Replication of detector simulations using supervised machine learning

D. Benjamin,[1] S. Chekanov,[1] W. Hopkins,[1] Y. Li,[2] and J. R. Love[1]

[1] *High Energy Physics Division, Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, IL 60439, USA*

[2] *Computational Science Division, Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, IL 60439, USA*

(Dated: October 29, 2019)

Abstract

Accurately and computationally rapidly modeling stochastic detector response for complex LHC experiments involving many particles from multiple interaction points requires the development of novel techniques. A study aimed at finding a fast transformation from truth-level physics objects to reconstructed detector-level physics objects is presented. This study used Delphes fast simulation based on an LHC-like detector geometry for inputs for a multi-categorizing machine learning (ML) algorithms. This ML transfer algorithms, with sufficient optimization, could have a wide range of applications: improving phenomenological studies by using a better detector representation, speeding up fast simulations based on parametric description of LHC detector responses, and allowing for more efficient production of Geant4 simulation by only simulating events within an interesting part of phase space.

I. INTRODUCTION

A cornerstone of particle collision experiments is Monte Carlo (MC) simulations of physics processes followed by simulations of detector responses. With increased complexity of such experiments, such as those at the Large Hadron Collider (LHC), the detector simulations become increasing complex and time consuming. For example, the time required to simulate Geant4 [1] hits and to reconstruct from such hits physics objects (electrons, muons, taus, jets) needs a factor 100-1000 more CPU time than the creation of typical Monte Carlo events that represent physics processes according to theoretical models ("truth level" MC event generation). A possible method to speed up simulations of detector responses is to apply neural networks (NN) trained using the Geant4-based simulations, and use such supervised NN for transforming truth-level MC objects (jets and other identified particles) to objects modified by detectors ("detector-level").

A typical simulation of detector responses stochastically modifies positions and energies of particles and jets created by MC generators at the truth-level. Another important component of such simulations is to introduce additional particles due to misreconstructed energy deposits in active detector volumes (examples include misreconstructed electrons or photons which are, in fact, hadronic jets). The latter effects represent a significant complication for the so-called "fast" or "parameterized" detector simulations, such as Delphes [2]. Nevertheless, parameterized detector simulations have been proven to be a vital tools for physics performance and phenomological studies.

The main advantage of detector parameterization based on machine learning is that a neural networks can automatically learn the features introduced by detailed full simulations, therefore, handcrafting parameters to represent resolutions and inefficiencies, as it was done in Delphes and for upgrade studies, is not required. A neural network trained using realistic detector simulation should memorize the transformation from truth-level to the detector-level quantities without manual binning of quantities by analyzers. Another advantage is that the NN approach can introduce a complex interdependence of variables which is currently difficult to implements in parameterized simulations.

As a first step towards parameterized detector simulations using machine learning techniques, it is instructive to investigate how a transformation from the truth-level MC to detector-level objects can be performed, leaving aside the question of introducing objects

that are created by misreconstructions.

II. TRADITIONAL PARAMETERIZED FAST SIMULATIONS

In abstract terms, a typical variable f_i that characterizes a particle/jet, such as transverse momentum (p_{T}), pseudorapidity (η), can be viewed as a multivariate transform F of the original variable ξ_1^T at truth-level:

$$\xi_1 = F(\xi_1^T, \xi_2^T, \xi_3^T, ...\xi_N^T).$$

Generally, such a transform depends on several other variables ξ_2^T .. ξ_N^T characterizing this (or other) objects at the truth level. For example, the extent at which jet transverse momentum, p_{T} is modified by a detector depends on the original truth-level transverse momentum ($\xi_1^T = p_T^T$), pseudorapidity η, flavor of jets and other effects that can be inferred from the truth level. Similarly if particular detector modules in the azimuthal angle (ϕ) are not active, this would introduce an additional dependence of this transform on ϕ.

Typical parameterized simulations ignore the full range of correlations between the variables. In most cases, the above transform is reduced to a single variable, or two (as in the case of Delphes simulations where energy resolution of clusters depend on the original energies of particles and their positions in η). In order to take into account correlations between multiple parameters characterizing transformations to the detector level, the following steps have to be undertaken:

- create a grid in the hypercube with the dimension N_b^N, where N_b is the number of histogram bins for the distributions $f_1 - f_i^N$ representing "resolution" smearing. This can be done numerically, using frequencies, or using analytically using "resolution functions".

