Department of Physics and Astronomy

University of Heidelberg

Master thesis

in Physics

submitted by

Jona Ackerschott

born in Lüdenscheid

2023

Shifting Perspective: Accurate Unfolding Flows for $t\bar{t}h$-Production through Phase-Space Parameterizations

This Master thesis has been carried out by Jona Ackerschott

at the

Institute for Theoretical Physics

under the supervision of

Prof. Dr. rer. nat. Tilman Plehn

and

apl. Prof. Dr. Monica Dunford

Kirchhoff-Institute for Physics

Perspektivwechsel: Präzise Unfolding Flows für $t\bar{t}h$-Produktion mithilfe von Phasenraum-Parametrisierungen

Es wurde untersucht, wie Normalizing-Flow-Unfolding die derzeitige Sensitivität für CP-Verletzung in semi-leptonischer $t\bar{t}h$-Produktion mit $h \to \gamma\gamma$ am HL-LHC potentiell verbessern kann. Wir werden sehen, dass geeignete Parton-Level Phasenraum-Parametrisierungen, ergänzt durch neuartige Spline Coupling-Blöcke, attraktive Werkzeuge zur Rekonstruktion von Massenpeaks und anderer generativer Schwächen sind. Ich stelle ein Flow-Netzwerk vor, das deutlich zwischen Signal-Daten mit einer absoluten CP-Phase von $\pi/4$ und dem SM unterscheiden kann, anhand jeder von vielen präzise rekonstruierten CP-sensitiven Parton-Level Observablen. Das Netzwerk liefert außerdem eine deutliche Sensitivität für den kleineren Wert $\alpha = \pi/8$ und dem vor kurzem demonstrierten HL-LHC-Limit, $\alpha = 13°$. Gleichzeitig konnte keine Sensitivität für das Vorzeichen der CP-Phase erreicht werden. Es gibt allerdings, wie ich demonstrieren werde, starke Indizien dafür, dass verwertbare Informationen über das Vorzeichen bereits in den Detektordaten fehlen, wenn ISR berücksichtigt wird.

Shifting Perspective: Accurate Unfolding Flows for $t\bar{t}h$-Production through Phase-Space Parameterizations

I explored how normalizing flow unfolding can potentially improve the current sensitivity for CP-violation in semi-leptonic $t\bar{t}h$-production with $h \to \gamma\gamma$ at the HL-LHC. We will see that appropriate parton-level phase-space parameterizations, supplemented with novel spline coupling blocks, are attractive tools to reconstruct narrow mass peaks and other generative weaknesses associated with this task. I propose a flow network that can differentiate distinctly between signal-only data, with an absolute CP-phase of $\pi/4$, and the SM, based on each of many precisely reconstructed parton-level CP-sensitive observables. The network further provides evident sensitivity to the lower value $\alpha = \pi/8$ and the recently projected HL-LHC bound for background-inclusive data, $\alpha = 13°$. At the same time, no sensitivity for the sign of the CP-phase could be achieved. There are, however, as I will demonstrate, strong indications that practically usable sign information is already absent in detector-level data if ISR is taken into account.

Contents

1. Introduction

The analysis strategy of large hadron collider (LHC) data has shifted over the last decade. After the discovery of the Higgs boson [1, 2] the era of testing specific postulated models ended [3]. With the abundance of beyond Standard Model (BSM) theories today, modern analysis strategies try to be as model agnostic as possible, often through the use of effective field theorys (EFTs) [4]. At the same time, the planned high luminosity large hadron collider (HL-LHC) is projected to collect 25 times the amount of data from the first two LHC runs, bringing existing analysis methods to their limit [3]. With the simultaneous rise of machine learning, a natural tool has emerged to deal with these new challenges [3, 5–7].

The numerical simulation chain of analytically inaccessible parton shower, hadronization and detector effects, tainting the information of the hard scattering process, is central to LHC analyses. Generative Networks, including generative adversarial networks (GANs) [8–24], variational auto-encoders (VAEs) [19–22, 25, 26], normalizing flows [24, 26–29] and, more recently, diffusion models [30–33] and transformers [32–34] have shown great potential for improvement here [35]. Among these architectures, (conditional) normalizing flows have proven to be especially useful for inverting simulations [36], most importantly in the context of unfolding [37, 38]. Classical techniques for unfolding are limited to the analysis of single observable distributions with a pre-defined binning and require the manual reconstruction of interesting observables on detector-level [39]. This restriction can be lifted through machine learning [37, 40].

In this thesis, a normalizing flow is employed to unfold the phase-space of top-pair associated Higgs production, with a particular focus on CP-sensitive observables. The objective is to improve the detection sensitivity of CP-violation in the Higgs sector. We know that many observables can be important here, most of them difficult to reconstruct from detector data [41]. Normalizing flow unfolding can take care of this reconstruction, not only for a few observables, but for all phase-space dimensions simultaneously [37]. This technique further unfolds single detector events statistically, allowing for valid unfolding with even a low number of events. As we will see, this feature also allows us to analyze the information loss at detector-level in the context of specific observables. On the other hand, the unfolding accuracy is not equally distributed across observables with this method. Intermediate particle mass distributions are a well-known weakness in particular [37], while being abundant in $t\bar{t}h$. I will demonstrate that specific choices of phase-space parameterizations greatly improve the accuracy of intermediate mass distributions, while also allowing us to shift the focus of our network to those observables that are of interest to us. I will further show that information loss induced model dependence becomes an issue when unfolding a complex process like $t\bar{t}h$.

We will see that normalizing flow unfolding gives us excellent accuracy when unfolding many CP-sensitive observables of $t\bar{t}h$, while specifically struggling with observables that would allow us to test CP-symmetry directly. In particular, we will discuss some indications that there might be no practical information at all about these observables in $t\bar{t}h$ detector data.

The layout of this thesis is as follows. In Chapter 2 we will start with the fundamentals of normalizing flows. An introduction to unfolding will be given in Chapter 3. We will also discuss and compare classical unfolding techniques as well as Omnifold and normalizing flow unfolding. For Chapter 4 we will cover the basics of CP-sensitive observables in the context of CP-violation in the Higgs sector and specifically review CP-sensitive $t\bar{t}h$ observables. Finally a flow-based unfolding model for $t\bar{t}h$, with a focus on CP-sensitive observables, will be presented in Chapter 5. Conclusions will be summarized in Chapter 6.

2. Normalizing Flows

In the bigger part of this chapter, we will closely follow refs. [42, 43]. Normalizing flows are diffeomorphisms, often constructed with a neural network, that model complex probability distributions by transforming (normalizing) them into a simple base distribution [44, 45]. Inverting this diffeomorphism then allows for efficient sampling and density estimation of the target distribution.

More formally, suppose we have a probability density $q : \mathbb{R}^n \to \mathbb{R}$. When we apply a diffeomorphism $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^n$ to q, we can compute the resulting density $p(\boldsymbol{x})$ with the Jacobian determinant of $\boldsymbol{f} = \boldsymbol{g}^{-1}$ as[1]

$$p(\boldsymbol{x}) = q(\boldsymbol{f}(\boldsymbol{x})) \left|\det(\mathrm{D}\boldsymbol{f}(\boldsymbol{x}))\right| . \tag{2.1}$$

In the following, we will refer to q as the *base distribution*, its domain as the *base domain*, to p as the *target distribution* and to its domain as the *target domain*. Furthermore, we will call the components z_i, x_i of elements from either the base or target domain *channels*.

Note that we can construct any distribution p this way; as long as reasonable assumptions on p and q are satisfied, it is always possible to find a fitting diffeomorphism [43, 46]. Naturally, this diffeomorphism can be arbitrarily complex, so the question arises of how we can construct g in general. There are two ways: *finite composition* and *continuous-time flows*, both usually involving neural networks [43]. We will introduce both of these approaches in the following sections.

[1] More technically, we apply $\boldsymbol{g}$ by computing the pushforward of the underlying probability measure of q.

In addition to being able to model flexible diffeomorphisms, we need a method of fitting such a transformation to a target distribution. Suppose we have a base distribution $q(\boldsymbol{z})$ and a diffeomorphism $\boldsymbol{g}_{\boldsymbol{\phi}}$. The goal is to find an optimal set of parameters $\boldsymbol{\phi}$ for which $\boldsymbol{g}_{\boldsymbol{\phi}}$ models a certain target distribution $p(\boldsymbol{x})$ as closely as possible. For this we can optimize $\boldsymbol{\phi}$ with respect to some objective $\mathcal{L}(\boldsymbol{\phi})$. One of the most popular choice here is the Kullback-Leibler (KL) divergence [47], which measures the discrepancy of two arbitrary distributions $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ as the expectation of their logarithmic difference, i.e.

$$\mathrm{KL}(p(\boldsymbol{x}) \parallel q(\boldsymbol{x})) := E_{x\sim p(\boldsymbol{x})}\left[\log\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\right)\right]. \tag{2.2}$$

Hence, we can use the KL-divergence as an objective to minimize this discrepancy, giving us two good choices for $\mathcal{L}(\boldsymbol{\phi})$. The first of these is the forward KL divergence, defined as

$$\mathcal{L}(\boldsymbol{\phi}) = \mathrm{KL}(p(\boldsymbol{x}) \parallel p(\boldsymbol{x}\,|\,\boldsymbol{\phi})), \tag{2.3}$$

where $p(\boldsymbol{x}\,|\,\boldsymbol{\phi})$ is the distribution that our model generates from the base distribution, given parameters $\boldsymbol{\phi}$. When writing this out, using the change of variables formula from Eq. (2.1) and ignoring terms constant in $\boldsymbol{\phi}$, we obtain

$$\mathcal{L}(\boldsymbol{\phi}) = -E_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[\log(p(\boldsymbol{x}\,|\,\boldsymbol{\phi}))\right]. \tag{2.4}$$

Conversely, if we swap the two arguments in Eq. (2.3) we get the reverse KL divergence

$$\begin{aligned}\mathcal{L}(\boldsymbol{\phi}) &= \mathrm{KL}(p(\boldsymbol{x}\,|\,\boldsymbol{\phi}) \parallel p(\boldsymbol{x})) \\ &= E_{x\sim p(\boldsymbol{x}\,|\,\boldsymbol{\phi})}\left[\log\left(\frac{p(\boldsymbol{x}\,|\,\boldsymbol{\phi})}{p(\boldsymbol{x})}\right)\right],\end{aligned} \tag{2.5}$$

This one yields

$$\begin{aligned}\mathcal{L}(\boldsymbol{\phi}) &= E_{x\sim p(\boldsymbol{x}|\boldsymbol{\phi})}\left[\log(p(\boldsymbol{x}|\boldsymbol{\phi})) - \log(p(\boldsymbol{x}))\right] \\ &= -E_{\boldsymbol{z}\sim q(\boldsymbol{z})}\left[\log(p(\boldsymbol{g}_{\boldsymbol{\phi}}(\boldsymbol{z}))) + \log|\det(\mathrm{D}\,\boldsymbol{g}_{\boldsymbol{\phi}}(\boldsymbol{z}))|\right].\end{aligned} \tag{2.6}$$

Again we have ignored constant terms.

The decision of which objective to use typically boils down to the question if evaluating the target density $p(x)$ or sampling from it is cheaper. In the first case, computing the expression for the reverse KL divergence is preferable, while the forward divergence is a better choice in the second case. Additionally the forward KL-divergence, as a pure log-likelihood loss when written out, is the only option if we want to built a Bayesian model. We will discuss this further in Section 2.5. Note that we need to approximate the expectation value in Eq. (2.4) and Eq. (2.6) in practice, typically by Monte Carlo methods.

2.1. Finite Composition Architectures

Finite composition flows are built by dividing $\boldsymbol{g}$ into a finite number of simpler diffeomorphisms $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m$, i.e.

$$\boldsymbol{g} = \boldsymbol{g}_m \circ \ldots \circ \boldsymbol{g}_1. \tag{2.7}$$

This allows us to focus on a single $\boldsymbol{g}_i$ without having to worry about global issues, since we can construct the inverse and Jacobian from the components. In particular if $\boldsymbol{f}_1 \circ \ldots \circ \boldsymbol{f}_m$ are the respective inverses of the $\boldsymbol{g}_i$'s, we get

$$\boldsymbol{f} = \boldsymbol{f}_1 \circ \ldots \circ \boldsymbol{f}_m, \tag{2.8}$$

while the determinant of the Jacobian can be computed as

$$\det(\mathrm{D}\,\boldsymbol{f}(\boldsymbol{x})) = \prod_{i=1}^{m} \det(\mathrm{D}\,\boldsymbol{f}_i(\boldsymbol{x})). \tag{2.9}$$

2.1.1. Linear Flows

A reasonable starting point is considering linear components, i.e.

$$\boldsymbol{g}_i(\boldsymbol{z}) = A\boldsymbol{z}, \tag{2.10}$$

with an invertible matrix $A \in \mathbb{R}^{n\times n}$. For the determinant we get

$$\det(\mathrm{D}\,\boldsymbol{g}_i(\boldsymbol{z})) = \det(A) \tag{2.11}$$

In practical implementations, we have to restrict the form of A in some way for three reasons: A must be invertible, the parameterization of A has to be continuous and the inversion of A needs to be efficient. Particularly a continuous parameterization is not possible for general invertible matrices due to the discontinuous jump of corresponding determinants at zero. Concerning the last point, we specifically want to invert A faster than the $\mathcal{O}(n^3)$ time complexity required for arbitrary non-singular matrices. Factorizing A using a PLU or QR-decomposition as well as restricting A to be orthogonal are possible solutions to these problems [43].

It should be obvious that linear flows on their own are insufficient to model flexible diffeomorphisms. For instance, we cannot model non-exponential distributions with an exponential base distribution using just linear transformations. However linear flows are an important component to use in conjunction with autoregressive architectures [42, 43]. Particularly raw permutation matrices in Eq. (2.10) are often used in practice. There also is the slightly more general approach of using 'soft' permutation matrices $A \in \mathrm{SU}(n)$, which, however, scale poorly with n [48].

2.1.2. Autoregressive Flows

Autoregressive flow architectures construct $\boldsymbol{g}_i$ as

$$(\boldsymbol{g}_i(\boldsymbol{z}))_j = \tau\left(z_j, \mathbf{c}_j(z_1, \ldots, z_{j-1})\right) \tag{2.12}$$

with an invertible function τ, called the *transformer*,[2] and functions $\boldsymbol{c}_j$, called the *conditioners*. The conditioners do not have particular constraints and their dimensionality can be chosen arbitrarily to fit the respective transformer implementation. The definition Eq. (2.12) is invertible, as it should be, with the inverse

$$(\boldsymbol{f}_i(\boldsymbol{x}))_j = \boldsymbol{\tau}^{-1}(x_j, \boldsymbol{c}_j(z_1, \ldots, z_{j-1})). \tag{2.13}$$

Here $z_1, \ldots, z_{j-1} = (\boldsymbol{f}_i(\boldsymbol{x}))_1, \ldots, (\boldsymbol{f}_i(\boldsymbol{x}))_{j-1}$. Furthermore, the Jacobian of an autoregressive $\boldsymbol{g}_i$ is triangular. Because of this, we can simply compute the Jacobian determinant by multiplying the diagonal elements. This reduces the time complexity of this operation from $\mathcal{O}(n^3)$ to $\mathcal{O}(n)$.

Let me emphasize that this construction of normalizing flows is still *universal*, i.e. autoregressive flows, in theory, can still transform any base into any target distribution [43]. In practice we will lose this universality and autoregressive architectures can depend on the order of their inputs. For this reason, they can benefit from using multiple sub-flows $\boldsymbol{g}_i$ and interleaving them with linear flows—usually permutations—to mix up the channel order [43]. There are quite a few options for the transformer and the conditioner now.

Conditioner Architectures

A straightforward way to implement the conditioner is to model it directly as a neural network, for example, using a simple fully connected architecture. One can do this by taking a network that depends on the full input $\boldsymbol{z}$ and then

[2] Not to be confused with transformer models [49], which we will not mention again here. This terminology is adopted from Ref. [43] and will be used consistently in this thesis, due to lack of a better term.

removing connections between the i-th input and the first to i-th output. This is typically done by multiplying the weights with binary masking matrices.

Another option is to use a recurrent neural network (RNN) for $\boldsymbol{c}$. Here we start with an initial state $\boldsymbol{s}_1 \in \mathbb{R}^{j-1}$ which we iteratively update with an RNN R according to

$$\boldsymbol{s}_k = R(z_{k-1}, \boldsymbol{s}_{k-1}). \tag{2.14}$$

After we have obtained $\boldsymbol{s}_j$ we evaluate the conditioner as

$$\boldsymbol{c}_j(z_1, \dots, z_{j-1}) = \boldsymbol{c}_j(\boldsymbol{s}_j). \tag{2.15}$$

The initial state can either be chosen manually or added to the RNN as a parameter to be learned.

Both of these conditioner implementations are quite inefficient. RNN conditioners have to be evaluated sequentially such that evaluation has a time complexity of $\mathcal{O}(n)$. Masked conditioners are fast when evaluating the flow in the forward direction, but the efficiency of the inverse direction also grows with n [43]. On the other hand, masked autoregressive flows do not introduce additional restrictions on the conditioner and therefore do not sacrifice the universal approximation properties of autoregressive flows, provided the underlying network architecture is universal.

That being said if we want to evaluate both the forward and backward direction of the flow efficiently, *coupling blocks* are a good choice [50, 51]. Here, we choose some integer $k \in \{1, \dots, n\}$, often as $k = n/2$ rounded to an integer, and implement the conditioner as

$$\boldsymbol{c}_j(z_1, \dots, z_{j-1}) = \begin{cases} \textbf{const.} & \text{if} \quad j \leq k \\ \boldsymbol{F}(z_1, \dots, z_k) & \text{if} \quad j > k \end{cases}, \tag{2.16}$$

where $\boldsymbol{F}$ is an arbitrary function. In practice we would split the input $\boldsymbol{z}$ into two parts $\boldsymbol{z}_A = (z_1, \dots, z_k)$ and $\boldsymbol{z}_B = (z_{k+1}, \dots, z_n)$, which are then transformed pointwise—$\boldsymbol{z}_A$ independently and $\boldsymbol{z}_B$ in dependence of $\boldsymbol{z}_A$. One typically

alters the transformer architecture here and uses the identity for arguments $\boldsymbol{c}_j(z_1, \dots, z_{j-1})$ with $j \leq k$.

Inversion is trivial: Since the transformation of $\boldsymbol{z}_A$ is done independently from $\boldsymbol{z}_B$ we can undo it in parallel and use the result to invert the transformation of $\boldsymbol{z}_B$—also in parallel. This makes evaluation in both directions equally fast, as promised.

The downside of this efficiency is losing the approximative power of autoregressive flows. When composing multiple coupling blocks, while permuting the channels in between, it was shown that we regain quite universal networks, both empirically [52–54] and theoretically. On the theoretical side, however, the proof involves concatenating D coupling blocks to recreate a fully autoregressive flow, obviously defeating the purpose of being faster than other methods [43].

In reality, implementing coupling blocks as compositions with permutations in between them—or, more generally, linear transformations—is of course both helpful and necessary. Otherwise, there would always be channels which are never transformed by anything other than the identity.

Transformer Architectures

Let us start with a linear (affine) implementation:

$$\tau(z_j, \alpha_j, \beta_j) = \alpha_j z_j + \beta_j. \tag{2.17}$$

To ensure invertibility we can let the conditioner compute a parameter $\tilde{\alpha}_j$ instead and choose $\alpha_j = \exp(\tilde{\alpha}_j) \neq 0$. The Jacobian determinant becomes

$$\log|\det(\mathrm{D}\,\tau(\boldsymbol{z}))| = \sum_{i=1}^{n} \log|\alpha_i|. \tag{2.18}$$

This transformer form is very efficient and analytically invertible but lacks expressive power. Nonetheless, affine transformers can still produce flexible

networks, provided we use a sufficient number of $\boldsymbol{g}_i$'s in our flow composition.

More flexible transformers can be constructed by using a conical sum or composition of K simple functions τ_k, i.e.

$$\begin{aligned} \tau(\boldsymbol{z}, \alpha_k) &= \sum_{k=1}^{K} \alpha_k \tau_k(\boldsymbol{z}) \quad \text{or} \\ \tau(\boldsymbol{z}) &= \tau_K \circ \ldots \circ \tau_1(\boldsymbol{z}). \end{aligned} \tag{2.19}$$

The idea is that if the τ_k's are strictly monotonic, then τ will also be strictly monotonic and hence invertible. If we use monotonically increasing activation functions we can for example construct a fully connected neural network using this method, retaining universality through their flexibility. In this case, we can obtain the Jacobian determinant through backpropagation. In general of course these types of transformers might not be analytically invertible.

We can also define a transformer by integrating a positive function h, making the result monotonically increasing (i.e. invertible). In particular

$$\tau(z_j, \boldsymbol{\alpha}_j, \beta_i) = \int_0^{z_j} h(\zeta, \boldsymbol{\alpha}_j)\, \mathrm{d}\zeta + \beta_i. \tag{2.20}$$

With such an approach we can approximate any possible transformer τ. However, if we do not want to use numerical methods to invert τ, we have to constrain the form of h. For instance, we can make h a positive polynomial of degree $2K$, which can be written as the sum of the squares of 2 or more polynomials with degree K (proposition 1.1.2 in Ref. [55]). If we choose K large enough, this choice of h can approximate any monotonically increasing function arbitrarily well [56]. At the same time, a so-constructed transformer is only analytically invertible for $K < 2$, since we can only solve polynomials up to degree four analytically. Note that the integral increases the degree of the polynomial by one, so that $K = 2$, gives us a polynomial of degree five.

Last but not least there are *spline transformers*. These provide an analytically

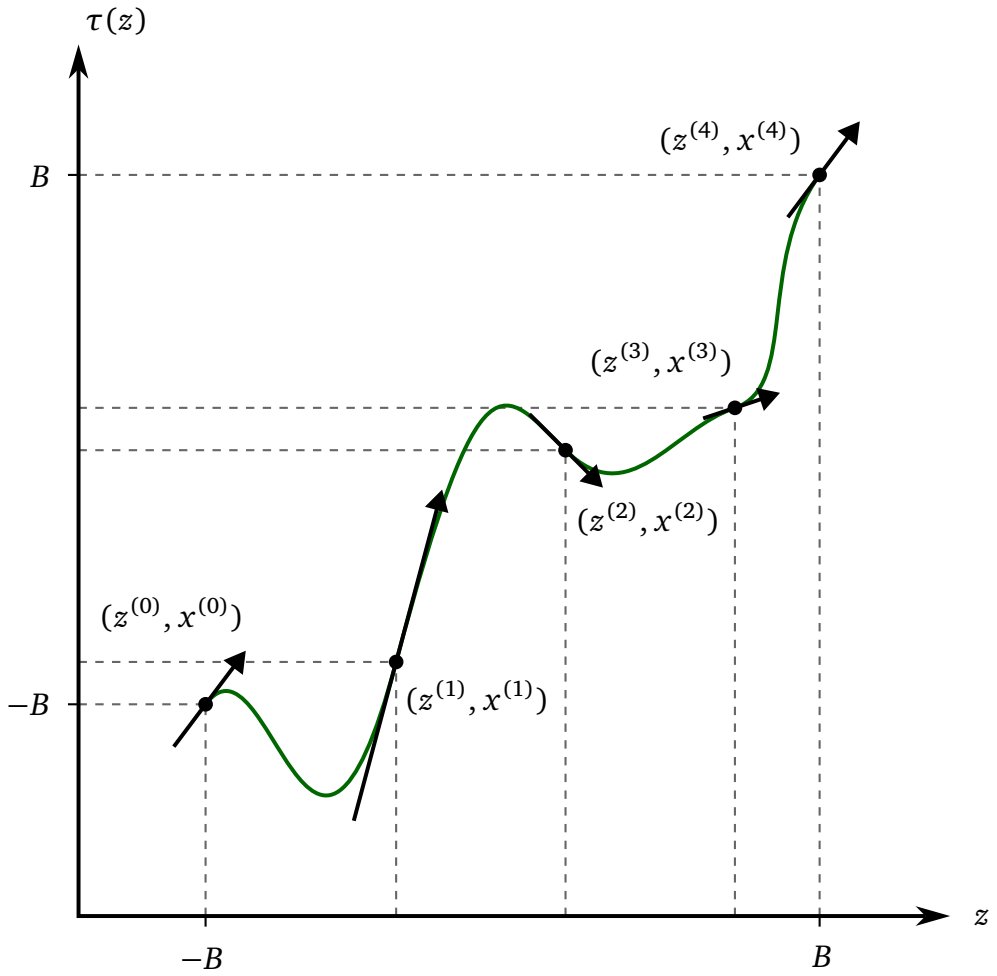

Figure 2.1.: Example of a spline transformation with $K = 4$. Here, also the derivatives at each point must be specified.

invertible alternative to previous transformers, which also have a high expressivity. Here we assume we want to construct a transformer $\tau : [-B, B] \to [-B, B]$, where B specifies the domain boundaries. For simplicity, we look at how τ acts on a single channel z, keeping in mind that all parameters of τ can be different across z_i's. We divide $[-B, B]$ into K intervals with edge points $z^{(k)}$ and construct τ piecewise by K simple transformations (cf. Fig. 2.1). In particular we define $K+1$ points $x^{(k)}$ and K transformations $\tau_k : [z^{(k)}, z^{(k+1)}] \to [x^{(k)}, x^{(k+1)}]$,

such that

$$\tau(z) = \tau_\kappa(z), \tag{2.21}$$

with κ chosen to satisfy $z^{(\kappa)} \leq z < z^{(\kappa+1)}$. As an example, we could choose τ_k to be linear, such that τ would become a piecewise linear function. For more complex τ_k (e.g. rational quadratics) we need further constraints and can for instance add the derivatives at each point $z^{(k)}$ as parameters to τ. Note that defining τ only on $[-B, B]$ is slightly restrictive. Realistically we would define B to be large enough to handle most cases and extend τ to be the identity for points outside of $[-B, B]$.

The forward and inverse directions of these transformations can be evaluated equally fast, while the flexibility of splines grows with K. Additionally, increasing flexibility is not too expensive; the evaluation time only grows with $\mathcal{O}(\log(K))$, when we obtain κ using binary search. Examples of spline implementations include linear and quadratic [57], linear-rational [58], cubic [59] and rational quadratic splines [54].

2.1.3. Residual Flows

Residual Flows implement g_i as the identity plus an arbitrary translation, i.e.

$$\boldsymbol{g}_i(\boldsymbol{z}) = \boldsymbol{z} + \boldsymbol{\Delta}(\boldsymbol{z}), \tag{2.22}$$

which is not invertible in general, so one has to restrict the form of $\boldsymbol{\Delta}(\boldsymbol{z})$ somehow.

One way to do this is to make $\boldsymbol{\Delta}(\boldsymbol{z})$ contractive with respect to some metric δ, such that

$$\delta(\boldsymbol{\Delta}(\boldsymbol{z}_1), \boldsymbol{\Delta}(\boldsymbol{z}_2)) \leq L\,\delta(\boldsymbol{z}_1, \boldsymbol{z}_2), \tag{2.23}$$

with $L < 1$ [60, 61]. The invertibility of $\boldsymbol{g}_i(\boldsymbol{z})$ is then easy to show using the Banach fixed-point theorem, which also suggests an iterative algorithm to compute the inverse [43]. Such contractive flows do not suffer from the

sparse Jacobians that are required for autoregressive architectures, which makes them very flexible. However, these more complex Jacobians are also expensive to compute which, together with the necessity of iterative sampling, often makes this architecture undesirable in practice.

There also exist other approaches to residual flows, known as planar, Sylvester and radial flows [44, 45, 62]. However, all of these methods suffer from inverses that are not analytically tractable and involve quite simple transformations, limiting their expressiveness.

