EPJ Web of Conferences **295**, 09009 (2024) https://doi.org/10.1051/epjconf/202429509009
CHEP 2023

Deep Learning to improve Experimental Sensitivity and Generative Models for Monte Carlo simulations for searching for New Physics in LHC experiments

José Salt[1,*], *Raúl* Balanzá[2], *Azael* Garcia[4], *Jon Ander* Gomez[2], *Santiago* González de la Hoz[1], *Julio* Lozano[3], *Roberto* Ruiz de Austri[1], and *Miguel* Villaplana[1]

[1]Instituto de Física Corpuscular (IFIC) - Universitat de València (UV) and CSIC, Valencia, Spain
[2]Universitat Politecnica de Valencia (UPV), Valencia, Spain
[3]Universidad de Alcalá de Henares (UAH), Spain
[4]Universitat de València (UV), Spain

Abstract. ML/DL techniques have shown their power in the improvement of several studies and tasks in HEP, especially in physics analysis. Our approach has been to take a number of the ML/DL tools provided by several open-source platforms and apply them to several classification problems, for instance, to the $t\bar{t}$ resonance extraction in the LHC experiments. Gradient-boosting Trees, Random Forest, Artificial Neural Networks (ANN), etc. have been used and optimized by means of adjusting several hyperparameters to control overfitting. On top of this, data simulation with traditional models is computationally very demanding, making the use of generative models an alternative for generating simulated Monte Carlo events with similar quality at a lower computational cost. This could help to produce more simulated data statistics available for better sensitivity and more accurate assessment of systematic errors in potential Physics Beyond Standard Model discoveries. In this work, we study the use of generative models based on Deep Learning as faster Monte Carlo event generators in the LHC context, reducing the time and energy cost of currently used methods. In particular, we focus on different configurations of Variational Autoencoders, taking as a starting point the well-known β-VAE and proposing the α-VAE as a new and simpler VAE architecture that improves the results in some experiments. Considerations will be made about the reliability of these simulated data when they are produced with very high statistics.

1 Introduction

The Standard Model (SM) [1] presents certain deficiencies and lacks an explanation for some fundamental aspects of the behavior of matter. LHC experiments, as ATLAS and CMS, were designed and built to solve those questions taking advantage of the capabilities of the Large Hadron Collider. One of the main objectives is to search for signals of the so-called Physics Beyond the Standard Model, an extension of the current model that should be able to explain the current unknowns. Comparing real data, obtained from experimentation, with simulated

*e-mail: jose.salt@ific.uv.es

[1]It is the theory of Particle Physics that describes the structure of matter and the forces that govern it and that is currently accepted as the one that better explains the observations and experimental results.

data, coming from theoretical models, it is possible to determine if those signals have been found, validating the discovery of new events. This contribution focuses on improving the sensitivity to discover new particles in LHC experiments and, as an example, we address the $t\bar{t}$ system produced in proton-proton collisions. This work presents two different parts.

The first part consists in the application of different ML/DL methods for the separation of signal from the background to search for $t\bar{t}$ resonances, which is our signal, in LHC experiments. This is a classification task and a comparative study among several ML methods is provided. The second part is devoted to the use of Generative Models to produce Simulated Data for searching New Physics in LHC experiments. Deep Generative Models aim to be an important contribution to Monte Carlo traditional data since these models can simulate events with similar quality at a lower cost. The traditional models based on numerical computation are much more computationally expensive, demanding much more energy and time.

2 PART I: Classification of $t\bar{t}$ events

Machine Learning classifiers have been applied to many physics analyses in LHC experiments. In the case we are addressing, the $t\bar{t}$ resonances, the study has been done using several ML methods applied to datasets of a dataset repository [1]. A first collection of interesting results was given in this contribution which was triggered by the work done in [2]: Extra-Trees[2], logistic regression and NN were included. Subsequently, we have completed the studies, especially with Decision Trees methods, since BDT ($BoostedDecisionTrees$) are used quite frequently in ATLAS with very good results. In our approximation, we do not make a particular breakdown for each type of background but we include them in a single generic background source.

