DEUTSCHES ELEKTRONEN-SYNCHROTRON
Ein Forschungszentrum der Helmholtz-Gemeinschaft

DESY 19-232
arXiv:1912.11055
December 2019

Machine Learning Amplitudes for Faster Event Generation

F. Bishara

Deutsches Elektronen-Synchrotron DESY, Hamburg

M. Montull

Deutsches Elektronen-Synchrotron DESY, Hamburg

and

Paul Scherrer Institut, Villigen PSI, Switzerland

and

Physik-Institut, Universität Zürich, Switzerland

ISSN 0418-9833

NOTKESTRASSE 85 – 22607 HAMBURG

DESY 19-232

Machine Learning amplitudes for faster event generation

Fady Bishara[1, *] and Marc Montull[1, 2, 3, †]
[1]DESY, Notkestraße 85, 22607 Hamburg, Germany
[2]Paul Scherrer Institut, Forschungsstrasse 111, 5232 Villigen PSI, Swizerland
[3]Physik-Institut, Universität Zürich, Winterthurerstrasse 190, CH-8057 Zürich, Switzerland

We propose to replace the exact amplitudes used in MC 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 LO 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 ~ 7 minutes and regressors of size $\lesssim 30$ Mb. These results suggest a possible new avenue to speed up MC event generators.

INTRODUCTION

The success of the LHC in discovering the Higgs boson is a testament to the impressive advancements made by the 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 an Herculean effort in recent years to compute, store, and automate higher loop calculations for SM and Beyond the SM (BSM) predictions [1–20]. As impressive as this has been, the use of N(N)LO results by the broader HEP community has been relatively low, in part due to the long 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 under way. 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 the advances in Machine Learning (ML) algorithms and tools, it is now possible to train ML regressors with pre-computed *slow* amplitudes, and use them to predict the same amplitudes accurately and in a fraction of the time.

As a proof of concept we study the $gg \to ZZ$ process which is loop induced at LO (see Fig. 1). We find that ML regressors can achieve prediction times $\mathcal{O}(10^3)$ faster than traditional tools while the predicted values for single and double differential distributions have errors below 0.1%. This was achieved with training times $\lesssim 7$ minutes on a single CPU core, and with a disk size for the trained regressors of a few to tens of megabytes (Mb).

Machine Learning algorithms are constantly finding new applications in HEP research (see [21–49] for concrete applications and [50–60] for recent reviews). Nonetheless, we are not aware of any work where these tools have been used to speed up time consuming amplitudes [61].

There are many processes where speeding up MC event generation can be immediately useful. Therefore, a next step after this work would be to test the ML regressors on other processes and implement them into a MC generator. We comment on further applications at the end of this letter.

A PROOF OF CONCEPT WITH $gg \to ZZ$

We chose to test the performance of ML regressors in approximating the $gg \to ZZ$ squared amplitude for several reasons. First of all, the LO contribution to this process arises at one loop (Fig. 1) and therefore is *slow*. Secondly, it was shown in Ref. [62] 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 phenomenlogical studies to test the SM or to search for New Physics (NP). In addition, it is relevant for NP searches where it consti-

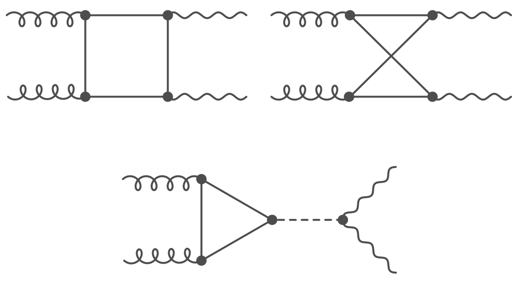

Figure 1. SM LO diagrams for $gg \to ZZ$, up to fermion momentum flow and crossings.

tutes a background to $pp \to ZH$ with H decaying to $\bar{b}b$ or to invisible new particles [63–68]. 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 the pair of ZZ bosons are on-shell, there are no resonant peaks. We leave the study of processes with s-channel resonances for future work. Furthermore, since the α_s dependence amounts to an overall rescaling of the amplitude squared, we can approximate the function using a fixed value of α_s and restore the scale dependence afterwards.