2.2. Continuous Architectures

Continuous-time flows (CTFs) are a second way of constructing normalizing flows, which we will discuss briefly for completeness. Instead of transforming our input z with a finite number m of transformations $\boldsymbol{g}_i$, we want to transform z continuously, giving us a function $\boldsymbol{u}(t)$. Now, our original input to the flow is $\boldsymbol{z} = \boldsymbol{u}(t_0)$, while the output becomes $\boldsymbol{u}(t_1)$, for some numbers t_0 and $t_1 > t_0$. The new transformation is controlled by giving an expression for the derivative of $\boldsymbol{u}(t)$, yielding an ordinary differential equation (ODE)

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t}(t) = \boldsymbol{h}(t, \boldsymbol{u}(t)). \tag{2.24}$$

If $\boldsymbol{h}$ is uniformly Lipschitz continuous in its second argument and continuous in t, it follows from the Picard-Lindelöf theorem that there exists a unique solution of Eq. (2.24). This also implies that $\boldsymbol{u}(t_1)$ is uniquely determined for each initial condition $\boldsymbol{u}(t_0)$, proving invertibility of CTFs. These continuity conditions can be implemented into many neural network architectures [63], while there are no other restrictions on $\boldsymbol{h}$—as opposed to finite composition flows.

Computing the actual flow transformation boils down to

$$\boldsymbol{g}(\boldsymbol{z}) = \boldsymbol{z} + \int_{t_0}^{t_1} \boldsymbol{h}(t, \boldsymbol{u}(t))\, \mathrm{d}t, \tag{2.25}$$

where $\boldsymbol{u}(t)$ is the solution of Eq. (2.24) with initial condition $\boldsymbol{u}(t_0) = \boldsymbol{z}$. The inverse is given as

$$\boldsymbol{g}^{-1}(\boldsymbol{x}) = \boldsymbol{x} - \int_{t_0}^{t_1} \boldsymbol{h}(t, \boldsymbol{u}(t))\, \mathrm{d}t. \tag{2.26}$$

Note that evaluation and inversion are completely equivalent in terms of computational cost, which is not always true for autoregressive flows.

Instead of the Jacobian of $\boldsymbol{g}$ we can compute the change in the logarithm of the base density. One can show that this is equal to the negative trace of the derivative of $\boldsymbol{h}$ [64]:

$$\frac{\mathrm{d} \log(q(\boldsymbol{z}(t)))}{\mathrm{d}t} = -\operatorname{Tr}\left(\mathrm{D}_{\boldsymbol{u}}\, \boldsymbol{h}(t, \boldsymbol{u}(t))\right). \tag{2.27}$$

If we integrate this, we obtain the density transformation under the flow:

$$\log(p(\boldsymbol{x})) = \log(q(\boldsymbol{z})) - \int_{t_0}^{t_1} \operatorname{Tr}\left(\mathrm{D}_{\boldsymbol{u}}\, \boldsymbol{h}(t, \boldsymbol{u}(t))\right) \mathrm{d}t. \tag{2.28}$$

Evaluating the Jacobian like this is computationally expensive however and requires $\mathcal{O}(D)$ passes of backpropagation. There are ways to remedy this, by either approximating the trace in Eq. (2.28) [65] or by constraining the architecture of $\boldsymbol{h}$ [66].

It should be clear that the integrals in Eq. (2.25) and Eq. (2.28) are in general not analytically feasible. We therefore need to use a numerical ODE solving method—either directly or in conjunction with the so-called adjoint method [64].

2.3. Conditional Flows

A conditional normalizing flows is a simple and worthwhile extension to standard flows. These allow us to learn conditional densities and are based on the observation that in

$$\mathcal{L}(\boldsymbol{\phi}) = \mathrm{KL}(p(\boldsymbol{x}) \parallel p(\boldsymbol{x}\,|\,\boldsymbol{\phi})), \tag{2.29}$$

nothing prevents us from replacing the target density $p(\boldsymbol{x})$ by a density $p(\boldsymbol{x}\,|\,\boldsymbol{c})$, yielding.[3]

$$\begin{aligned}\mathcal{L}(\boldsymbol{\phi}) &= \mathrm{KL}(p(\boldsymbol{x}\,|\,\boldsymbol{c}) \parallel p(\boldsymbol{x}\,|\,\boldsymbol{c}, \boldsymbol{\phi})) \\ &= -E_{\boldsymbol{x},\boldsymbol{c}\sim p(\boldsymbol{x}\,|\,\boldsymbol{c})}\left[\log(q(\boldsymbol{f}_{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{c}))) + \log|\det(\mathrm{D}\boldsymbol{f}_{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{c}))|\right].\end{aligned} \tag{2.30}$$

Here we now compute the expectation value over the conditional distribution. In reality, we would compute a Monte Carlo approximation of the expectation value by sampling matching values $\boldsymbol{x}, \boldsymbol{c}$ from $p(\boldsymbol{x}\,|\,\boldsymbol{c})$. The change of variable formula we use is

$$p(\boldsymbol{x}\,|\,\boldsymbol{c}) = q(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{c}))\,|\det(\mathrm{D}\boldsymbol{f}_{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{c}))|. \tag{2.31}$$

Ultimately conditional flows boil down to allowing for an additional argument $\boldsymbol{c}$ in $\boldsymbol{g}_{\boldsymbol{\phi}}$, such that $\boldsymbol{g}_{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{c})$ is a diffeomorphism if we fix $\boldsymbol{c}$ [67]. For e.g. autoregressive flows we would implement this by appending $\boldsymbol{c}$ to the arguments of the conditioner.

2.4. Flows on Riemannian Manifolds

In some instances our probability distribution $p(\boldsymbol{x})$ might not live in Euclidean space, for example if we parameterize $\boldsymbol{x}$ in terms of spherical coordinates. In

[3] In principle we could also allow for the base distribution to be conditioned on $\boldsymbol{c}$

these cases, applying the usual normalizing flow constructions defined on $\mathbb{R}^n$ can lead to problems, as we will see in Section 5.3.

Instead we would like to define our flow as $\boldsymbol{g} : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are two n-dimensional manifolds. It is clear that $\mathcal{X}$ and $\mathcal{Y}$ must be homeomorphic—since $\boldsymbol{g}$ is per definition a homeomorphism—, i.e. topologically equivalent. In the case that $\mathcal{X}$ and $\mathcal{Y}$ are also homeomorphic to $\mathbb{R}^n$ we can just define two maps $\boldsymbol{\phi} : \mathbb{R}^n \to \mathcal{X}$ and $\boldsymbol{\psi} : \mathbb{R}^n \to \mathcal{Y}$ and define g with the help of a standard euclidean flow $\boldsymbol{h}$:

$$\boldsymbol{g}(\boldsymbol{z}) = \boldsymbol{\psi} \circ \boldsymbol{h} \circ \boldsymbol{\phi}^{-1}(\boldsymbol{z}). \tag{2.32}$$

If we just look at $\boldsymbol{\psi}$ for now, we can compute the metric induced from the euclidean metric under $\boldsymbol{\psi}$ as

$$G_{\boldsymbol{\psi}}(\boldsymbol{u}) = \left(\mathrm{D}\,\boldsymbol{\psi}(\boldsymbol{u})\right)^{\top} \left(\mathrm{D}\,\boldsymbol{\psi}(\boldsymbol{u})\right). \tag{2.33}$$

This metric then gives us the volume change under $\boldsymbol{\psi}$. In particular, if we apply $\boldsymbol{\psi}$, any probability density $r(\boldsymbol{u})$ on $\mathbb{R}^n$ gets transformed according to

$$p(\boldsymbol{x}) = r(\boldsymbol{\psi}^{-1}(\boldsymbol{x}))\, \det\left(G_{\boldsymbol{\psi}}(\boldsymbol{\psi}^{-1}(\boldsymbol{x}))\right)^{-1/2}. \tag{2.34}$$

Naturally we can derive an analogous expressions for $\boldsymbol{\phi}$, which enables us to compute the full density transformation under the flow $\boldsymbol{g}(\boldsymbol{z})$.

Unfortunately, the above approach is only well-defined for manifolds $\mathcal{X}$ and $\mathcal{Y}$ that are topologically equivalent to Euclidean space. We can quickly encounter manifolds like spheres, tori or circles, for which this is no longer true. If we e.g. define a similar map as above for a sphere, we will encounter at least one coordinate singularity—usually at a pole. In this case $G_{\boldsymbol{\psi}}$ would vanish at the singularity and create a point of infinite density according to Eq. (2.34). This can lead to numerical instabilities in practice [43].

Overall, to the best of my knowledge, there currently exists—at least for finite composition normalizing flows—no established way of defining a normalizing flow on general Riemannian manifolds [42, 43].

2.5. Bayesian Flows

Bayesian neural networks (BNNs) allow the quantification of uncertainty in the predictions of a neural network. This is an important quality for networks in any discipline, but especially in a physics setting—where no major discovery can be claimed without knowledge about the uncertainty of our data—uncertainty quantification is essential.

The underlying idea of Bayesian networks is quite simple. During training, instead of learning a single set of model parameters $\boldsymbol{\phi}$, that are most likely to produce our output data, we learn a distribution $p(\boldsymbol{\phi})$ of model parameters. This distribution then assigns a probability to each parameter set, indicating how probable it is that the corresponding model reproduces our data. In particular, it captures the uncertainty in our model parameters due to the limited statistics of the training data [68], also called *epistemic uncertainty*. We can use $p(\boldsymbol{\phi})$ for inference and ultimately obtain a probability for each model prediction.

To formalize this, keeping in mind that we want to apply the theory of BNNs to normalizing flows, we will focus on (Bayesian) regression. Note that the rest of this section will be mostly based on [68]. Suppose we have a model $\boldsymbol{f}_{\boldsymbol{\phi}}$ with parameters $\boldsymbol{\phi}$ as well as two datasets $\boldsymbol{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ (input) and $\boldsymbol{Y} = \{\boldsymbol{y}_1, \dots, \boldsymbol{y}_n\}$ (output) where $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ are connected by some functional relation $\boldsymbol{f}$. We now want to find parameters $\boldsymbol{\phi}$ such that $\boldsymbol{f}_{\boldsymbol{\phi}}$ is likely to resemble $\boldsymbol{f}$, i.e. to have generated our data.

The Bayesian idea is postulating some prior distribution $p(\boldsymbol{\phi})$ of our parameters and update it according to Bayes' theorem. We start from the likelihood $p(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{\phi})$ of some $\boldsymbol{x}$ to generate $\boldsymbol{y}$ for model parameters $\boldsymbol{\phi}$. This gives us the posterior

$$p(\boldsymbol{\phi} \,|\, \boldsymbol{X}, \boldsymbol{Y}) = \frac{p(\boldsymbol{Y} \,|\, \boldsymbol{X}, \boldsymbol{\phi}) p(\boldsymbol{\phi})}{p(\boldsymbol{Y} \,|\, \boldsymbol{X})}, \tag{2.35}$$

where the denominator is the model evidence

$$p(\boldsymbol{Y} \mid \boldsymbol{X}) = \int p(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{\phi}) p(\boldsymbol{\phi}) \, \mathrm{d}\boldsymbol{\phi}. \tag{2.36}$$

If we have access to the posterior, we can *infer* the probability density of an output data point $\boldsymbol{y}$ to correspond to any input data point $\boldsymbol{x}$, as

$$p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y}) = \int p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\phi}) p(\boldsymbol{\phi} \mid \boldsymbol{X}, \boldsymbol{Y}) \, \mathrm{d}\boldsymbol{\phi}, \tag{2.37}$$

given the observed datasets $\boldsymbol{X}$ and $\boldsymbol{Y}$. This is called *inference* and gives us our desired probabilistic prediction.

In principle, we would be done here, however, the evidence (cf. Eq. (2.36)) needed for inference is typically intractable for most interesting models. So we need to approximate the true posterior by an easy-to-evaluate distribution $q_{\boldsymbol{\theta}}(\boldsymbol{\phi})$ with parameters $\boldsymbol{\theta}$. This technique is known as *variational inference (VI)*. The quality of this approximation can be optimized by minimizing the KL divergence [47] with respect to $\boldsymbol{\theta}$:

$$\mathrm{KL}(q_{\boldsymbol{\theta}}(\boldsymbol{\phi}) \parallel p(\boldsymbol{\phi} \mid \boldsymbol{X}, \boldsymbol{Y})) = \int q_{\boldsymbol{\theta}}(\boldsymbol{\phi}) \frac{q_{\boldsymbol{\theta}}(\boldsymbol{\phi})}{p(\boldsymbol{\phi} \mid \boldsymbol{X}, \boldsymbol{Y})} \, \mathrm{d}\boldsymbol{\phi}. \tag{2.38}$$

We can rewrite Eq. (2.38) by applying Eq. (2.35) and assuming $\boldsymbol{q_\theta}(\boldsymbol{\phi})$ is normalized. In this case, we get

$$\begin{aligned} &\mathrm{KL}(q_{\boldsymbol{\theta}}(\boldsymbol{\phi}) \parallel p(\boldsymbol{\phi} \mid \boldsymbol{X}, \boldsymbol{Y})) \\ &\quad = \mathrm{KL}(q_{\boldsymbol{\theta}}(\boldsymbol{\phi}) \parallel p(\boldsymbol{\phi})) - \int q_{\boldsymbol{\theta}}(\boldsymbol{\phi}) \log(p(\boldsymbol{Y} \mid \boldsymbol{\phi}, \boldsymbol{X})) + \log(p(\boldsymbol{Y} \mid \boldsymbol{X})). \end{aligned} \tag{2.39}$$

Since the evidence does not depend on $\boldsymbol{\theta}$, we can, fortunately, ignore it during optimization. Instead, we optimize only the first two terms, also known as the *evidence lower bound (ELBO)*. This gives us the following optimization

objective to maximize:

$$\mathcal{L}(\boldsymbol{\theta}) = \int q_{\boldsymbol{\theta}}(\boldsymbol{\phi}) \log(p(\boldsymbol{Y} \mid \boldsymbol{\phi}, \boldsymbol{X})) \, \mathrm{d}\boldsymbol{\phi} - \mathrm{KL}(q_{\boldsymbol{\theta}}(\boldsymbol{\phi}) \parallel p(\boldsymbol{\phi})). \tag{2.40}$$

Intuitively the first term optimizes $\boldsymbol{\theta}$ such that $\boldsymbol{f}_{\boldsymbol{\phi}}$ with $\boldsymbol{\phi} \sim q_{\boldsymbol{\theta}}(\boldsymbol{\phi})$ is on average close to $\boldsymbol{f}$, while the last term punishes distributions $q_{\boldsymbol{\theta}}(\boldsymbol{\phi})$ that differ too much from the prior. The latter is typically true for relatively complex distributions, i.e. the second term acts as a regularization for the model. This built-in regularization is a nice additional benefit of Bayesian networks and one can even show that popular stochastic regularization techniques—for example dropout—can be derived from Bayesian networks [68]. Note that, since $p(\boldsymbol{Y} \mid \boldsymbol{\phi}, \boldsymbol{X}) = \Pi_{i=1}^{N} p(\boldsymbol{y}_i \mid \boldsymbol{\phi}, \boldsymbol{x}_i)$, the log-likelihood term grows with more training data, while the second term in Eq. (2.40) does not. In other words, the prior becomes less and less important for more and more training data. This is exactly what we would expect since $p(\boldsymbol{\phi} \mid \boldsymbol{X}, \boldsymbol{Y})$, and thus $q_{\boldsymbol{\theta}}(\boldsymbol{\phi})$ for optimal training, becomes sharply peaked when the amount of training data grows large.

If we find an optimum $q_{\boldsymbol{\theta}}(\boldsymbol{\phi})$ of Eq. (2.40) we can compute our prediction according to Eq. (2.37) using Monte-Carlo (MC) integration

$$\begin{aligned} p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y}) &\approx \int p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\phi}) q_{\boldsymbol{\theta}}(\boldsymbol{\phi}) \, \mathrm{d}\boldsymbol{\phi} \\ &= E_{q_{\boldsymbol{\theta}}(\boldsymbol{\phi})} \left[p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\phi}) \right] \end{aligned} \tag{2.41}$$

While the KL-divergence term in Eq. (2.40) can be computed directly, we again face a tractability problem with the first log-likelihood term. Writing $p(\boldsymbol{Y} \mid \boldsymbol{\phi}, \boldsymbol{X}) = \prod_{i=1}^{N} p(\boldsymbol{y}_i \mid \boldsymbol{\phi}, \boldsymbol{x}_i)$, we obtain

$$\sum_{i=1}^{n} \int q_{\boldsymbol{\theta}}(\boldsymbol{\phi}) \log(p(\boldsymbol{y}_i \mid \boldsymbol{\phi}, \boldsymbol{x}_i)) \, \mathrm{d}\boldsymbol{\phi}. \tag{2.42}$$

First, we observe that the sum runs over the entire dataset, which is not practical. This can be fixed by using *data sub-sampling* (mini-batch optimization

in the machine learning context). In particular we generate a random set of indices $B \subset \{1, \dots, n\}$ with $|B| = m$ and approximate Eq. (2.42) as

$$-\frac{n}{m} \sum_{i \in B} \int q_{\boldsymbol{\theta}}(\boldsymbol{\phi}) \log(p(\boldsymbol{y}_i | \boldsymbol{\phi}, \boldsymbol{x}_i)) \, \mathrm{d}\boldsymbol{\phi} \tag{2.43}$$

Since we recover Eq. (2.42) as the expectation value of Eq. (2.43), optimization of the latter leads to values of $\boldsymbol{\theta}$ that are also optima of the former [69].

Second, the integral in Eq. (2.42) and Eq. (2.43) cannot be computed analytically. MC estimation is not straightforward either, since $q_{\boldsymbol{\theta}}$ depends on $\boldsymbol{\theta}$ with respect to which we need to differentiate when performing optimization. Luckily, there are techniques to remedy this. In particular, there are three main stochastic estimators for the gradient of this integral: the *score function estimator* [70–73], the *pathwise derivative estimator* (also known as the *re-parameterization trick*) [74–77], and the estimator proposed in Ref. [78] (called *characterstic function estimator* in Ref. [68]).

Empirically the score function estimator seems to have a higher variance than the pathwise derivative estimator while the variance of the characteristic function estimator is (empirically) the lowest. In practice high gradient variance leads to poor performance when optimizing an objective with stochastic gradient descent, such that estimators with lower gradients are preferable. On the other hand, while the characteristic function estimator seems to have low gradient variances, it does not generalize to non-Gaussian parameter distributions $q_{\boldsymbol{\theta}}(\boldsymbol{\phi})$. Therefore we will follow Ref. [68] in considering only the re-parameterization trick to estimate the integral in Eq. (2.43).

The idea is to re-parameterize $q_{\boldsymbol{\theta}}(\boldsymbol{\phi})$ as $q(\boldsymbol{\varepsilon})$ in such a way that $\boldsymbol{\phi} = \boldsymbol{g}(\boldsymbol{\theta}, \boldsymbol{\varepsilon})$, where $\boldsymbol{g}$ is differentiable and deterministic. For instance, we could re-parameterize a one-dimensional Gaussian distribution by writing $\phi = \mu + \sigma\varepsilon$, where we sample ε from a standard Gaussian. Since now $q(\boldsymbol{\varepsilon})$ does not depend on

the parameters $\boldsymbol{\theta}$, we can write the gradient of our integral as

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \int q_\theta(\boldsymbol{\phi}) \log(p(\boldsymbol{y}_i \,|\, \boldsymbol{\phi}, \boldsymbol{x}_i))\, \mathrm{d}\boldsymbol{\phi} \\
&\quad = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \int q(\boldsymbol{\varepsilon}) \log(p(\boldsymbol{y}_i \,|\, \boldsymbol{g}(\boldsymbol{\theta}, \boldsymbol{\varepsilon}), \boldsymbol{x}_i))\, \mathrm{d}\boldsymbol{\varepsilon} \\
&\quad = E_{q(\boldsymbol{\varepsilon})} \left[\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\phi}} \log(p(\boldsymbol{y}_i \,|\, \boldsymbol{\phi}, \boldsymbol{x}_i)) \, \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \boldsymbol{g}(\boldsymbol{\theta}, \boldsymbol{\varepsilon}) \right].
\end{aligned} \tag{2.44}$$

Realistically we compute this gradient by backpropagation.

We can further improve this method, by not sampling $\boldsymbol{\varepsilon}$ directly, which is quite inefficient. Instead we can sample the intermediate variables through which $\boldsymbol{\varepsilon}$ enters the likelihood in Eq. (2.44). This idea is introduced in Ref. [79] and called *local re-parameterization trick*. It greatly reduces the number of samples one has to generate, while also reducing the gradient variance.

The concept of BNNs can be directly applied to normalizing flows. For this we notice that the log-likelihood in the first term of the ELBO (cf. Eq. (2.40)) is simply given as the logarithm of the flows target distribution. Note that the resulting loss is essentially a modified forward KL-divergence (cf. Eq. (2.4)). In terms of rendering the actual model architecture Bayesian, for e.g. auto-regressive flows this boils down to replacing the conditioner with a Bayesian model.

3. Unfolding

Comparing data from particle collision experiments to theory predictions is not straightforward. During measurement, limited acceptance, finite resolution and other detector effects introduce errors that not only distort the produced data but may also lead to loss of information. At the LHC we face additional challenges. QCD effects involved in jet formation can no longer be computed analytically and have to be described with numerical methods, while error-prone jet reconstruction introduces more uncertainties into our data, further complicated by initial and final state radiation.

Fortunately, we have powerful MC-based simulations for these various distortions, such that we can make precise predictions based on a theory Lagrangian [80]. We refer to the application of any connected subset of such effects, happening at some point after the hard scattering, as *folding*. Although folding simulated data and comparing it to raw experimental data is a simple process and allows us to test any theory in principle, there are clear drawbacks to working in this manner. First of all, such data is detector-dependent, making comparisons between different experiments not easily possible [81, 82]. To compare experiment to theory when both is folded, one also needs to know the specifics of the detector, making long-time data preservation and analysis of old data more difficult. Second, detector simulations are slow compared to simulations of parton-level data, i.e. events after the hard scattering process [82]. If one could find the parton-level data corresponding to a measurement once, it would make subsequent analyses significantly faster and would also allow for analysis methods that use analytic expressions only available on parton-level. Third, our analysis might rely on specific observ-

ables that are not easily reconstructable after hadronization. In this case, we would have no other choice but to somehow reconstruct these observables on parton-level. Hence, developing methods and tools to *unfold* experimental data, i.e. to undo the folding, can be beneficial to many analyses.

3.1. Unfolding as an Inverse Problem

Folding and Unfolding are transformations between parton- and detector-level event distributions of a scattering process, where these levels are usually defined very loosely. Generally speaking, parton- and detector-level are stages involving any number of effects that occur—in sequence—after the hard scattering process and before the measurement at the detector. For instance, detector-level usually includes hadronization and detector effects, either with already reconstructed jets (a.k.a. reco-level) or with jet components. Parton-level, on the other hand, is typically defined either before parton-showers and hadronization or after but before detector effects (a.k.a. particle-level). If we let $p_{\text{part}}(\boldsymbol{x})$ and $p_{\text{det}}(\boldsymbol{y})$ denote our parton- and detector-level event distributions, defined on respective phase-spaces Ω_{part} and Ω_{det}, then folding can be described by a Fredholm integral equation of the first kind [83],

$$p_{\text{det}}(\boldsymbol{y}) = \int_{\Omega_{\text{part}}} R(\boldsymbol{y}, \boldsymbol{x}) p_{\text{part}}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}. \tag{3.1}$$

Note that we typically assume that the response function $R(\boldsymbol{y}, \boldsymbol{x})$, which is nothing but the conditional probability $p_{\text{det}}(\boldsymbol{y} \,|\, \boldsymbol{x})$, is independent of the model underlying $p_{\text{part}}(\boldsymbol{x})$. Unfolding now refers to finding a parton-level distribution $p_{\text{part}}(\boldsymbol{x})$ that results in a given detector-level density $p_{\text{det}}(\boldsymbol{y})$ when folded.

This is an ill-posed problem in the sense of Hadamard, i.e. there does not exist a unique solution $p_{\text{part}}(\boldsymbol{x})$ due to possible lack of information on detector-level [84]. In practice, however, we use unfolding procedures that give us a single result. This ignorance results in a dependence on the physics model,

which we can see easily by considering a response function that leads to a complete loss of information at detector-level. In this case, standard unfolding methods will always reconstruct the parton-level distribution that was used for fitting the unfolding model. This theoretical problem is inherent to all standard unfolding methods. In the following, we will argue this explicitly for each technique.

If we want to do 'full' unfolding to address this issue, we can e.g. enlarge our solution space and unfold to equivalence classes of parton-level densities. Suppose we have a continuous space of model hypotheses given by parameters α we want to test. In this case, we can condition our unfolding model on α and fit it to matching distributions $p_{\text{det}}^{(\alpha)}(\boldsymbol{y})$ and $p_{\text{part}}^{(\alpha)}(\boldsymbol{x})$ corresponding to each model. If we now unfold some detector distribution $p_{\text{det}}(\boldsymbol{y})$, we do this for each value of α. This gives us a family of parton-level distributions from which we can quantify the uncertainty that arises through the ill-posed nature of unfolding. To reiterate, even though it seems like the unfolding model is now dependent on α, it is not. This is because we do not technically have a well-defined unfolding model for a fixed value of α; only the complete conditioned model is well-defined. What is dependent on α though are the solutions $p_{\text{part}}^{(\alpha)}(\boldsymbol{x})$ we allow for any fixed α. In doing this, of course, we still do not account for parton-level densities that do not correspond to any of the considered models.

This approach can be impractical and defeats the purpose of unfolding when you want to forget about your detector-level data and just store detector-independent unfolded data. Unfolding for a fixed model still works if one assumes that the response function is approximately invertible or if one is just interested in rejecting the Standard Model (SM) hypothesis. However, if we cannot ignore missing information on detector-level and want to place bounds on some alternative BSM hypothesis, we have to do 'full' unfolding.

There is an additional problem. While any unfolding method relies on a good detector simulation [40], we can introduce additional errors through approximative simulations of the hard scattering process, which are used to

fit the unfolding model. With discrepancies in the simulation of parton-level data, we would still learn the proper response matrix, but only if R is exactly invertible. However, if this is not the case, the response matrix we learn might be biased towards our wrong MC-simulation and introduce errors when applied to real data. This problem is inherently equivalent to model dependence, so all the following statements we will make about it also apply to systematic simulation errors.

3.2. Classical Unfolding

The easiest way to invert the folding process from Eq. (3.1) is to limit ourselves to a few interesting observable distributions and discretize them by introducing a binning [40, 83]. This requires in particular, that we reconstruct detector-level proxies for interesting parton-level observables to unfold. We assume that Eq. (3.1) is still approximately valid, with x, y now only consisting of a few observables/proxies instead of all phase-space dimensions.

Suppose we describe $p_{\text{part}}(\boldsymbol{x})$ and $p_{\text{det}}(\boldsymbol{y})$ as histograms with n and m bins respectively, in which $\boldsymbol{h}_{\text{det}}$ and $\boldsymbol{h}_{\text{part}}$ are the bin heights, then Eq. (3.1) becomes a linear equation

$$\boldsymbol{h}_{\text{det}} = R\boldsymbol{h}_{\text{part}}. \tag{3.2}$$

This replaces $R(\boldsymbol{y}, \boldsymbol{x})$ with the $m \times n$ *response matrix* R. Other discretization methods approximate both probability densities with slightly more elaborate functions in each bin, or with fourier decomposition [83]. We will not discuss these further.

