University of Cambridge

Department of Physics

Part III Project Report

Machine Learning Based Simulation of Particle Physics Detectors

Candidate Number:
8286W

Supervisor:
Dr. Christopher Lester

15th May 2017

Machine Learning Based Simulation of Particle Physics Detectors

15th May 2017

Abstract

A key part of experimental particle physics is the simulation of a detector's response to an event. Current simulators are polarised between those that are fast and approximate and those that are accurate and slow. Generative Adversarial Networks (GANs) are a class of generative machine learning model which have demonstrated promise in producing artificial photo-realistic images. This study employs GANs for detector response simulation. Building on the work of Oliveira *et al.*, data generated by the hadronisation system PYTHIA and the detector simulator DELPHES are used to produce *jet-images*, converting jet energy deposits in calorimeter cells to pixel intensities in a 2D image. A Locally Aware GAN (LAGAN) is trained to generate counterfeits of two classes of such images: boosted W from W' decay, and QCD background. Generated images are seen to distinguish between the classes accurately, and match physical properties (transverse momentum, n-subjettiness) to a reasonable extent. Significant performance gains compared to current fast simulation is demonstrated.

Contents

1 Introduction

1.1 Motivation

Particle physics experiments involve colliding particles and measuring the properties of the objects produced. However, a given event could not only result in a variety of particle showers, but the response of the detector is also stochastic in nature. It is from this determination of track properties and object momenta that a physicist must infer the original event. Detector simulations are widely used to help with this inference by calculating what a response and output for a given event would be, and as such can be used as predictive tools for models.

Full, accurate simulations (AS) of the progress of particles through detectors, such as the commonly used GEANT 4 [1], can produce extremely accurate predictions of the measurements. However, they are computationally expensive. Approximate, fast simulators (FS), such as DELPHES [2], perform cruder calculations with a significant speed gain (AS ~10-1000s/event and FS ~0.01-1s/event [3]). The accuracy-performance imbalance between these two solutions leaves room for other potential avenues. One such route is the use of Machine Learning (ML) tools.

Generative models in ML attempt to learn a given probability distribution via exposure to samples, and thus accurately generate new elements of that distribution. Their capability has recently been significantly boosted by the burgeoning fields of neural networks and associated deep learning. Once the learning process is complete, such networks are demonstrably fast for appropriate usage. As such, a generative model capable of learning to simulate detector responses may strike a better performance-accuracy balance than current FS. The ultimate goal of further work would be a generative model which approaches an AS in terms of accuracy, at significantly lower computational costs.

1.2 Existing Research

Machine Learning (ML) and associated fields have received significant research attention in computer science, technology and engineering due to increases in computing power and demonstrations of the power and versatility of such techniques. We are now beginning to see efforts towards bringing these new tools to bear in HEP.

Generative Adversarial Networks (GANs) [4] are a subset of generative models which have recently come to the forefront of active research. This method trains two neural networks simultaneously, a generative model G and discriminative model D. As the names suggest, G generates "fake" data which D attempts to distinguish from the "real" training data. As each network is trained to improve at their task, they compete such that ultimately G produces new data which is indistinguishable in theory from the original distribution. GANs have shown strong performance in generating and manipulating photo-realistic images [5–9].

Also under active research are jets from boosted particles [10], and also where many ML tools are used. This project focusses on work done using jet-images produced by mapping jet energy deposits to an image [11, 12], particularly boosted jet identification [13–15]. Recent work has also demonstrated classification of jets using ML techniques from natural language processing [18]. Earlier this year, GANs were trained on jet-images generated from PYTHIA output (hadronisation and parton showers [16]) and showed promise in replicating physical distributions [17]. This represents an initial foray in to using GANs for HEP.

1.3 Report Outline

This report details an application of GAN based learning to jet-images from DELPHES (FS) output, building on the work of Oliveira *et al.* [17] who demonstrated the principle for PYTHIA data. Jet-images were produced and pre-processed from DELPHES output files, then used as training data for a GAN architecture. The trained network was then used to generate "fake" images. The quality of this output was assessed by comparison to the training set.

The following section outlines the theoretical background of GANs and the boosted jet process considered. Section 3 describes the methods employed, and Section 4 presents and discusses the outcomes of the training process. Final conclusions are presented in Section 5.