ML ALGORITHM AND TRAINING

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 be able to do so faster than existing dedicated programs $\sim 5 \cdot 10^{-3}$ $[s]$ per phase space (PS) point [69]. 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 (DNN) with `TensorFlow` [70], random forests [71–73], and gradient boosting machines (GBM) [74, 75]. From the outset, GBMs as implemented in `XGBoost` [76] outperformed the others by far in terms of speed, accuracy, and robustness against overfitting with very little tuning [77]. Therefore, all the results presented in this letter were obtained with `XGBoost` via the `scikit-learn` API.

As discussed above, we use the default or close to the default values for the hyper-parameters, except for the number of estimators (n), 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.

Datasets for Training and Prediction

To train and test the `XGBoost` regressor, we generated 18 million (18M) pairs of phase space points, $(\sqrt{\hat{s}}, \cos\theta)$, uniformly distributed in the region defined by,

$$\sqrt{\hat{s}} \in \left[2\sqrt{m_Z^2 + p_{T,\mathrm{cut}}^2}, 3\,\mathrm{TeV}\right], \quad \cos\theta \in [-1, 1] \times \sqrt{1 - \frac{4p_{T,\mathrm{cut}}^2}{\hat{s} - 4m_Z^2}}, \tag{1}$$

with $p_T^{\mathrm{cut}} = 1$ GeV to regulate the singularity in $\langle|\mathcal{M}|^2\rangle$ in the limit $p_T \to 0$ (similarly to what is done in `MCFM` [78] 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; we checked this explicitly up to 14 TeV.

We then computed the corresponding squared amplitudes required for training and testing `XGBoost` , using the `OpenLoops` software [79]. The full sample of 18M points was split into a training and prediction dataset with 3M and 15M points, respectively. To ensure that the data sets are statistically independent, we generated the phase space points using the `Python` implementation of the Mersenne Twister algorithm [80] which, when initialized properly, has a period of $2^{19937} - 1$.

Phase Space partitioning and multiple regressors

The function that we are trying to approximate, $\langle|\mathcal{M}|^2\rangle$, is peaked at $\cos\theta \to \pm 1$. This motivates an ansatz to break up the full phase space into smaller sub-regions with roughly equal integrated $d\langle|\mathcal{M}|^2\rangle/d\mathrm{PS}$ with the purpose of training one regressor per sub-region.

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

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

`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

Table I. Hyper-parameter settings used for all `XGBoost` regressors in this work. The parameters we attempted to optimize are shown above the split while the values for the parameters below the split are the `XGBoost` default ones with the exception of 'subsample', see text for details.

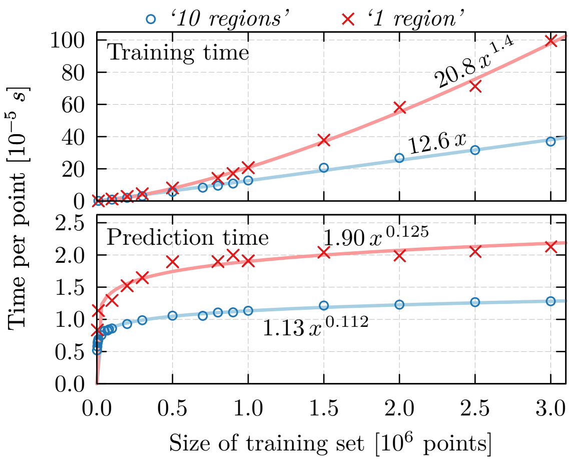

Figure 2. Prediction time per point as a function of the size of training set. The crosses (open circles) correspond to the results for the *'1 region'* (*'10 regions'*) regressors. The solid curves are simple power law fits and are shown in the legend.

with the idea of decreasing as much as possible the variation of the squared amplitude in each sub-region. This partitions the full phase space into 10 sub-regions each with its own dedicated `XGBoost` regressor that is trained on, and predicts in, only one sub-region. These partitions are delineated by dashed gray lines in the right half of Fig. 3.

For the remainder of this letter, we will refer to the ansatz with 10 regressors as the *'10 regions'* regressor, and to the one trained on the full domain defined by Eq. (1) as *'1 region'* .