To unfold, we can now invert the response matrix R, by either taking the inverse directly in the case $m = n$ or by taking the Moore-Penrose generalized inverse [83]. The latter gives us the matrix $A^\dagger$ that leads to a solution $\boldsymbol{h}_{\text{part}} = A^\dagger \boldsymbol{h}_{\text{det}}$ which minimizes the squared difference

$$(A\boldsymbol{h}_{\text{part}} - \boldsymbol{h}_{\text{det}}) V_{\boldsymbol{h}_{\text{det}}}^{-1} (A\boldsymbol{h}_{\text{part}} - \boldsymbol{h}_{\text{det}}), \tag{3.3}$$

where the inclusion of the covariance matrix $V_{\boldsymbol{h}_{\text{det}}}$ of $\boldsymbol{h}_{\text{det}}$ takes detector-level uncertainties into account. This method is usually preferred, since R is not invertible for $m = n$ in most cases [83]. Here we get a model dependence through $\boldsymbol{h}_{\text{part}}$, which we use to minimize Eq. (3.3).

This naive approach has the following issues. These are shared with iterative Bayesian unfolding, which will be introduced below. First, discretization is required, such that we have to decide beforehand on the number of bins and a set of observables that are of interest to us. Especially when talking about using unfolding to store data independent of detector specifics, this is a significant downside. Second, limiting our phase-space to only a few observables might lead to erroneous and biased results [40]. Third, classical unfolding only works properly when $\boldsymbol{h}_{\text{det}}$ approaches a continuous distribution, i.e. only for a sufficient number of event samples. And fourth, due to missing energy, jet combinatorics or initial state radiation (ISR), the phase-space on detector-level can differ, meaning that parton-level observables are not easily accessible. If we still want to apply a discretized unfolding method, we have to manually reconstruct interesting observables before the unfolding. It should be clear that classical methods give us quite some room for improvement.

3.2.1. Iterative Bayesian Unfolding

One widely used alternative to the matrix inversion approach above is iterative Bayesian unfolding (IBU) [85]. The idea is to use Bayes' theorem in writing

$$\begin{aligned} p_{\text{part}}(\boldsymbol{x}) &= \int_{\Omega_{\text{det}}} p_{\text{part}}(\boldsymbol{x}\,|\,\boldsymbol{y}) p_{\text{det}}(\boldsymbol{y})\,\mathrm{d}\boldsymbol{y} \\ &= \int_{\Omega_{\text{det}}} \frac{p_{\text{det}}(\boldsymbol{y}\,|\,\boldsymbol{x}) p_{\text{part}}(\boldsymbol{x})}{\int_{\Omega_{\text{part}}} p_{\text{det}}(\boldsymbol{y}\,|\,\boldsymbol{x}') p_{\text{part}}(\boldsymbol{x}')\,\mathrm{d}\boldsymbol{x}'}\, p_{\text{det}}(\boldsymbol{y})\,\mathrm{d}\boldsymbol{y}, \end{aligned} \tag{3.4}$$

where $p_{\text{det}}(\boldsymbol{y}\,|\,\boldsymbol{x}) = R(\boldsymbol{x}, \boldsymbol{y})$. We can solve this equation by postulating a prior $p^{(0)}_{\text{part}}(\boldsymbol{x})$ and computing iterative improvements $p^{(n)}_{\text{part}}(\boldsymbol{x})$ to this prior by

repeated application of Eq. (3.4), i.e.

$$p_{\text{part}}^{(n)}(\boldsymbol{x}) = \int_{\Omega_{\text{det}}} \frac{R(\boldsymbol{x},\boldsymbol{y})p_{\text{part}}^{(n-1)}(\boldsymbol{x})}{\int_{\Omega_{\text{part}}} R(\boldsymbol{x}',\boldsymbol{y})p_{\text{part}}^{(n-1)}(\boldsymbol{x}')\,\mathrm{d}\boldsymbol{x}'}\, p_{\text{det}}(\boldsymbol{y})\,\mathrm{d}\boldsymbol{x}, \tag{3.5}$$

Typically one discretizes, i.e. introduces binned distributions $h_{\text{part},i}$ and $h_{\text{det},i}$ of single observables, to apply this unfolding technique [81]:

$$h_{\text{part},j}^{(n)} = \sum_i \frac{R_{ij}h_{\text{part},j}^{(n-1)}}{\sum_k R_{ik}h_{\text{part},k}^{(n-1)}} h_{\text{det},i}. \tag{3.6}$$

Here R_{ij} is the discretized response matrix.

In IBU, the model dependence enters through the prior. Although the prior changes every iteration, it should be clear that some dependence will typically remain, even in the limit $n \to \infty$. For instance, if we have a response function with maximal information loss, the update rule of IBU would always recover the initial prior.

3.3. Omnifold

One notable unfolding model that avoids the discretization problems discussed above is *Omnifold* [40, 86]. This method uses classification neural networks to apply IBU to unbinned distributions. Suppose we have matching samples from simulated parton-level and detector-level distributions $p_{\text{part}}^{\text{sim}}(\boldsymbol{x})$ and $p_{\text{det}}^{\text{sim}}(\boldsymbol{y})$ and try to determine the true underlying parton-level distribution $p_{\text{part}}(\boldsymbol{x})$ from measured detector-level data $p_{\text{det}}(\boldsymbol{y})$. For this, let us write the true parton-level distribution as a reweighted version of the simulation:

$$p_{\text{part}}(\boldsymbol{x}) = \nu(\boldsymbol{x})p_{\text{part}}^{\text{sim}}(\boldsymbol{x}). \tag{3.7}$$

What happens now, in terms of these weights, if we apply IBU?. First, we define $\nu_0(\boldsymbol{x}) = 1$ and set our prior to $\nu_0(\boldsymbol{x}) p^{\text{sim}}_{\text{part}}(\boldsymbol{x})$, i.e. to the simulated parton-level distribution. For the n-th iteration, we start by calculating the evidence. Here, we apply our response function to the $(n-1)$-th parton level distribution, which yields the simulated detector-level distribution with appropriately chosen weights $\nu^{\text{push}}_{n-1}(\boldsymbol{y})$:

$$\nu^{\text{push}}_{n-1}(\boldsymbol{y}) p^{\text{sim}}_{\text{det}}(\boldsymbol{y}) = \int_{\Omega_{\text{part}}} R(\boldsymbol{x}, \boldsymbol{y}) \nu_{n-1}(\boldsymbol{x}) p^{\text{sim}}_{\text{part}}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}. \tag{3.8}$$

Then we use Bayes' theorem to compute an update to the parton level distribution. In our case, this just gives us new weights ν_n, defined through (cf. Eq. (3.5))

$$\nu_n(\boldsymbol{x}) p^{\text{sim}}_{\text{part}}(\boldsymbol{x}) = \int \frac{R(\boldsymbol{x}, \boldsymbol{y}) \nu_{n-1}(\boldsymbol{x}) p^{\text{sim}}_{\text{part}}(\boldsymbol{x})}{\nu^{\text{push}}_{n-1}(\boldsymbol{y}) p^{\text{sim}}_{\text{det}}(\boldsymbol{y})} p_{\text{det}}(\boldsymbol{y}) \, \mathrm{d}\boldsymbol{y}. \tag{3.9}$$

The central idea of Omnifold is that, for each iteration, we can compute the weights $\nu^{\text{push}}_{n-1}(\boldsymbol{y})$ and $\nu_n(\boldsymbol{x})$ through classifier re-weighting. In particular, a classification network can approximate the ratio $p(\boldsymbol{x})/q(\boldsymbol{x})$, when trained to distinguish two (unbinned) probability distributions $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$.

The Omnifold algorithm works as follows. First, we take the weights $\nu_{n-1}(\boldsymbol{x})$ of the simulated parton-level samples and 'push' it through our detector simulation by assigning each parton-level weight to the matching detector-level sample. This is illustrated as the first step in Fig. 3.1. Unfortunately, the pushed weights $\nu^{\text{push}}_{n-1}(\boldsymbol{y})$ are not a well-defined function, since multiple parton-level events might correspond to—and therefore assign multiple weights—to the same detector-level event. In practice, however, we can ignore this, because the next step does not need a proper function.

Second, we re-weight $\nu^{\text{push}}_{n-1}(\boldsymbol{y}) p^{\text{sim}}_{\text{det}}(\boldsymbol{y})$ to $p(\boldsymbol{y})$, giving us new weights $\omega_n(\boldsymbol{y})$ for $p^{\text{sim}}_{\text{det}}(\boldsymbol{y})$ (cf. Fig. 3.1, step two). The purpose of this step is twofold. To start, we convert the detector-level weights we obtained in the last step to a

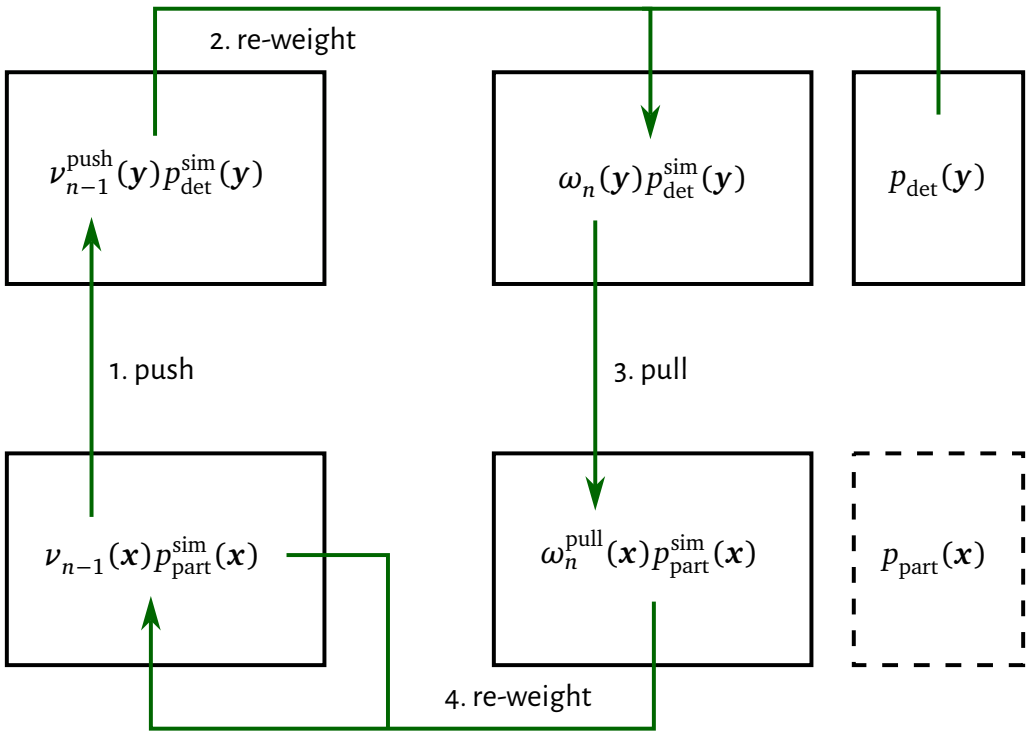

Figure 3.1.: The Omnifold algorithm visualized.

proper weighting function $\omega_n(y)$ on detector-level. In the classifier training we can just use the weight samples as is, without any definition problems. Then, although $\omega_n(y)p_{\text{det}}^{\text{sim}}(y)$ is just exactly $p(y)$ for a perfect classifier, the $p(y)$ samples do not have matching parton-level samples. By re-weighting, we first and foremost get weights that we can subsequently 'pull down' to parton-level.

Third, we do exactly that, i.e. pull the obtained weights down to parton-level (cf. Fig. 3.1, step three). This sounds simple, but what actually happens when we combine the last two steps is Eq. (3.9). This does nothing but inverting the detector simulation for $p_{\text{det}}(y)$ by computing an inverse with Bayes' theorem. In particular, we compute the inverse that connects $\nu_{n-1}^{\text{push}}(y)p_{\text{det}}^{\text{sim}}(y)$ to $\nu_{n-1}(x)p_{\text{part}}^{\text{sim}}(x)$. This is exactly the one we use to 'pull down' the weights $\omega_n(y)$. As already discussed, $R(x,y)$ is typically not invertible, so any 'inverse'

is prior dependent and gives us only one possible parton-level distribution.

Fourth, since we, again, do not have a proper function by just pulling down the weights, we have to do a final re-weighting (cf. Fig. 3.1, step four). Specifically, we re-weight $\nu_{n-1}(\boldsymbol{x})p^{\text{sim}}_{\text{part}}(\boldsymbol{x})$ to $\omega^{\text{pull}}_n(\boldsymbol{x})p^{\text{sim}}_{\text{part}}(\boldsymbol{x})$, where $\omega^{\text{pull}}_n(\boldsymbol{x})$ is defined similarly to $\nu_n(\boldsymbol{x})$ by

$$\omega^{\text{pull}}_n(\boldsymbol{x})p^{\text{sim}}_{\text{part}}(\boldsymbol{x}) = \int \frac{R(\boldsymbol{x},\boldsymbol{y})\nu_{n-1}(\boldsymbol{x})p^{\text{sim}}_{\text{part}}(\boldsymbol{x})}{\nu^{\text{push}}_{n-1}(\boldsymbol{y})p^{\text{sim}}_{\text{det}}(\boldsymbol{y})}\omega_n(\boldsymbol{y})p^{\text{sim}}_{\text{det}}(\boldsymbol{y})\,\mathrm{d}\boldsymbol{y}. \tag{3.10}$$

This re-weighting just gives us back $\omega^{\text{pull}}_n(\boldsymbol{x})$. But remember that we do not have this function—as a function—when we re-weight and that we can do the re-weighting without it.

Omnifold addresses the discretization problems of classical unfolding: We do not get the disadvantages of binning and we can unfold all phase-space dimensions simultaneously without having to focus on a few specific observables. This also means that the reconstruction of unaccessible parton-level observables is built into Omnifold and does not need to be done manually. Since Omnifold is based on IBU, it naturally retains the model dependence of this method through the choice of prior.

3.4. Unfolding with Conditional Normalizing Flows

Another unfolding method without discretization can be realized with normalizing flows [37]. Here, instead of trying to determine $p_{\text{part}}(\boldsymbol{x})$ from $p_{\text{det}}(\boldsymbol{y})$, we go one step further and determine $p_{\text{part}}(\boldsymbol{x}\,|\,\boldsymbol{y})$. This gives us an unfolded distribution for each detector event $\boldsymbol{y}$, making unfolding feasible for a low number of samples [37]. Suppose, using the notation of the previous section, we have matching parton- and detector-level samples from respective distributions $p^{\text{sim}}_{\text{part}}(\boldsymbol{x})$ and $p^{\text{sim}}_{\text{det}}(\boldsymbol{y})$. In this case, we can directly use a conditional flow to normalize $p^{\text{sim}}_{\text{part}}(\boldsymbol{x}\,|\,\boldsymbol{y})$ to a base distribution $q(\boldsymbol{z})$. This in turn allows

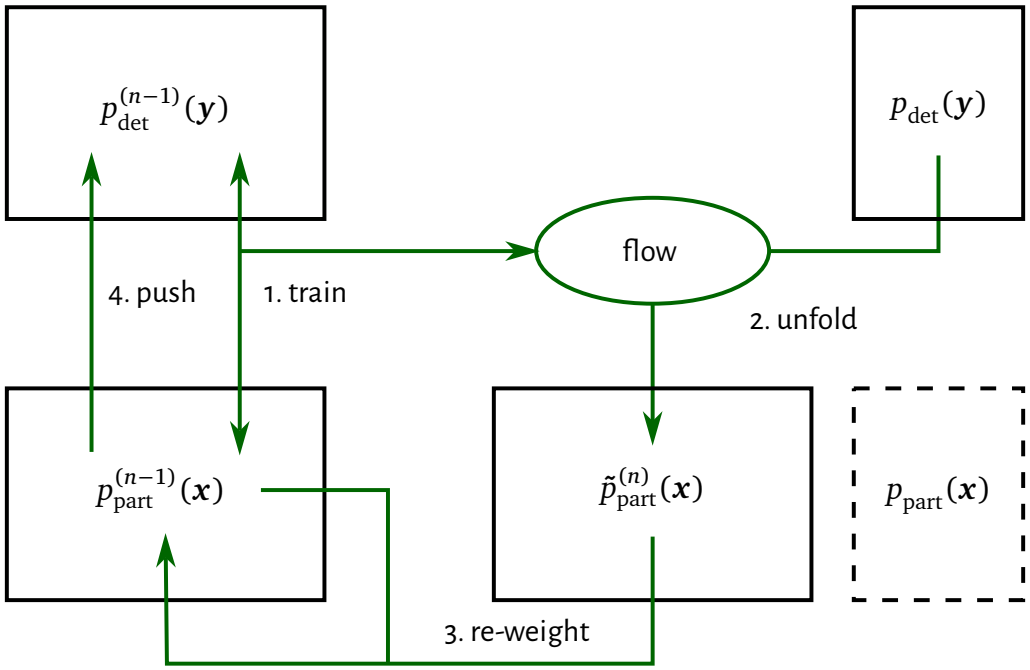

Figure 3.2.: Iterative normalizing flow unfolding visualized.

us to obtain $p_{\text{part}}^{\text{sim}}(x|y)$ by un-normalizing the base distribution under a fixed and previously unseen condition y. In other words, we give the network information about $p_{\text{part}}^{\text{sim}}(x)$ and the transfer function $R(y,x) = p_{\text{det}}(y|x)$ to obtain $p_{\text{part}}^{\text{sim}}(x|y)$. This is a direct application of Bayes theorem [37]

$$p_{\text{part}}^{\text{sim}}(x|y) = \frac{R(y,x)p_{\text{part}}^{\text{sim}}(x)}{p_{\text{det}}^{\text{sim}}(y)}. \tag{3.11}$$

Note that we recover (3.4) when we multiply by $p_{\text{det}}(y)$ and integrate. Hence, normalizing flow unfolding is essentially equivalent to the first step of IBU. This approach again suffers from model dependence, however, we also get an additional prior dependence since we omit further steps of IBU.

We can avoid this by using the iterative approach introduced in Ref. [38],

which is similar to Omnifold. For the n-th step, suppose we have previously trained a flow on distributions $p_{\text{part}}^{(n-1)}(\boldsymbol{x})$ and $p_{\text{det}}^{(n-1)}(\boldsymbol{y})$, giving us a conditional distribution $p_{\text{part}}^{(n)}(\boldsymbol{x}|\boldsymbol{y})$. The distributions we train on shall be the simulated parton- and detector-level distributions for the first step, which is visualized in Fig. 3.2. We can use this flow to unfold a measured distribution $p_{\text{det}}(\boldsymbol{y})$ (cf. Fig. 3.2, step two), obtaining

$$\tilde{p}_{\text{part}}^{(n)}(\boldsymbol{x}) = \int_{\Omega_{\text{det}}} p_{\text{part}}^{(n)}(\boldsymbol{x}|\boldsymbol{y}) p_{\text{det}}(\boldsymbol{y})\, \mathrm{d}\boldsymbol{y}. \tag{3.12}$$

We then reweight $p_{\text{part}}^{(n-1)}(\boldsymbol{x})$ to $\tilde{p}_{\text{part}}^{(n)}(\boldsymbol{x})$ and populate the weights to the matching detector-level samples (cf. Fig. 3.2, step three and four). This gives us two new distributions $p_{\text{part}}^{(n)}(\boldsymbol{x})$ and $p_{\text{det}}^{(n)}(\boldsymbol{y})$ which we can use for the next iteration. Note that this is exactly IBU; in every step, we use Bayes' theorem to obtain a new prior and new evidence from the posterior.

This approach shares the advantages of Omnifold: it is also unbinned, allows us to unfold all phase-space dimensions instead of just a few observables and does not require manual reconstruction of parton-level observables. On top of this, the normalizing flow approach allows us to unfold single events and does not require a high number of event samples for the unfolding procedure to be valid. Without the iterative approach from Ref. [38] however, we always get additional prior depedence compared to IBU and, equivalently, Omnifold.

4. CP-Violation in $t\bar{t}h$-Production

The violation of the combined charge and parity (CP) symmetry is a well-known condition to explain the matter-antimatter symmetry in our universe, first formulated by Sakharov [87]. Although there are terms in the Standard Model Lagrangian that lead to CP-violation, namely the complex phases of the CKM and PMNS matrices, these do not suffice to explain the observed ratio of matter to antimatter [88–90]. This naturally makes additional CP-violating interactions a clear target for experimental searches for physics beyond the SM. Such effects could potentially be realized by an extension of the Standard Model Higgs interactions, as i.e. the two-Higgs-doublet model [91].

Usually, we want to describe modifications of the SM Lagrangian without making too much assumptions about the new physics model. The standard approach to model-agnostically describe BSM physics are EFTs [92, 93]. These theories can model the effects of BSM particles at a given energy scale E without having to describe any details at the usually much larger scale Λ, where these new particles live.[1] The fundamental assumption here is that the scale E, where we use the EFT, is well separated from Λ. Under this condition, EFTs work by expanding a complex, potentially unknown, underlying Lagrangian as a perturbative series in E/Λ. The higher-dimensional operators in this expansion then must be consistent with the non-accidental symmetries of this Lagrangian. The operator couplings become new parameters of the theory that depend on the underlying BSM physics and can be fitted to data. What makes EFTs so useful is that they are still valid quantum field theorys (QFTs)

[1] This 'bottom-up' approach is of course not the only way to define an EFT, but it is the only one relevant here.

that are renormalizable order by order in the operator expansion.

However, the Higgs-fermion Yukawa couplings could include CP-violating effects already at tree level, through dimension four operators [94]. These effects would naturally manifest themselves around the electroweak symmetry breaking (EWSB) scale, the same scale at which we test the theory. Hence, we cannot describe the Higgs-fermion couplings with an EFT, since the fundamental assumption behind every effective field theory—the separation of scales—would no longer be valid. Nonetheless, we can describe other interactions, arising through dimension six operators, with an EFT, for instance the Higgs couplings to vector bosons [94–96]. However, dimension six operators become suppressed by the new physics scale and are thus harder to test.

Based on these considerations we will describe possible CP-violation in the Higgs-sector with the *Higgs characterization model* [94], a mixture of direct SM Lagrangian modifications and an EFT. Here we consider a generalization of the Higgs at the EWSB scale with various possibilities for its spin and parity. Furthermore, we assume that any additional new particles are located at a much larger scale Λ and describe these with an effective field theory. This approach is largely model-independent, as long as we look at a model that only modifies the EWSB scale within the given possibilities.

As already mentioned, the Higgs-fermion Yukawa coupling can contain CP-violating effects already at tree level, while specifically the Higgs-top interaction has the largest coupling strength, making it the most direct pathway to test CP-violation [97]. In the spin-zero case of the Higgs characterization model, we obtain the following expression for this coupling [94]

$$\mathscr{L} \supset -\frac{m_t}{v}\kappa_t \bar{t}(\cos(\alpha) + i\gamma_5 \sin(\alpha))th. \tag{4.1}$$

Here κ_t modifies the coupling strength while α, which we call *CP-phase*, describes the mixing between the CP-even and CP-odd interaction terms. The SM, only including the CP-even term, is given by $\kappa_t = 1$ and $\alpha = 0$, while the value $\alpha = \pi/2$ leads to a pure CP-odd interaction. We can relate

the Lagrangian in Eq. (4.1) with its CP-transformed version by replacing $\alpha \to -\alpha$. Note that we set $\kappa_{Ht\bar{t}} = \kappa_{At\bar{t}} = \kappa_t$ to obtain the above expression from Ref. [94].[2]

Since $h \to t\bar{t}$ is not allowed, we cannot test the Higgs-top interaction through a decay, making associated production the most direct probe of CP-violation in the Higgs sector [97], i.e.

$$pp \to t\bar{t}h \tag{4.2}$$

and the conjugated process. Among the Higgs decay modes, $h \to \gamma\gamma$ is the most promising. Despite limited statistics, this mode gives us the best sensitivity due to the di-photon mass being an excellent discrimination measure for backgrounds [41].

In general, there are two ways to test the CP-properties of a theory. One approach is to find a theory parameter that is associated with a CP-violating term in the Lagrangian and fit this parameter to experimental data. In our case, we could do this with α. Since the theory would link the parameter to CP violation, this test would constitute an indirect check of CP-violation. Another way is to test this symmetry directly, either by finding a process that is forbidden by CP-invariance or by comparing a process with its CP-conjugated version [98]. In the latter case, we can construct dedicated observables for which there are separate predictions with and without CP-violation. We will discuss general considerations about constructing observables like this below.

4.1. Constructing Direct CP-Observables

Two discrete space-time symmetries are contained in the Lorentz group and as such could be realized in nature: parity (P) and time-reversal (T). Among the

[2] This is without loss of generality as noted in [94]. Including both parameters only makes sense for practical reasons

symmetries concerning quantum numbers, charge conjugation (C) is deeply connected to P and T through the CPT-theorem [99]. These three symmetries are hence of particular importance in studying particle interactions. In the language of QFT, C and P act as linear operators, while the operator corresponding to T is anti-linear. If we let $|\psi_{p,s,n}\rangle$ be a free single-particle momentum eigenstate with 4-momentum p, spin s and quantum numbers (particle species) n, then these operators act as

$$\begin{aligned} C\,|\psi_{p,s,n}\rangle &= \eta_n^{(C)}\,|\psi_{p,s,\bar{n}}\rangle \\ P\,|\psi_{p,s,n}\rangle &= \eta_n^{(P)}\,|\psi_{p^*,s,n}\rangle \\ T\,|\psi_{p,s,n}\rangle &= \eta_n^{(T)}(-1)^{j+s}\,\langle\psi_{p^*,-s,n}|\,. \end{aligned} \tag{4.3}$$

Here p^* is defined as $p^* = (E,\vec{p})^* = (E,-\vec{p})$, j is the orbital quantum number of $|\psi_{p,s,n}\rangle$ and $\bar{n}$ are the quantum numbers of the anti-particle corresponding to n. The intrinsic phases $\eta_n^{(C)}$, $\eta_n^{(P)}$ and $\eta_n^{(T)}$ are species dependent and in general not directly measurable [100]. For parity one usually fixes the intrinsic phase for the proton, neutron and electron to be $+1$, which determines the intrinsic phases of other particles. The intrinsic phase for C is directly measurable for neutral particles, otherwise, we can choose it arbitrarily. Regarding T, anti-linearity implies that the phase $\eta_n^{(T)}$ is entirely unphysical and can be simply set to one [100].

The anti-linearity of time reversal has further consequences. A transformation under T reverses in- and out-states, which makes it effectively impossible to test T-symmetry directly [101]. In particular, one would have to prepare an initial state equivalent to the superposed final state of some respective process. It will help us to define a 'naive' time-reversal operator $\hat{T}$, which acts in the same way as T, but does not exchange in- and out-states, i.e.

$$\hat{T}\,|\psi_{p,s,n}\rangle = \eta_n^{(T)}(-1)^{j+s}\,|\psi_{p^*,-s,n}\rangle\,. \tag{4.4}$$

Note that the actions of C, P and T given in Eq. (4.3) differ for massless

particles [100]. However in the following, we will only need the transformation behavior of p, s and n, which does not change in this case.

Imagine we want to test CP-violation of a theory with Lagrangian $\mathcal{L}$, by comparing a specific scattering process with its CP-conjugated version. As stated above, we can construct observables with specific predictions with and without CP-violation to do this. Formally, following Ref. [102], consider a general symmetry transformation U, which shall be an arbitrary combination of C, P and $\hat{T}$. Any Observable $O(i,f)$, generally depending on the initial state i and final state f of the respective process, can now be decomposed into two parts that are respectively even and odd under process conjugation by U:

$$\begin{aligned} \mathcal{O}_{\text{even}}(i,f) &= \frac{1}{2}\Big(\mathcal{O}(i,f) + \mathcal{O}(i^U, f^U)\Big) \\ \mathcal{O}_{\text{odd}}(i,f) &= \frac{1}{2}\Big(\mathcal{O}(i,f) - \mathcal{O}(i^U, f^U)\Big), \end{aligned} \tag{4.5}$$

where i^U and f^U denote the U-conjugated initial and final states. Therefore, we can consider purely U-even and U-odd observables without loss of generality.

