Dissertation
submitted to the
Combined Faculties of the Natural Sciences and Mathematics
of the Ruperto-Carola-University of Heidelberg, Germany
for the degree of
Doctor of Natural Sciences

Put forward by
Marco Bellagente
born in Desio
Oral examination: XX.XX.2022

Go with the Flow

Normalizing Flow applications for High Energy Physics

Referees: Prof. Dr. Jan-Martin Pawlowski

Prof. Dr. Monica Dunford (?)

Abstract

bla bla

Zusammenfassung

bla bla Auf Deutsch

Contents

1 Introduction **1**

2 Generative Models **3**

3 Unfolding in particle physics **5**

4 GAN unfolding **7**
4.1 Introduction . . . 7
4.2 GAN unfolding . . . 8
4.3 Fully conditional GAN . . . 11
4.4 New physics injection . . . 15
4.5 Outlook . . . 17

5 INN unfolding **19**
5.1 Introduction . . . 20
5.2 Unfolding basics . . . 21
5.2.1 Binned toy model and locality . . . 21
5.2.2 Bayes' theorem and model dependence . . . 22
5.2.3 Reference process $pp \to ZW$ . . . 23
5.3 Unfolding detector effects . . . 24
5.3.1 Naive INN . . . 24
5.3.2 Noise-extended INN . . . 26
5.3.3 Conditional INN . . . 28
5.4 Unfolding with jet radiation . . . 32
5.4.1 Individual n-jet samples . . . 32
5.4.2 Combined n-jet sample . . . 34
5.5 Outlook . . . 36

6 Bayesian Neural Networks **37**
6.1 Introduction . . . 37
6.2 Generative networks with uncertainties . . . 38
6.2.1 Uncertainties on event samples . . . 38
6.2.2 Invertible Neural Networks . . . 39
6.2.3 Bayesian INN . . . 40
6.3 Toy events with uncertainties . . . 42
6.3.1 Wedge ramp . . . 43
6.3.2 Kicker ramp . . . 44

6.3.3 Gaussian ring . . . 45
6.3.4 Errors vs training statistics . . . 47
6.3.5 Marginalizing phase space . . . 48
6.4 LHC events with uncertainties . . . 50
6.5 Outlook . . . 52

7 Latent Space Refinement **55**

8 Summary **57**

Preface

The research presented in this thesis was conducted at the Institute for Theoretical Physics at Heidelberg University from February 2019 to February 2022. The contents of the Chapters **??**-**??** are based on work in collaboration with other authors and have previously been published as

[1] M. Bellagente, A. Butter, G. Kasieeczka, T. Plehn and R. Winterhalder,
"How to GAN away Detector Effects",
SciPost Phys. **8** (2020) no. 4, 070, `arXiv:1912.00477 [hep-ph]`

[2] M. Bellagente, A. Butter, G. Kasieeczka, T. Plehn, R. Winterhalder, L. Ardizzone and U. Köthe,
"Invertible networks or partons to detector and back again",
SciPost Phys. **9** (2020) 074, `arXiv:2006.06685 [hep-ph]`

[3] M. Bellagente, M. Luchmann, M. Haußmann and T. Plehn,
"Understanding Event- Generation Networks via Uncertainties",
`arXiv:2104.04543 [hep-ph]`

[4] R. Winterhalder, M. Bellagente, and B. Nachmann
"Latent Space Refinement for Deep Generative Models",
`arXiv:2106.00792 [stat.ML]`

1 | Introduction

The motivation behind the prediction of a fundamental scalar particle in the Standard Model (SM), the Higgs boson, was to grant a mechanism for the generation of the masses of the electroweak gauge bosons via electroweak symmetry breaking (EWSB) [**?**, **?**, **?**]. The discovery of a Higgs boson at the Large Hadron Collider (LHC) [**?**, **?**] strongly hints at EWSB indeed being the mechanism behind the mass generation of the SM particles. One of the pivotal tasks of the LHC and future colliders is to probe both the local and global structure of the Higgs potential, which is reflected in the couplings of the Higgs boson to other SM particles and in its self-coupling, respectively. In this thesis, we present a global view on Higgs couplings at the LHC to extend our understanding of the EWSB sector and to set universal constraints on new physics that might be hiding in it.

Driven by the question what current data reveal about the EWSB sector and new physics that might be hiding in it, in this thesis we aim at increasing the precision of Higgs-coupling measurements and combining them in a comprehensive framework. This requires us to rethink the way we perform, interpret and combine experimental analyses in a way that fully exploits the available data. To tackle this challenge, we take a multi-prong approach: First, we focus on the improvement of an individual Higgs-production and decay channel by applying modern analysis techniques. Second, we perform global analyses of the Higgs-gauge sector for the LHC and a potential future upgrade of the LHC in a model-independent framework.

Motivated by the experimental advances of LHC Run II, we perform a global fit of the Higgs-gauge sector based on Higgs and di-boson measurements as well as electroweak precision data in Chapter **??**. We include momentum-related kinematic distributions and examine the impact of the different LHC Run II measurements on the reach of our global analysis in detail. On the theory side, we broaden our view on the Higgs sector by expanding the set of considered dimension-six operators from 10 to 18 with respect to previous SFitter analyses [**?**, **?**]. This extension of our operator set will bring us a significant step closer to a global SMEFT fit at dimension six. We discuss how the additional fermionic Higgs-gauge operators have a relevant impact on a global fit of the Higgs-gauge sector despite the strong constraints from electroweak precision data.

In Chapter **??**, we will summarize our results and give an outlook to further improvements and extensions of the concepts discussed in this thesis. Each of the lines of research mentioned above will aid in constructing a global view of Higgs couplings at the LHC as well as its proposed future upgrade and will bring us one step closer to probing if EWSB is indeed described by the simple structure of the SM Higgs potential. The derived limits on Higgs couplings in the SMEFT framework can be mapped onto constraints for UV-complete BSM models [**?**, **?**]. Furthermore, they provide a key ingredient for future tests of the global structure of the Higgs potential. In summary, the thorough investigation of Higgs couplings at the LHC is crucial to gain a deeper understanding of the structure of the Higgs sector and EWSB on a fundamental level.

2 | Generative Models

bla

3 | Unfolding in particle physics

bla

4 | GAN unfolding

LHC analyses directly comparing data and simulated events bear the danger of using first-principle predictions only as a black-box part of event simulation. We show how simulations, for instance, of detector effects can instead be inverted using generative networks. This allows us to reconstruct parton level information from measured events. Our results illustrate how, in general, fully conditional generative networks can statistically invert Monte Carlo simulations. As a technical by-product we show how a maximum mean discrepancy loss can be staggered or cooled.

4.1 Introduction

Our understanding of LHC data from first principles is a unique strength of particle physics. It is based on a simulation chain which starts from a hard process described by perturbative QCD, and then adds the logarithmically enhanced QCD parton shower, fragmentation, hadronization, and finally a fast or complete detector simulation [**?**]. This simulation chain is publicly available and relies on extremely efficient, fast, and reliable Monte Carlo techniques.

Unfortunately, there is a price for this efficiency: while in principle such a Monte Carlo simulation as a Markov process can be inverted at least statistically, in practice we have to employ approximations. This asymmetry has serious repercussions for LHC analyses, where for instance we do not have access to the likelihood ratio of the hard process. Even worse, it seriously limits our interpretation of LHC results because we cannot easily show results in terms of observables accessible by perturbative QCD. For typical ATLAS or CMS limit reporting this might seem less relevant, but every so often we want to be able to understand such a result more quantitatively.

We propose to use generative networks or GANs [**?**] to invert Monte Carlo simulations. There are many examples showing that we can GAN such simulations, including phase space integration [**?**, **?**], event generation [**?**, **?**, **?**, **?**], detector simulations [**?**, **?**, **?**, **?**, **?**, **?**, **?**], unfolding [**?**], and parton showers [**?**, **?**, **?**, **?**]. The question is if and how we can invert them. We start with a naive GAN inversion and see how a mismatch between local structures at parton level and detector level leads to problems. We then introduce the first fully conditional GAN [**?**] (FCGAN) in particle physics to invert a fast detector simulation [**?**] for the process

$$pp \to ZW^\pm \to (\ell^-\ell^+)\,(jj)\,, \tag{4.1}$$

as illustrated in Fig. 5.1. We will see how the fully conditional setup gives us all the required properties of an inverted detector simulation.

We note that our approach is not targeted at combining detector unfolding [**?**, **?**, **?**] with optimized inference [**?**, **?**, **?**]. A powerful application for unfolded kinematic distributions to the hard process could be global analyses. For instance in the electroweak and Higgs sector exotics resonance searches turn out to be among the most interesting input and pose a challenge when including them [**?**]. In contrast, global analyses in the top sector [**?**, **?**, **?**] successfully rely on unfolded information to different levels of the hard process, for instance the top pair production process [**?**, **?**]. At the same time, alternative methods like simplified template cross sections lose a sizeable amount of information [**?**]. The same method would also allow us to directly compare first-principles QCD predictions with modern LHC measurements. In addition, our fast inversion might help with advanced statistical techniques like the matrix element method [**?**, **?**, **?**, **?**, **?**, **?**].

But most importantly, our FCGAN first serves as an example how we can invert Monte Carlo simulations to understand the physics behind modern LHC analyses based on a direct comparison of data and simulations. Here the GAN benefits from the excellent interpolations properties of neural networks. Second, faithfully preserves local structures leading to a large degree of model independence in the unfolding procedure.

4.2 GAN unfolding

A standard method for fast detector simulation is smearing the outgoing particle momenta with a detector response function. This allows us to generate and sample from a probability distribution of smeared final-state momenta for a single parton-level event. For the inversion we need to rely on event samples, as we can see from a simple example: we start from a sharp Z-peak at the parton level and broaden it with detector effects. Now we look at a detector-level event in the tail and invert the detector simulations, for which we need to know in which direction in the invariant mass the preferred value m_Z lies. This implies that unfolding detector effects requires a model hypothesis, which can be thought of as a condition in a probability of the inversion from the detector level. The problem with this point of view is that the parton-level distribution of the invariant mass requires a dynamic reconstruction of the Breit-Wigner peak, which is not easily combined with a Markov process. In any case, from this argument it is clear that unfolding only makes sense at the level of large enough event samples.

For our example we rely on two event samples: we start with events at the parton level, simulated with madgraph [**?**]. For the second sample we first apply pythia not including initial state radiation. Technically this means that we only have to deal with a fixed number of partons in the final state and that we can more easily match partons and jets. Further, we apply delphes [**?**] as a fast detector simulation and reconstruct the smeared jet 4-momenta with a jet algorithm included in fastjet [**?**]. For lepton 4-momenta we can directly compare the parton-level output with the detector-level output. From Ref. [**?**] we know how to set up a GAN to either generate detector-level events from parton-level events or vice versa. In our current setup the events are unweighted set of four 4-vectors, two jets and two leptons, but it can be easily adapted to weighted events. The final-state masses are fixed to the parton-level values. Our hadronic final state is defined at the level of jet 4-vectors. This does not mean that in a possible application we take a parton shower at face value. All we do is assume that there is a correspondence between a hard parton and its hadronic final state, and that the parton 4-momentum can be reconstructed with the help of a jet algorithm. The question if for instance an anti-k_T algorithm is an appropriate description of sub-jet physics does not arise as long as the jet algorithm reproduces the hard parton momentum.

Our GAN comprises a generator network G competing against a discriminator network D in a min-max game, as illustrated in Fig. 4.2. For the implementation we use keras (v2.2.4) [**?**] with a tensorflow (v1.14) backend [**?**]. As the starting point, G is randomly initialized to produce an output, typically with the same dimensionality as the target space. It induces a probability distribution $P_G(x)$ of a target space element x, in our case a parton-level event. To be precise, the generator obtains a batch of detector level event as input and generates a batch of parton level events as output, i.e.

Figure 4.1: Sample Feynman diagram contributing to WZ production, with intermediate on-shell particles labelled.

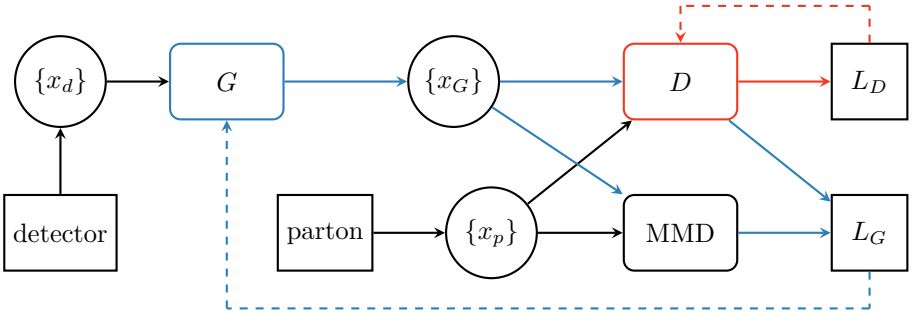

Figure 4.2: Structure of a naive unfolding GAN. The input $\{x_d\}$ describes a batch of events sampled at detector level and $\{x_{G,p}\}$ denotes events sampled from the generator or parton-level data. The blue (red) arrows indicate which connections are used in the training of the generator (discriminator).

$G(\{x_d\}) = \{x_G\}$. The discriminator is given batches $\{x_G\}$ and $\{x_p\}$ sampled from P_G and the parton-level target distribution P_p. It is trained as a binary classifier, such that $D(x \in \{x_p\}) = 1$ and $D(x) = 0$ otherwise. Following the conventions of Ref. [?] the discriminator loss function is defined as

$$L_D = \langle -\log D(x) \rangle_{x \sim P_p} + \langle -\log (1 - D(x)) \rangle_{x \sim P_G} \ . \tag{4.2}$$

We add a regularization and obtain the regularized Jensen-Shannon GAN loss function [?]

$$L_D^{(\text{reg})} = L_D + \lambda_D \left\langle (1 - D(x))^2 \, |\nabla \phi|^2 \right\rangle_{x \sim P_p} + \lambda_D \left\langle D(x)^2 \, |\nabla \phi|^2 \right\rangle_{x \sim P_G} , \tag{4.3}$$

with a properly chosen pre-factor λ_D and where we define $\phi(x) = \log \frac{D(x)}{1-D(x)}$. The discriminator training at fixed P_p and P_G alternates with the generator training, which is trained to maximize the second term in Eq.(4.2) using the truth encoded in D. This is efficiently encoded in minimizing

$$L_G = \langle -\log D(x) \rangle_{x \sim P_G} \ . \tag{4.4}$$

If the training of the generator and the discriminator with their respective losses Eq.(4.3) and Eq.(4.4) is properly balanced, the distribution P_G converges to the parton-level distribution P_p, while the optimized discriminator is unable to distinguish between real and generated samples.

If we want to describe phase space features, for instance at the LHC, it is useful to add a maximum mean discrepancy (MMD) [?] contribution to the loss function *. It allows us to compare pre-defined distributions, for instance the one-dimensional invariant mass of an intermediate particle. Given batches of true and generated parton-level events we define the additional contribution to the generator loss as

$$\text{MMD} = \left[\langle k(x, x') \rangle_{x,x' \sim P_G} + \langle k(y, y') \rangle_{y,y' \sim P_p} - 2 \langle k(x, y) \rangle_{x \sim P_G, y \sim P_p} \right]^{1/2} , \tag{4.5}$$

with another pre-factor λ_G. Note that we use MMD instead of MMD^2 to enhance the sensitivity of the model [?]. In Ref. [?] we have compared common choices, like Gaussian or Breit-Wigner kernels with a given width σ,

$$k_{\text{Gauss}}(x, y) = \exp \frac{-(x-y)^2}{2\sigma^2} \qquad \text{or} \qquad k_{\text{BW}}(x, y) = \frac{\sigma^2}{(x-y)^2 + \sigma^2} \ . \tag{4.6}$$

As a naive approach to GAN unfolding we use detector-level event samples as generator input. The network input is always a set of four 4-vectors, one for each particle in the final state, with their masses fixed [?]. In the GAN setup we train our network to map detector-level events to parton-level events. Both networks consist of 12 layers with 512 units per layer. With $\lambda_G = 1$, $\lambda_D = 10^{-3}$ and a batch size of 512 events, we run for 1200 epochs and 500 iterations per epoch.

*For all details on combining GANs with MMD we refer to the original paper [?].

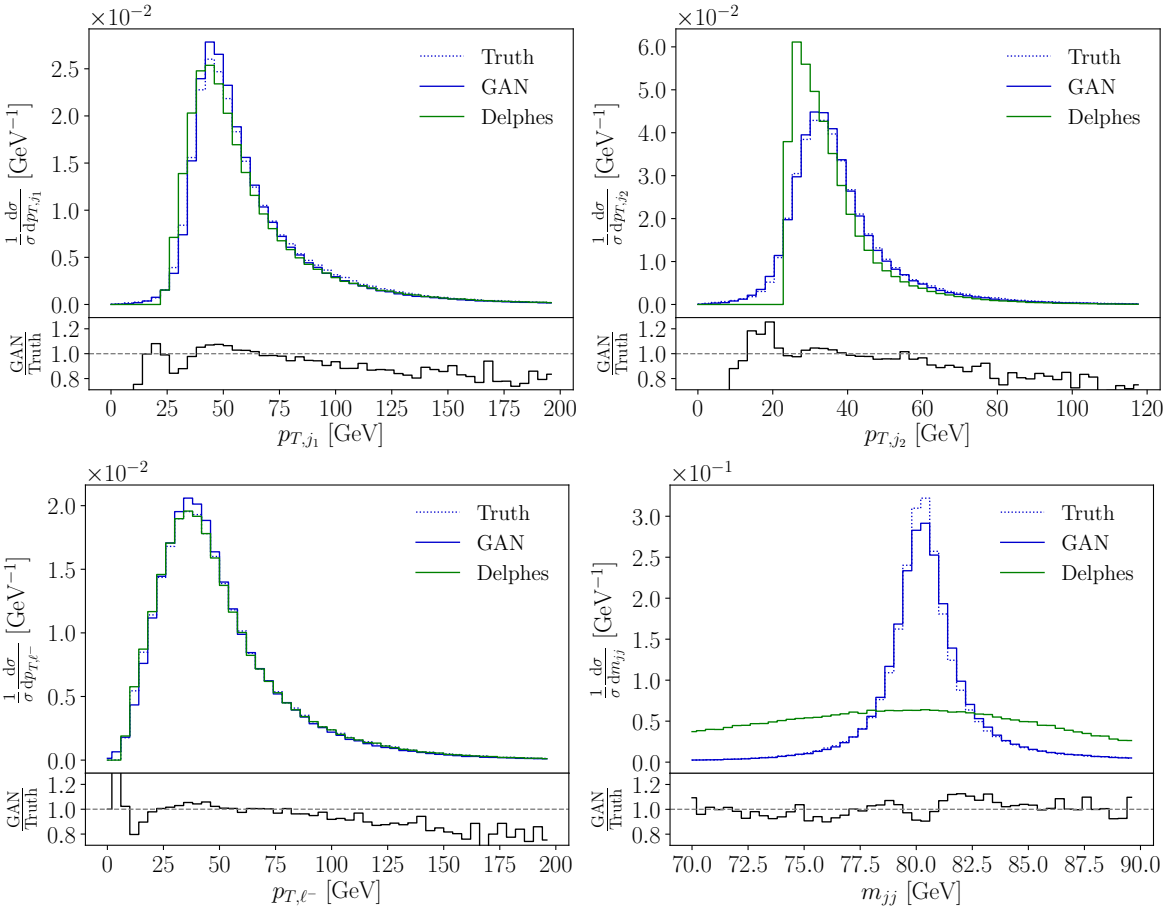

Figure 4.3: Example distributions for parton level truth, after detector simulation, and GANned back to parton level. The lower panels give the ratio of parton level truth and reconstructed parton level.

For our $Z_{\ell\ell}W_{jj}$ process we generate 300k events at LO using madgraph (v2.6.7) [**?**] (without any generation cuts) with the standard pythia (v8.2) shower [**?**] and then simulate the detector effects event-by-event with delphes (v3.3.3) [**?**] using the standard ATLAS card. For the reconstruction of the jets we use the anti-k_t jet algorithm [**?**] with $R = 0.6$ which is performed via the fastjet (v3.1.3) [**?**] package included in delphes. To keep our toy setup simple we select events with exactly two jets and a pair of same-flavor opposite-sign leptons, specifically electrons. At the detector level both jets are required to fulfill $p_{T,j} > 25$ GeV and $|\eta_j| < 2.5$ GeV. At detector level jets are sorted by p_T. We assign each jet to a corresponding parton level object based on their angular distance. The detector and parton level leptons are assigned based on their charge. While the resulting smearing of the lepton momenta will only have a modest effect, the observed widths of the hadronically decaying W-boson will be much larger than the parton-level Breit-Wigner distribution. For this reason, we focus on showing hadronic observables to benchmark the performance of our setup.

In Fig. 4.3 we compare true parton-level events to the output from a GAN trained to unfold the detector effects. We run the unfolding GAN on a set of statistically independent, but simulation-wise identical sets of detector-level events. Both, the relatively flat p_{T,j_1} and the peaked m_{jj} distributions agree well between the true parton-level events and the GAN-inverted sample, indicating that the statistical inversion of the detector effect works well.

A great advantage of this GAN approach is that, strictly speaking, we do not need event-by-event matched samples before and after detector simulation. The entire training is based on batches of typically 512 events, and these batches are independently chosen from the parton-level and detector-level samples. Increasing the batch size within the range allowed by the memory size and hence reducing

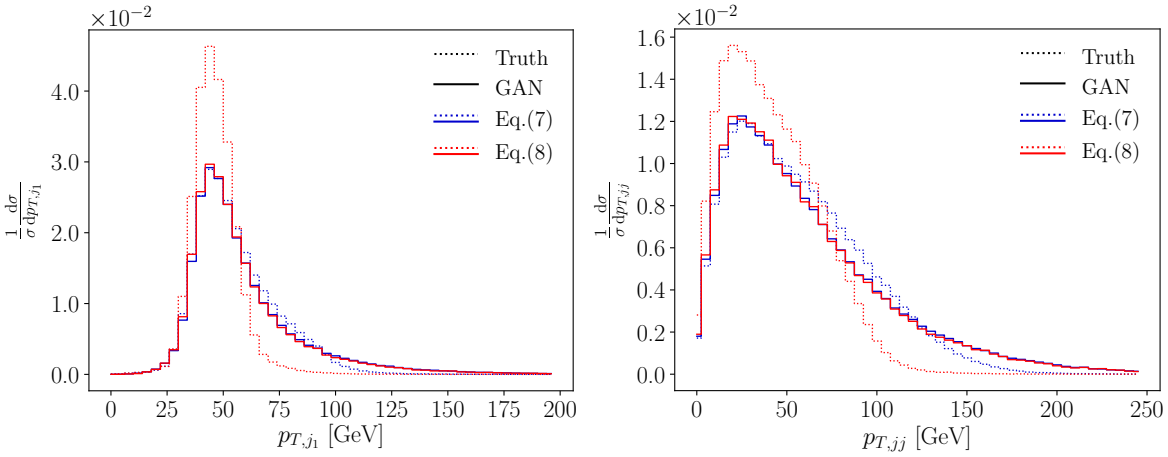

Figure 4.4: Parton level truth and GANned distributions when we train the GAN on the full data set but only unfold parts of phase space defined in Eq.(4.7) and Eq.(4.8).

the impact of event-wise matching will actually improve the GAN training, because it reduces statistical uncertainties [**?**].