To compare the performance of the *'1 region'* and *'10 regions'* regressors, we first train the *'1 region'* regressor on a given dataset. Then, to train each of the ten regressors that make up the *'10 regions'* regressor, we split the same dataset according to the regions defined in Eq. (2). In the end, each of these ten regressors making the *'10 regions'* regressor is only trained on a fraction between 2.4% and 24% of the total dataset, corresponding to the fraction of its phase space area (since the PS is uniformly sampled).

Training time

We benchmark the time it took to train the *'1 region'* and each of the *'10 regions'* 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 be easily be improved by a factor of a few to ten.

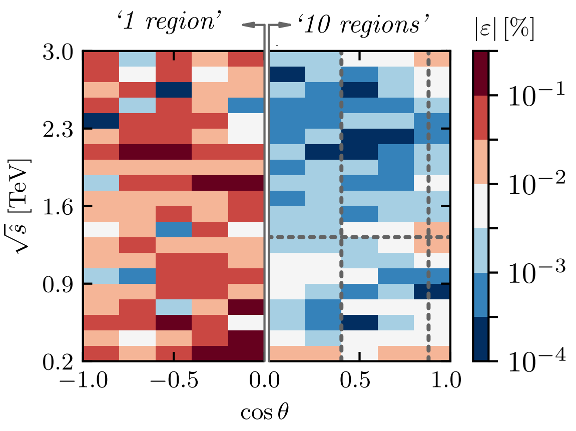

Figure 3. Absolute value of the percentage relative error per bin of the double differential distribution, $d^2\langle|\mathcal{M}|^2\rangle/d\cos\theta d\sqrt{\hat{s}}$. Each bin has size 140×0.2 (GeV, $\cos\theta$). The total number of training (prediction) points is 3M (16M). **Left:** *'1 region'* regressor. **Right:** *'10 regions'* regressor with dashed gray lines showing the sub-regions defined in Eq. (2).

The results of the timing tests for the training phase are shown in Fig. 2 (top panel) for both the *'1 region'* and *'10 regions'* regressors; in addition, a simple power law fit to the points is given. For a training dataset size of 3M PS points, the *'1 region'* (*'10 regions'*) regressors took $\sim 16\,(7)$ minutes to train. In the case of the *'10 regions'* regressors – there are 10 of them – we added up the times it took to train each one of them.

RESULTS

In order to benchmark the trained *'1 region'* and *'10 regions'* regressors defined above, we study the relative error of their predictions, and measure their evaluation times. The relative error is defined as,

$$\varepsilon = \frac{\langle|\mathcal{M}|^2\rangle_{\texttt{OpenLoops}} - \langle|\mathcal{M}|^2\rangle_{\texttt{XGBoost}}}{\langle|\mathcal{M}|^2\rangle_{\texttt{OpenLoops}}}\,. \qquad (3)$$

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

Accuracy of predictions

Figure 3 shows the relative error on the sum of the predicted $d^2\langle|\mathcal{M}|^2\rangle/d\text{PS}$ values in each bin for the *'1 region'* and *'10 regions'* regressors. Each of the regressors was trained on 3M points and the relative errors were computed from predictions of 15M points. Each bin has a size of 140 GeV $\times\,0.2$, which is appropriate for phenomenological studies at the LHC. The left and

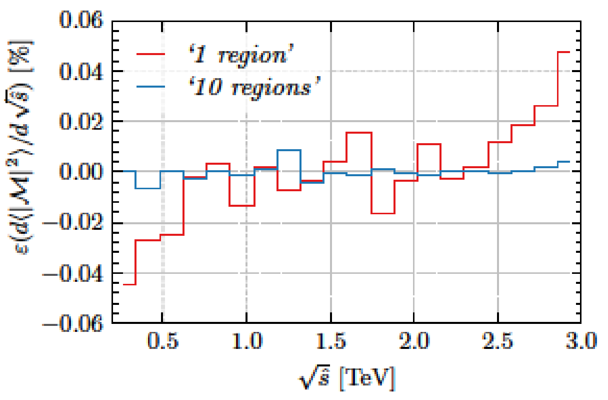

Figure 4. Relative error of $d\langle|\mathcal{M}|^2\rangle/d\sqrt{\hat{s}}$. The red and the blue curves correspond to the *'1 region'* and *'10 regions'* regressors respectively (see Sec.).

