Amplifying Statistics using Generative Models

Anja Butter
Institut für Theoretische Physik, Universität Heidelberg
Philosophenweg 16, 69120 Heidelberg, Germany

Sascha Diefenbacher
Institut für Experimentalphysik, Universität Hamburg
Luruper Chaussee 149, 22761 Hamburg, Germany

Gregor Kasieczka
Institut für Experimentalphysik, Universität Hamburg
Luruper Chaussee 149, 22761 Hamburg, Germany

Benjamin Nachman
Physics Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA
Berkeley Institute for Data Science, University of California, Berkeley, CA 94720, USA

Tilman Plehn
Institut für Theoretische Physik, Universität Heidelberg
Philosophenweg 16, 69120 Heidelberg, Germany

Abstract

A critical question concerning generative networks applied to physics simulations is if the generated events add statistical precision beyond the training sample. We show for a simple example how generative networks indeed amplify the training statistics. We quantify their impact through an amplification factor or equivalent numbers of sampled events.

1 Introduction

A defining feature of particle physics is that we can produce high fidelity simulations with nearly the same complexity as nature in order to perform inference. These simulations include physical processes spanning nearly 20 orders of magnitude in length scales, which is possible from a factorized Markov Chain Monte Carlo approach. The upcoming high-luminosity run of the Large Hadron Collider (LHC) will produce a data set more than 25 times the data set collected in the 2010s and 2020s, with precision requirements seriously challenging the current simulation tools. One way to speed up the simulations and, in turn, improve their precision is to employ modern machine learning.

A variety of generative machine learning models have been proposed, including well-studied methods such as generative adversarial networks (GAN) [1, 2], variational autoencoders [3, 4], and variations of normalizing flows [5, 6]. This work will focus on GANs, the most widely studied approach in high energy physics so far. Fast precision simulation in particle physics starts with phase space integration [7, 8], phase space sampling [9–11], and amplitude networks [12, 13]. Especially interesting are NN-based event generation [14–18], event subtraction [19], detector simulations [20–28], or fast parton showers [29–32]. Deep generative models can also improve searches for physics beyond the Standard Model [33] or anomaly detection [34, 35]. Finally, GANs allow us to unfold detector

effects [36, 37], surpassed only by the consistent statistical treatment of conditional invertible networks [38]. Roughly, these applications fall into two categories, (i) generative networks accelerating or augmenting Monte Carlo simulations or (ii) generative networks offering entirely new analysis opportunities. For the first kind of application the known big question is *how many more events can we sensibly GAN before we are limited by the statistics of the training sample*?

A seemingly straightforward and intuitive answer to this question is *as many examples as were used for training, because the network does not add any physics knowledge* [39]. However, there are reasons to think that a generative model actually contains more statistical power than the original data set. The key property of neural networks in particle physics is their powerful interpolation in sparse and high-dimensional spaces. It is also behind the success of the NNPDF parton densities [40] as the first mainstream application of machine learning to particle physics theory. This advanced interpolation should provide GANs with additional statistical power. We even see promising hints for network extrapolation in generating jet kinematics [16].

Indeed, neural networks go beyond a naive interpolation in that their architectures define basic properties of the functions it parameterizes. For example, some kind of smoothness criterion combined with a typical resolution adds information to discrete training data. An extreme baseline which we will use in this submission is the case where the true density distribution is known in terms of a few unknown parameters. With this information it is always better to fit those parameters and compute statistics with the functional form than to estimate the statistics directly from the data.

In the machine learning literature this kind of question is known for example as data amplification [41], but not extensively discussed. An interesting application is computer games, where the network traffic limits the available information and a powerful computer still generates realistic images or videos. This practical question leads to more formal question of sampling correctors [42]. Alternatively, it can be linked to a classification problem to derive scaling laws for networks fooling a given hypothesis test [41, 43]. In this submission we will use a much simpler approach, close to typical particle physics theory applications. If we know the smooth truth distribution, we can bin our space to define quantiles — intervals containing equal probability mass — and compute the χ^2-values for sampled and GANned approximations. Our simple example will be a camel back function because it is similar to multi-modal smooth distributions common in HEP.

