicity in the angular coordinate

because a different coordinate system can be chosen. The stage-wise training also makes the training significantly more efficient in terms of training time.
Generally, it is promising to see that the model, which was designed for the generation of jets, also performs well on conditioned calorimeter simulation. This could further motivate the use of foundation models in HEP, as already discussed in Ref. [191].
Overall, I think that point-cloud-based models are a promising direction for future work, especially in calorimeter simulation. Many other models that took part in the CaloChallenge were working on a grid-based representation. However, I believe that in practice this is not directly transferable to a realistic detector and that the point clouds are a more natural representation for calorimeter showers.

CHAPTER 7

Modelling Parametrised Distributions

The kinematic distributions from the recoreded events left by a beyond SM physics process, like a supersymmetric extension of the SM, are dependent on a priori unknown parameters. To search for signatures of the process, samples are generated for many parameter combinations to trace the likelihood of these unknown parameters by comparing simulated distributions to data. However, since even for the MSSM there are too free parameters, simplified models are usually studied by experimentalists to focus only on a subset of parameters that have an effective impact on the distributions for a given selection of events. For the simplified model studied in this chapter, there are three a priori unknown mass parameters that are scanned. These are the masses of the gluino $m_{\tilde{g}}$, the chargino $m_{\tilde{\chi}_1^\pm}$, and the neutralino $m_{\tilde{\chi}_1^0}$, where the latter is the LSP. Due to the many possible combinations of these three parameters, the chargino mass is usually expressed as a function of the gluino mass and the LSP mass:

$$m_{\tilde{\chi}_1^\pm} = \frac{m_{\tilde{\chi}_1^0} + m_{\tilde{g}}}{2}. \tag{7.1}$$

Otherwise, generating MC samples for every combination is computationally infeasible. The impact of this arbitrary choice of the chargino mass on the sensitivity of a signal versus background classifier is investigated in this chapter. It is further studied how well a generative model can *interpolate* between the distributions for

different parameter choices, i.e. learn and model the dependence of the distribution on the unknown parameter.
First, a toy example of a parametrised two-dimensional Gaussian distribution is discussed. This two-dimensional example highlights the differences between Continuous Normalising Flows (CNFs), trained with Conditional Flow Matching (CFM) and Optimal Transport-based Conditional Flow Matching (OT-CFM), and conditioned discrete Normalising Flows[1] (NFs) to show which model is more suitable to morph one distribution into another. Then, the potential problems with using an NN-based classifier to distinguish between a parameterised signal distribution and a background distribution are explored. Lastly, a search for signatures left by particles arising from a supersymmetric extension of the SM is introduced. For this, the framework of the simplified model `T5qqqqWW` for gluino pair production is used. On this dataset, the same studies as on the two-dimensional examples are discussed, and it is investigated whether synthetic data benefits the median statistical significance of the search.

7.1 Rotating Multivariate Gaussian

In the toy example, a family of Gaussian distributions is studied. The distributions are parametrised by an angle ϕ, which rotates the covariance matrix Σ:

$$\Sigma = \begin{pmatrix} 1.0 & -0.99 \\ -0.99 & 1.0 \end{pmatrix}. \tag{7.2}$$

The rotation matrix R is given by:

$$R(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix}. \tag{7.3}$$

The rotated covariance is then given by:

$$\Sigma_{\text{rotated}}(\phi) = R(\phi)\Sigma R(\phi)^T. \tag{7.4}$$

[1] Note that starting from here, the "discrete" is not omitted any more, to reduce the confusion of the many acronyms and models.

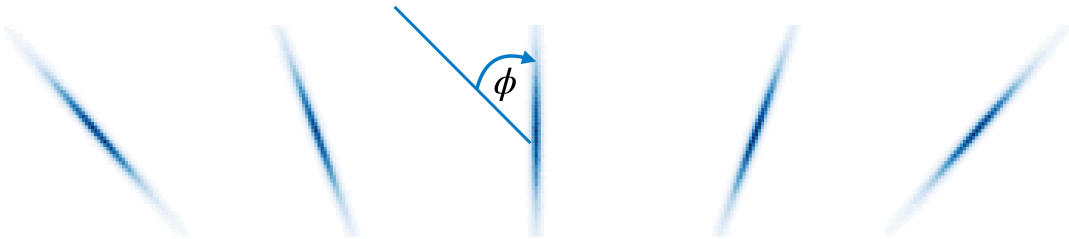


Figure 7.1: The toy distribution has a free parameter ϕ that rotates the covariance matrix of the Gaussian. Note that the two dimensions of the Gaussian are strongly negatively correlated. From left to right, 5 equidistant steps of the rotation from 0 to $\frac{\pi}{2}$ are depicted. The second distribution at $\phi = \frac{\pi}{8}$ is defined to be the distribution chosen by nature.

This rotation of the distribution is illustrated in Fig. 7.1 for 5 different angles ϕ between 0 and $\frac{\pi}{2}$. This angle ϕ stands in correspondence to the chargino mass from the SUSY application discussed later, as the distribution is dependent on the value of this parameter. The range of $[0, \frac{\pi}{2}]$ covers all possible distributions.
To further draw the analogy to particle physics, it is assumed that the "true" distribution in "nature" is realised for one particular choice of $\phi = \frac{\pi}{8}$. It is then studied whether an NN-based classifier is sensitive to the true signal distribution, if it was trained on signal distributions from different choices of the parameter ϕ. The background in these studies is given by a standard Gaussian distribution $\mathcal{N}(0, \mathbb{I}_2)$, which overlaps with the signal distribution for all choices of ϕ. It is then investigated, whether a generative model can capture the dependence of the distribution on the parameter ϕ. In this example, the generative model is thus trained to transform the signal distribution from $\phi = 0$ to $\phi = \frac{\pi}{2}$.

7.1.1 Modelling the Dependence on ϕ

As was shown previously, normalising flows and their continuous version, perform best at modelling unknown low-dimensional distributions. However, here the goal is not to model a target distribution, but instead to model a whole family of distributions. Conditioned discrete NFs would provide a suitable candidate, but since in practice only samples from few choices of the underlying parameter ϕ are available, it is unclear how well NFs can model this dependence. CNFs trained with (OT-)CFM

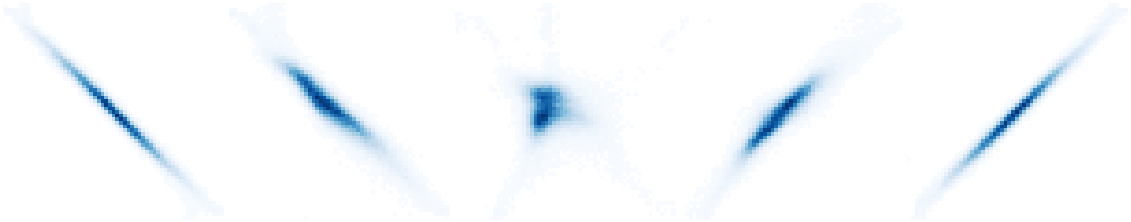


Figure 7.2: The figure depicts samples drawn from a conditioned discrete NF that was trained on samples of the first and fifth distribution. The condition of the start- and endpoint are chosen to be -1 and 1. The three figures in the middle are samples for the condition chosen to be $[-\frac{1}{2}, 0, \frac{1}{2}]$ respectively. This figure shows that conditioned discrete NFs are not the optimal candidate.

provide a more suitable inductive bias, since they are directly trained to transform one distribution into another.

7.1.2 Conditioned Discrete Normalising Flows

The same conditioned discrete NF architecture from Section 4.4.4 is used and trained on samples from the Gaussian distribution with the rotated covariance matrix $\Sigma_{\text{rotated}}(\phi)$ at $\phi = 0$ and $\phi = \frac{\pi}{2}$. In this case, the condition in this context is the angle ϕ which is arbitrarily set to -1 for $\phi = 0$ and 1 for $\phi = \frac{\pi}{2}$. The NF is then evaluated at 5 different values of the condition $[-1, -0.5, 0, 0.5, 1]$ corresponding to $\phi = \frac{\pi}{2} \cdot \frac{i}{4},\ i \in [0, 1, 2, 3, 4]$.
These results are depicted in Fig. 7.2. Although the NF excels at modelling the distribution at the start- and endpoint, it does not capture the dependence on the condition as desired, rather it produces a mixture of samples from the start- and endpoint. However, this should be expected since the NF is not incentivised during training to enforce a smooth transformation between different values of the condition.

7.1.3 Continuos Normalising Flows

It was shown already in Section 4.7.3 in Fig. 4.19c that CNFs trained with OT-CFM can smoothly transform a chosen distribution into another. Hence, they are also a promising candidate for this application. In Fig. 7.3 the results of training a

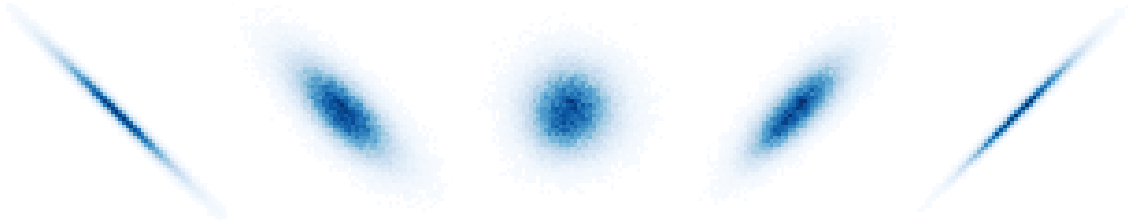


Figure 7.3: The figure depicts samples drawn from the trajectory of a continuous NF that was trained with OT-CFM to transform the Gaussian and $\phi = 0$ in the Gaussian where the covariance matrix has been rotated by an angle $\phi = \frac{\pi}{2}$. Surprisingly, the OT path of the CFM seems to go through a Gaussian distribution with a diagonal covariance matrix.

CNF on the rotating Gaussian example are depicted. At first, one might think that the results of this are worse than for the NF, but there is a subtle but crucial detail in this example. The dependence of the distribution as a function of ϕ is not equal to the optimal transport path. The CNF moves points that are above the diagonal, which goes from $(-\infty, \infty)$ to $(\infty, -\infty)$, to the top right, and points below the diagonal to the bottom left. This is best illustrated when plotting the trajectory of individual points during the solution of the ODE, as depicted in Fig. 7.4. Introducing a midpoint during the training of the CNF, fixes the path of the CNF to the desired one. Instead of moving the points from the distribution at $\phi = 0$ to $\phi = \frac{\pi}{2}$, the model is trained to first move the distribution at $\phi = 0$ to $\frac{\pi}{4}$ and from there to the distribution at $\frac{\pi}{2}$. The results from training on this setup are depicted in Fig. 7.5, yielding the desired results.

Note that there is a significant conceptual difference between using conditioned discrete NF or CNFs to transform a distribution into another. In the NF case, the model is trained to transform the distribution $\phi = 0$ and $\phi = \frac{\pi}{2}$ to a Gaussian. The condition in this case just helps to disentangle the latent space, as was shown in Section 4.4.4 in Fig. 4.16. In the CNF trained with CFM, the model is specifically trained to transform a chosen distribution to the other.

Figure 7.4: Trajectories of the points that are moved by the CNF. Due to the ambiguity of possible paths, the CNF treats points above the diagonal differently than points below the diagonal. (left) Points above the diagonal, i.e. points for which the sum of the first and second coordinate is greater than zero, are moved to the top right. (right) Points below the diagonal are moved to the bottom left.

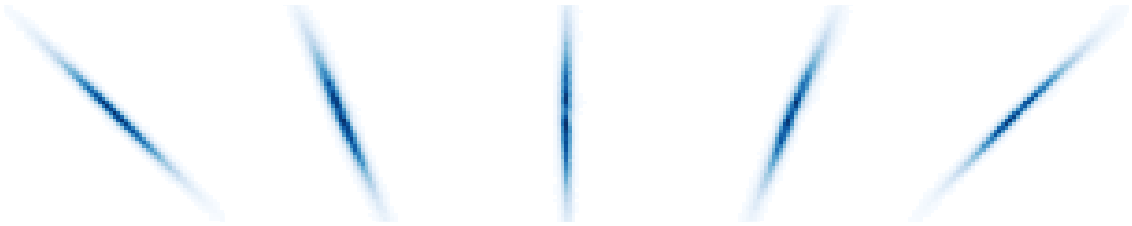
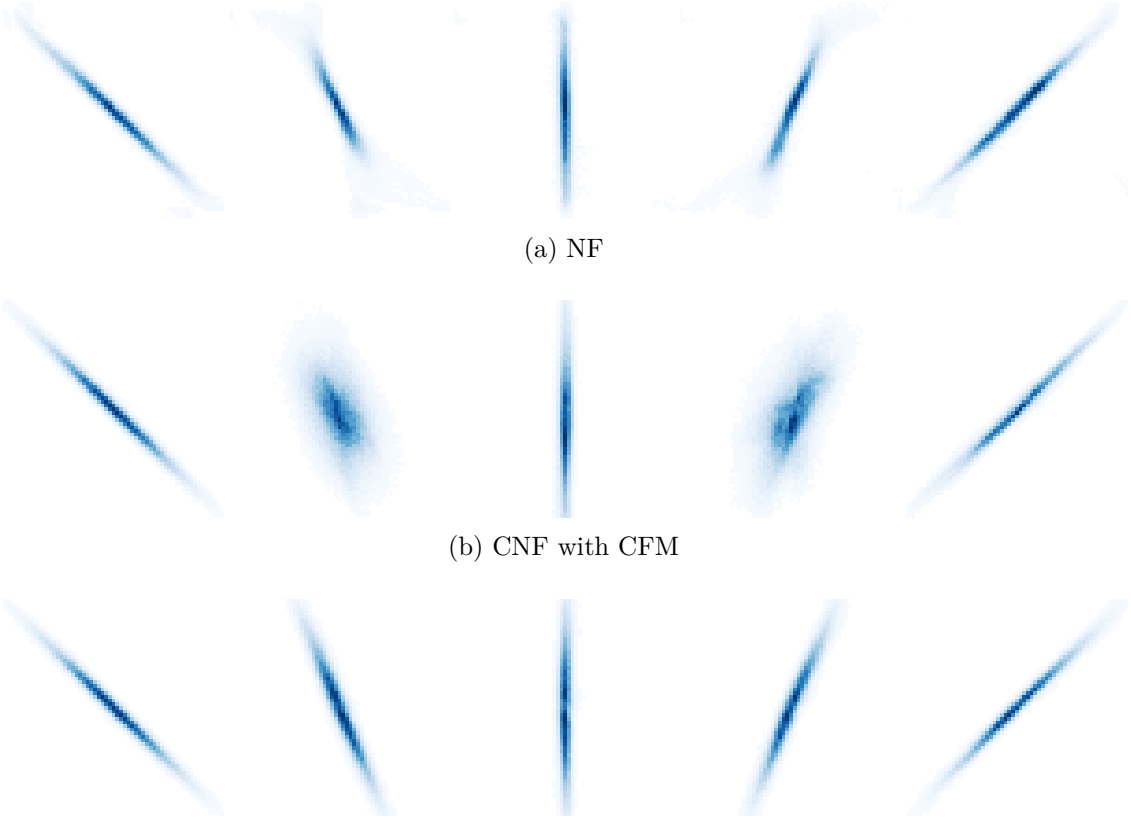


Figure 7.5: Results of training the CNF with OT-CFM, when constraining the possible paths by adding a midpoint through which the CNF should transport the distribution. The results are nearly indistinguishable from the true distribution that was used to generate the samples.

To comprehensively compare the different approaches, Fig. 7.6 depicts the results when training on a dataset that contains the midpoint of a discrete NF and CNFs trained with CFM and OT-CFM[2]. Even though the midpoint is added, neither the CNF trained with CFM, nor the discrete NF do not produce the desired results, since the distribution at $\phi \in [\frac{\pi}{8}, \frac{3\pi}{8}]$ is significantly different from the underlying distribution, at these parameter choices. Together with the results from Section 4.7.4, this suggests that CNFs trained with OT-CFM provide the best candidate to interpolate between two distributions.

[2]For the NF the condition corresponding to the midpoint is chosen to be 0.



(a) NF

(b) CNF with CFM

(c) CNF with OT-CFM

Figure 7.6: Comparison of the different models used to transform one distribution into another. (top) Adding the midpoint to the training certainly improves the modelling at $\phi = 0$. However, the samples at $\phi = \frac{\pi}{8}$ and $\phi = \frac{3\pi}{8}$ show significant differences to the desired distribution. (middle) The CNF trained with CFM also does not model the 2^{nd} and 4^{th} distribution correctly. (bottom) The OT-CFM performs best at modelling the dependence of the distribution of ϕ.

A Different Perspective on Generative Models

In the previous chapters, NFs, GANs and CNFs were used to generate synthetic data. The generative models were considered black boxes, that create synthetic data from noise. However, generative models can be seen as transformations that reshape a standard Gaussian distribution to the distribution of PYTHIA/GEANT4 data. Using the OT-CFM framework to construct the probability path, further enforces that this reshaping is done with minimal transport cost. From this perspective, it becomes clear why discrete NFs are not the best candidates for the transformation of a chosen distribution into another, since they are usually trained to transform the data distribution into a standard Gaussian distribution[3].

7.1.4 Binary Classification Studies

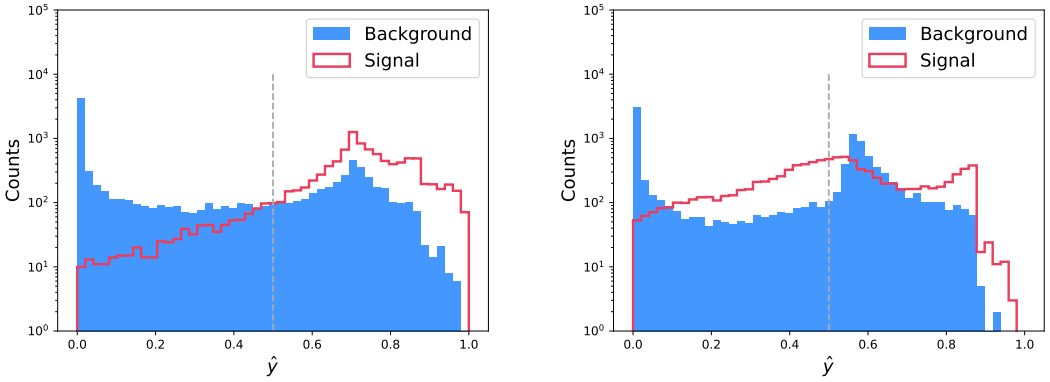
Here, the parallel to the HEP searches for new particles is explored further. An NN-based classifier is commonly used to isolate signatures left by potential beyond SM physics, hidden in the large SM background. On the toy example, a standard Gaussian distribution $\mathcal{N}(0, \mathbb{I}_2)$ is used to represent the background distribution. Two training paradigms are compared:

1. Baseline approach: a classifier is trained on signal samples from the underlying distribution at $\phi = \{0, \frac{\pi}{4}, \frac{\pi}{2}\}$.