Our natural physical case for applying them to search a new particle X of unknown mass which is a resonance which decays to $t\bar{t}$ pair. This new particle, which is the signal, can be one of different candidates according to different theoretical scenarios. Here we take the Z' model but there are several other ones as KK (Kaluza-Klein) graviton, KK gluon, etc. On the other hand, $t\bar{t}$ are also the final state of SM processes and, in this case, we have different background sources [4]. Figures 1 show the structure of the collision event and the Feynman diagrams of the signal and the background respectively.

2.1 Simulated Data

The study has been performed using a repository of datasets [3]. These events have been obtained by using Pythia and MADGRAPH [5]. In the schema given in Figure 1 (left), the semileptonic channel of a $t\bar{t}$ event displayed: one top is decaying into Wb and W going into lepton + neutrino, and the other top (antitop) decaying into Wb and W into two quarks (hadronized jets). Feynman diagrams corresponding to the signal and background are shown in Figure 1. In these diagrams, we display an SM process and a BSM process. The first one is an SM QCD [3] process which will be considered as a background event and the second one is a Beyond the Standard Model with a resonance - named X- formed by $t\bar{t}$ and the overall $t\bar{t}$ system.

In Figure 1 (left), the anatomy of top decays for a given event is shown. Taking into account the particles in the final state, one can consider the kinematic variables: transverse momentum, pseudorapidity and azimuth of the lepton (muon or electron); transverse momentum, pseudorapidity and azimuth of the 4 more energetic leading jets: b tagging of these four

[2]Extra-Trees is an abbreviated expression of Extremely Randomized Trees
[3]QCD is the acronym of Quantum ChromoDynamics

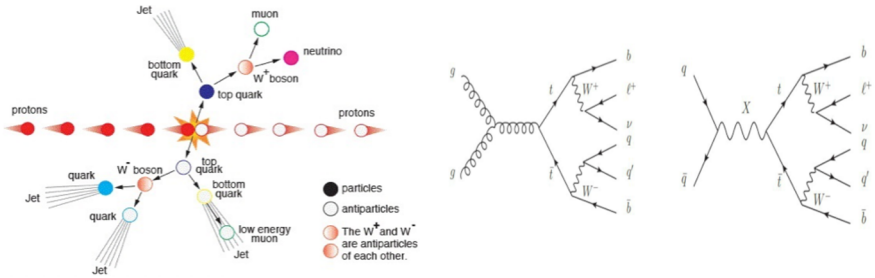

Figure 1. Anatomy of a collision event with $t\bar{t}$ in the final state (a) and Feynman diagrams of $t\bar{t}$ events (b).

jets and the transverse momentum and azimuth of missing transverse energy. On top of these low-level variables, the event data contain five more variables, the so-called high-level variables, corresponding to the invariant mass of the W decay, both the top and the antitop decay separately. In order to speed up the prospect analysis we performed our studies applying the ML techniques on datasets placed in the repository given in [3].

2.2 Discriminant variables

From the point of view of data information, each event is a collection of several variables or features. Each of these variables provide a certain level of discrimination between SM $t\bar{t}$ events and BSM $t\bar{t}$ events, allowing the separation of the signal from the background. Thus there is a dataset for the background events and five datasets for the signal events consisting of events generated with five different values of the resonance mass: 500 GeV, 750 GeV, 1000 GeV, 1250 GeV, and 1500 GeV in order to scan a plausible range of masses.

2.3 Machine Learning Methods

Several Machine Learning Methods have been applied to this set of features. In this case, a more extensive study has been performed with respect to the one reported in [1]: BDT and Random Forest; and simple, complex and parameterized Neural Networks.

2.3.1 Parameterized Neural Networks

The study carried out in [1] was focused in trying to obtain an improvement using Parameterized NN. One of the possible solutions to this problem would be the use of parameterized models, which are based on including mass as one additional feature. In a real case, the idea would be to train the model with masses from the simulations. When making predictions on real data, a mass parameter would be added to them. Several tests could be done with masses suspected to be the mass of particle X, and they do not have to be the masses used for training. These possible masses could be estimated from the inspection of the invariant mass distributions of the $t\bar{t}$ system. The improvement in the application of this method has been quite marginal but there will be a continuation within a further analysis activity.

