IOP Publishing Mach. Learn.: Sci. Technol. 6 (2025) 025061 https://doi.org/10.1088/2632-2153/addbc1

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
28 February 2025

REVISED
22 April 2025

ACCEPTED FOR PUBLICATION
21 May 2025

PUBLISHED
12 June 2025

PAPER

Flow annealed importance sampling bootstrap meets differentiable particle physics

Annalena Kofler[1,2,3], **Vincent Stimper**[1,4,5], **Mikhail Mikhasenko**[6,7], **Michael Kagan**[8] and **Lukas Heinrich**[3,*]

[1] Max Planck Institute for Intelligent Systems Tübingen, Max-Planck-Ring 4, 72076 Tübingen, Germany
[2] Max Planck Institute for Gravitational Physics (Albert Einstein Institute), Potsdam Science Park, Am Mühlenberg 1, 14476 Potsdam, Germany
[3] TUM School of Natural Sciences, Physics Department, Technical University of Munich, James-Frank-Str. 1, 85747 Garching, Germany
[4] Isomorphic Labs, 280 Bishopsgate, EC2M 4RB, London United Kingdom
[5] University of Cambridge, Department of Engineering, Trumpington Street, CB2 1PZ Cambridge, United Kingdom
[6] Institute for Experimental Physics I, Ruhr University Bochum, Universitätsstraße 150, 44801 Bochum, Germany
[7] Excellence Cluster ORIGINS, Boltzmannstr. 2, 85748 Garching, Germany
[8] SLAC National Accelerator Laboratory, 2575 Sand Hill Rd, Menlo Park, CA, 94025-7015, United States of America
[*] Author to whom any correspondence should be addressed.

E-mail: l.heinrich@tum.de

Keywords: normalizing flows, particle physics, differentiable programming, annealed importance sampling, generative modeling

Abstract

High-energy physics requires the generation of large numbers of simulated data samples from complex but analytically tractable distributions called matrix elements. Surrogate models, such as normalizing flows, are gaining popularity for this task due to their computational efficiency. We adopt an approach based on flow annealed importance sampling bootstrap (FAB) that evaluates the differentiable target density during training and helps avoid the costly generation of training data in advance. We show that FAB reaches higher sampling efficiency with fewer target evaluations in high dimensions in comparison to other methods.

1. Introduction

In the advent of the high-luminosity phase at the Large Hadron Collider (LHC), significant speed-ups in the simulation software are required to analyze the increasing amount of data [1–3]. The simulated data are compared to measured data from particle collisions to understand the underlying fundamental physics processes in more detail. One important step in the LHC simulation chain is the generation of samples ('events') based on *matrix elements* (MEs). MEs describe the dynamical information contained in particle interactions. The ME $\mathcal{M} = \langle p_1, \ldots, p_n | \mathcal{M} | p_a, p_b \rangle$ quantifies the transition from an initial state with momenta p_a and p_b to a final state with n outgoing particles described by their momenta $p_1, p_2, \ldots, p_n$. Using quantum field theory, MEs can be constructed by summing the contributions of all possible Feynman diagrams $\mathcal{M} = \mathcal{M}_1 + \mathcal{M}_2 + \mathcal{M}_3 + \ldots$ (also called 'channels'). Individual MEs can be calculated using the Feynman rules [4], which can become computationally expensive to evaluate when higher-order terms are included. The dimensionality of the ME depends on the number of outgoing particles n, each having three spatial degrees of freedom. Taking mass and momentum constraints into account, the dimensionality of MEs amounts to $3n - 4$. From a machine learning perspective, one can interpret MEs as unnormalized distributions $p(x)$ over the outgoing 4-momenta $\{p_1, \ldots, p_n\}$ which we will denote as x in the following to simplify notation. Via the Feynman rules, MEs can be evaluated analytically; however, sampling from them is hard since they can be high-dimensional for large n and multi-modal due to contributions from different channels. Additionally, they are defined on limited support originating from mass and momentum constraints and can exhibit divergences. It is possible to simplify the complicated multi-modal structure of MEs by decomposing the ME into its dominant channels, an approach referred to as multi-channeling [5].

Standard sampling algorithms such as `MadGraph` [6, 7], `SHERPA` [8], and `PYTHIA` [9, 10] rely on multi-channeling to reduce the complexity of the ME and employ adaptive Monte Carlo methods similar to

VEGAS [11] to approximate the distribution of the individual channels. More recently, machine learning-based surrogate models like normalizing flows have shown to improve the efficiency of the sampling process compared to standard methods like VEGAS and multiple approaches have been proposed over the last years. In general, normalizing flows can be trained using divergence measures as loss functions that quantify the difference between the target and the flow distribution: information about the target is either included by training with samples from the target or by evaluating the target distribution analytically. When training normalizing flows with the maximum-likelihood loss based on target samples [12], we have to rely on the costly generation of a large training data set. Therefore, approaches that evaluate the distribution of interest directly during training with samples from the flow have been the focus in event generation: Most papers employ neural importance sampling (NIS) [13] which has been adopted to phase space sampling and combined with multi-channeling in the works of [14–16]. Building on these developments, the efficiency of this approach can be improved by introducing a replay buffer and a VEGAS based initialization, as well as adapting the coupling between dimensions in the normalizing flow [17, 18]. NIS without multi-channeling has been employed in the work of [19] where they extend the target to include a background density and develop a specific training scheme to boost the efficiency at the beginning of training. To make previous results accessible to non-specialist users, a dedicated library has been developed recently [20].

In this work, we take into account that gradients need to be backpropagated through the ME distribution to update the flow parameters when evaluating the target with samples from the flow. Therefore, this training mode usually requires differentiable MEs, which have only recently been proposed [21] and employed [22] for normalizing flow training. A similar method, called flow annealed importance sampling bootstrap (FAB) [23], also relies on the evaluation of a differentiable target density and has been developed to obtain samples from Boltzmann distributions of molecules. FAB uses annealed IS (AIS) with Hamiltonian Monte Carlo (HMC) transition steps to improve the quality of the normalizing flow samples towards the distribution of interest. The resulting AIS samples and their weights are used to train the normalizing flow. Running HMC requires a differentiable target distribution, and we are the first to perform HMC-based updates on MEs. To compare the general performance of different training methods, we do not include domain-specific physics information via multi-channeling.

Contributions. In this work, (1) we adopt FAB [23] to event generation in the field of high-energy physics (HEP). (2) We compare FAB with an alternative density evaluation-based method using the reverse Kullback–Leibler divergence (rKLD) [24] as a loss function, as well as with sample-based maximum-likelihood training with the forward KL divergence (fKLD) loss. Finally, (3) we provide a detailed performance comparison between the methods based on the number of ME evaluations, as this step can be expensive.

2. Method

A normalizing flow [24–26] is a density estimator that consists of a series of learnable, invertible transformations that construct a differentiable bijection between a simple, chosen base distribution and an expressive flow distribution $q_\theta(x)$. To generate samples $x \sim q_\theta(x)$, samples are obtained from the base distribution and passed through the subsequent transformations. The density $q_\theta(x)$ can be evaluated for a given data point x by passing it through the inverse chain of transformations and evaluating the density of the base distribution.