2. Interpolation approach: a classifier is trained on signal data drawn from the CNF, which was trained to transform the signal distribution from $\phi = 0$ to $\phi = \frac{\pi}{2}$. Note that the midpoint at $\frac{\pi}{4}$ is used during training as discussed previously. To generate the signal samples, the ODE associated with the CNF is solved from $t = [0, 1]$ with 200 time steps. Then, the signal samples are drawn uniformly from all time steps.

Note, that both approaches have access to the equal amount of "real" data drawn from the ground truth distributions. In this toy example, both models are *tested* on a sample of the underlying, unseen "true" distribution at $\phi = \frac{\pi}{8}$.

[3]Note that two NFs ϕ_1, ϕ_2 can transform one distribution (X_1) into another (X_2) by transforming to an intermediary Gaussian $X_2 = \phi_2^{-1} \odot \phi_1(X_1)$. But this construction will always go through a standard Gaussian, which is not desired.

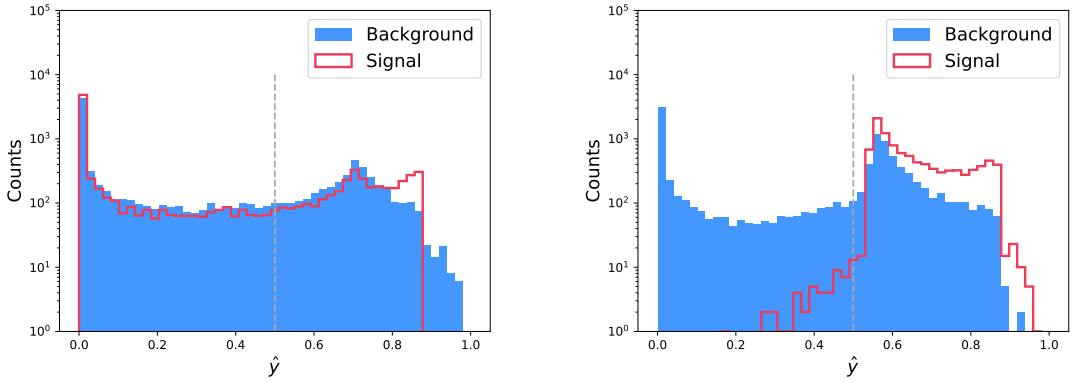


(a) Baseline (b) Interpolation

Figure 7.7: Validation of the baseline and interpolation training paradigm on a holdout dataset, where the signal samples come from the same ϕ choice as seen during training. The figure shows the prediction $\hat{y}$ on signal samples (red) and background samples (blue). (left) The baseline classifier performs better at recognising signal samples and is also slightly better at rejecting background samples. (right) For the classifier that was trained on synthetic signal samples from the CNF, a large portion of the signal is predicted to be background.

The classifier is a simple fully connected NN, with 4 layers with 128 nodes each. It is trained with Focal Loss with $\gamma = 1.5$, described in Section 3.1.1. As validation of the training, both models are evaluated on an independent set of samples coming from the same distribution as seen during training, i.e. the signal distributions for $\phi \in [0, \frac{\pi}{4}, \frac{\pi}{2}]$. Figure 7.7 depicts the output of both classifiers $\hat{y}$ on signal and background samples. The figure indicates that the baseline classifier is better at recognising signal samples and also slightly better at rejecting background samples. However, when testing the classifier on signal samples from the distribution with $\phi = \frac{\pi}{8}$, the benefits of the training on the synthetic samples become apparent. The corresponding results are depicted in Fig. 7.8, which again shows the output of the classifier $\hat{y}$ for signal and background samples. On the left, the prediction of the baseline classifier is depicted, which highlights that the baseline classifier does not recognise the signal for a different ϕ choice. On the right, the prediction of the classifier that was trained on synthetic signal samples coming from the CNF is depicted. This classifier predicts the correct class ($\hat{y} > 0.5$) for the majority of signal

samples, which strongly motivates the idea that the synthetic data can be used to improve the classifier sensitivity to the signal. It is also interesting to see that it performs significantly better than on the validation. A possible explanation for this is that the signal at $\phi = 0$ and $\phi = \frac{\pi}{2}$ are at the boundaries of the signal support.



(a) Baseline (b) Interpolation

Figure 7.8: Comparison of the baseline classifier to the one trained on signals samples drawn from the generative model. The figure shows the output of the classifier $\hat{y}$ for signal samples (red) at $\phi = \frac{\pi}{8}$ and background samples (blue). (left) The baseline classifier does not recognise the signal coming from the signal distribution, as it is almost identically distributed as the background distribution. (right) The model, which was trained on the synthetic signal samples from the CNF, recognises the signal class significantly better.

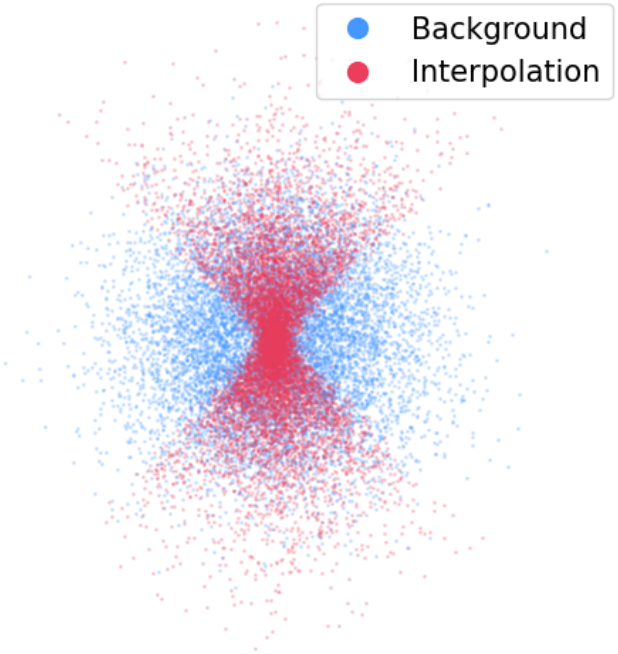


Figure 7.9: The figure depicts samples from the background class (blue) and samples drawn from the generative model (red) that interpolates between the signal distributions at $\phi = 0$ and $\phi = \frac{\pi}{2}$. Note that in this context, a sample refers to a single point, with two coordinates. By construction, both classes have significant overlap, which makes it difficult for a classifier to distinguish the two classes.

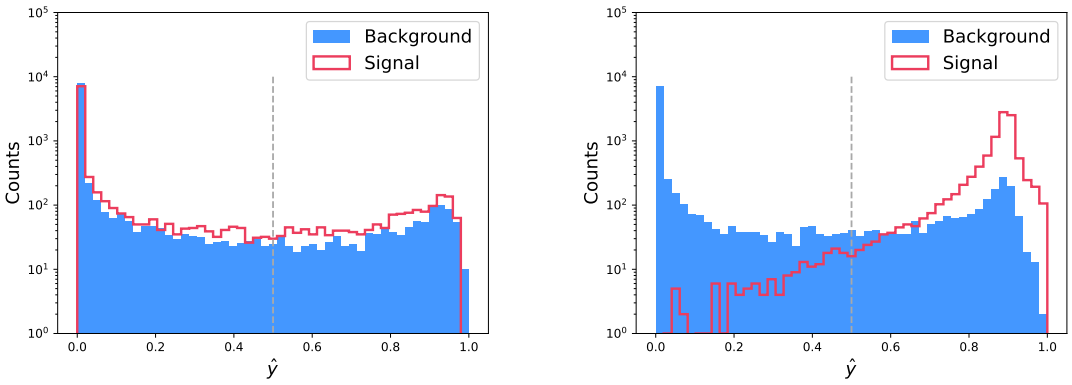
Considering the support of the signal and background distribution as visualised in Fig. 7.9 raises an important point. The signal class not only completely overlaps with the background distribution, but the density is also similar and only a factor two higher than the background distribution. To make the distribution of the two classes more separable, a four-dimensional toy example is constructed. Instead of drawing one sample $\boldsymbol{x} = (x_1, x_2)$ from the signal and background distribution, two samples $\boldsymbol{x}, \boldsymbol{y}$ are drawn and concatenated[4], yielding a four-dimensional sample:

$$z = (\boldsymbol{x}, \boldsymbol{y}) = (x_1, x_2, y_1, y_2) \in \mathbb{R}^4. \tag{7.5}$$

This leads to a different classification problem, which is now over a four-dimensional space, where a distinctive signature is present in signal samples, which a classifier can use to distinguish versus the background. Namely, a line connecting the former and latter two coordinates has a small vertical distance to the origin for signal samples, which is very unlikely for background samples. The performance of the two approaches on the higher dimensional problem is shown in Fig. 7.10 and motivate the training on the synthetic signal samples even further, as now the interpolation approach gets even better at recognising the signal and does not get significantly worse at rejecting background samples.

Note that the CNF is modelling the dependence of the signal distribution on ϕ exceptionally well on this toy example. Furthermore, the overlap of the signal distributions for different choices on ϕ is small, which can explain the observed results. However, it is still interesting to see how badly the baseline classifier performs on data for different choices of ϕ than it has seen during training. It is especially interesting to see that the baseline classifier also performs as badly on the higher dimensional problem. This indicates that the classifier trained with the baseline approach does not learn the signal-specific signature, that a line connecting the first two coordinates with the last two coordinates passes close by the origin. To draw the correspondence to the SUSY search; the parameter underlying the beyond SM signal distribution, i.e. the chargino mass, is a priori unknown, similar to the parameter ϕ on this toy example. However, for the training of the classifier a choice for its value needed

[4]Note, that these two samples always come from either the signal or background distribution. Additionally, the signal samples always come from the same choice of ϕ.



(a) Baseline (b) Interpolation

Figure 7.10: Comparison of a baseline classifier to the one that was trained on signals from the generative model evaluated for the data distribution, where $\phi = \frac{\pi}{8}$. A histogram of the predictions $\hat{y}$ on background samples (blue) and signal samples (red) is shown, (left) Although the baseline classifier could learn that signal samples lie close to a line through the origin when projected to a two-dimensional plane, it does not generalise and does not recognise the signal distribution with $\phi = \frac{\pi}{8}$ at all. (right) The model that is trained on the synthetic signal samples drawn from the generative model not only recognises the signal class accurately, but the similar shapes of the background distribution shown on the left and right suggest, that the classifier does not get significantly worse at recognising background samples as before on the two-dimensional classification.

to be made. But it is very unlikely that the signal distribution for the parameter choice of nature is identical to the one from the training distribution. The current approach employed in the search for gluino pair production is similar to the baseline approach[5]. However, in this study, this lead to a classifier that did not recognise the signal from other parameter choices.

[5]However, instead of samples from three values of the chargino mass, only samples of one choice are used.

7.2 Search for Gluino Pair Production

Now that the concepts have been established, the analysis of the simplified model for gluino pair production (`T5qqqqWW`) is introduced. The signature left by the signal process is characterised by a high jet multiplicity, a large missing transverse momentum $p_{\mathrm{T}}^{\mathrm{miss}}$ and the presence of two W bosons. The analysis focuses on the case where one of the two W bosons decays leptonically to a muon $\mu^{\pm}$ or an electron $e^{\pm}$ together with a neutrino ν. Since the gluino mass is chosen to be in the TeV range, its decay products are expected to carry a high transverse momentum, which motivates the use of hadronic transverse momentum, defined as the sum of the transverse momentum of all jets. Before introducing the signal region, *b tagging*, an important concept common in HEP, needs to be introduced, which refers to the identification of b jets. Due to the long lifetime of b mesons ($\sim 1.6 \cdot 10^{-12}$), they travel some distance before decaying. This can can be used to identify jets originating from b quarks. In this thesis, the `deepJet` tagger [192] is employed to tag b jets. The signal region of the simplified model considered here requires the absence of b tagged jets to suppress background from top pair production, since tops almost exclusively decay to b quarks together with a W boson. The main selection criteria used to define the signal region are given in Table 7.1[6].

Table 7.1: Signal region cuts

Leptons with $p_{\mathrm{T}} > 25$ GeV:	$n_{\mathrm{leptons}} = 1$
Hardest jet:	$p_{\mathrm{T}}^{\mathrm{jet},1} > 80$ GeV
Second-hardest jet:	$p_{\mathrm{T}}^{\mathrm{jet},1} > 80$ GeV
$L_T = p_{\mathrm{T}}^{\mathrm{miss}} + p_{\mathrm{T}}^{\mathrm{lep}}$:	$L_T > 250$ GeV
$H_T = \sum_{\mathrm{jets}} p_{\mathrm{T}}$:	$H_T > 500$ GeV
Number of jets:	$n_{\mathrm{jets}} \geq 5$
Number of b-tagged jets:	$n_{\mathrm{b}} = 0$

Figure 7.11 shows the signal and the different background contributions in the signal region. For the signal sample, the gluino and neutralino mass were chosen as $m_{\tilde{g}} = 2200$ GeV and $m_{\tilde{\chi}_1^0} = 100$ GeV respectively. The chargino mass $m_{\tilde{\chi}_1^{\pm}}$ is set

[6] For brevity some aspects are omitted, e.g. the criteria for good and veto leptons, the choice of the b-tagging working point and more. The detailed signal region definition is given in Ref. [193].

to the mean of both masses, or equivalently the LSP mass plus half of the mass difference. Going from left to right, top to bottom, the first nine plots show the transverse momentum p_{T}, the pseudorapidity η and the azimuthal angle ϕ of the lepton and the two hardest jets. Next is the H_T and then the distribution of the number of jets is shown. Following are the masses of the two hardest jets, the magnitude of the missing transverse momentum magnitude $p_{\mathrm{T}}^{\mathrm{miss}} = \left|\boldsymbol{p}_T^{\mathrm{miss}}\right|$ and its azimuthal component ϕ^{miss}. The missing transverse momentum is an important variable for most searches for dark matter candidates. It is based on momentum conservation in the x-y plane and is defined as the negative vectorial sum of the transverse momentum of all reconstructed particles. Since the neutralino is stable, it evades detection, leading to a significant momentum imbalance in the transverse plane.

The final plot depicts $\Delta\phi$, which is the angle in the transverse plane between the lepton and the reconstructed W boson. The W boson is reconstructed assuming that the $p_{\mathrm{T}}^{\mathrm{miss}}$ comes exclusively from the neutrino arising from the leptonic W decay. The $\Delta\phi$ variable is especially discriminative versus SM backgrounds, since for signal events, the $\boldsymbol{p}_T^{\mathrm{miss}}$ vector is randomised by the transverse momentum of the LSPs, as illustrated in Fig. 7.12. Note that it peaks at 0 for the SM background because the lepton and the W are strongly correlated.

The background is simulated by the full simulation of the CMS detector, and the signal is simulated with the `FastSim` package. A striking difference between the two simulation approaches is that for `FastSim` the collision point is assumed to be in the exact middle of the detector, which is not the case in the real experiment, something that is accounted for in the full simulation. The effect of the collision point not being in the exact centre of the x-y plane leads to a modulation in ϕ, which is especially apparent in the azimuthal component of the missing transverse momentum ϕ^{miss}. This needs to be considered when training the signal versus background classifier by reweighting the signal samples based on their ϕ^{miss}, as discussed in Section 7.2.4.

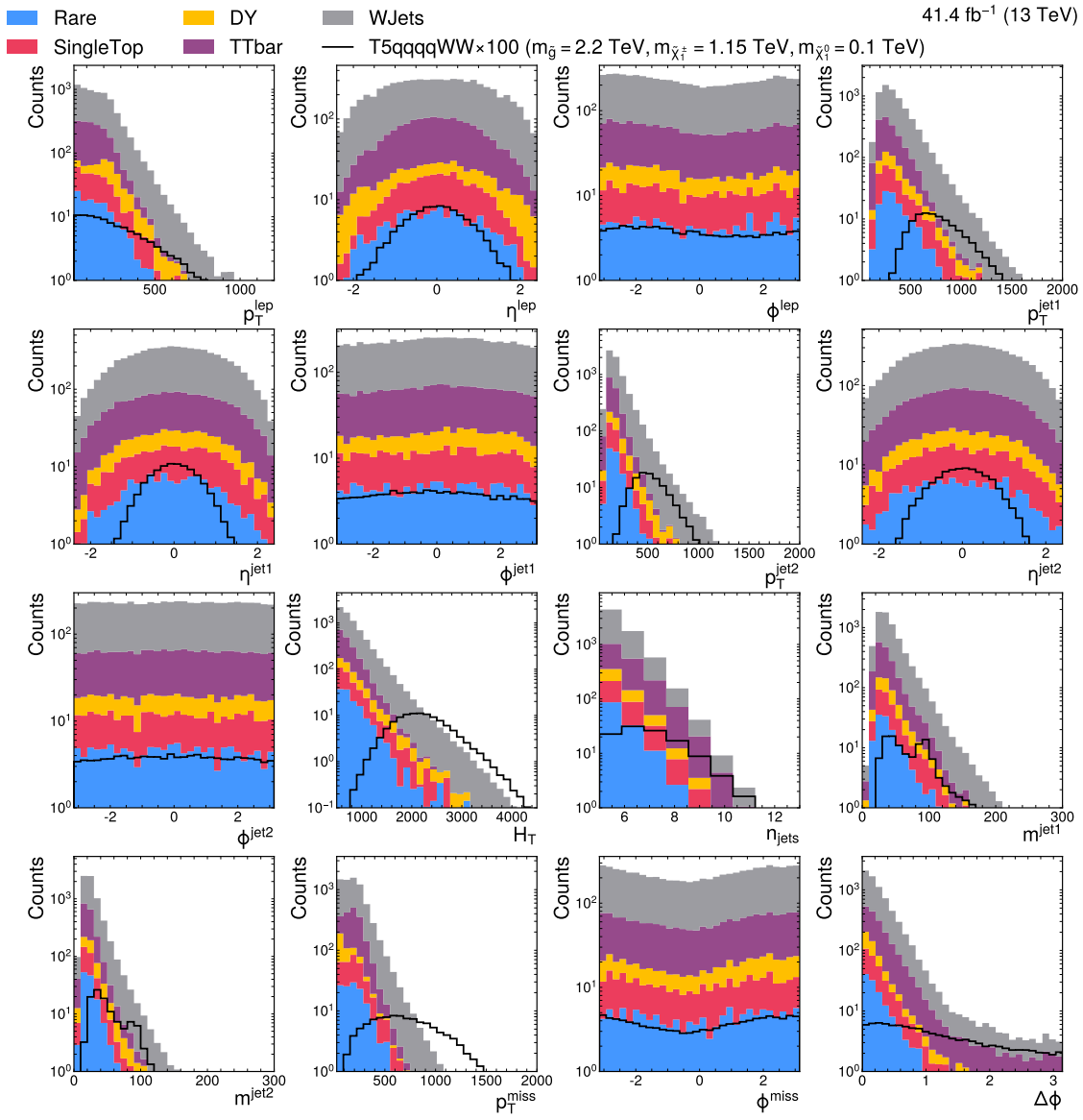


Figure 7.11: Signal and background composition in the signal region of the search after gluino pair production. The background class "Rare" contains the production of top quarks in association with vector-bosons and diboson contributions.

