Generative Adversarial Network (GAN) to simulate t $\bar{t}$ events

Kaviarasan Selvam

July 7, 2017

Abstract

This document describes an attempt to use Generative Adversarial Network(GAN) to simulate t $\bar{t}$ production in pp collision events at the Large Hadron Collider. Various GAN models were used to learn the distribution of features that describe t $\bar{t}$ events. Despite not reaching the accuracy required for LHC data analysis (mainly due to a lack of time), this project highlights the potential of this technology as a tool for dataset augmentation, alternative to the commonly used fast simulation software packages.

Contents

1 Introduction

1.1 Generative Adversarial Network (GAN)

Generative Adversarial Networks (GAN) are machine learning algorithms that consist of two neural networks (a generator and a discriminator) trained in competition. The Generator is typically taught to map from a chosen distribution to a particular data distribution of interest. The Discriminator is simultaneously taught to discriminate between instances from the true data distribution and generated instances produced by the Generator. The adversarial training consists of the minimization of a predefined loss function up to when the Generator learns to maximally confuse the Discriminator. In this project, GANs were used in an attempt to simulate t $\bar{t}$ production in pp collision events at the Large Hadron Collider.

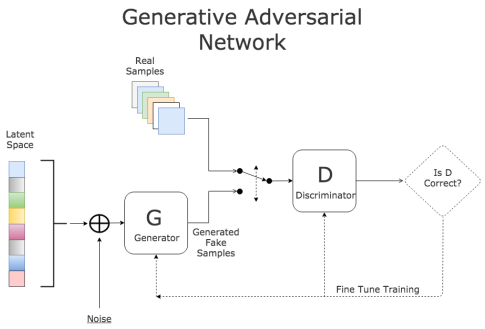


Figure 1: A typical example of a GAN

1.2 Dataset

The dataset is obtained from a sample of t $\bar{t}$ events from 13 TeV collisions, generated with PYTHIA and then processed with the CMS card of the Delphes detector, which returns particles reconstructed by a simplified particle-flow reconstruction algorithm. Despite the large size of LHC collision events (about 1 Mb per event), the amount of information entering a physics study is quite small (a few kB per event), thanks to the centralized event reconstruction and particle selection. An event can then be represented with a few quantities, such as the four-momenta of the leptons and jets in the event.

For each jet, we store the P_T, η, ϕ, Mass and Btag For the lepton, we store the Charge, if the Lepton is an electron, P_T, η, ϕ, Photon Isolation, Charged Hadron Isolation and Neutral Hadron Isolation.

Jets are ranked according to their P_T. To facilitate the training, each jet P_T other than the first one is divided by the first jet P_T. The ϕ is measured fixing ϕ=0 for the lepton, breaking the rotation symmetry on the transverse

plane.

Only the first 10 highest-P_T jets and 2 highest-P_T leptons are retained. When less jets or leptons are found, zero-padding is applied, i.e. the missing features are replaced with zeros.

1.3 Tools for Analysis

Once the GANs were constructed, all training and analysis was done using the CERN techlab machines, which houses 4 NVIDIA GTX 1080 GPUs, or on the Caltech TITANX GPUs. All the algorithms for the different GAN models were written in Python, using Keras as front end for the TensorFlow machine learning framework.

1.4 Analysis Method

Once a training is performed, the goodness of the result is judged looking at the difference between the input and output distribution. This was done by plotting histograms of the event features using ROOT in Python.

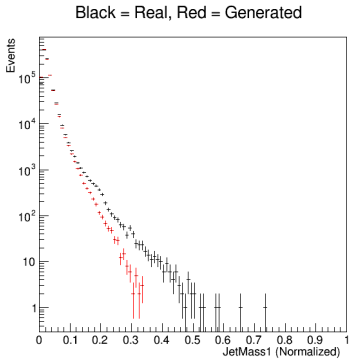


Figure 2: A example of plotting actual and generated features for comparison

In addition, the model is asked to reproduce the distribution of a given number of high-level features, computed from the input quantities.

$$H_T = \sum_j p_T^j + \sum_l p_T^l$$

$$MH_T = -\sum_j \vec{p_T^j} - \sum_l \vec{p_T^l}$$

$$M_T = 2p_T^l |MH_T|(1 - 2\cos(\Delta\phi(M\vec{H}_T, p_T^l)))$$

These physical quantities are also compared to verify if the GAN is learning the physics that define t $\bar{\text{t}}$ events based on a statistical approach.

1.5 Parameters in GANs

GANs are structurally complex neural networks. There are multiple parameters that are used throughout the different layers in both networks of the GAN. For each of the models that were built, this is an example of the parameters that were used and manipulated for testing purposes

1. Activation functions
2. Loss functions
3. Optimizers
4. Number of neurons in each layer of network

2 Training and analysis

2.1 Model 1

The first GAN model was constructed by using simple Dense layers to construct the Generator and Discriminator. This naive model was constructed as a first step to set a benchmark to improve upon when more complicated GAN models were built.

