Flow Annealed Importance Sampling Bootstrap meets Differentiable Particle Physics

Annalena Kofler[*]
MPI for Intelligent Systems, Tübingen
MPI for Gravitational Physics, Potsdam
Technical University of Munich

Vincent Stimper
Isomorphic Labs
MPI for Intelligent Systems, Tübingen
University of Cambridge

Mikhail Mikhasenko
Ruhr University Bochum
ORIGINS Excellence Cluster, Munich

Michael Kagan
SLAC National Accelerator Laboratory

Lukas Heinrich
Technical University of Munich

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

Context. 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 points 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), that is, unnormalized distributions $p(x)$ over the outgoing 4-momenta x that are derived from a theoretical model. MEs can be evaluated analytically; however, sampling from them is hard since they can be high-dimensional and multi-modal. Additionally, they are defined on limited support originating from mass and momentum constraints and can exhibit divergences. Standard sampling algorithms such as `MadGraph` [4], `SHERPA` [5], and `PYTHIA` [6, 7] simplify MEs based on physics knowledge and employ adaptive Monte Carlo methods to approximate the distribution. More recently, machine learning-based surrogate models like normalizing flows have gained momentum to improve the efficiency of the sampling process [8–16]. As training normalizing flows with the maximum-likelihood loss relies on the costly generation of a large training data set, alternative approaches show potential that evaluate the distribution of interest directly during training [8–11, 13–16]. These methods evaluate the target distribution with samples from the normalizing flow, which requires gradients to be backpropagated through this distribution to update the flow parameters. Therefore, this training mode usually requires differentiable MEs, which have only recently been proposed [17] and employed [15] for normalizing

[*] Corresponding author: `annalena.kofler@tuebingen.mpg.de`

flow training. A similar method, called Flow Annealed importance sampling Bootstrap (FAB) [18], 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 Importance Sampling (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.

Contributions. In this work, (1) we adopt FAB [18] 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) [19] as a loss function, as well as with sample-based maximum-likelihood training with the forward Kullback-Leibler 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 [19–21] 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 $q_\theta(x)$.

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 Kullback-Leibler (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).

Forward KL Divergence (fKLD). If samples $x \sim p(x)$ are cheaply available, the fKLD serves as a loss function: $D_{\mathrm{KL}}(p \parallel q_\theta) = \mathbb{E}_p\left[\log(p(x)/q_\theta(x))\right]$ which can be simplified to the negative log-likelihood loss $\mathcal{L}_{\mathrm{fKLD}} = -\mathbb{E}_p\left[\log q_\theta(x)\right] = -\frac{1}{N}\sum_{i=1}^{N} \log q_\theta(x_i)$ over N data points. It can be shown that this loss function results in a density q_θ with mass-covering properties [18]. The flow distribution $q_\theta(x)$ is evaluated with samples from $p(x)$ and the gradient computation is straightforward: $\nabla_\theta \mathcal{L}_{\mathrm{fKLD}} = -\frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log q_\theta(x_i)$.

Reverse KL Divergence (rKLD). Another way of quantifying the difference between two distributions is via the rKLD, $D_{\mathrm{KL}}(q_\theta \parallel p) = \mathbb{E}_{q_\theta}\left[\log(q_\theta(x)/p(x))\right]$. This loss function has mode-seeking properties, meaning that it is not guaranteed that the optimized distribution $q_\theta(x)$ covers all modes of $p(x)$ [18]. To obtain a gradient for this loss function $\nabla_\theta \mathcal{L}_{\mathrm{rKLD}} = \nabla_\theta \mathbb{E}_{q_\theta}\left[\log(q_\theta(x)/p(x))\right]$, 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 Fig. 2b for illustration. As a result, $p(x)$ has to be differentiable[†].

FAB. Both versions of the KL divergence are special cases of the α-divergence [22]

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

where fKLD corresponds to $\alpha \to 1$ and rKLD to $\alpha \to 0$ [22]. In FAB [18], $D_{\alpha=2}$ 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 $p(x)$. Since samples from the flow might poorly fit the target distribution at the beginning of training, AIS [23] is employed to pass the samples through a chain of intermediate distributions $q_1, ..., q_{M-1}$ with HMC as a transition operator. The

[†] 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 [13]. However, this approach requires two subsequent applications of the flow transformations and is more costly.

intermediate distributions q_i are chosen to interpolate between the flow distribution $q_0 = q_\theta$ and the AIS target $q_M = p^2/q_\theta$. Importance weights w_{AIS} can be computed for the AIS samples x_{AIS} which allow evaluating the surrogate loss function $\mathcal{S}(\theta) = -\mathbb{E}_{\mathrm{AIS}}\left[\bar{w}_{\mathrm{AIS}} \log q_\theta(\bar{x}_{\mathrm{AIS}})\right]$ 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 $\{\bar{x}_{\mathrm{AIS}}, \bar{w}_{\mathrm{AIS}}\}$ are sampled from the buffer based on the importance weights to perform multiple gradient updates per iteration. Fig. 2c shows a schematic illustration of FAB; for further details, see Midgley et al. [18].