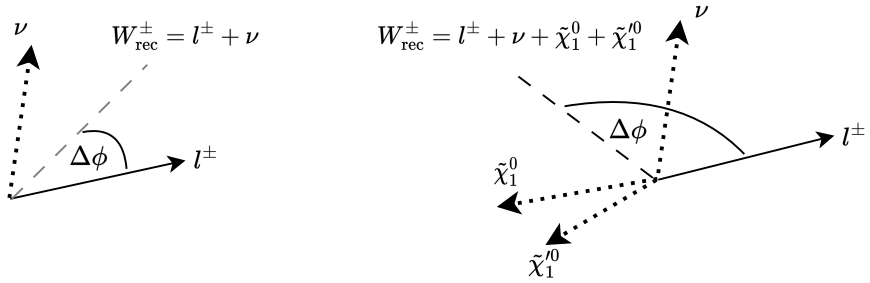


Figure 7.12: Schematic describing the calculation of the $\Delta\phi$. (left) For background events, the only source of missing transverse momentum comes from the neutrino ν coming from the W decay and thus $\Delta\phi$ clusters at small values, since the lepton l and $\boldsymbol{p}_{\text{T}}^{\text{miss}}$ are strongly correlated. (right) For signal events, the $\boldsymbol{p}_{\text{T}}^{\text{miss}}$ is randomised by the $\boldsymbol{p}_{\text{T}}^{\text{miss}}$ arising from the two neutralinos that evade detection.

7.2.1 Interpolating the Dependence on the Chargino Mass

Here, the angle ϕ from the previous study on the rotating Gaussian is identified by the chargino mass $m_{\tilde{\chi}_1^{\pm}}$. This identification also brings important considerations as for the rotating Gaussian, the choice of path had to be implicitly constrained by choosing a midpoint, such that points only move in a horizontal direction. Now, it is difficult to generalise such a constraint to a higher dimensional space, not to speak of the physical constraints that could be present in physics data. However, the exploration of such a constraint is an interesting direction for future work, fusing ML even further with HEP.
For the following, MC samples of the signal are produced at three different values of the chargino masses determined by the *mass mixing* ϵ:

$$m_{\tilde{\chi}_1^{\pm}} = m_{\tilde{\chi}_1^0} + \epsilon \cdot (m_{\tilde{g}} - m_{\tilde{\chi}_1^0}),\ \epsilon \in \{0.25, 0.5, 0.75\}. \tag{7.6}$$

Note that the sample at $\epsilon = 50\%$ mass mixing is not used during training, unless stated explicitly, and serves to test the generalisation property of the classifiers trained with the training paradigms included in the following. In Fig. 7.13 the previously shown histograms are depicted for the different choice of the chargino mass. They reveal that there are significant differences between the different choices of the chargino mass. The first 15 histograms depict the variables that go into the signal versus the background classifier. The last subfigure on the bottom right

depicts $\Delta\phi$, which is a discriminating variable between the different chargino masses. As previously discussed, it is calculated from the lepton $\boldsymbol{p}_{\mathrm{T}}$ and $\boldsymbol{p}_{\mathrm{T}}^{\mathrm{miss}}$. Similar to the relative invariant mass studied in Chapter 5, it serves as a powerful summary statistic to test whether the generative model can capture the correlations between the variables accurately.

For this training, three different kinds of preprocessing are used for the different groups of variables:

1. Box-Cox scaling was used on the following variables that follow an approximately exponential distribution: $[p_{\mathrm{T}}^{\mathrm{lep}}, p_{\mathrm{T}}^{\mathrm{jet},1}, p_{\mathrm{T}}^{\mathrm{jet},2}, p_{\mathrm{T}}^{\mathrm{miss}}, m^{\mathrm{jet},1}, m^{\mathrm{jet},2}, H_T]$

2. Standard scaling was used for the approximately Gaussian distributed variables: $[\eta_T^{\mathrm{lep}}, \eta_T^{\mathrm{jet},1}, \eta_T^{\mathrm{jet},2}, n_{\mathrm{jets}}]$. Note that the number of jets n_{jets} was dequantised first by adding uniform noise.

3. A Quantile scaling, which makes use of the Probability Integral Transform, to transform from an approximately uniform distribution to a standard Gaussian distribution was used for: $[\phi^{\mathrm{lep}}, \phi^{\mathrm{jet},1}, \phi^{\mathrm{jet},2}, \phi^{\mathrm{miss}}]$

A CNF is then trained with OT-CFM to transform the data distribution at $\epsilon = 25\%$ mass mixing into the one at $\epsilon = 75\%$ mass mixing. Note that unlike previously, the midpoint at $\epsilon = 50\%$ mass mixing is not used to train the generative model.

7.2.2 Neural Network Architecture & Training

For the studies on this dataset, it proved that a more complicated information aggregation method, as used previously in Chapter 5 and 6, does not lead to improved performance. A possible explanation is that these more sophisticated methods especially perform well in high-dimensional unstructured data, which is not the case with the fifteen high-level variables in this study. The hyperparameter choice for this model is given in Table 7.2.

To quantify the performance, two distance measures are used:

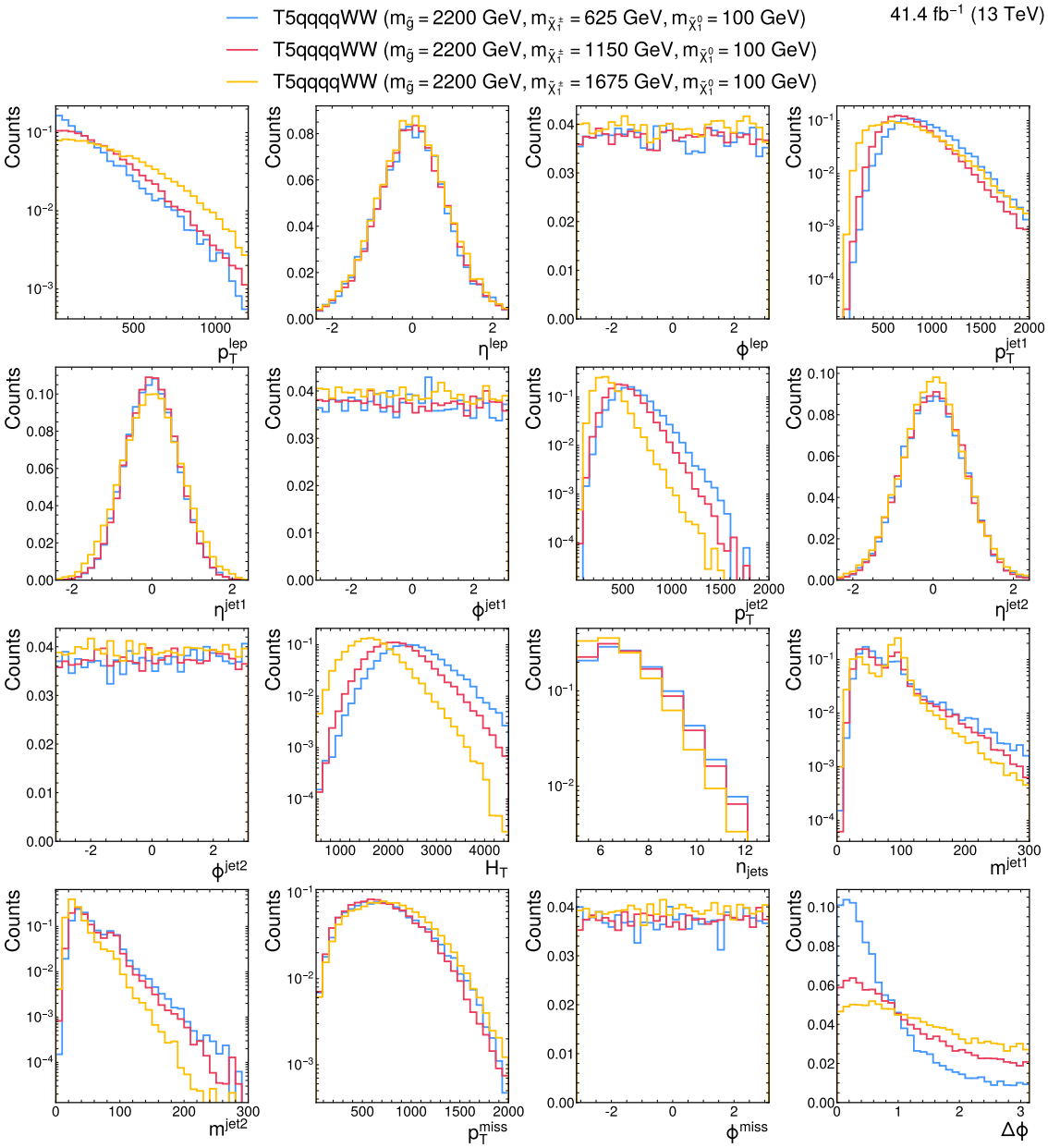


Figure 7.13: Distribution of the signal for different chargino masses in the signal region of the search after gluino pair production. The signal is shown for three different chargino masses.

Table 7.2: Hyperparameters for the CNF

Parameter	Value
`activation`	leaky_relu
`batch_size`	1024
`hidden_dim`	1024
`layers`	4
`lr`	0.001
`max_epochs`	5000
`weight_decay`	0.00005

1. A χ^2-distance as used in Ref. [194]:

$$\langle S^2 \rangle = \frac{1}{2} \sum_{i=1}^{n_{\text{bins}}} \frac{(h_i^{\text{MC}} - h_i^{\text{CNF}})^2}{h_i^{\text{MC}} + h_i^{\text{CNF}}}, \tag{7.7}$$

   which is calculated over a histogram $\boldsymbol{h}$ with n_{bins} bins.

2. Mean distance m between the empirical cumulative distribution functions (ecdf) $c(x)$:

$$m = \frac{1}{b-a} \int_a^b |c(x)_{\text{MC}} - c(x)_{\text{CNF}}| \, dx, \tag{7.8}$$

   where a and b determine the bounds of the support of the distribution. The ecdf is approximated by the cumulative sum over the bins of the histogram. Similarly, the integral in Eq. 7.8 is approximated by averaging the difference over all bins in the histogram.

These distances are computed and averaged over all marginal distributions on a testing dataset between the MC data at $\epsilon = 75\%$ mass mixing and data from the CNF at $\epsilon = 75\%$ mass mixing. For the results shown in the following, the model version used for the testing is chosen based on these two metrics.

7.2.3 Results

In Fig. 7.14 synthetic data drawn from the CNF by solving the associated ODE from 25% mass mixing to 75% mass mixing is depicted. Since the ODE can be solved in both directions, it is also possible to go into the reverse direction from $\epsilon = 75\%$ to $\epsilon = 25\%$ mass mixing with the same model. This is done by solving

the ODE backwards from $t = 1$ to $t = 0$. The results from this are shown in Fig. 7.15. Both figures suggest that the model is capturing the distributions and their correlations accurately. Some mismodeling is apparent in the invariant masses of the second-hardest jet and ϕ^{miss}. Concerning the mass distribution, it appears that it is difficult for the model to learn to convert the two-peaked structure to a single peak, which is especially visible when sampling in the reverse direction.

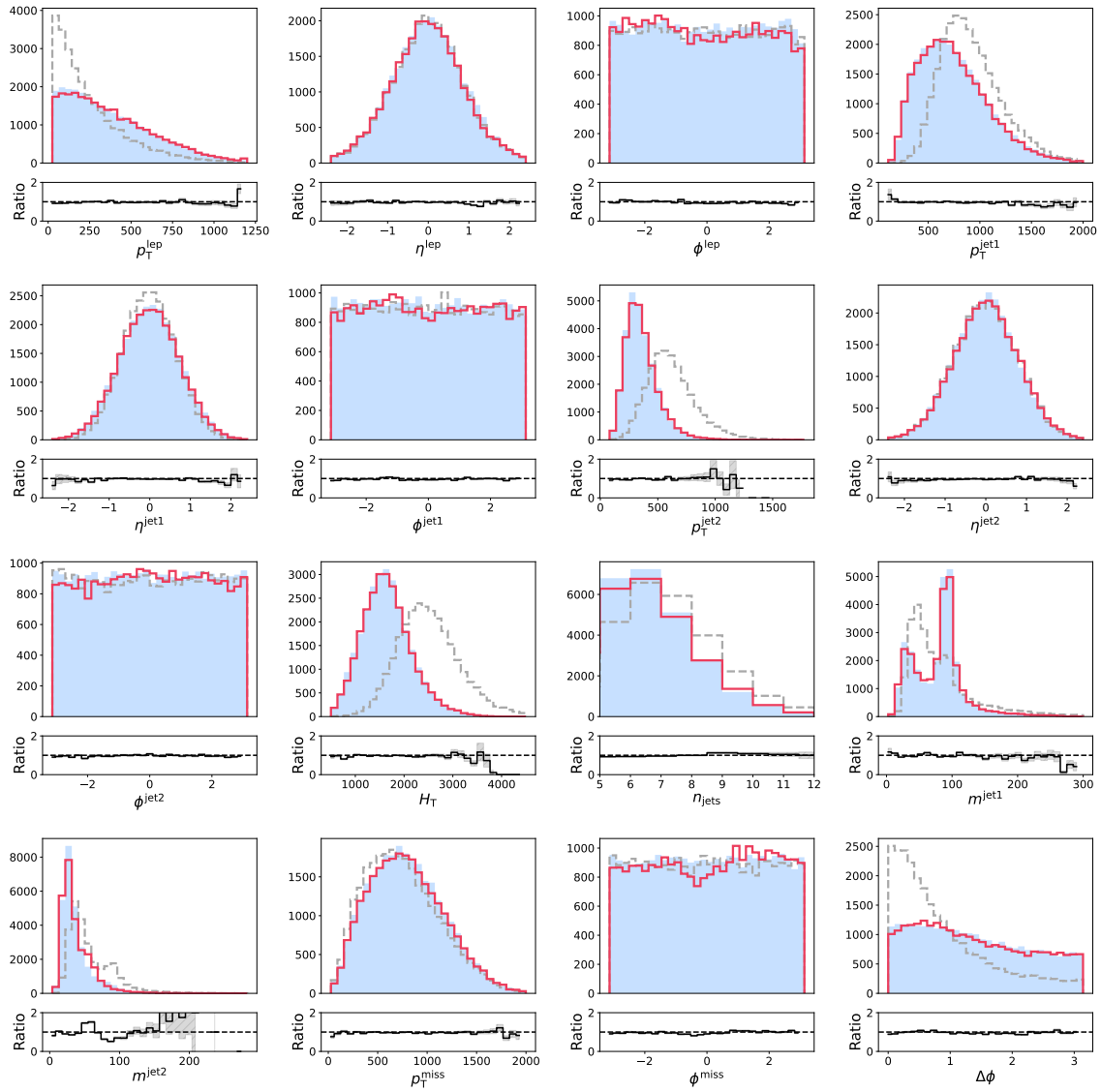


Figure 7.14: Results from modelling the chargino mass dependence from the 25% mass mixing to 75% mass mixing. The MC sample at $\epsilon = 75\%$ mass mixing is shown in blue, the synthetic samples drawn from the CNF are shown as a red line. The distribution at $\epsilon = 25\%$ mass mixing is depicted as a grey dashed line. Below the figure, the ratio between MC and the generated yields per bin is shown, together with an uncertainty band resulting from assuming Poisson uncertainty. The $\Delta\phi$ variable serves as a test to check whether the model captures correlations accurately. Except for the mass of the subleading jet and the azimuthal component of the missing momentum, no strong mismodelling is apparent in the other variables.

