Letter

Machine learning amplitudes for faster event generation

Fady Bishara[1,*] and Marc Montull[1,2,3,†]

[1]*Deutsches Elektronen-Synchrotron DESY, Notkestraße 85, 22607 Hamburg, Germany*
[2]*Paul Scherrer Institut, Forschungsstraße 111, 5232 Villigen PSI, Switzerland*
[3]*Physik-Institut, Universität Zürich, Winterthurerstrasse 190, CH-8057 Zürich, Switzerland*

(Received 16 April 2020; revised 22 September 2020; accepted 16 March 2023; published 11 April 2023)

We propose to replace the exact amplitudes used in Monte Carlo event generators for trained machine learning regressors, with the aim of speeding up the evaluation of slow amplitudes. As a proof of concept, we study the process $gg \to ZZ$, whose leading-order amplitude is loop induced. We show that gradient boosting machines like XGBoost can predict the fully differential distributions with errors below 0.1%, and with prediction times $\mathcal{O}(10^3)$ faster than the evaluation of the exact function. This is achieved with training times ~ 23 minutes and regressors of size $\lesssim 22$ Mb. We also find that XGBoost performs well over the entire phase space, while interpolation gives much larger errors in regions where the function is peaked. These results suggest a possible new avenue to speed up Monte Carlo event generators.

DOI: 10.1103/PhysRevD.107.L071901

I. INTRODUCTION

The success of the LHC in discovering the Higgs boson is a testament to the impressive advancements made by the high-energy physics (HEP) community in understanding accelerators, detectors, and to make accurate Standard Model (SM) predictions. As a result, the LHC is rapidly evolving from an *energy frontier machine*, capable of discovering new resonances, to a *precision machine*, capable of measuring small deviations over precise SM predictions.

Due to the key role of higher-order corrections in precision physics, there has been a Herculean effort in recent years to compute, store, and automate higher loop calculations for SM and beyond-the-SM (BSM) processes [1–20]. As impressive as this has been, the widespread usage of these results by the broader high-energy physics community has been relatively low, in part due to the long computing times required to evaluate amplitudes beyond tree level. This evaluation time increases dramatically with the loop order, and makes certain Monte Carlo (MC) event simulations at one loop already unfeasible. Nonetheless, higher loop effects will become more important as the precision from the experimental and theoretical sides keeps improving. This calls for innovations to reduce evaluation times for slow amplitudes. One possible avenue to do just that is to improve the traditional tools and techniques—an effort that is well underway. In this work, however, we take a new and different approach to address these issues.

The main goal of this work is to show that thanks to advances in machine learning (ML) algorithms and tools, it is now possible to train ML regressors with precomputed slow amplitudes, and use them to predict the same amplitudes accurately and in a fraction of the time. In recent years, ML algorithms have increasingly been finding new applications in HEP research. For example, see Refs. [21–51] for concrete applications and Refs. [52–62] for recent reviews.

While we focus on a one-loop-induced process in this work, the ultimate goal is to apply the methods developed here to the finite remainder of two-loop virtual amplitudes. The evaluation of these amplitudes is currently the leading bottleneck in MC event generation at next-to-next-to-leading order (NNLO) [63]. In particular, the slowest amplitudes arise from the interference between the leading (Born) amplitude and the highest loop-order one. Thus, for any given process, these amplitudes will have the lowest final-state multiplicity because they do not involve any extra real emissions. And, since most of the available two-loop amplitudes are for $2 \to 2$ processes, our focus will be on low-dimensional phase spaces. For a practical implementation of our regressors, we only need to generalize our results to four dimensions to allow for off-shell bosons. While this generalization is yet to be fully demonstrated, preliminary results lead us to believe that it will be achievable.

As a proof of concept, we study the $gg \to ZZ$ process, which is loop induced at leading order (see Fig. 1). We find that ML regressors can achieve prediction times of $\mathcal{O}(10^3)$ times faster than traditional tools, while the predicted

*fady.bishara@desy.de
†marc.montull@gmail.com

values for single- and double-differential distributions have errors below 0.1%. This performance is achieved with training times $\lesssim 23$ minutes on a single CPU core, and with a disk size for the trained regressors of a few to tens of megabytes (Mb). We also compare the performance of our ML regressors with interpolation in the Supplemental Material [65]. We find that the regressors perform well over the entire phase space (PS), while interpolation gives larger errors in regions where the function is peaked.

A logical next step after this work would be to test the ML regressors on other processes and implement them into a MC generator: particularly, the aforementioned two-loop virtual amplitudes for diboson processes. We comment on this, and further applications, at the end of this paper. It is also worth emphasizing that the regressors we develop here cover the entire phase space of a given process. This means that they only need to be trained once and used repeatedly for any combination of kinematic selection cuts that could be imposed during event generation. Furthermore, the gain in speed compensates for the training time after the first ~ 150 k evaluations ever.