2.1. Training

To optimize the flow parameters θ such that $q_\theta(x)$ matches the (unnormalized) target distribution $p(x)$ more closely, different forms of the KL divergence are utilized, resulting in two main approaches: normalizing flows can either be trained with available samples from the target distribution $x \sim p(x)$ (fKLD), or by evaluating the density $p(x)$ with samples from the flow $x \sim q_\theta(x)$ (rKLD and FAB).

fKLD. If samples $x \sim p(x)$ are cheaply available, the fKLD serves as a loss function:

$$D_{\mathrm{KL}}\left(p \parallel q_\theta\right) = \mathbb{E}_{x\sim p(x)}\left[\log \frac{p\left(x\right)}{q_\theta\left(x\right)}\right] .$$

It can be simplified to the negative log-likelihood loss

$$\mathcal{L}_{\mathrm{fKLD}} = -\mathbb{E}_{x\sim p(x)}\left[\log q_\theta\left(x\right)\right] = -\frac{1}{N}\sum_{i=1}^{N} \log q_\theta\left(x_i\right)$$

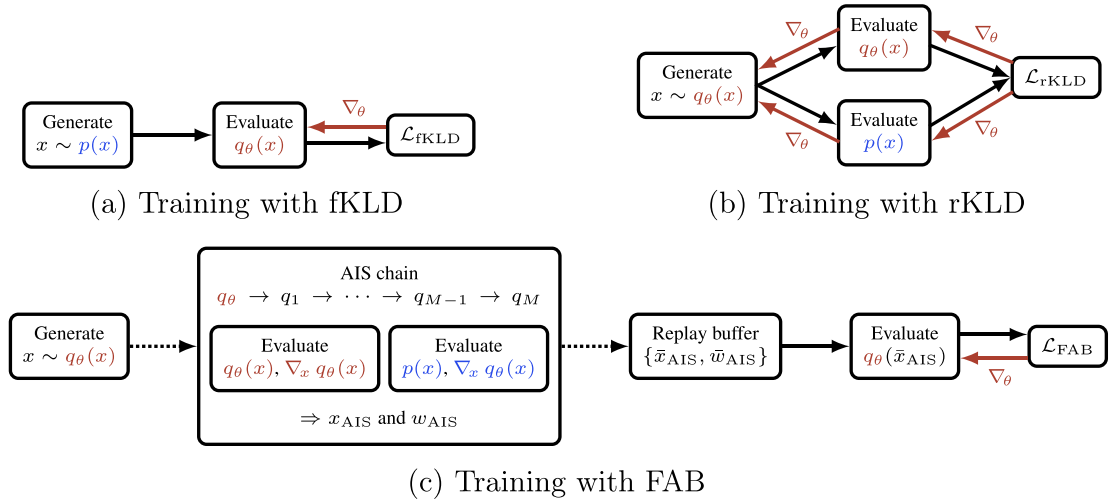


Figure 1. Visualization of compared methods. The black arrows illustrate the forward pass, while the red arrows mark the backpropagation of the gradient to update the flow parameters θ. For rKLD, the gradient propagation to the normalizing flow $q_\theta(x)$ (red) requires a differentiable target distribution $p(x)$ (blue). For FAB, the HMC updates compute the gradient of the distribution of interest, while the backpropagation of the gradient is stopped for the AIS samples $\bar{x}_{\mathrm{AIS}}$ and weights $\bar{w}_{\mathrm{AIS}}$. The dotted lines indicate that these steps do not occur when a gradient update is performed repeatedly with samples from the replay buffer.

over N data points where we drop terms independent of θ. It can be shown that this loss function results in a density q_θ with mass-covering properties [23]. The flow distribution $q_\theta(x)$ is evaluated with samples from $p(x)$ and the gradient computation is straightforward: $\nabla_\theta \mathcal{L}_{\mathrm{fKLD}} = -\frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log q_\theta(x_i)$.

rKLD. Another way of quantifying the difference between two distributions is via the rKLD,

$$D_{\mathrm{KL}}\left(q_\theta \parallel p\right) = \mathbb{E}_{x\sim q_\theta(x)}\left[\log \frac{q_\theta(x)}{p(x)}\right] .$$

Compared to fKLD, we sample from the normalizing flow and evaluate the flow density $q_\theta(x)$ as well as the target distribution $p(x)$ with the obtained samples. As a result, $p(x)$ has to be analytically available which is the case for MEs. This loss function has mode-seeking properties, meaning that it is not guaranteed that the optimized distribution $q_\theta(x)$ covers all modes of $p(x)$ [23]. To obtain a gradient for this loss function

$$\nabla_\theta \mathcal{L}_{\mathrm{rKLD}} = \nabla_\theta \mathbb{E}_{x\sim q_\theta(x)}\left[\log \frac{q_\theta(x)}{p(x)}\right] ,$$

the gradient has to be propagated through the evaluation of the target distribution $p(x)$, since the samples $x \sim q_\theta(x)$ used for the estimation of the expectation value depend on the parameters θ themselves. This is visualized in figure 1(b) for illustration. As a result, $p(x)$ has to be differentiable[9].

FAB. Both versions of the KL divergence are special cases of the α-divergence [27]

$$D_\alpha\left(p \parallel q_\theta\right) = -\frac{1}{\alpha(1-\alpha)} \int p(x)^\alpha \; q_\theta(x)^{1-\alpha} \; \mathrm{d}x \,, \tag{1}$$

where fKLD corresponds to $\alpha \to 1$ and rKLD to $\alpha \to 0$ [27]. In FAB [23], $D_{\alpha=2}$ is chosen as a loss function since it minimizes the variance of the importance weights $w = p(x)/q_\theta(x)$—a desirable property of a well-performing density estimator $q_\theta(x)$—and the resulting distribution $q_\theta(x)$ has mass covering properties with respect to $p(x)$ [27]. Since samples from the flow might poorly fit the target distribution at the beginning of training, AIS [28] is employed to pass the samples through a chain of intermediate distributions $q_1, \ldots, q_{M-1}$ with HMC as a transition operator. The intermediate distributions q_i are chosen to interpolate between the flow distribution $q_0 = q_\theta$ and the AIS target $q_M = p^2/q_\theta$. For all HMC steps between two intermediate AIS distributions, the flow as well as the target distribution have to be evaluated and the gradient of both distributions is required to perform momentum updates. As a result, we require a

[9] One can prevent differentiating through the target distribution by mapping the flow samples (with stopped gradient) back through the flow in the reverse direction and evaluating the loss function in latent space [17]. However, this approach requires two subsequent applications of the flow transformations and is more costly.

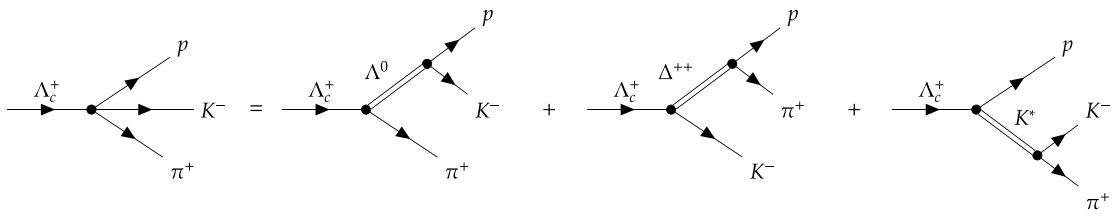


Figure 2. The $\Lambda_c^+ \to pK^-\pi^+$ decay (left-most diagram) consists of three decay channels characterized by their resonances Λ_0 (center-left), Δ^{++} (center-right), and K^* (right), each visualized in Feynman-like diagrams.

differentiable target distribution for FAB with HMC. Importance weights w_{AIS} can be computed for the AIS samples x_{AIS} which allow evaluating the surrogate loss function

$$\mathcal{S}(\theta) = -\mathbb{E}_{\text{AIS}}\left[\bar{w}_{\text{AIS}} \log q_\theta\left(\bar{x}_{\text{AIS}}\right)\right] ,$$

where the bar over variables indicates that the gradient is not propagated through the AIS sequence in the backward pass. To reduce the cost of evaluating the target distribution and its gradient for each intermediate distribution and each HMC step in between, the AIS samples can be stored in a replay buffer. Pairs $\{\bar{x}_{\text{AIS}}, \bar{w}_{\text{AIS}}\}$ are sampled from the buffer based on the importance weights to perform multiple gradient updates per iteration. Figure 1(c) shows a schematic illustration of FAB; for further details, see [23].