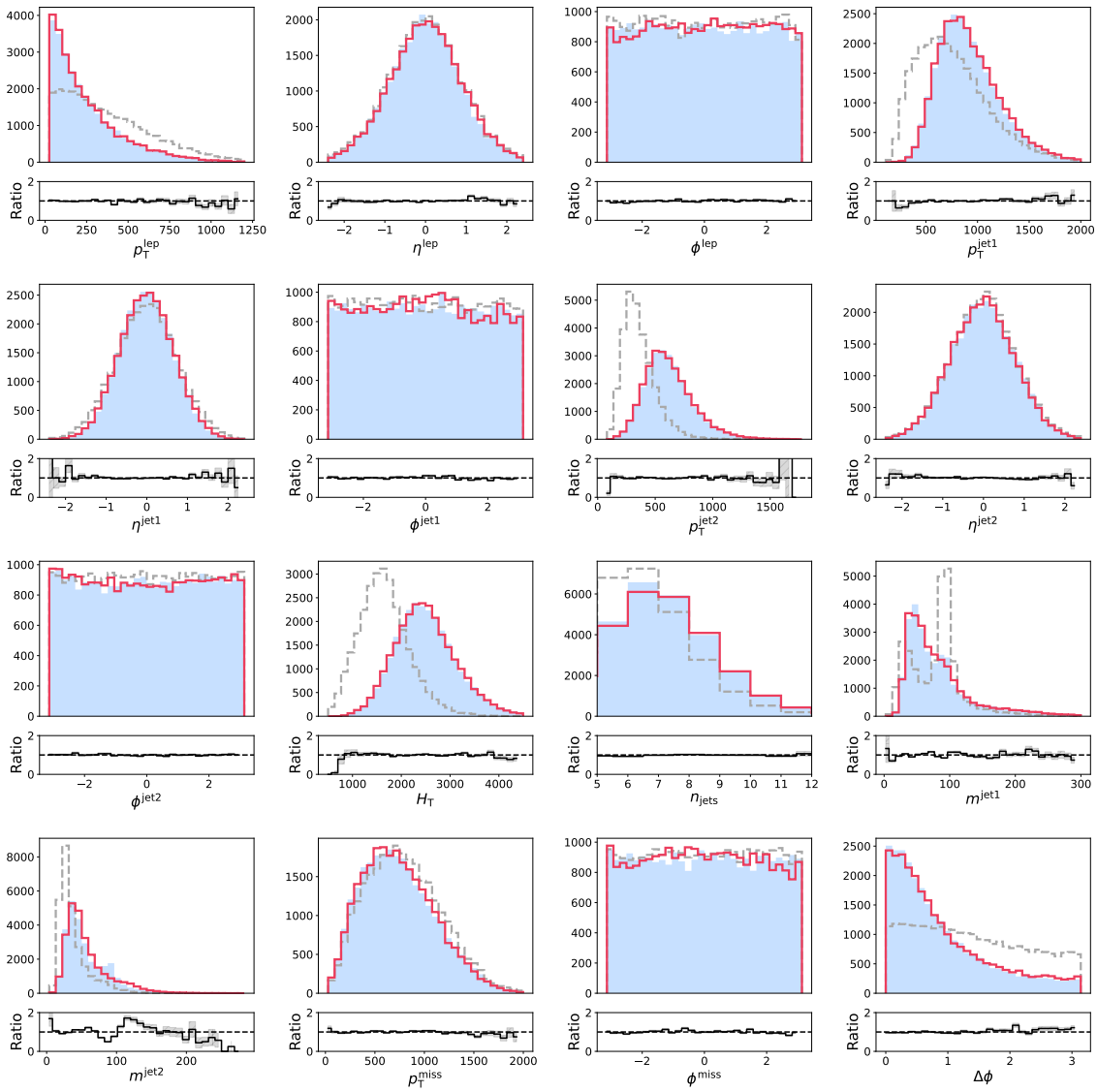


Figure 7.15: Results from modelling the signal distributions, starting from the 75% mass mixing to $\epsilon = 25\%$ mass mixing. The MC sample at $\epsilon = 25\%$ mass mixing is represented in blue, synthetic samples at $\epsilon = 25\%$ mass mixing are shown as a red line. The distribution at $\epsilon = 75\%$ mass mixing is depicted as a grey dashed line. Below the figure, the ratio between MC and the generated yields per bin is shown, together with an uncertainty band resulting from assuming Poisson uncertainty. The $\Delta\phi$ variable serves as a test to check whether the model captures correlations accurately. While the model captures most of the distributions well, in the reconstructed mass of the subleading jet, significant discrepancies are visible.

Since the metrics considered in Sec. 7.2.2 only concern marginal histograms, the binary classifier evaluation previously discussed in Sec. 4.8.2 is additionally used to determine whether the synthetic data agrees well with the MC simulation. The results of the binary classification distinguishing MC samples from synthetic samples at $\epsilon = 75\%$ mass mixing are shown in Fig. 7.16. The ROC suggests that the generative model accurately samples the density, since the curve is close to the diagonal and the AUC is close to 0.5, implying that the classifier is mostly random guessing.

7.2.4 Binary Classification Studies

The signal distributions shown in Fig. 7.13 vary significantly as function of the chargino mass, especially in $\Delta\phi$. As such, it is not directly clear whether a classifier that is trained only on samples from arbitrarily chosen mass points can generalise to other mass points. The worst thing that could happen is that the classifier does not recognise the signal coming from different chargino masses, similar as on the rotating Gaussian toy. For the training of this classifier, the same preprocessing as for the training of the CNF is used. Adding $\Delta\phi$ to the variables in the training did not improve the performance of the classification. Corresponding to the example of the rotating Gaussian, a baseline and interpolation training paradigm are used. However, an additional sanity test is introduced, where the training signal distributions contain the "true" parameter choice:

1. *Baseline approach*: for this training, the classifier is trained on the MC simulated signal samples at $\epsilon = 25\%$ and $\epsilon = 75\%$ mass mixing.

2. *Interpolation approach*: for this training, the classifier is trained on samples drawn from the CNF, which models the chargino mass dependence of the signal distribution from $\epsilon = 25\%$ mass mixing to $\epsilon = 75\%$ mass mixing. Note that the ODE associated with sampling is solved in both time directions, once starting from the sample at $\epsilon = 25\%$ mass mixing and once from the sample at $\epsilon = 75\%$ mass mixing. From both directions, the equal number of samples are used for the training.

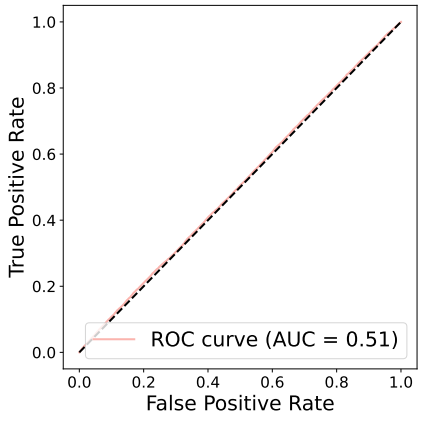


Figure 7.16: The ROC of the classifier distinguishing MC vs samples drawn from the CNF at 75% mass mixing. The ROC is close to the diagonal, suggesting that the classifier cannot distinguish the MC generated samples from the synthetic ones drawn from the CNF.

3. *Sanity test*: for this training, the MC simulated sample at $\epsilon = 50\%$ mass mixing is additionally included in the training.

After training, the three different classifiers are tested on a holdout set of background events and signal events at 50% mass mixing.

Additional Preprocessing Steps

To account for the absence of the ϕ^{miss} modulation in the signal samples, the signal events are reweighted. The weights are determined by calculating the ratio for every bin i over 100 bins from the weighted histogram of ϕ^{miss}:

$$w^i = \frac{n_{\text{b}}^i}{n_{\text{b}}} / \frac{n_{\text{s}}^i}{n_{\text{s}}}, \tag{7.9}$$

where $n_{s/b}^i$ is the yield per bin of the signal respective background histogram and $n_{s/b}$ is the total number of yields in the signal, respectively background histogram. After this, another reweighting is applied, such that the total number of expected events is the same as before the reweighting. The histograms of the reweighted signal distributions are compared to the background distributions in Fig. 7.17.

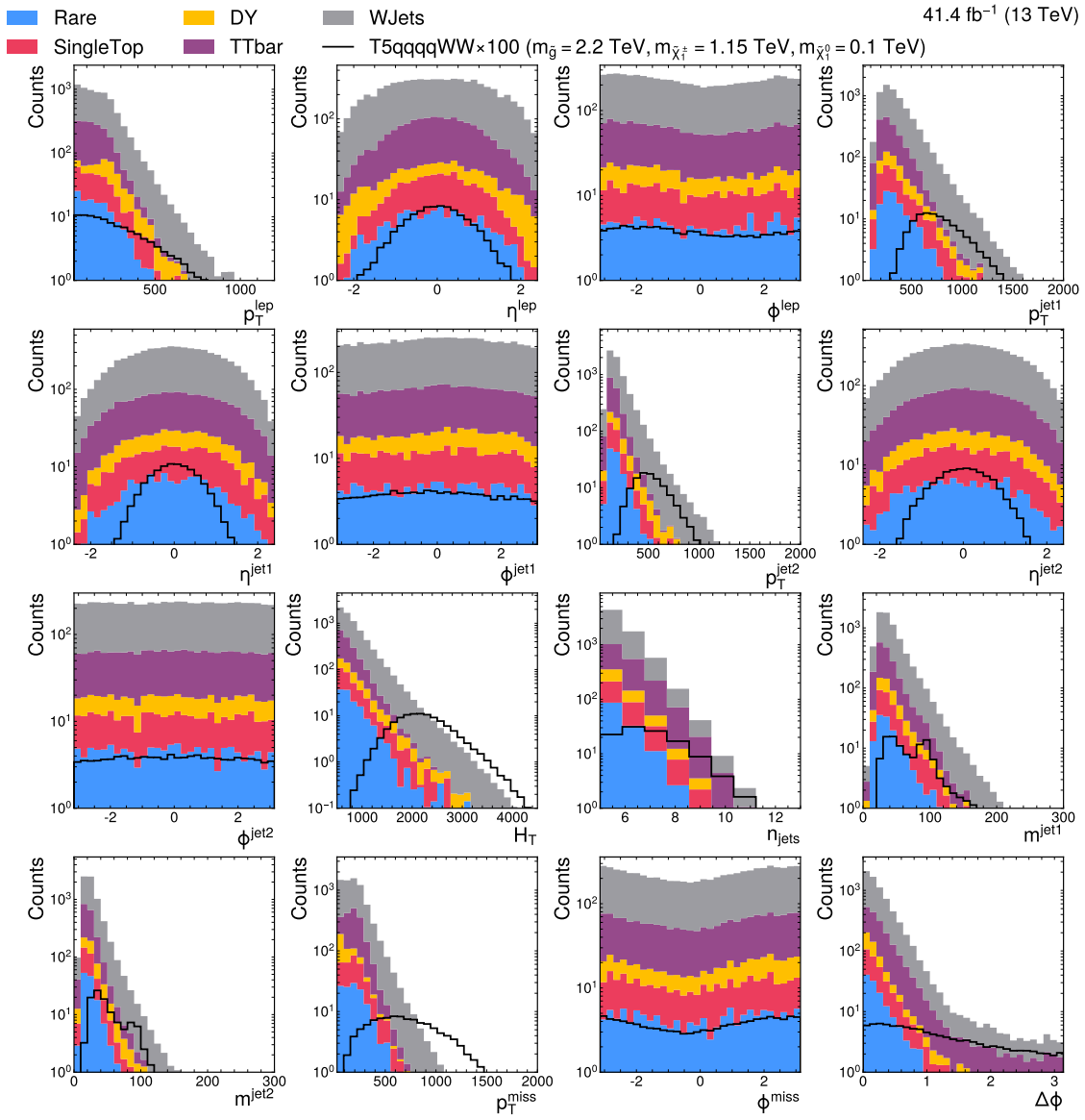


Figure 7.17: The histogram of the training variables after the ϕ^{miss} distribution has been reweighted to have the same shape as the background distribution.

Neural Network Architecture & Training

For the studies on this dataset, the same NN architecture is used as for the CNF, where the only difference is the number of output nodes. The hyperparameters are given in Table 7.3. Since there is a large imbalance in the sum of weighted signal events and weighted background events, the training is balanced as follows. Every training batch contains as many background events as signal events. For the signal and background class, the events are separately sampled with repetition, where the event weight is considered during sampling. This is done by sampling them from a multinomial distribution determined by the individual event weights. Focal loss with $\gamma = 1.75$ is used to obtain a high signal purity in bins with high $\hat{y}$.

Table 7.3: Hyperparameters of the binary classifier

Parameter	**Value**
`activation`	leaky_relu
`batch_size`	1024
`input_dim`	16
`hidden_dim`	64
`layers`	4
`lr`	0.001
γ	1.75
`max_epochs`	1000
`weight_decay`	0.00005

Evaluation

The evaluation procedure follows the approach in Ref. [53], where the discovery significance is introduced. To quantify the expected statistical significance of the analysis, the discovery statistic q_μ given in Eq. 7.10 is used, which is calculated from the likelihood ratio of the background-only hypothesis and the *composite* signal + background hypothesis:

$$q_0 = -2 \log \frac{\mathcal{L}(0, \hat{\hat{\tau}}|n) \cdot c(\hat{\hat{\tau}})}{\mathcal{L}(\hat{\mu}, \hat{\tau}|n)}, \tag{7.10}$$

where $\mathcal{L}$ is the likelihood, the symbol $\hat{}$ marks the maximum of the likelihood, the symbol $\hat{\hat{}}$ marks the maximum for a fixed value of the signal strength μ and $c(\tau)$ is a constraint on the background scale factor τ. The *background scale factor* τ is referred to as a *nuisance* parameter because it impacts the calculation of the likelihood, however it is not of direct interest, since the goal is to determine the signal strength. Note that the signal strength μ and the background scale factor τ are anti-correlated. This test statistic is computed from the histogram of the NN output $\hat{y} > 0.9$, with three bins.
The Asimov dataset is used to compute the expected statistical significance of discovering the signal, and as such the following relations hold:

$$n = b^{\mathrm{MC}} + s^{\mathrm{MC}}, \tag{7.11}$$

$$\hat{\mu} = 1, \tag{7.12}$$

$$\hat{\tau} = 1, \tag{7.13}$$

$$\tag{7.14}$$

where b^{MC} and s^{MC} are the expected number of events from the MC simulation. The likelihood is derived from Poisson statistics:

$$\mathcal{L}(\lambda = \mu s^{\mathrm{MC}} + \tau b^{\mathrm{MC}}|n) = \frac{e^{-\lambda}\lambda^n}{n!}. \tag{7.15}$$

The following constraint is used for the background scale factor τ:

$$c(\tau) = \exp\left(\frac{-(\tau b^{\mathrm{MC}} - b^{\mathrm{MC}})^2}{\sigma_b^2}\right), \tag{7.16}$$

where σ_b^2 is the variance associated with the expected number of background events. The background scale factor τ is to account for the uncertainty on the expected number of background events. In the denominator, the background scale factor is set to 1, since on Asimov the observed number of background events is equal to the ones expected from the MC simulation. The constraint given in Eq. 7.16 is needed,

as otherwise the profiled best fit value for the background scale factor for $\mu = 0$ is:

$$\hat{\hat{\tau}} = \frac{n}{b^{\mathrm{MC}}} = \frac{s^{\mathrm{MC}} + b^{\mathrm{MC}}}{b^{\mathrm{MC}}}. \tag{7.17}$$

However, there is prior knowledge about the uncertainty of the background yield estimation, given by σ_b:

$$\sigma_b = \sqrt{\sum w^2}, \tag{7.18}$$

where w are the weights associated to the background samples. If there is a large uncertainty on the number of background events, the constraint is small and τ can take the value given by Eq. 7.17. However, if the uncertainty is small, then the constraint becomes very negative for a large value of τ. It is most illustrative to think about the case where the uncertainty on the background approaches 0; any value of τ different from one leads to the likelihood being 0.

In this study, Equation 7.10 is evaluated over a histogram with three bins, denoted by $\boldsymbol{n} = (n_1, n_2, n_3)$, with $\hat{y} > 0.9$. The highest bin is constructed by scanning the bin width, until the relative uncertainty on the background is below 30%. The remaining two bins are chosen to be of equal width to cover the remaining distance to 0.9. Applying the logarithm in Eq. 7.10 and expanding all terms is summarised as:

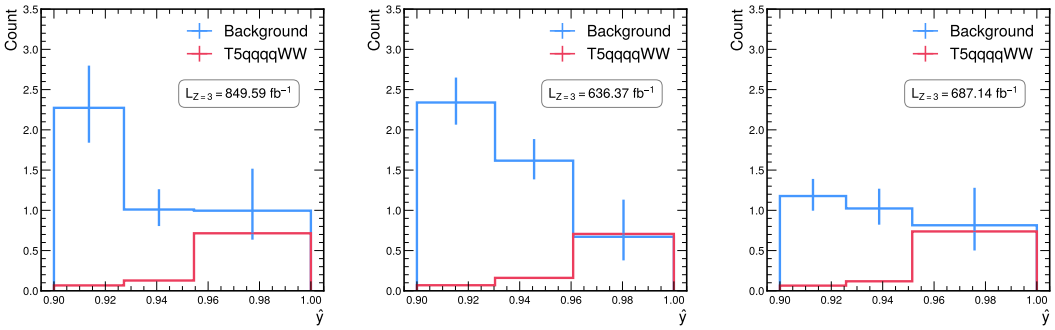
$$q_0 = -2 \max_{\tau} \sum_{i=1}^{3} n_i(\log(\tau b_i^{\mathrm{MC}}) - \log(s_i^{\mathrm{MC}} + b_i^{\mathrm{MC}})) - \tau b_i^{\mathrm{MC}} + (s_i^{\mathrm{MC}} + b_i^{\mathrm{MC}}) - c(\tau). \tag{7.19}$$

Since the statistical significance of discovery is computed, the null hypothesis $\mu = 0$ is tested. To calculate the statistical significance without using an excessive number of toys, Cowan et al. [53] proposed that in the large sample approximation, Wilks' [195] and Walds' [196] theorem can be used to obtain an analytical form of the statistical significance Z:

$$Z = \sqrt{q_0}. \tag{7.20}$$

This is used to estimate the expected (or median) statistical significance in the following.

Since the statistical significance is not a very insightful quantity, instead the required luminosity for a statistical significance of $Z = 3$ is quoted. This required luminosity is calculated by replacing s with αs, and b with αb in Eq. 7.15, and scanning α



(a) Baseline Approach (b) Interpolation Approach (c) Sanity Approach

Figure 7.18: Histogram of the NN output $\hat{y} > 0.9$ of the test set for all three approaches. The bin width is chosen on the validation set, such that the top-most bin has a background uncertainty $< 30\%$. The other two bins are chosen to be equidistant to 0.9.

until an expected statistical significance of $Z = 3$ is reached. While increasing the luminosity, the statistical background uncertainty[7] σ_b is reduced by $\sqrt{\alpha}$. Note that although this uncertainty follows from Poisson statistics, the choice of the number of generated MC events is a systematic uncertainty for the experiment.