II. A PROOF OF CONCEPT WITH $gg \to ZZ$

We choose to test the performance of ML regressors in approximating the $gg \to ZZ$ squared amplitude for several reasons. First, the LO contribution to this process arises at one loop (Fig. 1) and is, therefore, relatively slow. Second, it was shown in Ref. [66] that this process contributes the bulk ($\sim 60\%$) of the full NNLO correction of hadronic Z-boson pair production, making its computation imperative when performing phenomenological studies to test the SM or to search for new physics (NP). In addition, it is relevant for NP searches where it constitutes a background to $pp \to ZH$ with H decaying to $\bar{b}b$ or to invisible new particles [67–72]. At the same time, this process is simple enough to avoid unnecessary complications: the squared amplitude only depends on two variables, the center-of-mass energy and the polar angle θ—i.e., $|\mathcal{M}(\sqrt{\hat{s}}, \cos\theta)|^2$. Furthermore, when the pair of Z bosons are exactly on shell, there are no resonant peaks [73]. We leave the study of processes with s-channel resonances for future work. Moreover, since the α_s dependence amounts to an overall rescaling of the squared amplitude, we can approximate the function using a fixed value of α_s and restore the scale dependence afterwards.

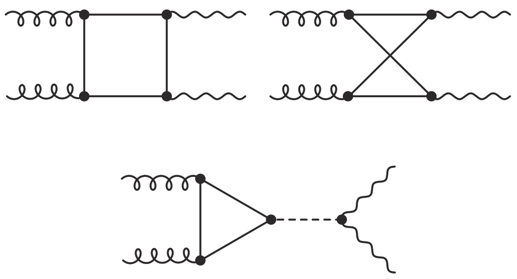

FIG. 1. SM LO diagrams for $gg \to ZZ$, up to fermion momentum flow and crossings.

III. ML ALGORITHM AND TRAINING

A. Choosing a machine learning algorithm

The problem we are trying to address here requires, above all, two features from an ML algorithm: first, it must be able to approximate the true function over the entire domain as accurately as possible; second, it must be able to do so faster than existing dedicated programs, which consume $\sim 5 \times 10^{-3}$ $[s]$ per phase-space point [74]. An additional bonus feature is for the model to be lightweight—i.e., to have a small disk size, $\lesssim \mathcal{O}(100)$ Mb, so that it is easy to distribute quickly.

With this in mind, we evaluated several algorithms suited for regression in the early stages of this work. In particular, we tested deep neural networks (DNNs) with TensorFlow [76], random forests [77–79], and gradient boosting machines (GBMs) [80,81]. From the outset, GBMs as implemented in XGBoost [82] outperformed the others by far in terms of speed and accuracy with very little tuning [83]. Therefore, all the results presented in this paper were obtained with XGBoost via the SCIKIT-LEARN API.

As discussed above, we used the default or close to the default values for the hyperparameters, except for the number of estimators (n), the maximum depth of the trees (m_d), and the learning rate (l_r), for which we performed a small scan: $n \in [10, 1000]$, $m_d \in [10, 800]$, and $l_r \in [0.01, 0.3]$. Based on this bare-bones optimization, the final set of parameters used in this work are given in Table I.

B. Datasets for training and prediction

To train and test the XGBoost regressor, we generated 16.5×10^6 (16.5M) pairs of phase space points, $(\sqrt{\hat{s}}, \cos\theta)$, uniformly distributed in the region defined by

TABLE I. Hyperparameter settings used for all XGBoost regressors in this work. The parameters we attempted to optimize are shown above the split. The values for the parameters below the split are the XGBoost default ones with the exception of “subsample,” see the text for details.

XGBoost parameter	Value
n_estimators	200
max_depth	50
Learning rate	0.1
min_child_weight	1
γ	0
colsample_bytree	1
Subsample	0.75
Booster	gbtree
Objective	reg:squarederror

$$\sqrt{\hat{s}} \in [2\sqrt{m_Z^2 + p_{T,\mathrm{cut}}^2}, 3\ \mathrm{TeV}],$$
$$\cos\theta \in [-1, 1] \times \sqrt{1 - \frac{4p_{T,\mathrm{cut}}^2}{\hat{s} - 4m_Z^2}}. \qquad (1)$$

We then computed the squared matrix element, $\langle|\mathcal{M}|^2\rangle$, using OpenLoops 2 [75]. Here, the angle brackets denote an average over the colors and helicities of the incoming gluons and a sum over the polarizations of the Z bosons. The transverse momentum cut, $p_{T,\mathrm{cut}} = 1$ GeV, is used to regulate a singularity in the limit $p_{T,Z} \to 0$, similarly to what is done in MCFM [86] and Madgraph_aMC@NLO [2]. We chose $(\sqrt{\hat{s}})_{\max} = 3$ TeV as an arbitrary cutoff relevant for LHC physics. Nevertheless, it is straightforward, and inconsequential, to extend the cutoff to the collider center-of-mass energy by adding a new regressor for the extended PS region without a penalty on the precision; we checked this explicitly up to $\sqrt{\hat{s}} = 14$ TeV.

The full sample of 16.5M points was split into training and prediction datasets with 1.5M and 15M points, respectively. To ensure that the datasets are statistically independent, we generated the phase-space points using the PYTHON implementation of the Mersenne twister algorithm [87] which, when initialized properly, has a period of $2^{19937} - 1$.