The big challenge arises when we want to unfold an event sample which is not statistically equivalent to the training data; in other words, the unfolding model is not exactly the same as the test data. As a simple example we train the GAN on data covering the full phase space and then apply and test the GAN on data only covering part of the detector-level phase space. Specifically, we apply the two sets of jet cuts

$$\text{Cut I}: \quad p_{T,j_1} = 30 \ldots 100 \text{ GeV} \tag{4.7}$$

$$\text{Cut II}: \quad p_{T,j_1} = 30 \ldots 60 \text{ GeV} \quad \text{and} \quad p_{T,j_2} = 30 \ldots 50 \text{ GeV} , \tag{4.8}$$

which leave us with 88% and 38% of events, respectively. This approach ensures that the training has access to the full information, while the test sample is a significantly reduced sub-set of the full sample.

In Fig. 4.4 we show a set of kinematic distributions, for which we GAN only part of the phase space. As before, we can compare the original parton-level shapes of the distributions with the results from GAN-inverting the fast detector simulation. We see that especially the GANned $p_{T,j}$ distribution is strongly sculpted by the phase space cuts. This indicates that the naive GAN approach to unfolding does not work once the training and test data sets are not statistically identical. In a realistic unfolding problem we cannot expect the training and test data sets to be arbitrarily similar, so we have to go beyond the naive GAN setup described in Fig. 4.2. The technical reason for this behavior is that events which are similar or, by some metric, close at the detector level are not guaranteed to be mapped onto events which are close on the parton level. Looking at classification networks this is the motivation to apply variational methods, for instance upgrade autoencoders to variational autoencoders. For a GAN we discuss a standard solution in the next section.

4.3 Fully conditional GAN

The way out of the sculpting problem when looking at different phase space regions is to add a conditional structure to the GAN [**?**] shown in Fig. 4.2. The idea behind the conditional setup is not to learn a deterministic link between input and output samples, because we know that without an enforced structure in the weight or function space the generator does not benefit from the structured input. In other words, the network does not properly exploit the fact that the detector-level and parton-level data sets in the training sample are paired. A second, related problem of the naive GAN is that once trained the model is completely deterministic, so each detector-level event will always

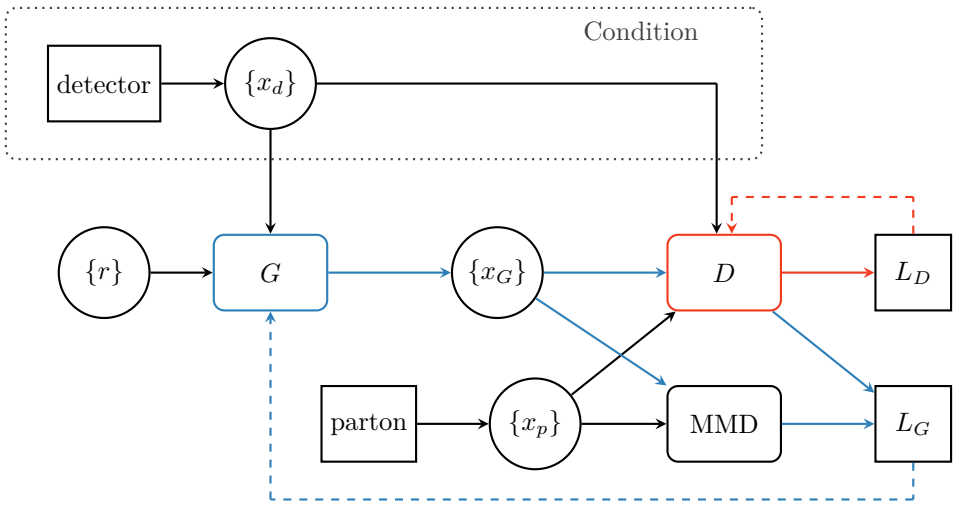

Figure 4.5: Structure of our fully conditional FCGAN. The input $\{r\}$ describes a batch of random numbers and $\{x_{G,d,p}\}$ denotes events sampled from the generator, detector-level data, or parton-level data. The blue (red) arrows indicate which connections are used in the training of the generator (discriminator).

be mapped to the same parton-level events. This goes against the physical intuition that this entire mapping is statistical in nature.

In Fig. 4.5 we introduce a fully conditional GAN (FCGAN). It is identical to our naive network the way we train and use the generator and discriminator. However, the input to the generator are actual random numbers $\{r\}$, and the detector-level information $\{x_d\}$ is used as an event-by-event conditional input on the link between a set of random numbers and the parton-level output, i.e. $G(\{r\}, \{x_d\}) = \{x_G\}$. This way the FCGAN can generate parton-level events from random noise but still using the detector-level information as input. To also condition the discriminator we modify its loss to

$$L_D \to L_D^{(\mathrm{FC})} = \langle -\log D\,(x,y)\rangle_{x\sim P_p, y\sim P_d} + \langle -\log\left(1 - D\,(x,y)\right)\rangle_{x\sim P_G, y\sim P_d} \ , \tag{4.9}$$

and the regularized loss function changes accordingly,

$$\begin{aligned} L_D^{(\mathrm{reg})} \to L_D^{(\mathrm{reg,\,FC})} = L_D^{(\mathrm{FC})} &+ \lambda_D \left\langle \left(1 - D(x,y)\right)^2 |\nabla\phi|^2 \right\rangle_{x\sim P_p, y\sim P_d} \\ &+ \lambda_D \left\langle D\,(x,y)^2\ |\nabla\phi|^2 \right\rangle_{x\sim P_G, y\sim P_d} , \end{aligned} \tag{4.10}$$

again using the conventions of Ref. [**?**]. The generator loss function now takes the form

$$L_G \to L_G^{(\mathrm{FC})} = \langle -\log D\,(x,y)\rangle_{x\sim P_G, y\sim P_d} \ . \tag{4.11}$$

Parameter	Value	Parameter	Value
Layers	12	Batch size	512
Units per layer	512	Epochs	1200
Trainable weights G	3M	Iterations per epoch	500
Trainable weights D	3M	Number of training events	3×10^5
λ_G	1		
λ_D	10^{-3}		

Table 4.1: FCGAN setup.

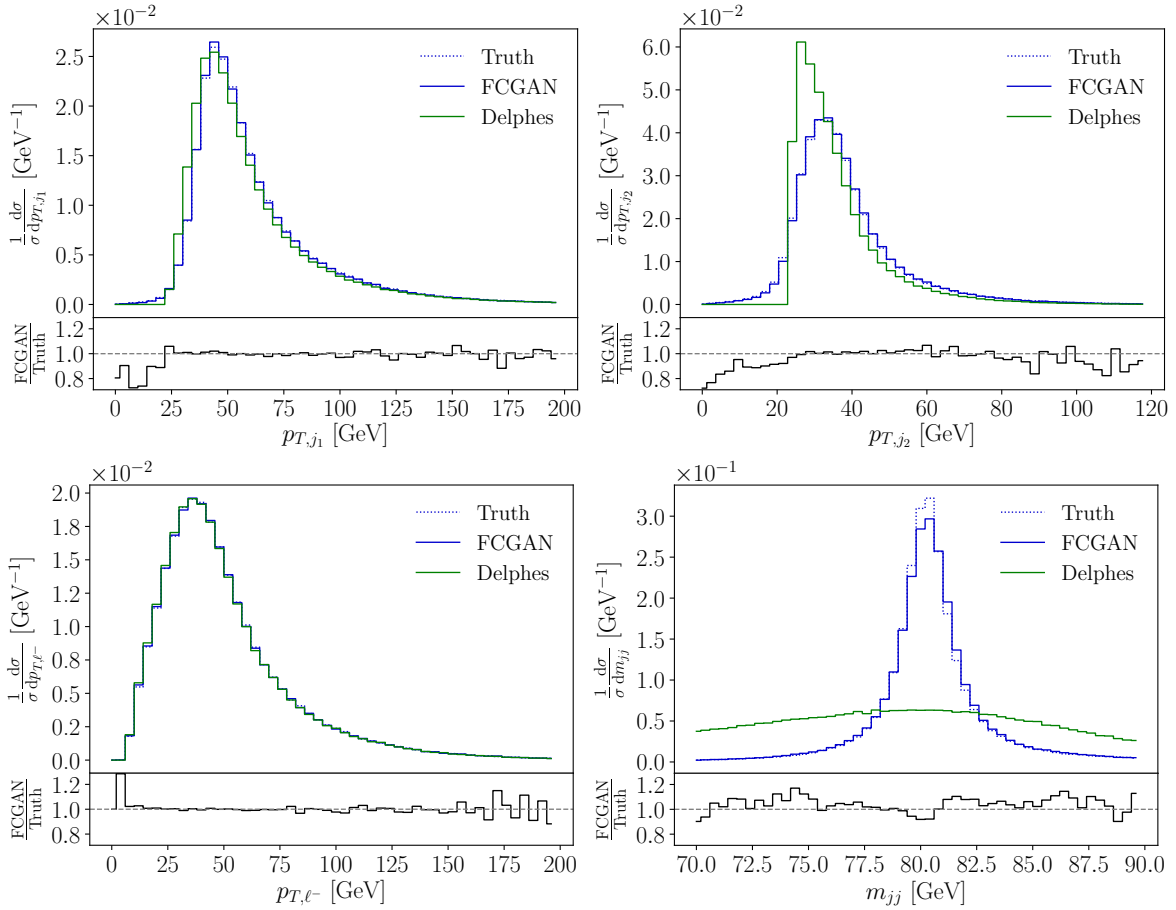

Figure 4.6: Example distributions for parton level truth, after detector simulation, and FCGANned back to parton level. The lower panels give the ratio of parton level truth and reconstructed parton level. The lower panels give the deviation between parton level truth and reconstructed parton level. To be compared with the naive GAN results in Fig. 4.3.

Note, that we do not build a conditional version of the MMD loss. The hyper-parameters of our FCGAN are summarized in Tab. 4.1. Changing from a naive GAN to a fully conditional GAN we have to pay a price in the structure of the training sample. While the naive GAN only required event batches to be matched between parton level and detector level, the training of the FCGAN actually requires event-by-event matching.

In Fig. 4.6 we compare the truth and the FCGANned events, trained on and applied to events covering the full phase space. Compared to the naive GAN, inverting the detector effects now works even better. The systematic under-estimate of the GAN rate in tails no longer occurs for the FCGAN. The reconstructed invariant W-mass forces the network to dynamically generate a very narrow physical width from a comparably broad Gaussian peak. Using our usual MMD loss developed in Ref. [**?**] we reproduce the peak position, width, and peak shape to about 90%. We emphasize that the MMD loss requires us to specify the relevant one-dimensional distribution, in this case m_{jj}, but it then extracts the on-shell mass or width dynamically. The multi-kernel approach we use in this case is explained in the Appendix.

As for our naive ansatz we now test what happens to the network when the training data and the test data do not cover the same phase space region. We train on the full set of events, to ensure that the full phase space information is accessible to the network, but we then only apply the network to the 88% and 38% of events passing the jet cuts I and II defined in Eq.(4.7) and Eq.(4.8). We show the results in Fig. 4.7. As observed before, especially the jet cuts with only 40% survival probability

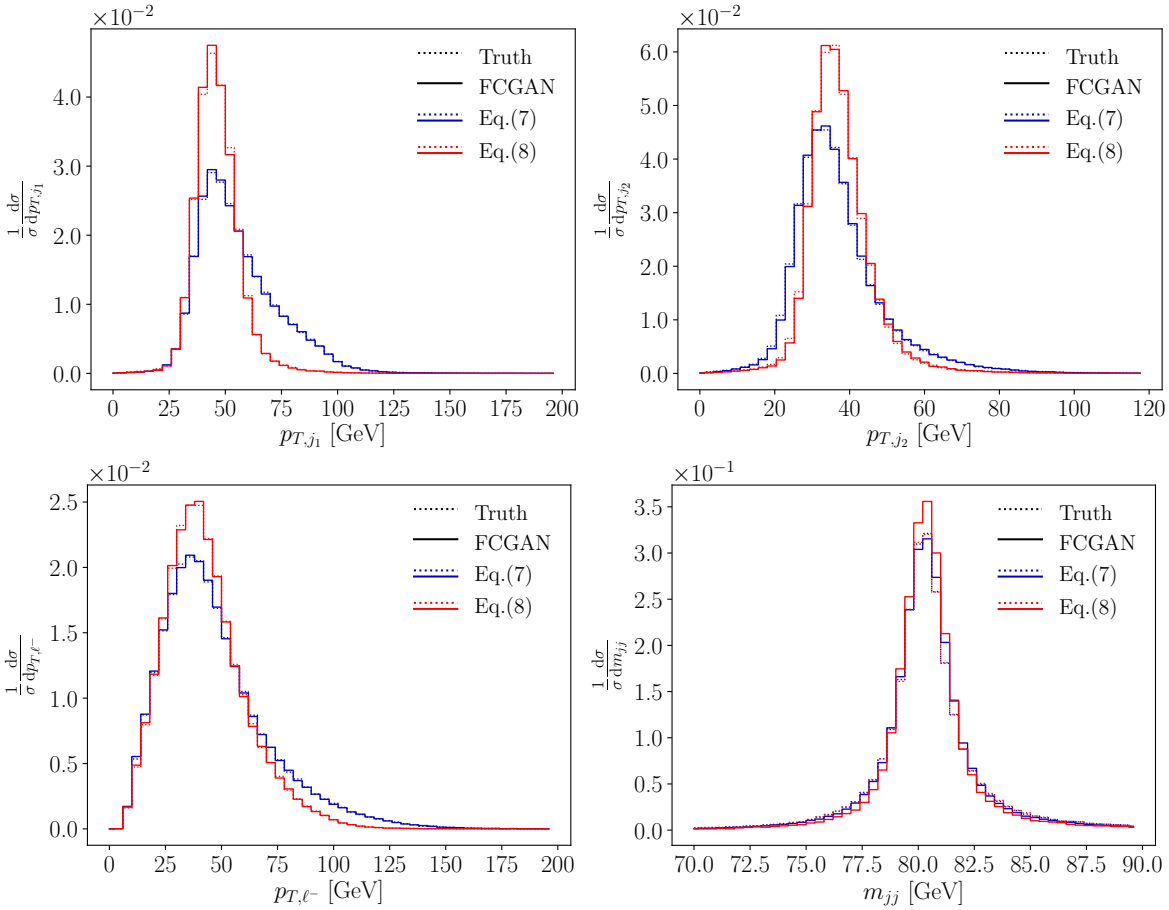

Figure 4.7: Parton level truth and FCGANned distributions when we train the GAN on the full data set but only unfold parts of phase space defined in Eq.(4.7) and Eq.(4.8). To be compared with the naive GAN results in Fig.4.4.

shape our four example distributions. However, we see for example in the $p_{T,jj}$ distribution that the inverted detector-level sample reconstructs the patterns of the true parton-level events perfectly. This comparison indicates that the FCGAN approach deals with differences in the training and test samples very well.

Because physicists and 4-year olds follow a deep urge to break things we move on to harsher cuts on the inclusive event sample. We start with

$$\text{Cut III}: \quad p_{T,j_1} = 30 \ldots 50 \text{ GeV} \quad p_{T,j_2} = 30 \ldots 40 \text{ GeV} \quad p_{T,\ell^-} = 20 \ldots 50 \text{ GeV} , \tag{4.12}$$

which 14% of all events pass. In Fig. 4.8 we see that also for this much reduced fraction of test events corresponding to the training sample the FCGAN inversion reproduces the true distributions extremely well, to a level where it appears not really relevant what fraction of the training and test data correspond to each other.

Finally, we apply a cut which not only removes a large fraction of events, but also cuts into the leading peak feature of the p_{T,j_1} distribution and removes one of the side bands needed for an interpolation,

$$\text{Cut IV}: \quad p_{T,j_1} > 60 \text{ GeV} . \tag{4.13}$$

For this choice 39% of all events pass, but we remove all events at low transverse momentum, as can be seen from Fig. 4.6. This kind of cut could therefore be expected to break the unfolding. Indeed, the red lines in Fig. 4.8 indicate that we have broken the m_{jj} reconstruction through the FCGAN.

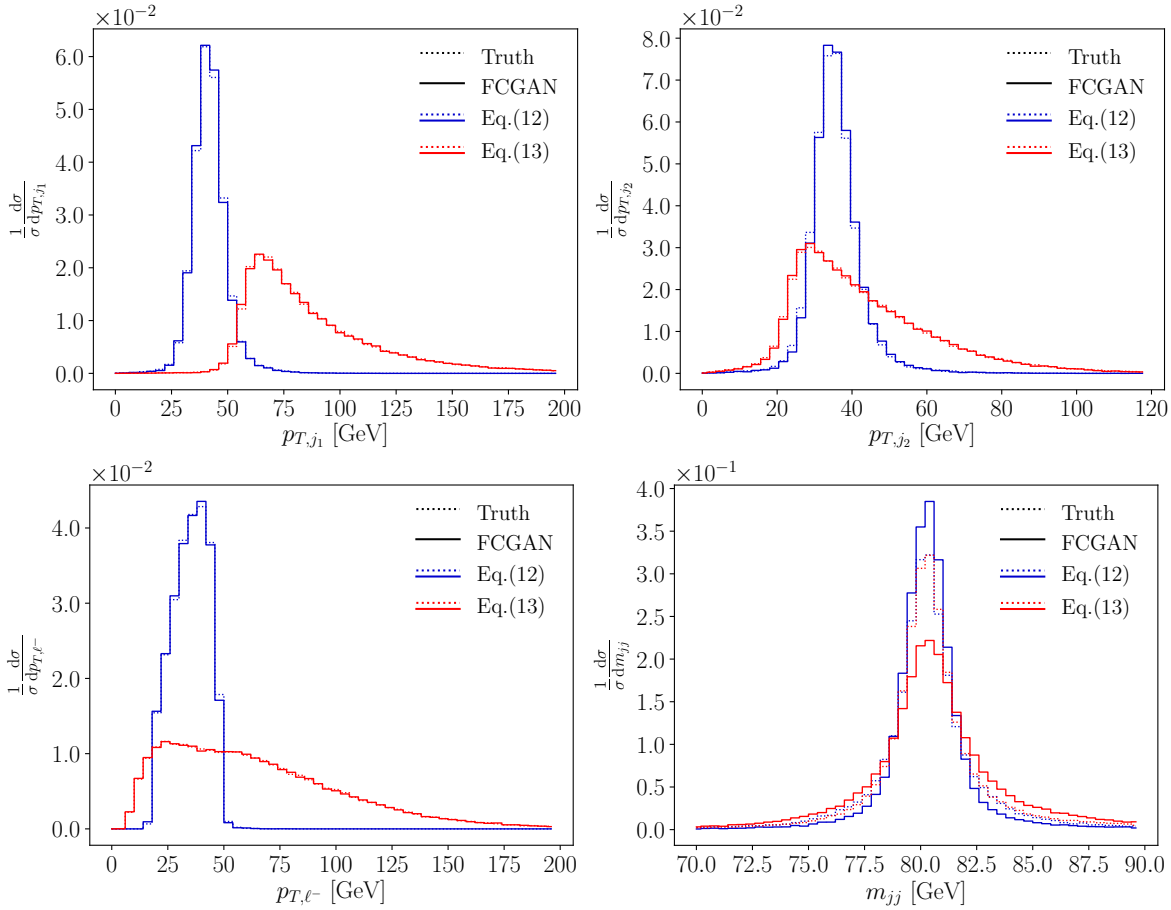

Figure 4.8: Parton level truth and FCGANned distributions when we train the GAN on the full data set but only unfold parts of phase space defined in Eqs.(4.12) and (4.13).

However, all other (shown) distributions still agree with the parton-level truth extremely well. The problem with the invariant mass distribution is that our implementation of the MMD loss is not actually conditional. This can be changed in principle, but standard implementations are somewhat inefficient and the benefit is not obvious at this stage. At this stage it means that, when pushed towards it limits, the network will first fail to reproduce the correct peak width in the m_{jj} distribution, while all other kinematic variables remain stable.

Finally, just like in Ref. [**?**] we show 2-dimensional correlations in Fig. 4.9. We stick to applying the network to the full phase space and show the parton level truth and the FCGAN-inverted events in the two upper panels. Again, we see that the FCGAN reproduces all features of the parton level truth with high precision. The bin-wise relative deviation between the two 2-dimensional distributions only becomes large for small values of E_{j_1}, where the number of training events is extremely small.

4.4 New physics injection

As discussed before, unfolding to a hard process is necessarily model-dependent. Until now, we have always assumed the Standard Model to correctly describe the parton-level and detector-level events. An obvious question is what happens if we train our FCGAN on Standard Model data, but apply it to a different hypothesis. This challenge becomes especially interesting if this alternative hypothesis differs from the Standard Model in a local phase space effect. It then allows us to test if the generator

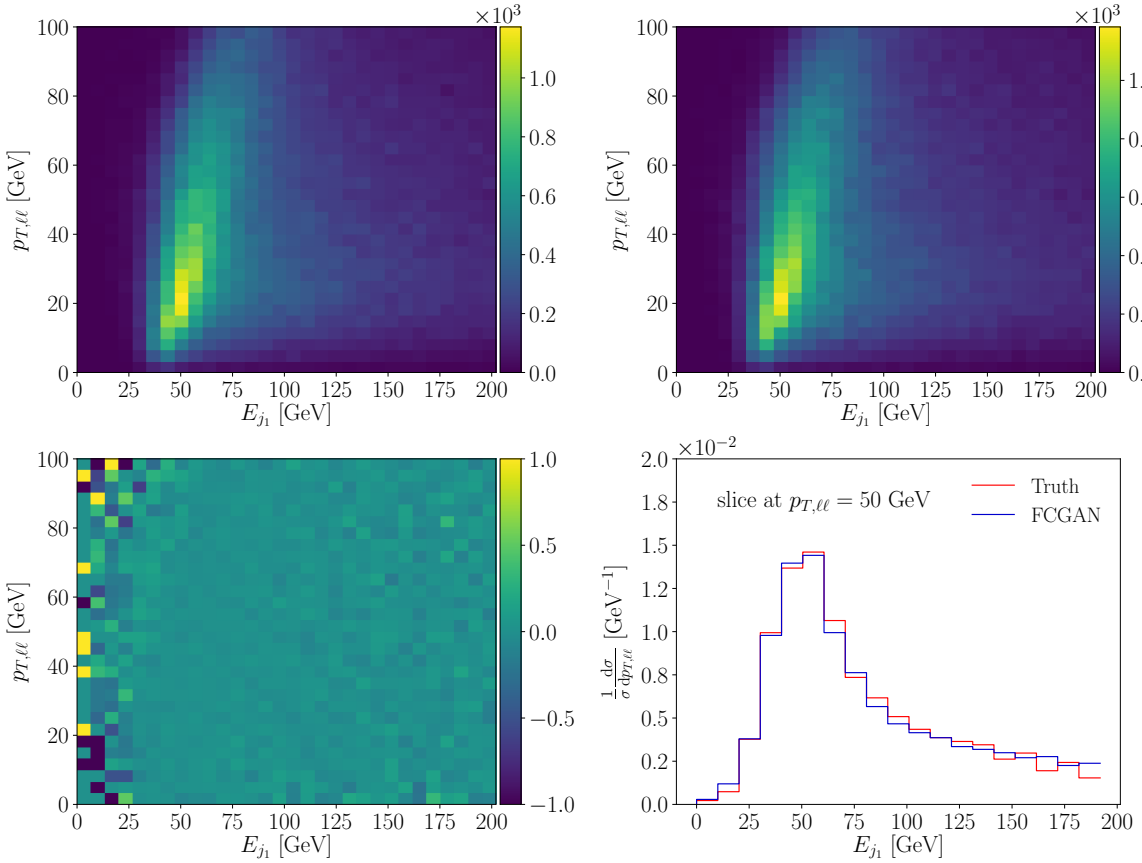

Figure 4.9: Two-dimensional parton level truth (upper left) and FCGANned (upper right) distributions when we train the GAN on the full data set and unfold over the full phase space. The lower panels show the relative deviation between truth and FCGANned and the one-dimensional E_{j_1} distribution along fixed $p_{T,\ell\ell}$.

networks maps the parton-level and detector-level phase spaces in a structured manner. Such features of neural networks are at the heart of all variational constructions, for instance variational autoencoders which are structurally close to GANs. Observing them for GAN unfolding could turn into a significant advantage over alternative unfolding methods.