7.2.5 Discussion

To have a fair comparison between the three approaches, the same validation set, comprising MC simulated data, and the signal distribution from 25% and 75% mass mixing, is used for all approaches. The classifier checkpoint with the maximal discovery significance on the validation set is chosen for the evaluation on the test set. The test set contains an independent signal sample at 50% mass mixing and an independent background sample. In Fig. 7.18 the results on the test set are shown for the three different approaches. The figure indicates that the background uncertainty becomes comparable to the expected number of signal events.
To estimate the uncertainty coming from the training, the classifier is retrained

[7]The scaling of the uncertainty on the expected background events derives from the fact that the relative Poisson uncertainty $\frac{\sigma_b}{b}$ scales with $\frac{1}{\sqrt{N}}$, where N is the number of MC samples. This assumes that if more data is recorded, also more MC samples will be available.

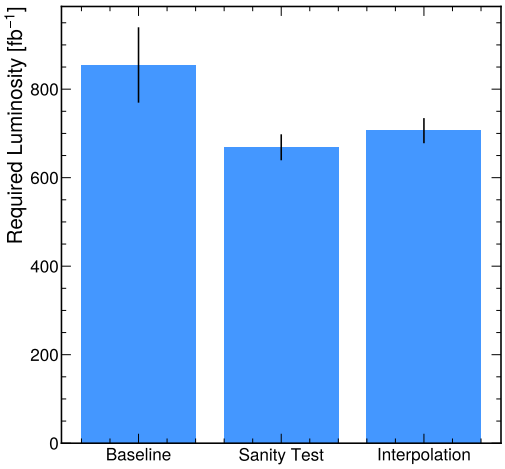


Figure 7.19: Comparison of the different approaches on the signal versus background classification. The required luminosity for an expected statistical significance of $Z = 3$ is shown for the three different approaches. For every category, the model is retrained ten times and mean, and the error on the mean is estimated over these ten runs. For the baseline approach, the uncertainty is significantly larger than for the other two approaches. The result from the interpolation approach is compatible with the performance of the Sanity test.

ten times with the same hyperparameters. In Figure 7.19 the mean together with the error on the mean is shown for all approaches. This reveals that the benefit of training on synthetic samples is significantly smaller than previously with the rotating Gaussian. Still, the interpolation approach performs significantly better than the baseline approach and yields a similar mean and error on the mean as the sanity test.

7.3 Conclusions from Modelling Parametrised Distributions

In this chapter, first the difference between conditioned discrete NFs and CNFs trained with CFM and OT-CFM was demonstrated. Then, potential improvements of a classifier recognising a parameterised signal distribution, for a choice of the

underlying parameter unseen during training, through the use of synthetic data were investigated. Two examples were studied:

- The rotating Gaussian; a toy example, where the signal distribution is strongly dependent on an a priori unknown parameter ϕ. A classifier trained on samples drawn from a CNF that models the dependence of the signal distribution performed significantly better than the baseline approach, which only trained on samples used for the training of the generative model. The baseline approach performed significantly worse and was agnostic to the signal distribution at another ϕ choice. However, it is difficult to attribute whether this gain in performance derives from the quality of the CNF or because the overlap of the signal at different values of ϕ is small.

- The search for gluino pair production; here, the signal distribution is dependent on a priori unknown parameters that need to be scanned. For three mass parameters in the signal, it is not feasible to scan all combinations. Thus, the chargino mass is set to the LSP mass m_{LSP} plus half ($\epsilon = 50\%$) of the mass difference Δm between the LSP and gluino: $m_{\tilde{\chi}_1^\pm} = m_{\text{LSP}} + \epsilon\Delta m$. An alternative to choosing one mass mixing is to train a CNF trained with OT-CFM, to transform the signal distribution at $\epsilon = 0.25\%$ to the distribution at $\epsilon = 0.75\%$. It was investigated, whether a classifier distinguishing Standard Model background from a signal arising from a SUSY signal benefits from training on synthetic data compared to training on Monte Carlo simulated data at $\epsilon \in [25\%, 75\%]$ only. To compare the two approaches, the classifier is evaluated on a testing sample containing signal samples from $\epsilon = 50\%$. Training on synthetic data, results in a higher statistical significance, or equivalently in a lower required luminosity for a median statistical significance of $Z = 3$, of the likelihood-ratio-based hypothesis test of the background-only versus the signal+background hypothesis. Note that both approaches were constructed such that the equal amount of MC generated data is used to allow a fair comparison.

This suggests that if a data distribution is strongly dependent on an underlying parameter, the training on synthetic data from a generative model strongly benefits the classification performance. However, in the case of HEP data in the search for

gluino pair production, the baseline approach does not perform as badly. Although the distributions for different values of the underlying parameters of the signal seem fairly different, the baseline model can still generalise to different parameter choices to a certain extent. Still, the baseline approach requires a significantly higher luminosity ($\sim 20 \pm 12\%$) than the interpolation approach to obtain a statistical significance $Z = 3$ of discovering the signal from a different chargino mass than seen during training. However, note that during these studies, no other systematics than the limited number of MC samples were considered during the computation of the statistical significance of the hypothesis test.

Conclusion

In this thesis, different applications of generative modelling to high-energy particle physics were investigated and discussed. Starting from the low-dimensional Two Moons dataset, three important conclusions can already be drawn from the performance of different algorithms.

1. The synthetic data should have at least as many degrees of freedom as real data, if the data is used as input for neural networks. Otherwise, the neural networks can make use of this information, which is not intended.

2. The possibility of mode collapse of generative adversarial networks should always be considered. While it is simple to recognise in two dimensions, this becomes significantly harder to spot in many dimensions.

3. Generative models should not be seen as black boxes that transform noise to synthetic data. A much more accurate interpretation is provided by them reshaping a distribution into another. Continuous normalising flows when trained with optimal transport-based conditional flow matching can enforce a path this reshaping should follow.

A more physics inspired toy example is the generation of jets, which was studied with the `JetNet` datasets. Although no intent lies in replacing PYTHIA data, its 90 or even 450 dimensions teach three other important lessons:

1. Discrete normalising flows perform excellent on low-dimensional data, but their capability to model correlations quickly deteriorates with higher dimensional data.

2. The computational complexity and the scaling of a point-cloud-based model with the number of points is crucial. To make the training feasible, the point-cloud-based model should scale linearly with the number of points.

3. Although attention is more common in natural language processing, it also proves to be suited for point clouds. To keep the computation complexity linear, a mean-field approximation of the information exchange in a point cloud is a powerful tool. However, recognising attention as a normalised weighted sum, highlights the importance of incorporating the number of constituents into the aggregation.

Then, the first more practical problem was studied with the CaloChallenge data. Surprisingly, a model designed for the generation of jets, also performs well at calorimeter simulation. With minor modifications, the previously found mean-field aggregation can also be used for conditional generation. The main two conclusions that should be drawn are:

1. The representation of calorimeter energy deposits as point clouds is not only more realistic than a grid-based representation. But it also allows arbitrary detector geometries and is more memory efficient if the detector occupancy is low.

2. Point-cloud-based model can generalise well, which was shown when a model trained on dataset 2 was successfully transferred to dataset 3 of the CaloChallenge. This motivates a more efficient training, e.g. where the model is trained on different clustering stages with different granularity.

Finally, it was studied whether a signal versus background classifier can benefit from training with synthetic data. This is especially viable if the signal distribution is strongly dependent on an underlying, a priori unknown, parameter, and there is not enough Monte Carlo simulated training data available for all combinations of the underlying parameters. From these studies, the following two conclusions can be drawn:

1. The transport plan with the lowest cost might not be the most intuitive one. While it is possible to enforce a certain path by providing samples on the desired trajectory, this is expected to become more difficult in a higher dimension.

2. In the search for gluino pair production, a classifier trained on signal samples corresponding to arbitrarily chosen mass points can generalise to an extent. However, training on samples of a generative model that interpolates the dependence of the signal distribution on the underlying parameter, improves the sensitivity of the statistical hypothesis test of the signal+background hypothesis versus the background-only hypothesis.

From a more technical view, the following, highly subjective view of different approaches for generative modelling can be given:

- *Generative adversarial models*: while they allow for much freedom in designing the network architecture, their unstable training and mode collapse makes them my least favourite model class. Although there were multiple solutions proposed to cure this aspect, from my personal experience, they rarely completely resolved these issues. However, once such a model actually converges, it is almost unbeatable in terms of quality and inference speed when the potential presence of mode collapse is disregarded.

- *Discrete normalising flows*: their performance is excellent for a low number of dimensions. In these studies, normalising flows that use rational quadratic spline coupling layers performed best. The only problem is, once they do not work as desired, it is difficult to improve their performance.

- *Continuous normalising flows*: while training them without flow matching is slow, they became my favourite model to work with. They allow for much flexibility as the generative adversarial networks when designing the neural network architecture, but they are also stable to train. They seem to combine the advantages of both previously mentioned models at the cost of a significantly slower generation time.

Working with these different models and exploring these various aspects has been an unimaginably interesting journey, and I strongly motivate further investigations in this exciting field. There are many interesting pathways to make use of the versatility of generative models. I think that the largest expected gain comes from a point-cloud-based generative model for detector simulation.

APPENDIX A

Two Moons: Multiclass Classifier Test

To rule out that the classifier is unable to recognise the mode collapse present in the data drawn from the GANs, here only data from the AE, NSGAN and RQS are compared. However, the test still does not recognise the mode collapse present in the NSGAN because it assigns the same probability to real data to come from the NSGAN and RQS NF.

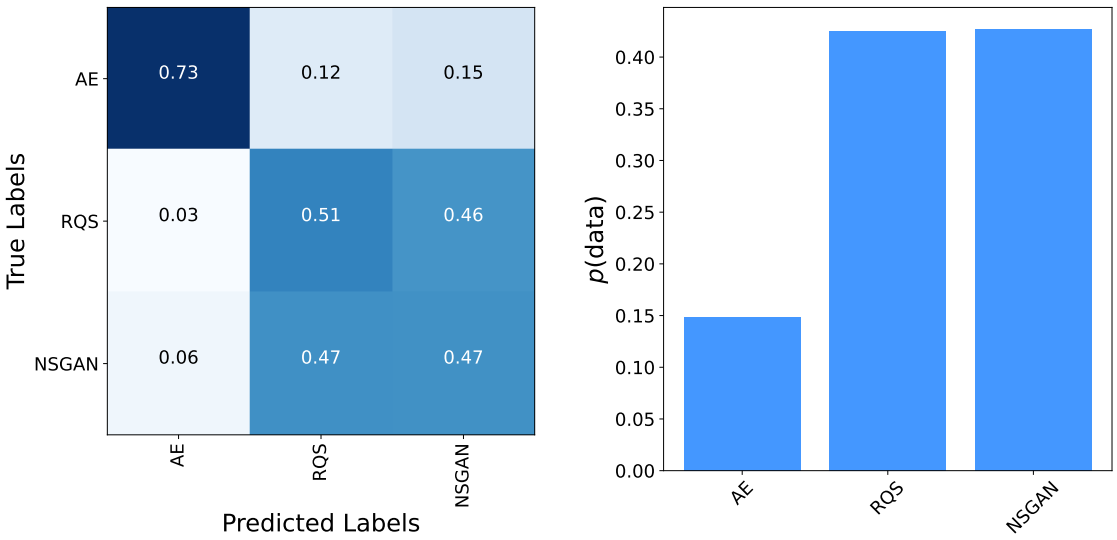


Figure A.1: (left) Confusion matrix of the multi-classifier, showing that it can indeed recognize most of the classes. (right) The average probability the multi-classifier assigns to real data to come from the different models under test.

APPENDIX B

Additional JetNet Results

B.1 JetNet150: No Box-Cox Preprocessing for NFs

When the $p_{\mathrm{T}}^{\mathrm{rel}}$ is only standard scaled in the pre-processing step, the NF does not model the tail of the distribution, which is shown in Fig. B.1.

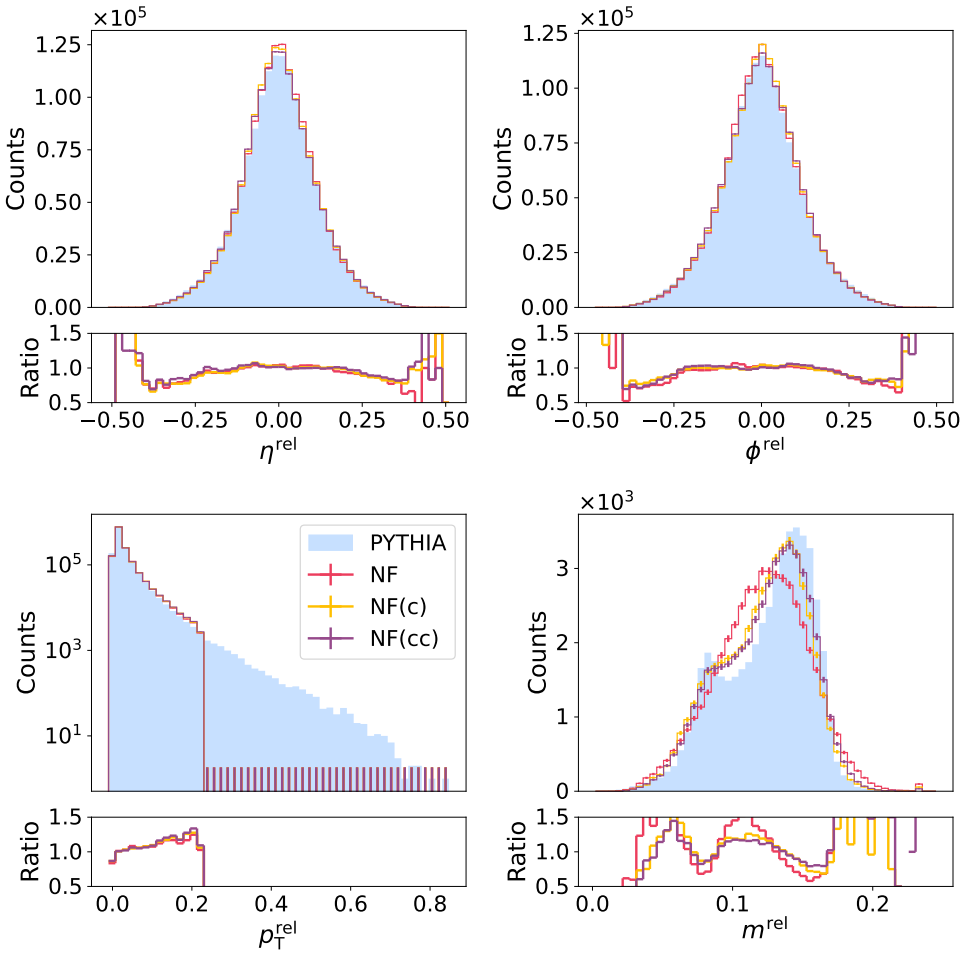


Figure B.1: Comparison of synthetic data drawn from the NF-based models to PYTHIA samples (blue), below each plot the ratio of real divided by synthetic data is shown. Going from left to right and top to bottom, the first three histograms depict the relative pseudorapidity, azimuthal angle and transverse momentum of all particles drawn from the model. The first two figures might indicate that the VNF (red) is sampling the underlying data distribution accurately. Even worse problems become evident when considering the histogram of the relative invariant mass, which is calculated from all particles in the jet, shown on the bottom right. It reveals that the VNF is unable to capture the correlations between the particles correctly. Introducing conditioning (yellow) improves the mass modelling, adding a mass constraint enhances it further (violet).

B.2 `JetNet150`: No Box-Cox Preprocessing for Equivariant NFs

Similarly, for the equivariant NF models, the tail of the $p_{\mathrm{T}}^{\mathrm{rel}}$ is also not modelled if only standard scaling is used for preprocessing.

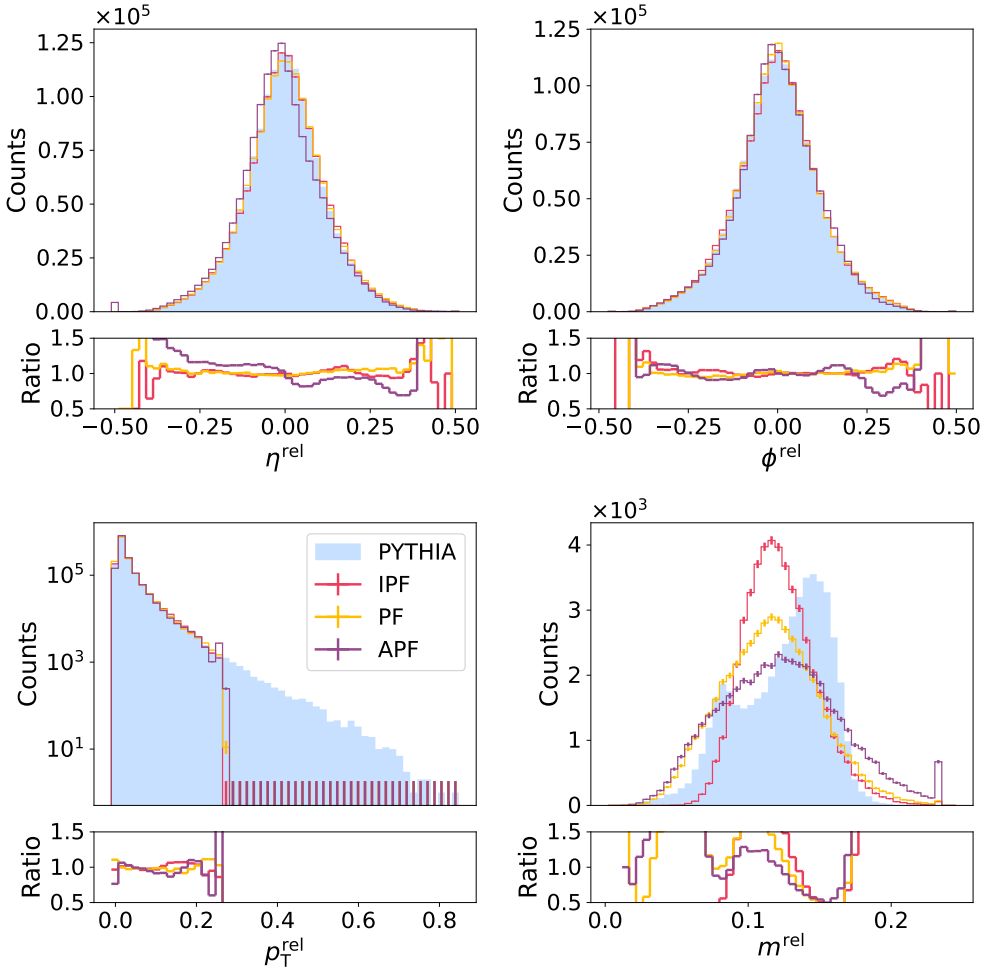


Figure B.2: Comparison of synthetic data to PYTHIA samples (blue), below each plot the ratio of real divided by synthetic data is shown. Going from left to right and top to bottom, the first three histograms depict the relative pseudorapidity, azimuthal angle and transverse momentum of all particles drawn from the model. The figure on the bottom right depicts the invariant mass distribution calculated from the particles in every jet. Similar to the VNF, the IPF (red) and the PF (yellow) excel at modelling the marginal features, except for the tail of the $p_{\mathrm{T}}^{\mathrm{rel}}$ distribution. However, the mass distribution reveals that the correlations between the plots are not modelled correctly. For the APF (violet), the modelling of the marginal features deteriorates.