C. Phase-space partitioning and multiple regressors

The function, $\langle|\mathcal{M}|^2\rangle$, that we are trying to approximate is peaked at the threshold and at $\cos\theta \to \pm 1$ (see the Supplemental Material [65]). This motivates an ansatz to break up the full phase space into smaller subregions of roughly equal phase-space volumes with the purpose of training one regressor per subregion. The aim of this procedure is to have more regressors in regions where the derivatives of the function are large and fewer regressors where the derivatives are small.

For example, choosing $\cos\theta$ and $\sqrt{\hat{s}}$ regions defined by

$$\cos\theta \in \{-1, -0.94, -0.7, 0.7, 0.94, 1\} \times \sqrt{1 - \frac{4p_{T,\mathrm{cut}}^2}{\hat{s} - 4m_Z^2}},$$
$$\sqrt{\hat{s}} \in \{2\sqrt{m_Z^2 + p_{T,\mathrm{cut}}^2}, 1.3\ [\mathrm{TeV}], 3\ [\mathrm{TeV}]\} \qquad (2)$$

partitions the full phase space into ten subregions, each with its own dedicated XGBoost regressor. These partitions are delineated by dashed gray lines in the right half of Fig. 2.

For the remainder of this paper, we will refer to the ansatz with ten regressors as the "ten-region" regressors, and to the one trained on the full domain defined by Eq. (1) as the "one-region" regressor. To compare the performance of the one-region and ten-region regressors, we first train the one-region regressor on a given dataset of size N points.

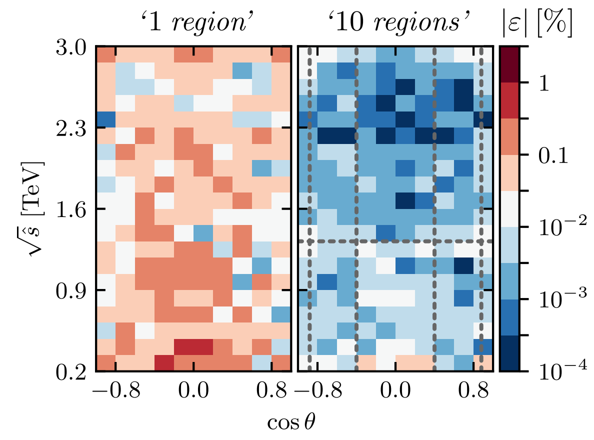

FIG. 2. Absolute value of the percentage relative error per bin of the double differential distribution, $d^2\langle|\mathcal{M}|^2\rangle/d\cos\theta d\sqrt{\hat{s}}$. Each bin has size 141×0.2 (GeV, $\cos\theta$). The total number of training (prediction) points is 1.5M (15M). Left: "one-region" regressor. Right: "ten-region" regressor, where the dashed gray lines delineate the subregions defined in Eq. (2).

We then train each of the ten regressors that make up the ten-region set on a dataset of size $N/10$, such that the total number of training points is the same in both cases.

D. Training time

We benchmark the time it took to train the one-region and each of the ten-region regressors on a single CPU core of an Intel® Xeon® CPU model `E5-2640V4` @ 2.40 GHz on `x86_64` architecture. Since XGBoost can train and predict on multiple cores by default, the times reported here are quite conservative. In practice, modern desktop machines with at least four cores are increasingly common, and so training and prediction times can easily be improved by a factor of a few to 10.

For a training dataset size of 1.5M PS points, the one-region (ten-region) regressors took $\sim 4(23)$ minutes to train. In the ten-region case, we sum the times it took to train each of the ten regressors. The results of the timing tests for the training phase are good fits to simple power laws:

$$\text{One-region}: \ t = 2.45 \times N^{1.35},$$
$$\text{Ten-region}: \ t = 15.5 \times N^{0.957}, \qquad (3)$$

where N is the size of the training set in millions (10^6) and t is the total training time in minutes.

IV. RESULTS

In order to benchmark the trained one-region and ten-region regressors defined above, we study the relative error of their predictions and measure their evaluation times. The relative error at each phase-space point is defined as

$$\varepsilon = \frac{\langle|\mathcal{M}|^2\rangle_{\texttt{OpenLoops}} - \langle|\mathcal{M}|^2\rangle_{\texttt{XGBoost}}}{\langle|\mathcal{M}|^2\rangle_{\texttt{OpenLoops}}}. \tag{4}$$

As for the training times, we measure the prediction times on a single core of the same CPU described above.

A. Accuracy of predictions

Figure 2 shows the relative error, as defined in Eq. (4), on the sum of the predicted $d^2\langle|\mathcal{M}|^2\rangle/d\mathrm{PS}$ values in each bin for the one-region and ten-region regressors. Each of the regressors was trained on 1.5M points, and the relative errors were computed from predictions of 15M points. Each bin has a size of 141 GeV $\times$ 0.2, which is appropriate for phenomenological studies at the LHC. The left and right sides of the plot correspond to the one-region and ten-region regressors, respectively, and they are shown side by side for ease of comparison. In addition, the right panel is overlaid with the boundaries of the subregions defined in Eq. (2). We find that the one-region regressor has a maximum relative error per bin of 0.4%, while the ten-region regressor has a maximum error of 0.07%.