U-odd observables are particularly useful because their expectation value vanishes—i.e. has a clear prediction—under the assumption of a CP-invariant theory. To see this, consider an initial state that is statistically invariant under U, i.e. where the distribution $p(i)$ of initial states is U-symmetric. We can write the expectation value of a U-odd observable $\mathcal{O}$ with the matrix element $\mathcal{M}(i,f,\mathcal{L})$ of the process as

$$\langle \mathcal{O} \rangle_{\mathcal{L}} = \int \mathrm{d}\Pi_i \int \mathrm{d}\Pi_f \, |\mathcal{M}(i,f,\mathcal{L})|^2 \, \mathcal{O}(i,f) \, p(i). \tag{4.6}$$

If we rewrite the phase space integration variables as their U-conjugated versions and use the invariance of the matrix element and $p(i)$ under U, we obtain

$$\langle \mathcal{O} \rangle_{\mathcal{L}} = \int \mathrm{d}\Pi_{i^U} \int \mathrm{d}\Pi_{f^U} \, |\mathcal{M}(i,f,\mathcal{L})|^2 \, \mathcal{O}(i^U, f^U) \, p(i). \tag{4.7}$$

Under the additional assumption that the initial and final phase space is U-

symmetric we can use that $\mathcal{O}$ is U-odd to recover the original expectation value with a flipped sign; so $\langle\mathcal{O}\rangle_{\mathcal{L}}$ indeed vanishes.

This motivates us to define *genuine U-odd* observables $\mathcal{O}$ of a theory with Lagrangian $\mathcal{L}$ as satisfiying

$$\langle O\rangle_{\mathcal{L}} = 0 \quad \text{if} \quad U\mathcal{L}U^{-1} = \mathcal{L}. \tag{4.8}$$

Note that this definition also implicitly includes observables that for example compare the probabilities between two U-conjugated processes, an expression that is often used to test for CP-violation [98, 101, 102]. Explicitly, we could do this by dividing our phase space Ω into two disjoint and U-conjugate parts Ω_1 and Ω_2 as well as defining $\mathcal{O}$ to be $+1$ on Ω_1 and -1 on Ω_2. In this sense the definition of genuine U-odd observables from Ref. [102] is quite general.

In the case of CP-violation, we can classify CP-odd observables into two categories according to their transformation behaviour under $\hat{T}$. To see why this classification is sensible, consider the optical theorem

$$\begin{aligned}&\mathcal{M}(i,f) - \mathcal{M}^*(f,i)\\ &\quad = i\sum_X \int \mathrm{d}\Pi_X\,(2\pi)^4\delta(p_i - p_X)\,\mathcal{M}(i,X)\mathcal{M}(f,X).\end{aligned} \tag{4.9}$$

Here the sum on the right-hand side goes over a complete set of *on-shell* intermediate states X. Interactions with such real intermediate states are called *final state interactions (FSIs)* or *re-scattering*. In the absence of FSIs we can use Eq. (4.9) and the CPT theorem to show

$$\mathcal{M}^*(i^{\hat{T}}, f^{\hat{T}}) = \mathcal{M}(f^{\hat{T}}, i^{\hat{T}}) = \mathcal{M}(i^{CP}, f^{CP}). \tag{4.10}$$

In particular $|\mathcal{M}(i^{\hat{T}}, f^{\hat{T}})|^2 = |\mathcal{M}(i^{CP}, f^{CP})|^2$. This has two important implications. First, Eq. (4.10) tells us that the expectation of a $\hat{T}$-odd observable always vanishes in a CP-invariant theory as well. In other words, in the absence of FSIs, any $\hat{T}$-odd observable is also *genuine* CP-odd and can be used to test for

CP-violation without having to check its CP-transformation behaviour [102]. Note, however, that in the presence of re-scattering, CP-even and $\hat{T}$-odd observables can also have a non-vanishing expectation value in a CP-violating theory [98, 101, 103]. This means, these observables can 'fake' CP-violation if we do not check for their transformation properties under CP.

Second, we can show that there is a crucial difference between CP-odd $\hat{T}$-even and CP-odd $\hat{T}$-odd observables. For the first class of observables, we can show that an expectation value will always vanish in the absence of FSIs [102, 104], rendering them useless for detecting CP-violation. To see this, we will look at such an observable $\mathcal{O}$ under the assumption that $p(i)$ is CP- and $\hat{T}$-invariant and consider

$$\begin{aligned}\langle O\rangle_{\mathcal{L}} = \frac{1}{2}\int \mathrm{d}\Pi_i \int \mathrm{d}\Pi_f \Big(|\mathcal{M}(i,f,\mathcal{L})|^2 - |\mathcal{M}(i^{CP},f^{CP},\mathcal{L})|^2 \Big) \\ \times \mathcal{O}(i,f)p(i).\end{aligned} \tag{4.11}$$

Naturally, this expectation value vanishes in a CP-invariant theory. In a CP-violating theory we can again use Eq. (4.10), to rewrite the second squared matrix element as $|\mathcal{M}(i^{CP},f^{CP},\mathcal{L})|^2 = |\mathcal{M}(i^{\hat{T}},f^{\hat{T}},\mathcal{L})|^2$. Since $\mathcal{O}$ is $\hat{T}$-even, we can again redefine our integration variables by conjugating them under $\hat{T}$ such that the difference of the two matrix elements vanishes. This proofs that FSIs are required to detect CP-violation with CP-odd and $\hat{T}$-even observables. For CP-odd and $\hat{T}$-odd observables, no similar restriction exists.

To understand all of this a little bit deeper, let us look at the squared matrix element difference in Eq. (4.11) again. These matrix elements generally have the following form [101]

$$\begin{aligned}\mathcal{M}(i,f) &= \sum_i A_i e^{\delta_i+\phi_i} \\ \mathcal{M}(i^{CP},f^{CP}) &= \sum_i A_i e^{\delta_i-\phi_i+\theta}.\end{aligned} \tag{4.12}$$

In these expressions the δ_i's are called *CP-conserving phases* (or strong phases)

since they do not change sign under the conjugation, while the ϕ_i's are called *CP-violating phases* (or weak phases). The global phase θ is purely conventional. Our first observation is that we need two or more amplitudes A_i to contribute to the process in question to have any chance at a non-vanishing expectation value. Second, looking at only two interfering amplitudes for simplicity, we obtain the following difference in matrix elements

$$|\mathcal{M}(i,f)|^2 - |\mathcal{M}(i^U,f^U)|^2 = -4A_1A_2\sin(\delta_2-\delta_1)\sin(\phi_2-\phi_1). \quad (4.13)$$

Clearly $\delta_1 \neq \delta_2$ and $\phi_1 \neq \phi_2$ is required to obtain a non-vanishing expectation value $\langle O\rangle_{\mathcal{L}}$.

In particular, we need at least one non-zero *CP*-conserving and *CP*-violating phase [98, 101, 103]. For the latter, we do not have a problem since *CP*-violating phases occur naturally in a *CP*-violating theory (i.e. α in Eq. (4.1)), typically from complex couplings in the Lagrangian [101]. *CP*-conserving phases, however, do not necessarily exist in the process and can come from two sources [101]. The first one is a trace of an even number of γ matrices together with γ_5 occurring in the squared matrix element of the process in question. Such a trace gives a complex contribution to the matrix element and must come from some Lorentz invariant Levi-Civita product like $\varepsilon_{\alpha\beta\gamma\delta}k^\alpha p_1^\beta p_2^\gamma p_3^\delta$. In the rest frame defined by k^μ such a term becomes a triple product $\vec{p}_1\cdot(\vec{p}_2\times\vec{p}_3)$. The second one are FSIs. In particular, if we have a loop interaction $A\to A$ concerning some state A, the optical theorem (cf. Eq. (4.9)) implies that FSIs generate an imaginary part of the amplitude, giving us a *CP*-conserving phase. With these two origins of *CP*-conserving phases in mind, we can see why the expectation value of *CP*-odd and $\hat{T}$-odd observables can be non-vanishing in a *CP*-violating theory—even without re-scattering [98].

Note that Levi-Civita products that appear in the matrix element are themselves *CP*-odd. They have to be, otherwise, they could not give rise to a *CP*-violating phase. Moreover, they are also $\hat{T}$-odd, which makes them natural observable candidates for *CP*-violation detection, albeit not the only ones.

4.2. Direct CP-Observables in $t\bar{t}h$-Production

Let us apply the above discussion to di-leptonic $t\bar{t}h$-production, following Ref. [105]. Using the Higgs-top coupling introduced in Eq. (4.1), we can write down the leading order matrix element of either the (dominant) gluon or the quark-antiquark induced channel as

$$\begin{aligned} |\mathcal{M}(XX \to t\bar{t}h)|^2 = \kappa_t^2 \cos(\alpha)^2 f_1(p_i \cdot p_j) \\ + \kappa_t^2 \sin(\alpha)^2 f_2(p_i \cdot p_j) \\ + \kappa_t^2 \cos(\alpha)\sin(\alpha) \sum_{k=1}^{15} g_k(p_i \cdot p_j) P_k, \end{aligned} \tag{4.14}$$

where XX denotes either gg or $q\bar{q}$. If we let q_1 and q_2 denote the gluon/quark momenta, this expression depends on six four-vectors p_i: The top spin four-vectors s_t, $s_{\bar{t}}$ as well as p_t, $p_{\bar{t}}$, $q = (q_1 - q_2)/2$ and $Q = (q_1 + q_2)/2$. The functions f_1, f_2 and g_k only depend on the scalar products of these six four vectors, while P_k denotes the fifteen expressions of the form $\varepsilon_{\alpha\beta\gamma\delta} p_a^\alpha p_b^\beta p_c^\gamma p_d^\delta =: \varepsilon(p_a, p_b, p_c, p_d)$ that can be constructed from them.

Here we see precisely how the presence of the P_k's is necessary to generate a CP-odd matrix element term (i.e. odd under $\alpha \to -\alpha$). In particular the P_k's are observables that are odd under CP and $\hat{T}$ transformations. Together with the CP-invariant initial state—invariant in the $t\bar{t}$ rest frame that is—and the CP-invariant phase space of $t\bar{t}h$ production, we can conclude that these are also genuine CP-odd. Furthermore, due to the appearance of the P_k-induced CP-odd term in the matrix element, we can also expect that the expectation value of these Levi-Civita products will not vanish in a CP-violating theory.

Let us consider the P_k's closer. Among these fifteen observables, there are only five that we can hope to observe. This is because we cannot measure q, due to insufficient knowledge about the initial state at the LHC. One can further show that, for the two P_k's which contain only a single spin vector, expectation values (asymmetries) vanish [106]. This leaves us with only the

three observables

$$\begin{aligned} P_1 &= \varepsilon(p_t, p_{\bar{t}}, s_t, s_{\bar{t}}) \\ P_2 &= \varepsilon(Q, p_t, s_t, s_{\bar{t}}) \\ P_3 &= \varepsilon(Q, p_{\bar{t}}, s_t, s_{\bar{t}}). \end{aligned} \tag{4.15}$$

The discussion so far ignored the top decays, which contribute to the overall process. These can be included by using the narrow width approximation (i.e. assuming on-shell top quarks) [105]. In this case, we can factorize the full matrix element into contributions from both decays and pure $t\bar{t}h$ production. This allows us to relate the top spins to the decay product momenta [107]

$$\begin{aligned} s_t &= -\frac{p_t}{m_t} + \frac{m_t}{(p_t \cdot p_{\ell^+})} p_{\ell^+} \\ s_{\bar{t}} &= -\frac{p_{\bar{t}}}{m_t} - \frac{m_t}{(p_{\bar{t}} \cdot p_{\ell^-})} p_{\ell^-}. \end{aligned} \tag{4.16}$$

More generally, we can always expect top spin correlations to be sensitive to the kinematics of top decay products. The relation goes according to [108, 109]

$$\frac{1}{\Gamma_t} \frac{\mathrm{d}\Gamma}{\mathrm{d}\xi_X} = \frac{1}{2} \Big(1 + \beta_X P_t \cos(\xi_X) \Big), \tag{4.17}$$

where Γ (Γ_t) is the (total) top decay rate, ξ_X is the angle between top decay product X and the top spin and P_t is the polarization of the top. The factor β_X is a unique number corresponding to a certain decay product and is called *spin analyzing power*. Note that this situation is special to the top quark. Due to its short lifetime, it can pass spin information to its decay products before hadronization or spin decorrelation effects take place [41].

Using Eq. (4.16) we can express (4.15) as:

$$\begin{aligned} P_1 &= \frac{m_t^2}{(p_t \cdot p_{\ell^+})(p_{\bar{t}} \cdot p_{\ell^-})} \varepsilon(p_t, p_{\bar{t}}, p_{\ell^-}, p_{\ell^+}) \\ P_2 &= \frac{m_t^2}{(p_t \cdot p_{\ell^+})(p_{\bar{t}} \cdot p_{\ell^-})} \Big(\varepsilon(p_t, p_{\bar{t}}, p_{\ell^-}, p_{\ell^+}) + \varepsilon(p_h, p_{\bar{t}}, p_{\ell^-}, p_{\ell^+}) \end{aligned}$$

$$+\frac{p_t \cdot p_{\ell^+}}{m_t^2}\varepsilon(p_h, p_{\bar{t}}, p_t, p_{\ell^-})\Bigg)$$

$$P_3 = \frac{m_t^2}{(p_t \cdot p_{\ell^+})(p_{\bar{t}} \cdot p_{\ell^-})}\Big(-\varepsilon(p_t, p_{\bar{t}}, p_{\ell^-}, p_{\ell^+}) + \varepsilon(p_h, p_t, p_{\ell^-}, p_{\ell^+})$$
$$+\frac{p_{\bar{t}} \cdot p_{\ell^-}}{m_t^2}\varepsilon(p_h, p_{\bar{t}}, p_t, p_{\ell^+})\Bigg). \tag{4.18}$$

This gives us five new CP-odd Levi-Civita products; the advantage here being that we can construct them without needing direct access to the top spins. The sensitivity of the two observables containing only one lepton momenta is negligible. Among the remaining three products, the most sensitive is $\varepsilon(p_t, p_{\bar{t}}, p_{\ell^-}, p_{\ell^+})$ [105]. There are more options to construct CP-odd observables for $t\bar{t}h$, for example by combining the observables that we have already discussed, or by defining similar expressions from only final state momenta [105].

Levi-Civita products can be expressed as triple products which are in turn related to azimuthal angle differences. For instance, in the $t\bar{t}$ rest frame with $\vec{p}_t$ parallel to the z-axis, $\varepsilon(p_t, p_{\bar{t}}, p_{\ell^-}, p_{\ell^+})$ becomes

$$\begin{aligned}\varepsilon(p_t, p_{\bar{t}}, p_{\ell^-}, p_{\ell^+}) &= m_{t\bar{t}}\, \vec{p}_t \cdot (\vec{p}_{\ell^-} \times \vec{p}_{\ell^+}) \\ &= m_{t\bar{t}}\, |\vec{p}_t|\, p_{\mathrm{T},\ell^-}\, p_{\mathrm{T},\ell^+} \sin(\Delta\phi_{\ell^-\ell^+}),\end{aligned} \tag{4.19}$$

where the prefactor in front of the sine is CP-even and $\hat{T}$-even, making $\Delta\phi_{\ell^-\ell^+}$ CP-odd and $\hat{T}$-odd. Hence, we can also use such azimuthal angle differences to look for CP-violation instead of the full Levi-Civita products.

4.3. CP-Sensitive Observables in $t\bar{t}h$-Production

The CP-odd observables defined above naturally carry sensitivity to the CP-phase. In particular, they must be sensitive to the sign of the CP-phase by definition, since CP-conjugation of the Lagrangian is equivalent to a sign

change in α. As already mentioned we can also probe for CP-violation indirectly, without the use of genuine CP-odd observables, by fitting the value of α to observations.

Specifically, several observables were studied in the literature, which are especially sensitive to the absolute value of α, without being genuine CP-odd observables. A collection of prominent indirectly CP-sensitive observables was analyzed in Ref. [41] for all possible decay channels of the top quarks. These include the masses m_h, m_{th}, $m_{\bar{t}h}$ and $m_{t\bar{t}h}$, the azimuthal angle and pseudorapidity differences $\Delta\phi_{t\bar{t}}$, $\Delta\eta_{t\bar{t}}$ between both tops, the transverse momentum $p_{\mathrm{T},h}$ of the Higgs as well as the $t\bar{t}$ momentum projection product $b_4 = p_{z,t}p_{z,\bar{t}}/p_t p_{\bar{t}}$ [110] and the Collins-Soper (CS) angle θ_{CS} [111, 112]. The CS angle is defined as the polar angle of the top quark in the Collins-Soper frame, the $t\bar{t}$ rest frame in which the z-axis forms equal angles with the two beam directions. Ref. [41] discussed respective observables $\mathcal{O}$ in terms of Fisher information of the distribution $p(\mathcal{O}|\alpha^2)$ at the SM point $\alpha = 0$ (and $\kappa_t = 1$). The squared value of α was considered here, due to these observables being CP-even. Overall the pseudorapidity difference $\Delta\eta_{t\bar{t}}$ and the CS angle were identified as the non-direct CP observables, which are most sensitive to α.

Furthermore, Ref. [41] investigated the sensitivity of genuine CP-odd azimuthal angle differences $\Delta\phi_{AB}^{t\bar{t}}$ in the $t\bar{t}$ rest frame. These are defined between two decay products A and B from the top and the anti-top respectively. Here the distribution $p(\mathcal{O}|\alpha)$ was considered due to their dependence on the sign of α. For all top and antitop decay modes—fully hadronic, semi-leptonic and di-leptonic—it was concluded that respectively $\Delta\phi_{j_{\mathrm{soft}}j_{\mathrm{soft}}}^{t\bar{t}}$, $\Delta\phi_{\ell j_{\mathrm{soft}}}^{t\bar{t}}$ and $\Delta\phi_{\ell\ell}^{t\bar{t}}$ is the observable most sensitive to α. Here j_{soft} is a proxy for the d quark and is defined as the jet with the lowest transverse momentum in the rest frame of the respective top.

Note that many of these observables are quite difficult to construct at reco-level. We need to reconstruct the top momenta for all of them, except for m_h and $p_{T,h}$, while the genuine CP-odd observables additionally require us to use inefficient

proxies for the true jet associated with a quark. The reconstruction of the top momenta is made difficult due to missing momenta from the neutrinos as well as the combinatorial issues associated with jets. Due to these reasons, unfolding—in the sense of at least reversing parton-shower and hadronization effects to some degree—is an essential requirement for the analysis of CP-sensitive observables associated with $t\bar{t}h$ production.

In the following chapter we will develop an unfolding technique for $t\bar{t}h$ which makes use of the entire information on reco-level, allows for the reconstruction of the full parton level phase space and makes the statistical unfolding of single reco-level events possible. We will use the above considerations to guide our construction.

5. A Normalizing Flow Network for Unfolding $t\bar{t}h$-Production

In this chapter, I will present an unfolding model for $t\bar{t}h$ production, which is intended for the reconstruction of CP-sensitive observables on parton-level with high accuracy. The foundation of this model is a conditional normalizing flow, constructed with neural networks.

There are two main challenges with this task. First, while the clear advantage of modern unfolding methods like Omnifold and normalizing flow based techniques is the inclusion of the whole phase-space, in practice our accuracy is limited; how exactly we build our model can have a significant impact on the reconstruction efficiency of specific observables. Therefore, this model takes particular care to achieve optimal unfolding accuracy for CP-sensitive observables relevant to $t\bar{t}h$. At the same time, it was built such that it retains good performance for the remaining phase space. In particular, we will focus on the two direct and non-direct CP-sensitive observables that Ref. [41] identified as most promosing to detect CP-violation. These are the azimuthal angle difference $\Delta\phi_{\ell d}^{t\bar{t}}$ between the charged lepton and the down quark in the $t\bar{t}$ rest frame and the Collins-Soper angle θ_{CS}.

Second, the reconstruction of intermediate particle mass distributions is exceptionally poor when using naive flow architectures [37]. We will see in Section 5.2, after we discussed the training dataset in 5.1, how we can address this issue. The presented solution will also allow us to improve the reconstruction of our two CP-observables, while posing another problem which we

will fix in Section 5.3. Finally, we will discuss the architectural details of our unfolding model in Section 5.4 and its performance in Section 5.5.

5.1. Training Datasets

The goal is to unfold the final state of semi-leptonic $t\bar{t}h$ production, with the Higgs decaying to two photons, i.e.

$$\begin{aligned} pp \to t\bar{t}h &\to (bu\bar{d})(\bar{b}\ell^-\bar{\nu})(\gamma\gamma) \\ pp \to t\bar{t}h &\to (b\ell^+\nu)(\bar{b}\bar{u}d)(\gamma\gamma) \\ pp \to \bar{t}th &\to (\bar{b}\bar{u}d)(b\ell^+\nu)(\gamma\gamma) \\ pp \to \bar{t}th &\to (\bar{b}\ell^-\bar{\nu})(bu\bar{d})(\gamma\gamma). \end{aligned} \tag{5.1}$$

Note that we are interested in $pp \to t\bar{t}h$ and the conjugated process, which are not equivalent in a scenario with CP-violation. However, it does not matter which top decays leptonically and which one decays hadronically. For this reason, without loss of generality, we can assume that t always decays leptonically and $\bar{t}$ always decays hadronically. This also fixes the particles in the final state; in particular, the lepton will always have positive charge.

For this thesis, four datasets were used to investigate the unfolding performance of a normalizing flow model. Three contain events from $t\bar{t}h$ production, including ISR, as described by the Higgs characterization model Lagrangian (cf. (4.1)), with $\kappa_t = 1$ and either $\alpha = -\pi/4$, $\alpha = 0$ (SM) or $\alpha = \pi/4$ [94]. The fourth dataset only describes SM $t\bar{t}h$ production without ISR.

We look at these specific values since they approximately describe the current experimental bounds. In particular, there are two recent experimental $t\bar{t}h$ studies by the ATLAS and CMS collaborations [113, 114], which exclude a CP-phase above 43° and 55° respectively. The performance of our model on these datasets should thus provide an appropriate benchmark.

The parton- and detector-level event samples for each dataset were simu-

lated at leading order and center-of-mass energy $\sqrt{s} = 14$ TeV. Note that we, more specifically, unfold from reco-level, i.e. we have jets on detector-level that are already reconstructed. To simulate $h \to \gamma\gamma$, an effective vertex was used, utilizing a modified Higgs effective field theory (HEFT) based on Ref. [115]. The simulation was done with `MadGraph5_aMC@NLO` [80] using the `NNPDF2.3QED` parton distribution function [116]. For the detector-level distribution parton showering, hadronization and detector effects were applied with `Pythia8` [117], and `Delphes3` [118] using the default ATLAS HL-LHC card. No generation cuts were applied. Event selection was performed according to the following criteria. Each event must contain exactly two photons, two b-tagged jets, one lepton and at least two light jets. Additionally, each event must satisfy the following acceptance cuts

$$\begin{array}{cccc} p_{\mathrm{T},b} > 25 \text{ GeV} & p_{\mathrm{T},j} > 25 \text{ GeV} & p_{\mathrm{T},\ell} > 15 \text{ GeV} & p_{\mathrm{T},\gamma} > 15 \text{ GeV} \\ |\eta_b| < 4 & |\eta_j| < 5 & |\eta_\ell| < 4 & |\eta_\gamma| < 4. \end{array}$$

For ISR datasets, we will restrict the maximal number of jets to be six, while events with less than six jets get zero-padded.

For our purposes, we define our parton-level phase space as containing the momenta

$$\left(p_h, p_b, p_\ell, p_\nu, p_{\bar{b}}, p_u, p_d\right), \tag{5.2}$$

where we leave the Higgs intact. On detector-level we have

$$\left(p_{\gamma_1}, p_{\gamma_2}, p_{b_1}, p_\ell, p_\nu, p_{b_2}, p_{j_1}, \dots, p_{j_n}\right). \tag{5.3}$$

Here the photons, bottom quark and light-jet momenta are p_{T}-sorted respectively. Without ISR, the number of light jets is $n = 2$.[1]

In a realistic setting, we additionally have to consider background processes, e.g. continuum $t\bar{t}\gamma\gamma$-production as the dominant $t\bar{t}h$ background. However, with the unfolded results of a normalizing flow model, we can only compute the conditional distribution $p(\boldsymbol{x}|\boldsymbol{y}, S)$ of possible $t\bar{t}h$ parton-level events $\boldsymbol{x}$ for

[1] Note that we do not consider final state radiation here.

a detector-level signal event $\boldsymbol{y}$. Our goal is now to compute the probability for a specific parton-level event $\boldsymbol{x}$ given an event $\boldsymbol{y}$ that is either signal or background. We can write

$$\begin{aligned} p(\boldsymbol{x}|\boldsymbol{y}) &= \sum_{T\in\{S,B\}} p(\boldsymbol{x}|\boldsymbol{y},T)p(T|\boldsymbol{y}) \\ &= p(\boldsymbol{x}|\boldsymbol{y},S)p(S|\boldsymbol{y}) + p(\boldsymbol{x})(1-p(S|\boldsymbol{y})) \end{aligned} \tag{5.4}$$

Here $p(S|\boldsymbol{y})$ and $p(B|\boldsymbol{y})$ are the probabilities that some event $\boldsymbol{y}$ is signal or background respectively, and can be obtained by e.g. training a classification network on simulated signal and background data. $p(\boldsymbol{x}|\boldsymbol{y},S)$ and $p(\boldsymbol{x})$ denote the probabilities of certain $t\bar{t}h$ parton-level events $\boldsymbol{x}$, when observing an event $\boldsymbol{y}$ that is assumed to be either signal or background respectively. Note that the second probability does not depend on $\boldsymbol{y}$ since background events cannot give us information on x beyond prior knowledge. Again, an unfolding model gives us the first probability, while $p(\boldsymbol{x})$ is constrained through the finite (compact) phase-space of $t\bar{t}h$ at a finite center-of-mass energy and through our model assumptions. In particular

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta} \;= \int |\mathcal{M}(\boldsymbol{x},\boldsymbol{\theta})|^2\, p(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta} \tag{5.5}$$

where $\mathcal{M}(\boldsymbol{x},\boldsymbol{\theta})$ is the $t\bar{t}h$ matrix element and $p(\boldsymbol{\theta})$ describe our prior assumptions about any model parameters $\boldsymbol{\theta}$. For instance, in our case, the background does not contain a Higgs and thus cannot give us information about α or κ_t (cf. (4.1)). This shows that it is reasonable for us to not consider background when constructing our unfolding model.

5.2. Phase-Space Parameterization

If we want to encode the parton-level final state of a scattering process, the direct approach is to use a list of cartesian four-momenta components. Naturally, due to the on-shell conditions, this method creates a parameterization with

redundancies and allows the network to generate unphysical events. But also the obvious solution—simply removing the energies—leads to problems.

Empirically we know that intermediate particle mass distributions are difficult to reconstruct when using a naive parameterization. For this reason, previous studies added the maximum mean discrepancy (MMD) between the unfolded and true mass distribution to the forward KL divergence in the network loss [37, 119].

