ANL-HEP-XXXXX

Replication of detector simulations using supervised machine learning

D. Benjamin, S. Chekanov, W. Hopkins, Y. Li, J. R. Love,

HEP Division, Argonne National Laboratory,

9700 S. Cass Avenue, Argonne, IL 60439, USA

(Dated: May 9, 2019)

Abstract

I. ACTION ITEMS FOR EDITORS

- Fix missing references (Doug).
    - ReLU reference?
- Write something about the hyperparameters (Ying).
    - Maybe rerun the hyperparameter scan again?
- Normalize binned NN η comparison histograms to really see shape differences.
- Write conclusion (Jeremy)
    - Focus on truth smearing replacement and how this is automated.

II. 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) requires 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.

III. 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.

IV. 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 neural-network 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. In this example, we show a single hidden layer (in principle, the NN can also be deep with several hidden layers)

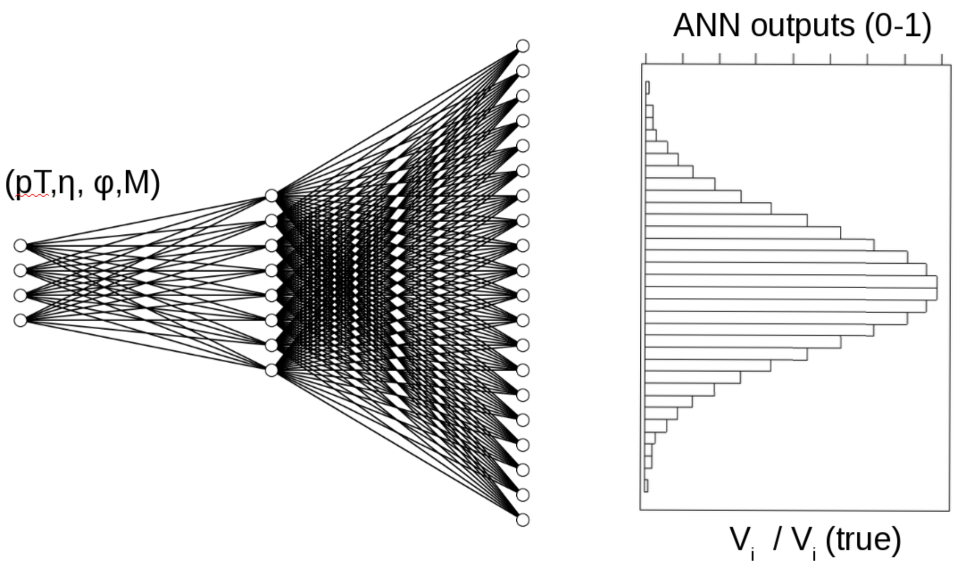

FIG. 1. A schematic representation of the NN architecture for modelling the detector response to truth-level input variables.

V. 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 $t\bar{t}$+jets and γ+jets What was the ratio, which give a high rate of jets and lepton. 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. The poor modeling of the NN below $p_{\mathrm{T}} \sim 50$ GeV is due to the $p_{\mathrm{T}} > 15$ GeV requirement made on the reconstructed and truth jets. The requirement on the truth jets in combination with the width of the $p_{\mathrm{T}}^{\mathrm{truth}} - p_{\mathrm{T}}^{\mathrm{reco}}$, shown in Figure 3, results in insufficient information being available to transform truth-level jet p_{T}s to reconstruction-level jet p_{T}s.

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.