To this end we add a fraction of resonant W' events from a triplet extension of the Standard Model [**?**], representing the hard process

$$pp \to W'^* \to ZW^\pm \to (\ell^-\ell^+)\,(jj) \tag{4.14}$$

to the test data. We simulate these events with madgraph using the model implementation of Ref. [**?**] and denote the new massive charged vector boson with a mass of 1.3 TeV and a width of 15 GeV as W'. For the test sample we combine the usual Standard Model sample with the W'-sample in proportions $90\% - 10\%$. The other new particles do not appear in our process to leading order. Because we want to test how well the GAN maps local phase space structures onto each other, we deliberately choose a small width $\Gamma_{W'}/M_{W'} \sim 1\%$, not exactly typical for such strongly interacting triplet extensions.

The results for this test are shown in Fig. 4.10. First, we look at transverse momentum distribution of final-state particles, which are hardly affected by the new heavy resonance. Both, the leading jet and the lepton distributions are essentially identical for both truth levels and the FCGAN output. The same is true for the invariant mass of the hadronically decaying W-boson, which nevertheless provides a useful test of the stability of our training and testing.

Finally, we show the reconstructed W'-mass in the lower-right pane. Here we see the different (nor-

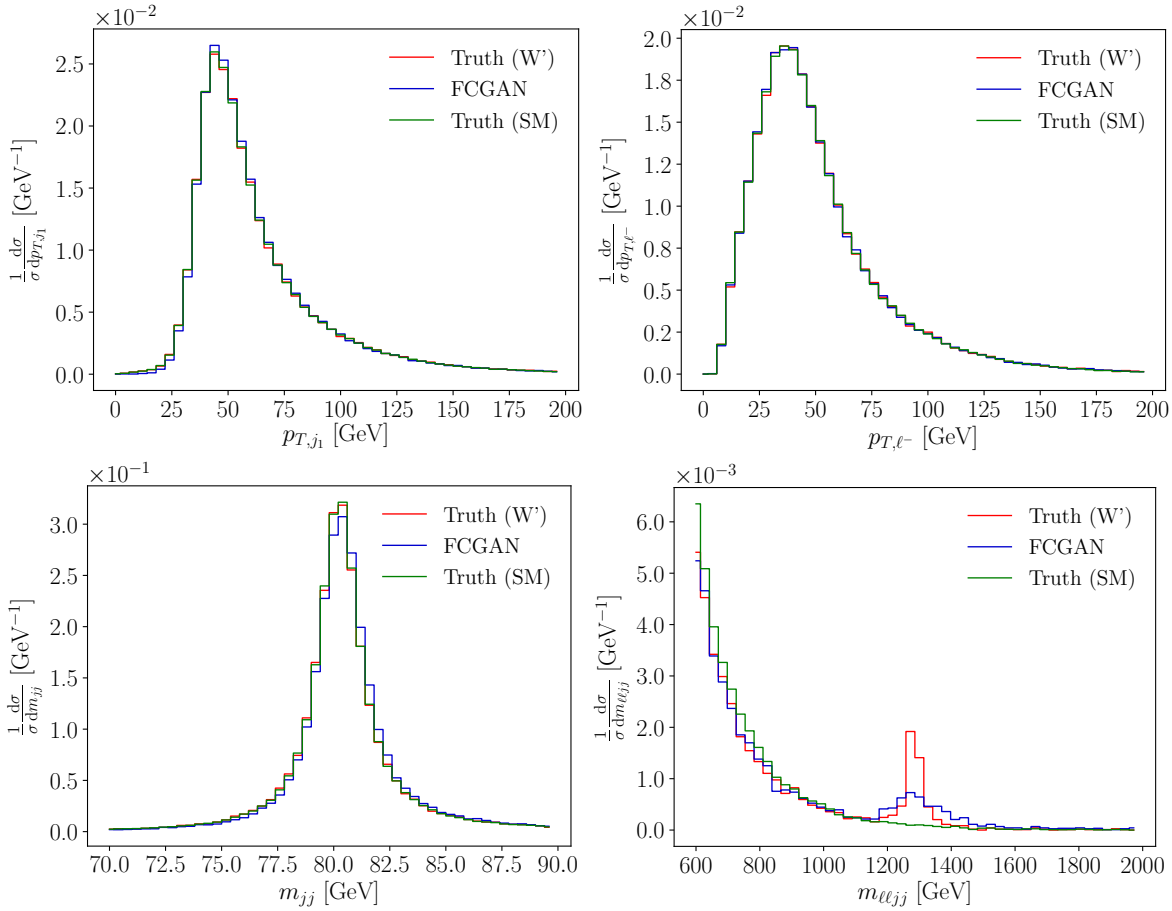

Figure 4.10: Parton level truth and FCGANned distributions when we train the network on the Standard Model only and unfold events with an injection of 10% W' events. The mass of the additional s-channel resonance is 1.3 TeV.

malized) truth-level distributions for the Standard Model and the W'-injected sample. The FCGAN, trained on the Standard Model, keeps track of local phase space structures and reproduces the W' peak faithfully. It also learn the W'-mass as the central peak position very well. The only issue is the W'-width, which the network over-estimates. However, we know already that dynamically generated width distributions are a challenge to GANs and require for instance an MMD loss. Nevertheless, Fig. 4.10 clearly shows that GAN unfolding shows a high degree of model independence, making use of local structures in the mapping between the two phase spaces. We emphasize that the additional mass peak in the FCGANned events is not a one-dimensional feature, but a localized structure in the full phase space. This local structure is a feature of neural networks which comes in addition to the known strengths in interpolation.

4.5 Outlook

We have shown that it is possible to invert a simple Monte Carlo simulation, like a fast detector simulation, with a fully conditional GAN. Our example process is $WZ \to (jj)(\ell\ell)$ at the LHC and we GAN away the effect of standard delphes. A naive GAN approach works extremely well when the training sample and the test sample are very similar. In that case the GAN benefits from the fact that we do not actually need an event-by-event matching of the parton-level and detector-level samples.

If the training and test samples become significantly different we need a fully conditional GAN to

invert the detector effects. It maps random noise parton-level events with conditional, event-by-event detector-level input and learns to generate parton-level events from detector-level events. First, we noticed that the FCGAN with its structured mapping provides much more stable predictions in tails of distributions, where the training sample is statistics limited. Then, we have shown that a network trained on the full phase space can be applied to much smaller parts of phase space, even including cuts in the main kinematic features. The FCGAN successfully maintains a notion of events close to each other at detector level and at parton level and maps them onto each other. This approach only breaks eventually because the MMD loss needed to map narrow Breit-Wigner propagators is not (yet) conditional in our specific setup.

Finally, we have seen that the network reproduces an injected new physics signal as a local structure in phase space. This large degree of model independence reflects another beneficial feature of neural networks, namely the structured mapping of the linked phase spaces.

5 | INN unfolding

Abstract

For simulations where the forward and the inverse directions have a physics meaning, invertible neural networks are especially useful. A conditional INN can invert a detector simulation in terms of high-level observables, specifically for ZW production at the LHC. It allows for a per-event statistical interpretation. Next, we allow for a variable number of QCD jets. We unfold detector effects and QCD radiation to a pre-defined hard process, again with a per-event probabilistic interpretation over parton-level phase space.

5.1 Introduction

The unique feature of LHC physics from a data science perspective is the comparison of vast amounts of data with predictions based on first principles. This modular prediction starts with the Lagrangian describing the hard scattering, then adds perturbative QCD providing precision predictions, resummed QCD describing parton showers and fragmentation, hadronization, and finally a full detector simulation [?]. In this so-defined forward direction all simulation modules are based on Monte Carlo techniques, and in the ideal world we would just compare measured and simulated events and draw conclusions about the hard process. This hard process is where we expect to learn about new aspects of fundamental physics, for instance dark matter, extended gauge groups, or additional Higgs bosons.

Because our simulation chain works only in one direction, the typical LHC analysis starts with a new, theory-inspired hypothesis encoded in a Lagrangian as new particles and couplings. For every point in the new physics parameter space we simulate events, compare them to the measured data using likelihood methods, and discard the new physics hypothesis. This approach is inefficient for a variety of reasons:

1. The best way to compare two hypotheses is the log-likelihood ratio based on new physics and Standard Model predictions for the hard process. Using this ratio in the analysis is the idea behind the matrix element method [?,?,?,?,?,?], but usually this information is not available [?].

2. New physics hypotheses have free model parameters like masses or couplings, even if an analysis happens to be independent of them. If the predicted event rates follow a simple scaling, like for a truncated effective theory, this is simple, but usually we need to simulate events for each point in model space.

3. There is a limit in electroweak or QCD precision to which we can reasonably include predictions in our standard simulation tools. Beyond this limit we can, for instance, only compute a limited set of kinematic distributions, which excludes these precision prediction from standard analyses.

4. Without a major effort it is impossible for model builders to derive competitive limits on a new model by recasting an existing analysis.

All these shortcomings point into the same direction: we need to invert the simulation chain, apply this inversion to the measured data, and compare hypotheses at the level of the hard scattering. For hadronization and fragmentation an approximate inversion is standard in that we always apply jet algorithms to extract simple parton properties from the complex QCD jets. For the detector simulation either at the level of particles or at the level of jets this problem is usually referred to as detector unfolding. For instance in top physics we also unfold kinematic information to the level of the decaying top quarks, assuming that the top decays are correctly described by the standard model [?,?]. Going beyond detector effects we know what for many analyses QCD jet radiation adds little to our new physics search. This is certainly true whenever soft and collinear radiation can be simulated by spin-averaged parton showers depending only logaritmically on the global energy scale of the hard process. In that case we should also be able to also unfold QCD jet radiation as the last simulation step. This is the final goal of our paper.

Technically, we propose to use invertible networks (INNs) [?,?,?] to invert part of the LHC simulation chain. This application builds on a long list of one-directional applications of generative or similar networks to LHC simulations, including phase space integration [?,?], amplitudes [?,?], event generation [?,?,?,?,?], event subtraction [?], detector simulations [?,?,?,?,?,?,?,?,?], parton showers [?,?,?,?], or searches for physics beyond the Standard Model [?]. INNs are an alternative class of generative networks, based on normalizing flows [?,?,?,?]. In particle physics such normalizing flow networks have proven useful for instance in phase space generation [?], linking integration with generation [?,?], or anomaly detection [?].

Our INN study on unfolding detector-level events [?] to the hard scattering builds on similar attempts with a standard GAN [?] and a fully conditional GAN analysis [?]. In Sec. 5.3 we show how the bijective structure of the INN makes their training especially stable. If we add sufficiently many

random numbers to the INN we can start generating probability distributions in the parton-level phase space. The conditional INN (cINN) [?, ?] adds even more sampling elements to the generation of unfolded configurations. For arbitrary kinematic distributions we can test the calibration of this generative network output using truth information and find that unlike GANs the cINN lives up to its generative promise: for a single detector-level event the cINN generates probability distributions in the multi-dimensional parton-level phase space.

Next, we show in Sec. 5.4 how the inversion can link two phase spaces with different dimension. This allows us to unfold based on a model with a variable number of final state particles at the detector level and is crucial to include higher-order perturbative corrections. We show how the cINN can account for jet radiation and unfolds it together with the detector effects. In other words, the network distinguishes between jets from the hard process and jets from QCD radiation and it also unfolds the kinematic modifications from initial state radiation, to provide probability distributions in the parton-level phase space of a hard process.

We note that our examples only cover analyses where subjet information factorizes from the hard process, for instance in terms of (mis-)tagging efficiencies. For analyses going beyond this level, like searches for long-lived particles, we need to skip the jet algorithm stage and instead include the full calorimeter and tracking information. In principle and assuming the availability of a proper detector simulations our ideas might still work for these applications, but for the time being we ignore these complications.

5.2 Unfolding basics

Unfolding particle physics events is a classic example for an inverse problem [?, ?, ?]. In the limit where detector effects can be described by Gaussian noise, it is similar to unblurring images. However, actual detector effects depend on the individual objects, the global energy deposition per event, and the proximity of objects, which means they are much more complicated than Gaussian noise. The situation gets more complicated when we add effects like QCD jet radiation, where the radiation pattern depends for instance on the quantum numbers of the incoming partons and on the energy scale of the hard process.

What we do know is that we can describe the measurement of phase space detector-level distributions $\mathrm{d}\sigma/\mathrm{d}x_d$ as a random process, just as the detector effects or jet radiation can be simulated by a set of random numbers describing a Markov process. This means that also the inversion or extraction of the parton-level distribution $\mathrm{d}\sigma/\mathrm{d}x_p$ is a statistical problem.

5.2.1 Binned toy model and locality

As a one-dimensional toy example we can look at a binned (parton-level) distribution $\sigma_j^{(p)}$ which gets transformed into another binned (detector-level) distribution $\sigma_j^{(d)}$ by the kernel or response function g_{ij},

$$\sigma_i^{(d)} = \sum_{j=1}^{N} g_{ij} \sigma_j^{(p)} \; . \tag{5.1}$$

We can postulate the existence of an inversion with the kernel $\bar{g}$ through the relation

$$\sigma_k^{(p)} = \sum_{i=1}^{N} \bar{g}_{ki} \sigma_i^{(d)} = \sum_{j=1}^{N} \left(\sum_{i=1}^{N} \bar{g}_{ki} g_{ij} \right) \sigma_j^{(p)} \quad \text{with} \quad \sum_{i=1}^{N} \bar{g}_{ki} g_{ij} = \delta_{kj} \; . \tag{5.2}$$

If we assume that we know the N^2 entries of the kernel g, this form gives us the N^2 conditions to compute its inverse $\bar{g}$. We illustrate this one-dimensional binned case with a semi-realistic smearing

matrix

$$g = \begin{pmatrix} 1-x & x & 0 \\ x & 1-2x & x \\ 0 & x & 1-x \end{pmatrix} . \tag{5.3}$$

We illustrate the smearing pattern with two input vectors, keeping in mind that in an unfolding problem we typically only have one kinematic distribution to determine the inverse matrix $\bar{g}$,

$$\begin{aligned} \sigma^{(p)} = n \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \quad &\Rightarrow \quad \sigma^{(d)} = \sigma^{(p)} , \\ \sigma^{(p)} = \begin{pmatrix} 1 \\ n \\ 0 \end{pmatrix} \quad &\Rightarrow \quad \sigma^{(d)} = \sigma^{(p)} + x \begin{pmatrix} n-1 \\ -2n+1 \\ n \end{pmatrix} . \end{aligned} \tag{5.4}$$

The first example shows how for a symmetric smearing matrix a flat distribution removes all information about the detector effects. This implies that we might end up with a choice of reference process and phase space such that we cannot extract the detector effects from the available data. The second example illustrates that for bin migration from a dominant peak the information from the original $\sigma^{(p)}$ gets overwhelmed easily. We can also compute the inverse of the smearing matrix in Eq.(5.3) and find

$$\bar{g} \approx \frac{1}{1-4x} \begin{pmatrix} 1-3x & -x & x^2 \\ -x & 1-2x & -x \\ x^2 & -x & 1-3x \end{pmatrix} , \tag{5.5}$$

where we neglect the sub-leading x^2-terms whenever there is a linear term as well. The unfolding matrix extends beyond the nearest neighbor bins, which means that local detector effects lead to a global unfolding matrix and unfolding only works well if we understand our entire data set. The reliance on useful kinematic distributions and the global dependence of the unfolding define the main challenges once we attempt to unfold the full phase space of an LHC process.

5.2.2 Bayes' theorem and model dependence

Over the continuous phase space a detector simulation can be written as

$$\frac{\mathrm{d}\sigma}{\mathrm{d}x_d} = \int \mathrm{d}x_p \; g(x_d, x_p) \; \frac{\mathrm{d}\sigma}{\mathrm{d}x_p} , \tag{5.6}$$

where x_d is a kinematic variable at detector level, x_p the same variable at parton level, and g a kernel or transfer function which links these two arguments. We ignore efficiency factors for now, because they can be absorbed into the parton-level rate. To invert the detector simulation we define a second transfer function $\bar{g}$ such that [**?**, **?**, **?**]

$$\frac{\mathrm{d}\sigma}{dx_p} = \int \mathrm{d}x_d \; \bar{g}(x_p, x_d) \; \frac{\mathrm{d}\sigma}{\mathrm{d}x_d} = \int \mathrm{d}x'_p \; \frac{\mathrm{d}\sigma}{dx'_p} \int \mathrm{d}x_d \; \bar{g}(x_p, x_d) g(x_d, x'_p) . \tag{5.7}$$

This inversion is fulfilled if we construct the inverse $\bar{g}$ of g defined by

$$\int \mathrm{d}x_d \; \bar{g}(x_p, x_d) g(x_d, x'_p) = \delta(x_p - x'_p) , \tag{5.8}$$

all in complete analogy to the binned form above. The symmetric form of Eq.(5.6) and Eq.(5.7) indicates that g and $\bar{g}$ are both defined as distributions. In the g-direction we use Monte Carlo simulation and sample in x_p, while $\bar{g}$ needs to be sampled in $g(x_p)$ or x_d. In both directions this statistical nature implies that we should only attempt to unfold sufficiently large event samples.

The above definitions can be linked to Bayes' theorem if we identify the kernels with probabilities. We now look at $\bar{g}(x_d|x_p)$ in the slightly modified notation as the probability of observing x_d given

the model prediction x_p and $g(x_p|x_d)$ gives the probability of the model x_p being true given the observation x_d [?,?]. In this language Eq.(5.6) and (5.7) describe conditional probabilities, and we can write something analogous to Bayes' theorem,

$$\bar{g}(x_p|x_d)\,\frac{\mathrm{d}\sigma}{\mathrm{d}x_d} \sim g(x_d|x_p)\,\frac{\mathrm{d}\sigma}{\mathrm{d}x_p}\,. \tag{5.9}$$

In this form $\bar{g}(x_p|x_d)$ is the posterior, $g(x_d|x_p)$ as a function of x_p is the likelihood, $\mathrm{d}\sigma/\mathrm{d}x_p$ is the prior, and the model evidence $\mathrm{d}\sigma/\mathrm{d}x_d$ fixes the normalization of the posterior. From standard Bayesian analyses we know two things: (i) the posterior will in general depend on the prior, in our case the kinematics of the underlying particle physics process or model; (ii) when analyzing high-dimensional spaces the prior dependence will vanish when the likelihood develops a narrow global maximum.

If the posterior $\bar{g}(x_p|x_d)$ in general depends on the model $\mathrm{d}\sigma/\mathrm{d}x_p$, then Eq.(5.7) does not look useful. On the other hand, Bayesian statistics is based on the assumption that the prior dependence of the posterior defines an iterative process where we start from a very general prior and enter likelihood information step by step to finally converge on the posterior. The same approach can define a kinematic unfolding algorithm [?]. We will not discuss these methods further, but come back to this model dependence throughout our paper.

5.2.3 Reference process $pp \to ZW$

To provide a quantitative estimate of unfolding with an invertible neural networks we use the same example process as in Ref. [?],

$$pp \to ZW^\pm \to (\ell^-\ell^+)\,(jj)\,, \tag{5.10}$$

One of the contributing Feynman diagrams is shown in Fig. 5.1. With jets and leptons in the final state we can test the stability of the unfolding network for small and for large detector effects. We generate the ZW events using madgraph [?] without any generation cuts and then simulate parton showering with pythia [?] and the detector effects with delphes [?] using the standard ATLAS card. For jet clustering we use the anti-k_T algorithm [?] with $R = 0.6$ implemented in fastjet [?]. All jets are required to have

$$p_{T,j} > 25\ \text{GeV} \qquad \text{and} \qquad |\eta_j| < 2.5\,. \tag{5.11}$$

For the hadronically decaying W-boson the limited calorimeter resolution will completely dominate over the parton-level Breit-Wigner distribution. After applying the cuts we have 320k events which we split into 90% training and 10% test data.

In a first step, corresponding to Ref. [?] we are only interested in inverting these detector effects. These results are shown in Sec. 5.3. For the simulation this implies that we switch off initial state radiation as well as underlying event and pile-up effects and require exactly two jets and a pair of same-flavor opposite-sign leptons. The jets and corresponding partons are separately ordered by p_T. The detector and parton level leptons are assigned by charge. This gives us two samples matched event by event, one at the parton level (x_p) and one including detector effects (x_d). Each of them is given as an

Figure 5.1: Sample Feynman diagram contributing to ZW production, with intermediate on-shell particles labeled.

unweighted set of four 4-vectors. These 4-vectors can be simplified if we assume all external particles at the parton level to be on-shell. Obviously, this method can be easily adapted to weighted events.

In a second step we include initial state radiation and allow for additional jets in Sec. 5.4. We still require a pair of same-flavor opposite-sign leptons and at least two jets in agreement with the condition in Eq.(5.11). The four jets with highest p_T are then used as input to the network, ordered by p_T. Events with less than 4 jets are zero-padded. This second data set is only used for the conditional INN.

5.3 Unfolding detector effects

We introduce the conditional INN in two steps, starting with the non-conditional, standard setup. The construction of the INN we use in our analysis combines two goals [**?**]:

1. the mapping from input to output is invertible and the Jacobians for both directions are tractable;

2. both directions can be evaluated efficiently. This second property goes beyond some other implementations of normalizing flow [**?**, **?**].

While the final aim is not actually to evaluate our INN in both directions, we will see that these networks can be extremely useful to invert a Markov process like detector smearing. Their bi-directional training makes them especially stable.

In Sec. 5.3.3 we will show how the conditional INN retains a proper statistical notion of the inversion to parton level phase space. This avoids a major weakness of standard unfolding methods, namely that they only work on large enough event samples condensed to one-dimensional or two-dimensional kinematic distributions. This could be a missing transverse energy distribution in mono-jet searches or the rapidities and transverse momenta in top pair production. To avoid systematics or biases in the full phase space coverage required by the matrix element method, the unfolding needs to construct probability distributions in parton-level phase space, including small numbers of events in tails of kinematic distributions.