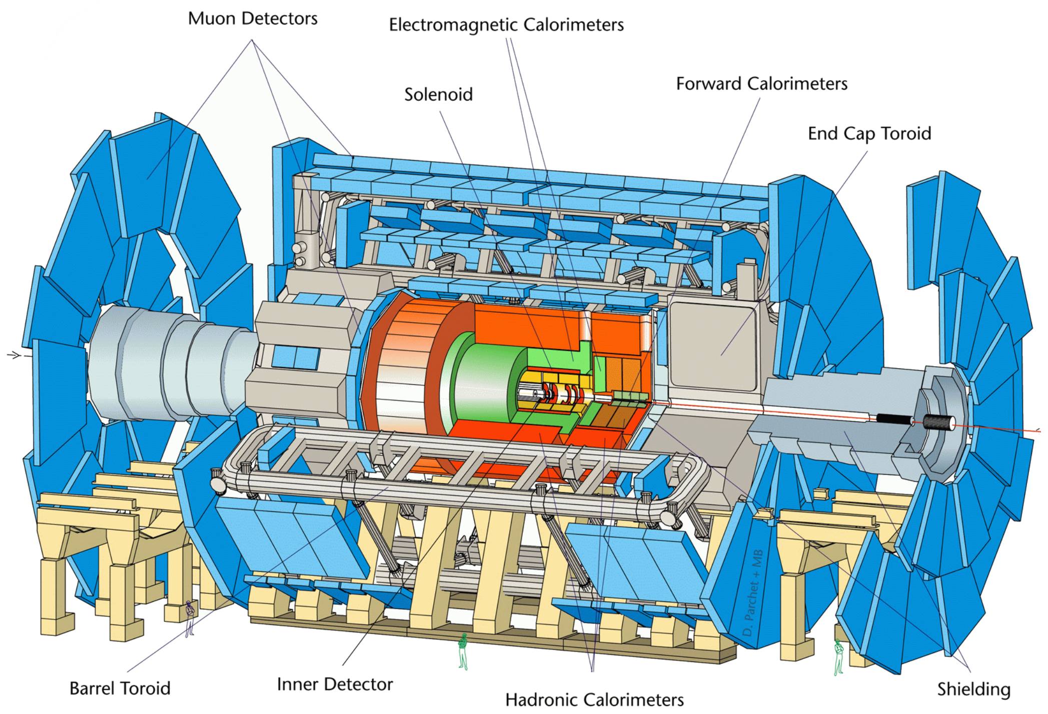


Figure 1: Cutaway diagram showing components of ATLAS detector. Source: [19].

2 Background & Theory

2.1 Detectors & Simulations

Modern particle detectors are a complex set of measurement systems working in unison. Taking as an example the ATLAS experiment, we can see in Fig. 1 a cutaway of the detector set up. Simplistically, the energy and momentum of particles travelling out from the collision are measured by the calorimeters. The solenoids create a controlled magnetic field, the direction of travel in the field and the curvature of the particles give measurements of charge and mass respectively. A detailed description of the detector can be found in Ref. [20].

The response of a detector to a particular particle event is therefore non-trivial to predict, particularly due to the high degree of stochastic variation between any two measurements. A significant difficulty faced by LHC experiments is the jets of hadronisation produced by quarks or gluons, as only the final branches of the jets are measured by the calorimeters. Without an understanding of this response, however, a physicist cannot infer the event that took place, or indeed make predictions about measurements that will be made. It is in this arena that simulations of detector behaviour are crucial.

Fig. 2 outlines a typical simulation sequence. The first step in the process involves a matrix element calculator, such as the commonly used MADGRAPH5 [21]. This piece of software loads a given model (particles and interactions), and calculates all the tree-level diagrams which take the given initial particles to the final particles. Using this information, Monte-Carlo methods are used to calculate the matrix element using a given number of events. Next, an event generator, such as PYTHIA, calculates the subsequent interactions, decays parton showers and performs hadronisation [22].

The results of this are the "truth events", which for our purposes are the inputs, $\mathbf{x}$, to any simulator under consideration. The simulator performs two key tasks, that of calculating the response of the detector as the particles travel through it, and reconstructing the underlying events from such information. The reconstruction step is performed in much the same way as actual experiments, so is a useful representation of the simulated results.

An AS then propagates the input through a detailed model of the detectors to calculate the response. DELPHES, the FS under consideration, takes a modular approach by separating the various components of the detector and performing approximate calculation and addition of stochasticity at each stage. Details of this can be found in [2], a summary is shown in Fig. 3.