This model did not perform well and this behavior was predicted. This model lacked the complexity needed to learn the the distribution of the features. The network might have been too simple to learn the complex nature of the values in the dataset.

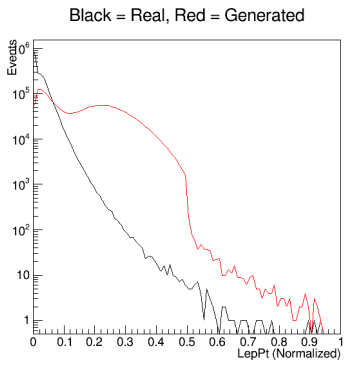


Figure 3: An example of the GAN not being able to learn the distribution of the feature (Lepton P_T)

2.2 Model 2

The next step in this project was to use a fully functioning GAN model that was built for a different dataset, and to try and customize that GAN

to be trained with the t $\bar{t}$ dataset. Online research of GANs mostly resulted in models that were used for image classification. Therefore, a model that was used for image classification purposes on the MNIST dataset was chosen (http://www.kdnuggets.com/2016/07/mnist-generative-adversarial-model-keras.html). This model was primarily constructed using Convolutional Neural Network (CNN), a neural network that is typically used for image detection purposes due to it's ability to learn the correlations between neighboring pixels on an image.

In order to use the chosen example, the features of the t $\bar{t}$ dataset had to be reshaped into a pseudo-image. The array of 66 features was reshaped into an array of 6x12 pixels with zero padding to fill up the extra pixels. This rather arbitrary arrangement of features into a pseudo-image was used as an initial test to obtain a working GAN model which would set another benchmark for future improvements.

This CNN-based GAN performed better than the simple model that was made up of only Dense layers. At this point in the project, there was a working GAN algorithm that was producing less than average results.

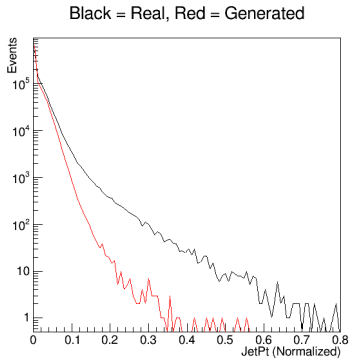


Figure 4: Slight improvement of results (Jet P_T)

2.3 Model 3

In search for a more meaningful representation of the features into a pseudo-image, a model that separated the features into Jet features and Lepton features and trained them separately at certain points of the network was chosen.

The Generator's input diverged into 2 branches, one to generate the Jet pseudo-image, and the other to generate the Lepton pseudo-image. The Discriminator received both these images simultaneously and reduced their dimensionality until they merged and underwent a further reduction of dimensionality. The various Keras Application Program Interface (API) facilitated the construction of this complicated network. The network was constructed with minimal complexity to enable the addition of complexity

to the model if it was identified to have the potential to produce better results.

Unfortunately, this network failed to produce reasonable results. In fact, this network which was trained on a more meaningful pseudo-image representation, performed worse than the previous model (Model 2). Even with minimal complexity, the diverging and converging network architecture was difficult to debug and correct.

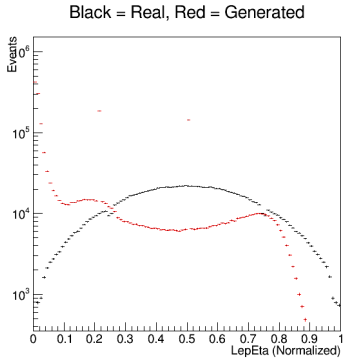


Figure 5: An example of how badly this GAN performed (Lepton η)

2.4 Model 4

Limited knowledge about the inner workings of the previous network, which failed, lead to reverting back to a GAN model that trained on 1 image instead of 2. However, unlike Model 2, this new GAN was trained on a pseudo-image that had a more meaningful representation than an arbitrary reshape of the 66 features.

A meaningful representation was chosen to convert the (x, 66) dataset into a (x, 8, 10) pseudo-image (the extra pixels are zero-padding). A very simple CNN-based GAN was built and complexity was added to it to increase its ability to learn the different distributions of the features. At this stage of the project, the results that were obtained were the best that was achieved since the project was started. Therefore, it was concluded that it would be reasonable to maintain a single pseudo-image representation of the features in each event. However, this model was still a work in progress.