5.3.1 Naive INN

While it is clear from our discussion in Ref. [**?**] that a standard INN will not serve our purpose, we still describe it in some detail before we extend it to a conditional network. Following the conventions of our GAN analysis and in analogy to Eqs.(5.6) to (5.8) we define the network input as a vector of hard process information $x_p \in R^{D_p}$ and the output at detector level via the vector $x_d \in \mathbb{R}^{D_d}$. If the dimensionalities of the spaces are such that $D_p < D_d$ we add a noise vector r with dimension $D_d - D_p$ to define the bijective, invertible transformation,

$$\begin{pmatrix} x_p \\ r \end{pmatrix} \overset{[}{\leftarrow} \text{unfolding} : \bar{g}]\textsc{Pythia},\textsc{Delphes} : g \rightarrow\!\!\leftarrow\!\!\longrightarrow x_d \,. \tag{5.12}$$

A correctly trained network g with the parameters θ then reproduces x_d from the combination x_p and r. Its inverse $\bar{g}$ instead reproduces the combination of x_p and r from x_d.

The defining feature of the INN illustrated in Fig. 5.2 is that it learns both directions of the bijective mapping in parallel and encodes them into one network. Such a simultaneous training of both directions is guaranteed by the building blocks of the network, the invertible coupling layers [**?**, **?**]. For notational purposes we ignore the random numbers in Eq.(6.4) and assume that this layer links an input vector x_p to an output vector x_d after splitting both of them in halves, $x_{p,i}$ and $x_{d,i}$ for $i = 1, 2$. The relation between input and output is given by a sub-network, which encodes arbitrary functions $s_{1,2}$ and $t_{1,2}$. Using an element-wise multiplication $\odot$ and sum one could for instance define an output $x_{d,1}(x_p) = x_{p,1} \odot s_2(x_{p,2}) + t_2(x_{p,2})$. In order to avoid numerical instabilities caused by the division

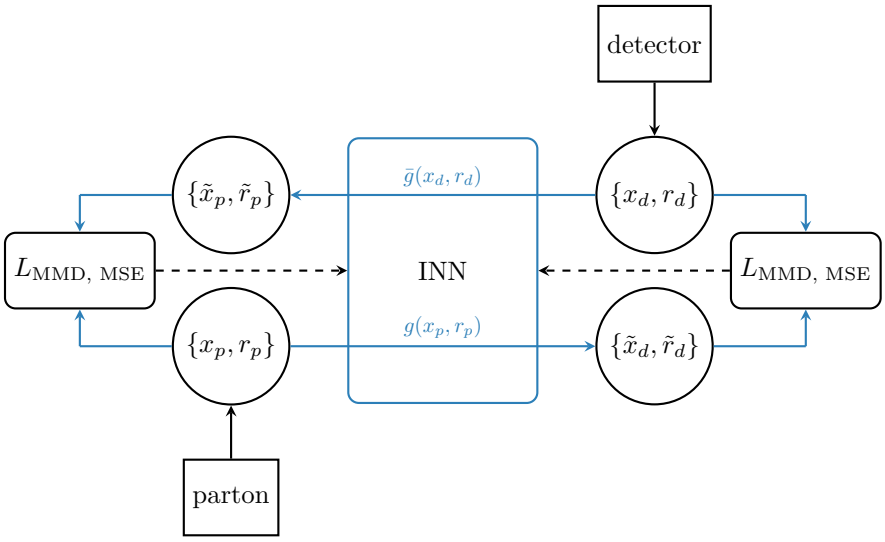

Figure 5.2: Structure of INN. The $\{x_{d,p}\}$ denote detector-level and parton-level events, $\{r_{d,p}\}$ are random numbers to match the phase space dimensionality. A tilde indicates the INN generation.

with $s(x)$ in the inverse direction, we include an exponential to obtain

$$\begin{pmatrix} x_{d,1} \\ x_{d,2} \end{pmatrix} = \begin{pmatrix} x_{p,1} \odot e^{s_2(x_{p,2})} + t_2(x_{p,2}) \\ x_{p,2} \odot e^{s_1(x_{d,1})} + t_1(x_{d,1}) \end{pmatrix} \quad \Leftrightarrow \quad \begin{pmatrix} x_{p,1} \\ x_{p,2} \end{pmatrix} = \begin{pmatrix} (x_{d,1} - t_2(x_{p,2})) \odot e^{-s_2(x_{p,2})} \\ (x_{d,2} - t_1(x_{d,1})) \odot e^{-s_1(x_{d,1})} \end{pmatrix} . \tag{5.13}$$

By construction, this inversion works independent of the form of s and t. If we write the coupling block function as $g(x_p) \sim x_d$, again omitting the random numbers r, the Jacobian of the network function has a triangular form

$$\frac{\partial g(x_p)}{\partial x_p} = \begin{pmatrix} \text{diag } e^{s_2(x_{p,2})} & \text{finite} \\ 0 & \text{diag } e^{s_1(x_{d,1})} \end{pmatrix} , \tag{5.14}$$

so its determinant is easy to compute. Such coupling layer transformations define a so-called normalizing flow, when we view it as transforming an initial probability density into a very general form of probability density through a series of invertible steps. We can relate the two probability densities as long as the Jacobians of the individual layers can be efficiently calculated.

Since the first use of the invertible coupling layer, much effort has gone into improving its efficiency. The All-in-One (AIO) coupling layer includes two features, introduced by Ref. [**?**] and Ref. [**?**]. The first modification replaces the transformation of $x_{p,2}$ by a permutation of the output of each layer. Due to the permutation each component still gets modified after passing through several layers. The second modification includes a global affine transformation to include a global bias and linear scaling that maps $x \to sx + b$. Finally, we apply a bijective soft clamping after the exponential function in Eq.(5.13) to prevent instabilities from diverging outputs.

The INN in our simplified example combines three contributions to the loss function. First, it tests if in the DELPHES direction of Eq.(6.4) we indeed find $g(x_p) = x_d$ via the mean squared error (MSE) function. While this is theoretically sufficient to obtain the inverse function, also testing the inverse direction $\bar{g}(x_d) = x_p$ greatly improves the efficiency and stability of the training. Third, to resolve special sharp features like the invariant mass of intermediate particles we use the maximum mean discrepancy (MMD) as a distance measure between the generated and real distribution of these features.

Because we will also use the MMD in another function function [**?**] we review it briefly. An MMD loss allows us to compare any pre-defined distribution. For a relativistic phase space a critical narrow phase space feature is the invariant mass of intermediate particles. We can force the network to consider this one-dimensional distribution of the 4-vectors x_p for batches of parton-level and detector-level events,

$$\text{MMD} = \left[\langle k(x, x') \rangle_{x,x' \sim P_p} + \langle k(y, y') \rangle_{y,y' \sim P_d} - 2 \langle k(x, y) \rangle_{x \sim P_p, y \sim P_d} \right]^{1/2} . \tag{5.15}$$

In Refs. [**?**] and [**?**] we compare common choices, like Gaussian or Breit-Wigner kernels

$$k_{\text{Gauss}}(x,y) = \exp \frac{-(x-y)^2}{2\sigma^2} \qquad \text{or} \qquad k_{\text{BW}}(x,y) = \frac{\sigma^2}{(x-y)^2 + \sigma^2} \tag{5.16}$$

with a fixed or variable width σ [**?**]. Inside the INN architecture the Breit-Wigner kernel is the best choice to analyze the distribution of the random numbers as part of the loss function [**?**].

We now use the INN network to map parton-level events to detector-level events or vice-versa. In a statistical analysis we then use standard kinematic distributions and compare the respective truth and INN-inverted shapes for both directions. The left panels of Fig. 5.3 shows the transverse momentum distributions of the two jets and their invariant mass for both directions of the INN. The truth events at parton level and at detector level are marked as dashed lines. Starting from each of the truth events we can apply the INN describing the detector effects as $x_d = g(x_p)$ or unfolding the detector effects as $x_p = \bar{g}(x_d)$ in Eq.(6.4). The corresponding solid lines have to be compared to the dotted truth lines, where we need to keep in mind that at the parton level the relevant objects are quarks while at the detector level they are jets.

For the leading jet the truth and INNed detector-level agree very well, while for the second jet the naive INN fails to capture the hard cut imposed by the jet definition. For the invariant mass we find that the smearing due to the detector effects is reproduced well with some small deviations in the tails. In the unfolding direction both p_T distributions follow the parton level truth. The only difference is a systematic lack of events in the tail for the second quark. This is especially visible in the ratio of the INN-unfolded events and the parton-level truth, indicating that also at small p_T the network does not fill the phase space sufficiently. Combining both directions we see that in forward direction the INN produces a too broad p_T-distribution, the unfolding direction of the INN produces a too narrow distribution. The conceptual advantage of the INN actually implies a disadvantage for the inversion of particular difficult features. Finally, the invariant mass of the W is reproduced perfectly without any systematic deviation.

5.3.2 Noise-extended INN

While our simplified example in the previous section shows some serious promise of INNs, it fails to incorporate key aspects of the physical process. First of all, the number of degrees of freedom is not actually the same at parton level and at detector level. External partons are on their mass shell, while jets come with a range of jet masses. This mismatch becomes crucial when we include missing transverse momentum in the signature. We generally need fewer parameters to describe the partonic scattering than the detector-level process. For a fixed set of parton-level momenta we usually smear each momentum component to simulate the detector measurement. These additional degrees of freedom are of stochastic nature, so adding Gaussian random variable on the parton side of the INN could be a first step to address this problem.

To also account for potentially unobservable degrees of freedom at the parton level we extend each side of the INN by a random number vector. The mapping in Eq.(6.4) now includes two random number vectors with dimensions $D_{r_d} = D_p$ and $D_{r_p} = D_d$,

$$\begin{pmatrix} x_p \\ r_p \end{pmatrix} \xleftarrow{[\text{ unfolding} : \bar{g}]\text{PYTHIA,DELPHES} : g} \begin{pmatrix} x_d \\ r_d \end{pmatrix} . \tag{5.17}$$

In addition, a pure MSE loss can not capture the fact that the additional noise generates a distribution of detector-level events given fixed parton momenta. It would just predict of a mean value of this distribution and minimize the effect of the noise. A better solution is an MMD loss for each degree of freedom in the event and the masses of intermediate particles, as well as the Gaussian random variables. On the side of the random numbers this MMD loss ensures that they really only encode noise. Again it is beneficial for the training to use the inverse direction and apply additional MMD losses to the parton level events as well as the corresponding Gaussian inputs. Finally we add a weak MSE loss on the four vectors of each side to stabilize the training.

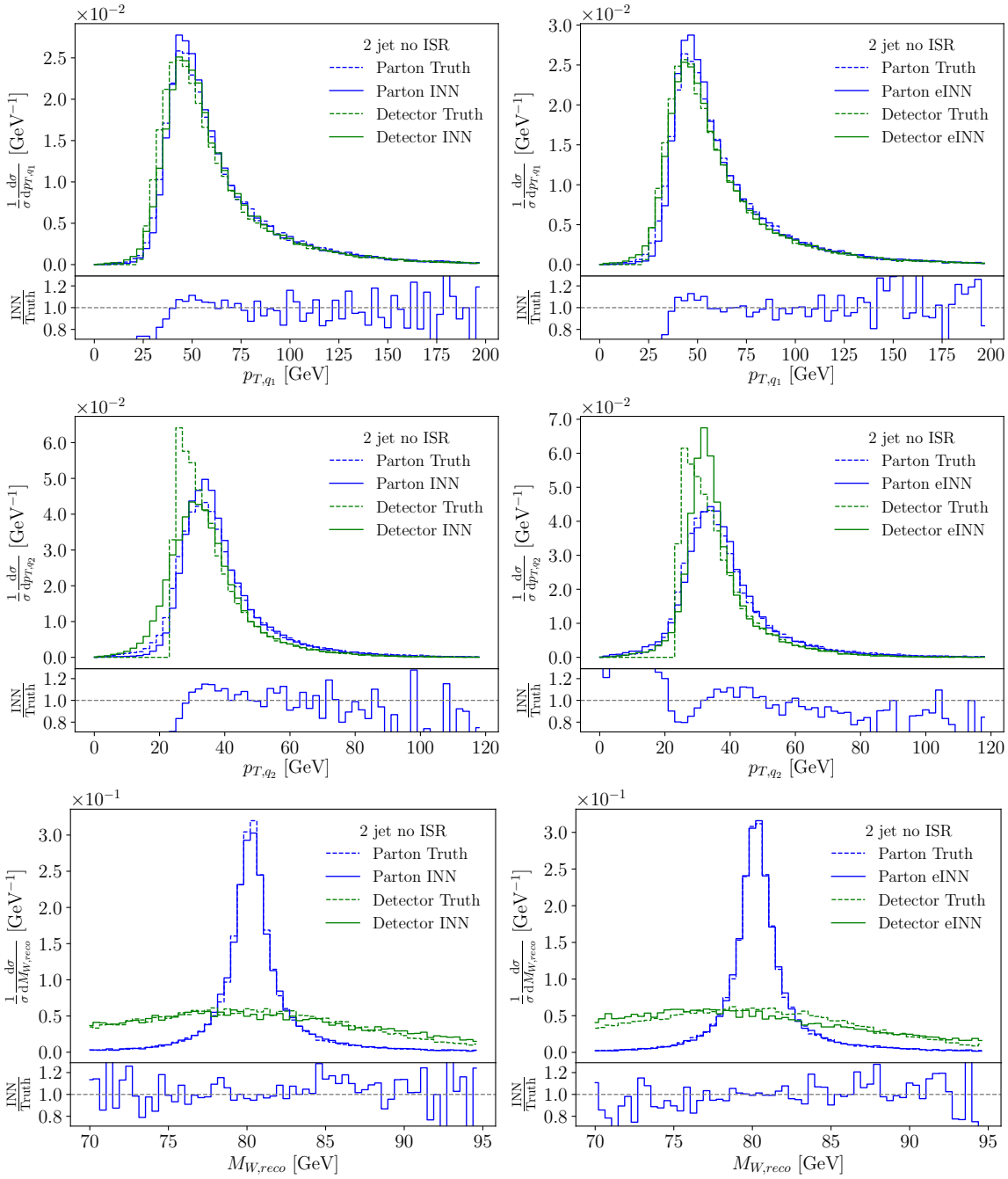

Figure 5.3: INNed $p_{T,q}$ and $M_{W,\text{reco}}$ distributions from a naive INN (left) and the noise-extended eINN (right). In green we compare the detector-level truth to INNed events transformed from parton level. In blue we compare the parton-level truth to INNed events transformed from detector level. The secondary panels show the ratio of INNed events over parton-level truth. More distributions can be found in the pdf files submitted to the arXiv.

In the right panels of Fig. 5.3 we show results for this noise-extended INN (eINN). The generated distributions are similar to the naive INN case and match the truth at the parton level. A notable difference appears in the second jet, the weak spot of the naive INN. The additional random numbers and MMDs provide more freedom to generate the peak in the forward direction and also improve the unfolding in the low-p_T and high-p_T regimes.

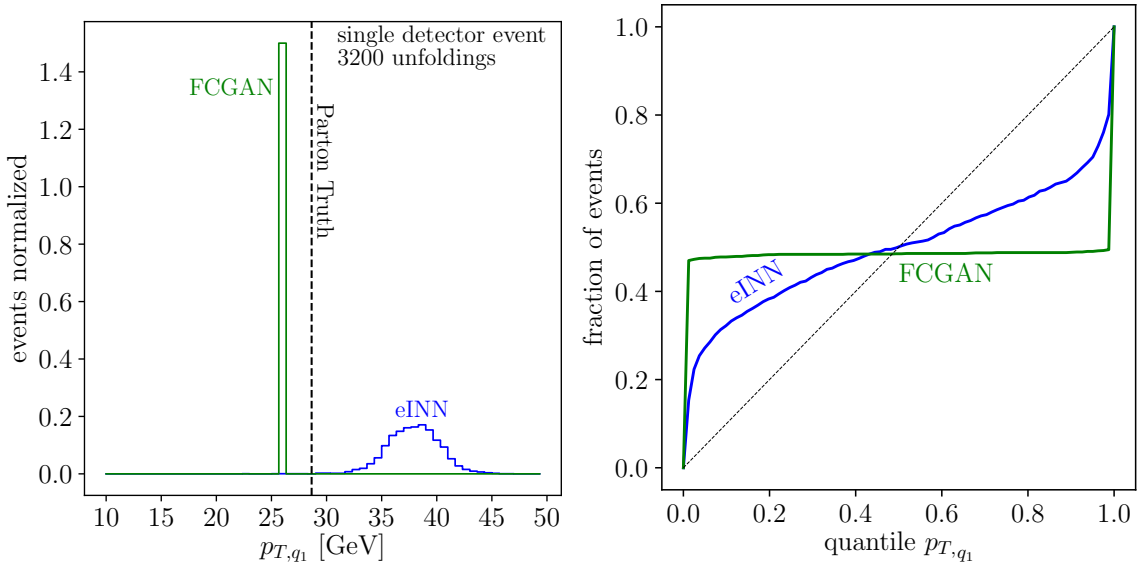

Figure 5.4: Left: illustration of the statistical interpretation of unfolded events for one event. Right: calibration curves for p_{T,q_1} extracted from the FCGAN and the noise-extended eINN.

Aside from the better modeling, the noise extension allows for a statistic interpretation of the generated distributions and a test of the integrity of the INN-inverted distributions. In the left panel of Fig. 5.4 we illustrate the goal of the statistical treatment: we start from a single event at the detector level and generate a set of unfolded events. For each of them we evaluate for instance p_{T,q_1}. Already in this illustration we see that the GAN output is lacking a statistical behavior at the level of individual events, while the noise-extended eINN returns a reasonable distribution of unfolded events.

To see if the width of this INN output is correct we take 1500 parton-level and detector-level event pairs and unfold each event 60 times, sampling over the random variables. This gives us 1500 combinations like the one shown in the left panel of Fig. 5.4: a single parton-level truth configuration and a distribution of the INNed configuration. To see if the central value and the width of the INNed distribution can be interpreted statistically as a posterior probability distribution in parton phase space we analyse where the truth lies within the INN distribution for each of the 1500 events. For a correctly calibrated curve we start for instance from the left of the kinematic distribution and expect 10% of the 1500 events in the 10% quantile of the respective probability distribution, 20% of events in the 20% quantile, etc. The corresponding calibration curves for the noise-extended eINN are shown in the right panel of Fig. 5.4. While they indicate that we can attempt a statistical interpretation of the INN unfolding, the calibration is not (yet) perfect. A steep rise for the lower quantile indicates that too many events end up in the first 10% quantile. In other words, the distributions we obtain by sampling over the Gaussian noise for each event are too narrow.

While our noise-extended eINN takes several steps in the right direction, it still faces major challenges: the combination of many different loss functions is sensitive to their relative weights; the balance between MSE and MMD on event constituents has to be calibrated carefully to generate reasonable quantile distributions; when we want to extend the INN to include more detector-level information we have to include an equally large number of random variable on the parton level which makes the training very inefficient. This leads us again [**?**] to adopt a conditional setup.

5.3.3 Conditional INN

If a distribution of parton-level events can be described by n degrees of freedom, we should be able to use normalizing flows or an INN to map a n-dimensional random number vector onto parton-level 4-momenta. To capture the information from the detector-level events we need to condition the INN

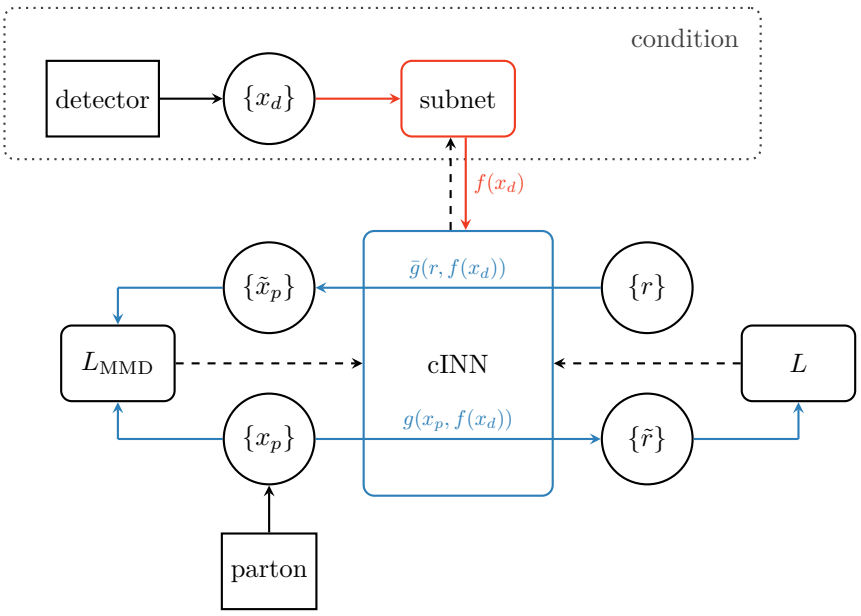

Figure 5.5: Structure of the conditional INN. The input are random numbers $\{r\}$ while $\{x_{d,p}\}$ denote detector-level and parton-level data. The latent dimension loss L follows Eq.(5.18), a tilde indicates the INN generation.

on these events [?, ?, ?], so we link the parton-level data x_p to random noise r under the condition of x_d. Trained on a given process the network should now be able to generate probability distributions for parton-level configurations given a detector-level event and an unfolding model. We note that the cINN is still invertible in the sense that it includes a bi-directional training from Gaussian random numbers to parton-level events and back. While this bi-directional training does not represent the inversion of a detector simulation anymore, it does stabilize the training by requiring the noise to be Gaussian.

A graphic representation of this conditional INN or cINN is given in Fig. 5.5. We first process the detector-level data by a small subnet, i.e. $x_d \to f(x_d)$, to optimize its usability for the cINN [?]. The subnet is trained alongside the cINN and does not need to be reversed or adapted. We choose a shallow and wide architecture of two layers with a width of 1024 internally, because four layers degrade already

Parameter	INN	eINN
Blocks	24	24
Layers per block	2	2
Units per layer	256	256
Trainable weights	$\sim$ 150k	$\sim$ 270k
Epochs	1000	1000
Learning rate	$8 \cdot 10^{-4}$	$8 \cdot 10^{-4}$
Batch size	512	512
Training/testing events	290k / 30k	290k / 30k
Kernel widths	$\sim 2, 8, 25, 67$	$\sim 2, 8, 25, 67$
$D_p + D_{r_p}$	$12 + 4$	$12 + 16$
$D_d + D_{r_d}$	$16 + 0$	$16 + 12$
λ_{MMD}	0.1 (masses only)	0.2
λ_{MMD} increase	-	-

Table 5.1: INN and noise-extended eINN setup and hyper-parameters, as implemented in pytorch(v1.2.0) [?].