For phenomenological studies, another important distribution is the singly differential one with respect to $\sqrt{\hat{s}}$. The relative error for this distribution is shown in Fig. 3 and is of $\mathcal{O}$(percent) and $\mathcal{O}$(permille) for the one-region and ten-regions cases, respectively.

In order to assess the effect of the size of the training set on the performance of the regressors, we show in Fig. 4 the fraction of points with relative error greater than 1%, 5%, and 10% using the full 15M phase-space point prediction dataset. The dashed (solid) curves correspond to the full (subdivided) phase space. Again, it is clear that subdividing the phase space is very effective in reducing the errors. Figure 4 also shows that, for the chosen hyperparameters (Table I), there is little benefit from using training datasets larger than 1M points. Furthermore, we find no overtraining even up to training datasets of 1.5M points.

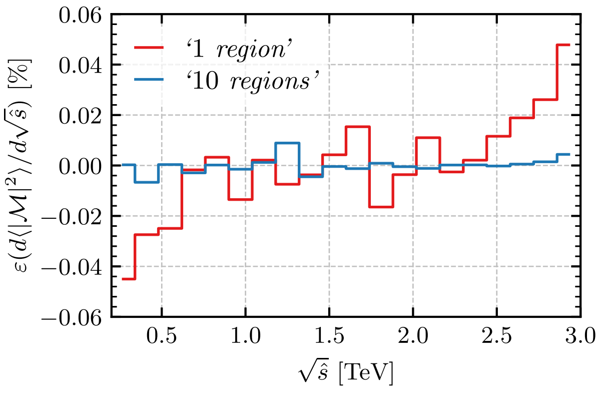

FIG. 3. Relative error of $d\langle|\mathcal{M}|^2\rangle/d\sqrt{\hat{s}}$. The red and the blue curves correspond to the one-region and ten-region regressors, respectively—see Eqs. (1) and (2).

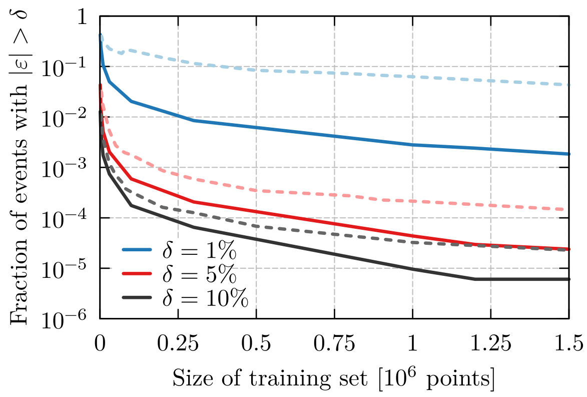

FIG. 4. Percentage of predicted points with an error greater than 1% (blue), 5% (red), and 10% (black) as a function of the number of trained points (the number of predicted points is 15M). The solid (dashed) curves correspond to the ten-region (one-region) regressors.

From the results presented in this section, the benefit of subdividing the phase space and training separate machines on the subregions is clear: the error between the one-region and ten-regions cases is reduced by an order of magnitude (see Figs. 2 and 3), while the prediction time is reduced by a factor of 2 (see Fig. 5).

B. Prediction speed

The time required to predict one phase-space point is a crucial performance metric for the trained regressor. Clearly, it must be much faster than the time required to evaluate the exact function (we use OpenLoops as our benchmark; see Ref. [88]). The results of the timing tests for the training phase are shown in Fig. 5 for both the one-region and ten-region regressors. In addition, a simple power-law fit to the points is shown for each one on the plot. For the ten-region regressor trained on 1M points, the prediction time is $\sim 1 \times 10^{-5}$ seconds in comparison to

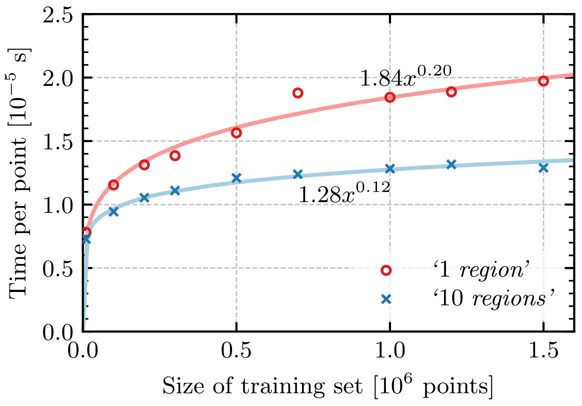

FIG. 5. Prediction time per point as a function of the size of training set. The open circles (crosses) correspond to the results for the one-region (ten-region) regressors. The solid curves are simple power-law fits and are shown in the plot.

8.7×10^{-3} seconds for the Fortran interface of OpenLoops—i.e., a factor of ~ 1000 speedup.

Note that the trained regressors can be packaged as a single, standalone C library. We checked that calling this library during an event generation run has negligible overhead.

C. Disk size

Another desirable feature for the standalone library is to be lightweight in terms of disk size. We find that for 1.5M points, the one-region (ten-region) regressor has a size of 2.6 (22) Mb. This makes these regressors ultraportable, and they could be downloaded on the fly during MC event generation. We envision that using these approximate amplitudes—i.e., regressors—could be given as an option to the user when generating events with popular MC event generators such as MG5_aMCNLO [1], sherpa [89], and whizard [90].