Table 1. Results of the performance of different ML methods

	Better RF			Better BDT			Better NN		
Mass	Acc	Kappa	F1-S	Acc	Kappa	F1-S	Acc	Kappa	F1-S
500	0.787	0.574	0.756	0.819	0.638	0.785	0.663	0.326	0.679
750	0.851	0.700	0.841	0.852	0.704	0.837	0.850	0.699	0.855
1000	0.895	0.790	0.891	0.889	0.779	0.882	0.914	0.829	0.915
1250	0.925	0,849	0.923	0.922	0.844	0.919	0.941	0.882	0.942
1500	0.946	0.892	0.946	0.944	0.888	0.942	0.958	0.916	0.958

2.3.2 Decision Trees

A deeper study using Decision Trees has been done triggered by the success of the application of BDT in LHC, in particular, in the analyses that led to the discovery of the Higgs. Boosted Decision Trees allow the use of trees with little classifying power, such as trees with little complexity or composed of a single node (stumps) as good classifiers. Adaptive Boosting (AdaBoost) algorithm has been used. This algorithm initially assigns the same weight to all events in the training dataset. After training the first simple tree, depending on whether an event has been correctly classified, a new weight is reassigned to each event generating a new dataset (samples are not replaced only the weights). Then, the next simple tree is trained, newer weights are reassigned to each event, and so on, till a predefined number of trees have been trained. On top of the BDT we have used Random Forest (RF) in order to get a full picture of the two DT approaches [6].

2.4 Comparison between different ML methods

In our approximation, we do not make a particular breakdown for each type of background but we include them in a single generic background source. Each ML method has different sets of hyperparameters which helps to improve its performance. For instance, in the case of BDT we fixed a maximum depth of trees and the number of iterations. For the rest of the methods, we have systematically tested different hyperparameters within their usual ranges of use. After evaluating the performance of different ML methods we extracted the best results for RF, BDT, and NN, shown in Table 1 using the metrics Accuracy, Kappa, and F1-score.

In summary: BDT give better results with a maximum depth of 3 levels and it is optimized with 300 iterations; BDT improves the results of RF at low masses, but RF is slightly better at high masses. Best results with RF were obtained using 5 variables and 500 DT estimators. NNs vs BDT and RF: simple NN gives the worst results when applied to low mass dataset with respect to BDT and RF, but they give better results at higher masses. Paremeterized NN fails to give significant improvement with respect to simple/complex Neural Networks, they interpolate well except for low masses since they do not interpolate better for low resonance masses [6].

2.5 Operative Comparison between RStudio and Python

This work has been developed using two tools: RStudio and Python; the first one is widely used in training courses for graduate students; Python has more potential and is very popular in this kind of ML activities. In general, the same task is 7-8 times faster in Python than in RStudio. Comparing 500-trees RF with 300-iterations BDT using AdaBoost, RF is two times faster than BDT. RStudio is more user-friendly than Python and is used for doing the first steps in ML.

3 PART II: Application of Generative Models

The standard way of simulating proton-proton collisions implies the following phases: (a) generation of events, (b) hadronization/fragmentation into particles, and (c) simulation of particle detection. As billions of events are required to have enough statistics, this process based on numerical computation becomes very expensive and time-consuming. Deep Generative Models: a class of machine learning models that aim to generate new data from learned probability distributions of the provided input, can be an alternative to Monte Carlo traditional methods for generating simulated events with similar quality at a lower cost.

In this work, we focus on Variational Autoencoders (VAEs), using the well-known β-VAE and proposing the α-VAE as a new approach. A VAE [9, 10] is a neural network architecture that compresses the input into a lower dimension representation, known as the *latent space*, and decompresses it trying to reconstruct the same input. β-VAEs incorporate regularization techniques during training in order to prevent overfitting and ensure desirable properties in the *latent space*, enabling the generation of new data from arbitrary numbers. The architecture of a β-VAE is composed of an *encoder* that compresses the input data as a distribution across the *latent space* by outputting two values per dimension in that space: the mean and standard deviation of a normal distribution. Followed by a *decoder* which uses information from the *latent space* to transform it back into the input dimensions.