The MMD is a measure for the discrepancy between two probability densities $p(x)$ and $q(y)$ that is defined for a particular function class $\mathcal{F}$ as

$$\text{MMD}[p, q, \mathcal{F}] = \sup_{f \in \mathcal{F}} \Big(E_{p(x)}[f(x)] - E_{p(y)}[f(y)] \Big) . \tag{5.6}$$

In particular $\text{MMD}[p, q, \mathcal{F}] = 0 \Leftrightarrow p = q$ [120]. By choosing our function class to be a reproducing kernel Hilbert space with kernel $k(x, y)$, we can relate the above definition to the approximation [120]

$$\begin{aligned} &\text{MMD}[\mathcal{F}, \boldsymbol{X}, \boldsymbol{Y}] \\ &\quad = \left[\frac{1}{m^2} \sum_{i,j=1}^{m} k(x_i, x_j) - \frac{2}{mn} \sum_{i,j=1}^{m,n} k(x_i, y_j) + \frac{1}{n^2} \sum_{i,j=1}^{n} k(y_i, y_j) \right]^{1/2} . \end{aligned} \tag{5.7}$$

Typical kernel choices are, for instance, Gaussian or Cauchy functions. While this expression scales quadratically with the sample size (i.e. batch size in practice), it is reasonably efficient to compute and defined in terms of two sets $\boldsymbol{X} = \{x_1, \dots, x_m\}$ and $\boldsymbol{Y} = \{y_1, \dots, y_n\}$ of samples from $p(x)$ and $q(y)$. In situations where both distributions are only accessible through samples, these features can make the MMD a useful tool in machine learning.

Using the MMD for unfolding can solve the problem of mass distributions, however, it has several drawbacks. First, we need phase space samples for the evaluation of the MMD, while the forward KL divergence requires samples from the base distribution of our flow. This means that we have to evaluate our model twice for each training step, effectively doubling the training time.

We typically need more epochs until our network has fully converged as well. Second, we have quite a few hyperparameters associated with an MMD loss, like the relative weight we give the loss term or the particular choice of kernel. In practice, the number of parameters can be even higher, since scheduling of the relative loss weight or a combination of different kernels can be required to obtain a good accuracy with decent performance. We also have to tune these parameters for each intermediate particle. Especially in our case, with two top quarks and W-bosons, this leads to a hyperparameter number which can get quickly out of hand. Third, we would have to undo our preprocessing—if we do more than just rescaling—for each training step. This is conceptually undesirable. For example, we might suffer additional performance losses if our preprocessing is slow by nature. The alternative is to make compromises when preprocessing our data. This can help performance but might lead to worse network accuracy. To conclude, an MMD loss is a slow and inelegant solution that does not scale well. Therefore, let us discuss the alternative.

Empirically unfolding models investigated in this thesis showed a significant improvement when troublesome masses were included in the parameterization of our parton-level phase space (cf. (5.2)); we might try to understand this in the following way. The main feature that distinguishes typical Breit-Wigner mass distributions from other phase space variables is their narrow width of usually only a few GeV. Such a narrow width is hard to learn for a generative network since it has to fine-tune the relation between its features very carefully. However, if the mass itself is a part of the phase-space parameterization, we typically rescale this particular phase space direction during preprocessing and avoid narrow distributions altogether. This, of course, is only a hypothesis. To properly understand this, one would likely have to carefully investigate a toy model.

Writing down a parameterization for $t\bar{t}h$ that features top and W masses directly is not entirely straightforward. We still have to apply the final state on-shell conditions, while not having the full final state momenta available anymore. A natural solution is to store respective parameters from the top, W and

one of their decay products, while simplifying the on-shell conditions through smart selection of the frame. This idea gives us the following parameterization for each top decay:

$$\left(m_t, p_{\mathrm{T},t}, \eta_t, \phi_t, m_W, \eta_W^t, \phi_W^t, \eta_{\ell/u}^W, \phi_{\ell/u}^W\right), \tag{5.8}$$

where we used jet coordinates to parameterize the four-momenta and the superscript indicates the (rest-)frame in which the parameters are defined. Note that this parameterization also separates, and thus simplifies, the kinematics of each decay. It works well in practice and yields excellent accuracy for the reconstruction of the top and W masses, while also improving the accuracy of our unfolding in the remaining phase space. The price we pay for this is worse performance when reconstructing the transverse light and bottom quark momenta; likely, due to them now only being available as a correlation of several other features.

It should be mentioned that this organization of the phase space is generalizable, to a certain degree, to other particle decays. For example, we can—and will—also parameterize the Higgs decay of $t\bar{t}h$ production in this fashion:

$$\left(m_H, p_{\mathrm{T},H}, \eta_H, \phi_H, \eta_{\gamma_1}^H, \phi_{\gamma_1}^H\right). \tag{5.9}$$

This does improve the reconstruction of the Higgs momenta, leading to an improvement in the CP-sensitive invariant mass distributions discussed in Ref. [41].

We can push this idea further by utilizing a parameterization that directly includes our CP-sensitive observables $\Delta\phi_{\ell d}^{t\bar{t}}$ and θ_{CS}. We start by storing $\vec{p}_{t\bar{t}}$, which we then use to boost into the Collin-Soper frame. There, we store the four-momentum of the leptonic top in spherical coordinates—which includes the Collin-Soper angle—and the mass of the hadronic top. Staying in the top frame, but rotating it such that $\boldsymbol{p}_t$ points in positive z direction, we further store the mass of the W as well as the four momenta of the W decay products—also in spherical coordinates. For the latter, we eliminate redundancies using the

on-shell conditions and the W mass, while we store $\Delta\phi^{t\bar{t}}_{\ell d} = \phi^{t\bar{t}}_{d} - \phi^{t\bar{t}}_{\ell}$ instead of $\phi^{t\bar{t}}_{d}$. Overall, this leaves us with

$$\begin{aligned}\Big(&\vec{p}_{t\bar{t}}, m_t, |\vec{p}_t^{\,\mathrm{CS}}|, \theta_t^{\mathrm{CS}}, \phi_t^{\mathrm{CS}}, m_{\bar{t}},\\ &\quad \mathrm{sign}(\Delta\phi^{t\bar{t}}_{\ell\nu})\, m_{W_\ell}, |\vec{p}^{\,t\bar{t}}_{\ell}|, \theta^{t\bar{t}}_{\ell}, \phi^{t\bar{t}}_{\ell}, |\vec{p}^{\,t\bar{t}}_{\nu}|,\\ &\quad \mathrm{sign}(\Delta\phi^{t\bar{t}}_{du})\, m_{W_h}, |\vec{p}^{\,t\bar{t}}_{d}|, \theta^{t\bar{t}}_{d}, \Delta\phi^{t\bar{t}}_{\ell d}, |\vec{p}^{\,t\bar{t}}_{u}|\Big).\end{aligned} \tag{5.10}$$

The sign factors in front of m_{W_ℓ} and m_{W_h} are unfortunately needed to ensure invertibility of (5.10), since we cannot infer the sign of $\Delta\phi^{t\bar{t}}_{\ell\nu}$ and $\Delta\phi^{t\bar{t}}_{du}$ otherwise. If we still want to include the W masses that is.

As one would expect, the sign factors reduce the reconstruction accuracy of the W masses. On the other hand, this parameterization proved to slightly enhance the unfolding accuracy of θ_{CS} and $\Delta\phi^{t\bar{t}}_{\ell d}$, while still leading to excellent precision when reconstructing the top masses and some significant improvements in the rest of the phase space. The exceptions here are the light quark momenta, for which we still get suboptimal accuracy.

Overall, both parameterization approaches greatly outperformed any unfolding model that relied on an MMD loss, while not suffering the drawbacks. The details of the parameterization in Eq. (5.10) are discussed in appendix A.

5.3. Periodic Splines

We argued in the previous section that non-trivial parameterizations are a great way of of constructing an accurate unfolding model for a complex process like $t\bar{t}h$. However, both of these parameterizations heavily rely on either spherical or jet coordinates, both of which feature azimuthal angles. This gives the manifold on which those parameterizations are defined a non-euclidean topology and as we discussed in Section 2.4, there is no established way of defining a finite composition flow on such a manifold. We can ignore this in practice, but this proved to result in very low unfolding accuracy in the

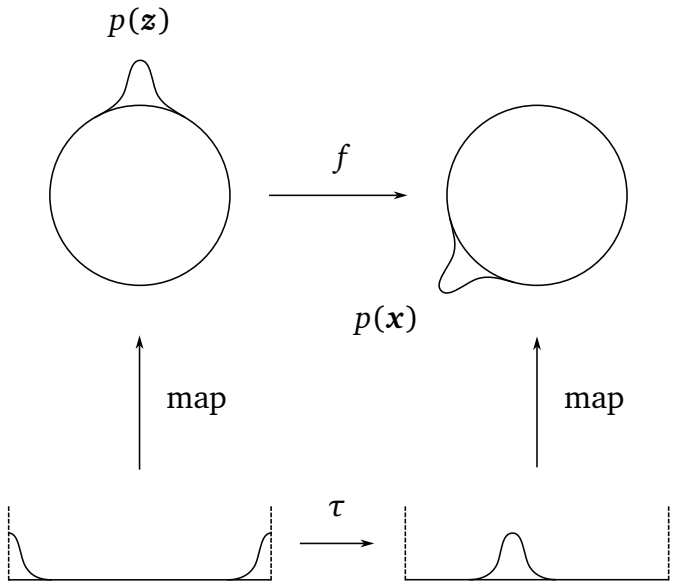

Figure 5.1.: For a classical spline transformer τ a simple shift f on S^1, of some distribution $q(\boldsymbol{z})$ to $p(\boldsymbol{x})$, involves reconstructing a single peak from a bimodal distribution with hard cuts on both modes.

boundary regions of azimuthal angle distributions.

To illustrate why this happens, consider e.g. a normalizing flow which only consists of a single coupling layer. For an azimuthal angle, such a flow effectively transforms distributions between two maps of the circle S^1. Suppose we now transform a Gaussian peak near the boundary region of the first map. As shown in Fig. 5.1, an euclidean flow cannot easily replicate even a simple shift of such a peak, leading to the aforementioned boundary problems.

I propose *periodic spline transformers* to remedy this. In particular, periodic spline transformers can properly handle the two types of topologies—open or closed—that a one-dimensional feature can have. This idea is inspired by the circular spline architecture introduced by Ref. [121]. We will discuss later

why this construction is insufficient for our use-case.

Let us start with classic spline transformers. As discussed in Section 2.1.2, again only looking at one channel z, we construct spline transformers $\tau : [-B, B] \to [-B, B]$ as a combination of K simple piecewise transformations $\tau_k : [z^{(k)}, z^{(k+1)}] \to [x^{(k)}, x^{(k+1)}]$, where $z^{(k)}$ and $x^{(k)}$ are ordered points from $[-B, B]$. This construction, in particular, requires the boundary points to be fixed, i.e. $z^{(0)} = -B$, $z^{(K)} = B$ as well as $x^{(0)} = \tau_1(z^{(0)}) = -B$ and $x^{(K)} = \tau_K(z^{(K)}) = B$. For complicated τ_k, e.g. rational quadratic splines, we also need to specify derivatives $\delta^{(k)}$ at each point $z^{(k)}$.

Now, let us set $B = \pi$ and interpret τ as a transformation between two maps of the circle S^1. We immediately notice that $z^{(0)}$ and $z^{(K)}$ now correspond to the same point on the circle, such that the derivatives at these points should match, i.e. $\delta^{(0)} = \delta^{(K)}$. Moreover, we note that fixed boundary points are no longer a sensible restriction, as this would effectively fix the map in which the flow operates, limiting its expressivity. This is because, as we discussed above, it is easy to construct a diffeomorphism on S^1 that can hardly be replicated by a classical spline transformer (cf. Fig. 5.1).

To 'unfix' the boundary points we can introduce a shift s of τ's target domain. We define

$$\tilde{\tau}_k(z) = \tau_k(z) + s + 2\pi m. \tag{5.11}$$

For the last term we choose m such that $\tilde{\tau}_k(z) \in [-\pi, \pi]$. This ensures matching domains between components of our composition flow. The shift becomes a new parameter that needs to be generated by the conditioner. At the same time, we have one less parameter due to the earlier restriction on the endpoint derivatives, leaving us with the same parameter number as classical spline blocks.[2] This makes it easy to generalize the discussion to a transformer

[2] In Ref. [54] the authors argue for setting $\delta^{(0)} = \delta^{(K)} = 1$ to match the linear nature of τ outside of $[-B, B]$. However, this limits the flexibility of the flow in the boundaries to avoid kinks at the edges. These kinks can only lead to problems for points beyond the boundary, which, ideally, should be avoided anyway and only occur seldom in practice. For this reason, I avoided fixed derivatives.

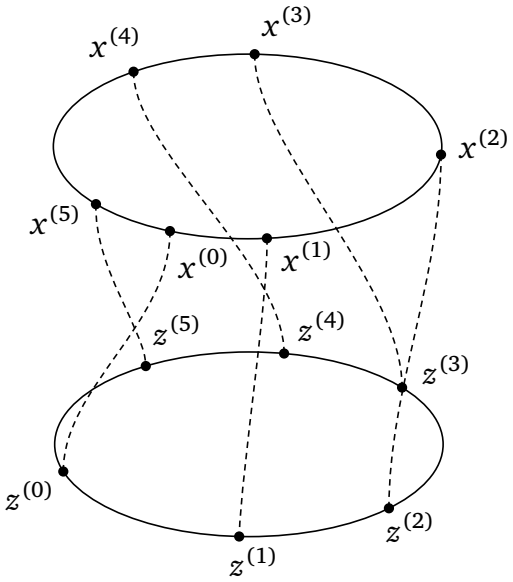

Figure 5.2.: Visualization of a periodic spline transformer. The method presented above is equivalent to choosing $K + 1$ points $x^{(k)}$ and $y^{(k)}$ *freely* on S^1, mapping them to each other while preventing crossings and interpolating the mapping between two adjacent points e.g. with a rational quadratic.

with mixed periodic and non-periodic channels. In this case, we can still use the same conditioner when we just reinterpret one of the derivative parameters as s for each periodic channel. In practical applications of this transformer, one has to keep track of which inputs are periodic—and thus need to be treated with the new method—and which are not. One can do this, for instance, by introducing a boolean vector, indicating for each channel if it is periodic. This one can then be used for masking out either channel type and transforming the remaining inputs in bulk. For coupling blocks, we can simply permute this vector with the inputs, to have access to the channel types at every block.

What do we have accomplished now? By introducing the shift s to the transformer, the network is now able to learn the optimal map to represent the transformed density on, sidestepping the problem presented in Fig. 5.1. We can visualize the new architecture as shown in Fig. 5.2. This intuitive picture should make it clear that requiring $x^{(0)}$ and $y^{(0)}$ to be the same point is an

artificial restriction that should be avoided on S^1. Specifically, it is apparent that we cannot avoid the problem shown in Fig. 5.1 with the circular splines proposed by Ref. [121], since this architecture has no such shift associated to it.

5.4. Architecture

Now that we have gathered all the major building blocks, let us discuss the complete picture of how to build an unfolding model for $t\bar{t}h$. The basis here is a conditional and Bayesian normalizing flow, implemented in `PyTorch` [122] and `FrEIA` [48], utilizing the local re-parameterization trick (cf. Section 2.5) [79]. As discussed in Section 2.5, Bayesian networks can capture statistical uncertainties in the training data. Moreover, the learned distribution of parameters can account for some dependence of the network output on the initialization, making the training results more stable and, most importantly, reproducible.

Next, the model makes use of a composite coupling block architecture with a periodic rational quadratic spline transformer. The conditioner is built as a simple fully connected neural network. Between coupling blocks I use fixed permutation layers—i.e. random permutations were generated once with a specific seed and are then used for every training. Note that this is an important detail: the permutations are an initial condition we cannot account for with a Bayesian architecture, and thus can lead to unstable results. The model was trained for 100 epochs using an ELBO-loss (cf. Eq. (2.40)), as described at the end of Section 2.5, and the Adam optimization algorithm [123]. For the base distribution we use, ignoring constant factors,

$$q(z) \sim \prod_{i \in I_{\neg\phi}} e^{z_i^2/2}. \tag{5.12}$$

Here $I_{\neg\phi}$ denotes the channel indices of all non-periodic channels. For the periodic channels, we use uniform distributions defined on $[-\pi, \pi]$, which

transformer type	periodic spline
spline type	rational quadratic
spline bin count	10
spline non-periodic domains	$[-5.0, 5.0] \rightarrow [-5.0, 5.0]$
coupling block count	16
conditioner layer count	5
conditioner layer dimension	256
VI prior type	gaussian
VI prior $\log(\sigma^2)$	1.0
number of epochs	100
batch size	1024
learning rate	2.0×10^{-4}
optimization algorithm	Adam
signal dataset size	∼1.2M events
training/testing split	80%/20%

Table 5.1.: Hyperparameters as well as architecture and dataset details of the presented unfolding model

only give a constant factor to (5.12), i.e. they only contribute through the Jacobian. More details about the model architecture are shown in table 5.1.

In terms of the dataset, the parameterization presented in (5.10) is used on parton-level, containing both θ_{CS} and $\Delta\phi^{t\bar{t}}_{\ell d}$ as well as the top and W masses. Additionally, the parameterization of the Higgs sector shown in (5.9) is applied on parton *and* detector level. Using the same parameterization on both levels slightly helps the network, since it does not have to learn the parameterization itself. Note that one can only do this for the Higgs sector, where there are no jets or missing energy we have to consider. On top of this, the one-dimensional distribution of each parameter in the parameterization gets normalized to a standard Gaussian (for periodic parameters) or a uniform distribution (for non-periodic parameters), using the standard `scikit-learn` [124] quantile transformation. Note that this transformation gets fitted to the training data.

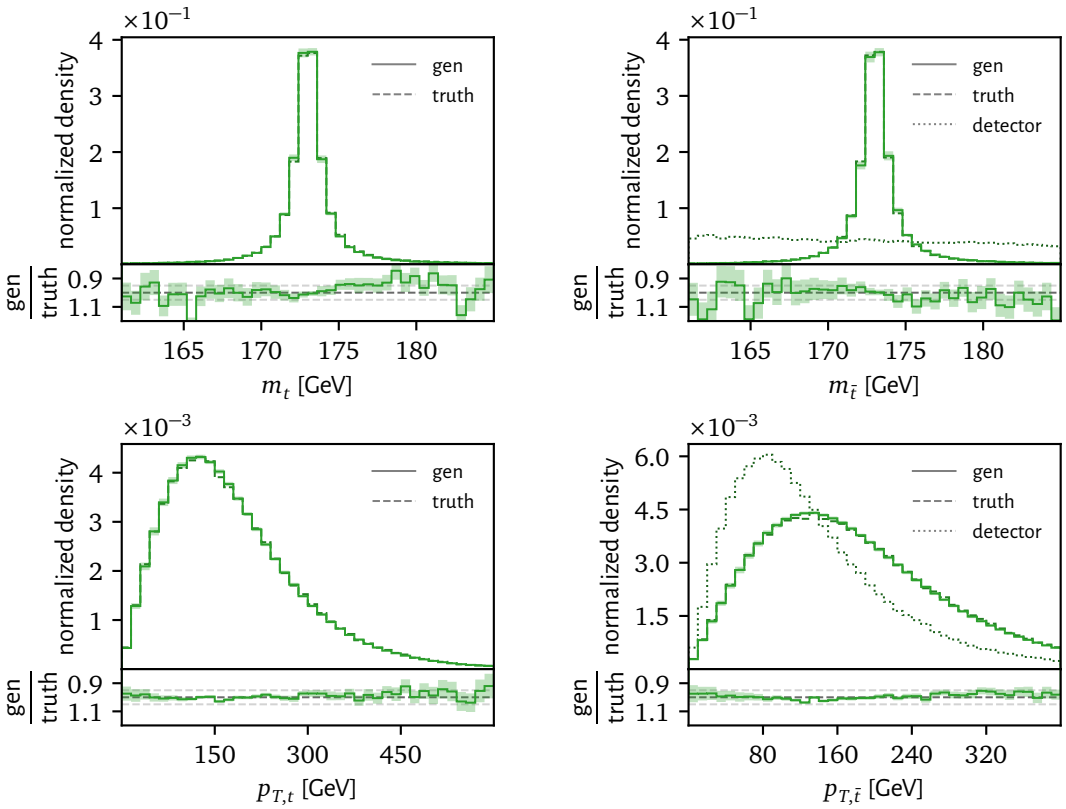

Figure 5.3.: Unfolded mass and transverse momentum distributions of the top quarks, compared to the parton level truth.

5.5. Results

Finally, let us discuss what a network can do, that was constructed according to the previous section. We start by considering the overall unfolding precision for the SM. While the unfolded distribution of the network will be dependent on the choice of physics model, due to the prior dependence in (3.11), the overall accuracy does hardly depend on this choice. For most of the results below, this is also true for the inclusion of ISR effects; exceptions will be specifically mentioned. So in the following, ISR will always be present unless stated otherwise.

To get a general sense for the performance of our model, let us start by

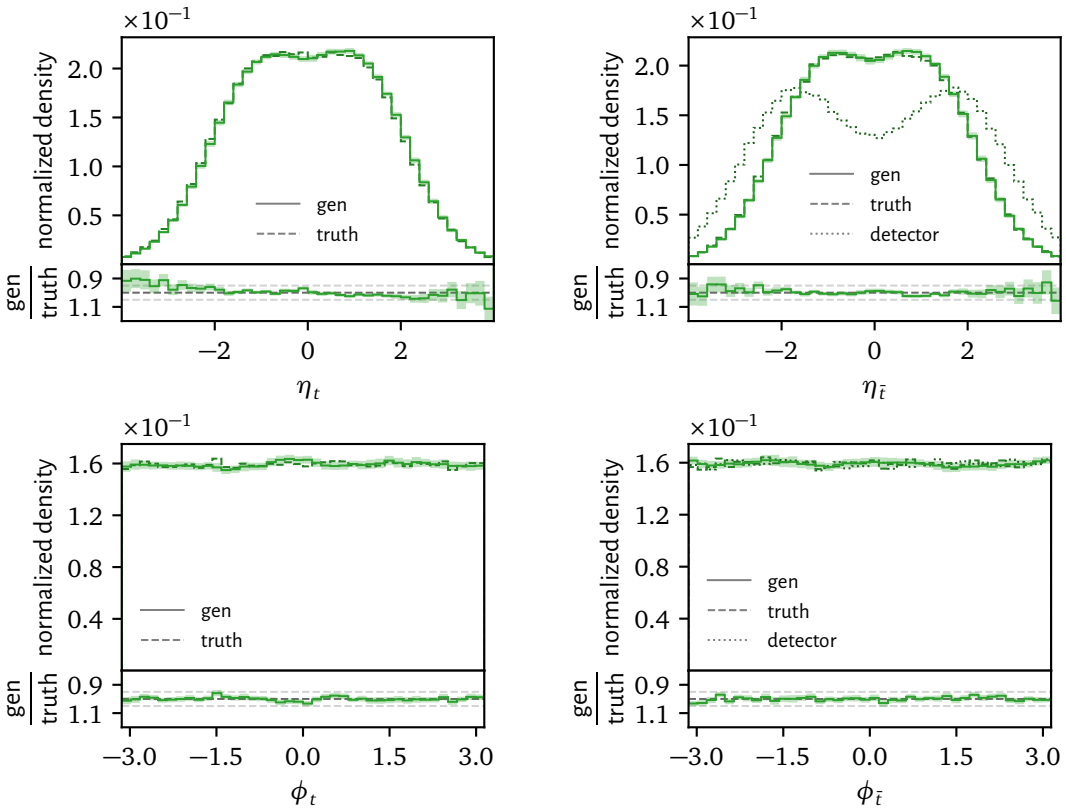

Figure 5.4.: Unfolded pseudorapidity and azimuthal angle distributions of the top quarks, compared to the parton level truth.

considering the reconstruction of the two top quarks in figs. 5.3 and 5.4. What I show here and in the following are the model-generated unfolded distributions of respective observables, compared to the parton truth. The uncertainties, shown at 1σ, of the generated densities come from the Bayesian architecture and are due to the limited statistics of the training data. In practice, these uncertainties are computed by sampling from the network parameter distribution multiple times, generating all distributions for each parameter set and taking the mean and standard deviation of the bin heights. If there are easily accessible proxies for a particle on detector-level, a detector-level distribution is shown as well. For the u and d quarks, we use respectively the hardest or second-to-hardest lab-frame light jets as a proxy. For the leptonic

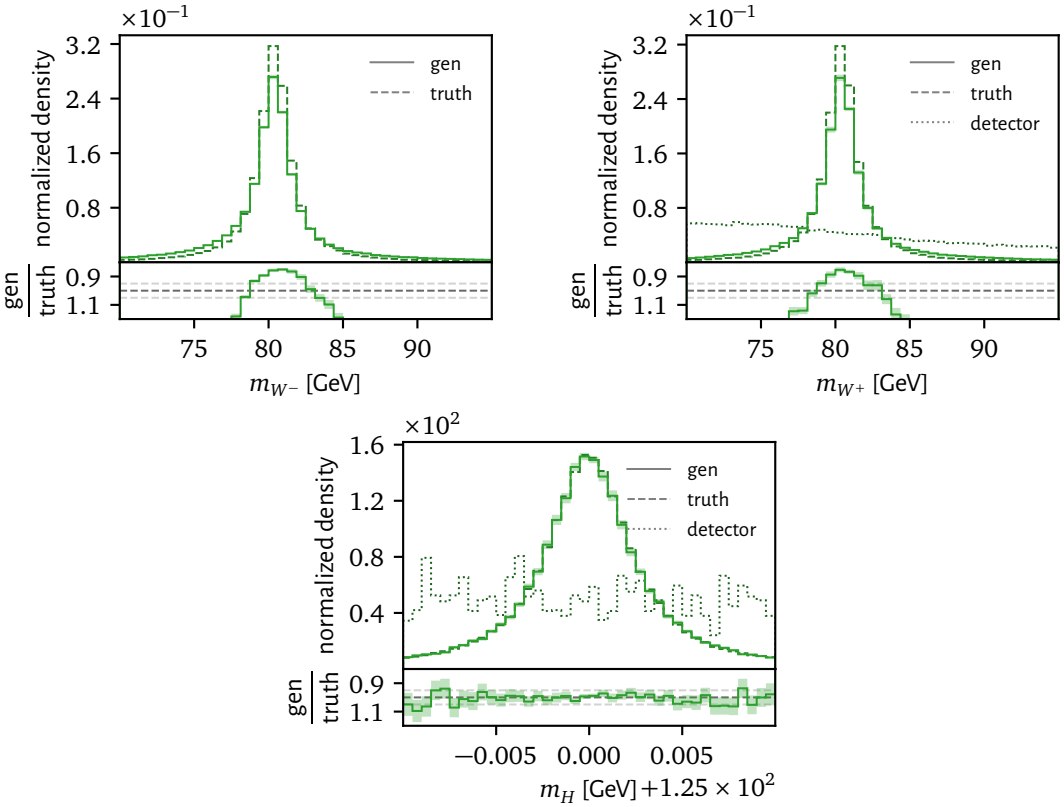

Figure 5.5.: Unfolded mass distributions of the top quarks, W bosons and Higgs, compared to the parton level truth.

and hadronic bottom quarks, this is done analogously with the b-tagged jets.

We can see that the distributions of both tops get reconstructed with only a few percent discrepancy between the unfolded distributions and the truth. In the tails, this deviation is necessarily a bit larger, due to low data statistics. In particular, we see how powerful our parameterization is at reconstructing the narrow top mass densities, without any additional treatment like an MMD loss. If we look at the remaining mass distributions in Fig. 5.5, we see that a dedicated parameterization is even able to reconstruct the—in our dataset—extremely narrow Higgs mass peak faithfully, which proved to be near impossible with an MMD loss. We also observe quite low precision in the W mass densities, due to the sign factor in (5.10). Note, however, that this

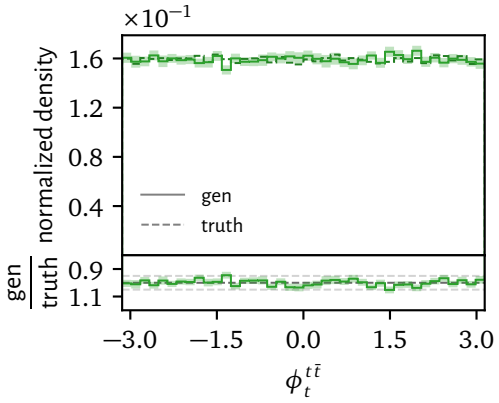

Figure 5.6.: Unfolded azimuthal angle distribution of the leptonic top quark in the $t\bar{t}$ rest frame, compared to the parton level truth.

reconstruction is still more accurate, than if we would omit the W mass from the parameterization.