2.2 Differentiable Matrix Elements

Both training with rKLD and FAB requires gradients of the target distribution. Recent developments in HEP provide us with differentiable implementations of complex amplitudes [17, 24, 15]:

$\Lambda_c^+ \to pK^-\pi^+$. For the Λ_c^+ decays [25–27], the `ComPWA` package [24] utilizes the `SymPy` engine [28] to formulate symbolic amplitude models whose compute graph can be transformed to `JAX` [29] for efficient gradient computation. We select the $\Lambda_c^+ \to pK^-\pi^+$ decay to serve as a two-dimensional example due to its complex resonant structure resulting from the presence of multiple decay chains. The amplitude can be visualized in a Dalitz plot [30], shown in Fig. 3.

$e^+e^- \to t\bar{t}, t \to W^+b, \bar{t} \to W^-\bar{b}$. As a higher-dimensional example, we choose the eight-dimensional $t\bar{t}$ matrix element which can be generated with `MadJAX` [17] and provides a differentiable implementation of the scattering amplitudes of `MadGraph` [4]. A differentiable non-linear transformation is applied to map the irregular phase space boundaries to the unit hypercube based on `RAMBO` [31]. It is not possible to map the distribution to infinite support since divergences can occur at the phase space boundary, which would result in non-zero probability mass at infinity.

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 (see App. C for details).

Performance Metrics. We evaluate the performance of the normalizing flows based on three metrics: (1) the fKLD evaluated on a test data set, (2) the importance sampling 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 importance sampling efficiency [32, 33] 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

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

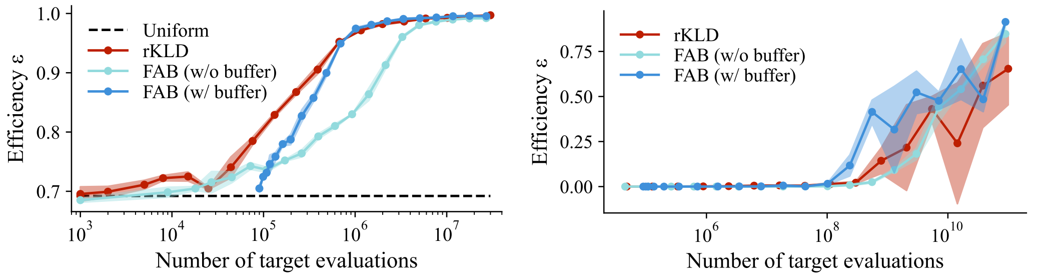
We elaborate on its derivation and similarities to the unweighting efficiency—which is commonly employed in HEP—in App. B. The integral estimate $\bar{I}$ of the target distribution $p(x)$ is defined as

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

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 the physics-agnostic grid-based integral estimation method called `VEGAS+` [34, 35] which is described in App. C in more detail. To illustrate the model performance, we provide histograms of flow samples for $\Lambda_c^+ \to pK^-\pi^+$ as well as a corner plot for the eight-dimensional ME in App. D. The performance metrics are summarized in Tab. 1 for both MEs. To make rKLD and FAB comparable, we report results based on training runs with the same number of target evaluations.

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

	$\Lambda_c^+ \to pK^-\pi^+$			$e^+e^- \to t\bar{t}, t \to W^+b, \bar{t} \to W^-\bar{b}$		
	$\mathbb{E}_p\left[\log(p/q_\theta)\right] \downarrow$	$\epsilon(\%) \uparrow$	$\bar{I}$	$\mathbb{E}_p\left[\log(p/q_\theta)\right] \downarrow$	$\epsilon(\%) \uparrow$	$\bar{I}$
`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 1: Importance sampling efficiency depending on the number of target evaluations required during training.

Since the target distribution can be costly to evaluate, we additionally show the trend for the importance sampling efficiency ϵ as a function of the number of target evaluations. Since fKLD training relies on a pre-computed training data set on which the normalizing flow can potentially be trained for an arbitrary number of epochs, we do not include it in this consideration. 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

$\boldsymbol{\Lambda_c^+ \to pK^-\pi^+}$**.** When training with the same number of target evaluations, rKLD and FAB with the replay buffer provide the best results with the highest importance sampling efficiency and a distribution that is most similar to the test data set (see Tab. 1). Additionally, the integral estimate computed with samples from the flow has the lowest standard deviations. When increasing the number of target evaluations, ϵ improves for rKLD more rapidly than for FAB w/o buffer. For FAB w/ buffer, samples have to be generated before 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 $\bar{w}_{\mathrm{AIS}}$, which aids the model significantly and the importance sampling efficiency improves at a faster rate compared to the flows trained with rKLD. Overall, flows trained with rKLD and both FAB approaches have no problem in modeling the complex 2D distribution, validated by histograms shown in Fig. 3.