Jet1PT	Jet2PT	Jet3PT	Jet4PT	Jet5PT	Jet6PT	Jet7PT	Jet8PT	Jet9PT	Jet10PT
Jet1Eta	Jet2Eta	Jet3Eta	Jet4Eta	Jet5Eta	Jet6Eta	Jet7Eta	Jet8Eta	Jet9Eta	Jet10Eta
Jet1phi	Jet2phi	Jet3phi	Jet4phi	Jet5phi	Jet6phi	Jet7phi	Jet8phi	Jet9phi	Jet10phi
Jet1Mass	Jet2Mass	Jet3Mass	Jet4Mass	Jet5Mass	Jet6Mass	Jet7Mass	Jet8Mass	Jet9Mass	Jet10Mass
Jet1Btag	Jet2Btag	Jet3Btag	Jet4Btag	Jet5Btag	Jet6Btag	Jet7Btag	Jet8Btag	Jet9Btag	Jet10Btag
Lep1Charge	Lep1IsEle	Lep1PT	Lep1eta	Lep1phi	Lep1IsoPhoton	Lep1IsoChHad	Lep1IsoNeuHad	0	0
Lep2Charge	Lep2IsEle	Lep2PT	Lep2eta	Lep2phi	Lep2IsoPhoton	Lep2IsoChHad	Lep2IsoNeuHad	0	0
0	0	0	0	0	0	0	0	0	0

Table 1: Pseudo-image

One big problem that the GAN faced was its inability to learn the uniform distribution of a particular type of feature: the Jet and Lepton ϕ. It was concluded that there was a possibility that the GAN might be spending too much resources on learning the symmetry of the uniform distribution. To overcome this problem, a new quantity called $\Delta\phi$ was used to replace phi. $\Delta\phi$ was defined as the angle between each Jet to the 1st Lepton, where the 1st Lepton ϕ was set to a value of 0. This modification altered the distribution of ϕ from a uniform distribution to a normal distribution. This change was agreed upon because the GAN was able to learn the normal distribution of other features in the dataset.

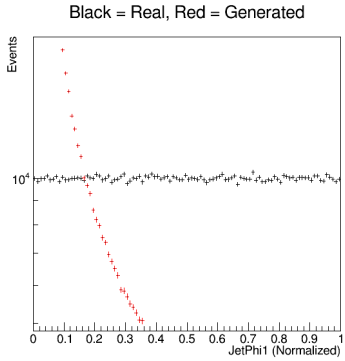


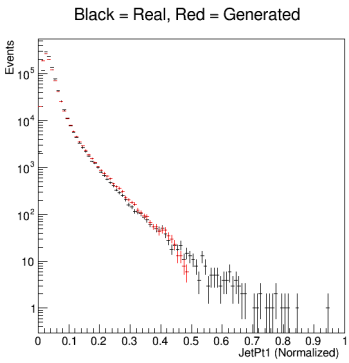
Figure 6: Major problems with the GAN's ability to learn ϕ distribution of 1st Jet

The Jet multiplicity was identified as another potential problem that could have been hindering the progress of the GAN. Each t $\bar{\text{t}}$ event is characterized by a maximum of 10 jets and 2 leptons. Each event has varying numbers of jets and leptons and this situation introduces additional sparsity and irregularities in the dataset. Therefore, in order to work on a more dense and regularized dataset, the dataset was reproduced by saving only events with exactly 4 Jets and 1 Lepton in hdf5 files. Therefore, each event was described by 28 instead of 66 features.

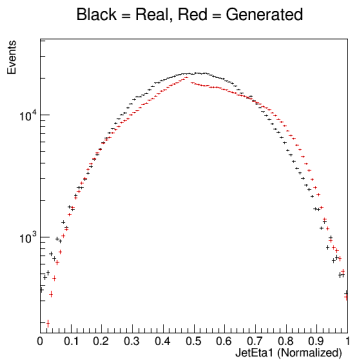
Jet1PT	Jet1eta	Jet1phi	Jet1Mass	Jet1Btag	LepCharge	LepIsEle
Jet2PT	Jet2eta	Jet2phi	Jet2Mass	Jet2Btag	LepPT	Lepeta
Jet3PT	Jet3eta	Jet3phi	Jet3Mass	Jet3Btag	Lepphi	LepIsoPhoton
Jet4PT	Jet4eta	Jet4phi	Jet4Mass	Jet4Btag	LepIsoChHad	LepIsoNeuHad

Table 2: Revised Pseudo-image

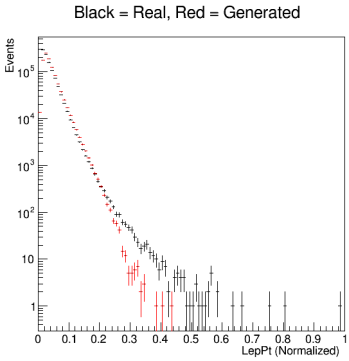
These changes resulted in significant improvements to the performance of the GAN. The GAN was able to learn and reproduce reasonably good distributions for most features.