2   One-dimensional camel back

The first function we study is a one-dimensional camel back, made out of two normalized Gaussians $N_{\mu,\sigma}(x)$ with mean μ and width σ,

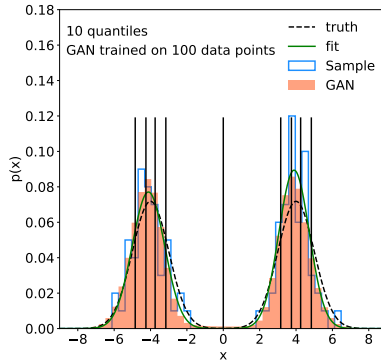$$P(x) = \frac{N_{-4,1}(x) + N_{4,1}(x)}{2} \ .$$

(1)

Figure 1: Camel back function as a 1D test case. We show the true distribution (black), a histogram with 100 sample points (blue), a fit to the samples data (green), and a high-statistics GAN sample (orange). Ten quantiles include 10% of the truth integral each.
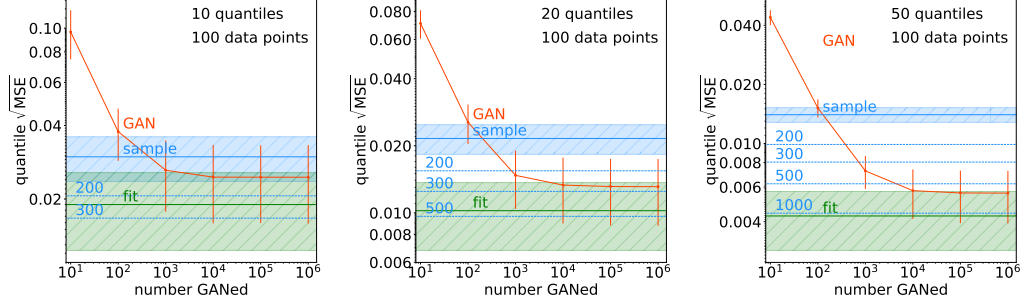
Figure 2: Quantile error for the 1D camel back function for sampling (blue), fit (green), and GAN (orange). Left to right we show results for 10, 20, and 50 quantiles.

We show this function in Fig. 1, together with a histogrammed data set of 100 points. As a benchmark we define a 5-parameter maximum-likelihood fit, where we assume that we know the functional form and determine the two means, the two widths and the relative height of the Gaussians in Eq. (1). We perform this fit using the IMINUIT [44] and PROBFIT [45] PYTHON packages. The correctly assumed functional form is much more than we can encode in a generative network architecture, so the network will not outperform the precision of this fit benchmark. On the other hand, the fit illustrates an optimal case, where in practice we usually do not know the true functional form.

To quantify the agreement for instance between the data sample or the fit on the one hand and the exact form on the other, we introduce 10, 20, or 50 quantiles. We illustrate the case of 10 quantiles also in Fig.1. We can evaluate the quality of an approximation to the true curve by computing the average quantile error

$$\text{MSE} = \frac{1}{N_{\text{quant}}} \sum_{j=1}^{N_{\text{quant}}} \left(x_j - \frac{1}{N_{\text{quant}}} \right)^2 \, , \tag{2}$$

where x_j is the estimated probability in each of the N_{quant} quantiles, which are defined with known boundaries. In Fig. 2 the horizontal lines show this measure for histograms with 100 to 1000 sampled points and for the fit to 100 points. For the 100-point sample we construct an error band by evaluating 100 statistically independent samples and computing its standard deviation. For the fit we do the same, i.e fit the same 100 independent samples and compute the standard deviation for the fit output. This should be equivalent to the one-sigma range of the five fitted parameters, if we take into account all correlations.