$\boldsymbol{e^+e^- \to t\bar{t}, t \to W^+b, \bar{t} \to W^-\bar{b}}$**.** In higher dimensions, the flow benefits similarly from the AIS procedure and the sampling of batches from the buffer as in 2D. Samples from regions where the flow is a poor approximation of the target have a high importance weight and are predominantly used in gradient updates, resulting in a significantly higher importance sampling 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 Fig. 1b and the performance evaluation in Tab. 1.

5 Summary and Conclusion

We have transferred FAB [18], which utilizes AIS with HMC as a transition operator, from molecular configuration modeling to HEP, building on recently introduced, differentiable implementations of matrix elements. 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 matrix element contributions via multi-channeling will likely lead to further performance improvements [9, 13, 14]. Additionally, our conceptual work needs to be combined with differentiable implementations of the parton density [36, 15] 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.

Code and Data Availability. The code is available on GitHub and the data can be downloaded from Edmond [37].

Acknowledgments and Disclosure of Funding

The authors thank the motivated reviewers for their valuable remarks and feedback which helped improve this paper. AK thanks Timothy Gebhard for extensive support in reviewing and formatting the draft 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 [38], ComPWA [24], corner [39], distrax [40], haiku [41], JAX [29], FAB-JAX [18], MadJAX [17], MadGraph [42], matplotlib [43], numpy [44], pylhe [45] SymPy [28], TensorWaves [46], and VEGAS+ [34, 35]. The accessible color schemes in our figures are based on [47].

References

[1] O. Aberle et al. *High-Luminosity Large Hadron Collider (HL-LHC): Technical design report*. CERN Yellow Reports: Monographs. CERN, Geneva, 2020. doi: 10.23731/CYRM-2020-0010. URL https://cds.cern.ch/record/2749422.

[2] CMS Offline Software and Computing. CMS phase-2 computing model: Update document. Technical report, CERN, Geneva, 2022. URL https://cds.cern.ch/record/2815292.

[3] ATLAS Collaboration. ATLAS software and computing HL-LHC roadmap. Technical report, CERN, Geneva, 2022. URL https://cds.cern.ch/record/2802918.

[4] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli, and M. Zaro. The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations. *JHEP*, 07:079, 2014. doi: 10.1007/JHEP07(2014)079.

[5] E. Bothmann et al. Event Generation with Sherpa 2.2. *SciPost Phys.*, 7(3):034, 2019. doi: 10.21468/SciPostPhys.7.3.034. URL https://scipost.org/10.21468/SciPostPhys.7.3.034.

[6] C. Bierlich et al. A comprehensive guide to the physics and usage of PYTHIA 8.3. *SciPost Phys. Codebases*, page 8, 2022. doi: 10.21468/SciPostPhysCodeb.8. URL https://scipost.org/10.21468/SciPostPhysCodeb.8.

[7] C. Bierlich et al. Codebase release 8.3 for PYTHIA. *SciPost Phys. Codebases*, pages 8–r8.3, 2022. doi: 10.21468/SciPostPhysCodeb.8-r8.3. URL https://scipost.org/10.21468/SciPostPhysCodeb.8-r8.3.

[8] C. Gao, J. Isaacson, and C. Krause. i-flow: High-dimensional integration and sampling with normalizing flows. *Machine Learning: Science and Technology*, 1(4):045023, 2020. doi: 10.1088/2632-2153/abab62. URL https://dx.doi.org/10.1088/2632-2153/abab62.

[9] E. Bothmann, T. Janßen, M. Knobbe, T. Schmale, and S. Schumann. Exploring phase space with Neural Importance Sampling. *SciPost Phys.*, 8:069, 2020. doi: 10.21468/SciPostPhys.8.4.069. URL https://scipost.org/10.21468/SciPostPhys.8.4.069.

[10] C. Gao, S. Höche, J. Isaacson, C. Krause, and H. Schulz. Event generation with normalizing flows. *Phys. Rev. D*, 101:076002, 2020. doi: 10.1103/PhysRevD.101.076002. URL https://link.aps.org/doi/10.1103/PhysRevD.101.076002.

[11] S. Pina-Otey, F. Sánchez, T. Lux, and V. Gaitan. Exhaustive neural importance sampling applied to monte carlo event generation. *Phys. Rev. D*, 102:013003, 2020. doi: 10.1103/PhysRevD.102.013003. URL https://link.aps.org/doi/10.1103/PhysRevD.102.013003.