B.3 `JetNet150`: Box-Cox Preprocessing for GANs

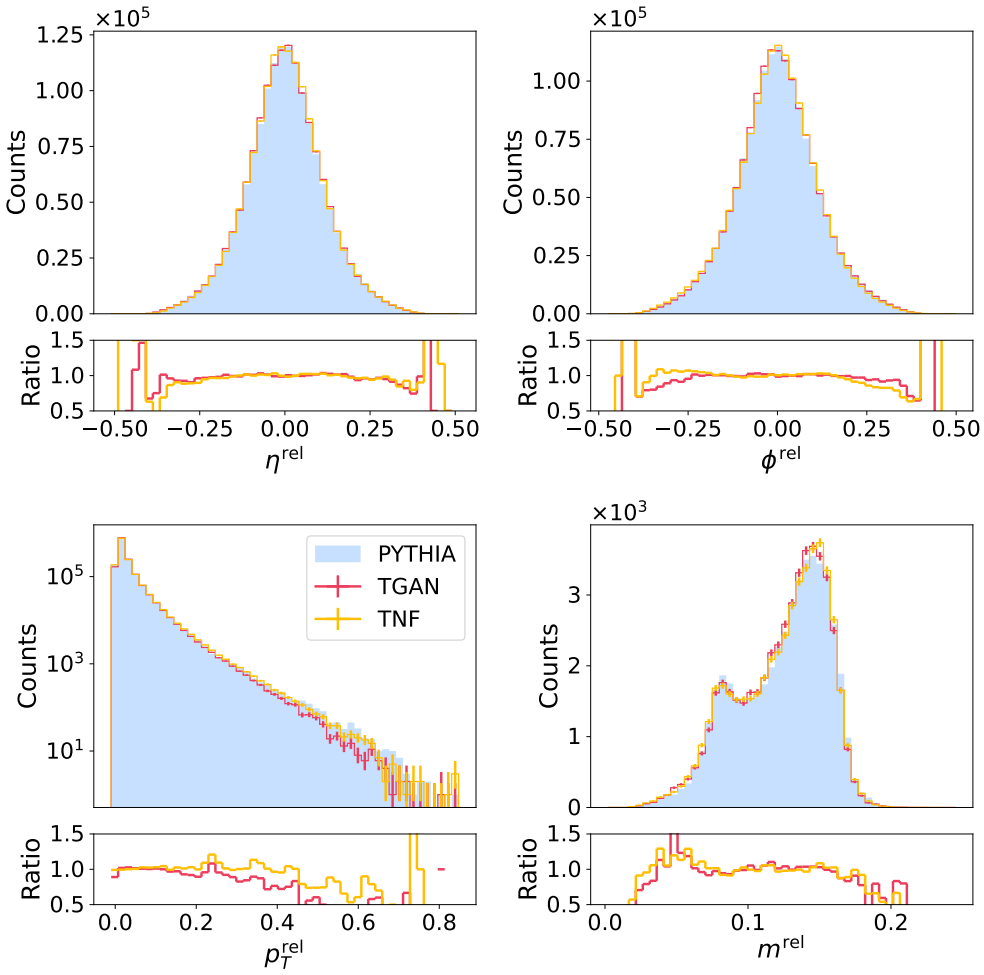


Figure B.3: Comparison of synthetic data drawn from the TNF (red) and TGAN (violet) to PYTHIA samples (blue). Below each plot, the ratio of real divided by synthetic data is shown. Going from left to right and top to bottom, the first three histograms depict the relative pseudorapidity, azimuthal angle and transverse momentum of all particles drawn from the model. For this training, Box-Cox preprocessing was used on $p_{\mathrm{T}}^{\mathrm{rel}}$. The figure on the bottom right depicts the invariant mass distribution calculated from the particles in every jet.

B.4 Quantitative Comparison without/with Box-Cox Preprocessing

Here, the quantitative results on the `JetNet30` top quark dataset are given, when no Box-Cox preprocessing is used for the preprocessing of the $p_{\mathrm{T}}^{\mathrm{rel}}$ coordinate for the normalising flow-based models, respectively if Box-Cox preprocessing is used for the GAN-based models.

Table B.1: Comparison of the proposed models on the top-quark `JetNet30` dataset. For the NFs only standard scaling is used for preprocessing, for the GAN-based models, box-cox scaling on $p_{\mathrm{T}}^{\mathrm{rel}}$ is used. Bold font is used to highlight the best-performing model, multiple models are marked at the same time if the metrics are compatible within 1σ. The in-sample distance between the training and testing sample (IN) is also given for reference, as well as the (at the time of the study) state-of-the-art model MPGAN [156]. The metrics discussed in Sec. 4.8 are given together with the number of parameters and the time needed to generate one jet.

Jet Class	Model	$W_1^M(\times 10^3)$	$W_1^P(\times 10^3)$	$W_1^{EFP}(\times 10^5)$	$KPD(\times 10^4)$	$FPD(\times 10^4)$	Time [μs]	#parameters
Top Quark	PF	11.3 ± 0.2	2.42 ± 0.03	31.2 ± 0.2	3k $\pm$100k	18k $\pm$1k	182.8	2.2M
	IPF	10.8 ± 0.2	2.74 ± 0.05	43.8 ± 0.4	8k $\pm$70k	24k $\pm$1k	182.3	2.2M
	APF	6.50 ± 0.08	2.29 ± 0.03	34.5 ± 0.3	1k $\pm$20k	15k $\pm$1k	182.4	2.4M
	NF	3.77 ± 0.09	1.65 ± 0.02	10.1 ± 0.1	10 ± 2	146 ± 3	44.5	1.9M
	NF(c)	3.7 ± 0.1	1.17 ± 0.03	8.20 ± 0.09	6.4 ± 0.6	107 ± 2	103.9	3.7M
	NF(cc)	3.0 ± 0.1	1.08 ± 0.02	5.04 ± 0.07	2.7 ± 0.2	37.6 ± 0.5	103.8	3.7M
	TNF (box-cox)	$\mathbf{0.6 \pm 0.1}$	$\mathbf{0.51 \pm 0.02}$	1.6 ± 0.1	0.4 ± 0.2	$\mathbf{5.9 \pm 0.2}$	3.8	7.1 M
	TGAN (box-cox)	0.9 ± 0.1	0.77 ± 0.01	1.9 ± 0.1	0.3 ± 0.3	9.8 ± 0.7	**0.1**	**80** k
	MPGAN	$\mathbf{0.54 \pm 0.07}$	0.70 ± 0.02	$\mathbf{0.84 \pm 0.07}$	$\mathbf{0.07 \pm 0.08}$	17.4 ± 0.8	35.7	$716k$
	IN	0.24 ± 0.07	0.09 ± 0.01	0.48 ± 0.08	-0.12 ± 0.06	0.6 ± 0.3	-	-

B.5 `JetNet150`: Results with Box-Cox Preprocessing

Here, the quantitative results on the `JetNet150` top quark dataset are given, when Box-Cox preprocessing is used for the preprocessing of the $p_{\mathrm{T}}^{\mathrm{rel}}$ coordinate.

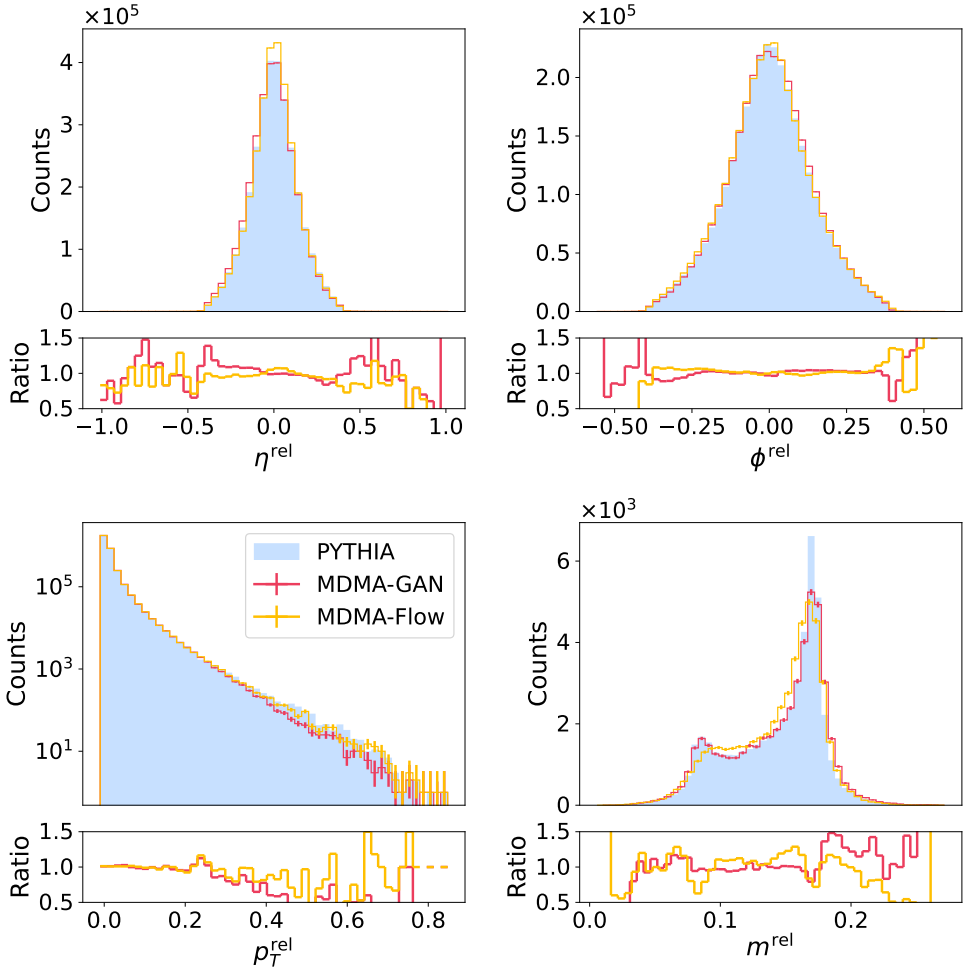


Figure B.4: Comparison of synthetic data drawn from the MDMA-GAN (red) and MDMA-Flow (yellow) when Box-Cox preprocessing is used on $p_{\mathrm{T}}^{\mathrm{rel}}$ to PYTHIA samples (blue), below each plot the ratio of real divided by synthetic data is shown. Going from left to right and top to bottom, the first three histograms depict the relative pseudorapidity, azimuthal angle and transverse momentum of all particles drawn from the model. The figure on the bottom right depicts the invariant mass distribution calculated from the particles in every jet.

Table B.2: Quantitative results on the `JetNet150`. The discussed models are compared to the SotA unconditional generative models (EPiC-GAN and EPiC-FM). Interestingly, when the Box-Cox preprocessing on $p_{\mathrm{T}}^{\mathrm{rel}}$ is used, both models perform slightly worse than with standard scaling.

Jet Class	Model	$W_1^M(\times 10^3)$	$W_1^P(\times 10^3)$	$W_1^{EFP}(\times 10^5)$	$KPD(\times 10^4)$	$FPD(\times 10^4)$	Time [μs]	#parameters
Top Quark	MDMA-GAN	1.2 ± 0.2	0.35 ± 0.01	4.2 ± 0.1	1.4 ± 0.9	19 ± 1	**7.9**	**111555**
	MDMA-Flow	0.8 ± 0.1	0.24 ± 0.02	$\mathbf{2.94 \pm 0.06}$	$\mathbf{-0.0 \pm 0.4}$	$\mathbf{1.9 \pm 0.3}$	4634.5	2345091
	EPiC-GAN	$\mathbf{0.66 \pm 0.09}$	0.591 ± 0.004	3.0 ± 0.1	2.3 ± 0.6	19 ± 1	62.5	424850
	EPiC-FM	1.48 ± 0.08	$\mathbf{0.057 \pm 0.009}$	4.0 ± 0.1	$\mathbf{-0.0 \pm 0.1}$	2.1 ± 0.3	6075.0	561330
	IN	0.3 ± 0.1	0.12 ± 0.02	1.1 ± 0.1	-0.1 ± 0.1	0.7 ± 0.3	0.0	0

B.6 `JetNet150`: Results with OT-CFM

Here, the results of the OT-CFM training of the CNF on the `JetNet150` dataset are given.

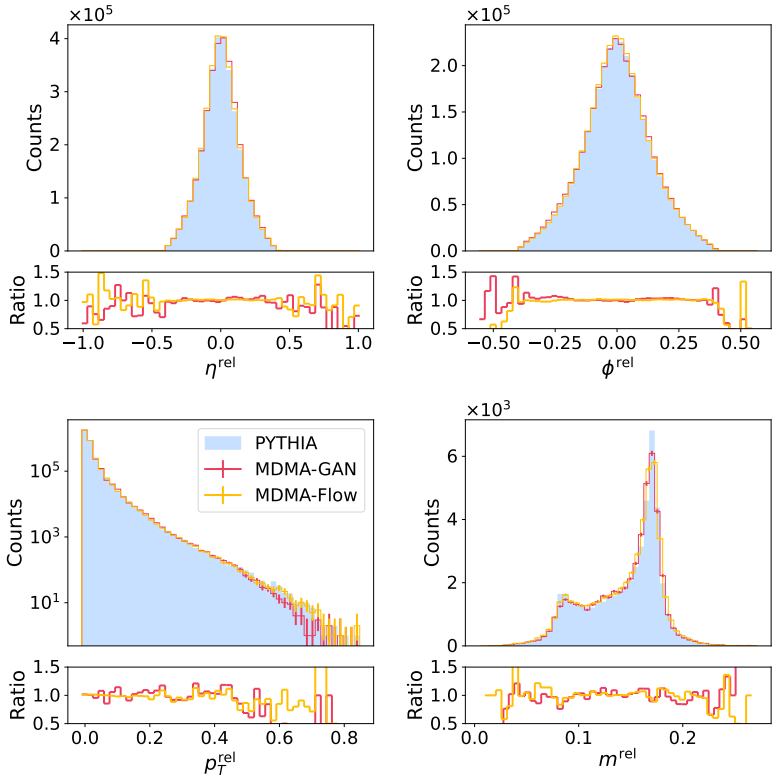


Figure B.5: Comparison of synthetic data drawn from the MDMA-Flow (yellow) when OT-CFM is used to ground truth samples (blue), below each plot the ratio of real divided by synthetic data is shown. Going from left to right and top to bottom, the first three histograms depict the relative pseudorapidity, azimuthal angle and transverse momentum of all particles drawn from the model. The figure on the bottom right depicts the invariant mass distribution calculated from the particles in every jet.

Table B.3: Quantitative results on the `JetNet150` when the OT-CFM training is used. The discussed models are compared to the SotA unconditional generative models (EPiC-GAN and EPiC-FM).