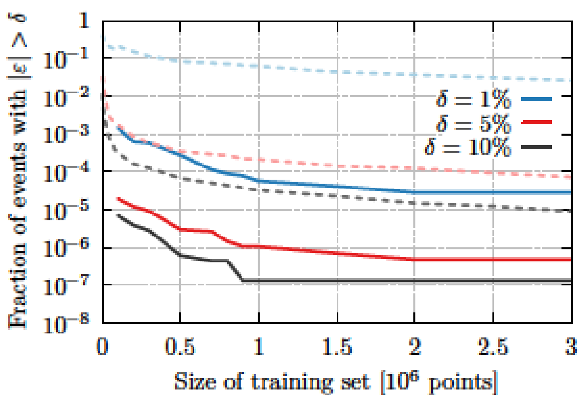

Figure 5. Percentage of predicted points with an error greater than 1% (blue), 5% (red), and 10% (black) as a function of the number of trained points (the number of predicted points is 15 million). The solid (dashed) curves correspond to the *'10 regions'* (*'1 region'*) regressors.

right sides of the plot correspond to the *'1 region'* and *'10 regions'* regressors, respectively, put together for easier comparison since the amplitude is symmetric under $\cos\theta \to -\cos\theta$. In addition, the right panel is overlaid with the boundaries of the sub-regions defined in Eq. (2). We find that the *'1 region'* regressor has a maximum relative error per bin of 0.3% while the *'10 regions'* regressor has a maximum error of 0.03%.

For phenomenological studies, another important differential distribution is the singly differential one with respect to $\sqrt{\hat{s}}$. The relative error for this distribution is shown in Fig. 4 and is of $\mathcal{O}$(percent) and $\mathcal{O}$(permille) for the *'1 region'* and *'10 regions'* regions respectively.

In order to assess the effect of the size of the training set on the performance of the machines, we show in Fig. 5 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 5 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 over-training even up to training datasets of 3M points.

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 *'1 region'* and *'10 regions'* is reduced by an order of magnitude, see Fig. 3 and Fig. 4, while the training and prediction times are reduced by a factor of two, see Fig. 2.

Prediction speed

The time required to predict one phase space point is a crucial performance metric for the trained machine. Clearly it must be much faster than the time required to evaluate the true function (we use `OpenLoops` as our benchmark). The results of the timing tests for the training phase are shown in Fig. 2 for both the *'1 region'* and *'10 regions'* regressors. In addition, a simple power law fit to the points is shown for each one on the plot. For the *'10 regions'* regressor trained on 1M points, the prediction time is $\sim 1 \times 10^{-5}$ seconds in comparison to 8.7×10^{-3} seconds for `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.

Disk size

Another desirable feature for the standalone packaged library is to be lightweight in terms of disk size. We find that for 1M points, the *'1 region'* (*'10 regions'*) regressor has a size is 2.6 (19) Megabytes. This makes these regressors ultra portable and could be downloaded on the fly during MC event generation – we envision machines of this type to be an option given to the user when generating events with popular MC event generators such as `MG5_aMCNLO` [1], `Sherpa` [81], and `Whizard` [82].

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 HEP. Our goal is to accurately predict the trained squared-amplitudes 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 the `XGBoost` regressor to predict the squared amplitudes for the $gg \to ZZ$ process which at LO is generated at one loop. Our results show that the `XGBoost` regressors deliver a **1000**-fold speedup in evaluation time with respect to `OpenLoops` with no more than **0.03**% relative error with respect to the *true* double differential distribution binned as in Fig. 3.

Another convenient feature of the `XGBoost` regressor studied in this letter, is its reduced training speed. Using the hyper-parameters given Table I, training on 1M uniformly generated PS points takes about **2** minutes on one CPU core. Moreover, since `XGBoost` is by default able to train and predict on multi-core CPUs, actual training and prediction times will be in practice faster by a factor proportional to the number of available CPU cores. `XGBoost` can also run on GPUs with some minor modifications, otherwise, `LightGBM` [83] works on GPUs by default and could even be a better performing regressor.

In addition, the disk size of the trained `XGBoost` regressor for this process, is at most **30** 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 sub-regions of the full phase space. A bonus feature of training more regressors on sub-regions is that their aggregate training and prediction times for a given dataset are 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 (*'1 region'*) trained on the full phase space region, and for ten regressors (*'10 regions'*) each trained on a sub-region.