V. SUMMARY AND CONCLUSIONS

The idea of using ML regressors to approximate squared amplitudes proposed in this work is a new application of machine learning techniques in high-energy physics. Our goal is to accurately predict the squared amplitudes using pretrained regressors in a fraction of the time it takes to evaluate the exact ones.

As a proof of concept, we studied the accuracy and speed of XGBoost regressors to predict the squared amplitude for the $gg \to ZZ$ process which is generated at one loop. Our results show that these regressors deliver a thousandfold speedup in evaluation time with respect to OpenLoops with no more than 0.07% error relative to the exact double differential distribution binned as in Fig. 2.

Another convenient feature of the XGBoost regressors studied in this paper is their fast training time. Using the hyperparameters given in Table I, training on 1.5M uniformly generated phase-space points takes about 23 minutes on one CPU core. Moreover, since XGBoost is by default able to train and predict on multicore CPUs, actual training and prediction times will be, in practice, faster by a factor proportional to the number of available CPU cores. In addition, the disk size of the trained regressor for this process is at most 22 Mb, making it easy to distribute on the fly during process generation in MC event generators.

Another important result of this work is to demonstrate that the errors on the predictions of the XGBoost regressor can be reduced by an order of magnitude by training independent regressors on separate subregions of the full phase space. A bonus feature of training more regressors on subregions is that their aggregate prediction time for a given dataset is reduced with respect to training a single regressor on the full phase space.

In Table II, we summarize the aforementioned performance benchmarks for one XGBoost regressor (one-region) trained on the full phase-space region, and for ten regressors (ten-region) each trained on a subregion.

TABLE II. Main characteristics of the two ML regressors trained on 1.5M points and predicting on 15M. The relative errors $\varepsilon_{\max}^{\text{bins}}$ and $\varepsilon_{\min}^{\text{bins}}$ stand for the relative errors in the bins of size 141×0.2 ($\sqrt{\hat{s}}$[GeV] $\times \cos\theta$) from Fig. 2.

	One-region	Ten-region	
$\lvert\varepsilon_{\min}^{\text{bins}}\rvert$ [%]	8×10^{-4}	6×10^{-5}	Fig. 2
$\lvert\varepsilon_{\max}^{\text{bins}}\rvert$ [%]	0.4	0.07	Fig. 2
$t_{\text{predict}}^{(1\text{ core})}[s/\text{point}]$	2×10^{-5}	10^{-5}	Fig. 5
$t_{\text{train}}^{(1\text{ core})}[s]$	254	1374	Fig. 5
Size [Mb]	2.6	22	

The success of the proof of concept studied in this work suggests many applications and further ideas to explore:

(1) Test the performance of ML regressors on qualitatively different channels with slow amplitudes. For instance, *i*) amplitudes with resonant s-channels (see the discussion in Ref. [73]), *ii*) (N)NLO amplitudes, and *iii*) $2 \to n$ processes.
(2) Test and benchmark other ML algorithms.
(3) Implement the trained regressors into an MC event generator. For this, it is necessary to generalize to a four-dimensional phase space to accommodate off-shell gauge bosons; this is of immediate interest as a followup.
(4) On a side note, it would be interesting to test the performance of GBMs on interpolating PDFs and NNLO grids.

ACKNOWLEDGMENTS

We are especially thankful to P. Englert for his contributions during the initial stages of this project. We are happy to thank M. F. Zoller and J. Schlenk for useful discussions. We would like to thank J. Zupan for encouraging us to think about new applications of machine learning in HEP. F. B. was supported by the ERC Starting Grant NewAve No. 638528. M. M. was supported by the Swiss National Science Foundation, under Project No. PP00P2176884. This research was partially supported by the Munich Institute for Astro and Particle Physics (MIAPP), which is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy No. EXC-2094–390783311, the Aspen Center for Physics, which is supported by National Science Foundation Grant No. PHY-1607611, and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC 2121 "Quantum Universe" No. 390833306.

[1] 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, J. High Energy Phys. 07 (2014) 079.

[2] V. Hirschi and O. Mattelaer, Automated event generation for loop-induced processes, J. High Energy Phys. 10 (2015) 146.

[3] R. Boughezal, J. M. Campbell, R. K. Ellis, C. Focke, W. Giele, X. Liu, F. Petriello, and C. Williams, Color singlet production at NNLO in MCFM, Eur. Phys. J. C **77**, 7 (2017).

[4] J. Campbell and T. Neumann, Precision phenomenology with MCFM, J. High Energy Phys. 12 (2019) 034.

[5] Z. Nagy, Next-to-leading order calculation of three jet observables in hadron hadron collision, Phys. Rev. D **68**, 094002 (2003).

[6] Z. Nagy, Three-Jet Cross Sections in Hadron-Hadron Collisions at Next-To-Leading Order, Phys. Rev. Lett. **88**, 122003 (2002).

[7] R. Gavin, Y. Li, F. Petriello, and S. Quackenbush, FEWZ 2.0: A code for hadronic Z production at next-to-next-to-leading order, Comput. Phys. Commun. **182**, 2388 (2011).