The training process can be described as follows: first, the input is encoded into a distribution across the latent space. Then, a point is sampled from that distribution in the latent space. After that, the sampled point is decoded, the reconstruction loss is computed, and finally, the error is backpropagated through the network.

The loss function minimized during β-VAE training consists of two components: a *reconstruction term*, L_{rec}, located in the output layer of the decoder, which focuses on improving the performance of the encoding-decoding process, and a *regularisation term* L_{KL}, located in the latent layer, which is proportional to the Kullback-Leibler (KL) divergence and is used to regularize the structure of the latent space by helping the distributions generated by the encoder to approximate a standard normal distribution. Therefore, the loss function can be written as:

$$L_{VAE} = (1 - \beta) * L_{rec} + \beta * L_{KL}$$

We propose using this kind of model to generate accurate Monte Carlo events from both existing Monte Carlo data (the ground truth in our experiments), and random numbers. During our experiments in [11], we found out that this VAE variant did not produce good enough results after trying different values of the β hyperparameter in both strategies. As expected, the strategy of using existing events from the *ground truth* data to generate values for the latent space by means of the encoder yielded better results than using random numbers.

In order to fix some of the problems that appeared while using β-VAEs, we defined the α-VAE architecture as a simpler variant of VAEs but with some differences that could yield better results in some cases. Its main feature is that after encoding the input in the usual way, we added Gaussian noise with zero mean and standard deviation $\sigma = \alpha$ to the latent representation of the input data. This variant does not constraint the values in the latent space to follow a Gaussian distribution with zero mean and unit variance, leaving the probability distribution of the latent space as unknown. This avoids the need for the Kullback-Leibler divergence layer, making this variant very inefficient for generating events from purely random numbers. We used this kind of model only for simulating collisions by taking *ground truth* data and applying a permutation also coming from a random normal distribution with the same mean and standard deviation used in the Gaussian noise during the training process.

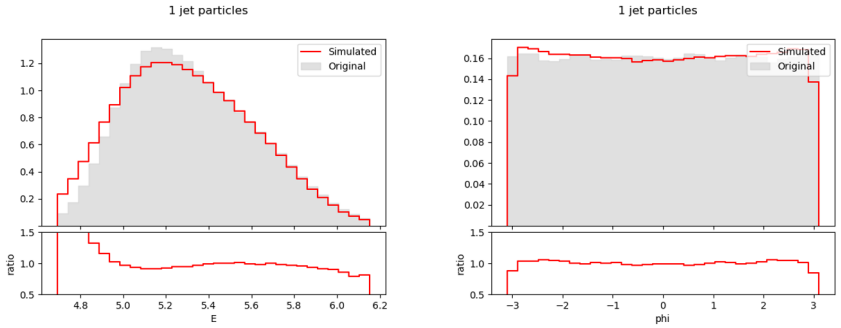

Figure 2. Histogram of first jet parameters E and ϕ using the β-VAE with $\beta = 0.001$.

3.1 Experimentation and Results

Data is provided in one-line-per-event CSV files [10], where each file corresponds to a process, and each line contains 3 event-specifiers, followed by the kinematic features for each object in the event: `event ID; process ID; event weight; MET; METphi; obj1, E1, pt1, eta1, phi1; obj2, E2, pt2, eta2, phi2; ...`

Event ID, process ID and event weight are not used in our experiments. `MET` is the magnitude E_T^{miss} and `MET`ϕ is the azimuth $\phi_{E_T^{miss}}$, both represent the transverse energy and azimuth of those objects that genuinely escape detection. The object identifiers `(obj1, obj2, ...)` are strings identifying detected particles, each one followed by 4 comma-separated values that specify the features of the particle: full energy (E), transverse momentum (p_T), pseudo-rapidity (η), and azimuth (ϕ). The ordering of particles inside each event is b-jets, jets, leptons, and photons. Inside each type, they are sorted in descending order according to their p_T. In our experiments with Standard Model events, we used a process which had a large enough statistics: $t\bar{t}$, the proton collision that produces two top quarks which decay to other particles. Regarding New Physics, we decided to train the two models that obtained the best results among all the executed experiments with a different type of process: *stop_02*.