2.2. Differentiable MEs

The training approaches explored and compared in this work are not limited to one area of event generation in particle physics. They only require a differentiable implementation of the ME, since training with rKLD and FAB requires gradients of the target distribution. Recent developments in HEP provide us with differentiable implementations of complex amplitudes [21, 22, 29]. We choose two examples of high interest for the particle physics community, one from flavor physics and one from collider physics:

$\Lambda_c^+ \to pK^-\pi^+$. In flavor physics, the resonances of baryons are studied to search for new states beyond the constituent-quark model [30, 31]. One recently analyzed example is the Λ_c^+ baryon and its decay into a proton, a kaon, and a pion [32–34]. This decay is particularly interesting since its resonance structure allows searches of *CP* symmetry violations [33] and new physics [35, 36]. When the Λ_c^+ baryon decays, it can do so via different decay channels, resulting in the $K^-\pi^+$, pK^-, and $p\pi^+$ systems. These decays are shown in figure 2 as Feynman-like diagrams. Each of these systems decays via an intermediate particle that can have resonances which are shown in a Dalitz plot in figure 3. A Dalitz plot is a physics-specific visualization of 2D amplitudes where the axes are chosen such that the histograms can be interpreted as an unnormalized density [37]. The resonances appear in a Dalitz plot as vertical, horizontal, and diagonal structures. In figure 3, the vertical bands with two prominent lines correspond to the decay of $\Lambda \to pK^-$ resulting from the $\Lambda(1520)$ and the $\Lambda(1670)$ resonances. The significant horizontal band corresponds to the $K^*(892)$ meson based on $K^* \to K^-\pi^+$. Decays in $\Delta^{++} \to p\pi^+$ are visible on the diagonal of the Dalitz plot where the $\Delta^{++}(1232)$ resonance is notable. An example for interference effects among resonances of different decay channels is the horizontal $K^*(892)$ band that gets shifted when crossing the vertical line of the $\Lambda(1670)$ resonance. Destructive interference is visible at the upper corner of the Dalitz plot resulting from reciprocal influences of $\Lambda(1520)$ and higher-mass K^* resonances [33]. The complex resonance structure of this decay is ideally suited for our study as it provides a low-dimensional, but challenging setting to compare the performance of different normalizing flow training methods. Additionally, the Λ_c^+ amplitude model is implemented in a differentiable way and the code is publicly available via the `ComPWA` package [29].

For the implementation of the Λ_c^+ decays, `ComPWA` utilizes the `SymPy` engine [38] to formulate symbolic amplitude models whose compute graph can be transformed to `JAX` [39] for efficient gradient computation. Since `ComPWA` was developed as a general purpose tool for analytically formulating amplitudes, researchers can implement new amplitude models. The specific shape of the Dalitz plot boundary originates from mass and momentum constraints. It would be suboptimal to train a normalizing flow directly on the Dalitz plot representation because the it could assign non-zero probability density outside the physically-plausible regions. To prevent this, we map the support of the ME to the unit interval in each dimension via a differentiable and invertible transformation which is explained in appendix A. Similar transformations are applied in standard samplers [7, 8] and we employ rational-quadratic spline flows because they are explicitly designed for distributions with limited support. It is not possible to map the distribution to infinite support

since kinematic divergences can occur at the phase space boundary [4], which would result in non-zero probability mass at infinity.

$e^+e^- \to t\bar{t}, t \to W^+b, \bar{t} \to W^-\bar{b}$. In collider physics, the $t\bar{t}$ reaction producing a W boson and a bottom quark is repeatedly investigated due to the coupling of non-standard model processes to heavy elementary particles like the top quark [40–43]. We employ $t\bar{t}$ production with unstable final states containing W bosons from an electron positron collider as a challenging, higher dimensional example. We selected an e^+e^- process since ML methods in event generation are relevant for future circular or linear electron–positron colliders [44, 45]. The eight-dimensional $t\bar{t}$ ME can be generated with `MadJAX` [21] which provides a differentiable implementation of the scattering amplitudes of `MadGraph` [6, 7]. With `MadJAX`, we can generate any leading-order amplitudes with spin $0, 1/2$ or 1 particles and the code can be extended to higher spin MEs. It is also possible to generate MEs for proton–proton collisions by including a differentiable implementation of the parton density [22, 46] which we leave for future work. To map the irregular phase space boundaries to the unit hypercube, a differentiable non-linear transformation based on the `RAMBO` algorithm is applied to the `MadJAX` ME [47]. The eight dimensions characterizing the final state of the selected ME can be interpreted as two rescaled, intermediate masses $\widetilde{M}_1$ and $\widetilde{M}_2$, as well as two angular variables $\cos\widetilde{\theta}_k$ and $\widetilde{\phi}_k$ with $k \in [1, 2, 3]$.

To summarize, both selected examples are based on libraries which allow users to generate differentiable implementations of arbitrary MEs. This means that the compared approaches of training normalizing flows can easily be applied to different MEs.

3. Experimental setup

Training settings. For each of the aforementioned MEs and training methods, we train three models with different random seeds and average their results to improve reliability. For details about the flow hyperparameters and train settings, we refer to appendix B for details.

Baseline. We compare our results with a physics-agnostic integral estimation method called `VEGAS+` [11, 48]. This grid-based optimization method divides the support into a regular, rectangular grid and estimates the integral contribution of each subspace. With this information, the grid is iteratively updated to focus on regions with large contributions and the value for the integral estimate is calculated as a weighted average over multiple runs. Through a combination of stratified sampling and IS, `VEGAS+` is fast and efficient and can account for correlations between dimensions. We describe the `VEGAS+` specific hyperparameters and tuning procedure in detail in appendix B. It is not possible to use `MadGraph` as a baseline for both reactions since it is tailored to collider physics and the complicated helicity structure of the Λ_c^+ decay is not implemented.

Performance metrics. We first provide qualitative performance comparisons for both MEs by showing histograms and a corner plot of samples obtained from `VEGAS+` as well as the normalizing flows. For the Λ_c^+ ME, we compare the results to samples obtained by rejection sampling with a uniform proposal distribution. For the $t\bar{t}$ reaction, we include the samples obtained from `MadGraph` as a reference. Both sets of samples correspond to the training data sets employed for fKLD training. Secondly, we evaluate the quantitative performance of the normalizing flows based on three metrics: (1) the fKLD evaluated on a test data set, (2) the IS efficiency calculated from the importance weights, and (3) the integral estimate of $p(x)$. The fKLD, introduced in section 2, quantifies the difference between the learned distribution $q_\theta(x)$ and the true target distribution $p(x)$ by evaluating the normalizing flow with the samples $x \sim p(x)$. If both distributions match almost everywhere, the KL divergence is zero.

The IS efficiency [49, 50] can be computed with samples from the normalizing flow $x_i \sim q_\theta(x_i)$ and their importance weights $w_i = p(x_i)/q_\theta(x_i)$ as

$$\epsilon = \frac{1}{N}\left[\sum_{i=1}^{N} w_i\right]^2 \left[\sum_{i=1}^{N} w_i^2\right]^{-1} . \tag{2}$$

We elaborate on its derivation and similarities to the unweighting efficiency—which is commonly employed in HEP—in appendix C. The integral estimate $\bar{I}$ of the target distribution $p(x)$ is defined as

$$\bar{I} = \int p(x)\, \mathrm{d}x = \int q_\theta(x)\, \frac{p(x)}{q_\theta(x)}\, \mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^{N} w_i \, . \tag{3}$$

For the 2D example, the sampling efficiency ϵ and the integral estimate $\bar{I}$ are calculated for 10^4 flow samples, while 10^6 flow samples are used for the 8D ME. We compare our results for $\bar{I}$ with `VEGAS+`. The performance metrics are summarized in table 1 for both MEs. To make rKLD and FAB comparable, we report results