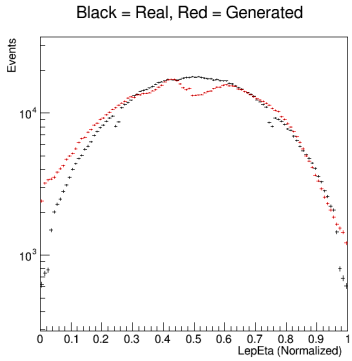
(a) Plot of 1st Jet P_T



(b) Plot of 1st Jet η



(c) Plot of Lepton P_T



(d) Plot of Lepton η

Figure 7: Plots of low-level features

Unfortunately, the problems that surrounded the Jet and Lepton ϕ remained unsolved. Another problem that is evident in the plots is how this GAN model and the previous models were unsuccessful in learning the 'tails' of certain distributions. This problem however is expected behavior because machine learning algorithms are large data-driven approaches and there were insufficient number of events that represented the values that made up the tail of the distribution.

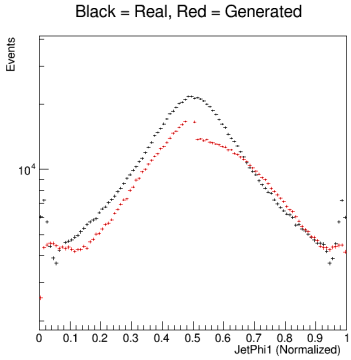
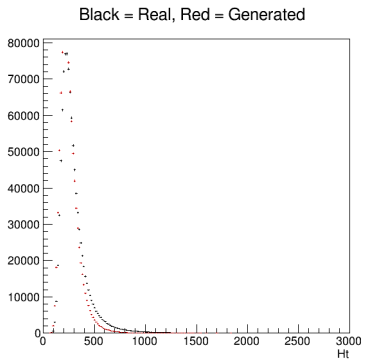


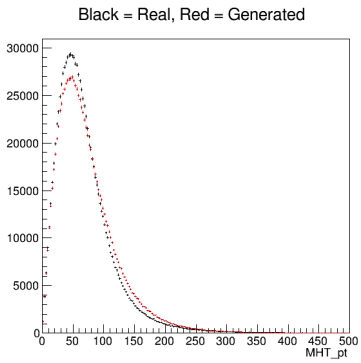
Figure 8: This plot demonstrates the disparity between real and generated $\Delta\phi$ values between the 1st Jet and 1st Lepton

3 Current Results

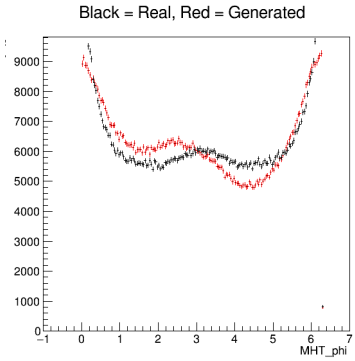
Once the Model 4 GAN was trained, it generated features that were then used to calculate the high-level physical values. This step is done to verify how well the the GAN's performance stacked up against the physics of experimentally-produced features.



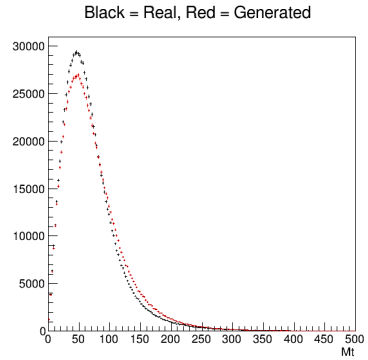
(a) Plot of H_T



(b) Plot of MH_Tp_T



(c) Plot of $MH_T\phi$



(d) Plot of M_T

Figure 9: Plots of high-level values

The problems with the ϕ features discussed in the previous section is evident in the $MH_T\phi$ plot. The other 3 plots demonstrate reasonable GAN performance with room for improvement.

4 Conclusion

A successful attempt at using GANs to simulate t $\bar{\text{t}}$ production in pp collision events at the Large Hadron Collider would have meant that in principle, we would be able to use machine learning algorithms to learn physics using a data-driven approach. Despite showing some promise, this project was unsuccessful in producing results that would validate this approach to learning new physics.

However, there are various other modifications that could be made to this GAN model. Modifications include reshaping the dataset, trying out different hyper parameters that are implemented in a GAN model and using different types of layers that make up a network's architecture. There's a possiblity that using Convolutional Neural Networks (CNNs)-based Generators and Discriminators is not the best approach to simulate t $\bar{\text{t}}$ events.

Unfortunately, due to a time constraint and academic responsibilities, there wasn't enough time to implement these changes. On a more positive note, these results will provide a benchmark for future GAN models that are constructed to simulate $t\,\bar{t}$ events.