- calculate "efficiencies" that model losses of particles/jets for each variable.

It should be pointed out that the calculation speed for parameterized simulations of one variable that depends on N other variables at the truth level depends as N_b^N since each object at the truth level should be placed inside the grid defined by N_b bins. Therefore, complex parameterisations of resolutions and efficiency's for $N > 2$ becomes CPU intensive.

III. MACHINE LEARNING APPROACH FOR FAST SIMULATION

Unlike the traditional approach for fast simulation using parameterized density functions for resolution variables and probability values for efficiency, a neural network approach offers an opportunity to formulate this problem in terms of NN nodes and their connections that scale as $N_b^N \cdot N$, which can speed up the fast simulations and, at the same time, can be used for learning more complex full simulations in an automated way.

In the case of objects, such as jets, a typical truth-level input are jet transverse momentum, η, ϕ and jet mass m, while the output is an array of output nodes that represent the binned probability density function (PDF) of the resolution for a single variable (such as jet p_{T}). Additional input variables can be jet flavor at the truth level, jet radius etc., i.e. any variable that can influence the output of such neural network. Figure 1 shows a schematic representation of the NN architecture for modelling detector response for a single output variable. The inputs of the NN are variables that can affect the object resolutions while the hidden layers are used to capture correlations between these variables and the the output which consist of a binned PDF. The aim is to have the NN learn the shape of this PDF, for example for the p_{T}, depending on other variables such as the η of the object.

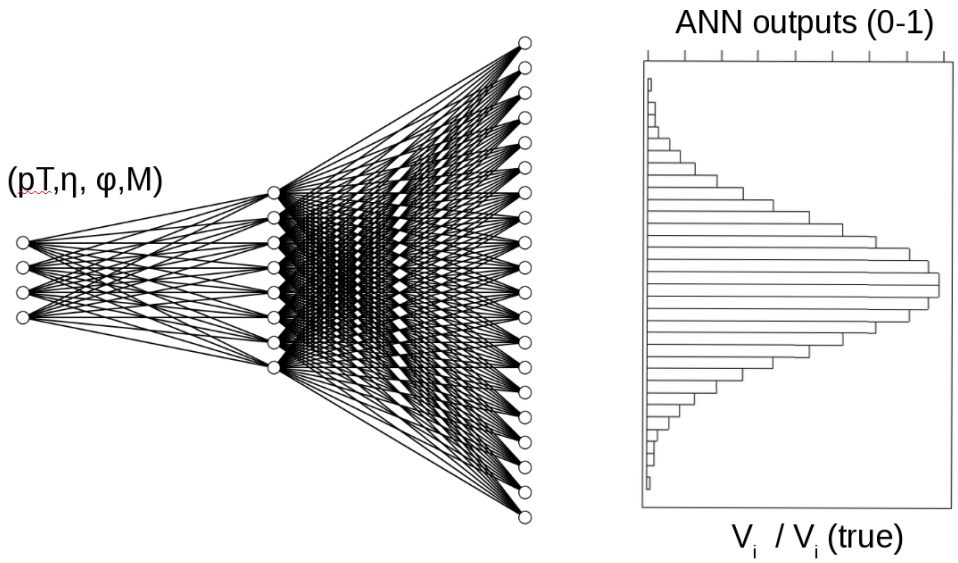


FIG. 1. A schematic representation of the NN architecture for modelling the detector response to truth-level input variables. The output of this NN is PDF for a single variable, e.g. $(p_{\mathrm{T}}^{\mathrm{reco}} - p_{\mathrm{T}}^{\mathrm{truth}})/p_{\mathrm{T}}^{\mathrm{truth}}$, which modified by the detector.