[12] B. Stienen and R. Verheyen. Phase space sampling and inference from weighted events with autoregressive flows. *SciPost Phys.*, 10:038, 2021. doi: 10.21468/SciPostPhys.10.2.038. URL https://scipost.org/10.21468/SciPostPhys.10.2.038.

[13] T. Heimel, R. Winterhalder, A. Butter, J. Isaacson, C. Krause, F. Maltoni, O. Mattelaer, and T. Plehn. MadNIS - Neural multi-channel importance sampling. *SciPost Phys.*, 15:141, 2023. doi: 10.21468/SciPostPhys.15.4.141. URL https://scipost.org/10.21468/SciPostPhys.15.4.141.

[14] T. Heimel, N. Huetsch, F. Maltoni, O. Mattelaer, T. Plehn, and R. Winterhalder. The MadNIS reloaded. *SciPost Phys.*, 17:023, 2024. doi: 10.21468/SciPostPhys.17.1.023. URL https://scipost.org/10.21468/SciPostPhys.17.1.023.

[15] T. Heimel, O. Mattelaer, T. Plehn, and R. Winterhalder. Differentiable MadNIS-Lite. *arXiv preprints*, 2024. URL https://arxiv.org/abs/2408.01486.

[16] N. Deutschmann and N. Götz. Accelerating HEP simulations with Neural Importance Sampling. *Journal of High Energy Physics*, 2024(3):83, 2024. doi: 10.1007/JHEP03(2024)083. URL https://doi.org/10.1007/JHEP03(2024)083.

[17] L. Heinrich and M. Kagan. Differentiable Matrix Elements with MadJax. *J. Phys. Conf. Ser.*, 2438(1):012137, 2023. doi: 10.1088/1742-6596/2438/1/012137. URL https://dx.doi.org/10.1088/1742-6596/2438/1/012137.

[18] L. I. Midgley, V. Stimper, G. N. C. Simm, B. Schölkopf, and J. M. Hernandez-Lobato. Flow annealed importance sampling bootstrap. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XCTVFJwS9LJ.

[19] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021. URL http://jmlr.org/papers/v22/19-1028.html.

[20] E. G. Tabak and E. Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217 – 233, 2010. URL https://dx.doi.org/10.4310/CMS.2010.v8.n1.a11.

[21] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, 2015. URL https://proceedings.mlr.press/v37/rezende15.html.

[22] T. Minka. Divergence measures and message passing. *Technical report, Microsoft Research*, 2005. URL https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-2005-173.pdf.

[23] R. M. Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001. doi: 10.1023/A:1008923215028. URL https://doi.org/10.1023/A:1008923215028.

[24] M. Michel, R. de Boer, and S. Pflueger. comPWA: Common partial wave analysis - making amplitude analysis transparant, understandable, and easy to start with, 2023. URL https://github.com/ComPWA.

[25] D. Marangotto. Extracting Maximum Information from Polarised Baryon Decays via Amplitude Analysis: The $\Lambda_c^+ \to pK^-\pi^+$ Case. *Adv. High Energy Phys.*, 2020:7463073, 2020. doi: 10.1155/2020/7463073. URL https://onlinelibrary.wiley.com/doi/abs/10.1155/2020/7463073.

[26] R. Aaij et al. Amplitude analysis of the $\Lambda_c^+ \to pK^-\pi^+$ decay and Λ_c^+ baryon polarization measurement in semileptonic beauty hadron decays. *Phys. Rev. D*, 108:012023, 2023. doi: 10.1103/PhysRevD.108.012023. URL https://link.aps.org/doi/10.1103/PhysRevD.108.012023.

[27] R. Aaij et al. Λ_c^+ polarimetry using the dominant hadronic mode. *JHEP*, 07:228, 2023. doi: 10.1007/JHEP07(2023)228.

[28] A. Meurer et al. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017. doi: 10.7717/peerj-cs.103. URL https://doi.org/10.7717/peerj-cs.103.

[29] J. Bradbury et al. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[30] R. H. Dalitz. Decay of τ mesons of known charge. *Phys. Rev.*, 94:1046–1051, 1954. doi: 10.1103/PhysRev.94.1046. URL https://link.aps.org/doi/10.1103/PhysRev.94.1046.

[31] S. Plätzer. Rambo on diet. *arXiv e-prints*, art. arXiv:1308.2922, 2013. doi: 10.48550/arXiv.1308.2922. URL https://arxiv.org/abs/1308.2922.

[32] L. Martino, V. Elvira, and F. Louzada. Effective sample size for importance sampling based on discrepancy measures. *Signal Processing*, 131:386–401, 2017. doi: https://doi.org/10.1016/j.sigpro.2016.08.025. URL https://www.sciencedirect.com/science/article/pii/S0165168416302110.