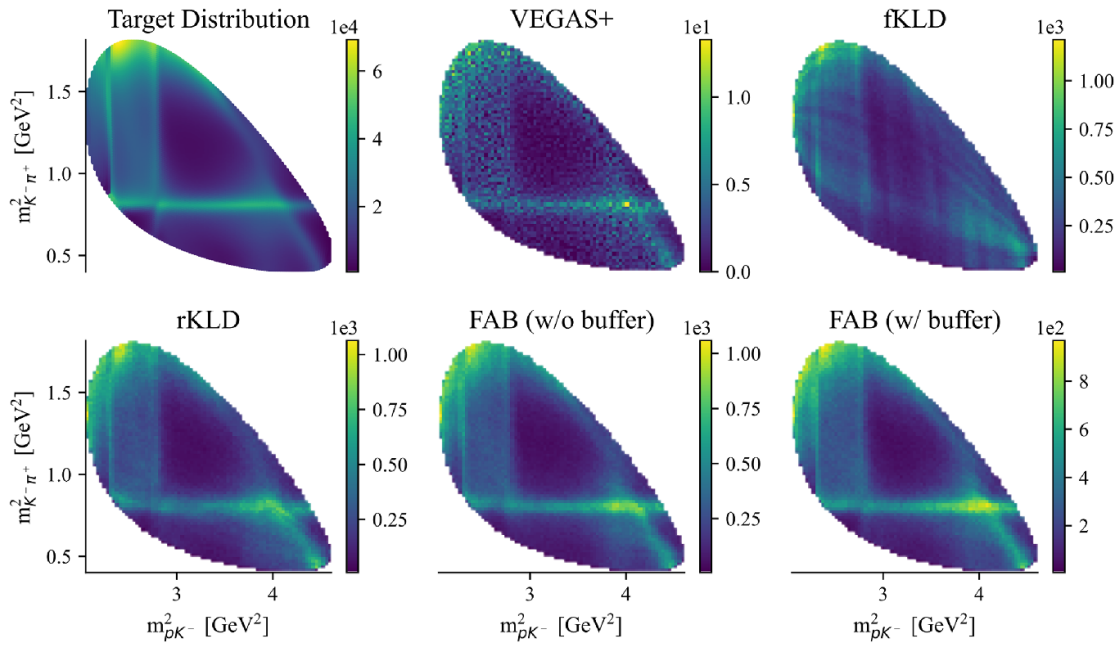


Figure 3. Comparison of the target density for the $\Lambda_c^+ \to pK^-\pi^+$ matrix element with histograms based on samples from `VEGAS+` and the best normalizing flow for each method.

based on training runs with the same number of target evaluations. Since the target distribution can be costly to evaluate, we additionally show the trend for the IS efficiency ϵ as a function of the number of target evaluations. We do not include fKLD training in this consideration because it relies on a pre-computed training data set on which the normalizing flow can potentially be trained for an arbitrary number of epochs. For rKLD, the ME is calculated once for each batch of flow samples, while the target density and its gradient need to be evaluated for every HMC step between intermediate AIS distributions for FAB. To reduce the number of target evaluations, FAB allows multiple gradient updates per iteration with samples from the replay buffer. However, depending on the buffer size, more target evaluations might be required at the beginning of training to fill up the buffer.

4. Results and discussion

$\Lambda_c^+ \to pK^-\pi^+$. To illustrate the complexity of the two-dimensional ME, we show the target density evaluated on a grid as a Dalitz plot in figure 3. We provide histograms of 10^6 unweighted samples obtained from the best performing normalizing flow for each investigated method to compare the sampling quality. This can qualitatively be compared to $\sim 2 \cdot 10^4$ samples and their weights obtained from a converged `VEGAS+` integrator. For fKLD, we observe that the horizontal line corresponding to the $K^*(892)$ resonance is not clearly visible compared to the results of `VEGAS+`, rKLD, and FAB. Furthermore, the crossing of the horizontal and diagonal lines appears as a blurred region for fKLD. This observation is consistent with figure 4, where we provide marginal histograms of the invariant mass distribution for $m^2_{pK^-}$ and $m^2_{K^-\pi^+}$. We average the results for three models trained with different random seeds and visualize the standard deviation per histogram bin. Overall, the binned samples obtained from the normalizing flows trained with fKLD show the largest standard deviations from samples obtained with rejection sampling. Flows trained with rKLD and FAB produce histograms that exhibit all features of the ME structure and provide the best results.

When training with the same number of target evaluations, rKLD and FAB with the replay buffer provide the best results with the highest IS efficiency and a distribution that is most similar to the test data set (see table 1). Additionally, the integral estimate computed with samples from the flow has the lowest standard deviations. The change in the sampling efficiency when increasing the number of target evaluations is shown in figure 5(a). We can observe that ϵ improves for rKLD more rapidly than for FAB w/o buffer. All normalizing flow models show the same IS efficiency of approximately 70% for a low number of target evaluations. This corresponds to the efficiency obtained from rejection sampling with a uniform proposal distribution (shown as a black, dashed line in figure 5(a)). This behavior can be explained by the fact that the normalizing flows have a uniform base distribution and are initialized with an identity mapping, resulting in a uniform distribution at the beginning of training. For FAB w/ buffer, samples have to be generated before

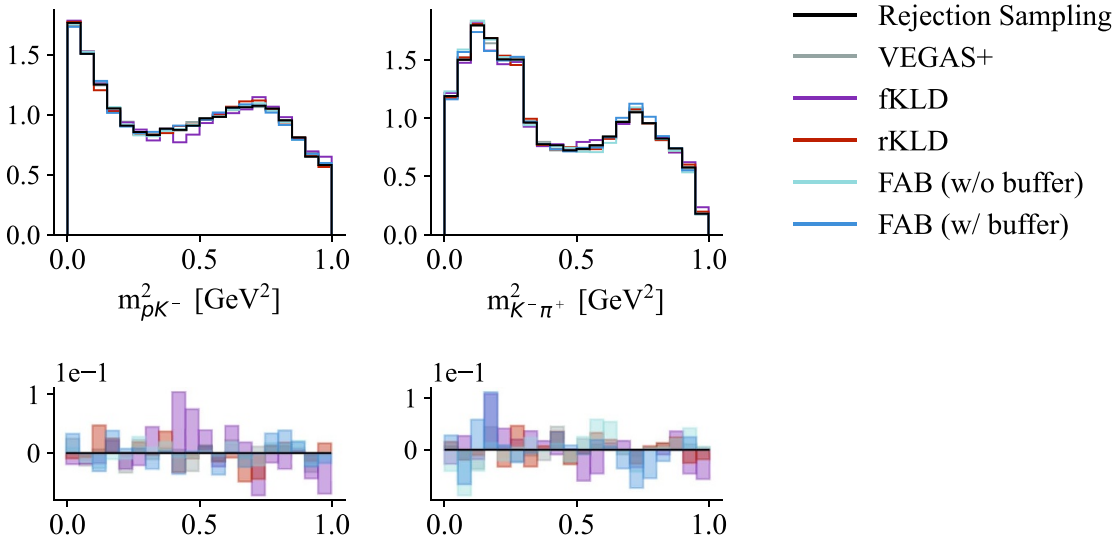
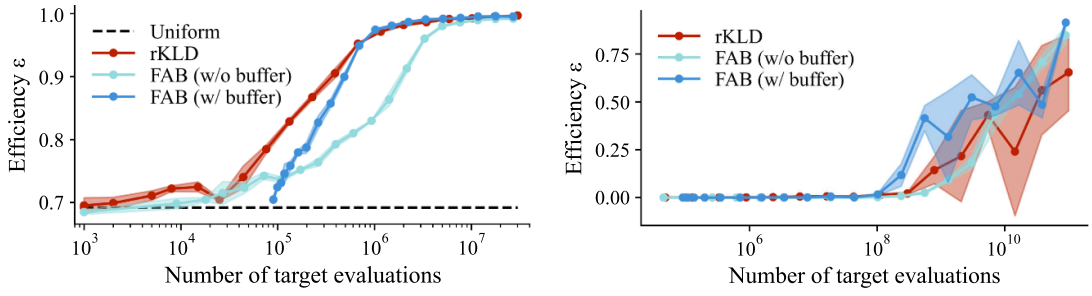


Figure 4. Normalized marginal histograms of the invariant masses for the $\Lambda_c^+ \to pK^-\pi^+$ matrix element based on samples from rejection sampling, `VEGAS+`, and the best normalizing flow for each method. The rejection sampling result serves as a baseline to visualize the deviation.

Table 1. Overview of performance metrics for the different MEs and methods.