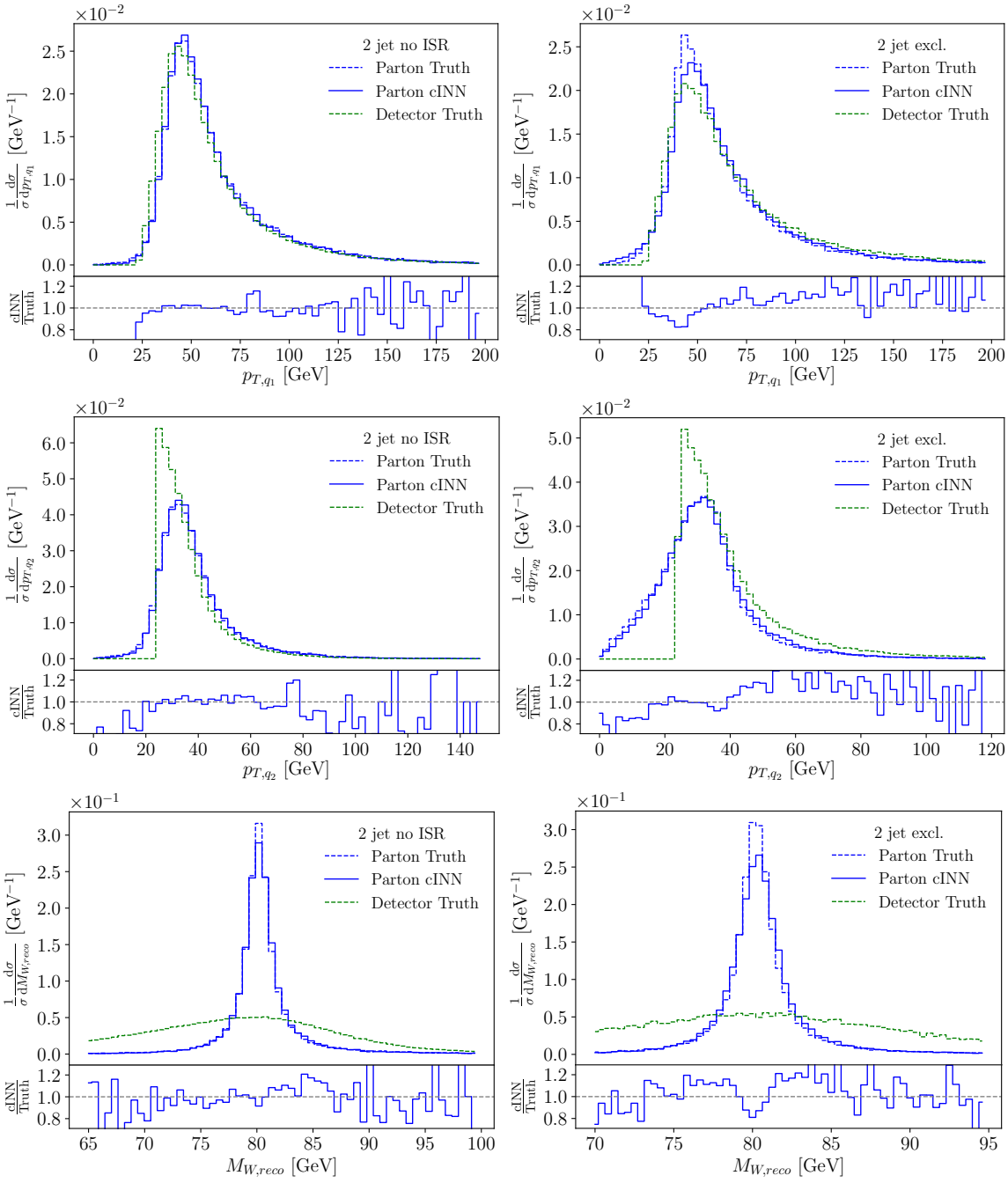

Figure 5.6: cINNed $p_{T,q}$ and $m_{W,\text{reco}}$ distributions. Training and testing events include exactly two jets. In the left panels we use a data set without ISR, while in the right panels we use the two-jet events in the full data set with ISR. The lower panels give the ratio of cINNed to parton-level truth.

the conditional information and allow the cINN to ignore it. When a deeper subnet is required we advertize to use an encoder, which is initialized by pre-training it as part of an autoencoder. We apply this technique when using the larger ISR input, where it leads to a more efficient training. After this preprocessing, the detector information is passed to the functions s_i and t_i in Eq.(5.13), which now depend on the input, the output, and on the fixed condition. Since the invertibility of the network is independent of the values of s_i and t_i, the network remains invertible between the parton-level events $\{x_p\}$ and the random variables $\{r\}$. This feature stabilizes the training. The cINN loss function is

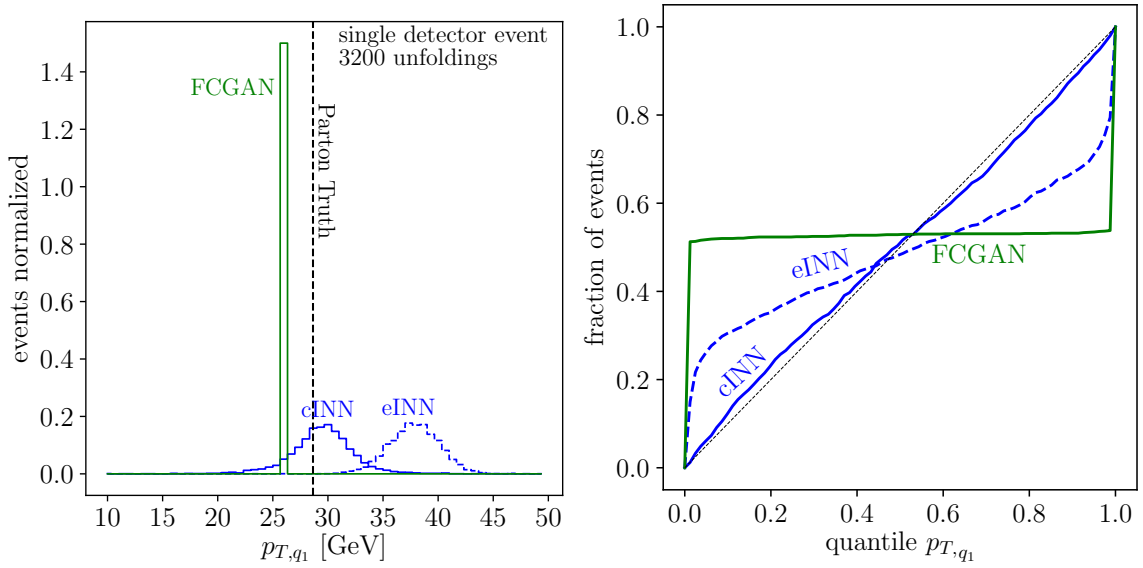

Figure 5.7: Left: illustration of the statistical interpretation of unfolded events for one event. Right: calibration curves for p_{T,q_1} extracted from the FCGAN and the noise-extended eINN, as shown in Fig. 5.4, and the cINN.

motivated by the simple argument that for the correct set of network parameters θ describing s_i and t_i we maximize the (posterior) probability $p(\theta|x_p, x_d)$ or minimize

$$\begin{aligned} L &= -\left\langle \log p(\theta|x_p, x_d) \right\rangle_{x_p\sim P_p, x_d\sim P_d} \\ &= -\left\langle \log p(x_d|x_p, \theta) + \log p(\theta|x_p) - \log p(x_d|x_p) \right\rangle_{x_p\sim P_p, x_d\sim P_d} \\ &= -\left\langle \log p(x_d|x_p, \theta) \right\rangle_{x_p\sim P_p, x_d\sim P_d} - \log p(\theta) + \text{const.} \\ &= -\left\langle \log p(g(x_p, x_d)) + \log \left| \frac{\partial g(x_p, x_d)}{\partial x_p} \right| \right\rangle_{x_p\sim P_p, x_d\sim P_d} - \log p(\theta) + \text{const.} \,, \end{aligned} \tag{5.18}$$

where we first use Bayes' theorem, then ignore all terms irrelevant for the minimization, and finally apply a simple coordinate transformation for the bijective mapping. The last term is a simple weight regularization, while the first two terms are called the maximum likelihood loss. Since we impose the latent distribution of the random variable $p(g(x_p, x_d))$ to produce a normal distribution centered around zero and with width one, the first term becomes

$$\log p(g(x_p, x_d)) = -\frac{||g(x_p, x_d))||_2^2}{2} \,. \tag{5.19}$$

The final network setup after tuning of the hyper-paramaeters are liste In Tab. 5.2. We verified that the network performance is stable under small changes of these parameters.

In the left panels of Fig. 5.6 we show the unfolding performance of the cINN, trained and tested on the same exclusive 2-jet events as the simpler INNs in Fig. 5.3. Unlike the naive and the noise-extended INNs we cannot evaluate the cINN in both direction, detector simulation and unfolding, so we focus on the detector unfolding. The agreement between parton-level truth and the INN-unfolded distribution is around 10% for the bulk of the p_T distributions, with the usual larger relative deviations in the tails. An interesting feature is still the cut $p_{T,j} > 20$ GeV at the detector level, because it leads to a slight shift in the peak of the p_{T,j_2} distribution. Finally, the reconstructed invariant W-mass and the physical W-width agree extremely well with the Monte Carlo truth owing to the MMD loss.

As in Fig. 5.4 we can interpret the unfolding output for a given detector-level event statistically. First, in the left panel of Fig. 5.7 we show a single event and how the FCGAN, INN, and cINN output is

distributed in parton level phase space*. The separation between truth and sampled distributions does not have any significance, but we see that the cINN inherits the beneficial features of the noise-extended eINN. In the right panel of Fig. 5.7 we again reconstruct the individual probability distribution from the unfolding numerically. We then determine the position of the parton-level truth in its respective probability distribution for the INN and the cINN. We expect a given percentage of the 1500 events to fall into the correct quantile of its respective probability distribution. The corresponding calibration curve for the cINN is added to the right panel of Fig. 5.7, indicating that without additional calibration the output of the cINN unfolding can be interpreted as a probability distribution in parton-level phase space for a single detector-level event, as always assuming an unfolding model. Instead of the transverse momentum of the harder parton-level quark we could use any other kinematic distribution at parton level. This marks the final step for a statistically interpretable unfolding.

5.4 Unfolding with jet radiation

In the previous chapter we use a simplified data set to explore different possibilities to unfold detector level information with invertible networks. We limit the data to events with exactly two jets, by switching off initial state radiation (ISR). This guarantees that the two jets come from the W-decay, so the network does not have to learn this feature. In a realistic QCD environment we do not have that information, because additional QCD jets will be radiated off the initial and final state partons. In this section we demonstrate how we can unfold a sample of events including ISR and hence with a variable number of jets. We know that with very few exceptions [**?**, **?**] the radiation of QCD jets does not help us understand the nature of the hard process. In such cases, we would like to interpret a measurement with an appropriately defined hard process, leading to the question if an unfolding network can invert detector effects and QCD jet radiation. Technically, this means inverting jet radiation and kinematic modifications to the hard process as, in our case, done by Pythia.

We emphasize that this approach requires us to define a specific hard process with any number of external jets and other features. We can illustrate this choice for two examples. First, a di-tau resonance search typically probes the hard process $pp \to \mu^+\mu^- + X$, where X denotes any number of additional, analysis-irrelevant jets. We invert the corresponding measurements to the partonic process $pp \to \mu^+\mu^-$. A similar mono-jet analysis instead probes the process $pp \to Z'j(j) + X$, where Z' is a dark matter mediator decaying to two invisible dark matter candidate. Depending on the analysis, the relevant process to invert is $pp \to Z'j$ or $pp \to Z'jj$, where a reported missing transverse momentum recoils against one or two hard jets. Because our inversion network in trained on Monte Carlo data, we automatically define the appropriate hard process when generating the training data. This covers any combination of signal and background matrix elements contributing to such a hard process, even non-SM processes to quantify a remaining model dependence. A final caveat — in the hard process we do not include subjet aspects at this stage. As long as subjet information is used for tagging purposes it factorizes from the hard process information and can easily be included in terms of efficiencies. A problem would arise in unfolding or inverting analyses relying on different hard processes, like a fat mono-jet analysis, where the above choice of recoil jets is left to a sub-jet algorithm.

5.4.1 Individual n-jet samples

In Sec. 5.3.3 we have shown that our cINN can unfold detector effects for ZW-production at the LHC. The crucial new feature of the cINN is that it provides probability distribution in parton-level phase space for a given detector-level event. The actual unfolding results are illustrated in Fig. 5.6, focusing on the two critical distribution known from the corresponding FCGAN analysis [**?**]. The event sample used throughout Sec. 5.3 includes exactly two partons from a W-decay with minimal phase space cuts on the corresponding jets. Strictly speaking, these phase space cuts are not necessary in this

*Throughout this paper we only compare to the FCGAN analysis [**?**], which we fully control. For standard unfolding methods used by ATLAS and CMS and for the new Omnifold method [**?**] we refrain from comments which would need to be based on an in-depth comparison.

simulation. The correct definition of a process described by perturbative QCD includes a free number of additional jets,

$$pp \to ZW^\pm + \text{jets} \to (\ell^-\ell^+)\,(jj) + \text{jets}\,, \tag{5.20}$$

For the additional jets we need to include for instance a p_T cut to regularize the soft and collinear divergences at fixed-order perturbation theory. The proper way of generating events is therefore to allow for any number of additional jets and then cut on the number of hard jets. Since ISR can lead to jets with larger p_T than the W-decay jets, an assignment of the hardest jets to hard partons does not work. We simply sort jets and partons by their respective p_T and let the network work out their relations. We limit the number of jets to four because larger jet number appear very rarely and would not give us enough training events.

Combining all jet multiplicities we use 780k events, out of which 530k include exactly two jets, 190k events include three jets and 60k have four or more jets. We split the data into 80% training data and 20% test data to produce the shown plots. For the network input we zero-pad the event-vector for events with less than four jets and add the number of jets as additional information. The training samples are then split by exclusive jet multiplicity, such that the cINN reconstructs the 2-quark parton-level kinematics from two, three, and four jets at the detector level.

As before, we can start with the sample including exactly two jets. The difference to the sample used before is that now one of the W-decay jets might not pass the jet p_T condition in Eq.(5.11), so it will be replaced by an ISR jet in the 2-jet sample. Going back to Fig. 5.6 we see in the right panel how these events are slightly different from the sample with only decay jet. The main difference is in p_{T,q_2}, where the QCD radiation produces significantly more soft jets. Still, the network learns these features, and the unfolding for the sample without ISR and the 2-jet exclusive sample has a similar quality. In Fig. 5.8 we see the same distributions for the exclusive 3-jet and 4-jet samples. In this case we omit the secondary panels because they are dominated by the statistical uncertainties of the training sample. For these samples the network has to extract the parton-level kinematics with two jets only from up to four jets in the final state. In many cases this corresponds to just ignoring the two softest jets and mapping the two hardest jets on the two W-decay quarks, but from the p_{T,q_2} distributions in Fig. 5.6 we know that this is not always the correct solution. Especially in the critical m_{jj} peak reconstruction we see that the network feels the challenge, even though the other unfolded distributions look fine.

Parameter	cINN no ISR	cINN ISR incl.
Blocks	24	24
Layers per block	2	3
Units per layer	256	256
Condition/encoder layers	2	8
Units per condition/encoder layer	1024	1024
Condition/encoder output dimension	256	256
Trainable weights	$\sim$ 2 M	$\sim$ 10 M
Encoder pre training epochs	-	300
Epochs	1000	900
Learning rate	$8 \cdot 10^{-4}$	$8 \cdot 10^{-4}$
Batch size	512	512
Training/testing events	290k / 30k	620k / 160k
Kernel widths	$\sim 2, 8, 25, 67$	$\sim 2, 8, 25, 67$
D_p	12	12
D_d	16	25
λ_{MMD}	0.5	0.04
λ_{MMD} increase	-	1.6 / 100 epochs

Table 5.2: cINN setup and hyper-parameters, as implemented in pytorch(v1.2.0) [**?**].

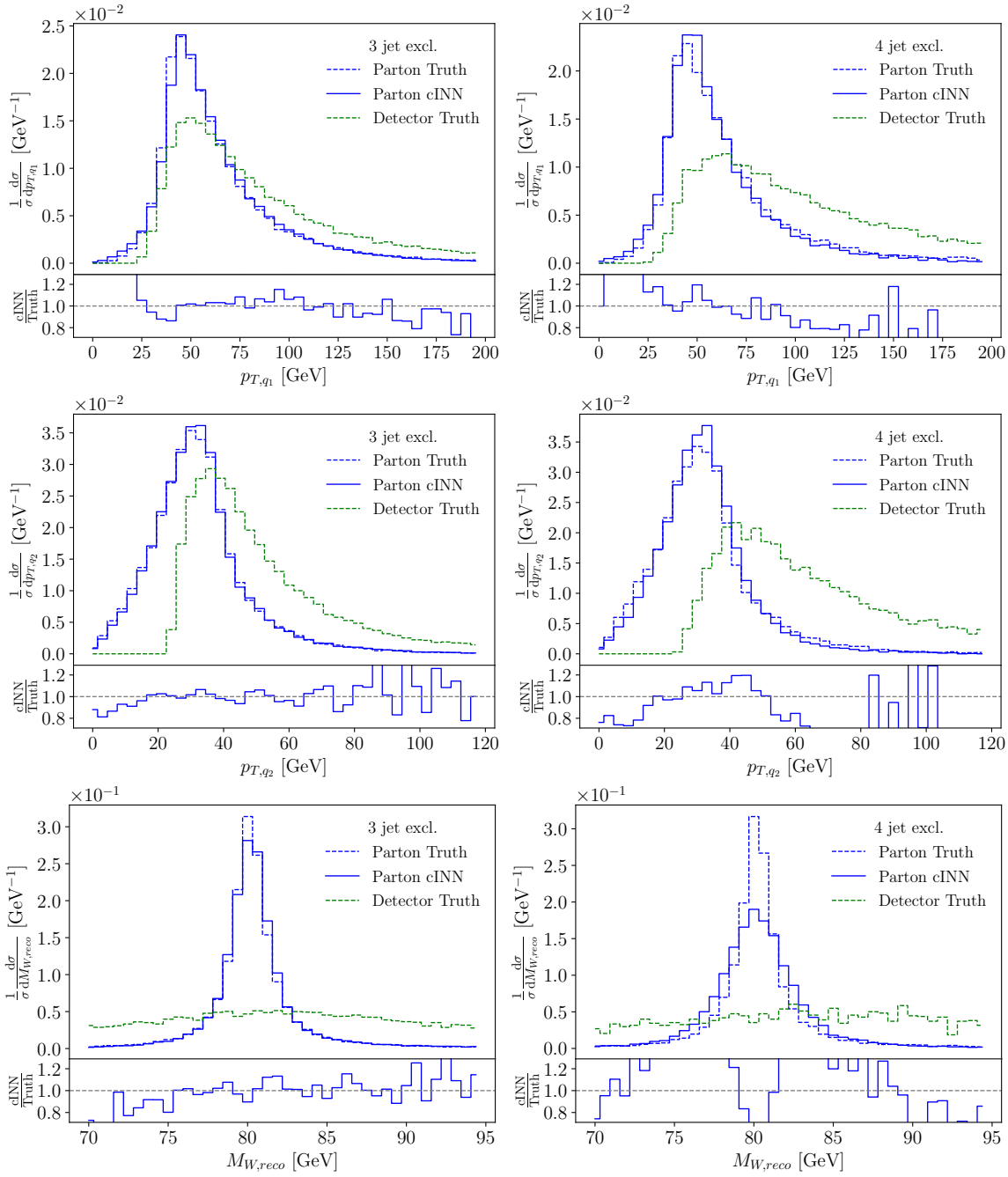

Figure 5.8: cINNed $p_{T,q}$ and $m_{W,\text{reco}}$ distributions. Training and testing events include exactly three (left) and four (right) jets from the data set including ISR.

5.4.2 Combined n-jet sample

The obvious final question is if our INN can also reconstruct the hard scattering process with its two W-decay quarks from a sample with a variable number of jets. Instead of separate samples as in Sec. 5.4.1 we now interpret the process in Eq.(5.20) as jet-inclusive. This means that the hard process includes only the two W-decay jets, and all additional jets are understood as jet radiation, described either by resummed ISR or by fixed-order QCD corrections. The training sample consists of the combination of the right panels in Fig. 5.6 and the two panels in Fig. 5.8. This means that the

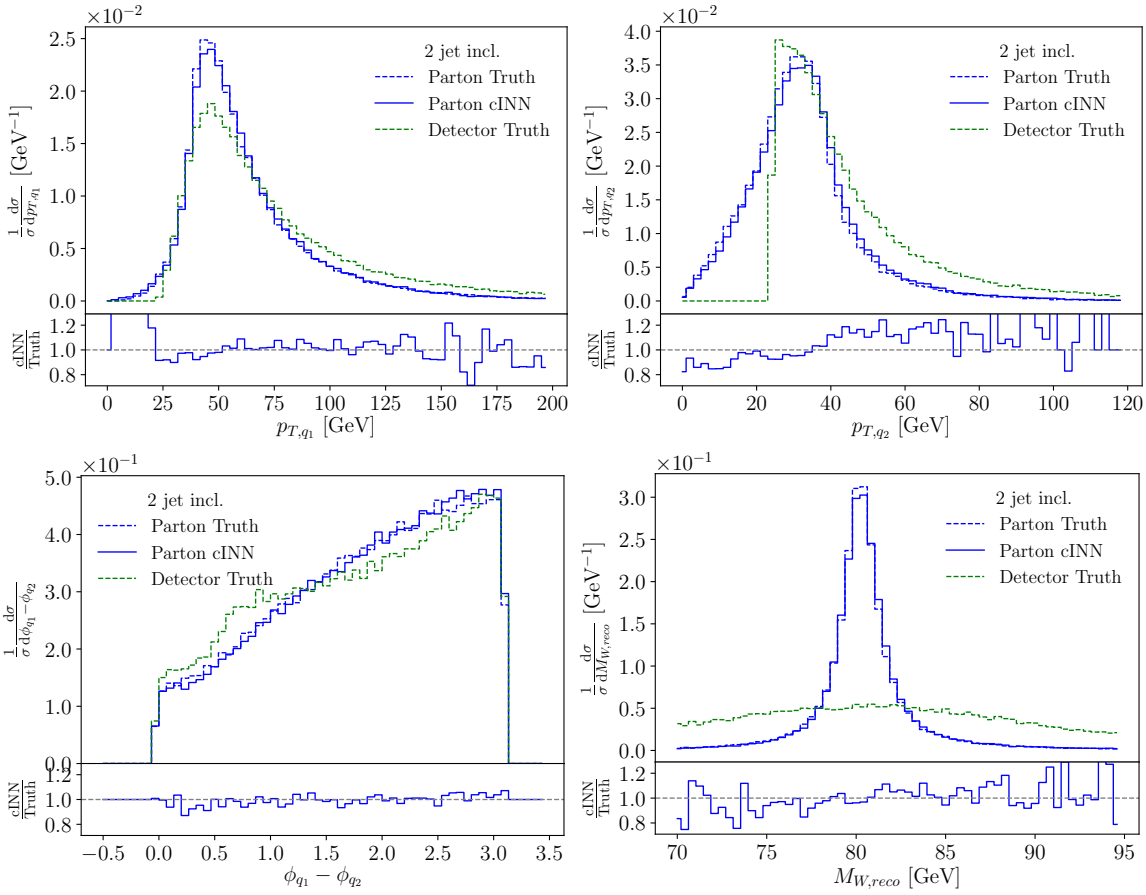

Figure 5.9: cINNed example distributions. Training and testing events include two to four jets, combining the samples from Fig. 5.6 and Fig. 5.8 in one network. At the parton level there exist only two W-decay quarks.

network has to deal with the different number of jets in the final state and how they can be related to the two hard jets of the partonic $ZW \to \ell\ell jj$ process. The number of jets in the final state is not given by individual hard partons, but by the jet algorithm and its R-separation.