Jet Class	Model	$W_1^M(\times 10^3)$	$W_1^P(\times 10^3)$	$W_1^{EFP}(\times 10^5)$	$KPD(\times 10^4)$	$FPD(\times 10^4)$	Time $[\mu s]$	#parameters
Top Quark	MDMA-GAN	1.0 ± 0.1	0.287 ± 0.009	$\mathbf{1.78 \pm 0.07}$	$\mathbf{-0.0 \pm 0.2}$	3.2 ± 0.5	**6.5**	**111555**
	MDMA-Flow (OT)	$\mathbf{0.53 \pm 0.04}$	0.20 ± 0.02	$\mathbf{1.91 \pm 0.09}$	$\mathbf{0.0 \pm 0.1}$	$\mathbf{1.3 \pm 0.4}$	3455.2	3119491
	MDMA-Flow	0.6 ± 0.1	$\mathbf{0.045 \pm 0.006}$	$\mathbf{1.93 \pm 0.06}$	$\mathbf{-0.1 \pm 0.3}$	$\mathbf{1.4 \pm 0.3}$	6880.4	3119491
	EPiC-GAN	0.7 ± 0.1	1.10 ± 0.02	2.86 ± 0.08	2.3 ± 0.6	19 ± 1	62.5	424850
	EPiC-FM	1.4 ± 0.1	$\mathbf{0.09 \pm 0.02}$	3.6 ± 0.1	$\mathbf{-0.0 \pm 0.1}$	2.1 ± 0.3	6075.0	561330
	IN	0.27 ± 0.09	0.13 ± 0.02	1.0 ± 0.1	-0.1 ± 0.1	0.7 ± 0.3	0.0	0

APPENDIX C

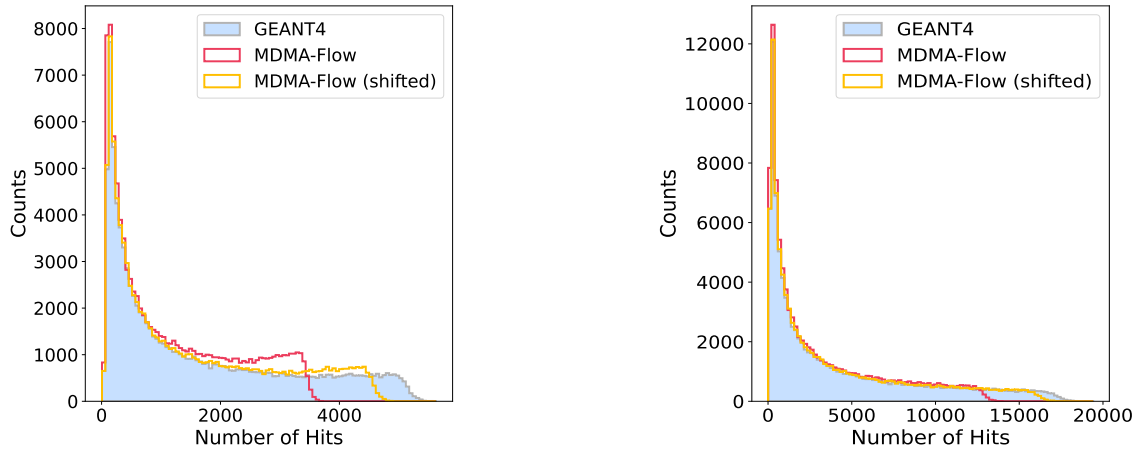
Additional Results: CaloChallenge

C.1 Shifting Hits for MDMA-Flow on Dataset 3

Here, the results of the Hit-shifting are shown for the MDMA-Flow. Similarly to the MDMA-GAN, the distributions agree significantly better if the post-processing is used, where if multiple hits are assigned to the same cell, the softer hits are moved to neighbouring cells.

C.2 Predicting Number of Hits

Here, the number of hits is deduced from the incoming energy E_{inc}, by fitting the dependency with a polynomial of degree 5, as depicted in Fig C.2. The results of the generative model when sampling the number of hits as a function of E are shown below, and demonstrate that the performance slightly decreases, which becomes especially apparent in the energy weight histograms in Fig. C.4. But note, that fitting the dependency with a polynomial fit is a basic approach. The distribution of the number of hits does not change significantly, as shown in Fig. C.5



(a) Dataset 2

(b) Dataset 3

Figure C.1: Histograms of the number of hits created by the GEANT4 (blue) simulation and after the voxelisation of the samples drawn from the MDMA-GAN (red), together with the effect of the post-processing (yellow). (left) On dataset 2 there are significantly more double hits if there are many cells hit in a shower. The post-processing effectively lowers the discrepancy between the GEANT4 simulation and MDMA-GAN (right) On dataset 3, the number of double hits is not as high as on dataset 2. The effect of post-processing again improves the agreement between the model and GEANT4 distribution significantly.

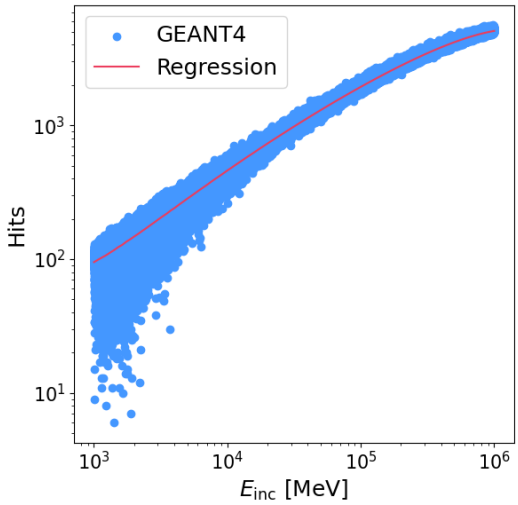


Figure C.2: Fit of the number of hits as a function of the energy of the incoming electron E_{inc}.

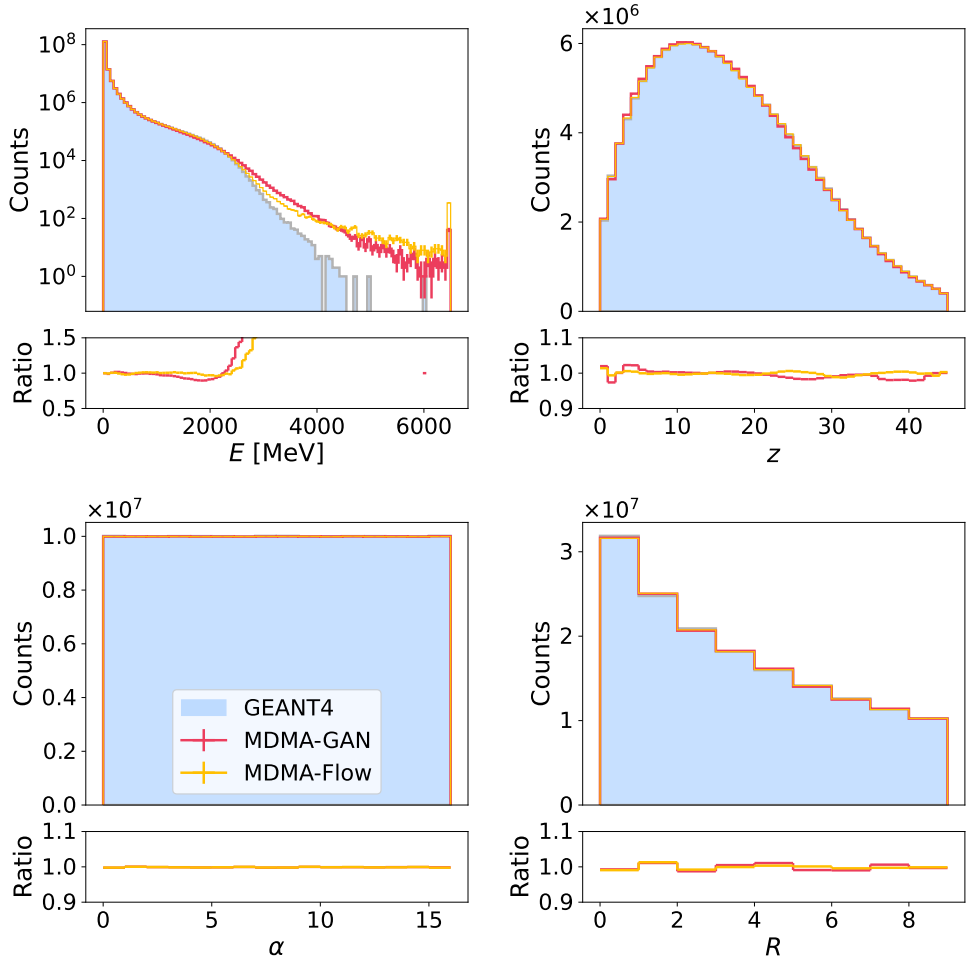


Figure C.3: Marginal distributions when deducing the number of hits from the energy of the incoming electron for the generative models (red, yellow) are compared to GEANT4 (blue). From top left to bottom right, the energy of the generated hits, the angular-, forward-, and radial coordinates are shown.

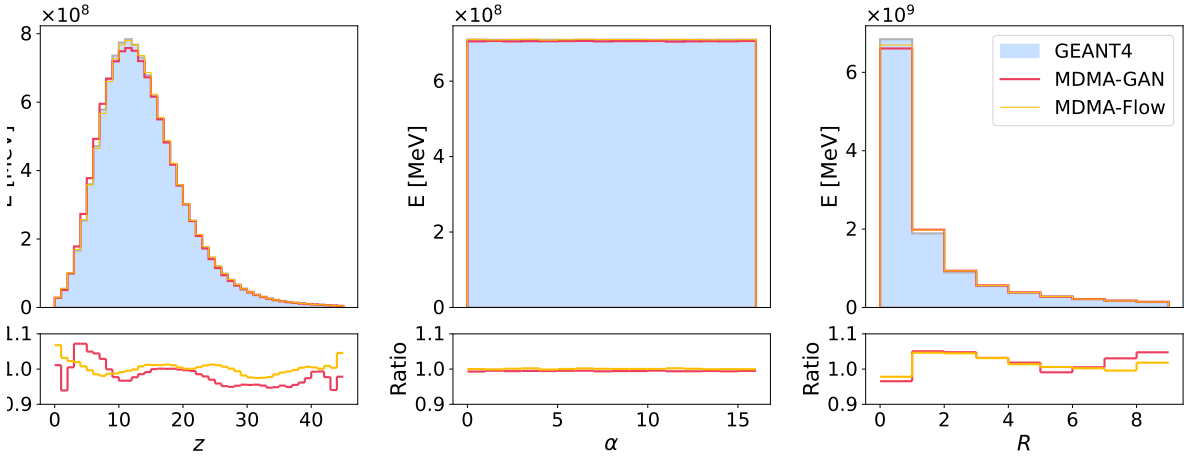


Figure C.4: Energy-weighted coordinate distributions when deducing the number of hits from the energy of the incoming electron for the generative models (red, yellow) are compared to GEANT4 (blue). From top left to bottom right, the angular-, forward-, and radial coordinates are shown.

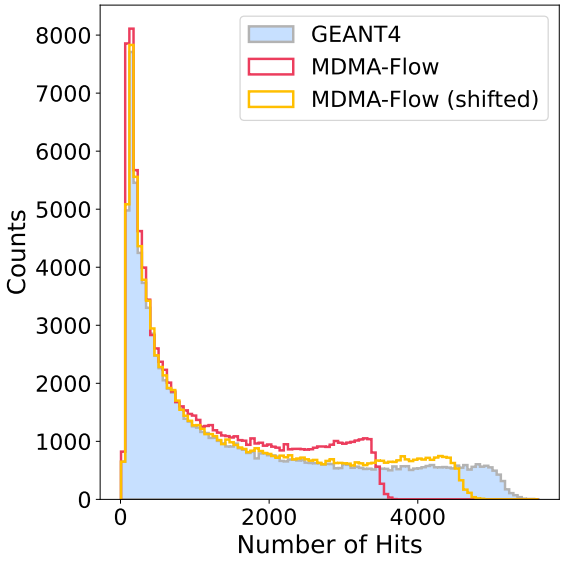


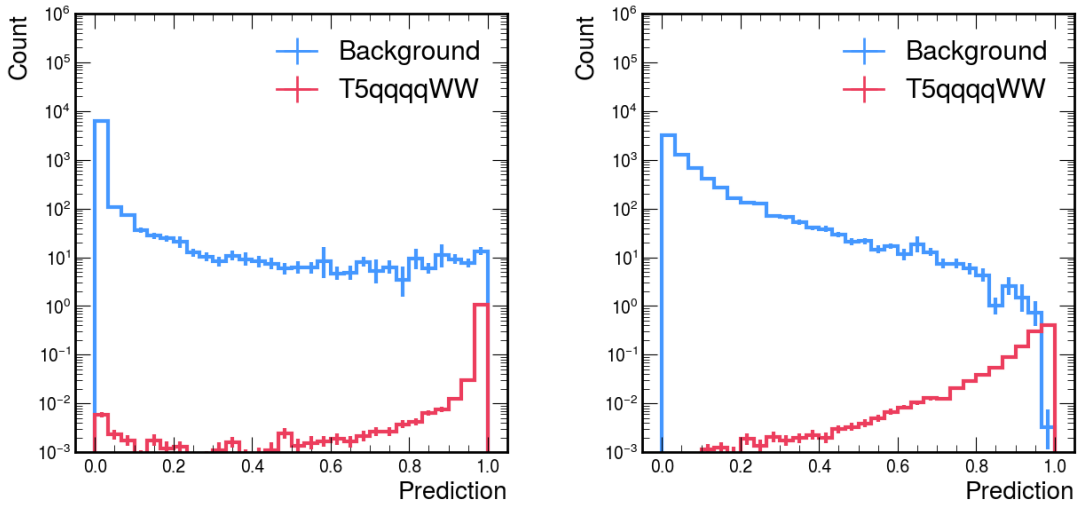
Figure C.5: Distribution of the number of hits of the voxelised point cloud when the number of hits is determined as a function of the energy. The MDMA-Flow still maps multiple hits to the same coordinates (red), shifting hits to neighbouring cells improves the agreement with GEANT4 data significantly.(red, yellow) are compared to GEANT4 (blue). From top left to bottom right, the angular-, forward-, and radial coordinates are shown.

APPENDIX D

Additional Results: Parametrised Distributions

D.1 Binary Cross Entropy vs Focal Loss

In Fig. D.1 the results from the classification when training with BCE are compared to the ones with Focal loss. These plots were made on the validation set, i.e. with $\epsilon \in [25, 75]\%$ mass mixing. Note that for the BCE classifier, the signal sharply peaks around 1, and the background peaks sharply around 0. However, also note the model trained with focal loss, makes significantly less confident incorrect prediction, which is crucial for the sensitivity to the signal process in the statistical inference.



(a) BCE

(b) Focal loss

Figure D.1: Comparison of a classifier trained with BCE (left) vs results of a classifier trained with Focal loss (right). Since a region for the statistical inference should be a signal pure as possible

Bibliography

[1] TASSO Collaboration. Evidence for planar events in e+ e- annihilation at high-energies. doi:10.1016/0370-2693(79)90830-X.

[2] UA1 Collaboration. Experimental Observation of Isolated Large Transverse Energy Electrons with Associated Missing Energy at $\sqrt{s} = 540$ GeV. doi:10.1016/0370-2693(83)91177-2.

[3] UA1 Collaboration. Experimental observation of lepton pairs of invariant mass around 95 gev at the CERN SPS collider. doi:10.1016/0370-2693(83)90188-0.

[4] E. Fermi. An attempt of a theory of beta radiation. 1. doi:10.1007/BF01351864.

[5] T. Kajita. Discovery of neutrino oscillations. doi:10.1088/0034-4885/69/6/R01.

[6] A. B. McDonald. Evidence for neutrino oscillations. i. solar and reactor neutrinos. doi:10.1016/j.nuclphysa.2005.02.102.

[7] R. L. Workman and Others. Review of particle physics. doi:10.1093/ptep/ptac097.

[8] Enrico Franco and Vittorio Lubicz. Quark mass renormalization in the MS-bar and RI schemes up to the NNLO order. doi:10.1016/S0550-3213(98)00438-6.

[9] CDF Collaboration. High-precision measurement of the W boson mass with the cdf ii detector. doi:10.1126/science.abk1781.

[10] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. The anti-k_t jet clustering algorithm. doi:10.1088/1126-6708/2008/04/063.

[11] S. L. Glashow. Partial symmetries of weak interactions. doi:10.1016/0029-5582(61)90469-2.

[12] Abdus Salam. Weak and electromagnetic interactions. doi:10.1142/9789812795915_0034.

[13] Steven Weinberg. A model of leptons. doi:10.1103/PhysRevLett.19.1264.

[14] F. Englert and R. Brout. Broken symmetry and the mass of gauge vector mesons. doi:10.1103/PhysRevLett.13.321.

[15] Peter W. Higgs. Broken symmetries and the masses of gauge bosons. doi:10.1103/PhysRevLett.13.508.

[16] Wim de Boer. The discovery of the higgs boson with the CMS detector and its implications for supersymmetry and cosmology. [arXiv:1309.0721].

[17] Jeffrey Goldstone, Abdus Salam, and Steven Weinberg. Broken symmetries. doi:10.1103/PhysRev.127.965.

[18] Michael E. Peskin and Daniel V. Schroeder. *An Introduction to Quantum Field Theory*. Westview Press, 1995.

[19] Super-Kamiokande Collaboration. Evidence for oscillation of atmospheric neutrinos. doi:10.1103/PhysRevLett.81.1562.

[20] F. Zwicky. On the masses of nebulae and of clusters of nebulae. doi:10.1086/143864.

[21] D. Larson et al. Seven-year wilkinson microwave anisotropy probe (wmap) observations: Power spectra and wmap-derived parameters. doi:10.1088/0067-0049/192/2/16.

[22] Anto Lonappan et al. Bayesian evidences for dark energy models in light of current observational data. doi:10.1103/PhysRevD.97.043524.

[23] C. Abel et al. Measurement of the permanent electric dipole moment of the neutron. doi:10.1103/PhysRevLett.124.081803.

[24] Giacomo Cacciapaglia et al. LHC missing-transverse-energy constraints on models with universal extra dimensions. doi:10.1103/PhysRevD.87.075006.

[25] A. Djouadi et al. The minimal supersymmetric standard model: Group summary report. In *GDR (Groupement De Recherche) - Supersymetrie*, 1998. [arXiv:hep-ph/9901246].

[26] Daniele Alves et al. Simplified models for LHC new physics searches. doi:10.1088/0954-3899/39/10/105005.

[27] Johan Alwall, Philip Schuster, and Natalia Toro. Simplified models for a first characterization of new physics at the LHC. doi:10.1103/PhysRevD.79.075020.

[28] CMS Collaboration. Search for supersymmetry in final states with a single electron or muon using angular correlations and heavy-object identification in proton-proton collisions at $\sqrt{s} = 13$ TeV. doi:10.1007/JHEP09(2023)149.

[29] Lyndon Evans and Philip Bryant. LHC Machine. doi:10.1088/1748-0221/3/08/S08001.

[30] R. Bruce, N. Fuster-Martínez, A. Mereghetti, D. Mirarchi, and S. Redaelli. Review of LHC run 2 machine configurations. In *9th LHC Operations Evian Workshop*, pages 187–197, Geneva, Switzerland, 2019.

[31] ATLAS Collaboration. The atlas experiment at the cern large hadron collider. doi:10.1088/1748-0221/3/08/S08003.

[32] CMS Collaboration. The CMS experiment at the cern LHC. doi:10.1088/1748-0221/3/08/S08004.

[33] ALICE Collaboration. The alice experiment at the cern LHC. doi:10.1088/1748-0221/3/08/S08002.

[34] LHCb Collaboration. The LHCb detector at the LHC. doi:10.1088/1748-0221/3/08/S08005.

[35] CMS Collaboration. The CMS trigger system. doi:10.1088/1748-0221/12/01/P01020.