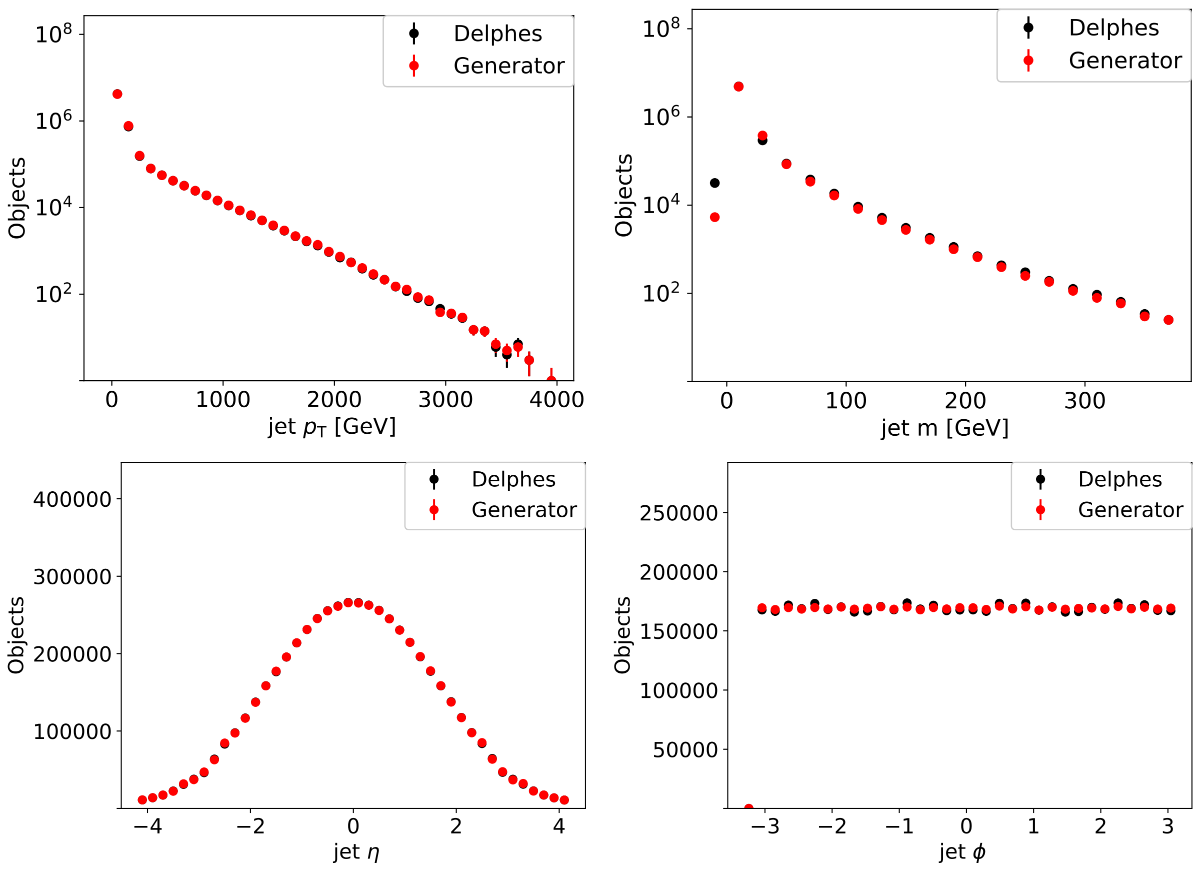

FIG. 2. Input variable shapes for truth-level (red) and detector-level quantities (black).

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, $p_{\mathrm{T}}^{\mathrm{truth}} - p_{\mathrm{T}}^{\mathrm{reco}}$, is also scaled to have values between 0 and 1. Only objects that are within the 1st and 99th percentile of the $p_{\mathrm{T}}^{\mathrm{truth}} - p_{\mathrm{T}}^{\mathrm{reco}}$ distribution are considered in this study since objects outside this range are typically not used in physics analyses.

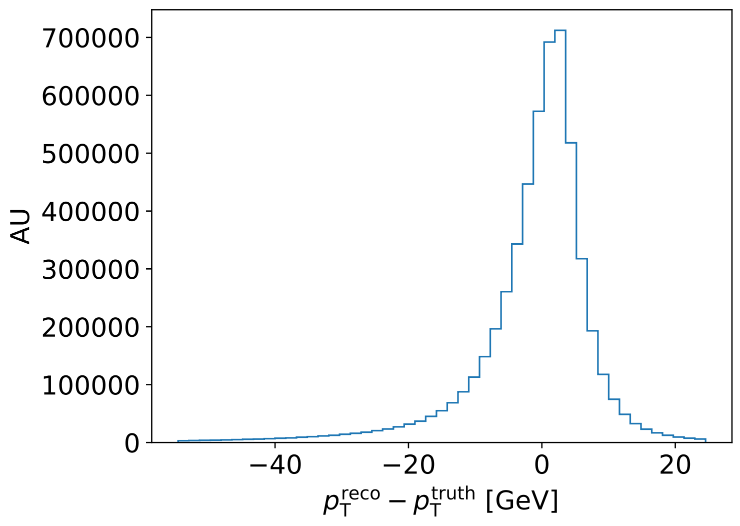

FIG. 3. Differences between truth-level and detector-level p_{T}.