	$\Lambda_c^+ \to pK^-\pi^+$			$e^+e^- \to t\bar{t}, t \to W^+b, \bar{t} \to W^-\bar{b}$		
	$\mathbb{E}_p[\log(p/q_\theta)] \downarrow$	$\epsilon(\%) \uparrow$	$\bar{I}$	$\mathbb{E}_p[\log(p/q_\theta)] \downarrow$	$\epsilon(\%) \uparrow$	$\bar{I}$
`VEGAS+`	—	67.52 ± 0.21	8926 ± 2	—	0.02 ± 0.01	1761 ± 309
fKLD	9.1581 ± 0.0011	87.02 ± 0.08	8925 ± 3	8.35 ± 0.16	1.75 ± 1.26	2116 ± 9
rKLD	9.0978 ± 0.0005	99.67 ± 0.01	8924 ± 4	7.74 ± 0.02	56.51 ± 40.14	2267 ± 88
FAB (w/o buffer)	9.1009 ± 0.0003	99.26 ± 0.08	8912 ± 7	7.79 ± 0.03	84.25 ± 4.51	2208 ± 1
FAB (w/ buffer)	9.0988 ± 0.0005	99.56 ± 0.05	8911 ± 2	7.747 ± 0.002	90.59 ± 0.01	2207.0 ± 0.1



(a) $\Lambda_c^+ \to pK^-\pi^+$ (b) $e^+e^- \to t\bar{t}, t \to W^+b, \bar{t} \to W^-\bar{b}$.

Figure 5. Importance sampling efficiency depending on the number of target evaluations required during training.

the start of training to fill the replay buffer, which results in an offset in the number of target evaluations. During training, the normalizing flow parameters are updated by sampling from the prioritized replay buffer based on $\bar{w}_{\text{AIS}}$, which aids the model significantly and the IS efficiency improves at a faster rate compared to the flows trained with rKLD. Since FAB evaluates the target several times in each iteration dependent on the chosen number of HMC steps and intermediate AIS distributions, it can benefit from additional information about the structure of the ME. Overall, flows trained with rKLD and both FAB approaches have no problem in modeling the complex 2D distribution.

$e^+e^- \to t\bar{t}, t \to W^+b, \bar{t} \to W^-\bar{b}$. We compare the corner plots of 10^6 flow samples for each method to samples from the training data set generated with `MadGraph` in figure 6. The latter serve as a ground truth and illustrate challenging properties of MEs: peaks at the boundary and correlations between dimensions are difficult for normalizing flows. Here, we visualize samples from the flow directly on the unit hypercube and

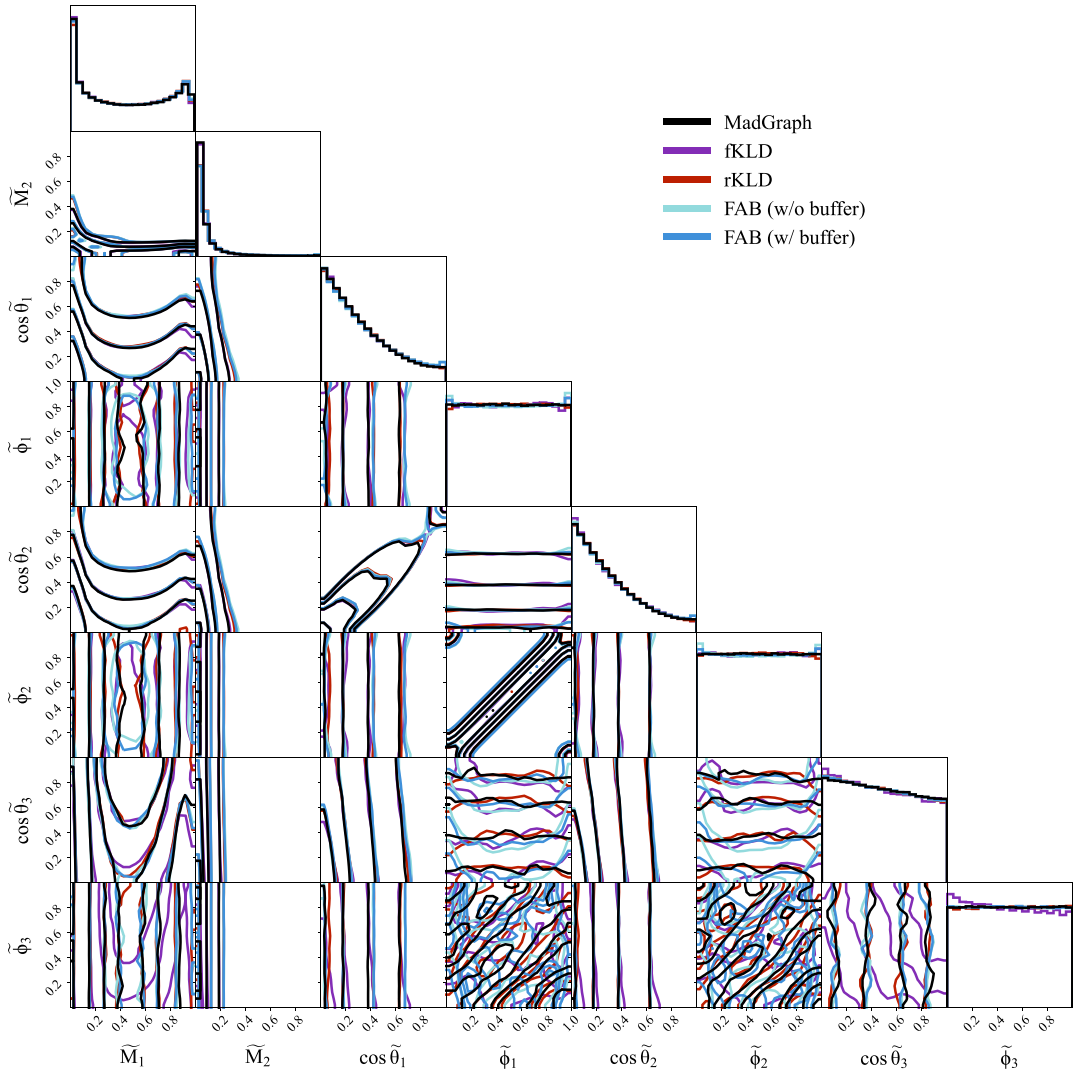


Figure 6. Corner plot with samples from `MadGraph` and from the best normalizing flows for each method.

one would need to apply the inverse `RAMBO` transformation [47] to obtain physical information for each outgoing particle. We observe that the normalizing flows especially deviate from the `MadGraph` distribution close to the phase space boundary which can be observed for all investigated methods. Correlations between dimensions, for example between the angles $\widetilde{\phi}_1 - \widetilde{\phi}_3$ as well as $\widetilde{\phi}_2 - \widetilde{\phi}_3$, appear to be challenging. While we report the performance metrics based on $\sim 10^5$ samples from an optimized `VEGAS+` integrator in table 1, we do not include the results in the corner plot since we observe large deviations explained by low efficiencies. When considering the number of target evaluations in figure 5(b), the flow benefits from the AIS procedure and the sampling of batches from the buffer dependent on the AIS importance weights. Samples from regions where the flow is a poor approximation of the target have a high weight and are predominantly used in gradient updates, resulting in a significantly higher IS efficiency with fewer target evaluations. Therefore, FAB w/ buffer reaches an efficiency of approximately 40 % with an order of magnitude fewer samples than rKLD and FAB without the buffer. We observe that one of the three training runs of FAB w/ buffer diverged and exclude it from figure 5(b) and the performance evaluation in table 1.

5. Summary and conclusion

We have transferred FAB [23], which utilizes AIS with HMC as a transition operator, from molecular configuration modeling to HEP, building on recently introduced, differentiable implementations of MEs. We have demonstrated that training FAB with a prioritized replay buffer is a promising approach for improving the efficiency for event generation, since passing the flow samples through an AIS chain guides training at early stages. In the future, we plan to scale this approach to more complex particle-interaction processes and

assess the performance improvement in greater detail. Including information about the individual ME contributions via multi-channeling will likely lead to further performance improvements [14, 17, 18]. Additionally, we plan to extend this conceptual work with differentiable implementations of the parton density [22, 46] to be applicable to proton–proton collisions. Overall, this work has the potential to improve the quality and speed of sampling methods employed at the LHC and facilitate the efficient analysis of high-luminosity events.

Data availability statement

The data required to reproduce findings of this study are openly available at the following URL/DOI: https://doi.org/10.17617/3.UZ786R.

Code availability statement

The code is available on GitHub at the following URL: https://github.com/annalena-k/FAB-meets-diffME.