[36] A. Sirunyan et al. Performance of the CMS Level-1 trigger in proton-proton collisions at $\sqrt{s} = 13$ TeV. doi:10.1088/1748-0221/15/10/P10017.

[37] A. Sirunyan et al. Electron and photon reconstruction and identification with the CMS experiment at the CERN LHC. doi:10.1088/1748-0221/16/05/P05014.

[38] CMS Collaboration. Performance of the CMS muon detector and muon reconstruction with proton-proton collisions at $\sqrt{s} = 13$ TeV. [arXiv:1804.04528]. URL `10.1088/1748-0221/13/06/P06015`.

[39] CMS Collaboration. Description and performance of track and primary-vertex reconstruction with the CMS tracker. doi:10.1088/1748-0221/9/10/P10009.

[40] CMS Collaboration. Particle-flow reconstruction and global event description with the CMS detector. doi:10.1088/1748-0221/12/10/P10003.

[41] Julia Bauer. *Prospects for the Observation of Electroweak Top Quark Production with the CMS Experiment*. PhD thesis, KIT, Karlsruhe, Dept. Phys., 2010.

[42] P. Adzic et al. Energy resolution of the barrel of the CMS electromagnetic calorimeter. doi:10.1088/1748-0221/2/04/P04004.

[43] CMS Collaboration. Simulation of the CMS electromagnetic calorimeter response at the energy and intensity frontier. doi:10.1088/1742-6596/1162/1/012007.

[44] CMS-HCAL Collaboration. Design, performance, and calibration of cms hadron-barrel calorimeter wedges. doi:10.1140/epjc/s10052-008-0573-y.

[45] CMS Collaboration. Performance of the CMS hadron calorimeter with cosmic ray muons and LHC beam data. doi:10.1088/1748-0221/5/03/T03012.

[46] S. Agostinelli et al. Geant4–a simulation toolkit. doi:10.1016/S0168-9002(03)01368-8.

[47] Sezen Sekmen. Recent developments in CMS fast simulation. doi:10.22323/1.282.0181.

[48] CMS Offline Software and Computing. Cms phase-2 computing model: Update document. Technical report, CERN, Geneva, 2022.

[49] G. Apollinari, O. Brüning, T. Nakamoto, and Lucio Rossi. High luminosity large hadron collider HL-LHC. doi:10.5170/CERN-2015-005.1.

[50] CMS Collaboration. The CMS HGCAL detector for HL-LHC upgrade. In *5th Large Hadron Collider Physics Conference*, 2017. [arXiv:1708.08234].

[51] Natascha Krammer. Accelerating Full and Fast Simulation of the CMS Experiment. Technical report, CERN, Geneva, 2024.

[52] Louis Portalès. L1 triggering on high-granularity information at the HL-LHC. doi:10.3390/instruments6040071.

[53] Glen Cowan, Kyle Cranmer, Eilam Gross, and Ofer Vitells. Asymptotic formulae for likelihood-based tests of new physics. doi:10.1140/epjc/s10052-011-1554-0. [Erratum: Eur.Phys.J.C 73, 2501 (2013)].

[54] Joosep Pata et al. Machine learning for particle flow reconstruction at cms. doi:10.1088/1742-6596/2438/1/012100.

[55] Daniel Guest et al. Jet flavor classification in high-energy physics with deep neural networks. doi:10.1103/PhysRevD.94.112002.

[56] Pierre Baldi, Peter Sadowski, and Daniel Whiteson. Searching for exotic particles in high-energy physics with deep learning. doi:10.1038/ncomms5308.

[57] Matthew Feickert and Benjamin Nachman. A living review of machine learning for particle physics. doi:10.48550/ARXIV.2102.02770.

[58] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning.

[59] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. URL `https://api.semanticscholar.org/CorpusID:195908774`.

[60] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. doi:10.1162/neco.1997.9.8.1735.

[61] Ashish Vaswani et al. Attention is all you need. [arXiv:1706.03762]. URL http://arxiv.org/abs/1706.03762.

[62] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[63] M. Erdmann, J. Glombitza, G. Kasieczka, and U. Klemradt. *Deep Learning For Physics Research*. World Scientific Publishing Company, 2021.

[64] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. doi:https://doi.org/10.1016/0893-6080(89)90020-8.

[65] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. doi:10.1038/323533a0.

[66] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. [arXiv:1708.02002].

[67] Adam Paszke et al. Pytorch: An imperative style, high-performance deep learning library. [arXiv:1912.01703].

[68] Herbert Robbins and Sutton Monro. A stochastic approximation method. doi:10.1214/aoms/1177729586.

[69] Tijmen Tieleman and Geoffery Hinton. RMSprop gradient optimization. URL http://www.cs.toronto.edu/tijmen/csc321/slides/lecture_slides_lec6.pdf.

[70] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. URL http://jmlr.org/papers/v12/duchi11a.html.

[71] Martin A. Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: the rprop algorithm. URL https://api.semanticscholar.org/CorpusID:16848428.

[72] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. [arXiv:1412.6980].

[73] Martin Heusel et al. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, page 6626, 2017. [arXiv:1706.08500].

[74] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts.

[75] Preetum Nakkiran et al. Deep double descent: Where bigger models and more data hurt. [arXiv:1912.02292].

[76] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. [arXiv:1711.05101].

[77] Nitish Srivastava et al. Dropout: A simple way to prevent neural networks from overfitting.

[78] Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. *Neural Networks: Tricks of the Trade: Second Edition.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[79] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[80] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. [arXiv:1502.01852].

[81] G. E. P. Box and D. R. Cox. An analysis of transformations. URL `http://www.jstor.org/stable/2984418`.

[82] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. [arXiv:1402.1869].

[83] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. [arXiv:1503.02406].

[84] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition.

[85] Matus Telgarsky. Benefits of depth in neural networks. [arXiv:1602.04485].

[86] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.7665`.

[87] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. [arXiv:1505.04597].

[88] Jia Deng et al. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[89] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. [arXiv:1603.05027].

[90] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[91] Taco Cohen and Max Welling. Group equivariant convolutional networks. [arXiv:1602.07576].

[92] Manzil Zaheer et al. Deep sets. [arXiv:1703.06114].

[93] Edward Wagstaff et al. On the limitations of representing functions on sets. [arXiv:1901.09006].

[94] Alexander Bogatskiy, Timothy Hoffman, David W. Miller, and Jan T. Offermann. Pelican: Permutation equivariant and lorentz invariant or covariant aggregator network for particle physics. [arXiv:2211.00454].

[95] Patrick T. Komiske, Eric M. Metodiev, and Jesse Thaler. Energy flow networks: Deep sets for particle jets. doi:10.1007/JHEP01(2019)121.

[96] Erik Buhmann, Gregor Kasieczka, and Jesse Thaler. EPiC-GAN: Equivariant point cloud generation for particle jets. doi:10.21468/SciPostPhys.15.4.130.

[97] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners.

[98] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. [arXiv:2006.16236].

[99] Kelvin Xu et al. Show, attend and tell: Neural image caption generation with visual attention. [arXiv:1502.03044].

[100] John Jumper, Richard Evans, and et al. Pritzel. Highly accurate protein structure prediction with alphafold. doi:10.1038/s41586-021-03819-2.

[101] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. [arXiv:2001.04451].

[102] Trevor Gale, Matei Zaharia, Cliff Young, and Erich Elsen. Sparse GPU kernels for deep learning. [arXiv:2006.10901].

[103] Yunyang Xiong et al. Nyströmformer: A nyström-based algorithm for approximating self-attention. [arXiv:2102.03902].

[104] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. [arXiv:2205.14135].

[105] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. [arXiv:2307.08691].

[106] Benno Kaech. `gen_model_playground`: A playground for generative models. `https://github.com/kaechb/gen_model_playground`, 2023. Accessed: 2023-12-12.

[107] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. [arXiv:1312.6114].

[108] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[109] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. doi:10.1561/2200000056.

[110] Ian J. Goodfellow et al. Generative adversarial networks. doi:10.48550/ARXIV.1406.2661.

[111] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. doi:10.48550/ARXIV.1703.10593.

[112] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. doi:10.48550/ARXIV.1511.06434.

[113] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. doi:10.48550/ARXIV.1611.07004.

[114] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. doi:10.48550/ARXIV.1609.02612.

[115] Thomas Schlegl et al. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. doi:10.48550/ARXIV.1703.05921.

[116] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. doi:10.48550/ARXIV.1703.10847.

[117] Mario Lucic et al. Are GANs created equal? a large-scale study. doi:10.48550/ARXIV.1711.10337.

[118] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, and Zhen Wang. Multi-class generative adversarial networks with the L2 loss function. [arXiv:1611.04076].

[119] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. [arXiv:1701.07875].

[120] Ishaan Gulrajani et al. Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, volume 30, page 5767. Curran Associates, Inc., 2017. [arXiv:1704.00028].

[121] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. URL `http://yann.lecun.com/exdb/mnist/`.

[122] Fisher Yu et al. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. [arXiv:1506.03365].

[123] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. [arXiv:1606.03498].

[124] Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. [arXiv:1602.07868].

[125] Sitao Xiang and Hao Li. On the effects of batch and weight normalization in generative adversarial networks. [arXiv:1704.03971].

[126] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. [arXiv:1802.05957].

[127] Hoang Thanh-Tung and Truyen Tran. On catastrophic forgetting and mode collapse in generative adversarial networks. [arXiv:1807.04015].

[128] John Ellis. The discovery of the gluon. doi:10.1142/S0217751X14300725.

[129] J. P. Agnelli et al. Clustering and classification through normalizing flows in feature space. doi:10.1137/100783522.

[130] Esteban Tabak and Eric Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. doi:10.4310/CMS.2010.v8.n1.a11.

[131] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538. PMLR, 2015. [arXiv:1505.05770].

[132] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. doi:10.48550/ARXIV.1410.8516.

[133] Claudius Krause and David Shih. Fast and accurate simulations of calorimeter showers with normalizing flows. doi:10.1103/PhysRevD.107.113003.

[134] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. doi:10.48550/ARXIV.1705.07057.

[135] Sascha Diefenbacher et al. L2lflows: Generating high-fidelity 3d calorimeter images. [arXiv:2302.11594].

[136] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. doi:10.48550/ARXIV.1605.08803.

[137] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. doi:10.48550/ARXIV.1807.03039.

[138] Ivan Kobyzev, Simon J. D. Prince, and Marcus A. Brubaker. Normalizing flows: An introduction and review of current methods. doi:10.1109/tpami.2020.2992934.

[139] George Papamakarios et al. Normalizing flows for probabilistic modeling and inference. doi:10.48550/ARXIV.1912.02762.

[140] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. doi:10.48550/ARXIV.1906.04032.

[141] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. `nflows`: normalizing flows in PyTorch. doi:10.5281/zenodo.4296287.

[142] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. [arXiv:1612.08083].

[143] Samuel Klein and Tobias Golling. Decorrelation with conditional normalizing flows. doi:10.48550/ARXIV.2211.02486.

[144] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. doi:10.48550/ARXIV.1806.07366.

[145] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. [arXiv:2006.11239].

[146] Prafulla Dhariwal and Alex Nichol. Diffusion models beat GANs on image synthesis. [arXiv:2105.05233].

[147] Yaron Lipman et al. Flow matching for generative modeling. [arXiv:2210.02747].

[148] Yang Song et al. Score-based generative modeling through stochastic differential equations. [arXiv:2011.13456].

[149] Robert J. McCann. A convexity principle for interacting gases. URL `https://api.semanticscholar.org/CorpusID:123005604`.

[150] Alexander Tong et al. Improving and generalizing flow-based generative models with minibatch optimal transport. [arXiv:2302.00482].

[151] Kilian Fatras et al. Minibatch optimal transport distances; analysis and applications. [arXiv:2101.01792].

[152] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. [arXiv:1512.00567].

[153] Min Jin Chong and David Forsyth. Effectively unbiased fid and inception score and where to find them. [arXiv:1911.07023].

[154] Tom Fawcett. An introduction to roc analysis. doi:10.1016/j.patrec.2005.10.010.

[155] Sung Hak Lim, Kailash A. Raman, Matthew R. Buckley, and David Shih. Galaxyflow: Upsampling hydrodynamical simulations for realistic gaia mock catalogs. [arXiv:2211.11765].

[156] Raghav Kansal et al. Particle cloud generation with message passing generative adversarial networks. [arXiv:2106.11535].

[157] Benno Kaech et al. Jetflow: Generating jets with conditioned and mass constrained normalising flows. In *21st International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT 2022)*, 2022. [arXiv:2211.13630].

[158] Benno Käch, Dirk Krücker, and Isabell Melzer-Pellmann. Point cloud generation using transformer encoders and normalising flows. [arXiv:2211.13623].

[159] Benno Käch, Isabell Melzer-Pellmann, and Dirk Krücker. Pay attention to mean-fields for point cloud generation, `https://ml4physicalsciences.github.io/2023/files/NeurIPS_ML4PS_2023_10.pdf`. URL `https://ml4physicalsciences.github.io/2023/files/NeurIPS_ML4PS_2023_10.pdf`.

[160] R. Frederix et al. The automation of next-to-leading order electroweak calculations. doi:10.1007/JHEP11(2021)085. [Erratum: JHEP 11, 085 (2021)].

[161] Torbjörn Sjöstrand et al. An introduction to pythia 8.2. doi:10.1016/j.cpc.2015.01.024.

[162] Raghav Kansal et al. Jetnet(1.1) [topquark], `10.5281/zenodo.6302454`, 2021.

[163] Benno Käch and Isabell Melzer-Pellmann. Attention to mean-fields for particle cloud generation. [arXiv:2305.15254].

[164] F. N. Fritsch and J. Butland. A method for constructing local monotone piecewise cubic interpolants. doi:10.1137/0905021.

[165] Pauli Virtanen et al. Scipy 1.0–fundamental algorithms for scientific computing in python. doi:10.1038/s41592-019-0686-2.

[166] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall/CRC, 1986.

[167] David W. Scott. On optimal and data based histograms. URL `https://api.semanticscholar.org/CorpusID:59500098`.

[168] Raghav Kansal et al. On the evaluation of generative models in high energy physics. [arXiv:2211.10295].

[169] Patrick T. Komiske, Eric M. Metodiev, and Jesse Thaler. Energy flow polynomials: A complete linear basis for jet substructure. doi:10.1007/JHEP04(2018)013.

[170] Arthur Gretton, Karsten Borgwardt, Malte J. Rasch, Bernhard Scholkopf, and Alexander J. Smola. A kernel method for the two-sample problem, 2008. [arXiv:0805.2368].

[171] Erik Buhmann et al. Epic-ly fast particle cloud generation with flow-matching and diffusion. [arXiv:2310.00049].

[172] Matthew Leigh et al. PC-JeDi: Diffusion for particle cloud generation in high energy physics. [arXiv:2303.05376].

[173] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. [arXiv:1812.04948].

[174] OpenAI. DALL · E 3, `https://openai.com/dall-e-3`, 2023.

[175] Roman Klokov, Edmond Boyer, and Jakob Verbeek. Discrete point flow networks for efficient point cloud generation. In *2020 European Conference on Computer Vision (ECCV)*, page 694, Cham, 2020. Springer International Publishing.

[176] Guandao Yang et al. PointFlow: 3D point cloud generation with continuous normalizing flows. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, page 4540, 2019.

[177] Erik Buhmann et al. Caloclouds: Fast geometry-independent highly-granular calorimeter simulation. [arXiv:2305.04847].

[178] Erik Buhmann et al. CaloClouds II: Ultra-fast geometry-independent highly-granular calorimeter simulation. [arXiv:2309.05704].

[179] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. doi:10.48550/ARXIV.1810.04805.

[180] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. doi:10.48550/ARXIV.1607.06450.

[181] Vinicius Mikuni, Benjamin Nachman, and Mariel Pettee. Fast point cloud generation with diffusion models in high energy physics. [arXiv:2304.01266].

[182] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. [arXiv:1503.02531].

[183] Claudius (ed) Krause et al. Calochallenge 2022: A community challenge for fast calorimeter simulation. accessed on March 12th, 2023, 2022.

[184] Michele Faucci Giannelli et al. Fast calorimeter simulation challenge 2022 - dataset 1. doi:10.5281/ZENODO.8099322.

[185] Atlas Collaboration. Datasets used to train the generative adversarial networks used in atlfast3. doi:10.7483/OPENDATA.ATLAS.UXKX.TXBN.

[186] Faucci Giannelli et al. Fast calorimeter simulation challenge 2022 - dataset 2. doi:10.5281/zenodo.6366271.

[187] Geant4 Collaboration. Par04 example. URL `https://gitlab.cern.ch/geant4/geant4/-/tree/master/examples/extended/parameterisations/Par04`.

[188] Faucci Giannelli et al. Fast calorimeter simulation challenge 2022 - dataset 3. doi:10.5281/ZENODO.6366324.

[189] Moritz Scham and Benno Käch. `caloutils`: Metrics and tools for evaluation of generative models for calorimeter showers, 2023. v.0.0.18.

[190] Anna Zaborowska. Level up your performance calculation of the fast shower simulation model. Online, 2022. Presentation at ML4Jets.

[191] Joschka Birk, Anna Hallin, and Gregor Kasieczka. Omnijet-α: The first cross-task foundation model for particle physics. [arXiv:2403.05618].

[192] Emil Bols et al. Jet flavour classification using DeepJet. doi:10.1088/1748-0221/15/12/P12012.

[193] Ashraf Kasem Abdellatif Mohamed. *Search for Dark Matter with machine learning techniques in leptonic final states and missing transverse energy at the CMS Experiment*. PhD thesis, RWTH Aachen U., RWTH Aachen U., Hamburg, 2021.

[194] Sascha Diefenbacher et al. DCTRGAN: Improving the precision of generative models with reweighting. doi:10.1088/1748-0221/15/11/P11004.

[195] S. S. Wilks. The large-sample distribution of the likelihood ratio for testing composite hypotheses. doi:10.1214/aoms/1177732360.

[196] Abraham Wald. Tests of statistical hypotheses concerning several parameters when the number of observations is large. URL `https://api.semanticscholar.org/CorpusID:54174575`.