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

- To test the performance of ML regressors on qualitatively different channels with *slow* amplitudes. For instance: *i)* Amplitudes with resonant s-channels *ii)* N(N)LO amplitudes *iii)* $2 \to n$ processes.
- Test and benchmark other ML algorithms.

	'1 region'	*'10 regions'*	
$\lvert\varepsilon^{\rm bins}_{min}\rvert$ [%]	$7 \cdot 10^{-5}$	$3 \cdot 10^{-5}$	Fig. 3
$\lvert\varepsilon^{\rm bins}_{max}\rvert$ [%]	0.3	0.03	Fig. 3
$t^{(1\text{ core})}_{\rm predict}$ [s/point]	$2 \cdot 10^{-5}$	10^{-5}	Fig. 2
$t^{(1\text{ core})}_{\rm train}$ [s]	977	390	Fig. 2
Size [Mb]	4.8	28	

Table II. Main characteristics of the two ML regressors trained on 3M points and predicting on 15M. The relative errors $\varepsilon^{\rm bins}_{max}$ and $\varepsilon^{\rm bins}_{min}$ stand for the relative errors in the bins of size 140 $\times$ 0.2 ($\sqrt{\hat{s}}$ [GeV] $\times \cos\theta$).

- Implement the trained ML regressors into an MC event generator.
- On a side note, it would be interesting to test the performance of GBM on interpolating PDFs and NNLO grids.

Acknowledgements. 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 (638528). M.M. was supported by the Swiss National Science Foundation, under Project Nos. 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 – EXC-2094 390783311, the Aspen Center for Physics, which is supported by National Science Foundation grant PHY-1607611, and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy EXC 2121 "Quantum Universe" – 390833306.

[*] Electronic address:fady.bishara@desy.de
[†] Electronic address:marc.montull@gmail.com

[1] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli, and M. Zaro, JHEP **07**, 079 (2014), arXiv:1405.0301 [hep-ph].
[2] V. Hirschi and O. Mattelaer, JHEP **10**, 146 (2015), arXiv:1507.00020 [hep-ph].
[3] R. Boughezal, J. M. Campbell, R. K. Ellis, C. Focke, W. Giele, X. Liu, F. Petriello, and C. Williams, Eur. Phys. J. **C77**, 7 (2017), arXiv:1605.08011 [hep-ph].
[4] J. Campbell and T. Neumann, (2019), arXiv:1909.09117 [hep-ph].
[5] Z. Nagy, Phys. Rev. **D68**, 094002 (2003), arXiv:hep-ph/0307268 [hep-ph].

[6] Z. Nagy, Phys. Rev. Lett. **88**, 122003 (2002), arXiv:hep-ph/0110315 [hep-ph].

[7] R. Gavin, Y. Li, F. Petriello, and S. Quackenbush, Comput. Phys. Commun. **182**, 2388 (2011), arXiv:1011.3540 [hep-ph].

[8] S. Camarda *et al.*, (2019), arXiv:1910.07049 [hep-ph].

[9] S. Catani, L. Cieri, G. Ferrera, D. de Florian, and M. Grazzini, Phys. Rev. Lett. **103**, 082001 (2009), arXiv:0903.2120 [hep-ph].

[10] S. Catani, D. de Florian, G. Ferrera, and M. Grazzini, JHEP **12**, 047 (2015), arXiv:1507.06937 [hep-ph].

[11] D. de Florian, G. Ferrera, M. Grazzini, and D. Tommasini, JHEP **11**, 064 (2011), arXiv:1109.2109 [hep-ph].

[12] S. Actis, A. Denner, L. Hofer, J.-N. Lang, A. Scharf, and S. Uccirati, Comput. Phys. Commun. **214**, 140 (2017), arXiv:1605.01090 [hep-ph].

[13] V. Hirschi, R. Frederix, S. Frixione, M. V. Garzelli, F. Maltoni, and R. Pittau, JHEP **05**, 044 (2011), arXiv:1103.0621 [hep-ph].

[14] Z. Li, J. Wang, Q.-S. Yan, and X. Zhao, Chin. Phys. **C40**, 033103 (2016), arXiv:1508.02512 [hep-ph].