As the number and type of detected particles in every simulated event are different, the architecture has three inputs and three outputs: (a) `MET` and `MET`ϕ, (b) a 2D mask with 19 one-hot vectors indicating detected particles up to a maximum of 19, (c) a 2D array with shape (19×4) for the four features of each detected particle. The four features are set to zero when the mask indicates no particle. We trained our β-VAE for 100 epochs with simulated data corresponding to the $t\bar{t}$ process, using categorical crossentropy as the loss function for the particle type detection (the mask), and the mean squared error for learning object features. The tested values for β were $\{0, 0.001, 0.01, 0.1, 0.2, 0.5, 0.7, 1\}$. Figure 2 shows the histograms for energy and azimuth of the first jet obtained with original and simulated data using a β-VAE with $\beta = 0.001$. The first jet is one of the most frequent particles in the original data, with much more statistics than other particles. That is why we obtained a proper adjustment of uniform distributions, especially with low values of β, and a great adjustment to the rest of distributions. The adjustment in the case of less frequent particles like the photon is very poor. We achieved an almost perfect simulation regarding the four particle features of more frequent particles, as well as for `MET` and `MET`ϕ, for all tested values of $\beta \leq 0.7$.

As a preliminary conclusion, the weight assigned to the Kullback-Leibler divergence in the loss function of our β-VAE must be significantly lower when compared to the one of the MSE. We used the same model of this experiment to train and generate events using the *stop_02* process with $\beta = 0.001$, obtaining also accurate results (see Figure 3).

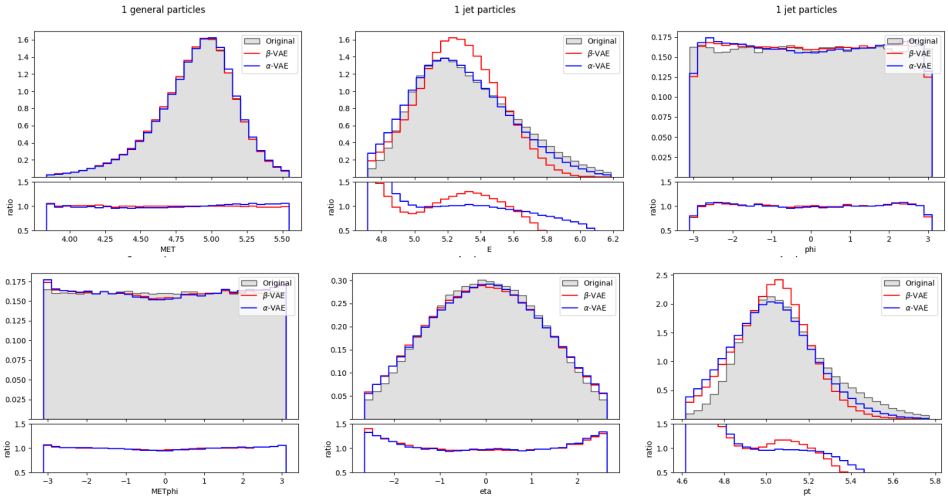

Figure 3. Comparative histograms of MET, MET_{phi} and parameters E, ϕ, η, and p_T of the first jet to compare the best model of β-VAE and α-VAE with *stop_02* events using $\beta = 0.001$ and $\alpha = 0.2$.

α-VAEs used the same three inputs/outputs explained above for β-VAEs. However, α-VAEs do not limit, as previously explained, the probability distribution of the latent space to follow a normal distribution with zero mean and unit variance. α-VAEs add Gaussian noise with $\mu = 0$ and $\sigma = \alpha$ to the latent space during training and inference. This effectively replaces the necessity of including the loss based on the KL divergence in the bottleneck layer. However, as the probability distribution in the latent space is unknown, α-VAEs are only effectively used to simulate events from existing ones.

We also experimented with an extra component in our models to add variability to the latent space: *Bayesian Gaussian Mixture Models* (BGMM) [12], that learned the distribution of latent representations and generated new samples from those distributions to be used as input to the decoder. The goal of using BGMMs was to avoid using events from the *ground truth* to obtain values for the latent space and can be used in both α- and β-VAEs. BGMMs took a high training time and the obtained results were not useful to be used for the goal of this work, so this approach was discarded. The results using the α-VAE with no BGMM were very accurate and showed that this version of VAE has great potential when generating events from already existing ones, obtaining, in some cases better results than the β-VAE. We also decided to use the model of this experiment to train and generate events using the *stop_02* process with $\alpha = 0.2$, obtaining consistent and accurate results with a type of process different than the one used when designing the model (see Figure 3).