[33] V. Elvira, L. Martino, and C. P. Robert. Rethinking the effective sample size. *International Statistical Review*, 90(3):525–550, 2022. doi: https://doi.org/10.1111/insr.12500. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/insr.12500.

[34] P. G. Lepage. A new algorithm for adaptive multidimensional integration. *Journal of Computational Physics*, 27(2):192–203, 1978. doi: https://doi.org/10.1016/0021-9991(78)90004-9. URL https://www.sciencedirect.com/science/article/pii/0021999178900049.

[35] P. G. Lepage. Adaptive multidimensional integration: vegas enhanced. *Journal of Computational Physics*, 439:110386, 2021. doi: https://doi.org/10.1016/j.jcp.2021.110386. URL https://www.sciencedirect.com/science/article/pii/S0021999121002813.

[36] S. Carrazza, J. M. Cruz-Martinez, and M. Rossi. Pdfflow: Parton distribution functions on gpu. *Computer Physics Communications*, 264:107995, 2021. ISSN 0010-4655. doi: https://doi.org/10.1016/j.cpc.2021.107995. URL https://www.sciencedirect.com/science/article/pii/S0010465521001077.

[37] A. Kofler, V. Stimper, M. Mikhasenko, M. Kagan, and L. Heinrich. Data for "Flow Annealed Importance Sampling Bootstrap meets Differentiable Particle Physics", 2024. URL https://doi.org/10.17617/3.UZ786R.

[38] Alberto Cabezas, Adrien Corenflos, Junpeng Lao, and Rémi Louf. BlackJAX: Composable Bayesian inference in JAX, 2024.

[39] D. Foreman-Mackey. corner.py: Scatterplot matrices in python. *The Journal of Open Source Software*, 1(2):24, 2016. doi: 10.21105/joss.00024. URL https://doi.org/10.21105/joss.00024.

[40] I. Babuschkin et al. The DeepMind JAX Ecosystem, 2020. URL http://github.com/deepmind.

[41] T. Hennigan, T. Cai, T. Norman, L. Martens, and T. Babuschkin. Haiku: Sonnet for JAX, 2020. URL http://github.com/deepmind/dm-haiku.

[42] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli, and M. Zaro. The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations. *JHEP*, 07:079, 2014. doi: 10.1007/JHEP07(2014)079. URL https://doi.org/10.1007/JHEP07(2014)079.

[43] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9 (3):90–95, 2007. doi: 10.1109/mcse.2007.55.

[44] C. R. Harris et al. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

[45] L. Heinrich, M. Feickert, E. Rodrigues, and A. P. Neuwirth. pylhe: v0.9.0, 2024. URL https://github.com/scikit-hep/pylhe.

[46] M. Fritsch, S. Pflüger, R. E. de Boer, W. Gradl, and K. Peters. ComPWA/tensorwaves: Python fitter package for multiple computational back-ends, 2024. URL https://github.com/ComPWA/tensorwaves.

[47] M. A. Petroff. Accessible Color Sequences for Data Visualization. *arXiv preprints*, 2021.

[48] M. D. Klimek and M. Perelstein. Neural network-based approach to phase space integration. *SciPost Phys.*, 9:053, 2020. doi: 10.21468/SciPostPhys.9.4.053. URL https://scipost.org/10.21468/SciPostPhys.9.4.053.

[49] K. Danziger, T. Janßen, S. Schumann, and F. Siegert. Accelerating Monte Carlo event generation – rejection sampling using neural network event-weight estimates. *SciPost Phys.*, 12:164, 2022. doi: 10.21468/SciPostPhys.12.5.164. URL https://scipost.org/10.21468/SciPostPhys.12.5.164.

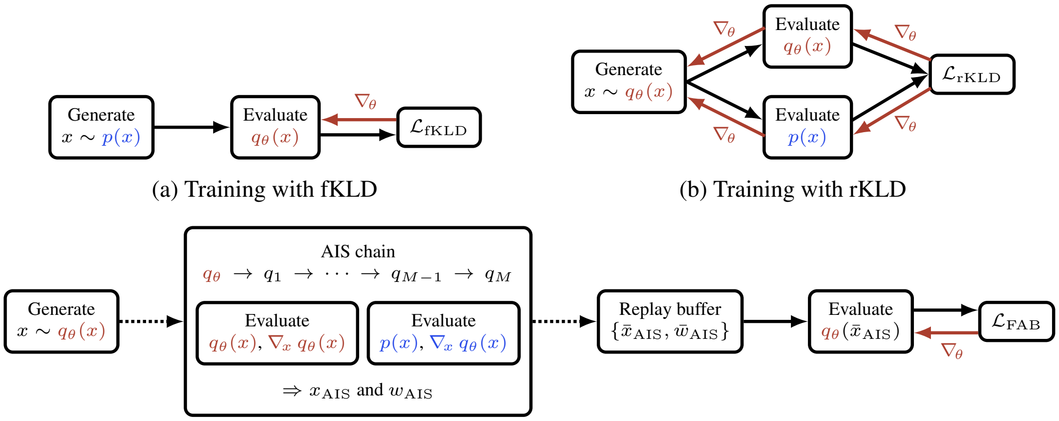
[50] M. Backes, A. Butter, T. Plehn, and R. Winterhalder. How to GAN Event Unweighting. *SciPost Phys.*, 10:089, 2021. doi: 10.21468/SciPostPhys.10.4.089. URL https://scipost.org/10.21468/SciPostPhys.10.4.089.