Another question we should address is if the periodic spline architecture is fully able to fix the non-euclidean parameter topology that (5.10) introduces. We see that the azimuthal angle distributions of both tops get unfolded properly, however, these are not featured in the parameterization. To directly see potential problems here, it is better to look at, for example, the azimuthal angle of the leptonic top in the $t\bar{t}$-frame in Fig. 5.6. With an euclidean flow, we would see strong deviations from the truth in the boundary regions, which, as we can see, is not the case here due to the use of periodic splines.

Besides the W mass distributions our model is somewhat imprecise when unfolding the transverse momentum distributions of quark final states (cf. Fig. 5.7). This particular weakness seems to be related to our choice of parameterization. On the other hand, jet combinatorics are likely a factor here, since we do not see any deviations in the transverse momenta of the lepton and the neutrino. Note that these effects persist even without the inclusion of ISR effects.

Now that we got a broad idea of general model performance, let us check the unfolding accuracy of CP-sensitive observables for $t\bar{t}h$. The choice of observ-

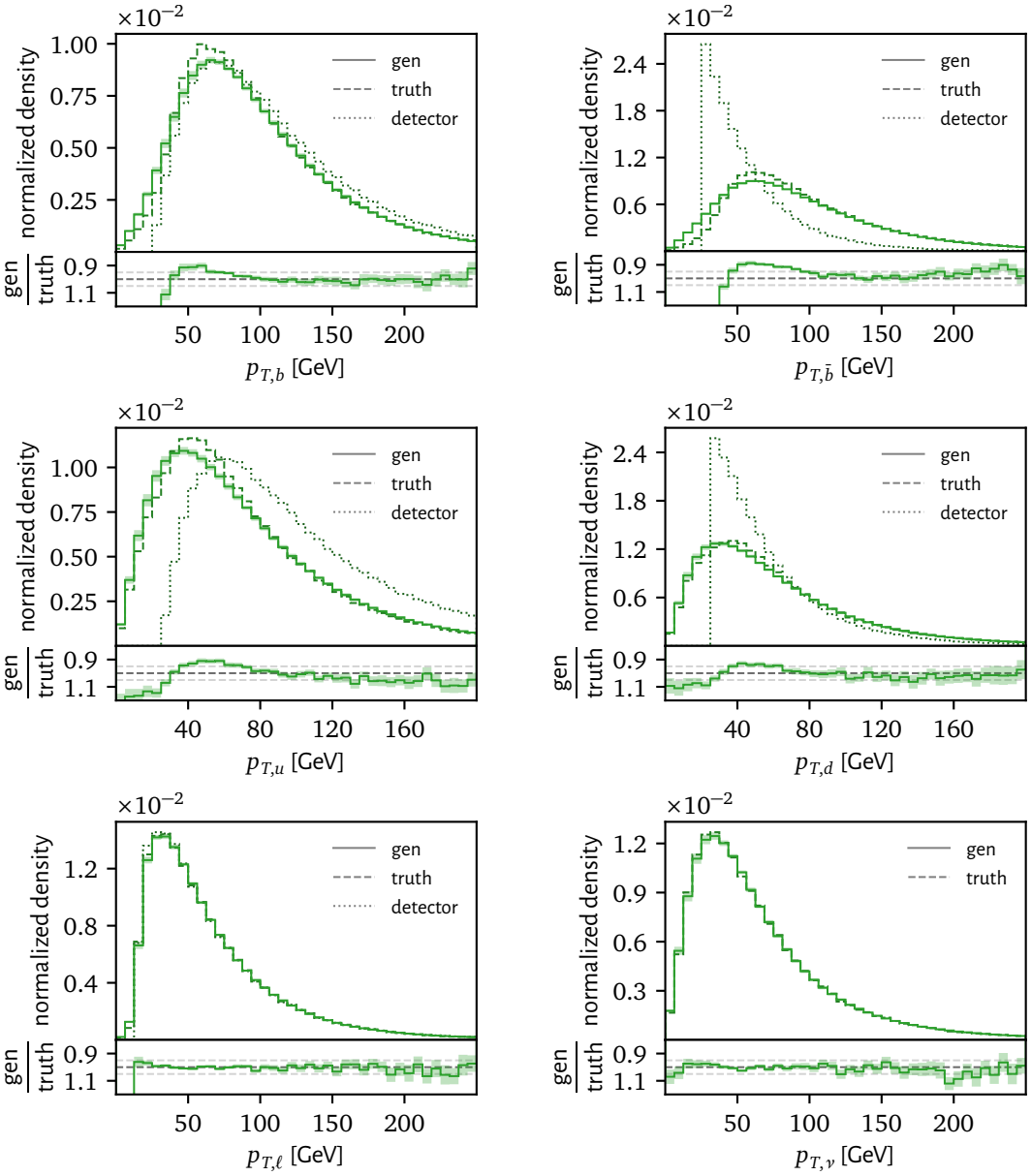

Figure 5.7.: Unfolded transverse momentum distributions of the b-, u- and d-quarks, as well as the lepton and neutrino.

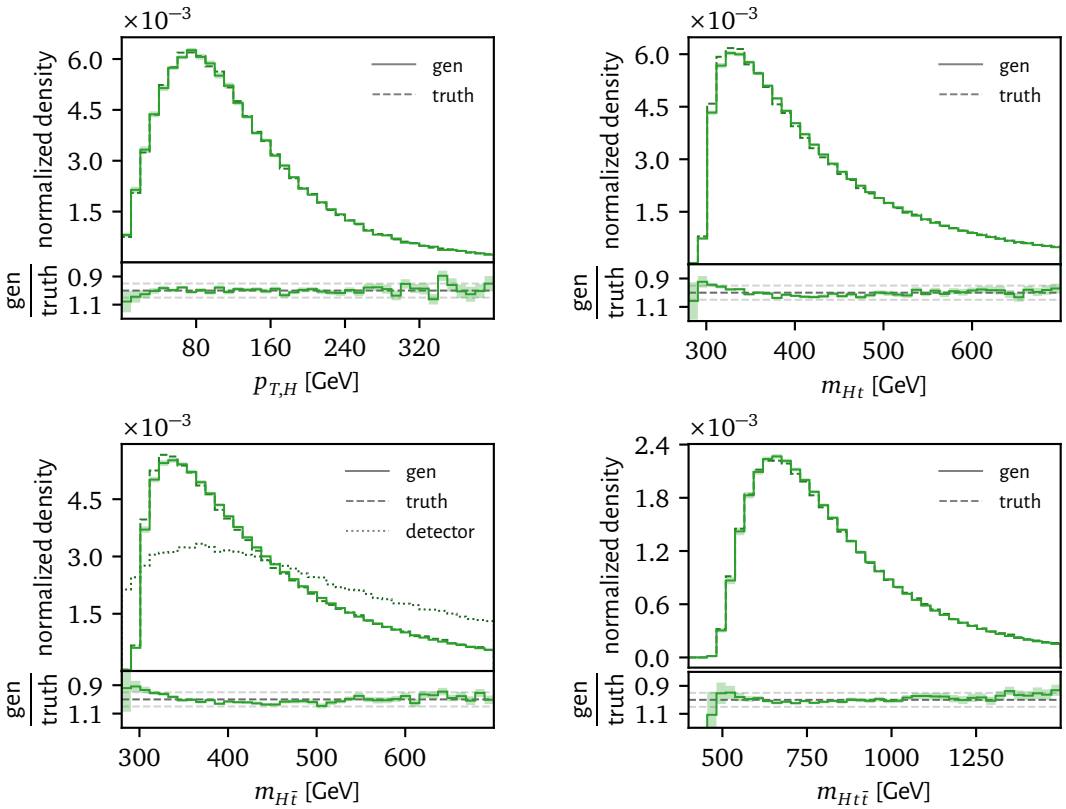

Figure 5.8.: Unfolded distributions of the CP-sensitive transverse Higgs momentum and various invariant masses among the top quarks and the Higgs.

ables is based on Ref. [41] as discussed in Section 4.3. For the (non-direct) CP-sensitive observables shown in figs. 5.8 and 5.9, we obtain a comparable performance to previous results regarding the top quarks. For the genuine CP-odd azimuthal angle differences, the accuracy is more limited, as shown in figs. 5.10 and 5.11.

The overall precise reconstruction of global distributions is a good starting point. However, ultimately we want the model to faithfully capture discrepancies induced by new physics in measured data. So reproducing global distributions is not enough and could in principle also be accomplished by a purely generative network without any detector-level input. To obtain indica-

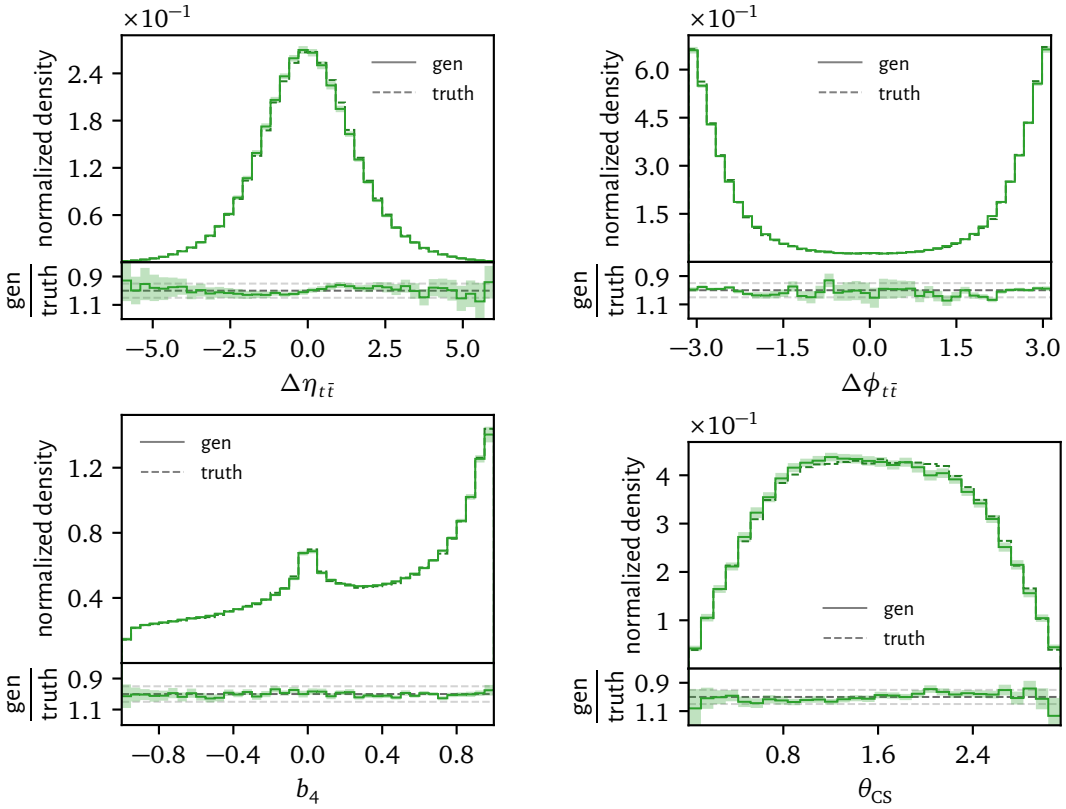

Figure 5.9.: Unfolded distributions of the CP-sensitive observables $\Delta\eta_{t\bar{t}}$, $\Delta\phi_{t\bar{t}}$, b_4 and θ_{CS}.

tions of how much the network actually respects the detector condition we can look at unfolded distributions $p(\boldsymbol{x}\,|\,\boldsymbol{y})$, where we fix the detector event $\boldsymbol{y}$. In Fig. 5.12 we show two examples of the conditional Collins-Soper angle distribution for two randomly selected detector events. We can see that we obtain a quite narrow distribution for the first event, compared to the width of the global θ_{CS} distribution. This indicates that the model can extract a good amount of information from this particular event. For the second event, the extracted amount of information is visibly reduced; we have two peaks which are both broader than the peak of the first event.

Obviously, we do not get very far by looking at single events. A better option is to quantify the amount of information in distributions like this with a single

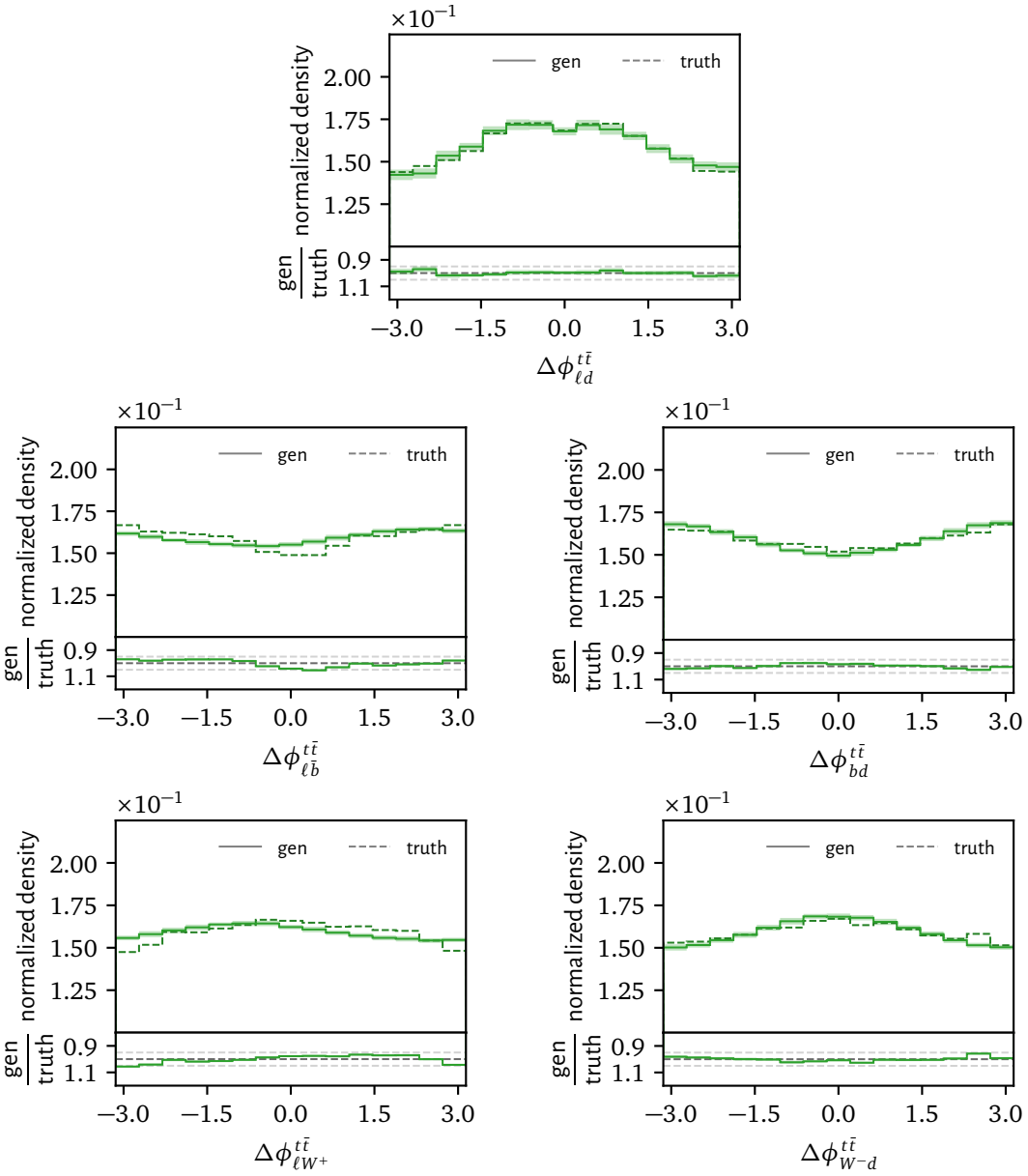

Figure 5.10.: Unfolded distributions of genuine CP-sensitive azimuthal angle differences including the lepton and the d-quark.

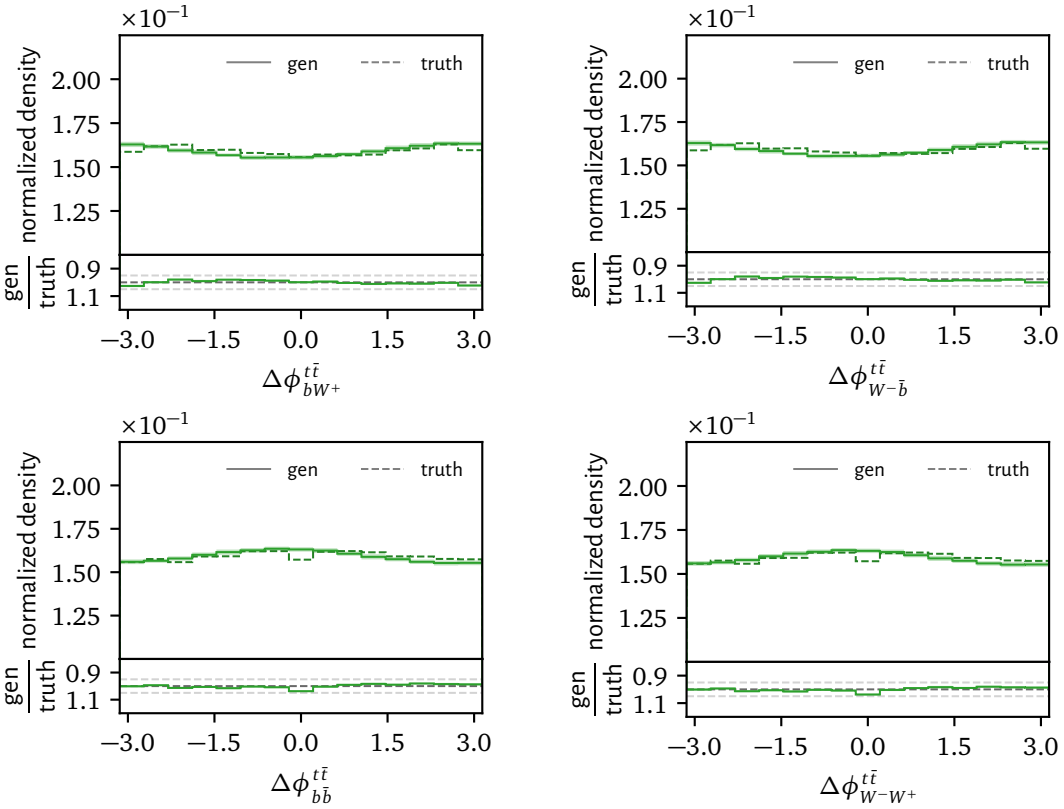

Figure 5.11.: Unfolded distributions of genuine CP-sensitive azimuthal angle differences including only the W bosons and the b-quarks.

number. We could then plot this number for a representative number of event samples. If we have a distribution $p(\mathcal{O}|\boldsymbol{y})$, for some observable $\mathcal{O} = \mathcal{O}(\boldsymbol{x})$, the Shannon entropy [125]

$$H = -\int_{\mathcal{O}_a}^{\mathcal{O}_b} p(\mathcal{O}|\boldsymbol{y}) \log(p(\mathcal{O}|\boldsymbol{y}))\ \mathrm{d}\mathcal{O}, \tag{5.13}$$

is a suitable approach for this.

Realistically we have to choose some interval $[\mathcal{O}_a, \mathcal{O}_b]$ over which to compute

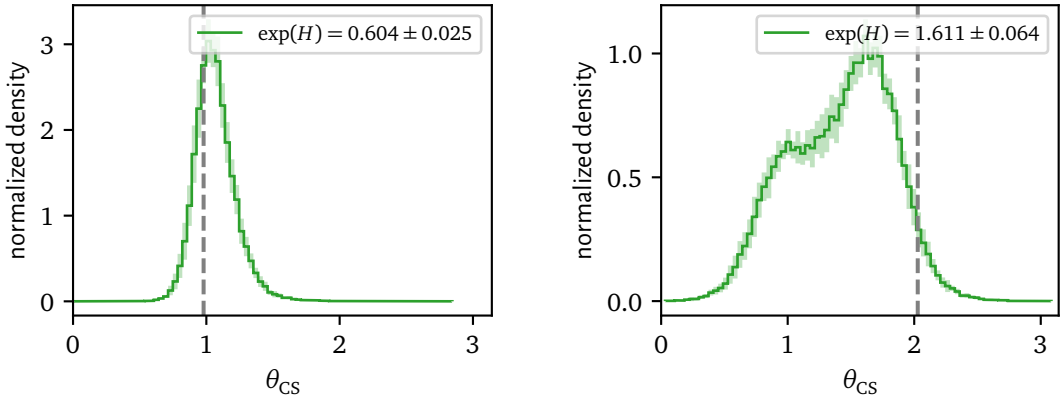

Figure 5.12.: Unfolded distributions of the Collins-Soper angle, conditioned on two randomly selected detector events. The corresponding parton truth to these events is shown by the dashed gray lines. We also show the exponential of the Shannon entropy for both distributions.

H and approximate it by introducing a binning:[3]

$$H \approx -\frac{\mathcal{O}_b - \mathcal{O}_a}{n} \sum_{k=1}^{n} h_k \log(h_k). \tag{5.14}$$

Here h_k corresponds to the k-th bin height. In the following we will compute any entropies using 1000 samples and 80 bins, while $[\mathcal{O}_a, \mathcal{O}_b]$ will always correspond to the interval in which we previously showed the global distribution. Moreover, we also compute uncertainties for the entropy, coming from the variations in the (Bayesian) network parameters.[4]

To make H more interpretable, we note that the maximal and minimal values H can take are $H = \log(\mathcal{O}_b - \mathcal{O}_a)$ and negative infinity, such that $0 \leq \exp(H) \leq \mathcal{O}_b - \mathcal{O}_a$.[5] In the former case, we have a uniform distribution with no informa-

[3] In principle we could get a better estimate for this entropy by letting our network estimate the density $p(\boldsymbol{x}|\boldsymbol{y})$ and use it to compute $p(\mathcal{O}|\boldsymbol{y})$. In this case, however, we would have to compute the Jacobian for every observable, which does not scale well without some form of automatic differentiation.

[4] In practice, this is done by sampling multiple parameter sets, computing the entropy for each and then taking the mean and standard deviation.

[5] This is of course not entirely true since the network has a prior. The entropy will thus

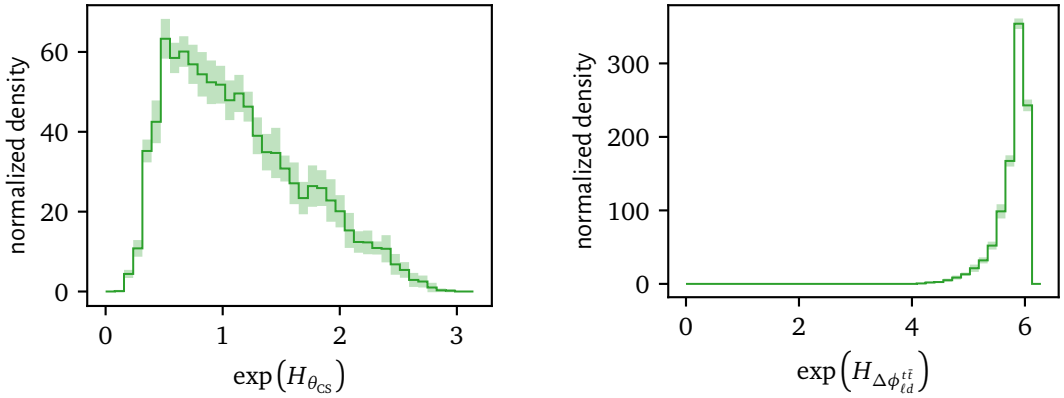

Figure 5.13.: Entropy distribution for the conditional probability densities of θ_{CS} and $\Delta\phi^{t\bar{t}}_{\ell d}$ for 1000 randomly selected detector events.

tion, in the latter case we have delta distribution with maximal information. This suggests to interpret $\exp(H)$ as some form of 'generalized distribution width'—for instance we see that $\exp(H) \sim \sigma$ for a gaussian—, that still has a proper definition for e.g. multimodal distributions. The exponential of the entropy for both distributions is shown in Fig. 5.12. We can see that these values indeed correspond to values one would intuitively associate with the 'width' of these distributions.

Now that we have a proper understanding of what the entropy tells us, let us consider entropy distributions for a sufficiently large sample of detector events. This should give us a quantitative statement about the amount of information the network is able to extract from any given detector event.

Such distributions are shown for θ_{CS} and $\Delta\phi^{t\bar{t}}_{\ell d}$ in Fig. 5.13. For θ_{CS} the distribution clearly leans towards zero, i.e. the most probable conditional densities are more or less similar to the first density in Fig. 5.12. They should thus contain a large amount of information about the value of θ_{CS}. For $\Delta\phi^{t\bar{t}}_{\ell d}$ the situation is reversed. The distribution leans very strongly to the right and most detector events contain close to no information about the underlying

also be limited by the entropy of the prior, i.e. the global distribution for the respective observable. However, in our case, a simplified picture will suffice.

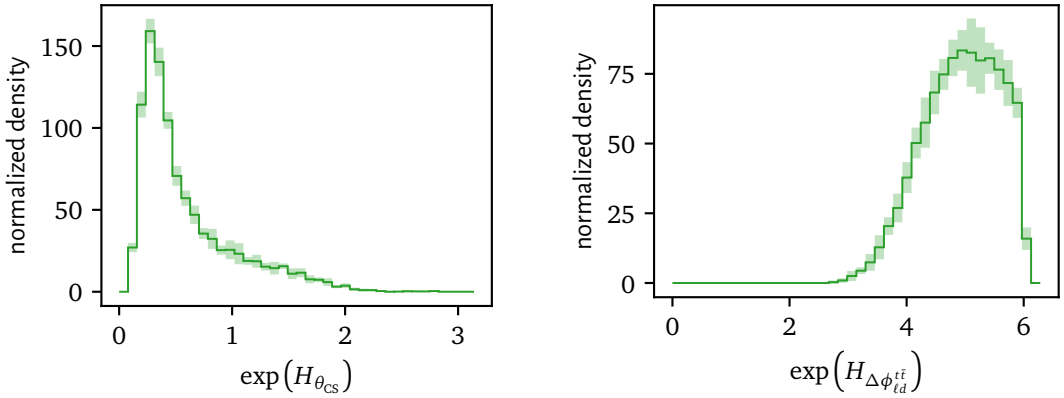

Figure 5.14.: Entropy distribution for the conditional probability densities of θ_{CS} and $\Delta\phi^{t\bar{t}}_{\ell d}$ for 1000 randomly selected detector events. These distributions were obtained by training our network on and unfolding a dataset without ISR.

value of $\Delta\phi^{t\bar{t}}_{\ell d}$ on parton-level. This picture persists across other CP-sensitive observables. For non-direct CP-sensitive observables we can extract a significant amount of information, comparable to θ_{CS}, while the opposite is true for genuine CP-odd observables. For the latter, the network essentially ignores the detector condition. This also means that the network will always generate the SM prior distributions (figs. 5.10 and 5.11), no matter what the input on detector-level is. Hence, we are not sensitive to new physics for these observables.