[8] S. Camarda *et al.*, DYTurbo: Fast predictions for Drell–Yan processes, Eur. Phys. J. C **80**, 251 (2020).

[9] S. Catani, L. Cieri, G. Ferrera, D. de Florian, and M. Grazzini, Vector boson production at hadron colliders: A fully exclusive QCD calculation at NNLO, Phys. Rev. Lett. **103**, 082001 (2009).

[10] S. Catani, D. de Florian, G. Ferrera, and M. Grazzini, Vector boson production at hadron colliders: Transverse-momentum resummation and leptonic decay, J. High Energy Phys. 12 (2015) 047.

[11] D. de Florian, G. Ferrera, M. Grazzini, and D. Tommasini, Transverse-momentum resummation: Higgs boson production at the Tevatron and the LHC, J. High Energy Phys. 11 (2011) 064.

[12] S. Actis, A. Denner, L. Hofer, J.-N. Lang, A. Scharf, and S. Uccirati, RECOLA: REcursive Computation of One-Loop Amplitudes, Comput. Phys. Commun. **214**, 140 (2017).

[13] V. Hirschi, R. Frederix, S. Frixione, M. V. Garzelli, F. Maltoni, and R. Pittau, Automation of one-loop QCD corrections, J. High Energy Phys. 05 (2011) 044.

[14] Z. Li, J. Wang, Q.-S. Yan, and X. Zhao, Efficient numerical evaluation of Feynman integrals, Chin. Phys. C **40**, 033103 (2016).

[15] L. Naterop, A. Signer, and Y. Ulrich, handyG—Rapid numerical evaluation of generalised polylogarithms in Fortran, Comput. Phys. Commun. **253**, 107165 (2020).

[16] T. Peraro, FiniteFlow: Multivariate functional reconstruction using finite fields and dataflow graphs, J. High Energy Phys. 07 (2019) 031.

[17] Z. Bern, L. J. Dixon, F. Febres Cordero, S. Höche, H. Ita, D. A. Kosower, and D. Maitre, Ntuples for NLO events at hadron colliders, Comput. Phys. Commun. **185**, 1443 (2014).

[18] V. Bertone, S. Carrazza, and N. P. Hartland, APFELgrid: A high performance tool for parton density determinations, Comput. Phys. Commun. **212**, 205 (2017).

[19] D. Maitre, G. Heinrich, and M. Johnson, N(N)LO event files: Applications and prospects, Proc. Sci. LL2016 (**2016**) 016.

[20] M. Spira, Higgs Boson production and decay at hadron colliders, Prog. Part. Nucl. Phys. **95**, 98 (2017).

[21] S. Otten, K. Rolbiecki, S. Caron, J.-S. Kim, R. Ruiz De Austri, and J. Tattersall, DeepXS: Fast approximation of MSSM electroweak cross sections at NLO, Eur. Phys. J. C **80**, 12 (2020).

[22] H. A. Chawdhry, M. L. Czakon, A. Mitov, and R. Poncelet, NNLO QCD corrections to three-photon production at the LHC, J. High Energy Phys. 02 (2020) 057.

[23] M. Bellagente, A. Butter, G. Kasieczka, T. Plehn, and R. Winterhalder, How to GAN away detector effects, SciPost Phys. **8**, 070 (2020).

[24] A. Andreassen and B. Nachman, Neural networks for full phase-space reweighting and parameter tuning, Phys. Rev. D **101**, 091901 (2020).

[25] B. Nachman, A guide for deploying Deep Learning in LHC searches: How to achieve optimality and account for uncertainty, SciPost Phys. **8**, 090 (2020).

[26] J. Bendavid, Efficient Monte Carlo integration using boosted decision trees and generative deep neural networks, arXiv:1707.00028.

[27] S. Otten, S. Caron, W. de Swart, M. van Beekveld, L. Hendriks, C. van Leeuwen, D. Podareanu, R. Ruiz de Austri, and R. Verheyen, Event generation and statistical sampling for physics with deep generative models and a density information buffer, Nat. Commun. **12**, 2985 (2021).

[28] M. D. Klimek and M. Perelstein, Neural network-based approach to phase space integration, SciPost Phys. **9**, 053 (2020).

[29] E. Bothmann and L. Debbio, Reweighting a parton shower using a neural network: The final-state case, J. High Energy Phys. 01 (2019) 033.

[30] R. D. Ball *et al.* (NNPDF Collaboration), Parton distributions from high-precision collider data, Eur. Phys. J. C **77**, 663 (2017).

[31] R. D. Ball *et al.* (NNPDF), Parton distributions for the LHC Run II, J. High Energy Phys. 04 (2015) 040.

[32] R. Santos, M. Nguyen, J. Webster, S. Ryu, J. Adelman, S. Chekanov, and J. Zhou, Machine learning techniques in searches for $t\bar{t}h$ in the $h \to b\bar{b}$ decay channel, J. Instrum. **12**, P04014 (2017).

[33] M. Farina, Y. Nakai, and D. Shih, Searching for new physics with deep autoencoders, Phys. Rev. D **101**, 075021 (2020).