[51] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios. Neural spline flows. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/7ac71d433f282034e088473244df8c02-Paper.pdf.

[52] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA, 2015.

Appendix

A Schematic comparison of training methods


Generate $x \sim p(x)$
Evaluate $q_\theta(x)$
∇_θ
$\mathcal{L}_{\text{fKLD}}$
Generate $x \sim q_\theta(x)$
Evaluate $q_\theta(x)$
Evaluate $p(x)$
∇_θ
∇_θ
∇_θ
∇_θ
$\mathcal{L}_{\text{rKLD}}$
Generate $x \sim q_\theta(x)$
AIS chain
$q_\theta \rightarrow q_1 \rightarrow \cdots \rightarrow q_{M-1} \rightarrow q_M$
Evaluate $q_\theta(x)$, $\nabla_x q_\theta(x)$
Evaluate $p(x)$, $\nabla_x q_\theta(x)$
$\Rightarrow x_{\text{AIS}}$ and w_{AIS}
Replay buffer $\{\bar{x}_{\text{AIS}}, \bar{w}_{\text{AIS}}\}$
Evaluate $q_\theta(\bar{x}_{\text{AIS}})$
$\mathcal{L}_{\text{FAB}}$
∇_θ

(a) Training with fKLD

(b) Training with rKLD

(c) Training with FAB

Figure 2: 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 $\bar{x}_{\text{AIS}}$ and weights $\bar{w}_{\text{AIS}}$. The dotted lines indicate that these steps do not occur when a gradient update is performed with samples from the replay buffer.

B Relationship of sampling efficiency and unweighting efficiency

Importance sampling efficiency. The (importance) sampling efficiency ϵ (c.f. Eq. 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 importance sampling (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) [32]. Through derivations outlined in [33], the ESS can be related to the estimate

$$\widehat{ESS} = N \frac{\left(\frac{1}{N}\sum_{i=1}^{N} w_i\right)^2}{\frac{1}{N}\sum_{i=1}^{N} w_i^2} = \frac{\left(\sum_{i=1}^{N} w_i\right)^2}{\sum_{i=1}^{N} w_i^2} = \frac{1}{\sum_{i=1}^{N} \bar{w}_i^2} \,. \tag{4}$$

with the normalized importance weights $\bar{w}_i = w_i / \sum_{j=1}^{N} w_j$ [32]. The importance sampling efficiency ϵ as defined in Eq. 2 corresponds to the normalization of Eq. 4.

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 $q(x)$ [48]. 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 [48, 9, 49] as

$$\epsilon_{\text{uw}} = \frac{\frac{1}{N}\sum_{i=1}^{N} w_i}{w_{\max}} \tag{5}$$

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.

Table 2: Comparison of importance sampling and unweighting efficiency for the different MEs and methods.

	$\Lambda_c^+ \to pK^-\pi^+$		$e^+e^- \to t\bar{t}, t \to W^+b, \bar{t} \to W^-\bar{b}$	
	$\epsilon(\%) \uparrow$	$\epsilon_{\mathrm{uw}}(\%) \uparrow$	$\epsilon(\%) \uparrow$	$\epsilon_{\mathrm{uw}}(\%) \uparrow$
`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

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 [48]. 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 $w_{\mathrm{rel}} = w_i / w_{\mathrm{max}}$. A sample is retained if a uniformly sampled random number R is smaller than w_{rel} [50, 49]. Finally, the unweighting efficiency can be computed as the average raw weights rescaled by w_{max} [9, 49].

Relationship of efficiencies. In [32], 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 $1/\max(\bar{w}_1, ..., \bar{w}_N)$ (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 [9, 10]. For this reason, we choose to report the performance based on the importance sampling efficiency in the main body of this work and provide additional estimates of the unweighting efficiency in Tab. 2. We do not apply bootstrap techniques that are designed to mitigate large outliers in the weight distribution [10].

C 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` [42] 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 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 [51] implemented in `distrax` [40] and `haiku` [41]. 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. 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 $l_{\mathrm{init}} = 0.005$ for the eight-dimensional ME for all subsequent runs.