Acknowledgments

The authors thank the reviewers of MLST and the NeurIPS workshop *Machine Learning and the Physical Sciences, 2024* for their helpful suggestions and interesting questions which helped to improve the manuscript. AK thanks Timothy Gebhard for extensive support in reviewing and formatting the draft as well as Ludwig Burger and Nicole Hartman for proof-reading and correcting the manuscript. The authors thank Matthew Feickert for help with running `MadGraph`. AK thanks Peter Lepage for answering questions about running `VEGAS+`. The computational work described in this manuscript was performed on the RAVEN cluster of the Max Planck Computing and Data Facility (MPCDF). We thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting AK. MK is supported by the US Department of Energy (DOE) under Grant DE-AC02-76SF00515. LH is supported by the Excellence Cluster ORIGINS, which is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy—EXC-2094-390783311. We also thank the Munich Institute for Astro-, Particle and BioPhysics (MIAPbP) which is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy—EXC-2094 - 390783311, as this work was partially performed at the MIAPbP workshop on Differentiable and Probabilistic Programming for Fundamental Physics.

Used software: This work has made use of many open-source Python packages, including `blackjax` [51], `ComPWA` [29], `corner` [52], `distrax` [53], `haiku` [54], `JAX` [39], `FAB-JAX` [23], `MadJAX` [21], `MadGraph` [7], `matplotlib` [55], `numpy` [56], `pylhe` [57] `SymPy` [38], `TensorWaves` [58], and `VEGAS+` [11, 48]. The accessible color schemes in our figures are based on [59].

Appendix A. Transformation of Dalitz plot to unit hypercube

While a Dalitz plot is the ideal visualization of the outgoing particles of a three-particle decay from a physics perspective, it is not ideal for normalizing flow training since the flow has to learn the support of the distribution. To prevent the normalizing flows from assigning non-zero probability density outside the phase space boundary, we map the Dalitz plot coordinates $(m^2_{pK^-}, m^2_{K^-\pi^+})$ to the unit hypercube with a differentiable and invertible transformation. First, we convert the Dalitz plot into the square Dalitz plot by transforming one of the invariant masses, e.g. $m^2_{K^-\pi^+}$ to the helicity polar angle $\cos\theta_{K^-\pi^+}$ [37]. The mapping between the two variables is defined as

$$\begin{aligned} m^2_{K^-\pi^+} = {} & m^2_{\pi^+} + m^2_{K^-} \\ & + \frac{1}{2m^2_{pK^-}} \left(m^2_{\Lambda_c^+} - m^2_{pK^-} - m^2_{\pi^+}\right) \left(m^2_{pK^-} + m^2_{K^-} - m^2_p\right) \\ & - \frac{1}{2m^2_{pK^-}} \cos\theta_{K^-\pi^+} \cdot \lambda^{\frac{1}{2}}\left(m^2_{\Lambda_c^+}, m^2_{pK^-}, m^2_{\pi^+}\right) \cdot \lambda^{\frac{1}{2}}\left(m^2_{pK^-}, m^2_{K^-}, m^2_p\right) , \end{aligned}$$

where the masses m_{π^+}, m_{K^-}, and m_p are constants and the kinematic Källén function is defined as

$$\lambda(x,y,z) = x^2 + y^2 + z^2 - 2xy - 2xz - 2yz\,.$$

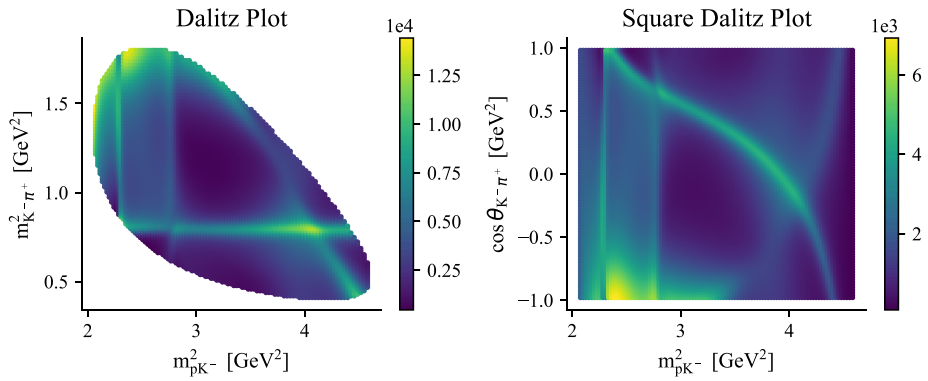


Figure A1. Dalitz plot of $\Lambda_c^+ \to pK^-\pi^+$ decay and transformation to square Dalitz plot.

After applying the non-linear transformation from $(m^2_{pK^-}, m^2_{K^-\pi^+})$ to $(m^2_{pK^-}, \cos\theta_{K^-\pi^+})$, we can visualize the resulting distribution in a square Dalitz plot in figure A1. We can observe that the previously horizontal line gets mapped into an almost diagonal line.

Appendix B. Hyperparameters

Data generation and training. The training data for fKLD consists of 10^4 samples for $\Lambda_c^+ \to pK^-\pi^+$ generated with rejection sampling and of 10^6 samples for $e^+e^- \to t\bar{t}$, $t \to W^+b$, $\bar{t} \to W^-\bar{b}$ produced with `MadGraph` [7] with a center of mass energy of 1 TeV. The test data sets are generated equivalently. The dimensionality of the resulting ME is defined by the number of degrees of freedom as $3n-4$ which we explain for illustration for the eight-dimensional case: each of the four outgoing particles is defined by its energy and three-dimensional momentum vector, resulting in 16 overall degrees of freedom. Conservation of energy and momentum reduces this number by four dimensions. Additionally, we know the masses of each stable outgoing particle, resulting in a $12-4=8$ dimensional phase space. For fKLD, the normalizing flows are trained with these datasets for 20 epochs in 2D and for 200 epochs in 8D. For rKLD, we train for 3×10^4 iterations (2D) and for 10^8 iterations (8D). The values for FAB were chosen based on the FAB hyperparameters such that we perform the same number of target evaluations during training compared to rKLD, resulting in 3×10^3 iterations (2D) and 10^7 iterations (8D). All compared models are trained on Nvidia A100 GPUs.

Normalizing flow. We use normalizing flows based on coupling layers and rational-quadratic spline transformations [60] implemented in `distrax` [53] and `haiku` [54]. For the base distribution, we choose a uniform distribution over the unit hypercube. Since changes in the normalizing flow hyperparameters affect the expressivity of the density estimator equally for all investigated methods, we perform hyperparameter tuning only for rKLD training because it has the shortest run time. We only tune the most important hyperparameters related to the architecture of the normalizing flow since this is computationally costly. For $\Lambda_c^+ \to pK^-\pi^+$, we subsequently vary the number of spline bins $n_\mathrm{b} \in [4,6,8,10,12,14,16]$, the number of flow transformations $n_\mathrm{t} \in [4,6,8,10,12,14,16]$, and the number of neurons per hidden layer $n_\mathrm{n} \in [10,50,100,150,200,250,300]$ for the fully connected conditioner network with two layers. For the 8D ME, we compare loss values and efficiencies for $n_\mathrm{b}, n_\mathrm{t} \in [6,8,10,12,14]$, and $n_\mathrm{n} \in [200,250,300,350,400]$. Considering the trade-off between expressivity and increase in optimization time, we select the following values based on the final training loss, validation loss, and efficiency: For $\Lambda_c^+ \to pK^-\pi^+$, the number of bins and transformations is $n_\mathrm{t} = n_\mathrm{b} = 10$, and a conditioner network with two hidden layers and 100 neurons each is used. For the 8D ME, we set $n_\mathrm{b} = n_\mathrm{t} = 14$, and use 400 neurons for each of the two hidden layers.