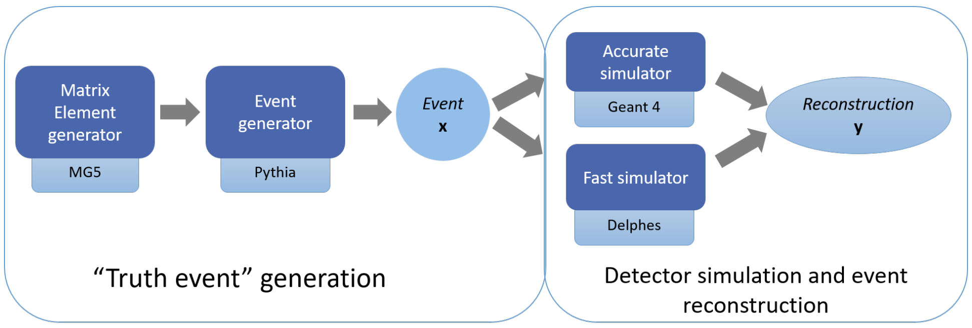


Figure 2: Summary of simulation process, from matrix element generation to final reconstruction from detector response.

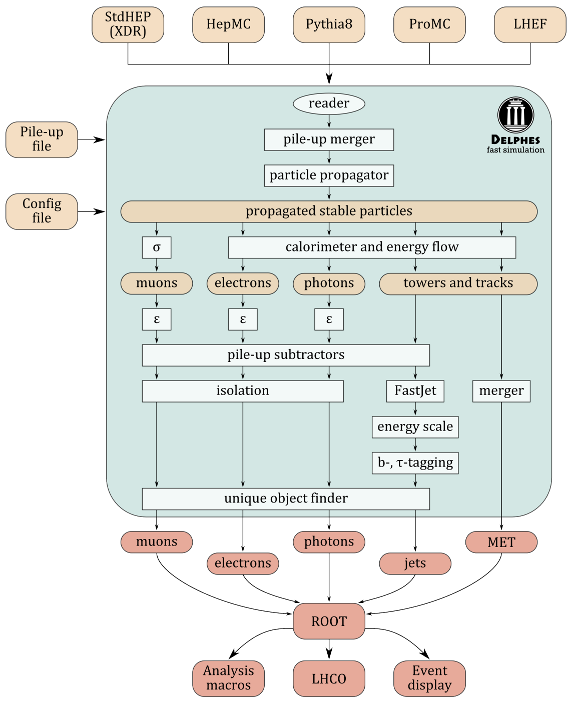


Figure 3: Summary of DELPHES modules and function. Source: [23].

2.2 Generative Adversarial Networks

Traditional and well developed machine learning can be understood very much in parallel with the Bayesian methods of experimental physics. We attempt to determine the most likely parameters, θ, for a given model, $\mathcal{M}$, for measurements y. This probability, $p(\theta|y,\mathcal{M})$, is the posterior. The usual technique is maximisation of the likelihood (probability of data given parameters) [24]:

$$p(y|\theta,\mathcal{M}) = \frac{p(\theta|y,\mathcal{M})p(y|\mathcal{M})}{p(\theta|\mathcal{M})}$$

In machine learning, computers perform this task using training data to learn the parameters, then subsequently make predictions on new data. Typically these actions are classification (labelling), regression (common in physics) and clustering (similarity of data points).

More recent developments have made significant inroads into *generative* models, wherein learning is done in order to produce new samples of data. In particular neural networks and deep learning have played a large role, as neural networks scale well with dimensions, are end-to-end differentiable (crucial for gradient based training) and can represent complex functions [25].

In the simplest case GANs consist of two competing networks G and D. G takes a latent noise variable z as input and outputs artificial data $\mathbf{y} = G(z;\theta_g)$, where θ_g are the parameters of the network. D takes either real or artificial data as input and outputs a scalar, $D(\mathbf{y})$, corresponding to the probability that $\mathbf{y}$ is real. G is trained to improve at fooling D and D is trained to improve at distinguishing real data from generated. This can be described by a min-max game with value function $V(D,G)$:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{y}\sim p_{\text{data}}(\mathbf{y})}[\log(D(\mathbf{y})] + \mathbb{E}_{\mathbf{z}\sim p_z}[\log(1-D(G(z)))], \quad (1)$$

where $\mathbb{E}_{\mathbf{y}\sim p_{\text{data}}(\mathbf{y})}$ expresses expectation over the data probability distribution. The system is summarised in Fig. 4.