Optimization. We use the `Adam` [52] 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 [18]. 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 a physics-agnostic integral estimation method called `VEGAS+` [34, 35]. 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 and importance sampling, `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 doubled the grid to 128 bins per dimension, kept the damping factor, and increased 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 performed checks like increasing the number of evaluations by a factor of 10 per iteration which provided stable results and did not show significant deviations in the integral estimate for both examples. However, the low importance sampling efficiency and large deviation on the integral estimate for the 8D ME (c.f., Tab. 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 matrix elements 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 [13, 14]. Although the `VEGAS+` optimization is significantly faster than training a normalizing flow, the flexibility of the latter results in higher importance sampling efficiencies. Since the main goal of training ME surrogates is to provide an optimally pre-trained model [14], the training time is amortized when generating a large number of events.

D Additional results

$\Lambda_c^+ \to pK^-\pi^+$. 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. 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. Physics information can be extracted by comparing the Dalitz plot histograms based on simulated samples and measured data [26, 27]. As we train the normalizing flow on $[0, 1]^2$, we apply a mapping to the flow samples to obtain the Dalitz plot representation. 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. We observe that fKLD

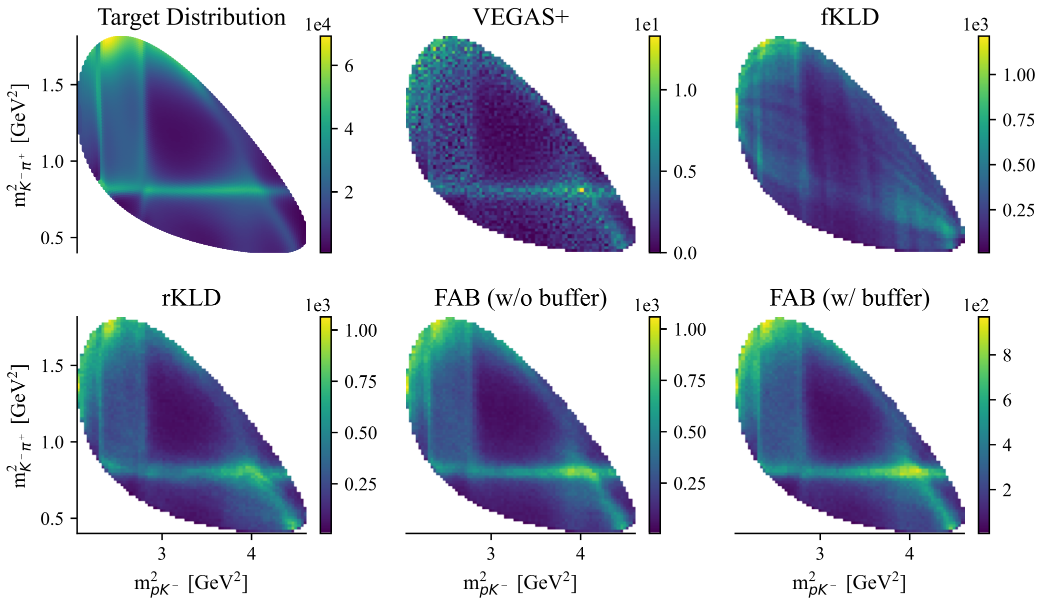


Figure 3: Comparison of the target density for the $\Lambda_c^+ \to pK^-\pi^+$ matrix element with histograms based on samples from `VEGAS+` and the best normalizing flow for each method.

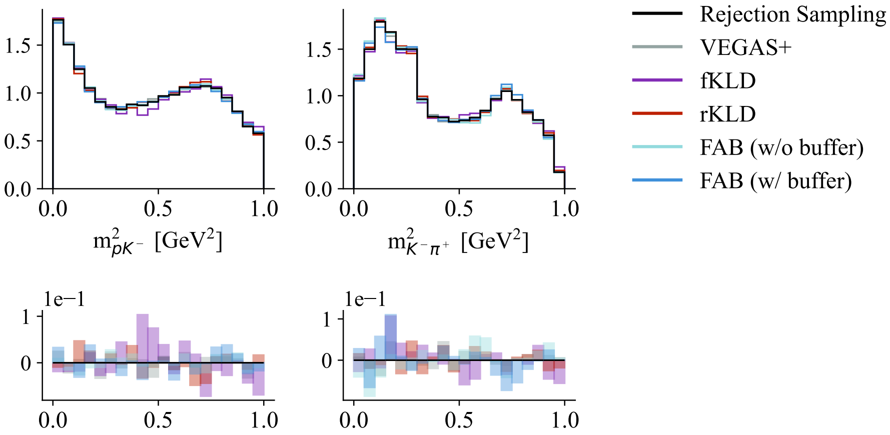


Figure 4: Normalized marginal histograms of the invariant masses for the $\Lambda_c^+ \to pK^-\pi^+$ matrix element based on samples from rejection sampling, `VEGAS+`, and the best normalizing flow for each method. The rejection sampling result serves as a baseline to visualize the deviation.

struggles to learn the horizontal and diagonal lines ("resonances"). Both normalizing flows trained with FAB produce histograms that exhibit the full structure of the ME and provide the best results. To additionally allow a quantitative visual comparison of higher-level quantities than the Dalitz plot, we include histograms of the invariant mass distribution for $m^2_{pK^-}$ and $m^2_{K^-\pi^+}$ in Fig. 4.