IV. MONTE CARLO SIMULATED EVENT SAMPLES

Monte Carlo events used for this analysis were generated using the Madgraph generator [3]. The simulated processes were a combination of equal parts $t\bar{t}$+jets and γ+jets, which give a high rate of jets in different environments. Hadronic jets were reconstructed with the FastJet package [4] using the anti-k_T algorithm [5] with a distance parameter of 0.4. The detector simulation was performed with the Delphes package [2] with an ATLAS-like detector geometry. The event samples used in this paper, before and after the fast simulation, are available from the HepSim database [6]. In this paper only the transformation from truth-level jets to detector-level jets and only for p_{T} was performed, however the methodology should be object and parameter agnostic. Only truth jets which have been matched to a reconstructed Delphes jet are used. For the matching criteria the reconstructed jet that has the smallest $\Delta R = \sqrt{\Delta\phi^2 + \Delta\eta^2}$, where $\Delta\phi = \phi^{\mathrm{truth}} - \phi^{\mathrm{reco}}$ and $\Delta\eta = \eta^{\mathrm{truth}} - \eta^{\mathrm{reco}}$, with respect to the truth jet is chosen. If this minimum ΔR is greater than 0.2, the truth-level jet is discarded. No other requirements are made on truth on reconstructed Delphes jets other than the $p_{\mathrm{T}} > 15$ GeV requirement made by Delphes. Only matched jets are used for this study since the aim of the study is to test whether an NN can changes in detector resolution as a function of kinematic properties of the jet (e.g. p_{T}, η, ϕ, m). The final number of training jets used is two million while 500,000 jets were used as a testing sample.

The distributions of quantities used as the input for the NN, p_{T}, η ϕ, m, are shown in Figure 2.

To facilitate gradient descent in all direction of the input variables, the input variables are all scaled to be in the range [0,1]. This avoids the p_{T} and the mass from having a disproportional affect on the training of the NNs. The output variable, $(p_{\mathrm{T}}^{\mathrm{reco}} - p_{\mathrm{T}}^{\mathrm{truth}})/p_{\mathrm{T}}^{\mathrm{truth}}$, is also scaled to have values between 0 and 1. Only objects that are within the 1^{st} and 99^{th} percentile of the $(p_{\mathrm{T}}^{\mathrm{reco}} - p_{\mathrm{T}}^{\mathrm{truth}})/p_{\mathrm{T}}^{\mathrm{truth}}$ distribution are considered in this study since objects outside this range are typically not used in physics analyses.

V. NEURAL NETWORK STRUCTURES

An NN is trained with four input parameters, the scaled p_{T}, η, ϕ, and m, and consist of five layers with 100 nodes each and with each node having a rectifier linear unit (ReLu) activation