It can be shown that there exists a unique solution to this, a saddle point, strictly when $p_{\text{data}} = p_G$, where p_G is the distribution produced by G [4]. This theoretical guarantee is a key advantage of GANs, along with the ability to train them using standard back propagation algorithms and the lack of Markov Chains which are often needed in other generative models. By modifying the parameters to bring the system closer to this optimum (training, e.g. by a gradient descent method such as ADAM [26]), we achieve a successful data generation scheme.

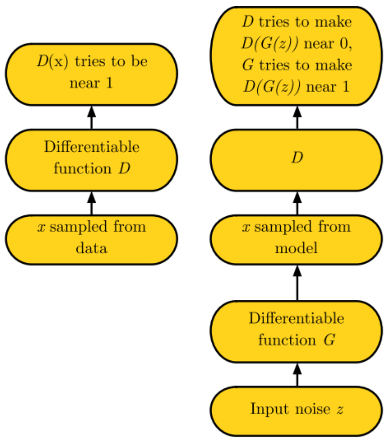

$D(x)$ tries to be near 1
Differentiable function D
x sampled from data
D tries to make $D(G(z))$ near 0, G tries to make $D(G(z))$ near 1
D
x sampled from model
Differentiable function G
Input noise z

Figure 4: Generative Adversarial Nets process. Adapted from Ref. [9].

2.3 Boosted Jets

The LHC has been able to probe unprecedented energy scales, especially with the upgrade to 13 TeV operation. For the first time, large numbers of massive particles (i.e. W, Z, top quark, Higgs boson) are being produced with transverse momenta p_T considerably larger than their rest mass m. Traditional reconstruction techniques identify all the decay products of such objects as a single jet, making distinguishing such jets from the large background of QCD jets a prominent problem [10]. As these *boosted* objects may also include contributing effects from physics beyond the standard model, there is considerable active study in this area.

Modern methods probe individual jets and their detailed substructure for identification and tagging. A key property is the number of hard ‘prongs’ of radiation in a jet. Electroweak boosted objects ($W/Z/H$) produce two prongs, while a boosted top quark produces three [27]. Background QCD processes only lead to a single hard prong [28].

The process considered in this investigation is the decay of a W' boson. The W' and Z' are massive gauge bosons which arise in extensions to electroweak theory. The simplest example of which is an additional $SU(2)$ gauge symmetry, i.e. $SU(2)_1 \times SU(2)_2 \times U(1)$, which is spontaneously broken to produce the standard electroweak $SU(2)$ [29]. These particles are predicted to have masses on the order of TeV.

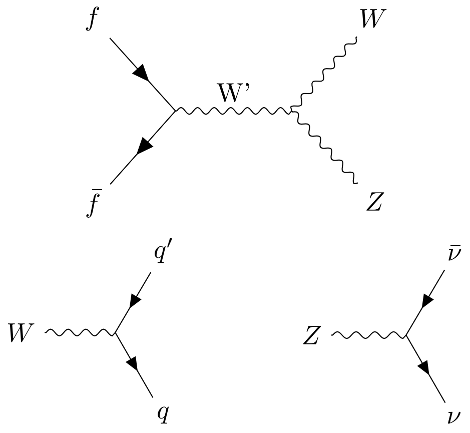


Figure 5: Feynman diagram for process under consideration. Fermion annihilation produces W' decaying to boosted W and Z.

The decay sequence imposed in this project is given by the Feynman diagrams in Fig. 5. The large mass of the decaying W' means the W is highly boosted, and decays to quarks producing a boosted jet. By forcing the Z boson to decay to neutrinos, we ensure it does not complicate our signal; it decays “silently”. The boosted W is expected to produce a two-pronged pattern. This study contrasts this process with background QCD jets.

3 Methods

3.1 Jet-Images

3.1.1 Calorimeter to Image

The jet-image generation largely follows the scheme described in Ref. [12], adapted for extraction from Delphes output ROOT files. Training data generation begins by running Delphes 3 (with internal Pythia) using the *DelphesPythia8* command [30]. Pythia is configured for collisions at $\sqrt{s} = 13\,\text{TeV}$, and a W' boson mass of 800 GeV. Delphes is configured to use the FastJet [31] package for jet-clustering (assigning particles to jets) using the anti-k_t algorithm [32], with a radius parameter of $R = 1.0$. The jet in each event with the highest transverse momentum is selected for the image. Trimming is also performed in this step, see Section 3.1.2 for details.