VI. 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 function. The output layer has 400 nodes with a softmax activation function. Finally, the NN is trained over 10 epochs with batch size 10 using the Adam [7] optimizer with a learning rate of . The NN is implemented using Keras [8] with a tensorflow [9] backend.

VII. RESULTS

After the NN has been trained to learn the pdf of $p_{\mathrm{T}}^{\mathrm{truth}} - p_{\mathrm{T}}^{\mathrm{reco}}$, the resulting learned pdf is compared to the Delphes pdf using the testing sample in Fig. 4. 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. ϕ, η, 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 of the NN-generated and Delphes jet $p_{\mathrm{T}}$for the testing sample is shown in Fig. 5. The NN reproduces the jet p_{T} distribution of Delphes within 5% for reconstructed jets with $p_{\mathrm{T}} > 50$ GeV.

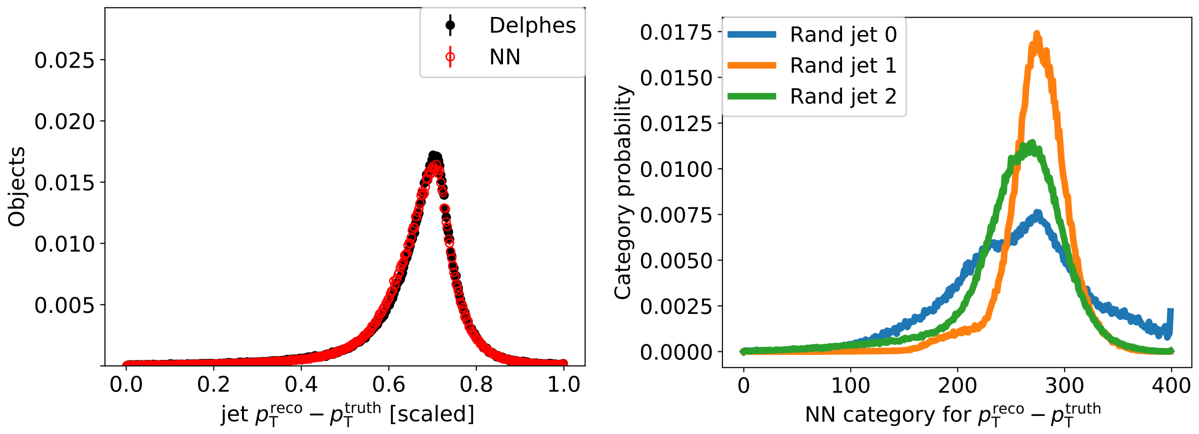

FIG. 4. Left: NN-generated jet $p_T^{truth} - p_T^{reco}$ compared to detector-level jet $p_T^{truth} - p_T^{reco}$. Right: NN-generated jet pdfs for three randomly selected truth-level jets.

To test whether the NN learned correlations between input parameters and the p_T resolution, defined as $\frac{p_T^{truth} - p_T^{reco}}{p_T^{truth}}$, 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 detectors 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. 6. The training sample was chosen because it has significantly more jets and due the low number of jets that have $|\eta| > 3.2$, as can be seen in Fig. 2. The NN reproduces the central jet resolution very well and doesn't perform as well in the forward region.

VIII. HYPERPARAMETER SCAN

IX. CONCLUSION

We have shown that a truth-level quantity can be transformed to a reconstruction-level quantity using a multi-categorizing NN.