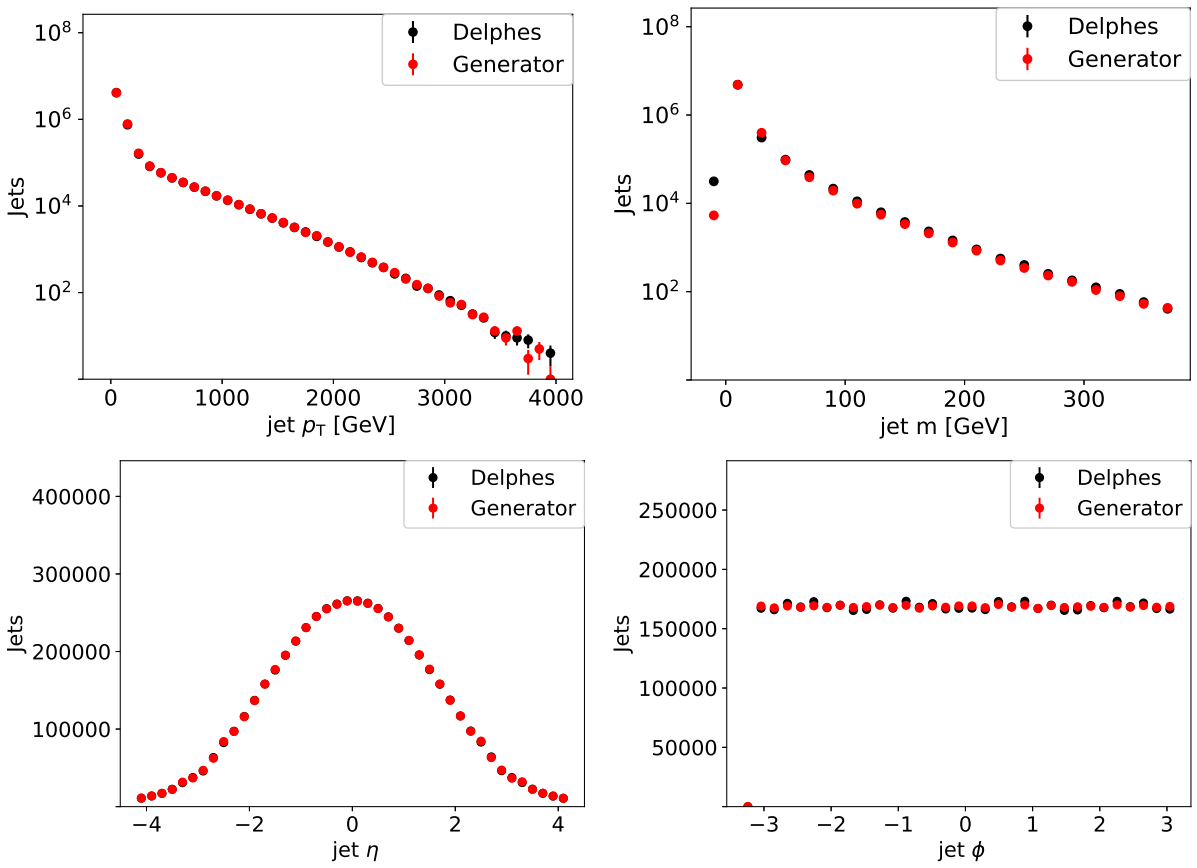


FIG. 2. Input variable shapes for truth-level (red) and detector-level quantities (black).

function. The output layer has 400 nodes with a softmax activation function. Finally, the NN is trained over 1000 epochs with batch size 1000 using the Adam [7] optimizer with a learning rate of 1×10^{-4}. The NN is implemented using Keras [8] with a TensorFlow [9] backend.

VI. RESULTS

After the NN has been trained to learn the PDF of $(p_{\mathrm{T}}^{\mathrm{reco}} - p_{\mathrm{T}}^{\mathrm{truth}})/p_{\mathrm{T}}^{\mathrm{truth}}$, the resulting learned PDF is compared to the Delphes PDF using the testing sample in Fig. **??**. Good agreement is observed between the Delphes and NN PDFs, showing that the NN has learned the bulk distribution.

The NN predicts a PDF for each jet based on its input parameters (i.e. p_{T}, ϕ, η, and m). The PDFs for a set of randomly selected jets are shown in Fig. 4. These PDFs are then randomly sampled to produce a NN jet that mimic the detector-level jet. A comparison