The output ROOT file (see Fig. 3) is processed to extract information about jet constituents. Jet-image axes correspond to the orthogonal directions of azimuthal angle ϕ and pseudorapidity η; pseudorapidity is defined by the polar angle θ as $\eta \equiv -\ln(\tan(\theta/2))$. Pixels of size 0.1×0.1 span a grid of angle values $\eta \times \phi \in [-1.25, 1.25] \times [-1.25, 1.25]$, forming a 25×25 pixel image. The intensity of pixel i, denoted I_i, is given by the sum of transverse energies (assuming clusters are massless, as shown in Appendix A) over all the calorimeter cells, indexed by c, that fall within the pixel[1],

$$I_i = \sum_c p_{T,i}^{(c)} = \sum_c \frac{E_i^{(c)}}{\cosh \eta_i^{(c)}}, \tag{2}$$

where $E^{(c)}$ and $\eta^{(c)}$ are the energy and pseudorapidity of cell c. Figure 6 shows a particle hitting an idealised cylindrical calorimeter tower configuration with a superimposed pixel grid.

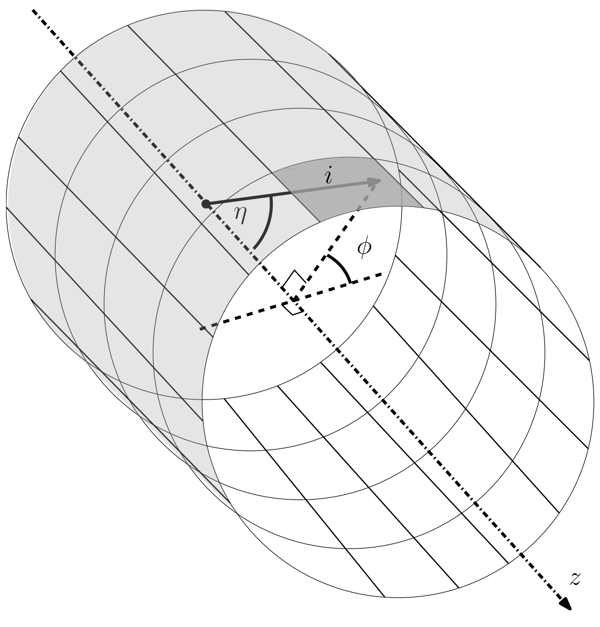


Figure 6: Idealised cylindrical calorimeter tower arrangement, with particles colliding along the z axis. A scattered particle hits a calorimeter cell within pixel i at polar angle and pseudorapidity values ϕ and η. Pixel division sizes are not representative; pseudorapidity diverges when approaching the $\pm z$ axis.

[1]The longitudinal segmentation of calorimeter towers is collapsed to a flat cylinder in our idealised picture

3.1.2 Pre-Processing

Pre-processing the images to exploit the inherent physical symmetries and remove obfuscating variations significantly improves performance. The steps employed are as follows:

1. **Trimming** [33]: The anti-k_t algorithm is applied to the jet to cluster it in to sub-jets with $R = 0.3 k_t$, and those with less than 5% of the transverse momentum of the overall jet are dropped. This helps to highlight the hard event under consideration, and reduces the effect of pileup (multiple proton-proton collisions in the same event). The trimming step is carried out in the initial Delphes run, via FastJet.

2. **Translation**: Using the sub-jet information from the trimming step, the jet is translated so that the sub-jet with highest transverse momentum is at the centre of the image. This is performed when initially pixelising the calorimeter measurements. Translations in ϕ are rotations about the collision axis, so pixel intensities are invariant. However, translations in η are Lorentz boosts. With the pixel intensity as defined in Eq. (2), the pixel intensity also remains invariant under such translations (see Appendix A).

3. **Rotation**: Once the image has been read in, it is rotated such that the sub-leading jet is directly below the leading, i.e. at an angle of $-\pi/2$ with the origin at the centre of the image. If the angle of rotation is not a multiple of $\pi/2$, which in general it is not, the rotated pixel grid will not align with the original. Thus for a general affine rotation pixel intensity redistribution via interpolation is required, this is performed using a cubic spline interpolation. If no second sub-jet is present, the image is rotated such that the principle component axis is vertically downwards. To counter the effects of interpolation, the sum of intensities is renormalised to be equal to the value before rotation in order to minimise the information lost.

4. **Flip**: The image is reflected about the central vertical axis if required such that the right is always the half with the higher total intensity. This further helps make sure the hardest features (the ones of physical interest) appear in similar positions, aiding the training procedure.