The first observation in Fig. 2 is that the agreement between the sample or the fit and the truth generally improves with more quantiles, which is simply a property of our quantile MSE error. Second, the precision of the fit corresponds to roughly 300 hypothetical data points for 10 quantiles, 500 hypothetical data points for 20 quantiles, and close to 1000 hypothetical data points for 50 quantiles. This means that for high resolution and an extremely sparsely populated 1D-phase space, the assumed functional value for the fit allows the data to have the same statistical power as a dataset with no knowledge of the functional form that is 10 times bigger. If we define the *amplification factor* as the ratio between asymptotic performance to training events, the factor when using the fit information would be about 10. The question is, how much is a GAN with its very basic assumptions worth, for instance in comparison to this fit?

We introduce a simple generative model using the generator-discriminator structure of a standard GAN. This architecture remains generic in the sense that we do not use specific knowledge about the data structure or its symmetries in the network construction. All neural networks are implemented using PYTORCH [46]. Our setup is illustrated in Fig. 3. The generator is a fully connected network (FCN). Its input consists of 1000 random numbers, uniformly sampled from $[-1, 1]$. It is passed to seven layers with 256 nodes each, followed by a final output layer with d nodes, where d is the number of phase space dimensions. To each fully-connected layer we add a 50% dropout layer [47] to reduce over-fitting which is kept active during generation. The generator uses the ELU activation function [48].
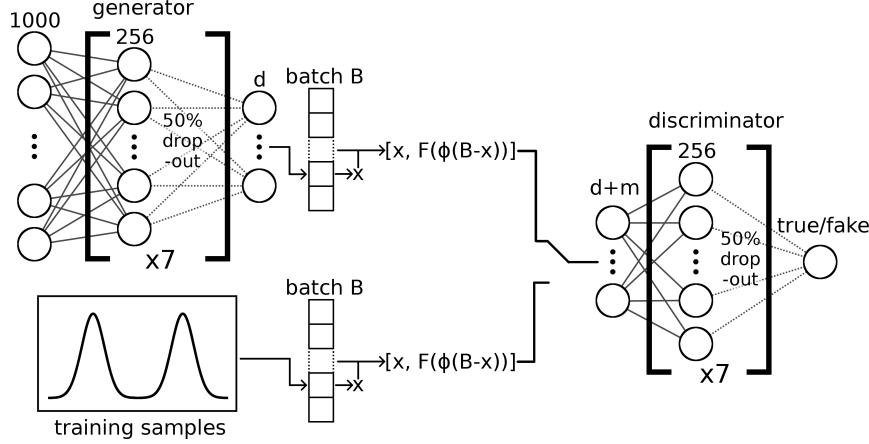
3

Figure 3: Diagram of our neural network architecture. The structure and usage of the embedding function Φ, the aggregation function F, and other hyperparameters are described in the main text.

The discriminator is also a FCN. To avoid mode collapse we give it access to per-batch statistics in addition to individual examples using an architecture inspired by DeepSets [49, 50]. This way its input consists of two objects, a data point $x \in \mathbb{R}^d$ and the full batch $B \in \mathbb{R}^{d,n}$, where n is the batch size and x corresponds to one column in B. First, we calculate the difference vector between x and every point in B, $B - x$ with appropriate broadcasting, so that $B - x \in \mathbb{R}^{d,n}$ as well. This gives the discriminator a handle on the distance of generated points. This distance is passed to an embedding function $\Phi : \mathbb{R}^{d,n} \to \mathbb{R}^{m,n}$, where m the size of the embedding. The embedding Φ is implemented as three 1D-convolutions (256 filters, 256 filters, m filters) with kernel size 1, stride 1 and no padding. Each of the convolutions uses a LEAKYRELU [51] activation function with a slope of 0.01. For the embedding size we choose $m = 32$.