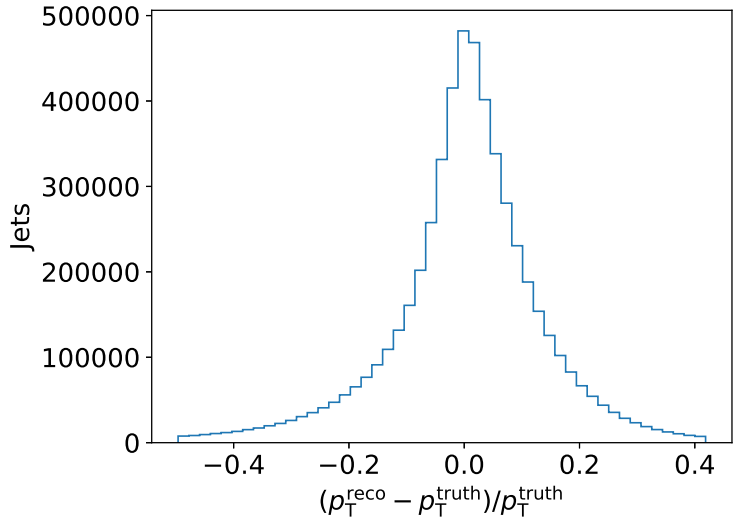
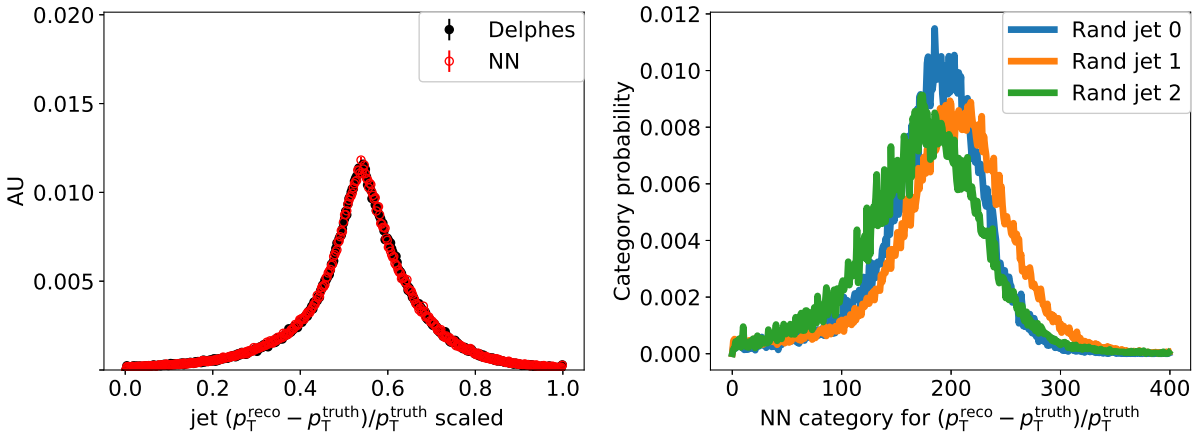


FIG. 3. Differences between truth-level and detector-level p_T.



(a) (b)

FIG. 4. NN-generated jet $(p_T^{reco} - p_T^{truth})/p_T^{truth}$ compared to detector-level jet $(p_T^{reco} - p_T^{truth})/p_T^{truth}$ (a). NN-generated jet PDFs for three randomly selected truth-level jets (b).

of the NN-generated and Delphes jet p_T distributions for the testing sample and for NNs trained with batch sizes 10000, 1000, 10, and 5 are shown in Fig. **??** while a the distributions at lower p_T are shown in Fig. **??**. The NN reproduces the jet p_T distribution of Delphes within 5% for reconstructed jets with $p_T > 20$ GeV for all batch size less than 100.

To test whether the NN learned correlations between input parameters and the p_T resolution, defined as $\frac{(p_T^{reco} - p_T^{truth})/p_T^{truth}}{p_T^{truth}}$, the jets were divided into central ($|\eta| < 3.2$) and forward ($|\eta| > 3.2$) jets, then the p_T resolution is compared between the two regions for both the

Delphes jets as well as the NN-generated jets. These two regions in the detector simulation have different calorimeter resolutions which results in different jet p_{T} resolutions and thus the p_{T} resolution is correlated with $|\eta|$. The resulting resolutions for both regions and for the training samples and jets are shown in Fig. 5. To quantify how the training batch size affects the NNs ability to reproduce the resolution of the forward region, which has significantly lower statistics and is a small subsample of the total training sample, the two-sample Kolmogorov–Smirnov statistic was calculated for NN and Delphes p_{T} resolution distribution for both the central and forward regions. The training sample was chosen for these tests because it had significantly more jets and due the low number of jets that have $|\eta| > 3.2$, as can be seen in Fig. 2. The NN clearly performs better for the rare subsample (the forward region) when trained on smaller batch sizes but the NN performs worse for batch size of five.