The final three steps are demonstrated in Fig. 7 using a simplified image. Figure 8 shows a random sample image generated from Pythia + Delphes (PD) with all pre-processing steps applied, more samples are given in Appendix B. In order to marginalise the effect of variations in transverse momentum for this study, p_T of jets used for training is restricted to $250\,\text{GeV} \leq p_T \leq 300\,\text{GeV}$. Similarly, a further cut on jet mass is imposed by requiring $60\,\text{GeV} \leq m \leq 100\,\text{GeV}$. Both cuts are performed using the value provided by the clustering algorithm for each jet. All jet-image generation was performed[2] using Python v2.7, using PyROOT (the Python interaction module for ROOT [34]) to read in Delphes objects. Numpy [35] was used for image processing, alongside Scikit-image [36] for the rotation. Matplotlib [37] was used to produce all plots in this report.

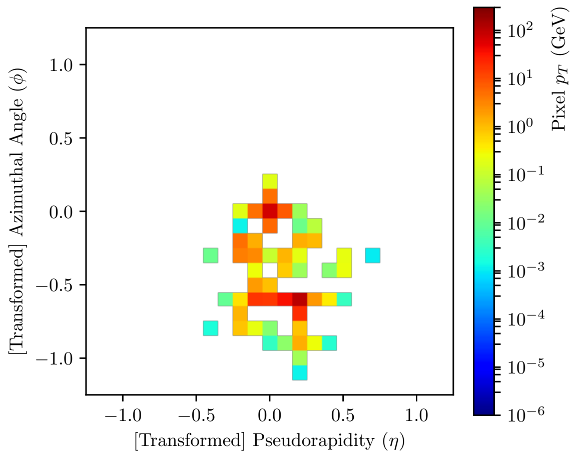


Figure 8: Sample jet-image from Pythia + Delphes data after pre-processing. High intensity centre (leading sub-jet) and sub-leading sub-jet directly beneath are seen. Pixel values are plotted on a log scale.

3.2 GAN

The GAN architecture and training used are fully discussed in [17], and are summarised here.

3.2.1 Architecture

The Locally Aware GAN (LAGAN) architecture used builds upon the Deep Convolutional formulation [5], which essentially consists of several convolutional filter layers before the fully connected neural network. This helps identify and generate specific features in an image. Whereas in the convolutional case a filter patch is slid across the entire image, in the Locally Aware case patches are assigned to a given part of the image. Therefore rather than N filters being convolved with the whole image, there are N distinct filters applied per patch of image, which are trained independently. This is key to breaking translational invariance and producing well defined local features as required.

Further deviations from traditional GANs are required to compensate for differences between natural images and jet-images. Jet-image pixel intensities are not confined to a fixed range (such as 0 - 255 in an 8-bit grayscale image), but rather need to cover several orders of magnitude. Furthermore, in a given jet-image the majority of pixels have a null value, leading to sparsity. The final non-linear activation layer in G is chosen to be a Rectified Linear Unit (ReLU) [38], which performs the operation

$$f(x) = \max(0, x)$$

on an input x. This helps produce a large number of null cells, and an unbounded maximum.

A common disadvantage of GANs is a proclivity for G to overwhelmingly produce a single sample which D struggles to classify, this is known as *mode-collapse* [4]. Minibatch discrimination [39], used here, allows D to exploit batch-level features, making collapse unfavourable for G. It also proved key for achieving high dynamic range and sparsity.

An auxiliary classification task for the discriminator, as described in the ACGAN system [6] is also employed. The Discriminator, as well as determining whether an image is real or fake, also assigns it a label corresponding to whether it is a 'signal' (W' jet) or 'noise' (QCD jet). Similarly, the Generator is tasked with producing an image conditioned on an input label corresponding to the process. Both models are thus minimising a second loss function L_C which can be expressed in terms of log-likelihoods as

$$\begin{aligned} L_C = &- \mathbb{E}[\log P(C = c \,|X_{real})] \\ &- \mathbb{E}[\log P(C = c \,|X_{fake})], \end{aligned} \tag{3}$$

where $P(C = c \,|X_{real})$ indicates the probability of the assigned class being correct given the sample X is real. This has not only been shown to aid the training process, but also demonstrates that a GAN could simulate a variety of physical processes, by conditioning on the input [40]. Figure 9 shows the total architecture used in this LAGAN system.

[2]Code available at github.com/ss2165/delphes-gans. Also submitted on USB drive with report.