[15] L. Naterop, A. Signer, and Y. Ulrich, (2019), arXiv:1909.01656 [hep-ph].

[16] T. Peraro, JHEP **07**, 031 (2019), arXiv:1905.08019 [hep-ph].

[17] Z. Bern, L. J. Dixon, F. Febres Cordero, S. Hche, H. Ita, D. A. Kosower, and D. Maitre, Comput. Phys. Commun. **185**, 1443 (2014), arXiv:1310.7439 [hep-ph].

[18] V. Bertone, S. Carrazza, and N. P. Hartland, Comput. Phys. Commun. **212**, 205 (2017), arXiv:1605.02070 [hep-ph].

[19] D. Maitre, G. Heinrich, and M. Johnson, *Proceedings, 13th DESY Workshop on Elementary Particle Physics: Loops and Legs in Quantum Field Theory (LL2016): Leipzig, Germany, April 24-29, 2016*, PoS **LL2016**, 016 (2016), arXiv:1607.06259 [hep-ph].

[20] M. Spira, Prog. Part. Nucl. Phys. **95**, 98 (2017), arXiv:1612.07651 [hep-ph].

[21] M. Bellagente, A. Butter, G. Kasieczka, T. Plehn, and R. Winterhalder, (2019), arXiv:1912.00477 [hep-ph].

[22] A. Andreassen and B. Nachman, (2019), arXiv:1907.08209 [hep-ph].

[23] B. Nachman, (2019), arXiv:1909.03081 [hep-ph].

[24] J. Bendavid, (2017), arXiv:1707.00028 [hep-ph].

[25] S. Otten, S. Caron, W. de Swart, M. van Beekveld, L. Hendriks, C. van Leeuwen, D. Podareanu, R. Ruiz de Austri, and R. Verheyen, (2019), arXiv:1901.00875 [hep-ph].

[26] M. D. Klimek and M. Perelstein, (2018), arXiv:1810.11509 [hep-ph].

[27] E. Bothmann and L. Debbio, JHEP **01**, 033 (2019), arXiv:1808.07802 [hep-ph].

[28] R. D. Ball *et al.* (NNPDF), Eur. Phys. J. **C77**, 663 (2017), arXiv:1706.00428 [hep-ph].

[29] R. D. Ball *et al.* (NNPDF), JHEP **04**, 040 (2015), arXiv:1410.8849 [hep-ph].

[30] R. Santos, M. Nguyen, J. Webster, S. Ryu, J. Adelman, S. Chekanov, and J. Zhou, JINST **12**, P04014 (2017), arXiv:1610.03088 [hep-ex].

[31] M. Farina, Y. Nakai, and D. Shih, (2018), arXiv:1808.08992 [hep-ph].

[32] R. T. D'Agnolo and A. Wulzer, Phys. Rev. **D99**, 015014 (2019), arXiv:1806.02350 [hep-ph].

[33] A. Blance, M. Spannowsky, and P. Waite, JHEP **10**, 047 (2019), arXiv:1905.10384 [hep-ph].

[34] O. Cerri, T. Q. Nguyen, M. Pierini, M. Spiropulu, and J.-R. Vlimant, JHEP **05**, 036 (2019), arXiv:1811.10276 [hep-ex].

[35] S. Brochet, C. Delaere, B. Franois, V. Lematre, A. Mertens, A. Saggio, M. Vidal Marono, and S. Wertz, Eur. Phys. J. **C79**, 126 (2019), arXiv:1805.08555 [hep-ph].

[36] S. Banerjee, R. S. Gupta, J. Y. Reiness, S. Seth, and M. Spannowsky, (2019), arXiv:1912.07628 [hep-ph].

[37] P. T. Komiske, E. M. Metodiev, and M. D. Schwartz, JHEP **01**, 110 (2017), arXiv:1612.01551 [hep-ph].

[38] A. Coccaro, M. Pierini, L. Silvestrini, and R. Torre, (2019), arXiv:1911.03305 [hep-ph].

[39] A. Andreassen, I. Feige, C. Frye, and M. D. Schwartz, Eur. Phys. J. **C79**, 102 (2019), arXiv:1804.09720 [hep-ph].