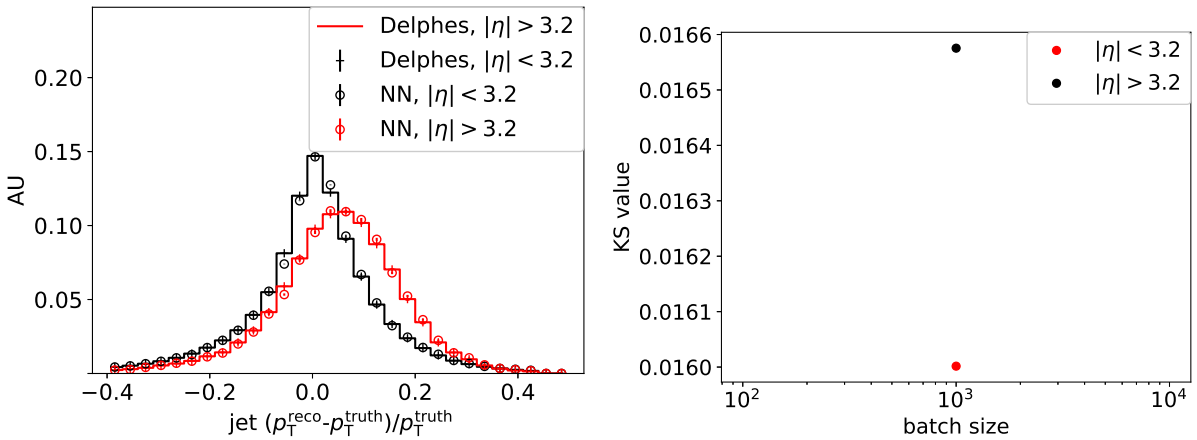
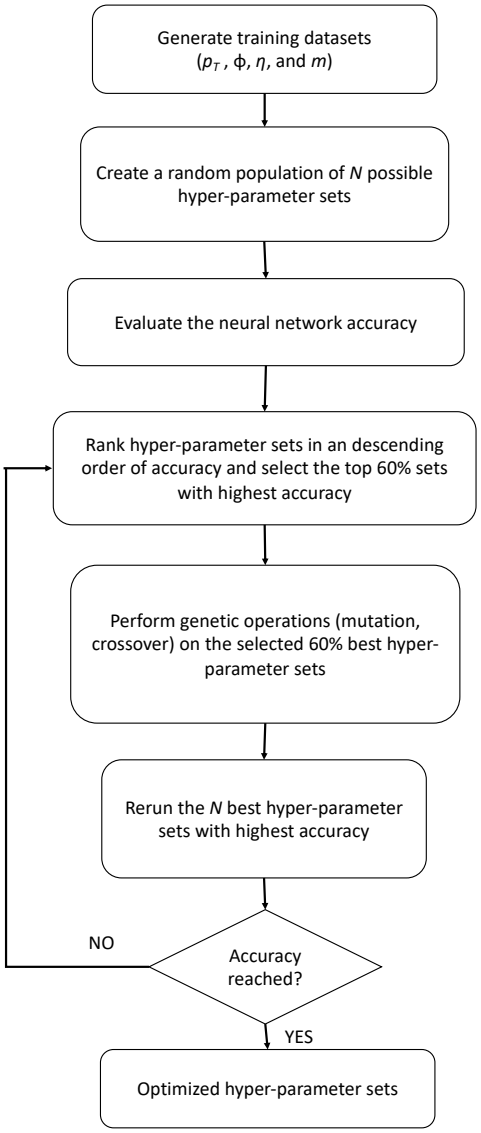


FIG. 5. Resolutions for the jet p_{T} (a) for the training sample for both the central and forward region using a batch size of 10 during the training of the model. The KS statistic for NN and Delphes resolutions as a function of batch size (b) is also shown. The red dots show the KS statistic for the central region while the black dots show the KS statistic for the forward region.

VII. HYPERPARAMETER OPTIMIZATION

In order to acquire the optimal performance with an NN with a minimal amount of parameters, a genetic algorithm (GA) was used to optimize the NN hyperparameters. Hyper parametrization, that is, the determination of the optimal parameters associated with a complex NN (number of layers, number of neurons on each layer, choice of the learning