[34] R. T. D'Agnolo and A. Wulzer, Learning new physics from a machine, Phys. Rev. D **99**, 015014 (2019).

[35] A. Blance, M. Spannowsky, and P. Waite, Adversarially-trained autoencoders for robust unsupervised new physics searches, J. High Energy Phys. 10 (2019) 047.

[36] O. Cerri, T. Q. Nguyen, M. Pierini, M. Spiropulu, and J.-R. Vlimant, Variational autoencoders for new physics mining at the large hadron collider, J. High Energy Phys. 05 (2019) 036.

[37] S. Brochet, C. Delaere, B. François, V. Lemaître, A. Mertens, A. Saggio, M. Vidal Marono, and S. Wertz, MoMEMta, a modular toolkit for the Matrix Element Method at the LHC, Eur. Phys. J. C **79**, 126 (2019).

[38] S. Banerjee, R. S. Gupta, J. Y. Reiness, S. Seth, and M. Spannowsky, Towards the ultimate differential SMEFT analysis, J. High Energy Phys. 09 (2020) 170.

[39] P. T. Komiske, E. M. Metodiev, and M. D. Schwartz, Deep learning in color: towards automated quark/gluon jet discrimination, J. High Energy Phys. 01 (2017) 110.
[40] A. Coccaro, M. Pierini, L. Silvestrini, and R. Torre, The DNNLikelihood: Enhancing likelihood distribution with Deep Learning, Eur. Phys. J. C **80**, 664 (2020).
[41] A. Andreassen, I. Feige, C. Frye, and M. D. Schwartz, JUNIPR: A Framework for Unsupervised Machine Learning in Particle Physics, Eur. Phys. J. C **79**, 102 (2019).
[42] A. Andreassen, I. Feige, C. Frye, and M. D. Schwartz, Binary JUNIPR: An interpretable probabilistic model for discrimination, Phys. Rev. Lett. **123**, 182001 (2019).
[43] L. de Oliveira, M. Kagan, L. Mackey, B. Nachman, and A. Schwartzman, Jet-images—deep learning edition, J. High Energy Phys. 07 (2016) 069.
[44] E. M. Metodiev, B. Nachman, and J. Thaler, Classification without labels: Learning from mixed samples in high energy physics, J. High Energy Phys. 10 (2017) 174.
[45] J. H. Collins, K. Howe, and B. Nachman, Anomaly Detection for Resonant New Physics with Machine Learning, Phys. Rev. Lett. **121**, 241803 (2018).
[46] J. H. Collins, K. Howe, and B. Nachman, Extending the search for new resonances with machine learning, Phys. Rev. D **99**, 014038 (2019).
[47] 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**, 182001 (2020).
[48] 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**, 042003 (2018).
[49] 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. D **97**, 014021 (2018).
[50] A. Chakraborty, S. H. Lim, and M. M. Nojiri, Interpretable deep learning for two-prong jet classification with jet spectra, J. High Energy Phys. 07 (2019) 135.
[51] S. H. Lim and M. M. Nojiri, Spectral analysis of jet substructure with neural networks: boosted higgs case, J. High Energy Phys. 10 (2018) 181.
[52] G. Luisoni, S. Poslavsky, and Y. Schroder, Track 3: Computations in theoretical physics—techniques and methods, J. Phys. Conf. Ser. **762**, 012077 (2016).
[53] A. Buckley, Computational challenges for MC event generation, J. Phys. Conf. Ser. **1525**, 012023 (2020).
[54] M. Abdughani, J. Ren, L. Wu, J. M. Yang, and J. Zhao, Supervised deep learning in high energy phenomenology: A mini review, Commun. Theor. Phys. **71**, 955 (2019).
[55] K. Albertsson *et al.*, Machine learning in high energy physics community white paper, J. Phys. Conf. Ser. **1085**, 022008 (2018).
[56] J. Albrecht *et al.* (HEP Software Foundation Collaboration), A roadmap for HEP software and computing R&D for the 2020s, Comput. Softw. Big Sci. **3**, 7 (2019).
[57] S. Carrazza, Machine learning challenges in theoretical HEP, J. Phys. Conf. Ser. **1085**, 022003 (2018).
[58] A. J. Larkoski, I. Moult, and B. Nachman, Jet substructure at the Large Hadron Collider: A review of recent advances in theory and machine learning, Phys. Rep. **841**, 1 (2020).
[59] D. Bourilkov, Machine and deep learning applications in particle physics, Int. J. Mod. Phys. A **34**, 1930019 (2020).
[60] J. Brehmer, F. Kling, I. Espejo, and K. Cranmer, MadMiner: Machine learning-based inference for particle physics, Comput. Softw. Big Sci. **4**, 3 (2020).
[61] J. Brehmer, S. Dawson, S. Homiller, F. Kling, and T. Plehn, Benchmarking simplified template cross sections in WH production, J. High Energy Phys. 11 (2019) 034.
[62] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, Machine learning and the physical sciences, Rev. Mod. Phys. **91**, 045002 (2019).
[63] A relevant example is diboson mediated four-lepton production, $pp \to V_1 V_2 \to 4\ell$. More specifically, Table 11 in the journal version of Ref. [64] shows that generating $pp \to 4e$ at NNLO takes more than a thousand times longer than at NLO.
[64] M. Grazzini, S. Kallweit, and M. Wiesemann, Fully differential NNLO computations with MATRIX, Eur. Phys. J. C **78**, 537 (2018).
[65] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevD.107.L071901 for additional details regarding the shape of the squared matrix element, comparison with interpolation, and the continuity of the approximated function across the boundaries of the phase-space subregions.
[66] F. Cascioli, T. Gehrmann, M. Grazzini, S. Kallweit, P. Maierhöfer, A. von Manteuffel, S. Pozzorini, D. Rathlev, L. Tancredi, and E. Weihs, ZZ production at hadron colliders in NNLO QCD, Phys. Lett. B **735**, 311 (2014).
[67] V. Khachatryan *et al.* (CMS Collaboration), Searches for invisible decays of the Higgs boson in pp collisions at $\sqrt{s} = 7, 8$, and $13 TeV$, J. High Energy Phys. 02 (2017) 135.
[68] M. Aaboud *et al.* (ATLAS Collaboration), Observation of $H \to b\bar{b}$ decays and VH production with the ATLAS detector, Phys. Lett. B **786**, 59 (2018).
[69] A. M. Sirunyan *et al.* (CMS Collaboration), Observation of Higgs boson Decay to Bottom Quarks, Phys. Rev. Lett. **121**, 121801 (2018).
[70] S. Banerjee, C. Englert, R. S. Gupta, and M. Spannowsky, Probing Electroweak Precision Physics via boosted Higgs-strahlung at the LHC, Phys. Rev. D **98**, 095012 (2018).
[71] S. Banerjee, R. S. Gupta, J. Y. Reiness, and M. Spannowsky, Resolving the tensor structure of the Higgs coupling to Z-bosons via Higgs-strahlung, Phys. Rev. D **100**, 115004 (2019).
[72] G. Arcadi, A. Djouadi, and M. Raidal, Dark Matter through the Higgs portal, Phys. Rep. **842**, 1 (2020).
[73] We envision dealing with Breit-Wigner resonances in two ways. For higher-order squared amplitudes, one can normalize by a lower order (or even the Born-level) amplitude—i.e., to produce a fully differential k factor. Another possibility is to approximate the amplitude itself, rather than its square, and decompose it into subamplitudes where the resonant propagators can be factored out. The need to address this case arises when considering four-lepton