[40] A. Andreassen, I. Feige, C. Frye, and M. D. Schwartz, Phys. Rev. Lett. **123**, 182001 (2019), arXiv:1906.10137 [hep-ph].

[41] L. de Oliveira, M. Kagan, L. Mackey, B. Nachman, and A. Schwartzman, JHEP **07**, 069 (2016), arXiv:1511.05190 [hep-ph].

[42] E. M. Metodiev, B. Nachman, and J. Thaler, JHEP **10**, 174 (2017), arXiv:1708.02949 [hep-ph].

[43] J. H. Collins, K. Howe, and B. Nachman, Phys. Rev. Lett. **121**, 241803 (2018), arXiv:1805.02664 [hep-ph].

[44] J. H. Collins, K. Howe, and B. Nachman, Phys. Rev. **D99**, 014038 (2019), arXiv:1902.02634 [hep-ph].

[45] A. Andreassen, P. T. Komiske, E. M. Metodiev, B. Nachman, and J. Thaler, (2019), arXiv:1911.09107 [hep-ph].

[46] M. Paganini, L. de Oliveira, and B. Nachman, Phys. Rev. Lett. **120**, 042003 (2018), arXiv:1705.02355 [hep-ex].

[47] M. Paganini, L. de Oliveira, and B. Nachman, Phys. Rev. **D97**, 014021 (2018), arXiv:1712.10321 [hep-ex].

[48] A. Chakraborty, S. H. Lim, and M. M. Nojiri, JHEP **19**, 135 (2020), arXiv:1904.02092 [hep-ph].

[49] S. H. Lim and M. M. Nojiri, JHEP **10**, 181 (2018), arXiv:1807.03312 [hep-ph].

[50] G. Luisoni, S. Poslavsky, and Y. Schroder, *Proceedings, 17th International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT 2016): Valparaiso, Chile, January 18-22, 2016*, J. Phys. Conf. Ser. **762**, 012077 (2016), arXiv:1604.03370 [hep-ph].

[51] A. Buckley, in *19th International Workshop on Advanced Computing and Analysis Techniques in Physics Research: Empowering the revolution: Bringing Machine Learning to High Performance Computing (ACAT 2019) Saas-Fee, Switzerland, March 11-15, 2019* (2019) arXiv:1908.00167 [hep-ph].

[52] M. Abdughani, J. Ren, L. Wu, J. M. Yang, and J. Zhao, Commun. Theor. Phys. **71**, 955 (2019), arXiv:1905.06047 [hep-ph].

[53] K. Albertsson *et al.*, *Proceedings, 18th International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT 2017): Seattle, WA, USA, August 21-25, 2017*, J. Phys. Conf. Ser. **1085**, 022008 (2018), arXiv:1807.02876 [physics.comp-ph].

[54] J. Albrecht *et al.* (HEP Software Foundation), Comput. Softw. Big Sci. **3**, 7 (2019), arXiv:1712.06982 [physics.comp-ph].

[55] S. Carrazza, *Proceedings, 18th International Workshop on Advanced Computing and Analysis Techniques in*

Physics Research (ACAT 2017): Seattle, WA, USA, August 21-25, 2017, J. Phys. Conf. Ser. **1085**, 022003 (2018), arXiv:1711.10840 [hep-ph].

[56] A. J. Larkoski, I. Moult, and B. Nachman, (2017), arXiv:1709.04464 [hep-ph].

[57] D. Bourilkov, (2019), arXiv:1912.08245 [physics.data-an].

[58] J. Brehmer, F. Kling, I. Espejo, and K. Cranmer, (2019), arXiv:1907.10621 [hep-ph].

[59] J. Brehmer, S. Dawson, S. Homiller, F. Kling, and T. Plehn, JHEP **11**, 034 (2019), arXiv:1908.06980 [hep-ph].

[60] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborov, Rev. Mod. Phys. **91**, 045002 (2019), arXiv:1903.10563 [physics.comp-ph].

[61] The only similar works we found is Ref. [84] where DNN are used to predict NLO cross sections for the pMSSM-19 and Ref. [85] ML regressors were used as interpolators to store the NNLO QCD amplitude for $pp \to 3\gamma$.