rate, mini-batch size), is a challenging optimization problem in a high dimensionality space, especially for nonlinear NNs. Ultimately, the range of hyperparameter values that must be explored grows dramatically as the complexity of the deep NN increases. To address this challenge, we have used a (GA), which is an evolutionary algorithm that mimics the process of natural selection. Previously successful applications of the GA in similar contexts include the determination of the parameters of a complex force field [10, 11]. A flowchart of the ML/GA optimization protocol is depicted in Fig. 6. The optimization starts with a set of parent parameters, which is defined as the population. For hyperparameter optimization of the NN, the parents sets are number of layers, number of neurons on each layer, choice of the learning rate or mini-batch size. Some of the individuals in this population present a better fit, which in the context of NNs means a higher value for the accuracy of the network. Fit parents survive and are allowed to mate, which is accomplished by crossing patterns with other fit individuals. During crossover, random mutations in the genes are also allowed, to a certain degree, to avoid a stagnant gene pool and a better sampling of the parameters space. The offspring individuals form the next generation of parents and this process continues until some predefined criteria are met. Sets of parameters are then ranked in ascending order. After the ranking, a nonlinear roulette wheel selection[**?**] was performed to select the best 60% members, that is, the ones with lowest values of accuracy, which were then subjected to genetic operations: mutation and crossover with a 3% crossover rate. These mutations introduce sufficient diversity into the population, and the nonlinear selection scheme helps to avoid premature convergence of the ML/GA run. After the genetic operations, both the parent and offspring sets of parameters are ranked by their value of accuracy. The best hyperparameter sets are then chosen to constitute the next generation. Such an optimization routine ensures that only satisfactory hyperparameter sets survive after each generation; upon repeating this workflow for sufficient generations and sampling viable regions in the parameter space, we performed three separate ML/GA runs starting with different random populations. From each of the converged ML/GA run, we chose the final hyperparameter set corresponding the highest value of accuracy.



(a)

FIG. 6. Flow chart of the hyperparameter optimization used to optimize the resolution NN.

VIII. CONCLUSION

We have shown that a truth-level quantity can be transformed to a reconstruction-level quantity using a multi-categorizing NN. The NN learned the changes in resolutions of different regions of the detector based on the NN inputs during training. For the NN to learn these correlations between variables for small subsets of objects, such as for forward jets, small batch size training was necessary. The NN learned the truth-to-reconstruction transformation without requiring manual binning to capture the differences in resolutions of particular

subsamples. This method should be easily extendable to additional reconstructed quantities and could be used to model the ATLAS and CMS detector. The method described in this paper thus allows for automated detector parameterization which can facilitate phenomenological, efficient Geant4 simulation, and upgrade studies.

ACKNOWLEDGMENTS

The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory ("Argonne"). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan. http://energy.gov/downloads/doe-public-access-plan. Argonne National Laboratory's work was funded by the U.S. Department of Energy, Office of High Energy Physics under contract DE-AC02-06CH11357.

[1] S. Agostinelli et al. GEANT4: A Simulation toolkit. *Nucl. Instrum. Meth. A*, 506:250, 2003.

[2] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi. DELPHES 3, A modular framework for fast simulation of a generic collider experiment. *JHEP*, 02:057, 2014.

[3] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, et al. The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations. *JHEP*, 07:079, 2014.

[4] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. FastJet User Manual. *Eur. Phys. J. C*, 72:1896, 2012.

[5] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. The anti-k_t jet clustering algorithm. *JHEP*, 04:063, 2008.

[6] S.V. Chekanov. HepSim: a repository with predictions for high-energy physics experiments. *Advances in High Energy Physics*, 2015:136093, 2015. Available as http://atlaswww.hep.anl.gov/hepsim/.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, page arXiv:1412.6980, Dec 2014.

[8] François Chollet et al. Keras. https://keras.io, 2015.

[9] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[10] Ying Li, Hui Li, Frank C. Pickard, Badri Narayanan, Fatih G. Sen, Maria K. Y. Chan, Subramanian K. R. S. Sankaranarayanan, Bernard R. Brooks, and Benoît Roux. Machine learning force field parameters from ab initio data. *Journal of Chemical Theory and Computation*, 13(9):4492–4503, 2017. PMID: 28800233.

[11] Md Mahbubul Islam, Grigory Kolesov, Toon Verstraelen, Efthimios Kaxiras, and Adri C. T. van Duin. ereaxff: A pseudoclassical treatment of explicit electrons within reactive force field simulations. *Journal of Chemical Theory and Computation*, 12(8):3463–3472, 2016. PMID: 27399177.