In Fig. 5.9 we show a set of unfolded distributions. First, we see that the $p_{T,j}$ thresholds at the detector level are corrected to allow for $p_{T,q}$ values to zero. Next, we see that the comparably flat azimuthal angle difference at the parton level is reproduced to better than 10% over the entire range. Finally, the m_{jj} distribution with its MMD loss re-generates the W-mass peak at the parton level almost perfectly. The precision of this unfolding is not any worse than it is for the case where the number of hard partons and jets have to match and we only unfold the detector effects.

In Fig. 5.10 we split the unfolded distributions in Fig. 5.9 by the number of 2, 3, and 4 jets in the detector-level events. In the first two panels we see that the transverse momentum spectra of the hard partons are essentially independent of the QCD jet radiation. In the language of higher-order calculations this means that we can describe extra jet radiation with a constant K-factor, if necessary with the appropriate phase space mapping. Also the reconstruction of the W-mass is not affected by the extra jets, confirming that the neural network correctly identifies the W-decay jets and separates them from the ISR jets. Finally, we test the transverse momentum conservation at the unfolded parton level. Independent of the number of jets in the final state the energy and momentum for the pre-defined hard process is conserved at the 10^{-4} level. The kinematic modifications from the ISR simulation are unfolded correctly, so we can compute the matrix element for the hard process and use it for instance for inference.

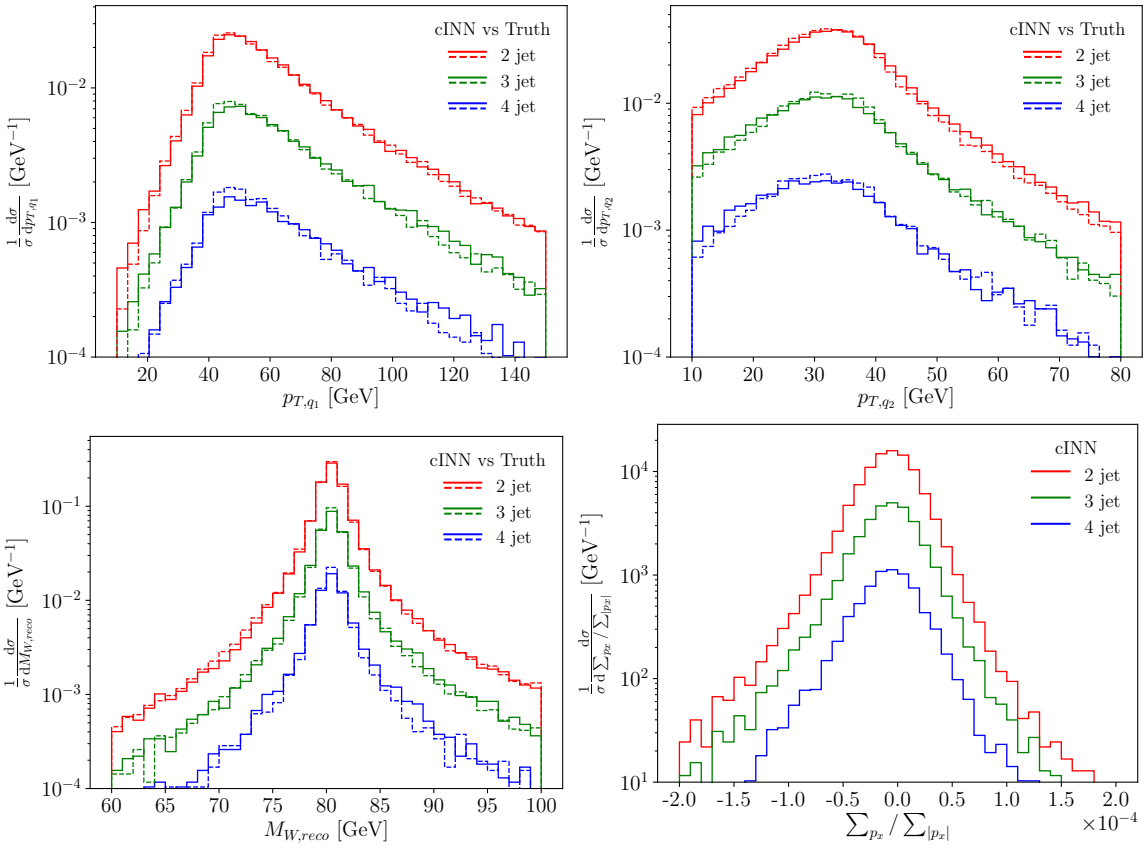

Figure 5.10: cINNed example distributions. Training and testing events include two to four events, combining the samples from Fig. 5.6 and Fig. 5.8 in one network. The parton-level events are stacked by number of jets at detector level.

5.5 Outlook

We have shown how an invertible network (INN) and in particular a conditional INN can be used to unfold detector effects for the simple example process of $ZW \to \ell\ell jj$ production at the LHC. The cINN is not only able to unfold the process over the entire phase space, it also gives correctly calibrated posterior probability distributions over parton-level phase space for given detector-level events. This feature is new even for neural network unfolding.

Next, we have extended the unfolding to a variable number of jets in the final state. This situation will automatically appear whenever we include higher-order corrections in perturbative QCD for a given hard process. The hard process at parton level is defined at the training level. We find that the cINN also unfolds QCD jet radiation in the sense that it identifies the ISR jets and corrects the kinematics of the hard process to ensure energy-momentum conservation in the hard scattering.

In combination, these features should enable analysis techniques like the matrix element method and efficient ways to communicate analysis results including multi-dimensional kinematic distributions. While the ZW production process used in this analysis, we expect these results to carry over to more complex processes with intermediate particles [**?**] and the impact of a SM-training hypothesis should be under control [**?**], the next step will be to test this new framework in a realistic LHC example with proper precision predictions and a focus on uncertainties. As for any analysis method suitable for the coming LHC runs, the challenge will be to control the full uncertainty budget at the per-cent level.[†]

[†]We are very happy to share our code upon request, if colleagues are interested in tackling any such open questions.

6 | Bayesian Neural Networks

bla bla

Abstract

Following the growing success of generative neural networks in LHC simulations, the crucial question is how to control the networks and assign uncertainties to their event output. We show how Bayesian normalizing flow or invertible networks capture uncertainties from the training and turn them into an uncertainty on the event weight. Fundamentally, the interplay between density and uncertainty estimates indicates that these networks learn functions in analogy to parameter fits rather than binned event counts.

6.1 Introduction

The role of first-principle simulations in our understanding of large data sets makes LHC physics stand out in comparison to many other areas of science. Three aspects define the application of modern big data methods in this field:

- ATLAS and CMS deliver proper big data with excellent control over uncertainties;
- perturbative quantum field theory provides consistent precision predictions;
- fast and reliable precision simulations generate events from first principles.

The fact that experiments, field theory calculations, and simulations control their uncertainties implies that we can work with a complete uncertainty budget, including statistical, systematic, and theory uncertainties. To sustain this approach at the upcoming HL-LHC, with a data set more than 25 times the current Run 2 data set, the theory challenge is to provide faster simulations and keep full control of the uncertainties at the per-cent level and better.

In recent years it has been shown that modern machine learning can improve LHC event simulations in many ways [**?**]. Promising techniques include generative adversarial networks (GAN) [**?**, **?**, **?**], variational autoencoders [**?**, **?**], and normalizing flows [**?**, **?**, **?**, **?**, **?**], including invertible networks (INNs) [**?**, **?**, **?**]. They can improve phase space integration [**?**, **?**], phase space sampling [**?**, **?**, **?**], and amplitude computations [**?**, **?**]. Further developments are fully NN-based event generation [**?**, **?**, **?**, **?**, **?**], event subtraction [**?**], event unweighting [**?**, **?**], detector simulation [**?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**], or parton showering [**?**, **?**, **?**, **?**, **?**]. Generative models will also improve searches for physics beyond the Standard Model [**?**], anomaly detection [**?**, **?**], detector resolution [**?**, **?**], and inference [**?**, **?**, **?**]. Finally, conditional GANs and INNs allow us to invert the simulation chain to unfold detector effects [**?**, **?**] and extract the hard scattering process at parton level [**?**]. The problem with these applications is that we know little about

1. how these generative networks work, and
2. what the uncertainty on the generative network output is.

As we will see in this paper, these two questions are closely related.

In general, we can track statistical and systematic uncertainties in neural network outputs with Bayesian networks [?, ?, ?, ?]. Such networks have been used in particle physics for a long time [?, ?, ?]. For the LHC we have proposed to use them to extract uncertainties in jet classification [?] and jet calibration [?]. They can cover essentially all uncertainties related to statistical, systematic, and structural limitations of the training sample [?]. Similar ideas can be used as part of ensemble techniques [?]. We propose to use a Bayesian INN (BINN) to extract uncertainties on a generated event sample induced by the network training.

Because Bayesian networks learn the density and uncertainty maps in one pass, their relation offers us fundamental insight into the way an INN learns a distribution. While Bayesian classification [?] and regression networks [?] highlight the statistical and systematic nature of uncertainties, our Bayesian generative network exhibits a very different structure. We will discuss the learning pattern of the Bayesian INN in details for a set of simple toy processes in Sec. 6.3, before we apply the network to a semi-realistic LHC example in Sec. 6.4.

6.2 Generative networks with uncertainties

We start by reminding ourselves that we often assume that a generative model has learned a phase-space density perfectly, so the only remaining source of uncertainty is the statistics of the generated sample binned in phase space. However, we know that such an assumption is not realistic [?, ?], and we need to estimate the effect of statistical or systematic limitations of the training data. The problem with such a statistical limitation is that it is turned into a systematic shortcoming of the generative model [?] — once we generate a new sample, the information on the training data is lost, and the only way we might recover it is by training many networks and comparing their outcome. For most applications this is not a realistic or economic option, so we will show how an alternative solution could look.

6.2.1 Uncertainties on event samples

Uncertainties on a simulated kinematic or phase space distribution are crucial for any LHC analysis. For instance, we need to know to what degree we can trust a simulated p_T-distribution in mono-jet search for dark matter. We denote the complete phase space weight for a given phase space point as $p(x)$, such that we can illustrate a total cross section as

$$\sigma_{\text{tot}} = \int_0^1 dx\, p(x) \quad \text{with} \quad p(x) > 0 \,. \tag{6.1}$$

In this simplified notation x stands for a generally multi-dimensional phase space. For each phase space position, we can also define an uncertainty $\sigma(x)$.

One contribution to the error budget are systematic and theory uncertainties, $\sigma_{\text{th/sys}}(x)$. The former reflect our ignorance of aspects of the training data, which do not decrease when we increase the amount of training data. The latter captures the degree to which we trust our prediction, for instance based on self-consistency arguments. For example accounting for large, momentum-dependent logarithms we can compute it from the phase space position, or for an unweighted event, alone. If we use a numerical variation of the factorization and renormalization scale to estimate a theory uncertainty, we typically re-weight events with the scales. Another uncertainty arises from the statistical limitations of the training data, $\sigma_{\text{stat}}(x)$. For instance in mono-jet production, the tails of the predicted p_T-distribution for the Standard Model will at some point be statistics limited. In the Gaussian limit, a statistical uncertainty can be defined by binning the phase space and in that limit we expect a scaling like $\sigma_{\text{stat}}(x) \sim \sqrt{p(x)}$, and we will test that hypothesis in detail in Sec. 6.3.

Once we know the uncertainties as a function of the phase space position, we can account for them as additional entries in unweighted or weighted events. For instance, relative uncertainties can be easily

added to unweighted events,

$$\text{ev}_i = \begin{pmatrix} \sigma_{\text{stat}}/p \\ \sigma_{\text{syst}}/p \\ \sigma_{\text{th}}/p \\ \{x_{\mu,j}\} \\ \{p_{\mu,j}\} \end{pmatrix} , \qquad \text{with } \mu = 0 \ldots 3 \text{ for each particle } j. \tag{6.2}$$

The entries σ or σ/p are smooth functions of phase space. The challenge in working with this definition is how to extract σ_{stat} without binning. We will show how Bayesian networks give us access to limited information in the training data. Specific theory and systematics counterparts can be either computed directly or extracted by appropriately modifying the training data [**?**, **?**].

6.2.2 Invertible Neural Networks

To model complex densities such as LHC phase space distributions, we can employ normalizing flows [**?**, **?**,**?**,**?**]. They use the fact we can transform a random variable $z \sim p_Z(z)$ using a bijective map $G : z \to x$ to a random variable $x = G(z)$ with the density

$$p_X(x) = p_Z(z) \left| \det \frac{\partial G(z)}{\partial z} \right|^{-1} = p_Z\big(G^{-1}(x)\big) \left| \det \frac{\partial G^{-1}(x)}{\partial x} \right| . \tag{6.3}$$

Given a sample z from the base distribution, we can then use the map G to generate a sample from the target distribution going in the forward direction. Alternatively, we can use a sample x from the target distribution to compute its density using the inverse direction. We will suppress the subscripts in the distributions p_Z, p_X whenever the density is clear from the context, to lighten the notation.

For this to be a useful approach, we require the base distribution p_Z to be simple enough to allow for effective sample generation, G to be flexible enough for a non-trivial transformation, and its Jacobian determinant to be effectively computable. If these constraints are fulfilled, G gives us a powerful generative pipeline to model the phase space density,

$$\text{base distribution } z \sim p_Z \overset{[}{\leftarrow} \overline{G}(x)]G(z) \to\!\leftarrow\!\longrightarrow \text{phase space distribution } x \sim p_X , \tag{6.4}$$

where $\overline{G}(x) = G^{-1}(x)$.

To fulfill the first constraint, we choose the base distribution p_Z to be a multivariate Gaussian with a mean zero and an identity matrix as the covariance. The construction of G relies on the property that the composition of a chain of simple invertible nonlinear maps gives us a complex map. In contrast, the determinant of the Jacobian of the composition remains simple in the sense that we can decompose it into the product of determinants of each of the individual transformations. There exists a broad literature of different transformations, each with different strengths and weaknesses [**?**]. We rely on the real non-volume preserving flow [**?**] in the invertible neural network (INN) formulation [**?**].

An INN composes multiple transformation maps into coupling layers with the following structure. The input vector z into a layer is split in half, $z = (z_1, z_2)$, allowing us to compute the output $x = (x_1, x_2)$ of the layer as

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} z_1 \odot e^{s_2(z_2)} + t_2(z_2) \\ z_2 \odot e^{s_1(x_1)} + t_1(x_1) \end{pmatrix} , \tag{6.5}$$

where s_i, t_i $(i = 1, 2)$ are arbitrary functions, and $\odot$ is the element-wise product. In practice each is a small multi-layer perceptron. This transformation has the benefit of being easily invertible. Given a vector $x = (x_1, x_2)$ the inverse given as

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} (x_1 - t_2(z_2)) \odot e^{-s_2(z_2)} \\ (x_2 - t_1(x_1)) \odot e^{-s_1(x_1)} \end{pmatrix} . \tag{6.6}$$

Additionally, its Jacobian is an upper triangular matrix

$$\frac{\partial G(z)}{\partial z} = \begin{pmatrix} \text{diag}\left(e^{s_2(z_2)}\right) & \text{finite} \\ 0 & \text{diag}\left(e^{s_1(x_1)}\right) \end{pmatrix} , \tag{6.7}$$

whose determinant is just the product of the diagonal entries, irrespective of the entries on the off-diagonal. As such it is computationally inexpensive, easily composable, yet still allows for complex transformations.

We refer to the overall map composing a sequence of such coupling layers as $G(z;\theta)$, where we collected the parameters of the individual nets s, t of each layer into a joint θ. Note that each coupling layer has a separate set of nets, whose indices we suppress (e.g. s^l, t^l for the l-th layer). We can then learn the overall model via a maximum likelihood approach. It relies on the assumption that we have access to a data set of N samples $\mathcal{D} = \{x_1, \dots, x_N\}$ of the intractable target phase space distribution $p_X^*(x)$ and want to fit our model distribution $p_X(x;\theta)$ via the INN G. The maximum likelihood loss is

$$\begin{aligned} \mathcal{L}_{\text{ML}} &= -\sum_{n=1}^{N} \log p_X(x_n;\theta) \\ &= -\sum_{n=1}^{N} \log p_Z\left(\overline{G}(x_n;\theta)\right) + \log \left| \det \frac{\partial \overline{G}(x_n;\theta)}{\partial x_n} \right| . \end{aligned} \tag{6.8}$$

Given the structure of $\overline{G}(x;\theta)$ and the base distribution p_Z each of the terms is tractable and can be computed efficiently. We can approximate the sum over the complete training data via a mini-batch and optimize the overall objective with a stochastic gradient descent approach. Note that one can see this maximum likelihood approach as minimizing the Kullback-Leibler (KL) divergence between the true but unknown phase space distribution $p_X^*(x)$ and our approximating distribution $p_X(x;\theta)$.

6.2.3 Bayesian INN

The invertible neural net provides us with a powerful generative model of the underlying data distribution. However, it lacks a mechanism to account for our uncertainty in the transformation parameters θ themselves. To model it, we switch from deterministic transformations to probabilistic transformations, replacing the deterministic sub-networks $s_{1,2}$ and $t_{1,2}$ in each of the coupling layers with Bayesian neural nets. In this section, we first review the structure of a classical Bayesian neural net (BNN) [**?**, **?**] as used in a supervised learning task, and then explain how we can use BNNs for our problem of modeling the phase space density, extending the INN into a Bayesian invertible neural net (BINN).

Bayesian Neural Net Assuming a data set $\mathcal{D}$ consisting of N pairs of observations $(\mathbf{x}_i, y_i)$, $\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$, in the supervised learning problem we want to model the relation $y = f_\theta(\mathbf{x})$ though a neural network parameterised by weights θ. Placing a prior over the weights and allowing for some observation noise, the generative model is given as

$$\begin{aligned} \theta &\sim p(\theta) , \\ y_i|\theta, \mathbf{x}_i &\sim p(y_i|\theta, \mathbf{x}_i), \qquad\qquad i = 1, \dots, N . \end{aligned} \tag{6.9}$$

In case of a regression with $y_i \in \mathbb{R}$ we often use a Gaussian likelihood, $p(y_i|\theta, \mathbf{x}_i) = \mathcal{N}\left(y_i|f_\theta(\mathbf{x}_i), \alpha^{-1}\right)$, and a Gaussian prior over the weights $p(\theta) = \mathcal{N}\left(\theta|\mathbf{0}, \beta^{-1}\mathbf{1}\right)$, with precisions α, β and $\mathbf{1}$ the identity matrix of suitable dimensionality [**?**]. We are not bound to these distributions and could for example choose a prior with a strongly sparsifying character for further regularization [**?**, **?**]. Given the highly nonlinear structure of f_θ the posterior $p(\theta|\mathcal{D})$ is, for practically relevant applications, analytically intractable. While MCMC-based approaches can work for specific use cases and small networks [**?**], they quickly become too expensive for large architectures, so we instead rely on variational inference (VI) [**?**]. A VI-based model approximates the posterior $p(\theta|\mathcal{D})$ with a tractable simplified family of

distributions, $q_\phi(\theta)$, parameterized by ϕ. We will rely on mean-field Gaussians throughout this work, learning a separate mean and variance parameter for each network weight. These parameters are learned by minimizing the KL-divergence

$$\min_\phi \mathrm{KL}\big(q_\phi(\theta), p(\theta|\mathcal{D})\big) \; . \tag{6.10}$$

However, this objective is intractable, as it relies on the unknown posterior. Using Bayes' theorem we reformulate it as

$$\begin{aligned} \mathrm{KL}\big(q_\phi(\theta), p(\theta|\mathcal{D})\big) &= -\int d\theta \; q_\phi(\theta) \log \frac{p(\mathcal{D}|\theta)p(\theta)/p(\mathcal{D})}{q_\phi(\theta)} \\ &= -\int d\theta \; q_\phi(\theta) \log p(\mathcal{D}|\theta) - \int d\theta \; q_\phi(\theta) \log \frac{p(\theta)}{q_\phi(\theta)} + \log p(\mathcal{D}) \; . \end{aligned} \tag{6.11}$$

Now, the log evidence $\log p(\mathcal{D})$ is bounded from below as

$$\begin{aligned} \log p(\mathcal{D}) &= \mathrm{KL}\big(q_\phi(\theta), p(\theta|\mathcal{D})\big) + \int d\theta \; q_\phi(\theta) \log p(\mathcal{D}|\theta) - \mathrm{KL}\big(q_\phi(\theta), p(\theta)\big) \\ &\geq \int d\theta \; q_\phi(\theta) \log p(\mathcal{D}|\theta) - \mathrm{KL}\big(q_\phi(\theta), p(\theta)\big) \; . \end{aligned} \tag{6.12}$$

Maximizing this evidence lower bound (ELBO) then is equivalent to minimizing Eq.(6.10), giving us as the objective without the intractable posterior

$$\mathcal{L}_\mathrm{ELBO} = \sum_{i=1}^N \Big\langle \log p(y_i|\theta, \mathbf{x}_i) \Big\rangle_{\theta\sim q_\phi(\theta)} - \mathrm{KL}\big(q_\phi(\theta), p(\theta)\big) \; . \tag{6.13}$$

This turns the inference problem into an optimization problem, which allows us to take advantage of gradient descent methods such as Adam [**?**]. As the choice of prior $p(\theta)$ is under our control, the KL-term between the variational posterior and the prior is tractable. The intractable expectation in the first term we can approximate by taking S samples from the variational posterior and instead of computing the gradient over the whole data set in each iteration switch to a stochastic gradient setup, approximating the sum with a mini-batch of size M, giving us

$$\mathcal{L}_\mathrm{ELBO} \approx \frac{N}{M} \sum_{i=1}^M \frac{1}{S} \sum_{s=1}^S \log p(y_i|\theta^{(s)}, \mathbf{x}_i) - \mathrm{KL}\big(q_\phi(\theta), p(\theta)\big) \qquad \text{with} \quad \theta^{(s)} \sim q_\phi(\theta) \; . \tag{6.14}$$

In practice, it is often sufficient to approximate the expectation via a single sample ($S = 1$) per forward pass to keep the computational cost low and further rely on local re-parametrization [**?**] to reduce the variance of the gradients.