This poses the question if this is due to bad network performance or if there is simply no information in the dataset. For starters, we see in Fig. 5.14 that we get significantly more information on $\Delta\phi^{t\bar{t}}_{\ell d}$—and overall for that matter—, if we do not include ISR effects in our dataset. So in principle, the network can extract some information on $\Delta\phi^{t\bar{t}}_{\ell d}$, but the additional ISR jets make it particularly difficult to resolve the combinatorics and reconstruct the $t\bar{t}$ frame as well as the d quark.

I further tried to simplify the network architecture to a bare minimum, by using $\Delta\phi^{t\bar{t}}_{\ell d}$ as the only channel of the flow. In this case, the network does two things. First, it computes spline parameters by processing respective detector

events with a fully connected conditioner network. Second, it uses these spline parameters to transform a one-dimensional uniform distribution into $\Delta\phi_{\ell d}^{t\bar{t}}$. Since even a single spline transformation is able to learn arbitrary distributions, given enough bins, the second step does not realistically limit the network. So the only question we are asking here, is if a fully connected neural network can extract any significant information on $\Delta\phi_{\ell d}^{t\bar{t}}$ from a detector level event. And in fact, the network did not prove to be able to extract any more information on $\Delta\phi_{\ell d}^{t\bar{t}}$ than if we do a full parton-level phase space reconstruction. This gives us at least a good indication that, realistically, ISR-inclusive $t\bar{t}h$ detector data might not have any information about $\Delta\phi_{\ell d}^{t\bar{t}}$—and likely other genuine CP-odd observables—associated with it. For this reason, we will mainly focus on the non-direct CP-sensitive observables for the rest of this section.

Now, let us finally address unfolding in the context of BSM data. In particular we want to consider two BSM scenarios with $\alpha = \pi/4$ and $\alpha = -\pi/4$. For these, we want to know how accurately we can differentiate between each scenario and SM data, based on non-direct CP-sensitive observable distributions. As discussed in Section 3.1, we can reject any physics model hypothesis, just using a network that is trained on data from this model. In this case, we can unfold our measured data and look for any discrepancies between the unfolded data and the parton truth of our hypothesis. However, if we want to accept a hypothesis, we need to do this by rejecting alternatives, since our unfolded distribution will always have some form of prior dependence. Hence, we need to train a model for each hypothesis we are interested in. Alternatively, we could also train a model that is conditioned on the relevant model parameter(s). If we then unfold our detector data once under every hypothesis and compare the result with the respective truth, we can identify the most likely model for which the unfolded data is closest to the truth distribution. Furthermore, we can reject hypotheses with significant deviations from this truth.

Let me demonstrate this concept with concrete examples. In Fig. 5.15 the results of unfolding an SM detector level distribution under different model

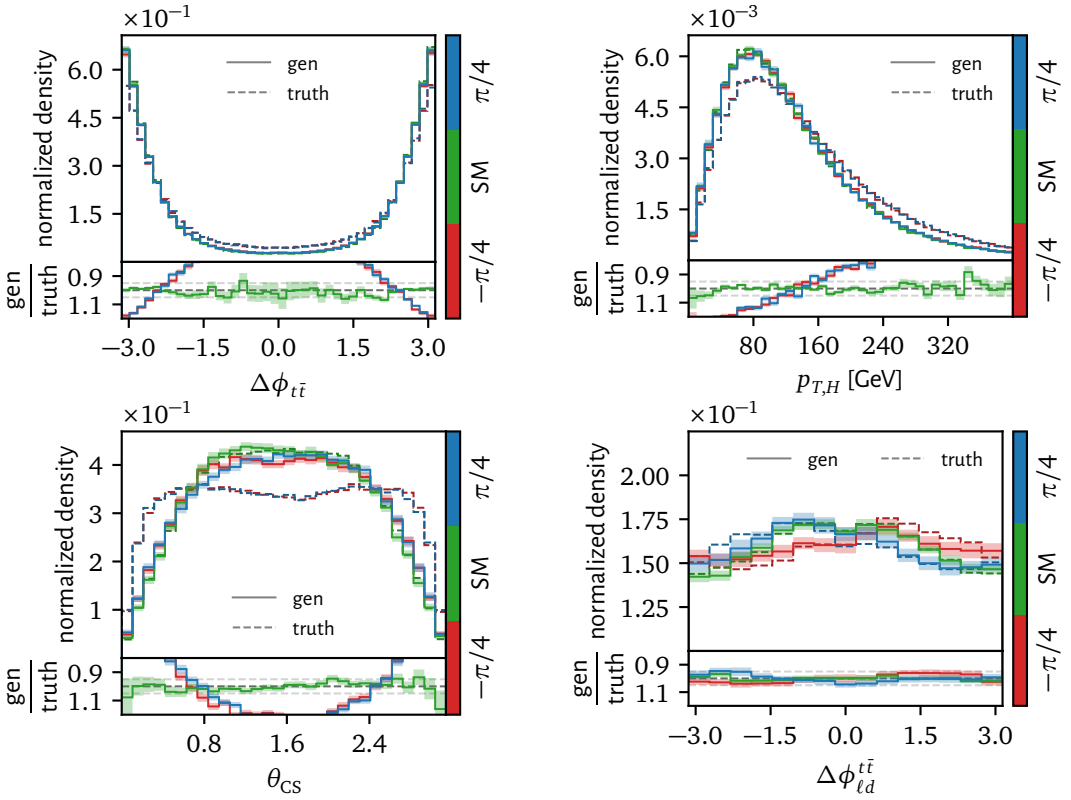

Figure 5.15.: Unfolded distributions of θ_{CS} and $\Delta\phi_{\ell d}^{t\bar{t}}$, obtained by unfolding SM detector data with our model, if trained on either of the hypotheses $\alpha = -\pi/4$, $\alpha = 0$ or $\alpha = \pi/4$. The ratios in the bottom panels are computed between the model and truth distributions corresponding to the same hypothesis.

hypotheses are shown. We can see that we get some deviations between the three generated distributions due to the prior dependence. These deviations can be interpreted as an additional systematic uncertainty of the unfolding procedure. While the network strongly deviates from the prior for θ_{CS}, $\Delta\eta_{t\bar{t}}$ and b_4, in the case of $\Delta\phi_{\ell d}^{t\bar{t}}$ the prior gets closely reconstructed for each hypothesis. For the non-direct CP-sensitive observables, it is thus quite clear that we can reject $\alpha = \pi/4$ as well as $\alpha = -\pi/4$ and can accept the SM. For $\Delta\phi_{\ell d}^{t\bar{t}}$ on the other hand, we cannot visually accept or reject any hypothesis without further information.

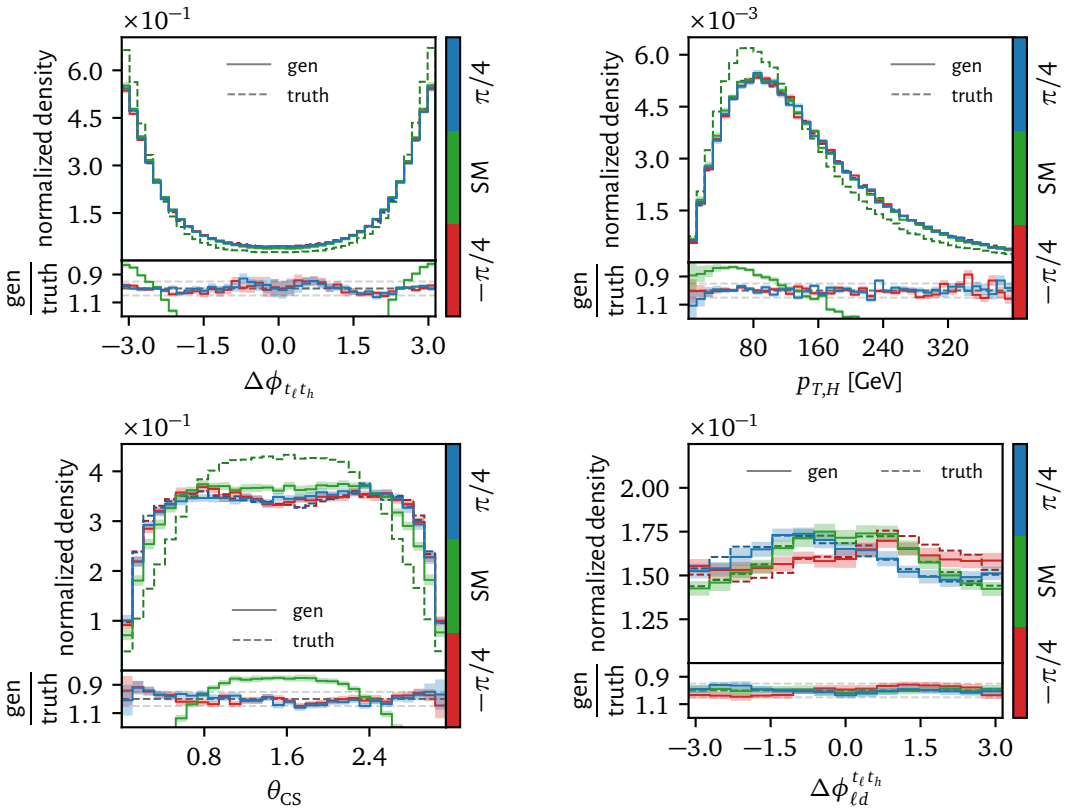

Figure 5.16.: Unfolded distributions of θ_{CS} and $\Delta\phi_{\ell d}^{t\bar{t}}$, obtained by unfolding detector data corresponding to $\alpha = \pi/4$ with our model, trained on either of the hypotheses $\alpha = -\pi/4$, $\alpha = 0$ or $\alpha = \pi/4$. The ratios in the bottom panels are computed between the model and truth distributions corresponding to the same hypothesis.

In Fig. 5.16 analogous unfolded distributions are shown for detector data corresponding to $\alpha = \pi/4$. Here, we can make similar conclusions. We again see some shaping, this time of the unfolded distribution under the SM hypothesis towards the SM prior for θ_{CS}.

The model also has sensitivity to smaller (absolute) values of α, as illustrated in Fig. 5.17. Note that we consider $\alpha = 13°$ here since Ref. [41] projected this bound on α for the HL-LHC with classical kinematic reconstruction methods. We cannot make a direct comparison, however, since background is not included in the present analysis. While the prior induced variations of the

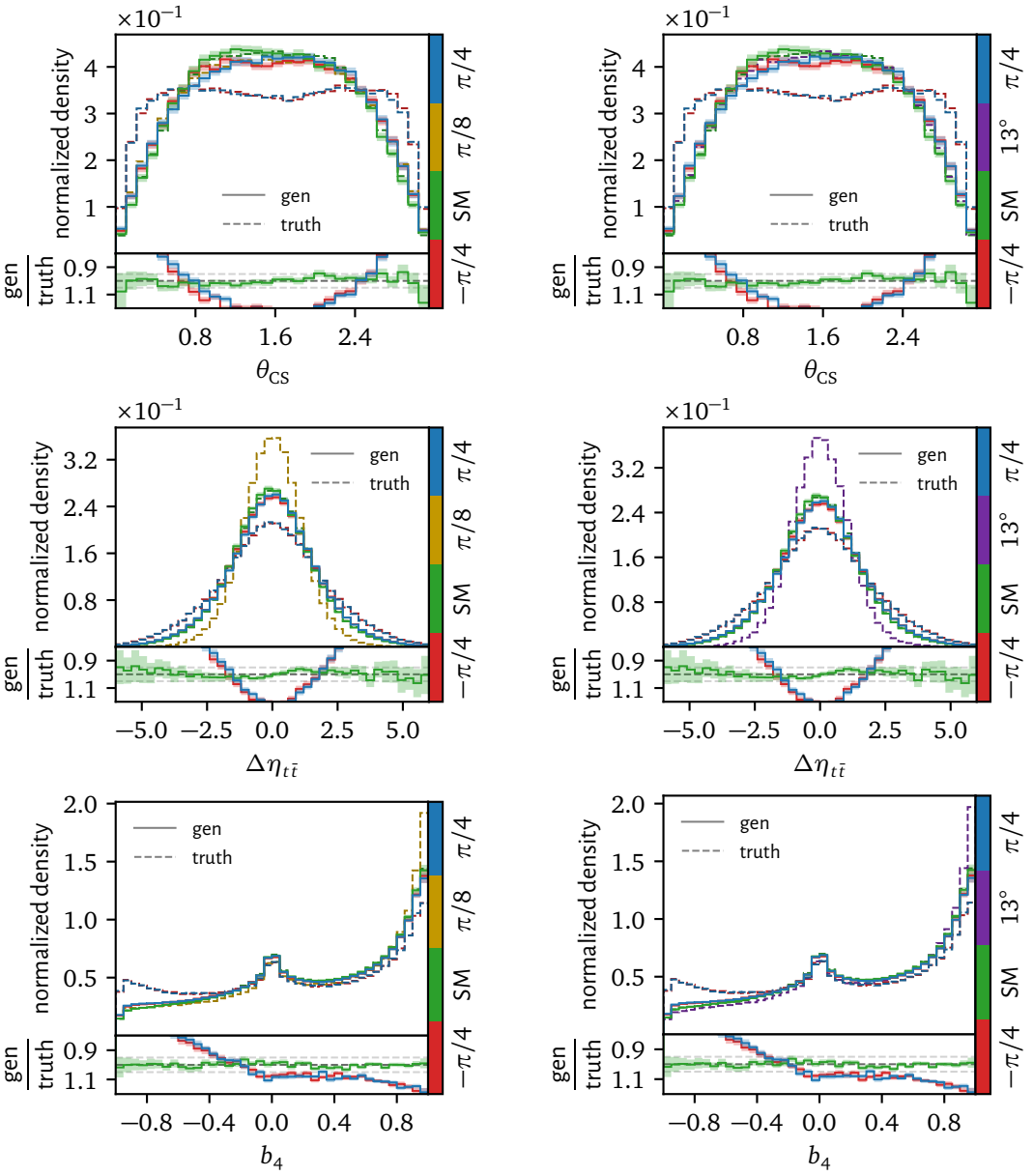

Figure 5.17.: Unfolded distributions of θ_{CS} and $\Delta\phi_{\ell d}^{t\bar{t}}$, obtained by unfolding SM detector data with our model, trained on either of the hypothesis $\alpha = -\pi/4$, $\alpha = 0$ or $\alpha = \pi/4$. The truth for $\alpha = \pi/8$ and $\alpha = 13°$ is also shown. The ratios in the bottom panels are computed between the model and truth distributions corresponding to the same hypothesis.

generated distributions in Fig. 5.17 are not sufficiently small to hope for sensitivity from θ_{CS}, we obtain better results for $\Delta\eta_{t\bar{t}}$ and b_4. For $\Delta\eta_{t\bar{t}}$ we obtain a clear discrepancy between the generated distributions and the truths for both $\alpha = \pi/8$ and $\alpha = 13°$. For b_4, on the other hand, it is not clear if we have sensitivity since the deviations here are quite small. Note that no model was trained under the hypotheses corresponding to these α values, but we assume that the variations in the generated distributions for the $\alpha = \pi/4$ and $\alpha = -\pi/4$ hypotheses are reasonably representative of similar distributions for $\pi/8$ and 13°. I also did not unfold detector data correponding to $\pi/8$ and 13°, however, it is reasonable to expect similar variations in the generated distributions. Under this assumption, we can expect to accurately discern between SM data and data corresponding to $\alpha = \pi/8$ and $\alpha = 13°$.

These examples illustrate why a single SM-trained network is never able to accept BSM hypotheses. If we would compare this distribution to parton-level truths for different values of α, we would obtain a match for some α whose absolute value is slightly smaller than $\pi/4$. In a realistic setting, we could only hope that deviations are small enough for our use-case. Otherwise, we have to rely on a conditioned or multiple networks, as we do here.

How can we quantify the rejection or acceptance of a model hypothesis? A naive route, which we will take here, is to compare binned generated and expected distributions with a χ^2-test [126]. In particular, for our three model hypotheses $\alpha = -\pi/4$, SM and $\alpha = \pi/4$, we want to unfold detector data, simulated using that particular model, to obtain a binned distribution $N_{i,\text{exp}}$. We then compare this unfolded data to the distributions $N_{i,\text{gen}}$, which we actually get when we unfold test data with all three models. In reality, this test data would be measured data. For n bins we obtain

$$\chi^2_{\text{red}} = \frac{1}{n-1} \sum_{i=1}^{n} \frac{(N_{i,\text{gen}} - N_{i,\text{exp}})^2}{N_{i,\text{gen}} + N_{i,\text{exp}}}. \tag{5.15}$$

Here we use the Poisson errors $\sqrt{N_{i,\text{gen}}}$ and $\sqrt{N_{i,\text{exp}}}$ of the bin heights in the

denominator, while $n-1$ are the degrees of freedom of our binned distribution, if we know the total number of events.

We can further use $\chi^2 = (n-1)\chi^2_{\text{red}}$ to compute the p-value of our data, given some model hypothesis. In tables 5.2 and 5.3 the χ^2_{red} and p-values are listed for unfolded data corresponding to the SM and $\alpha = \pi/4$ respectively, under different model hypotheses. With very high levels of certainty, we correctly accept the SM and reject both BSM hypotheses in table 5.2 for all non-direct CP-sensitive observables; the only exception is m_H. The results for $\Delta\phi^{t\bar{t}}_{\ell d}$ are also shown and as expected, here we cannot differentiate between the three hypotheses with significance. For illustration purposes, results for the two and three-dimensional correlations $\theta_{\text{CS}}, \Delta\eta_{t\bar{t}}$ and $\theta_{\text{CS}}, \Delta\eta_{t\bar{t}}, b_4$ are also shown respectively. In table 5.3 we get similar numbers and correctly accept the $\alpha = \pi/4$ case, while rejecting the SM. Again, m_H is the exception among the non-direct CP-sensitive observables. For $\Delta\phi^{t\bar{t}}_{\ell d}$ we see a slight but insignificant preference for the SM in Table 5.2, while, interestingly, $\Delta\phi_{\ell d}$ alone is enough to rule out the SM, as we can see from the p-value in 5.3. This indicates that the small amount of information contained in $\Delta\phi^{t\bar{t}}_{\ell d}$ (cf. Fig. 5.13) is enough to make a difference here. However, it does not suffice to differentiate $\pi/4$ from $-\pi/4$. The $\alpha = -\pi/4$ case is not shown but leads to comparable results as $\alpha = \pi/4$. Note that we cannot differentiate between $\alpha = \pi/4$ and $\alpha = -\pi/4$, since, again, we cannot appropriately reconstruct observables that are sensitive to the sign of α.

Note that our current approach is equivalent to one iteration of IBU, as explained in Section 3.4. Hence, we would expect that we can further reduce the prior dependence on the non-direct CP-sensitive observables, by including more iterations. In particular, we can apply the method from Ref. [38] to do this. In practice and for the non-direct CP-observables, this could suffice to eliminate the need for multiple networks trained on different theory parameters, if we want to accept BSM hypotheses. However, we already discussed in Section 3.2.1 that IBU (and any approach based on it) can generally not remove the full prior dependence. And we have to expect that the genuine CP-odd

	$\chi^2_{\text{red}}\left(-\frac{\pi}{4}\right)$	$\chi^2_{\text{red}}(\text{SM})$	$\chi^2_{\text{red}}\left(\frac{\pi}{4}\right)$
$p_{T,H}$	58.41 ± 0.51	1.03 ± 0.20	59.94 ± 0.65
m_H	1.03 ± 0.14	0.86 ± 0.16	0.97 ± 0.19
m_{Ht}	36.99 ± 0.91	1.01 ± 0.17	37.24 ± 0.77
$m_{H\bar{t}}$	31.49 ± 0.84	1.00 ± 0.21	32.79 ± 0.73
$m_{Ht\bar{t}}$	48.44 ± 0.90	0.93 ± 0.16	49.4 ± 1.0
$\Delta\eta_{t\bar{t}}$	90.5 ± 1.5	1.00 ± 0.19	88.6 ± 1.8
$\Delta\phi_{t\bar{t}}$	76.5 ± 1.7	0.94 ± 0.17	75.8 ± 2.2
b_4	69.7 ± 1.5	1.01 ± 0.20	66.7 ± 1.6
θ_{CS}	101.1 ± 1.6	0.97 ± 0.15	98.4 ± 2.0
$\Delta\phi^{t\bar{t}}_{\ell d}$	1.18 ± 0.20	0.99 ± 0.15	1.25 ± 0.22
θ_{CS}, $\Delta\eta_{t\bar{t}}$	156.7 ± 2.5	0.92 ± 0.19	152.3 ± 3.0
θ_{CS}, $\Delta\eta_{t\bar{t}}$, b_4	182.5 ± 3.3	0.88 ± 0.20	176.1 ± 3.8
	$p\left(-\frac{\pi}{4}\right)$	$p(\text{SM})$	$p\left(\frac{\pi}{4}\right)$
$p_{T,H}$	0 ± 0	0.47 ± 0.31	0 ± 0
m_H	0.45 ± 0.26	0.72 ± 0.25	0.54 ± 0.31
m_{Ht}	0 ± 0	0.46 ± 0.27	0 ± 0
$m_{H\bar{t}}$	0 ± 0	0.51 ± 0.31	0 ± 0
$m_{Ht\bar{t}}$	0 ± 0	0.61 ± 0.25	0 ± 0
$\Delta\eta_{t\bar{t}}$	0 ± 0	0.53 ± 0.29	0 ± 0
$\Delta\phi_{t\bar{t}}$	0 ± 0	0.60 ± 0.29	0 ± 0
b_4	0 ± 0	0.49 ± 0.30	0 ± 0
θ_{CS}	0 ± 0	0.53 ± 0.26	0 ± 0
$\Delta\phi^{t\bar{t}}_{\ell d}$	0.26 ± 0.23	0.51 ± 0.26	0.19 ± 0.21
θ_{CS}, $m_{Ht\bar{t}}$	0 ± 0	0.60 ± 0.26	0 ± 0
θ_{CS}, $\Delta\eta_{t\bar{t}}$, b_4	0 ± 0	0.63 ± 0.26	0 ± 0

Table 5.2.: Reduced χ^2 and p-values obtained by unfolding SM data and comparing it to the distribution the model would generate if different model hypotheses were true. The test dataset consisted of 240 000 events, while 64 total bins were used per observable combination to bin the distributions.

	$\chi^2_{\text{red}}\left(-\frac{\pi}{4}\right)$	$\chi^2_{\text{red}}(\text{SM})$	$\chi^2_{\text{red}}\left(\frac{\pi}{4}\right)$
$p_{T,H}$	0.97 ± 0.14	57.4 ± 7.6	0.97 ± 0.20
m_H	0.93 ± 0.15	7.7 ± 3.8	0.93 ± 0.15
m_{Ht}	0.96 ± 0.18	36.2 ± 3.8	0.96 ± 0.20
$m_{H\bar{t}}$	1.02 ± 0.22	29.9 ± 4.5	0.99 ± 0.17
$m_{Ht\bar{t}}$	1.01 ± 0.14	42.4 ± 8.5	1.01 ± 0.18
$\Delta\eta_{t\bar{t}}$	0.99 ± 0.16	76 ± 14	0.99 ± 0.18
$\Delta\phi_{t\bar{t}}$	1.00 ± 0.22	63 ± 14	0.98 ± 0.16
b_4	0.98 ± 0.18	58 ± 11	0.97 ± 0.18
θ_{CS}	1.07 ± 0.17	83 ± 18	0.97 ± 0.18
$\Delta\phi^{t\bar{t}}_{\ell d}$	1.02 ± 0.19	4.2 ± 1.1	0.99 ± 0.14
θ_{CS}, $\Delta\eta_{t\bar{t}}$	1.01 ± 0.22	135 ± 26	0.99 ± 0.19
θ_{CS}, $\Delta\eta_{t\bar{t}}$, b_4	0.95 ± 0.26	157 ± 28	0.97 ± 0.23
	$p\left(-\frac{\pi}{4}\right)$	$p(\text{SM})$	$p\left(\frac{\pi}{4}\right)$
$p_{T,H}$	0.55 ± 0.26	0 ± 0	0.56 ± 0.32
m_H	0.63 ± 0.26	0 ± 0	0.61 ± 0.25
m_{Ht}	0.56 ± 0.28	0 ± 0	0.58 ± 0.32
$m_{H\bar{t}}$	0.50 ± 0.32	0 ± 0	0.52 ± 0.29
$m_{Ht\bar{t}}$	0.46 ± 0.24	0 ± 0	0.48 ± 0.29
$\Delta\eta_{t\bar{t}}$	0.51 ± 0.28	0 ± 0	0.53 ± 0.30
$\Delta\phi_{t\bar{t}}$	0.52 ± 0.31	0 ± 0	0.53 ± 0.25
b_4	0.54 ± 0.28	0 ± 0	0.55 ± 0.30
θ_{CS}	0.37 ± 0.27	0 ± 0	0.56 ± 0.30
$\Delta\phi^{t\bar{t}}_{\ell d}$	0.47 ± 0.29	0 ± 0	0.50 ± 0.24
θ_{CS}, $\Delta\eta_{t\bar{t}}$	0.47 ± 0.29	0 ± 0	0.50 ± 0.26
θ_{CS}, $\Delta\eta_{t\bar{t}}$, b_4	0.55 ± 0.29	0 ± 0	0.52 ± 0.27

Table 5.3.: Reduced χ^2 and p-values obtained by unfolding data, simulated with $\alpha = \pi/4$, and comparing it to the distribution the model would generate if different model hypotheses were true. The test dataset consisted of 240 000 events, while 64 total bins were used per observable combination to bin the distributions.

observables of $t\bar{t}h$ are an example for this. To remove the prior dependence from these observables with IBU (or equivalently, iterative normalizing flow unfolding) we would have to obtain, at some iteration point, a detector-level distribution (evidence) with a significant amount of information about the CP-odd observables. Since we already established that the detector simulation likely removes any significant information from the $\Delta\phi$'s (cf. Fig. 5.13), at least in theory, this cannot happen. In practice, the network might be able to use the little amounts of information that it can extract and remove some of the prior dependence. However, this will probably take large numbers of iterations.

6. Conclusion

The standard methodology has serious shortcomings when trying to unfold a process like $t\bar{t}h$. Missing energy and the combinatorial ambiguities associated with jets, especially in the presence of ISR, leads to a loss of information that is difficult to properly account for. Classical unfolding techniques rely here on the manual reconstruction of parton-level quantities as well as pre-selection of observables and binning [41, 83]. More modern approaches can avoid these issues through the use of machine learning [37, 40]. Normalizing flow unfolding is additionally able to unfold single events, lifting the requirement of sufficient statistics for unfolding to be valid.

In this thesis, we saw that conditional normalizing flows are an excellent tool to unfold $t\bar{t}h$-production. There are some difficulties associated with this process, in particular, the high number of intermediate particle mass peaks, jet combinatorics and missing neutrino energy. While normalizing flows can naturally capture the information loss associated with the latter two, generating any mass peak is difficult for a flow. I demonstrated how carefully chosen phase-space parameterizations can be excellent tools to deal with this particular and other generative weaknesses. I also showed that performance under parameterizations can suffer due to periodic parameters and introduced periodic spline transformers to solve this issue.

For signal-only detector-level data, the introduced model can very distinctly discern an absolute α value of $\pi/4$ and the SM, using many different non-direct CP-sensitive observables individually, while also having a clear sensitivity to the lower values $\alpha = \pi/8$ and $\alpha = 13°$. For direct CP-sensitive observables,

on the other hand, the model is hardly able to extract any information due to ISR effects, not giving us sensitivity to the sign of α. A model that purely reconstructs $\Delta\phi_{\ell d}^{t\bar{t}}$ cannot improve this situation, strongly indicating that information is already missing in the data. However, even the small amount of information we have is enough to distinctly rule out the SM for $\alpha = \pi/4$ detector data, based on the genuine CP-odd observable $\Delta\phi_{\ell d}^{t\bar{t}}$ alone.