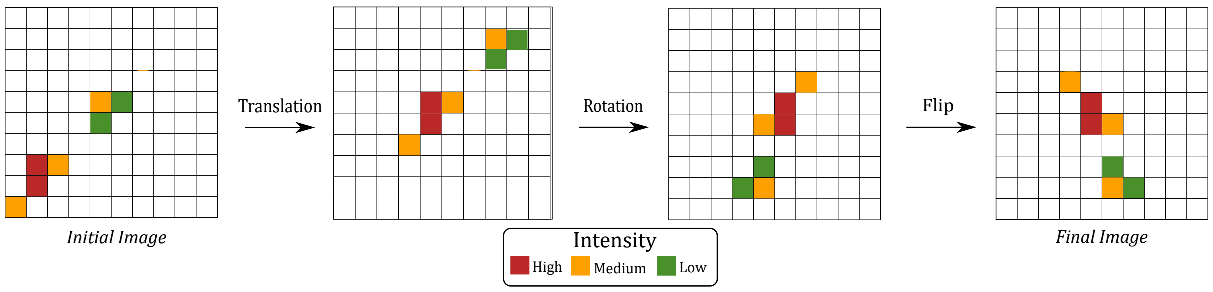


Figure 7: Demonstration of final three pre-processing steps applied to jet-images produced from PD output, with a simplified toy image. Details of the steps are in the text.

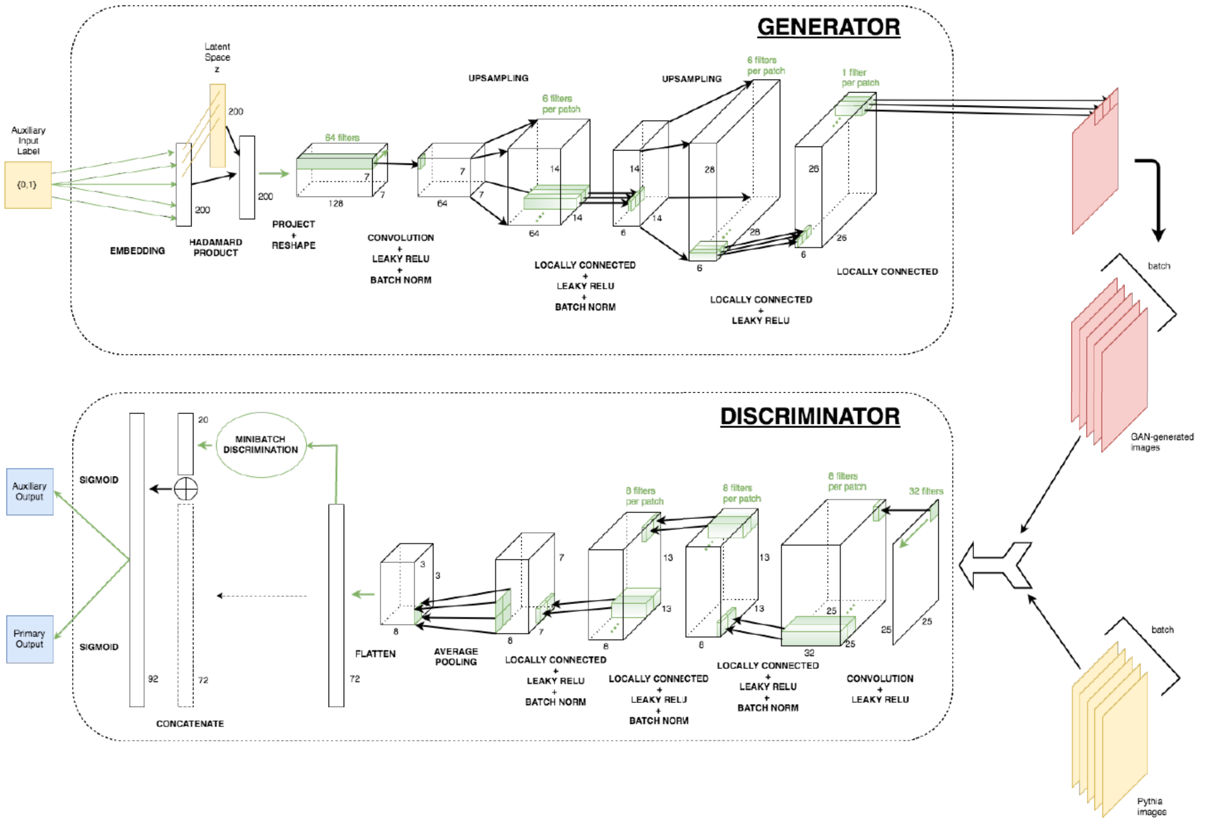


Figure 9: LAGAN architecture used for training. Source: [17].