$e^+e^- \to t\bar{t}, t \to W^+b, W^-\bar{b}$. We compare the corner plots of 10^6 flow samples for each method to samples from the training data set generated with `MadGraph` [42] in Figure 5. 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 one would need to apply the inverse `RAMBO` transformation [31] to obtain physical information for each outgoing particle. We can relate the dimensions of the unit hypercube to physical quantities that correspond to re-scaled versions of

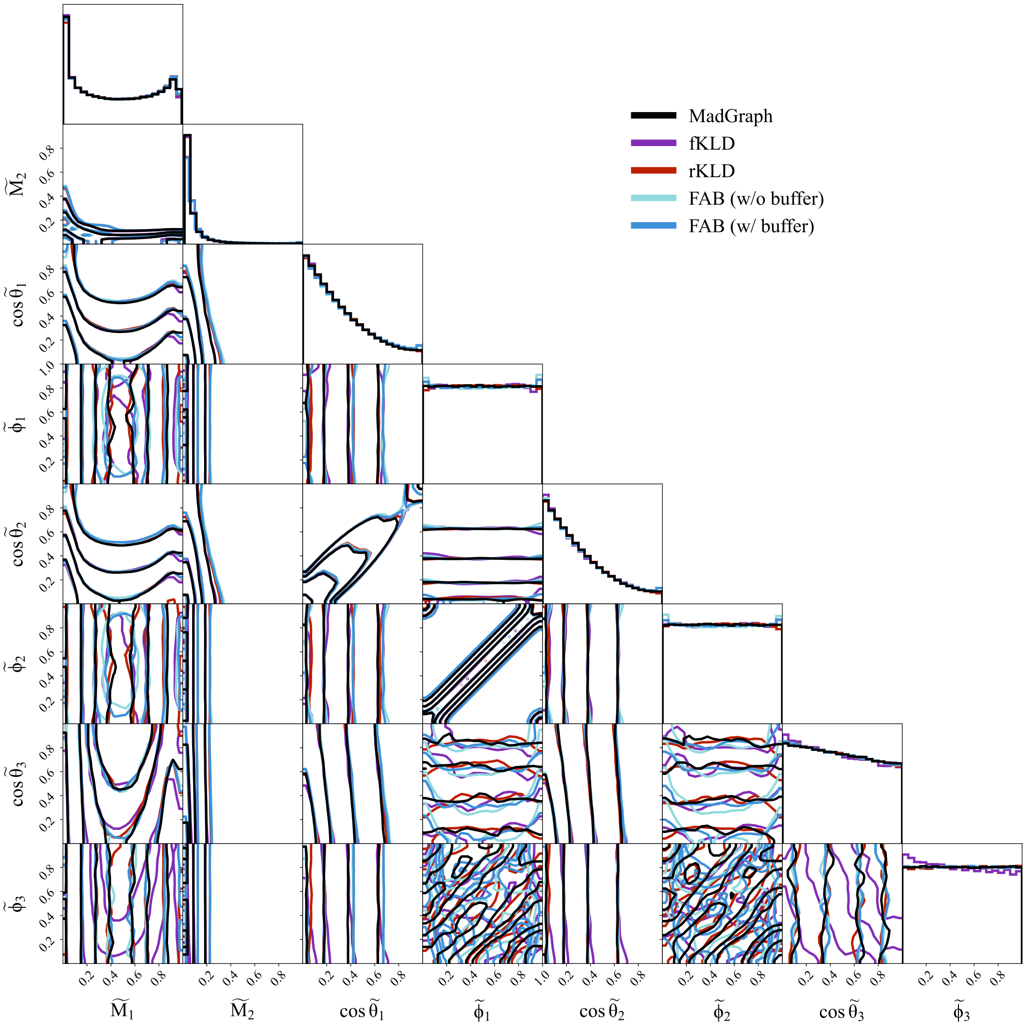


Figure 5: Corner plot with samples from `MadGraph` and from the best normalizing flows for each method.

intermediate masses $\widetilde{\mathrm{M}}$, the cosine of azimuthal angles $\cos\widetilde{\theta}$, and polar angles $\widetilde{\phi}$. 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 Tab. 1 and 2, we do not include the results in the corner plot since we observe large deviations explained by low efficiencies.