4 Conclusions and perspectives

This work have two well differentiated parts: In part I: We have addressed an event classification problem with $t\bar{t}$ in the final state in order to extract the BSM signal from the SM background using several ML methods applied to simulated datasets. These methods are: Decision Trees: BDT, Extra Trees and Random Forest; Neural Networks: Simple, Complex and Parameterized NN. A complete table is provided for comparison between ML methods. Complex NN give better classification performance than Decision Trees, except in the case of low masses of $t\bar{t}$ resonances. From the point of view of operative comparison, Python is faster but RStudio provides a more didactic framework.

Regarding the second part, from the results obtained with α- and β-VAEs, we can conclude that the approaches based on these architectures obtain accurate results for those particles that have large enough statistics, especially the leading particles. However, these network architectures are not able to cope with the particle types that appear at a lower frequency. Our results presented here were obtained using *ground truth* data. When using purely random values for the latent space (i.e., skipping the encoder) the results were poor even for the leading jets and b-jets. This is a first approach to the problem using these methodologies with promising results. We are refining the methods already used and testing alternative methods. Additionally, more research is needed using this kind of VAEs to obtain consistent and robust results when generating events from purely random values in the latent space.

This work was partially supported by MICINN under grant PID2019-104301RB-C21 and by Generalitat Valenciana with the GenT Programme, Spain.

References

[1] S. Campos Martinez, J. Salt, S, Gonzalez and M. Villaplana, Proceeding of CTD/WIT 2019, PROC-2019-011, presented at Connecting the Dots and Workshop on Intelligent Trackers, Valencia, April 2-5, 2019

[2] P. Baldi, K. Cranmer, T. Faucett, P. Sadowski and D. Whiteson, "Parameterized neural networks for high-energy physics", The European Physical Journal C, vol. 75-5, April 2016. DOI 10.1140/epjc/s10052-016-4099-4

[3] D. Whiteson, HEPMASS. UCI Machine Learning Repository, 2016. https://doi.org/10.24432/C5PP5W

[4] Georges Aad et al., "A search for $t\bar{t}$ resonances using lepton-plus-jets events in proton-proton collisions at $\sqrt{s}$ = 8 TeV with the ATLAS detector". In Journal of High Energy Physics, vol. 2015-8, 2015. DOI: https://doi.org/10.1007/JHEP08%282015%29148, arXiv:1505.07018v2 [hep-ex]

[5] J. Alwall et al., "The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations", arXiv:1405.0301 [hep-ph] INSPIRE

[6] A. García Rodríguez, "Estudio de la extracción de la señal resonante de sucesos $t\bar{t}$ mediante técnicas de Machine Learning en el experimento ATLAS". Bachelor's thesis. Universitat de València. June 2022. https://doi.org/10.5281/zenodo.8367903

[7] B. Hashemi, N. Armin, K. Datta, D Olivito, M. Pirini. "LHC analysis-specific datasets with Generative Adversarial Networks", arXiv:1901.05282V1, June 2019.

[8] DarkMachines community, LHCsimulationProject, Feb 2020, doi:10.5281/zenodo.3685861. Available at: https://zenodo.org/record/3685861

[9] D. P. Kingma and M. Welling, "Auto-encoding variational bayes", 2022. arXiv:1312.6114.

[10] T. Aarrestad, et al., "The Dark Machines Anomaly Score Challenge: Benchmark Data and Model Independent Event Classification for the Large Hadron Collider", SciPost Phys., vol. 12, p. 43, 2022.

[11] R. Balanzá García, "Use of deep learning generative models for Monte Carlo event simulation in the context of LHC experiments", bachelor's thesis, Universitat Politècnica de València, 2022. Available at: http://hdl.handle.net/10251/185480

[12] C.M., Bishop, "Pattern recognition and machine learning". Information Science and Statistics series, Vol. 4 No. 4. New York: Springer, 2006.