FAB. Additional hyperparameters have to be selected for FAB. We use two (linearly spaced) intermediate distributions M in the AIS sequence with a HMC transition operator containing a single iteration and three leap frog steps. The initial HMC acceptance rate is set to $p_\mathrm{acc} = 0.65$ and is tuned dependent on the number of actually accepted samples. Furthermore, we have to specify the number of gradient updates per iteration $L = 4$ (2D) and $L = 2$ (8D) in the case of buffered training. The most important variable in hyperparameter optimization is the HMC step size, since it depends on the support of the target distribution. To obtain a suitable estimate, we start one FAB run with an arbitrary step size and observe how the value is adjusted during training. We adopt the converged values of $l_\mathrm{init} = 0.05$ for the two-dimensional and

$l_{\text{init}} = 0.005$ for the eight-dimensional ME for all subsequent runs. We do not optimize the size of the FAB replay buffer since we do not expect a significant influence on the performance. We only take into account that the replay buffer should be sufficiently large such that sampling data points from the buffer is informative for normalizing flow training. Therefore, we set the minimal size of the buffer to 10 times the batch size (i.e. 10^4) and the maximal buffer size to 100 times the batch size (i.e. 10^5).

Optimization. We use the `Adam` [61] optimizer with a learning rate of 3×10^{-4} and train with a batch size of 10^3. We employ the gradient clipping scheme developed for FAB in all our runs, where we dynamically clip the gradient norm to 20 times the median of the last 100 gradient values and ignore very large gradients that are a factor 20 times larger than this median value [23]. While it is not necessary to use a scheduler in the 2D case, we employ a warm-up and cosine decay learning rate schedule for the eight-dimensional ME. We train with an initial and final learning rate of 10^{-5} as well as a peak learning rate of 3×10^{-4} which is reached after 10 epochs (for fKLD) and after 10^3 iterations (for rKLD and FAB).

Baseline. We compare our results with the physics-agnostic integral estimation method `VEGAS+` [11, 48]. This grid-based optimization method subdivides the support into a regular, rectangular grid and estimates the integral contribution of each subspace. With this information, the grid is iteratively updated to focus on regions with large contributions and the value for the integral estimate is calculated as a weighted average over multiple runs. Through a combination of stratified sampling and IS, `VEGAS+` is fast and efficient and can account for correlations between dimensions. For the 2D ME, we choose a `VEGAS+` grid with 64 bins in each dimension, a damping factor of $\alpha = 0.5$, two warm-up iterations with 10^3 evaluations per iteration, followed by 8 iterations with $2 \cdot 10^5$ evaluations per iteration for the integral estimates. For the 8D ME, we double the grid to 128 bins per dimension, keep the damping factor, and increase the number of evaluations to 10^4 for each of the two warm-up iterations. The integral estimate is obtained from 8 iterations with 10^5 evaluations. We optimized these hyperparameters to the best of our knowledge. To make sure that `VEGAS+` converged, we perform checks like increasing the number of evaluations by a factor of 10 per iteration which provide stable results and do not show significant deviations in the integral estimate for both examples. However, the low IS efficiency and large deviation on the integral estimate for the 8D ME (cf, table 1) indicate that `VEGAS+` struggles to adapt to the distribution. It is important to note that the results are obtained without multi-channeling since this physics information based on the contributing MEs is not provided to the normalizing flows either. Multi-channeling would lead to a performance improvement for both `VEGAS+` as well as flow-based methods [17, 18]. Although the `VEGAS+` optimization is significantly faster than training a normalizing flow, the flexibility of the latter results in higher IS efficiencies. Since the main goal of training ME surrogates is to provide an optimally pre-trained model [18], the training time is amortized when generating a large number of events.

Appendix C. Relationship of sampling efficiency and unweighting efficiency

IS efficiency. The (importance) sampling efficiency ϵ (cf 2) can be derived from the effective sample size *ESS* which allows the performance comparison of different Monte Carlo methods like Markov Chain Monte Carlo (MCMC) or IS based on a set of weighted samples. If we draw N samples from a less-than-ideal MCMC or IS proposal distribution $q(x)$, the *ESS* indicates the number of independent samples that these would be equivalent to if drawn directly from the target distribution $p(x)$. Therefore, the effective sample size can be defined proportional to the ratio of the variance of an ideal MC estimator (i.e. sampling from the target p) and the variance of the less-than-ideal MCMC or IS estimator (i.e. sampling from the proposal q) [49]. Through derivations outlined in [50], the ESS can be related to the variance of the importance weights via

$$\text{ESS} = \frac{N}{1 + \text{Var}_{q(x)}[w]},$$

and to the estimate

$$\widehat{\text{ESS}} = N \frac{\left(\frac{1}{N}\sum_{i=1}^{N} w_i\right)^2}{\frac{1}{N}\sum_{i=1}^{N} w_i^2} = \frac{1}{\sum_{i=1}^{N} \bar{w}_i^2}, \tag{C.1}$$

with the normalized importance weights

$$\bar{w}_i = \frac{w_i}{\sum_{j=1}^{N} w_j}.$$

The IS efficiency ϵ as defined in equation 2 corresponds to the normalization of equation C.1.

Table C1. Comparison of importance sampling and unweighting efficiency for the different MEs and methods.

	$\Lambda_c^+ \to pK^-\pi^+$		$e^+e^- \to t\bar{t}, t \to W^+b, \bar{t} \to W^-\bar{b}$	
	$\epsilon(\%) \uparrow$	$\epsilon_{\text{uw}}(\%) \uparrow$	$\epsilon(\%) \uparrow$	$\epsilon_{\text{uw}}(\%) \uparrow$
`VEGAS+`	67.52 ± 0.21	16.30 ± 0.42	0.02 ± 0.01	<0.01
fKLD	87.02 ± 0.08	22.50 ± 0.37	1.75 ± 1.26	0.02 ± 0.01
rKLD	99.67 ± 0.01	75.42 ± 1.87	56.52 ± 40.14	0.29 ± 0.20
FAB (w/o buffer)	99.26 ± 0.08	67.59 ± 1.38	84.25 ± 4.51	1.15 ± 0.27
FAB (w/ buffer)	99.56 ± 0.05	68.69 ± 0.89	90.59 ± 0.01	0.67 ± 0.07

Unweighting efficiency. The unweighting efficiency stems from the approach of refining samples obtained from the less-than-ideal proposal distribution $q(x)$ by keeping only a fraction of the samples in proportion to the ratio of the target distribution $p(x)$ and the proposal $q(x)$ [62]. The so-called *raw weight* $w_i = \frac{p(x_i)}{q(x_i)}$ corresponds to the importance weight in the MCMC and IS setting. The definition of the unweighting efficiency [14, 62, 63] as

$$\epsilon_{\text{uw}} = \frac{\frac{1}{N}\sum_{i=1}^{N} w_i}{w_{\text{max}}} \tag{C.2}$$

can be motivated in the following: in regions where the proposal distribution q overestimates the target (i.e. $w_i < 1$), only a fraction of the original samples proportional to w_i should be retained. However, in regions where the proposal underestimates the target, it is not possible to generate additional samples to match the target's density. Therefore, all available samples in those regions are kept which has to be compensated by reducing the retained fraction in overestimated regions. Such an adjustment maintains the correct relative shape of the distribution [62]. This explanation is equivalent to applying rejection sampling where the probability to keep or reject a sample is defined as the raw weight normalized by the (pre-computed) maximal weight in the integration volume $w_{\text{rel}} = w_i / w_{\text{max}}$. A sample is retained if a uniformly sampled random number R is smaller than w_{rel} [63, 64]. Finally, the unweighting efficiency can be computed as the average raw weights rescaled by w_{max} [14, 63].

Relationship of efficiencies. In [49], different formulations of the generalized effective sample size are explored based on a set of required and desirable conditions. They conclude that defining the *ESS* as $1/\sum_{i=1}^{N} \bar{w}_i^2$ (related to ϵ) and $1/\max(\bar{w}_1, \ldots, \bar{w}_N)$ (related to ϵ_{uw}) are proper and stable formulations. While both efficiencies suffer from large outlier weights, the unweighting efficiency is directly affected through w_{max} in the denominator as reported in [14, 16]. For this reason, we choose to report the performance based on the IS efficiency in the main body of this work and provide additional estimates of the unweighting efficiency in table C1. We do not apply bootstrap techniques that are designed to mitigate large outliers in the weight distribution [16].