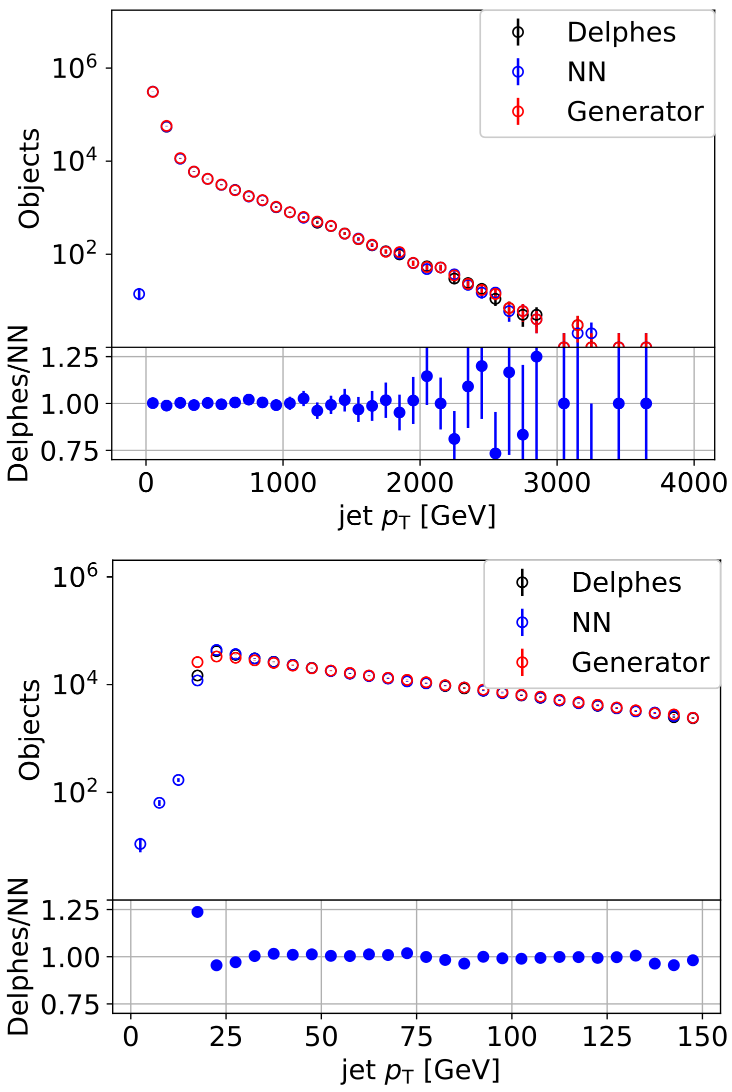

FIG. 5. NN-generated jet p_T, mass, η, ϕ compared to truth-level and detector-level jet features.

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/

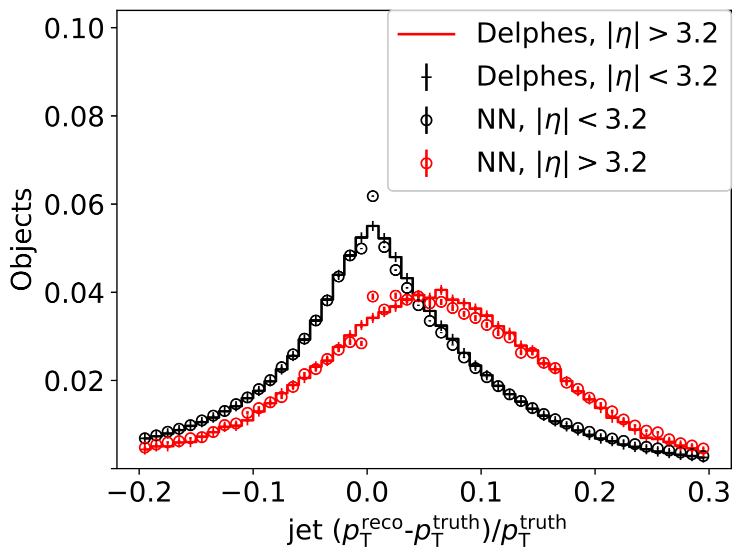

FIG. 6. Resolutions for the jet p_T for the training sample.

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.