We then use an aggregation function F : $\mathbb{R}^{m,n} \to \mathbb{R}^m$ along the batch-size direction. The network performance is largely independent of the choice of aggregation function. Still we find that our choice of standard deviation slightly outperforms other options. We then concatenate x and $F(\Phi(B - x))$ to a vector with length $d + m$. It is passed to the main discriminator network, an FCN consisting of a $d + m$ node input layer followed by seven hidden layers with 256 nodes each and a 1 node output layer. Mirroring the setup of the generator we once again intersperse each hidden layer with a 50% dropout layer. The discriminator uses 0.01 slope LEAKYRELU activation functions for the hidden layers and a SIGMOID function for the output.

During training we strictly alternate between discriminator and generator. Both networks use the ADAM optimizer [52] with $\beta_1 = 0.5$, $\beta_2 = 0.9$ and a learning rate of 5×10^{-5}. This learning rate gets multiplied by 0.9 every 1000 epochs. The GAN is trained for a total of 10.000 epochs. To regularize the discriminator training we use gradient penalty [53] with $\gamma = 0.01$. Additionally, we apply a uniform noise ranging from $[-0.1, 0.1]$ to the training data in every discriminator evaluation, once again to reduce over-fitting and mode-collapse. We chose a batch size of $n = 10$ and the whole data set is shuffled each epoch, randomizing the makeup of these batches. One training with 100 data points takes approximately one hour on a NVIDIA TESLA P100 GPU. The architectures used for the different dimensionalities are identical except for changes to the generator output and discriminator input sizes.

Returning to Fig. 2, the orange curve shows the performance of this GAN setup compared to the sample and to the 5-parameter fit. The GAN uncertainty is again derived by 100 independent trainings. We then compute the quantile error as a function of the number of GANned events and see how it saturates. This saturation is where GANning more events would not add more information to a given sample. Depending on the number of quantiles or the resolution this happens between 1000 and 10.000 GANned events, to be compared with 100 training events. This shows that it does make sense to generate more events than the training sample size.

On the other hand, training a GAN encodes statistical uncertainties in the training sample into systematic uncertainties in the network. This means that a GANned event does not carry the same

4

amount of information as a sampled event. The asymptotic value of the GAN quantile error should be compared to the expected quantile error for an increased sample, and we find that the 10,000 GANned events are worth between 150 and 500 sampled events, depending on the applied resolution. Thus, our simple GAN produces an amplification factor above five for a resolution corresponding to 50 quantiles. An interesting feature of the GAN is that it follows the fit result with a slight degradation, roughly corresponding to the quoted fit uncertainty. In that sense the analogy between a fit and a flexible NN-representation makes sense, and the GAN comes shockingly close to the highly constrained fit. This is despite the GAN being significantly more general and complex therefore more prone to over-fitting than the parameter fit.

3 Outlook

A crucial question for applications of generative models to particle physics simulations is how many events we can generate through the fast network before we exhaust the physics information encoded in the training sample. Generally, a neural network adds information or physics knowledge though the class of functions they represent. To answer this question for GANs, we split the phase space of a simple test function into quantiles and use the combined quantile MSE to compare sample, GAN, and a fit benchmark.

We find that it makes sense to GAN significantly more events than we have in the training sample, but those individual events carry less information than a training sample event. As GAN sampling can be much more computationally efficient than *ab initio* simulations, this results in a net statistical benefit. We define an *amplification factor* through the number of hypothetical training events with the same information content as a large number of GANned events. This amplification factor strongly depends on the number of quantiles or the phase space resolution. While we can never beat a low-parameter fit, the GAN comes surprisingly close and its amplification factor scales with the amplification factor of the fit.

While our focus is on GANs, our observed amplification is also relevant for other applications of neural networks. Specifically, it also applies to networks increasing the data quality through refining [37, 54] or reweighting [55, 56]. It will be interesting to see how the improvement scales with the amount of training data for these approaches.