We discussed that information loss, specifically with genuine CP-odd observables, and associated model dependence can play a significant role in the complex final state of $t\bar{t}h$; especially with the additional combinatorial problems that ISR introduces. I demonstrated that, in contrast to other unfolding techniques, normalizing flows allow us to analyze information loss directly through the full conditional distribution. But, as discussed in Section 3, all classical as well as contemporary unfolding techniques, including (iterative) normalizing flows, do not properly capture the uncertainties associated with these issues. Therefore, it becomes necessary to condition the unfolding on model parameters, drastically reducing practicality. These problems likely persist across similar processes and should be addressed.

Unfolding, be it in the form of just reversing detector effects or fully reconstructing parton-level observables, is a technique that is essential to many LHC studies. Normalizing flows allow us to mostly abstract away the many associated complexities and focus on the underlying physics. This method cannot limit analyses through artificial artifacts like badly chosen observables and binning, nor is it limited by low statistics. As many modern machine learning based techniques, flow unfolding can significantly boost our new physics sensitivity and simplify analyses, enabling us to fully exploit our potential for finding new physics at the HL-LHC. While highlighting some of the challenges, this thesis has shown how normalizing flows can be further improved towards this goal in the context of complex processes like $t\bar{t}h$.

A. Phase Space Parameterization Details

Let us discuss in detail how to apply and invert the phase space parameterization given in Eq. (5.10), i.e.

$$\begin{aligned}\Big(&\vec{p}_{t\bar{t}}, m_t, |\vec{p}_t^{\,\mathrm{CS}}|, \theta_t^{\mathrm{CS}}, \phi_t^{\mathrm{CS}}, m_{\bar{t}},\\ &\mathrm{sign}(\Delta\phi_{\ell\nu}^{t\bar{t}})\, m_{W_\ell}, |\vec{p}_\ell^{\,t\bar{t}}|, \theta_\ell^{t\bar{t}}, \phi_\ell^{t\bar{t}}, |\vec{p}_\nu^{\,t\bar{t}}|,\\ &\mathrm{sign}(\Delta\phi_{du}^{t\bar{t}})\, m_{W_h}, |\vec{p}_d^{\,t\bar{t}}|, \theta_d^{t\bar{t}}, \Delta\phi_{\ell d}^{t\bar{t}}, |\vec{p}_u^{\,t\bar{t}}|\Big).\end{aligned} \tag{A.1}$$

First we use $\vec{p}_{t\bar{t}}$ to boost into the Collins-Soper frame and store this momentum to be able to undo this boost. Recall that the Collins-Soper frame was defined as the $t\bar{t}$ rest frame whose z-axis forms equal angles with the beam axes. More precisely, the frame whose z-axis forms equal angles with one beam axis and the mirrored version of the other one. Usually, we go into this frame by boosting into some $t\bar{t}$ rest frame and then rotating the frame appropriately. However, we can also go to the Collins-Soper frame with two consecutive boosts—one longitudinal boost parallel to the beam axis and another transverse to it [127]. The longitudinal boost is then chosen such that $\vec{p}_{z,t\bar{t}} = 0$ afterward, where we assume that the beam axis coincides with the z-axis in the lab frame. Moreover, we choose the transverse boost such that $\vec{p}_{\mathrm{T},t\bar{t}} = 0$ afterwards. Overall, this ensures that $\vec{p}_{t\bar{t}} = 0$ after both boosts and that the z-axis is directed appropriately with respect to the beam axes.

In the Collins-Soper frame we further store m_t and $m_{\bar{t}}$ as well as $\vec{p}_t$ in spherical coordinates. Note that the polar angle of $\vec{p}_t$ is precisely the Collins-Soper

angle. We can use $\vec{p}_{\bar{t}} = -\vec{p}_t$ here to recover the momentum of the anti-top. Overall we now have stored eight parameters. Since we have six final states with four-momentum components each, minus one on-shell condition per final state, we need eighteen parameters overall.

Next, we want to rotate the frame so that $\vec{p}_t$ points in the z-direction. This simplifies the following calculations. We now start with the leptonic top decay, i.e. the decay of the regular top, and store $|\vec{p}_\ell|$, $\vec{\theta}_\ell$, ϕ_ℓ and $|\vec{p}_\nu|$ as well as $\text{sign}(\Delta\phi_{\ell\nu})m_{W_\ell}$. The last parameter is used as a proxy for $\Delta\phi_{\ell\nu}$ and ensures that the W-mass is directly part of the parameterization. If we do this analogously for the hadronic top decay, i.e. the decay of the anti-top, while relating $\ell \leftrightarrow d$ and $\nu \leftrightarrow u$, we obtain ten parameters more such that we have all eighteen needed parameters. Note that we store our CP-odd observable $\Delta\phi_{\ell d}$ for the hadronic top decay instead of ϕ_d.

But how do we undo the last part? First of all, the on-shell conditions of the lepton, neutrino and light quarks just give us the respective masses. The on-shell conditions of the b-quarks on the other hand are more complicated. Looking just at the leptonic b-quark we have

$$\begin{aligned} m_b^2 &= (p_t - p_{W^+})^2 \\ &= m_t^2 + m_{W^+}^2 - 2p_t \cdot (p_\ell + p_\nu). \end{aligned} \tag{A.2}$$

If we write out the last term and bring all masses to the same side we obtain

$$\begin{aligned} &E_t(|\vec{p}_\ell| + |\vec{p}_\nu|) - |\vec{p}_t||\vec{p}_\ell|\cos(\theta_\ell) - |\vec{p}_t||\vec{p}_\nu|\cos(\theta_\nu) \\ &\quad = \frac{1}{2}(m_t^2 + m_{W^+}^2 - m_b^2). \end{aligned} \tag{A.3}$$

We can use this equation to determine θ_ν, since we can infer all other quantities from the parameters we have. To determine the last missing piece, i.e. ϕ_ν, we notice that

$$m_{W^+}^2 = 2p_\ell \cdot p_\nu = 2|\vec{p}_\ell||\vec{p}_\nu|(1 - \cos(\theta_{\ell\nu})), \tag{A.4}$$

where we neglected all lepton, neutrino and light quark masses. The cosine of

the angle between ℓ and ν can be written as

$$\cos(\theta_{\ell\nu}) = \sin(\theta_\ell)\sin(\theta_\nu)\cos(\Delta\phi_{\ell\nu}) + \cos(\theta_\ell)\cos(\theta_\nu). \tag{A.5}$$

One can check this by writing out the scalar product $\vec{p}_\ell \cdot \vec{p}_\nu$ in spherical coordinates and using the cosine addition theorem. Eqs. (A.4) and (A.5) can be used together to determine $|\Delta\phi_{\ell\nu}|$ from m_{W^+}, since the cosine in (A.5) is insensitive to the sign of $\Delta\phi_{\ell\nu}$. But since we have stored sign($\Delta\phi_{\ell\nu}$) instead of m_{W^+}, this is not a problem and we can recover ϕ_ν. The parameterization of the hadronic top decay can be undone analogously.

List of Figures

List of Tables

Acronyms

BNN Bayesian neural network. 23, 27

BSM beyond Standard Model. 7, 30, 39, 77, 81, 82

CS Collins-Soper. 50

CTF continuous-time flow. 19

EFT effective field theory. 7, 39, 40

ELBO evidence lower bound. 24, 27, 63

EWSB electroweak symmetry breaking. 40

FSI final state interaction. 44–46

GAN generative adversarial network. 7

HEFT Higgs effective field theory. 54

HL-LHC high luminosity large hadron collider. 3, 4, 7, 54, 79, 87

IBU iterative Bayesian unfolding. 32–34, 36–38, 82, 85

ISR initial state radiation. 3, 4, 32, 53, 54, 65, 68, 76, 77, 86, 87, 92

KL Kullback-Leibler. 10, 11, 24, 25, 27, 56

LHC large hadron collider. 7, 47

MC Monte-Carlo. 25, 26, 28, 31

MMD maximum mean discrepancy. 56, 57, 59, 67

ODE ordinary differential equation. 19, 20

QFT quantum field theory. 39, 42

RNN recurrent neural network. 14

SM Standard Model. 3, 4, 30, 39, 40, 53, 65, 76–83, 86, 87, 92–94

VAE variational auto-encoder. 7

VI variational inference. 24, 64

Bibliography

[1] A. Collaboration, *Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC*, Physics Letters B **716**, 1 (2012), arXiv:1207.7214.

[2] C. Collaboration, *Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC*, Physics Letters B **716**, 30 (2012), arXiv:1207.7235.

[3] T. Plehn, A. Butter, B. Dillon and C. Krause, *Modern Machine Learning for LHC Physicists* (2022), arXiv:2211.01421.

[4] I. Brivio and M. Trott, *The Standard Model as an Effective Field Theory*, Phys. Rept. **793**, 1 (2019), arXiv:1706.08945.

[5] M. Feickert and B. Nachman, *A Living Review of Machine Learning for Particle Physics* (2021), arXiv:2102.02770.

[6] M. D. Schwartz, *Modern Machine Learning and Particle Physics*, Harvard Data Science Review (2021).

[7] P. Shanahan, K. Terao and D. Whiteson, *Snowmass 2021 Computational Frontier CompF03 Topical Group Report: Machine Learning* (2022), arXiv:2209.07559.

[8] B. Hashemi et al., *LHC analysis-specific datasets with Generative Adversarial Networks* (2019), arXiv:1901.05282.

[9] R. D. Sipio, M. F. Giannelli, S. K. Haghighat and S. Palazzo, *DijetGAN: a Generative-Adversarial Network approach for the simulation of QCD dijet events at the LHC*, Journal of High Energy Physics **2019** (2019).

[10] A. Butter, T. Plehn and R. Winterhalder, *How to GAN LHC events*, SciPost Physics **7** (2019).

[11] Y. Alanazi et al., *Simulation of Electron-Proton Scattering Events by a Feature-Augmented and Transformed Generative Adversarial Network (FAT-GAN)*, in Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (2021).

[12] L. de Oliveira, M. Paganini and B. Nachman, *Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis*, Computing and Software for Big Science **1** (2017).

[13] M. Paganini, L. de Oliveira and B. Nachman, *Accelerating Science with Generative Adversarial Networks: An Application to 3D Particle Showers in Multilayer Calorimeters*, Physical Review Letters **120** (2018).

[14] M. Paganini, L. de Oliveira and B. Nachman, *CaloGAN: Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks*, Physical Review D **97** (2018).

[15] P. Musella and F. Pandolfi, *Fast and Accurate Simulation of Particle Detectors Using Generative Adversarial Networks*, Computing and Software for Big Science **2** (2018).

[16] M. Erdmann, L. Geiger, J. Glombitza and D. Schmidt, *Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks* (2018), arXiv:1802.03325.

[17] M. Erdmann, J. Glombitza and T. Quast, *Precise Simulation of Electromagnetic Calorimeter Showers Using a Wasserstein Generative Adversarial Network*, Computing and Software for Big Science **3** (2019).

[18] D. Belayneh et al., *Calorimetry with deep learning: particle simulation and reconstruction for collider physics*, The European Physical Journal C **80** (2020).

[19] E. Buhmann et al., *Getting High: High Fidelity Simulation of High Granularity Calorimeters with High Speed*, Computing and Software for Big Science **5** (2021).

[20] E. Buhmann et al., *Decoding Photons: Physics in the Latent Space of a BIB-AE Generative Network*, EPJ Web of Conferences **251**, edited by C. Biscarat et al., 03003 (2021).

[21] S. Otten et al., *Event Generation and Statistical Sampling for Physics with Deep Generative Models and a Density Information Buffer* (2021), arXiv:1901.00875.

[22] S. Diefenbacher et al., *New Angles on Fast Calorimeter Shower Simulation* (2023), arXiv:2303.18150.

[23] E. Buhmann, G. Kasieczka and J. Thaler, *EPiC-GAN: Equivariant Point Cloud Generation for Particle Jets* (2023), arXiv:2301.08128.

[24] A. Butter et al., *Generative networks for precision enthusiasts*, SciPost Physics **14** (2023).

[25] K. Dohi, *Variational Autoencoders for Jet Simulation* (2020), arXiv:2009.04842.

[26] J. C. Cresswell et al., *CaloMan: Fast generation of calorimeter showers with density estimation on learned manifolds* (2022), arXiv:2211.15380.

[27] C. Krause and D. Shih, *CaloFlow: Fast and Accurate Generation of Calorimeter Showers with Normalizing Flows* (2023), arXiv:2106.05285.

[28] S. Diefenbacher et al., *L2LFlows: Generating High-Fidelity 3D Calorimeter Images* (2023), arXiv:2302.11594.

[29] A. Xu, S. Han, X. Ju and H. Wang, *Generative Machine Learning for Detector Response Modeling with a Conditional Normalizing Flow* (2023), arXiv:2303.10148.

[30] V. Mikuni and B. Nachman, *Score-based generative models for calorimeter shower simulation*, Physical Review D **106** (2022).

[31] V. Mikuni, B. Nachman and M. Pettee, *Fast Point Cloud Generation with Diffusion Models in High Energy Physics* (2023), arXiv:2304.01266.

[32] M. Leigh et al., *PC-JeDi: Diffusion for Particle Cloud Generation in High Energy Physics* (2023), arXiv:2303.05376.

[33] A. Butter et al., *Jet Diffusion versus JetGPT – Modern Networks for the LHC* (2023), arXiv:2305.10475.

[34] T. Finke, M. Krämer, A. Mück and J. Tönshoff, *Learning the language of QCD jets with transformers* (2023), arXiv:2303.07364.

[35] A. Butter et al., *Machine learning and LHC event generation*, SciPost Physics **14** (2023), arXiv:2203.07460.

[36] L. Ardizzone et al., *Analyzing Inverse Problems with Invertible Neural Networks* (2019), arXiv:1808.04730.

[37] M. Bellagente et al., *Invertible networks or partons to detector and back again*, SciPost Physics **9** (2020), arXiv:2006.06685.

[38] M. Backes, A. Butter, M. Dunford and B. Malaescu, *An unfolding method based on conditional Invertible Neural Networks (cINN) using iterative training* (2023), arXiv:2212.08674.

[39] V. Blobel, *Unfolding Methods in Particle Physics*, in PHYSTAT (2011), pages 240–251.

[40] A. Andreassen et al., *OmniFold: A Method to Simultaneously Unfold All Observables*, Physical Review Letters **124** (2020).

[41] R. K. Barman, D. Gonçalves and F. Kling, *Machine learning the Higgs boson-top quark CP phase*, Physical Review D **105** (2022).

[42] I. Kobyzev, S. J. Prince and M. A. Brubaker, *Normalizing Flows: An Introduction and Review of Current Methods*, IEEE Transactions on Pattern Analysis and Machine Intelligence **43**, 3964 (2021).

[43] G. Papamakarios et al., *Normalizing Flows for Probabilistic Modeling and Inference* (2021), arXiv:1912.02762.

[44] E. G. Tabak and C. V. Turner, *A Family of Nonparametric Density Estimation Algorithms*, Communications on Pure and Applied Mathematics **66**, 145 (2013).

[45] D. J. Rezende and S. Mohamed, *Variational Inference with Normalizing Flows* (2016), arXiv:1505.05770.

[46] V. I. Bogachev, A. V. Kolesnikov and K. V. Medvedev, *Triangular transformations of measures*, Sbornik: Mathematics **196**, 309 (2005).

[47] S. Kullback and R. A. Leibler, *On Information and Sufficiency*, Annals of Mathematical Statistics **22**, 79 (1951).

[48] L. Ardizzone et al., *Framework for Easily Invertible Architectures (FrEIA)* (2022).

[49] A. Vaswani et al., *Attention Is All You Need* (2017), arXiv:1706.03762.

[50] L. Dinh, D. Krueger and Y. Bengio, *NICE: Non-linear Independent Components Estimation* (2015), arXiv:1410.8516.

[51] L. Dinh, J. Sohl-Dickstein and S. Bengio, *Density estimation using Real NVP* (2017), arXiv:1605.08803.

[52] D. P. Kingma and P. Dhariwal, *Glow: Generative Flow with Invertible 1x1 Convolutions* (2018), arXiv:1807.03039.

[53] R. Prenger, R. Valle and B. Catanzaro, *WaveGlow: A Flow-based Generative Network for Speech Synthesis* (2018), arXiv:1811.00002.

[54] C. Durkan, A. Bekasov, I. Murray and G. Papamakarios, *Neural Spline Flows* (2019), arXiv:1906.04032.

[55] M. A. Marshall, *Positive polynomials and sums of squares*, in (2008).

[56] P. Jaini, K. A. Selby and Y. Yu, *Sum-of-Squares Polynomial Flow* (2019), arXiv:1905.02325.

[57] T. Müller et al., *Neural Importance Sampling* (2019), arXiv:1808.03856.

[58] H. M. Dolatabadi, S. Erfani and C. Leckie, *Invertible Generative Modeling using Linear Rational Splines* (2020), arXiv:2001.05168.

[59] C. Durkan, A. Bekasov, I. Murray and G. Papamakarios, *Cubic-Spline Flows* (2019), arXiv:1906.02145.

[60] J. Behrmann et al., *Invertible Residual Networks* (2019), arXiv:1811.00995.

[61] R. T. Q. Chen, J. Behrmann, D. Duvenaud and J.-H. Jacobsen, *Residual Flows for Invertible Generative Modeling* (2020), arXiv:1906.02735.

[62] R. van den Berg, L. Hasenclever, J. M. Tomczak and M. Welling, *Sylvester Normalizing Flows for Variational Inference* (2019), arXiv:1803.05649.

[63] H. Gouk, E. Frank, B. Pfahringer and M. J. Cree, *Regularisation of Neural Networks by Enforcing Lipschitz Continuity* (2020), arXiv:1804.04368.

[64] R. T. Q. Chen, Y. Rubanova, J. Bettencourt and D. Duvenaud, *Neural Ordinary Differential Equations* (2019), arXiv:1806.07366.

[65] W. Grathwohl et al., *FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models* (2018), arXiv:1810.01367.

[66] R. T. Q. Chen and D. Duvenaud, *Neural Networks with Cheap Differential Operators* (2019), arXiv:1912.03579.

[67] C. Winkler, D. Worrall, E. Hoogeboom and M. Welling, *Learning Likelihoods with Conditional Normalizing Flows* (2019), arXiv:1912.00042.

[68] Y. Gal, *Uncertainty in Deep Learning*, in (2016).

[69] H. Robbins and S. Monro, *A Stochastic Approximation Method*, The Annals of Mathematical Statistics **22**, 400 (1951).

[70] M. C. Fu, *Gradient Estimation*, in *Simulation*, Vol. 13, edited by S. G. Henderson and B. L. Nelson, Handbooks in Operations Research and Management Science, Elsevier (2006), Chapter 19, pages 575–616.

[71] P. W. Glynn, *Likelihood Ratio Gradient Estimation for Stochastic Systems*, Commun. ACM **33**, 75 (1990).

[72] J. Paisley, D. Blei and M. Jordan, *Variational Bayesian Inference with Stochastic Search* (2012), arXiv:1206.6430.

[73] R. J. Williams, *Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning*, Mach. Learn. **8**, 229 (1992).

[74] P. Glasserman, *Monte Carlo Methods in Financial Engineering*, 1st edition, Stochastic Modelling and Applied Probability №53, Springer (2003).

[75] D. P. Kingma and M. Welling, *Auto-Encoding Variational Bayes* (2022), arXiv:1312.6114.

[76] D. J. Rezende, S. Mohamed and D. Wierstra, *Stochastic Backpropagation and Approximate Inference in Deep Generative Models* (2014), arXiv:1401.4082.

[77] M. K. Titsias and M. Lázaro-Gredilla, *Spike and Slab Variational Inference for Multi-Task and Multiple Kernel Learning*, in Proceedings of the 24th International Conference on Neural Information Processing Systems, NIPS'11 (2011), pages 2339–2347.

[78] M. Opper and C. Archambeau, *The Variational Gaussian Approximation Revisited*, Neural Computation **21**, 786 (2009).

[79] D. P. Kingma, T. Salimans and M. Welling, *Variational Dropout and the Local Reparameterization Trick* (2015), arXiv:1506.02557.

[80] J. Alwall et al., *The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations*, Journal of High Energy Physics **2014** (2014).

[81] G. Cowan, *A Survey Of Unfolding Methods For Particle Physics*, Psychnology Journal (2002).

[82] H. B. Prosper and L. Lyons, editors, *Proceedings, PHYSTAT 2011 Workshop on Statistical Issues Related to Discovery Claims in Search Experiments and Unfolding, CERN,Geneva, Switzerland 17-20 January 2011*, CERN Yellow Reports: Conference Proceedings, Geneva: CERN (2011).

[83] V. Blobel, "Unfolding", in *Data Analysis in High Energy Physics*, John Wiley & Sons, Ltd (2013), Chapter 6, pages 187–225.

[84] A. Kirsch, *An Introduction to the Mathematical Theory of Inverse Problems*, eng, Third edition, Springer eBook Collection, Cham: Springer (2021), 1 Online–Ressource (XVII, 400 Seiten).

[85] G. D'Agostini, *A multidimensional unfolding method based on Bayes' theorem*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **362**, 487 (1995).

[86] P. Baroň, *Comparison of Machine Learning Approach to Other Commonly Used Unfolding Methods*, Acta Physica Polonica B **52**, 863 (2021).

[87] A. D. Sakharov, *Violation of CP invariance, C asymmetry, and baryon asymmetry of the universe*, Soviet Physics Uspekhi **34**, 392 (1991).

[88] P. Huet and E. Sather, *Electroweak Baryogenesis and Standard Model CP Violation*, Physical Review D **51**, 379 (1995).

[89] M. Gavela, P. Hernández, J. Orloff and O. Pène, *Standard Model CP-Violation and Baryon Asymmetry*, Modern Physics Letters A **09**, 795 (1994).

[90] M. Gavela et al., *Standard model CP-Violation and Baryon Asymmetry (II). Finite temperature*, Nuclear Physics B **430**, 382 (1994).

[91] P. Basler, M. Mühlleitner and J. Müller, *Electroweak Baryogenesis in the CP-Violating Two-Higgs Doublet Model* (2021), arXiv:2108.03580.

[92] C. P. Burgess, *Introduction to Effective Field Theory: Thinking Effectively about Hierarchies of Scale*, Cambridge University Press (2020).

[93] E. Geoffray, *Build from the SMEFT up, SMEFT global analyses as a bottom-up approach to constrain BSM physics*, eng, Heidelberg, (2023), 1 Online–Ressource (124 Seiten).

[94] P. Artoisenet et al., *A framework for Higgs characterisation*, Journal of High Energy Physics **2013** (2013).

[95] W. Buchmüller and D. Wyler, *Effective lagrangian analysis of new interactions and flavour conservation*, Nuclear Physics B **268**, 621 (1986).

[96] B. Grzadkowski, M. Iskrzyński, M. Misiak and J. Rosiek, *Dimension-six terms in the Standard Model Lagrangian*, Journal of High Energy Physics **2010** (2010).

[97] F. Boudjema, D. Guadagnoli, R. M. Godbole and K. A. Mohan, *Laboratory-frame observables for probing the top-Higgs boson interaction*, Physical Review D **92** (2015).

[98] J. F. Donoghue, *CP Violation and the Limits of the Standard Model*, in CP Violation and the Limits of the Standard Model (1995).

[99] A. Blum and A. Velasco, *The genesis of the CPT theorem*, The European Physical Journal H **47** (2022).

[100] S. Weinberg, *The Quantum Theory of Fields*, Vol. 1, Cambridge University Press (1995).

[101] G. C. Branco, L. Lavoura and J. P. Silva, *CP violation*, eng, International series of monographs on physics ARRAY(0x55a451674c60), Includes bibliographical references and index, Oxford [u.a.]: Clarendon Press (1999), XXI, 511 S.

[102] J. Brehmer, F. Kling, T. Plehn and T. M. P. Tait, *Better Higgs-CP tests through information geometry*, Phys. Rev. D **97**, 095017 (2018).

[103] T. Han and Y. Li, *Genuine CP-odd observables at the LHC*, Physics Letters B **683**, 278 (2010).

[104] D. Atwood, S. Bar-Shalom, G. Eilam and A. Soni, *CP violation in top physics*, Physics Reports **347**, 1 (2001).

[105] N. Mileo et al., *Pseudoscalar top-Higgs coupling: exploration of CP-odd observables to resolve the sign ambiguity*, Journal of High Energy Physics **2016** (2016).

[106] J. Ellis, D. S. Hwang, K. Sakurai and M. Takeuchi, *Disentangling Higgs-top couplings in associated production*, Journal of High Energy Physics **2014** (2014).

[107] T. Arens and L. M. Sehgal, *Energy correlation and asymmetry of secondary leptons in $e^+e^- \to t\bar{t}$*, Phys. Rev. D **50**, 4372 (1994).

[108] W. Bernreuther, A. Brandenburg, Z. Si and P. Uwer, *Top quark pair production and decay at hadron colliders*, Nuclear Physics B **690**, 81 (2004).

[109] M. Jeżabek, *Top quark physics*, Nuclear Physics B - Proceedings Supplements **37**, 197 (1994).

[110] J. F. Gunion and X.-G. He, *Determining the CP Nature of a Neutral Higgs Boson at the CERN Large Hadron Collider*, Physical Review Letters **76**, 4468 (1996).

[111] J. C. Collins and D. E. Soper, *Angular distribution of dileptons in high-energy hadron collisions*, Phys. Rev. D **16**, 2219 (1977).

[112] D. Gonçalves, J. H. Kim and K. Kong, *Probing the Top-Higgs Yukawa CP Structure in dileptonic $t\bar{t}h$ with M_2-Assisted Reconstruction*, Journal of High Energy Physics **2018** (2018).

[113] A. Collaboration, *CP Properties of Higgs Boson Interactions with Top Quarks in the $t\bar{t}H$ and tH Processes Using $H \to \gamma\gamma$ with the ATLAS Detector*, Physical Review Letters **125** (2020).

[114] C. Collaboration, *Measurements of $t\bar{t}H$ Production and the CP Structure of the Yukawa Interaction between the Higgs Boson and Top Quark in the Diphoton Decay Channel*, Physical Review Letters **125** (2020).

[115] J. Brod, U. Haisch and J. Zupan, *Constraints on CP-violating Higgs couplings to the third generation*, Journal of High Energy Physics **2013** (2013).

[116] R. D. Ball et al., *Parton distributions with QED corrections*, Nuclear Physics B **877**, 290 (2013).

[117] T. Sjöstrand, S. Mrenna and P. Skands, *A brief introduction to PYTHIA 8.1*, Computer Physics Communications **178**, 852 (2008).

[118] J. de Favereau et al., *DELPHES 3: a modular framework for fast simulation of a generic collider experiment*, Journal of High Energy Physics **2014** (2014).

[119] M. Bellagente et al., *How to GAN away detector effects*, SciPost Physics **8** (2020).

[120] A. Gretton et al., *A Kernel Two-Sample Test*, J. Mach. Learn. Res. **13**, 723 (2012).

[121] D. J. Rezende et al., *normalizing flows on tori and spheres* (2020), arXiv:2002.02428.

[122] A. Paszke et al., *PyTorch: An Imperative Style, High-Performance Deep Learning Library* (2019), arXiv:1912.01703.

[123] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization* (2017), arXiv:1412.6980.

[124] F. Pedregosa et al., *Scikit-learn: Machine Learning in Python*, Journal of Machine Learning Research **12**, 2825 (2011).

[125] C. E. Shannon, *A mathematical theory of communication.*, ACM SIGMOBILE Mob. Comput. Commun. Rev. **5**, 3 (2001).

[126] K. Pearson, *On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling*, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **50**, 157 (1900).

[127] J. Alcaraz Maestre, *Details on the Collins-Soper reference frame and lepton angular distributions in electroweak vector boson production at hadron colliders* (2020).

Erklärung:

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

Heidelberg, den 16. Juni 2023 .....