Bayesian INN As discussed in Sec. 6.2.2, our generative model of the density consists of a map $G : z \to x$ from a base distribution $p_Z(z)$ to the phase-space $p_X(x)$ parameterized via an INN. Replacing the deterministic sub-networks $s_{1,2}$ and $t_{1,2}$ in Eq.(6.5) with BNNs we get as the generative pipeline for our BINN

$$\begin{aligned} \theta &\sim p(\theta), \\ x|\theta &\sim p_X(x|\theta) = p_Z(\overline{G}(x;\theta)) \Big| \det \frac{\partial \overline{G}(x;\theta)}{\partial x} \Big| \; . \end{aligned} \tag{6.15}$$

Given a set of N observations $\mathcal{D} = \{x_1, \dots, x_N\}$ we can approximate the intractable posterior $p(\theta|\mathcal{D})$ as before with a mean-field Gaussian as the variational posterior $q_\phi(\theta)$. Learning the map and the

posterior then is achieved by maximizing the equivalent of the ELBO loss in Eq.(6.14) for event samples,

$$\begin{aligned}
\mathcal{L} &= \sum_{n=1}^{N} \langle \log p_X(x_n|\theta) \rangle_{\theta\sim q_\phi(\theta)} - \mathrm{KL}\big(q_\phi(\theta), p(\theta)\big) \\
&= \sum_{n=1}^{N} \Big\langle \log p_Z\big(\overline{G}(x_n;\theta)\big) + \log \Big| \det \frac{\partial \overline{G}(x_n;\theta)}{\partial x_n} \Big| \Big\rangle_{\theta\sim q_\phi(\theta)} - \mathrm{KL}\big(q_\phi(\theta), p(\theta)\big) \\
&\approx \frac{N}{M} \sum_{m=1}^{M} \frac{1}{S} \sum_{s=1}^{S} \log p_Z\big(\overline{G}(x_m;\theta^{(s)})\big) + \log \Big| \det \frac{\partial \overline{G}(x_m;\theta^{(s)})}{\partial x_m} \Big| - \mathrm{KL}\big(q_\phi(\theta), p(\theta)\big) \ , \qquad (6.16)
\end{aligned}$$

with a mini-batch of size M and S samples $\theta^{(s)}$ from the variational posterior $q_\phi(\theta)$. By design all three terms, the log likelihood, the log determinant of the Jacobian as well as the Kullback-Leibler divergence can be computed easily. Automatic differentiation [**?**] allows us to get the gradients of $\mathcal{L}$ with respect to ϕ in order to fit our generative pipeline via a stochastic gradient descent update scheme.

6.3 Toy events with uncertainties

Before we tackle a semi-realistic LHC setup, we first study the behavior of BINNs for a set of toy examples, namely distributions over the minimally allowed two-dimensional parameter space where in one dimension the density is flat. Aside from the fact that these toy examples illustrate that the BINN actually constructs a meaningful uncertainty distribution, we will use the combination of density and uncertainty maps to analyse how an INN actually learns a density distributions. We will see that the INN describes the density map in the sense of a few-parameter fit, rather than numerically encoding patches over the parameter space independently.

The default architecture for our toy models is a network with 32 units per layer, three layers per coupling block, and a total of 20 coupling blocks. It's implemented in PyTorch [**?**]. More details are given in Tab. 6.1. The most relevant hyperparameter is the number of coupling blocks in that more blocks provide a more stable performance with respect to several trainings of the same architecture. Generally, moderate changes for instance of the number of units per layer do not have a visible impact on the performance. For each of the trainings we use a sample of 300k events. The widths of the Gaussian priors is set to one. We check that variations of this over several orders of magnitude did not have a significant impact on the performance.

Parameter	Flow
Hidden layers (per block)	3
Units per hidden layer	32
Batch size	512
Epochs	300
Trainable weights	75k
Optimizer	Adam
(α, β_1, β_2)	(1×10^{-3}, 0.9, 0.999)
Coupling layers	20
Training size	300k
Prior width	1

Table 6.1: Hyper-parameters for all toy models, implemented in pytorch(v1.4.0) [**?**].

6.3.1 Wedge ramp

Our first toy example is a two-dimensional ramp distribution, linear in one direction and flat in the other,

$$p(x,y) = \text{Linear}(x \in [0,1]) \times \text{Const}(y \in [0,1]) = x \times 2 \; . \tag{6.17}$$

The second term ensures that the distribution $p(x,y)$ is normalized to one, and the network output is shown in Fig. 6.1. The network output are unweighted events in the two-dimensional parameters space, (x,y). We show one-dimensional distributions after marginalizing over the unobserved direction and find that the network reproduces Eq.(6.17) well.

In Fig. 6.2 we include the predictive uncertainty given by the BINN. For this purpose we train a network on the two-dimensional parameter space and evaluate it for a set of points with $x \in [0,1]$ and a constant y-value. In the left panel we indicate the predictive uncertainty as an error bar around the density estimate. Throughout the paper we always remove the phase space boundaries, because we know that the network is unstable there, and the uncertainties explode just like we expect. The relative uncertainty grows for small values of x and hence small values of $p(x,y)$, and it covers the deviation of the extracted density from the true density well. These features are common to all our network trainings. In the central and right panel of Fig. 6.2 we show the relative and absolute predictive uncertainties. The error bar indicates how much σ_pred varies for different choices of y. We compute it as the standard deviation of different values of σ_pred, after confirming that the central values agree within this range. As expected, the relative uncertainty decreases towards larger x. However, the absolute uncertainty shows a distinctive minimum in σ_pred around $x \approx 0.45$. This minimum is a common feature in all our trainings, so we need to explain it.

To understand this non-trivial uncertainty distribution $\sigma_\text{pred}(x)$ we focus on the non-trivial x-coordinate and its linear behavior

$$p(x) = ax + b \qquad \text{with} \qquad x \in [0,1] \; . \tag{6.18}$$

Because the network learns a density, we can remove b by fixing the normalization,

$$p(x) = a\left(x - \frac{1}{2}\right) + 1 \; . \tag{6.19}$$

If we now assume that a network acts like a fit of a, as it will turn out useful, we can relate the uncertainty Δa to an uncertainty in the density,

$$\sigma_\text{pred} \equiv \Delta p \approx \left|x - \frac{1}{2}\right| \; \Delta a \; . \tag{6.20}$$

The absolute value appears because the uncertainties are defined to be positive, as encoded in the usual quadratic error propagation. The uncertainty distribution has a minimum at $x = 1/2$, close to the observed value in Fig. 6.2.

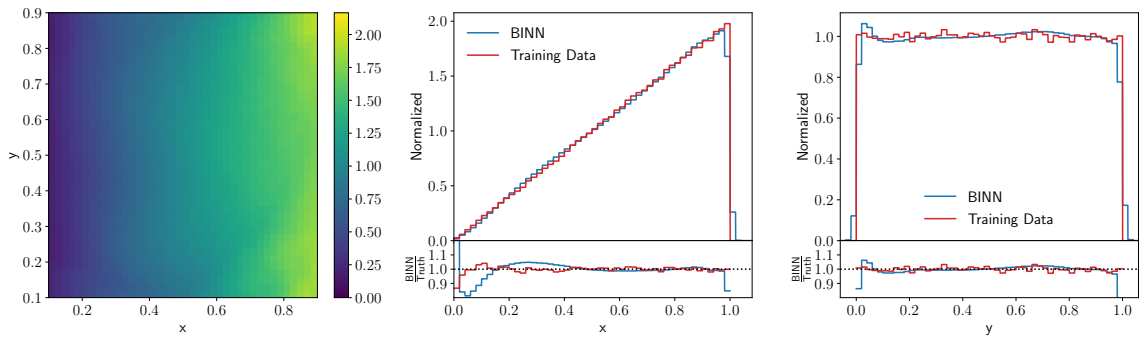

Figure 6.1: Two-dimensional and marginal densities for the linear wedge ramp.

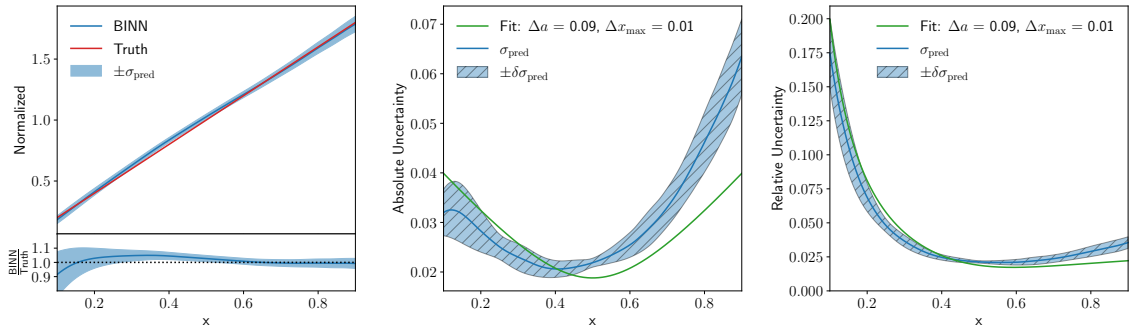

Figure 6.2: Density and predictive uncertainty distribution for the wedge ramp. In the left panel the density and uncertainty are averaged over several lines with constant y. In the central and right panels, the uncertainty band on σ_{pred} is given by their variation. The green curve represents a two-parameter fit to Eq.(6.23).

The differences between the simple prediction in Eq.(6.20) and our numerical findings in Fig. 6.2 is that the predictive uncertainty is not symmetric and does not reach zero. To account for these sub-leading effects we can expand our very simple ansatz to

$$p(x) = ax + b \qquad \text{with} \qquad x \in [x_{\text{min}}, x_{\text{max}}] \,. \tag{6.21}$$

Using the normalization condition we again remove b and find

$$p(x) = ax + \frac{1 - \frac{a}{2}(x_{\text{max}}^2 - x_{\text{min}}^2)}{x_{\text{max}} - x_{\text{min}}} \,. \tag{6.22}$$

Again assuming a fit-like behavior of the flow network we expect for the predictive uncertainty

$$\sigma_{\text{pred}}^2 \equiv (\Delta p)^2 = \left(x - \frac{1}{2}\right)^2 (\Delta a)^2 + \left(1 + \frac{a}{2}\right)^2 (\Delta x_{\text{max}})^2 + \left(1 - \frac{a}{2}\right)^2 (\Delta x_{\text{min}})^2 \,. \tag{6.23}$$

Adding x_{max} adds an x-independent offset. Also accounting for x_{min} does not change the x-dependence of predictive uncertainty. The slight shift of the minimum and the asymmetry between the lower and upper boundaries in x are not explained by this argument. We ascribe them to boundary effects, specifically the challenge for the network to describe the correct approach towards $p(x) \to 0$.

The green line in the lower panels of Fig. 6.2 gives a two-parameter fit of Δa and Δx_{max} to the σ_{pred} distribution from the BINN. It indicates that there is a hierarchy in the way the network extracts the x-independent term with high precision, whereas the uncertainty on the slope a is around 4%.

6.3.2 Kicker ramp

We can test our findings from the linear wedge ramp using the slightly more complex quadratic or kicker ramp,

$$p(x, y) = \text{Quadr}(x \in [0, 1]) \times \text{Const}(y \in [0, 1]) = x^2 \times 3 \,. \tag{6.24}$$

We show the results from the network training for the density in Fig. 6.3 and find that the network describes the density well, limited largely by the flat, low-statistics approach towards the lower boundary with $p(x) \to 0$.

In complete analogy to Fig. 6.2 we show the complete BINN output with the density $p(x, y)$ and the predictive uncertainty $\sigma_{\text{pred}}(x, y)$ in Fig. 6.4. As for the linear case, the BINN reproduces the density well, deviations from the truth being within the predictive uncertainty in all points of phase space. We remove the phase space boundaries, where the network becomes unstable and the predictive

uncertainties grows correspondingly. The indicated error bar on $\sigma_{\text{pred}}(x, y)$ is given by the variation of the predictions for different y-values, after ensuring that their central values agree. The relative uncertainty at the lower boundary $x = 0$ is large, reflecting the statistical limitation of this phase-space region. An interesting feature appears again in the absolute uncertainty, namely a maximum-minimum combination as a function of x.

Again in analogy to Eq.(6.21) for the wedge ramp, we start with the parametrization of the density

$$p(x) = a\,(x - x_0)^2 \qquad \text{with} \qquad x \in [x_0, x_{\text{max}}]\,, \tag{6.25}$$

where we assume that the lower boundary coincides with the minimum and there is no constant offset. We choose to describe this density through the minimum position x_0, coinciding the the lower end of the x-range, and x_{max} as the second parameter. The parameter a can be eliminated through the normalization condition and we find

$$p(x) = 3\frac{(x - x_0)^2}{(x_{\text{max}} - x_0)^3}\,. \tag{6.26}$$

If we vary x_0 and x_{max} we can trace two contributions to the uncertainty in the density,

$$\begin{aligned} \sigma_{\text{pred}} \equiv \Delta p \supset \frac{9}{(x_{\text{max}} - x_0)^4} \left|(x - x_0)\left(x - \frac{x_0}{3} - \frac{2x_{\text{max}}}{3}\right)\right| \Delta x_0 \\ \text{and} \qquad \sigma_{\text{pred}} \equiv \Delta p \supset \frac{9}{(x_{\text{max}} - x_0)^4}\,(x - x_0)^2\,\Delta x_{\text{max}}\,, \end{aligned} \tag{6.27}$$

one from the variation of x_0 and one from the variation of x_{max}. In analogy to Eq.(6.23) they need to be added in quadrature. If the uncertainty on Δx_0 dominates, the uncertainty has a trivial minimum at $x = 0$ and a non-trivial minimum at $x = 2/3$. From Δx_{max} we get another contribution which scales like $\Delta p \propto p(x)$. In Fig. 6.4 we clearly observe both contributions, and the green line in the lower panels is given by the corresponding 2-parameter fig to the σ_{pred} distribution from the BINN.

6.3.3 Gaussian ring

Our third example is a two dimensional Gaussian ring, which in terms of polar coordinates reads

$$p(r, \phi) = \text{Gauss}(r > 0; \mu = 4, w = 1) \times \text{Const}(\phi \in [0, \pi])\,, \tag{6.28}$$

We define the Gaussian density as the usual

$$\text{Gauss}(r) = \frac{1}{\sqrt{2\pi}\,w} \exp\left[-\frac{1}{2w^2}(r - \mu)^2\right] \tag{6.29}$$

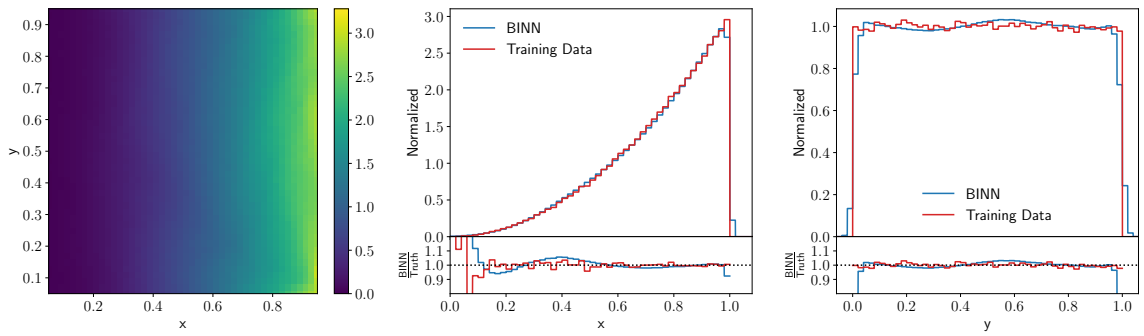

Figure 6.3: Two-dimensional and marginal densities for the quadratic kicker ramp.

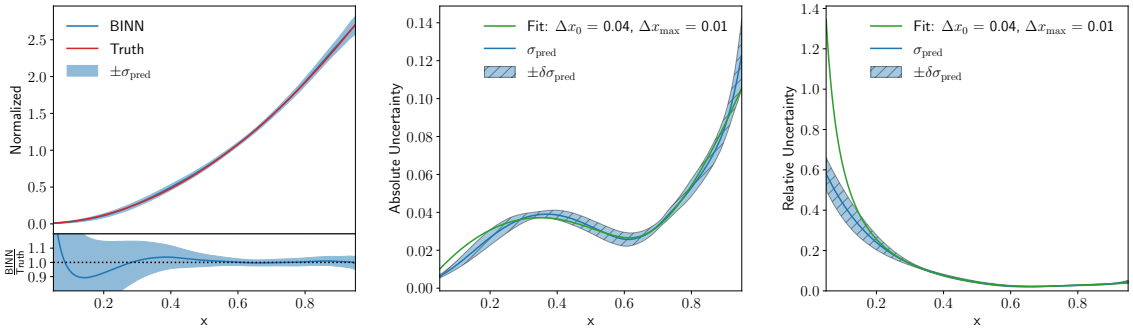

Figure 6.4: Density and predictive uncertainty distribution for the kicker ramp. In the left panel the density and uncertainty are averaged over several lines with constant y. In the central and right panels, the uncertainty band on σ_{pred} is given by their variation. The green curve represents a two-parameter fit to Eq.(6.27).

The density defined in Eq.(6.28) can be translated into Cartesian coordinates as

$$p(x,y) = \text{Gauss}(r(x,y); \mu = 4, w = 1) \times \text{Const}(\phi(x,y) \in [0,\pi]) \times \frac{1}{r(x,y)} \tag{6.30}$$

where the additional factor $1/r$ comes from the Jacobian. We train the BINN on Cartesian coordinates, just like in the two examples before, and limit ourselves to $y > 0$ to avoid problems induced by learning a non-trivial topology in mapping the latent and phase spaces. In Fig. 6.5 we once again see that our network describes the true two-dimensional density well.

In Fig. 6.6 we show the Cartesian density but evaluated on a line of constant angle. This form includes the Jacobian and has the expected, slightly shifted peak position at $r_{\text{max}} = 2 + \sqrt{3} = 3.73$. The BINN returns a predictive uncertainty, which grows towards both boundaries. The error band easily covers the deviation of the density learned by the BINN and the true density. While the relative predictive uncertainty appears to have a simple minimum around the peak of the density, we again see that the absolute uncertainty has a distinct structure with a local minimum right at the peak. The question is what we can learn about the INN from this pattern in the BINN.

As before, we describe our distribution in the relevant direction in terms of convenient fit parameters. For the Gaussian radial density these are the mean μ and the width w used in Eq.(6.28). The contributions driven by the extraction of the mean in Cartesian coordinates reads

$$\begin{aligned} \sigma_{\text{pred}} \equiv \Delta p \supset \left| \frac{G(r)}{r} \frac{\mu - r}{w^2} \right| \Delta\mu \\ \text{and} \qquad \sigma_{\text{pred}} \equiv \Delta p \supset \left| \frac{(r-\mu)^2}{w^3} - \frac{1}{w} \right| \Delta w \; . \end{aligned} \tag{6.31}$$

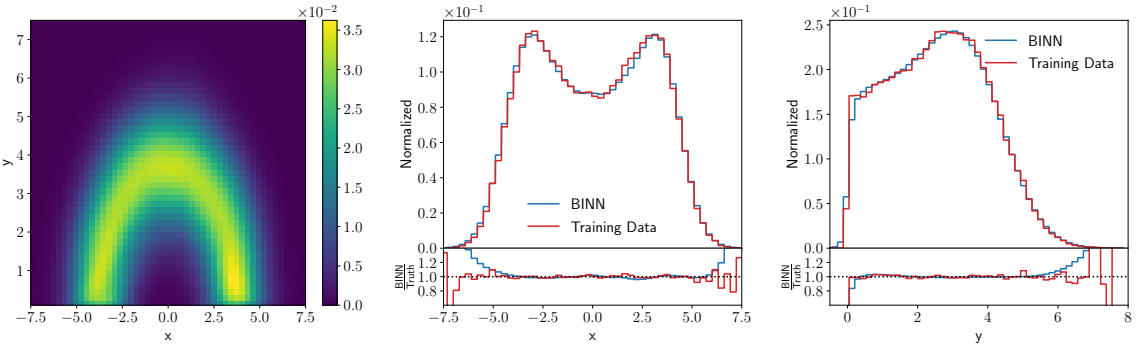

Figure 6.5: Two-dimensional and marginal densities for the Gaussian (half-)ring.

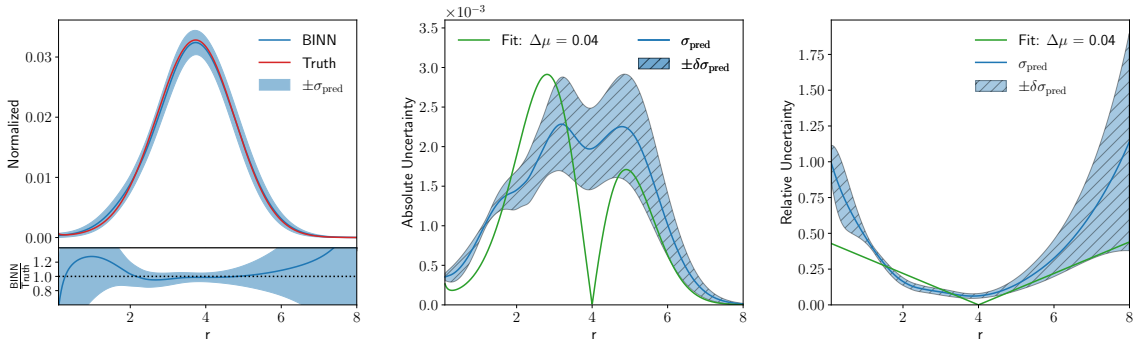

Figure 6.6: Cartesian density and predictive uncertainty distribution for the Gaussian ring. In the left panel the density and uncertainty are averaged over several lines with constant ϕ. In the central and right panels, the uncertainty band on σ_{pred} is given by their variation. The green curve represents a two-parameter fit to Eq.(6.31).

In analogy to Eq.(6.23) the two contributions need to be added in quadrature for the full, fit-like uncertainty. The contribution from the the mean has a minimum at $r = \mu = 4$ and is otherwise dominated by the exponential behavior of the Gaussian, just as we observe in the BINN result. In the opposite limit of $\Delta\mu \ll \Delta w$ the uncertainty develops the maxima at $r = 3$ and $r = 5$, which we observe in Fig. 6.6. In the lower panels we show a one-parameter fit of the BINN output and find that the network determined the mean of the Gaussian as $\mu = 4 \pm 0.037$.

6.3.4 Errors vs training statistics

Even though it is clear from the above discussion that we cannot expect the predictive uncertainties to have a simple scaling pattern, like for the regression [**?**] and classification [**?**] networks, there still remains the question how the BINN uncertainties change with the size of the training sample.

In Fig. 6.7 we show how the BINN predictions for the density and uncertainty change if we vary the training sample size from 10k events to 1M training events. Note that for all toy models, including the kicker ramp in Sec. 6.3.2, we use 300k training events. For the small 10k training sample, we see that the instability of the BINN density becomes visible even for our reduced x-range. The peak-dip pattern of the absolute uncertainty, characteristic for the kicker ramp, is also hardly visible, indicating that the network has not learned the density well enough to determine its shape. Finally, the variation of the predictive density explodes for $x > 0.4$, confirming the picture of a poorly trained BINN. As a rough estimate, the absolute uncertainty at $x = 0.5$ with a density value $p(x, y) = 0.75$ ranges around $\sigma_{\text{pred}} = 0.11 \ldots 0.15$.