production via an off-shell diboson pair, and its detailed investigation is the subject of future work.
[74] We determined this evaluation time using OpenLoops [75] to generate the exact $gg \to ZZ$ squared amplitudes for the training and prediction sets. The CPU core used for this timing is the same one used for all timings in this paper; see the main text for the exact specification.
[75] F. Buccioni, J.-N. Lang, J. M. Lindert, P. Maierhöfer, S. Pozzorini, H. Zhang, and M. F. Zoller, OpenLoops 2, Eur. Phys. J. C **79**, 866 (2019).
[76] M. Abadi *et al.*, TensorFlow: Large-scale machine learning on heterogeneous systems (2015), software available from tensorflow.org.
[77] M. N. Wright and A. Ziegler, ranger: A fast implementation of random forests for high dimensional data in $C++$ and R, J. Stat. Softw. **77**, 1 (2017).
[78] T. K. Ho, in *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1)—Volume 1*, ICDAR '95 (IEEE Computer Society, Washington, DC, USA, 1995), p. 278.
[79] L. Breiman, Random forests, Mach. Learn. **45**, 5 (2001).
[80] L. Breiman, Arcing the edge, Technical Report 486, Statistics Department, University of California at Berkeley, 1997.
[81] J. H. Friedman, Greedy function approximation: A gradient boosting machine, Ann. Stat. **29**, 1189 (2001).
[82] T. Chen and C. Guestrin, XGBoost, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining—KDD 'Z16* (2016), 10.1145/2939672.2939785.
[83] XGBoost rose to prominence by winning the Kaggle Higgs Challenge. Since then, it has been consistently on the top of the ladder in a large number of the Kaggle competitions [84,85], outperforming other ML architectures, like DNNs or SVMs.
[84] https://www.kaggle.com/sudalairajkumar/winning-solutions-of-kaggle-competitions.
[85] https://www.kaggle.com/rajiao/winning-solutions-of-kaggle-competitions.
[86] J. M. Campbell and R. K. Ellis, MCFM for the Tevatron and the LHC, Nucl. Phys. B, Proc. Suppl. **205–206**, 10 (2010).
[87] M. Matsumoto and T. Nishimura, Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator, ACM Trans. Model. Comput. Simul. **8**, 3 (1998).
[88] We also tested MadLoops [2] but found it to be a factor of a few slower than OpenLoops [75].
[89] E. Bothmann *et al.*, Event generation with Sherpa 2.2, SciPost Phys. **7**, 034 (2019).
[90] W. Kilian, S. Brass, T. Ohl, J. Reuter, V. Rothe, P. Stienemeier, and M. Utsch, New Developments in WHIZARD Version 2.6, in *Proceedings of the International Workshop on Future Linear Collider (LCWS2017) Strasbourg, France, 2017* (2018), arXiv:1801.08034.