Broader Impact

While the intended beneficiaries of accelerated physics simulation are the multinational experimental collaborations at the Large Hadron Collider and elsewhere, there are additional communities that would directly benefit. In particular, if we can show that deep generative simulations can increase the statistical power from their training datasets while still maintaining high fidelity, then individual members of the community will be able to use these tools for their own research and education purposes. The traditional computationally expensive physics-based simulations are typically not usable by such community members so deep generative models have a great potential for equalizing access. Most problems in industry and society at large do not have precise first principles generative models to train deep generative models, so these studies are likely not relevant beyond research and education for physical science.

Acknowledgments and Disclosure of Funding

References

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial nets*, in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14. MIT Press, Cambridge, MA, USA, 2014.

[2] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, *Generative adversarial networks: An overview*, IEEE Signal Processing Magazine **35** (Jan, 2018) 53–65.

[3] D. P. Kingma and M. Welling, *Auto-encoding variational bayes.*, in *ICLR*, Y. Bengio and Y. LeCun, eds. 2014.

[4] D. P. Kingma and M. Welling, *An introduction to variational autoencoders*, Foundations and Trends® in Machine Learning **12** (2019) 4, 307–392.

[5] D. J. Rezende and S. Mohamed, *Variational inference with normalizing flows*, in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15. JMLR.org, 2015.

[6] I. Kobyzev, S. Prince, and M. Brubaker, *Normalizing flows: An introduction and review of current methods*, IEEE Transactions on Pattern Analysis and Machine Intelligence (2020) 1–1.

[7] M. D. Klimek and M. Perelstein, *Neural Network-Based Approach to Phase Space Integration*, arXiv:1810.11509 [hep-ph].

[8] J. Bendavid, *Efficient Monte Carlo Integration Using Boosted Decision Trees and Generative Deep Neural Networks*, arXiv:1707.00028 [hep-ph].

[9] E. Bothmann, T. Janßen, M. Knobbe, T. Schmale, and S. Schumann, *Exploring phase space with Neural Importance Sampling*, SciPost Phys. **8** (2020) 4, 069, arXiv:2001.05478 [hep-ph].

[10] C. Gao, J. Isaacson, and C. Krause, *i-flow: High-dimensional Integration and Sampling with Normalizing Flows*, arXiv:2001.05486 [physics.comp-ph].

[11] C. Gao, S. Höche, J. Isaacson, C. Krause, and H. Schulz, *Event Generation with Normalizing Flows*, Phys. Rev. D **101** (2020) 7, 076002, arXiv:2001.10028 [hep-ph].

[12] F. Bishara and M. Montull, *(Machine) Learning Amplitudes for Faster Event Generation*, arXiv:1912.11055 [hep-ph].

[13] S. Badger and J. Bullock, *Using neural networks for efficient evaluation of high multiplicity scattering amplitudes*, arXiv:2002.07516 [hep-ph].

[14] S. Otten *et al.*, *Event Generation and Statistical Sampling with Deep Generative Models and a Density Information Buffer*, arXiv:1901.00875 [hep-ph].

[15] B. Hashemi, N. Amin, K. Datta, D. Olivito, and M. Pierini, *LHC analysis-specific datasets with Generative Adversarial Networks*, arXiv:1901.05282 [hep-ex].

[16] R. Di Sipio, M. Faucci Giannelli, S. Ketabchi Haghighat, and S. Palazzo, *DijetGAN: A Generative-Adversarial Network Approach for the Simulation of QCD Dijet Events at the LHC*, JHEP **08** (2020) 110, arXiv:1903.02433 [hep-ex].

[17] A. Butter, T. Plehn, and R. Winterhalder, *How to GAN LHC Events*, SciPost Phys. **7** (2019) 075, arXiv:1907.03764 [hep-ph].

[18] Y. Alanazi, N. Sato, T. Liu, W. Melnitchouk, M. P. Kuchera, E. Pritchard, M. Robertson, R. Strauss, L. Velasco, and Y. Li, *Simulation of electron-proton scattering events by a Feature-Augmented and Transformed Generative Adversarial Network (FAT-GAN)*, arXiv:2001.11103 [hep-ph].

[19] A. Butter, T. Plehn, and R. Winterhalder, *How to GAN Event Subtraction*, arXiv:1912.08824 [hep-ph].