[62] F. Cascioli, T. Gehrmann, M. Grazzini, S. Kallweit, P. Maierhfer, A. von Manteuffel, S. Pozzorini, D. Rathlev, L. Tancredi, and E. Weihs, Phys. Lett. **B735**, 311 (2014), arXiv:1405.2219 [hep-ph].

[63] V. Khachatryan *et al.* (CMS), JHEP **02**, 135 (2017), arXiv:1610.09218 [hep-ex].

[64] M. Aaboud *et al.* (ATLAS), Phys. Lett. **B786**, 59 (2018), arXiv:1808.08238 [hep-ex].

[65] A. M. Sirunyan *et al.* (CMS), Phys. Rev. Lett. **121**, 121801 (2018), arXiv:1808.08242 [hep-ex].

[66] S. Banerjee, C. Englert, R. S. Gupta, and M. Spannowsky, Phys. Rev. **D98**, 095012 (2018), arXiv:1807.01796 [hep-ph].

[67] S. Banerjee, R. S. Gupta, J. Y. Reiness, and M. Spannowsky, Phys. Rev. **D100**, 115004 (2019), arXiv:1905.02728 [hep-ph].

[68] G. Arcadi, A. Djouadi, and M. Raidal, (2019), arXiv:1903.03616 [hep-ph].

[69] We determined this evaluation time using `OpenLoops` [86] 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 letter, see main text.

[70] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," (2015), software available from tensorflow.org.

[71] M. N. Wright and A. Ziegler, Journal of Statistical Software **77**, 1 (2017).

[72] 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) pp. 278–.

[73] L. Breiman, Machine Learning **45**, 5 (2001).

[74] L. Breiman, *Arcing the edge*, Tech. Rep. (Technical Report 486, Statistics Department, University of California at , 1997).

[75] J. H. Friedman, Ann. Statist. **29**, 1189 (2001).

[76] T. Chen and C. Guestrin, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 16 (2016), 10.1145/2939672.2939785.

[77] `XGBoost` rose to prominence by winning the Kaggle Higgs Callenge. Since then it has been consistently on the top of the ladder in a large number of the Kaggle competitions [87, 88] outperforming other ML architectures, like DNN's or SVM's.

[78] J. M. Campbell and R. K. Ellis, *Proceedings, 10th DESY Workshop on Elementary Particle Theory: Loops and Legs in Quantum Field Theory: Woerlitz, Germany, April 25-30, 2010*, Nucl. Phys. Proc. Suppl. **205-206**, 10 (2010), arXiv:1007.3492 [hep-ph].

[79] We also tested `MadLoops` [2] and found it to be a factor of a few slower than `OpenLoops` [86] for the $gg \to ZZ$ process. For this reason, we used the faster evaluation time of `OpenLoops` as a benchmark.

[80] M. Matsumoto and T. Nishimura, ACM Trans. Model. Comput. Simul. **8**, 3 (1998).

[81] E. Bothmann *et al.*, SciPost Phys. **7**, 034 (2019), arXiv:1905.09127 [hep-ph].

[82] W. Kilian, S. Brass, T. Ohl, J. Reuter, V. Rothe, P. Stienemeier, and M. Utsch, in *International Workshop on Future Linear Collider (LCWS2017) Strasbourg, France, October 23-27, 2017* (2018) arXiv:1801.08034 [hep-ph].

[83] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, in *Advances in Neural Information Processing Systems 30*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, Inc., 2017) pp. 3146–3154.

[84] S. Otten, K. Rolbiecki, S. Caron, J.-S. Kim, R. Ruiz De Austri, and J. Tattersall, (2018), arXiv:1810.08312 [hep-ph].

[85] H. A. Chawdhry, M. L. Czakon, A. Mitov, and R. Poncelet, (2019), arXiv:1911.00479 [hep-ph].

[86] F. Buccioni, J.-N. Lang, J. M. Lindert, P. Maierhfer, S. Pozzorini, H. Zhang, and M. F. Zoller, Eur. Phys. J. **C79**, 866 (2019), arXiv:1907.13071 [hep-ph].

[87] `https://www.kaggle.com/sudalairajkumar/winning-solutions-of-kaggle-competitions`, .

[88] `https://www.kaggle.com/rajiao/winning-solutions-of-kaggle-competitions`, .