For 100k training events we see that the patterns discussed in Sec. 6.3.2 begin to form. The density and uncertainty encoded in the network are stable, and the peak-dip with a minimum around $x = 2/3$ becomes visible. As a rough estimate we can read off $\sigma_{\text{pred}}(0.5) \approx 0.06 \pm 0.03$. For 1M training events the picture improves even more and the network extracts a stable uncertainty of $\sigma_{\text{pred}}(0.5) \approx 0.03 \pm 0.01$. Crucially, the dip around $x \approx 2/3$ remains, and even compared to Fig. 6.4 with its 300k training events the density and uncertainty at the upper phase space boundary are much better controlled.

Finally, we briefly comment on a frequentist interpretation of the BINN output. We know from simpler Bayesian networks [**?**, **?**] that it is possible to reproduce the predictive uncertainty using an ensemble of deterministic networks with the same architecture. However, from those studies we also know that our class of Bayesian networks has a very efficient built-in regularization, so this kind of comparison is not trivial. For the BINN results shown in this paper we find that the detailed patterns in the absolute uncertainties are extracted by the Bayesian network much more effectively than they would be for ensembles of deterministic INNs. For naive implementations with a similar network size and no fine-tuned regularization these patterns are somewhat harder to extract. On the other hand, in stable regions without distinctive patterns the spread of ensembles of deterministic networks reproduces the

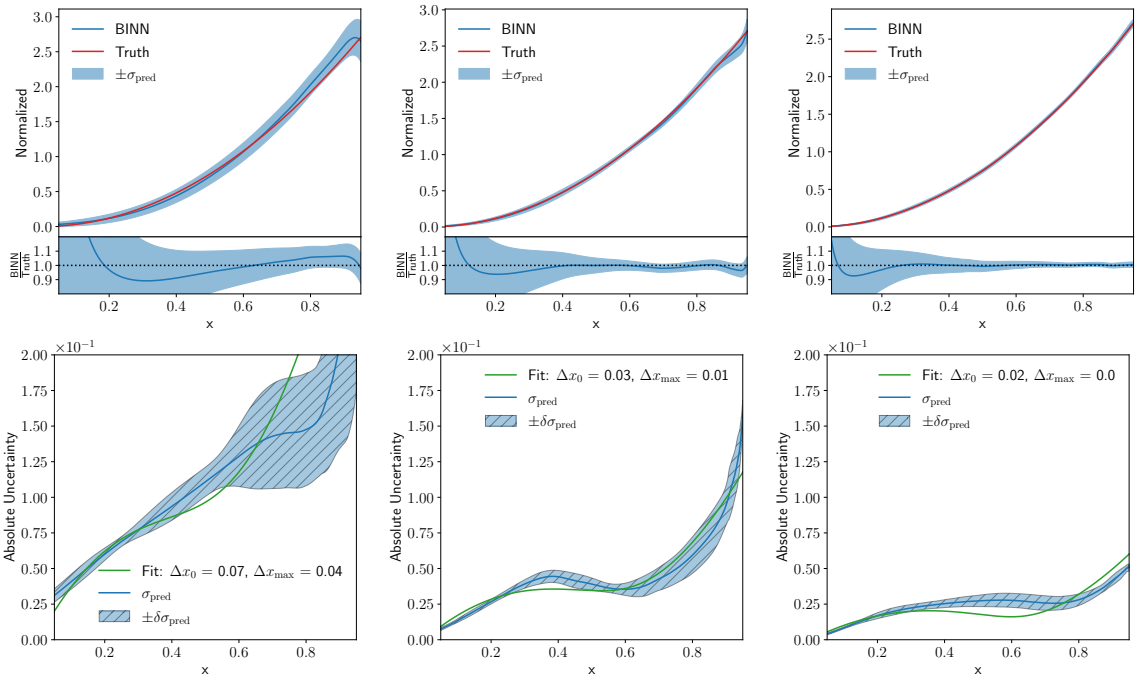

Figure 6.7: Dependence of the density (upper) and absolute uncertainty (lower) on the training statistics for the kicker ramp. We illustrate BINNs trained on 10k, 100k, and 1M events (left to right), to be compared to 300k events used for Fig. 6.4. Our training routine ensures that all models receive the same number of weights updates, regardless of the training set size.

predictive uncertainty reported by the BINN.

6.3.5 Marginalizing phase space

Before we move to a more LHC-related problem, we need to study how the BINN provides uncertainties for marginalized kinematic distribution. In all three toy examples the two-dimensional phase space consists of one physical and one trivial direction. For instance, the kicker ramp in Sec. 6.3.2 has a quadratic physical direction, and in a typical phase space problem we would integrate out the trivial, constant direction and show a one-dimensional kinematic distribution. From our effectively one-dimensional uncertainty extraction we know that the absolute uncertainty has a characteristic maximum-minimum combination, as seen in the lower-right panel of Fig. 6.4.

To compute the uncertainty for a properly marginalized phase space direction, we remind ourselves how the BINN computes the density and the predictive uncertainty by sampling over the weights,

$$\begin{aligned} p(x,y) &= \int d\theta\, q(\theta)\, p(x,y|\theta) \\ \sigma^2_{\text{pred}}(x,y) &= \int d\theta\, q(\theta)\, [p(x,y|\theta) - p(x,y)]^2 \; . \end{aligned} \tag{6.32}$$

If we integrate over the y-direction, the marginalized density is defined as

$$\begin{aligned} p(x) = \int dy\, p(x,y) &= \int dy d\theta\, q(\theta)\, p(x,y|\theta) \\ &= \int d\theta\, q(\theta) \int dy\, p(x,y|\theta) \equiv \int d\theta\, q(\theta)\, p(x|\theta) \; , \end{aligned} \tag{6.33}$$

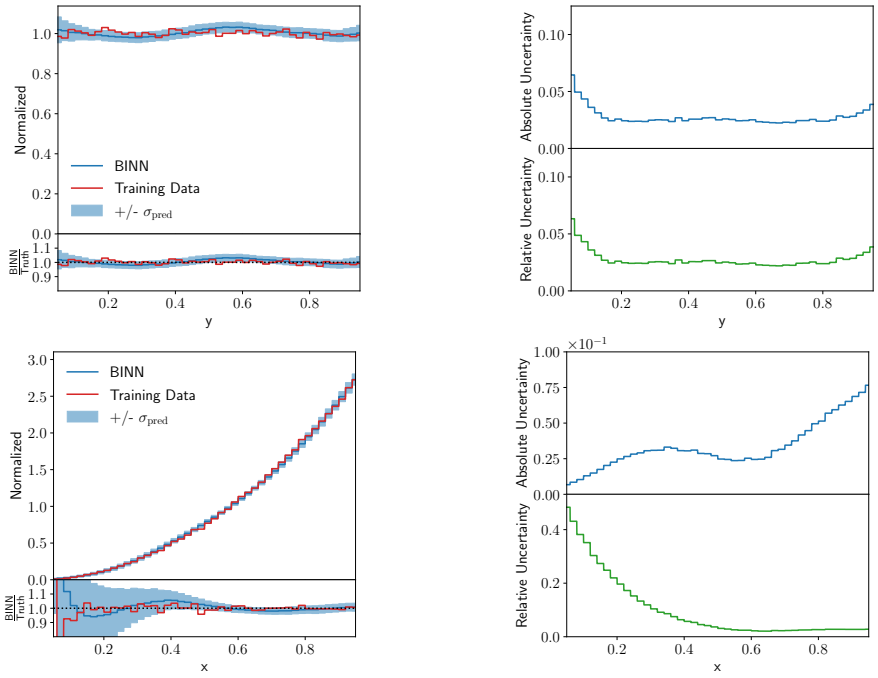

Figure 6.8: Marginalized densities and predictive uncertainties for the kicker ramp. Instead of the true distribution we now show the training data as a reference, to illustrate possible limitations. We use 10M phase space point to guarantee a stable prediction.

which implicitly defines $p(x|\theta)$ in the last step, notably without providing us with a way to extract it in a closed form. The key step in this definition is that we exchange the order of the y and θ integrations. Nevertheless, with this definition at hand, we can *define* the uncertainty on the marginalized distribution as

$$\sigma^2_{\text{pred}}(x) = \int d\theta\, q(\theta) \left[p(x|\theta) - p(x)\right]^2 \; . \tag{6.34}$$

We illustrate this construction with a trivial $p(x,y) = p(x,y_0)$, where we can replace the trivial y-dependence by a fixed choice $y = y_0$ just like for the wedge and kicker ramps. Here we find, modulo a normalization constant in the y-integration

$$\begin{aligned}
\sigma^2_{\text{pred}}(x) &= \int d\theta\, q(\theta) \left[p(x|\theta) - p(x)\right]^2 \\
&= \int d\theta\, q(\theta) \int dy \;\; \left[p(x,y_0|\theta) - p(x,y_0)\right]^2 \\
&= \int dy d\theta\, q(\theta) \;\; \left[p(x,y_0|\theta) - p(x,y_0)\right]^2 = \int dy\, \sigma^2_{\text{pred}}(x,y_0) = \sigma^2_{\text{pred}}(x,y_0) \; .
\end{aligned} \tag{6.35}$$

Adding a trivial y-direction does not affect the predictive uncertainty in the physical x-direction.

As mentioned above, unlike for the joint density, $p(x,y|\theta)$ we do not know the closed form of the marginal distributions $p(x)$ or $p(x|\theta)$. Instead, we can approximate the marginalized uncertainties through a combined sampling in y and θ. We start with one set of weights θ_i from the weight distributions, based on one random number per INN weight. We now sample N points in the latent space, z_j, and compute N phase space point x_j using the BINN configuration θ_i. We then bin the wanted phase space direction x and approximate $p(x|\theta_i)$ by a histogram. We repeat this procedure $i = 1 \ldots M$ times to extract M histograms with identical binning. This allows us to compute a mean and a standard deviation from M histograms to approximates $p(x)$ and $\sigma_{\text{pred}}(x)$. The approximation of σ_{pred} should be an over-estimate, because it includes the statistical uncertainty related to a finite

number of samples per bin. For $N \gg 1$ this contribution should become negligible. With this procedure we effectively sample $N \times M$ points in phase space.

Following Eq.(6.33), we can also fix the phase space points, so instead of sampling for each weight sample another set of phase space points, we use the same phase space points for each weight sampling. This should stabilize the statistical fluctuations, but with the drawback of relying only on an effective number of N phase space points. Both approaches lead to the same σ_{pred} for sufficiently large N, which we typically set to 10^5 ... 10^6. For the Bayesian weights we find stable results for $M = 30$ ... 50.

In Fig. 6.8 we show the marginalized densities and predictive uncertainties for the kicker ramp. In y-direction the density and the predictive uncertainty show the expected flat behavior. The only exception are the phase space boundaries, where the density starts to deviate slightly from the training data and the uncertainty correctly reflects that instability. In x-direction, the marginalized density and uncertainty can be compared to their one-dimensional counterparts in Fig.6.4. While we expect the same peak-dip structure, the key question is if the numerical values for $\sigma_{\text{pred}}(x)$ change. If the network learns the y-direction as uncorrelated additional data, the marginalized uncertainty should decrease through a larger effective training sample. This is what we typically see for Monte Carlo simulations, where a combination of bins in an unobserved directions leads to the usual reduced statistical uncertainty. On the other hand, if the network learns that the y-directions is flat, then adding events in this direction will have no effect on the uncertainty of the marginalized distribution. This would correspond to a set of fully correlated bins, where a combination will not lead to any improvement in the uncertainty. In Fig. 6.8 we see that the $\sigma_{\text{pred}}(x)$ values on the peak, in the dip, and to the upper end of the phase space boundary hardly change from the one-dimensional results in Fig.6.4. This confirms our general observation, that the (B)INN learns a functional form of the density in both directions, in close analogy to a fit. It also means that the uncertainty from the generative network training is not described by the simple statistical scaling we observed for simpler networks [**?**,**?**] and instead points towards a GANplification-like [**?**] pattern.

6.4 LHC events with uncertainties

As a physics example we consider the Drell-Yan process

$$pp \to Z \to e^+e^- \,, \tag{6.36}$$

with its simple $2 \to 2$ phase space combined with the parton density. The training set consists of an unweighted set of 4-vectors simulated with madgraph [**?**] at 13 TeV collider energy with the NNPDF2.3 parton densities [**?**]. We fix the masses of the final-state leptons and enforce momentum conservation in the transverse direction, which leaves us with a four-dimensional phase space. In our discussion we limit ourselves to a sufficiently large set of one-dimensional distributions. For these marginalized uncertainties we follow the procedure laid out in Sec. 6.3.5 with 50 samples in the BINN-weight space. In Tab. 6.2 we give the relevant hyper-parameters for this section.

Parameter	Flow
Hidden layers (per block)	2
Units per hidden layer	64
Batch size	512
Epochs	500
Trainable weights	$\sim$ 182k
Number of training events	$\sim$ 1M
Optimizer	Adam
(α, β_1, β_2)	(1×10^{-3}, 0.9, 0.999)
Coupling layers	20
Prior width	1

Table 6.2: Hyper-parameters for the Drell-Yan data set, implemented in pytorch(v1.4.0) [**?**].

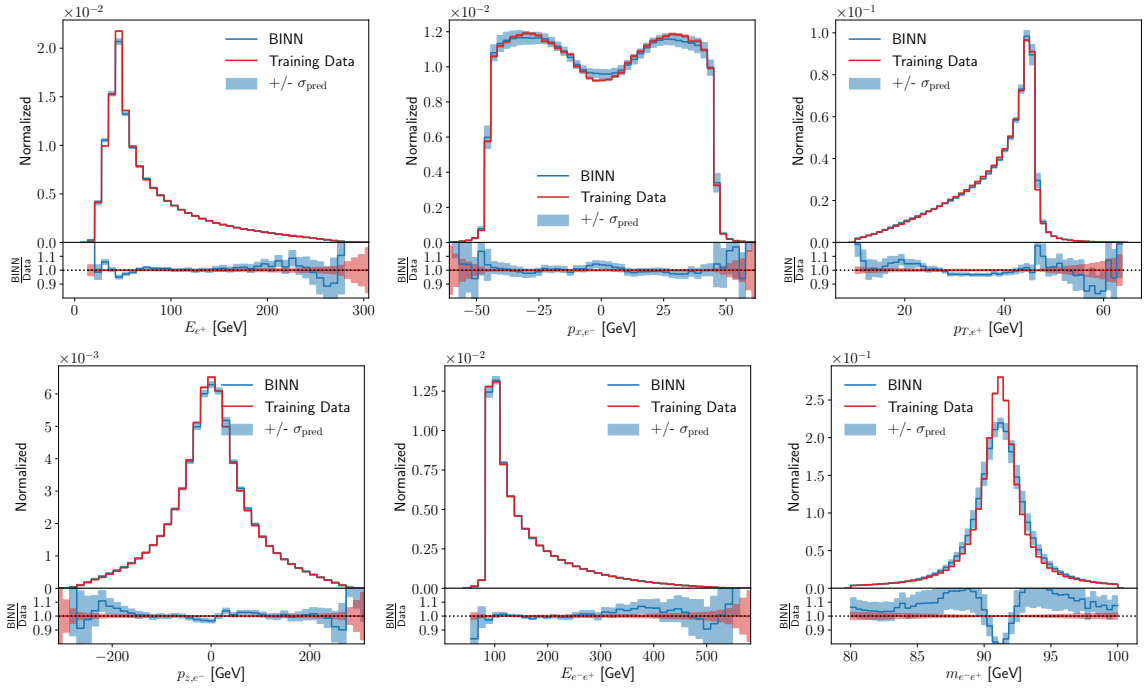

Figure 6.9: One-dimensional (marginalized) kinematic distributions for the Drell-Yan process. We show the central prediction from the BINN and include the predictive uncertainty from the BINN as the blue band. The red band indicates the statistical uncertainty of the training data per bin in the Gaussian limit.

To start with, we show a set of generated kinematic distributions in Fig. 6.9. The positron energy features the expected strong peak from the Z-resonance. Its sizeable tail to larger energies is well described by the training data to $E_e \approx 280$ GeV. The central value learned by the BINN becomes unstable at slightly lower values of 250 GeV, as expected. The momentum component p_x is not observable given the azimuthal symmetry of the detector, but it's broad distribution is nevertheless reproduced correctly. The predictive uncertainty covers the slight deviations over the entire range. What is observable at the LHC is the transverse momentum of the outgoing leptons, with a similar distribution as the energy, just with the Z-mass peak at the upper end of the distribution. Again, the predictive uncertainty determined by the BINN covers the slight deviations from the truth on the pole and in both tails. In the second row we show the p_z component as an example for a strongly peaked distribution, similar to the Gaussian toy model in Sec. 6.3.3.

While the energy of the lepton pair has a similar basic form as the individual energies, we also show the invariant mass of the electron-positron pair, which is described by the usual Breit-Wigner peak. It is well known that this intermediate resonance is especially hard to learn for a network, because it forms a narrow, highly correlated phase space structure. Going beyond the precision shown here would for instance require an additional MMD loss, as described in Ref. [**?**] and in more detail in Ref. [**?**]. This resonance peak is the only distribution, where the predictive uncertainty does not cover the deviation of the BINN density from the truth. This apparent failure corresponds to the fact that generative networks always overestimate the width and hence underestimate the height of this mass peak [**?**]. This is an example of the network being limited by the expressive power in phase space resolution, generating an uncertainty which the Bayesian version cannot account for.

In Fig. 6.10 we show a set of absolute and relative uncertainties from the BINN. The strong peak combined with a narrow tail in the E_e distribution shows two interesting features. Just above the peak the absolute uncertainty drops more rapidly than expected, a feature shared by the wedge and kicker ramps at their respective upper phase space boundaries. The shoulder around $E_e \approx 280$ GeV indicates that for a while the predictive uncertainty follows the increasingly poor modelling of the phase

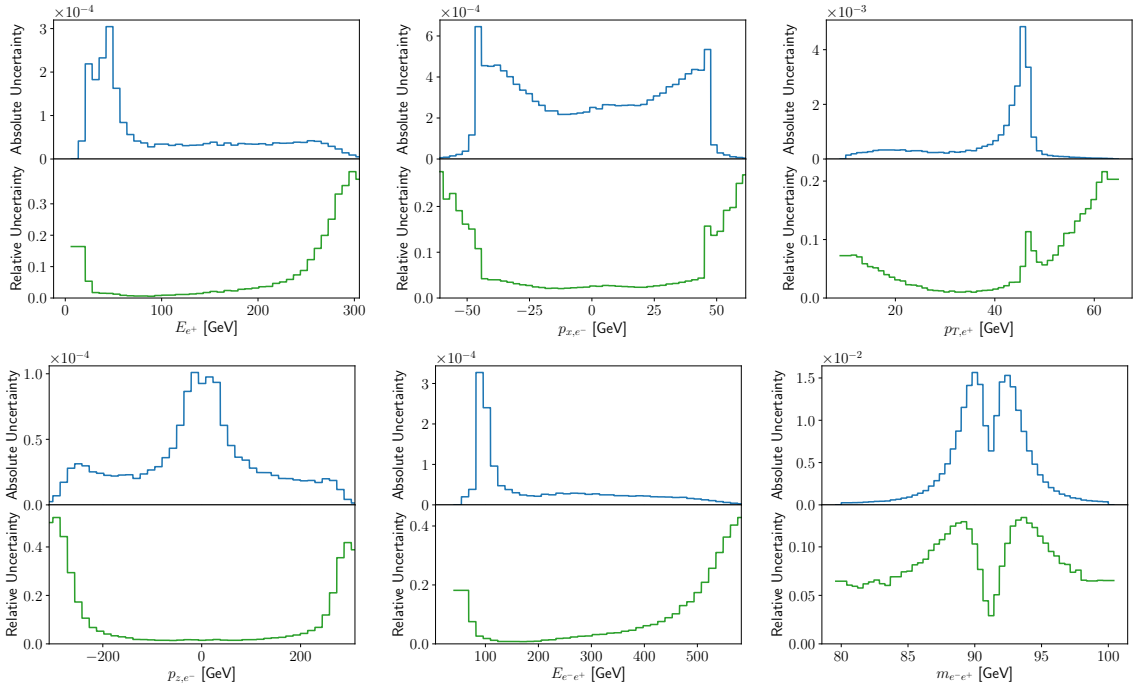

Figure 6.10: Absolute and relative uncertainties as a function of some of the kinematic Drell-Yan observables shown in Fig. 6.9.

space density by the BINN, to a point where the network stops following the truth curve altogether and the predictive uncertainty is limited by the expressive power of the network. Unlike the absolute uncertainty, the relative uncertainty keeps growing for increasing values of E_e. This behavior illustrates that in phase space regions where the BINN starts failing altogether, we cannot trust the predictive uncertainty either, but we see a pattern in the intermediate phase space regime where the network starts failing.

The second kinematic quantity we select is the (unobservable) x-component of the momentum. It forms a relative flat central plateau with sharp cliffs at each side. Any network will have trouble learning the exact shape of such sharp phase space patterns. Here the BINN keeps track of this, the absolute and the relative predictive uncertainties indeed explode. The only difference between the two is that the (learned) density at the foot of the plateau drops even faster than the learned absolute uncertainty, so their ratio keeps growing.

Finally, we show the result for the Breit-Wigner mass peak, the physical counterpart of the Gaussian ring model of Sec. 6.3.3. Indeed, we see exactly the same pattern, namely a distinctive minimum in the predictive uncertainty right on the mass peak. This pattern can be explained by the network learning the general form of a mass peak and then adjusting the mean and the width of this peak. Learning the peak position leads to a minimum of the uncertainty right at the peak, and learning the width brings up two maxima on the shoulders of the mass peak. In combination Fig. 6.9 and 6.10 clearly show that the BINN traces uncertainties in generated LHC events just as for the toy models. Again, some distinctive patterns in the predictive uncertainty can be explained by the way the network learns the phase space density.

6.5 Outlook

Controlling the output of generative networks and quantifying their uncertainties is the main task for any application in LHC physics, be it in forward generation, inversion, or inference. We have proposed to use a Bayesian invertible network (BINN) to quantify the uncertainties from the network training

for each generated event. For a series of two-dimensional toy models and an LHC-inspired application we have shown how the Bayesian setup indeed generates an uncertainty distribution, over the full phase space and over marginalized phase spaces. As expected, the learned uncertainty shrinks with an improved training statistics. Our method can be trivially extended from unweighted to weighted events by adapting the simple MLE loss. An intriguing result from our study is that the combined learning of the density and uncertainty distributions allows us to draw conclusions on how a normalizing-flow network like the BINN learns a distribution. We find that the uncertainty distributions are naturally explained by a fit-like behavior of the network, rather than a patch-wise learning of the density. For the LHC, this can be seen for instance in the non-trivial uncertainty for an intermediate Breit-Wigner resonance. These results are another step in understanding GANplification patterns [**?**] and might even allow us to use INNs to extrapolate in phase space.

Obviously, it remains to be seen how our observations generalize to other generative networks architectures. For the LHC, the next step should be an in-depth study of INN-like networks applied to event generation.

7 | Latent Space Refinement

8 | Summary