[20] M. Paganini, L. de Oliveira, and B. Nachman, *Accelerating Science with Generative Adversarial Networks: An Application to 3D Particle Showers in Multilayer Calorimeters*, Phys. Rev. Lett. **120** (2018) 4, 042003, arXiv:1705.02355 [hep-ex].

[21] M. Paganini, L. de Oliveira, and B. Nachman, *CaloGAN : Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks*, Phys. Rev. **D97** (2018) 1, 014021, arXiv:1712.10321 [hep-ex].

[22] P. Musella and F. Pandolfi, *Fast and Accurate Simulation of Particle Detectors Using Generative Adversarial Networks*, Comput. Softw. Big Sci. **2** (2018) 1, 8, arXiv:1805.00850 [hep-ex].

[23] M. Erdmann, L. Geiger, J. Glombitza, and D. Schmidt, *Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks*, Comput. Softw. Big Sci. **2** (2018) 1, 4, arXiv:1802.03325 [astro-ph.IM].

[24] M. Erdmann, J. Glombitza, and T. Quast, *Precise simulation of electromagnetic calorimeter showers using a Wasserstein Generative Adversarial Network*, Comput. Softw. Big Sci. **3** (2019) 4, arXiv:1807.01954 [physics.ins-det].

[25] ATLAS Collaboration, "Deep generative models for fast shower simulation in ATLAS." ATL-SOFT-PUB-2018-001, 2018. http://cds.cern.ch/record/2630433.

[26] ATLAS Collaboration, "Energy resolution with a GAN for Fast Shower Simulation in ATLAS." ATLAS-SIM-2019-004, 2019. https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PLOTS/SIM-2019-004/.

[27] D. Belayneh *et al.*, *Calorimetry with Deep Learning: Particle Simulation and Reconstruction for Collider Physics*, Eur. Phys. J. C **80** (2020) 7, 688, arXiv:1912.06794 [physics.ins-det].

[28] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol, and K. Krüger, *Getting High: High Fidelity Simulation of High Granularity Calorimeters with High Speed*, arXiv:2005.05334 [physics.ins-det].

[29] E. Bothmann and L. Debbio, *Reweighting a parton shower using a neural network: the final-state case*, JHEP **01** (2019) 033, arXiv:1808.07802 [hep-ph].

[30] L. de Oliveira, M. Paganini, and B. Nachman, *Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis*, Comput. Softw. Big Sci. **1** (2017) 1, 4, arXiv:1701.05927 [stat.ML].

[31] J. W. Monk, *Deep Learning as a Parton Shower*, JHEP **12** (2018) 021, arXiv:1807.03685 [hep-ph].

[32] A. Andreassen, I. Feige, C. Frye, and M. D. Schwartz, *JUNIPR: a Framework for Unsupervised Machine Learning in Particle Physics*, Eur. Phys. J. **C79** (2019) 2, 102, arXiv:1804.09720 [hep-ph].

[33] J. Lin, W. Bhimji, and B. Nachman, *Machine Learning Templates for QCD Factorization in the Search for Physics Beyond the Standard Model*, JHEP **05** (2019) 181, arXiv:1903.02556 [hep-ph].

[34] B. Nachman and D. Shih, *Anomaly Detection with Density Estimation*, Phys. Rev. D **101** (2020) 075042, arXiv:2001.04990 [hep-ph].

[35] O. Knapp, G. Dissertori, O. Cerri, T. Q. Nguyen, J.-R. Vlimant, and M. Pierini, *Adversarially Learned Anomaly Detection on CMS Open Data: re-discovering the top quark*, arXiv:2005.01598 [hep-ex].

[36] K. Datta, D. Kar, and D. Roy, *Unfolding with Generative Adversarial Networks*, arXiv:1806.00433 [physics.data-an].