ORCID iDs

Annalena Kofler https://orcid.org/0009-0008-5938-6215
Vincent Stimper https://orcid.org/0000-0002-4965-4297
Mikhail Mikhasenko https://orcid.org/0000-0002-6969-2063
Michael Kagan https://orcid.org/0000-0002-3386-6869
Lukas Heinrich https://orcid.org/0000-0002-4048-7584

References

[1] Aberle O *et al* 2020 High-luminosity large hadron collider (HL-LHC): technical design report *CERN Yellow Reports: Monographs* (CERN) (available at: https://cds.cern.ch/record/2749422)
[2] Software C O and Computing 2022 CMS phase-2 computing model: update document *Technical Report* (CERN Geneva) (available at: https://cds.cern.ch/record/2815292)
[3] (Collaboration A (ATLAS)) 2022 ATLAS software and computing HL-LHC roadmap *Technical Report* (CERN Geneva) (available at: https://cds.cern.ch/record/2802918)
[4] Griffiths D 1987 *The Feynman Calculus* (Wiley, Ltd) ch 6, pp 189–212
[5] Kleiss R and Pittau R 1994 *Comput. Phys. Commun.* **83** 141–6
[6] Alwall J, Herquet M, Maltoni F, Mattelaer O and Stelzer T 2011 *J. High Energy Phys.* JHEP06(2011)128
[7] Alwall J, Frederix R, Frixione S, Hirschi V, Maltoni F, Mattelaer O, Shao H S, Stelzer T, Torrielli P and Zaro M 2014 *J. High Energy Phys.* JHEP07(2014)079
[8] Bothmann E *et al* (Sherpa) 2019 *SciPost Phys.* **7** 034
[9] Bierlich C *et al* 2022 *SciPost Phys. Codebases* **8**

[10] Bierlich C et al 2022 SciPost Phys. Codebases 8–r8.3
[11] Lepage P G 1978 J. Comput. Phys. 27 192–203
[12] Stienen B and Verheyen R 2021 SciPost Phys. 10 038
[13] Müller T, Mcwilliams B, Rousselle F, Gross M and Novák J 2019 ACM Trans. Graph. 38 1–9
[14] Bothmann E, Janßen T, Knobbe M, Schmale T and Schumann S 2020 SciPost Phys. 8 069
[15] Gao C, Isaacson J and Krause C 2020 Mach. Learn.: Sci. Technol. 1 045023
[16] Gao C, Höche S, Isaacson J, Krause C and Schulz H 2020 Phys. Rev. D 101 076002
[17] Heimel T, Winterhalder R, Butter A, Isaacson J, Krause C, Maltoni F, Mattelaer O and Plehn T 2023 SciPost Phys. 15 141
[18] Heimel T, Huetsch N, Maltoni F, Mattelaer O, Plehn T and Winterhalder R 2024 SciPost Phys. 17 023
[19] Pina-Otey S, Sánchez F, Lux T and Gaitan V 2020 Phys. Rev. D 102 013003
[20] Deutschmann N and Götz N 2024 J. High Energy Phys. JHEP03(2024)083
[21] Heinrich L and Kagan M 2023 J. Phys.: Conf. Ser. 2438 012137
[22] Heimel T, Mattelaer O, Plehn T and Winterhalder R 2025 SciPost Phys. 18 017
[23] Midgley L I et al 2023 Flow annealed importance sampling bootstrap The 11th Int. Conf. on Learning Representations (https://openreview.net/forum?id = XCTVFJwS9LJ)
[24] Papamakarios G et al 2021 J. Mach. Learn. Res. 22 1–64
[25] Tabak E G and Vanden-Eijnden E 2010 Commun. Math. Sci. 8 217–33
[26] Rezende D and Mohamed S 2015 Variational inference with normalizing flows Proc. 32nd Int. Conf. on Machine Learning (Proc. Machine Learning Research) vol 37 (available at: https://proceedings.mlr.press/v37/rezende15.html) pp 1530–8
[27] Minka T 2005 Technical Report (Microsoft Research) (available at: www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-2005-173.pdf)
[28] Neal R M 2001 Stat. Comput. 11 125–39
[29] Michel M, de Boer R and Pflueger S 2023 ComPWA: common partial wave analysis - making amplitude analysis transparant, understandable, and easy to start with (available at: https://github.com/ComPWA)
[30] Adolph C et al (COMPASS Collaboration) 2017 Phys. Rev. D 95 032004
[31] Kaspar F M and Gerassimov S (COMPASS Collaboration) 2022 Rev. Mex. Fis. Suppl. 3 0308020
[32] Marangotto D 2020 Adv. High Energy Phys. 7463073
[33] Aaij R et al (LHCb Collaboration) 2023 Phys. Rev. D 108 012023
[34] Aaij R et al (LHCb Collaboration) 2023 J. High Energy Phys. JHEP07(2023)228
[35] Penalva N, Hernández E and Nieves J 2019 Phys. Rev. D 100 113007
[36] Hu Q-Y, Li X-Q, Yang Y-D and Zheng D-H 2021 J. High Energy Phys. JHEP02(2021)183
[37] Byckling E and Kajantie K 1973 Particle Kinematics (Wiley)
[38] Meurer A et al 2017 PeerJ Comput. Sci. 3 e103
[39] Bradbury J et al 2018 JAX: composable transformations of Python+NumPy programs (available at: http://github.com/google/jax)
[40] Abazov V M et al (D0 Collaboration) 2010 Phys. Rev. D 82 071102
[41] Chatrchyan S et al (CMS Collaboration) 2011 Phys. Rev. D 84 092004
[42] Aad G et al (The ATLAS collaboration) 2023 J. High Energy Phys. JHEP06(2023)138
[43] Aad G et al (The ATLAS collaboration) 2023 J. High Energy Phys. JHEP07(2023)141
[44] André K D J, Benedikt M, Oide K and Zimmermann F (FCC-ee Collaboration) 2025 PoS ICHEP 2024 830
[45] Balazs C et al (Linear Collider Vision) 2025 (arXiv:2503.19983)
[46] Carrazza S, Cruz-Martinez J M and Rossi M 2021 Comput. Phys. Commun. 264 107995
[47] Plätzer S 2013 arXiv:1308.2922
[48] Lepage G P 2021 J. Comput. Phys. 439 110386
[49] Martino L, Elvira V and Louzada F 2017 Signal Process. 131 386–401
[50] Elvira V, Martino L and Robert C P 2022 Int. Stat. Rev. 90 525–50
[51] Cabezas A, Corenflos A, Lao J and Louf R 2024 BlackJAX: composable Bayesian inference in JAX (arXiv:2402.10797)
[52] Foreman-Mackey D 2016 J. Open Source Softw. 1 24
[53] Babuschkin I et al 2020 The DeepMind JAX Ecosystem (available at: http://github.com/deepmind)
[54] Hennigan T, Cai T, Norman T, Martens L and Babuschkin T 2020 (Sonnet for JAX) (available at: http://github.com/deepmind/dm-haiku)
[55] Hunter J D 2007 Comput. Sci. Eng. 9 90–95
[56] Harris C R et al 2020 Nature 585 357–62
[57] Heinrich L, Feickert M, Rodrigues E and Neuwirth P A 2024 pylhe v0.9.0 (available at: https://github.com/scikit-hep/pylhe)
[58] Fritsch M, Pflüger S, de Boer R E, Gradl W and Peters K 2024 ComPWA/tensorwaves: python fitter package for multiple computational back-ends (available at: https://github.com/ComPWA/tensorwaves)
[59] Petroff M A 2021 arXiv:2107.02270
[60] Durkan C, Bekasov A, Murray I and Papamakarios G 2019 Neural spline flows Advances in Neural Information Processing Systems vol 32, ed H Wallach, H Larochelle, A Beygelzimer, F d' Alché-Buc, E Fox and R Garnett (Curran Associates, Inc.) (https://proceedings.neurips.cc/paper/2019/file/7ac71d433f282034e088473244df8c02-paper.pdf)
[61] Kingma D and Ba J 2015 Adam: a method for stochastic optimization Int. Conf. on Learning Representations (ICLR), (San Diega, CA, USA)
[62] Klimek M D and Perelstein M 2020 SciPost Phys. 9 053
[63] Danziger K, Janßen T, Schumann S and Siegert F 2022 SciPost Phys. 12 164
[64] Backes M, Butter A, Plehn T and Winterhalder R 2021 SciPost Phys. 10 089