[37] M. Bellagente, A. Butter, G. Kasieczka, T. Plehn, and R. Winterhalder, *How to GAN away Detector Effects*, SciPost Phys. **8** (2020) 4, 070, arXiv:1912.00477 [hep-ph].

[38] M. Bellagente, A. Butter, G. Kasieczka, T. Plehn, A. Rousselot, R. Winterhalder, L. Ardizzone, and U. Köthe, *Invertible Networks or Partons to Detector and Back Again*, arXiv:2006.06685 [hep-ph].

[39] K. T. Matchev and P. Shyamsundar, *Uncertainties associated with GAN-generated datasets in high energy physics*, arXiv:2002.06307 [hep-ph].

[40] NNPDF, L. Del Debbio, S. Forte, J. I. Latorre, A. Piccione, and J. Rojo, *Unbiased determination of the proton structure function F(2)**p with faithful uncertainty estimation*, JHEP **03** (2005) 080, arXiv:hep-ph/0501067.

[41] Y. Hao, A. Orlitsky, A. T. Suresh, and Y. Wu, *Data amplification: A unified and competitive approach to property estimation*, arXiv:1904.00070 [stat.ML].

[42] C. Canonne, T. Gouleakis, and R. Rubinfeld, *Sampling correctors*, arXiv:1504.06544 [cs.DS].

[43] B. Axelrod, S. Garg, V. Sharan, and G. Valiant, *Sample amplification: Increasing dataset size even when learning is impossible*, arXiv:1904.12053 [cs.LG].

[44] H. Dembinski, P. Ongmongkolkul, C. Deil, D. M. Hurtado, M. Feickert, H. Schreiner, Andrew, C. Burr, F. Rost, A. Pearce, L. Geiger, B. M. Wiedemann, and O. Zapata, *scikit-hep/iminuit: v1.4.9*, July, 2020.

[45] P. Ongmongkolkul, C. Deil, C. hsiang Cheng, A. Pearce, E. Rodrigues, H. Schreiner, M. Marinangeli, L. Geiger, and H. Dembinski, *scikit-hep/probfit: 1.1.0*, Nov., 2018.

[46] A. Paszke *et al.*, *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, Advances in Neural Information Processing Systems 32 (2019) 8024.

[47] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: A simple way to prevent neural networks from overfitting*, Journal of Machine Learning Research **15** (2014) 56, 1929.

[48] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)*, arXiv e-prints (Nov., 2015) , arXiv:1511.07289 [cs.LG].

[49] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola, *Deep sets*, arXiv:1703.06114 [cs.LG].

[50] P. T. Komiske, E. M. Metodiev, and J. Thaler, *Energy Flow Networks: Deep Sets for Particle Jets*, JHEP **01** (2019) 121, arXiv:1810.05165 [hep-ph].

[51] A. L. Maas, A. Y. Hannun, and A. Y. Ng, *Rectifier nonlinearities improve neural network acoustic models*, in *Proceedings of ICML Workshop on Deep Learning for Audio, Speech and Language Processing*. 2013.

[52] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, arXiv:1412.6980 [cs.LG].

[53] K. Roth, A. Lucchi, S. Nowozin, and T. Hofmann, *Stabilizing training of generative adversarial networks through regularization*, Advances in Neural Information Processing Systems (NIPS) (2017) , arXiv:1705.09367.

[54] M. Erdmann, L. Geiger, J. Glombitza, and D. Schmidt, *Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks*, Comput. Softw. Big Sci. **2** (2018) 1, 4, arXiv:1802.03325 [astro-ph.IM].

[55] A. Andreassen and B. Nachman, *Neural Networks for Full Phase-space Reweighting and Parameter Tuning*, Phys. Rev. D **101** (2020) 9, 091901, arXiv:1907.08209 [hep-ph].

[56] A. Andreassen, P. T. Komiske, E. M. Metodiev, B. Nachman, and J. Thaler, *OmniFold: A Method to Simultaneously Unfold All Observables*, Phys. Rev. Lett. **124** (2020) 18, 182001, arXiv:1911.09107 [hep-ph].