3.2.2 Training

Training was performed with gradient descent using the ADAM optimiser [26]. The noise variable input to the discriminator was a normally distributed vector of length 200, with 0 mean and standard deviation 1. Training was performed on batches of 100 images at a time, for 50 epochs (total runs of training over the whole dataset). Models were built and trained using KERAS v1.2 [41] with a TENSORFLOW v0.11 [42] backend.

Training was performed for two training image datasets of size ~6k and ~25k, each set containing approximately equal numbers of the two classes of jet-image (W' and QCD). A single training sequence on the larger dataset took approximately 35 h to run on the computational resources available (CPU), and so was determined to be a reasonable maximum size. The original PYTHIA only investigation was able to use GPUs, which significantly reduce training time, thus the authors could make use of a 800k image dataset.

4 Results and Discussion

This section presents and examines the outcomes of the investigation. Section 4.1 evaluates the effect of DELPHES on the produced training dataset of jet-images. The GAN generated images are presented in Section 4.2, and distributions they produce in associated jet variables are considered in Section 4.3. Section 4.4 outlines the computational advantage of the scheme presented.

4.1 Delphes Jet-Images

As jet-images have not previously been produced from DELPHES output, we first characterise them. Superficially, as the general FS function is smearing of particle tracks and measurements, we would expect hard centres of radiation to be broadened and spread over the images from PYTHIA+DELPHES (PD) output, compared to purely PYTHIA (P). Average images[3] generated from ~12.5k images for both W' (signal) and QCD (noise) decays are compared for P (data from the original investigation [17]) and PD.

Figure 10 shows the sets of average images on a log scale for pixel intensity. A plot of the difference between the images on a pixel by pixel basis (P - PD) is also shown, on a linear scale. The P images show circular symmetry in the low intensity regions away from the centre. The expected two hard prongs are seen in the W' average, while the QCD prongs are broader, with the lower lobe at a smaller intensity. The PD images do not display the circular symmetry, likely because the low intensity region (10^{-4} GeV and below) is not captured by the dimensions of the image; it has been smeared out of frame. Further investigations should increase the dimensions to investigate this area.

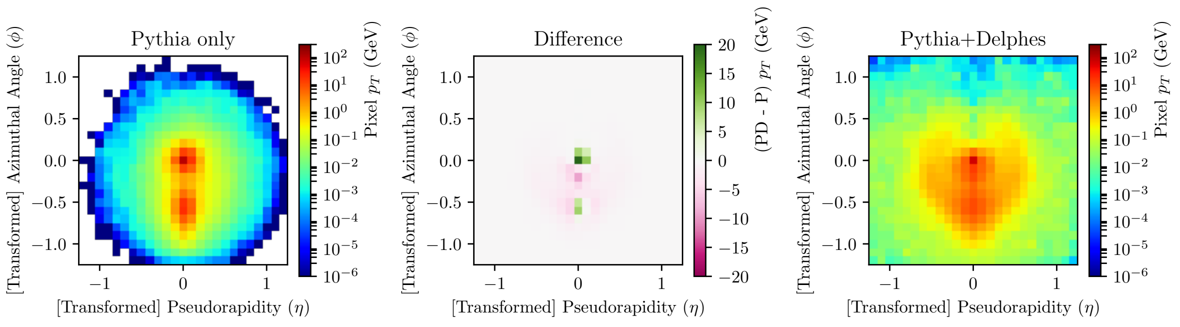
As expected, the lobes of high intensity have been spread in the PD images. The image of differences shows that in both cases the central, primary lobe, has a lower core intensity as compared to the P images, and it is spread mainly downwards in the $-\eta$ direction by DELPHES. We may infer then that the smearing of the leading sub-jet is largely towards the sub-leading. The secondary lobe (which is not distinguishable in the QCD difference) is spread in to something akin to an arrowhead pointing downwards, though it is not well distinguished from the primary lobe in some areas. This suggests that again the smearing is biased towards the primary lobe, but now with a large tangential component. This effect may also be caused by lower efficacy of the jet-finding algorithm after detector response has been factored in. Thus the sub-jets may not be as accurately translated/rotated over all images.

The blurring of high intensity cores can be better understood by comparing the pixel intensities along the central vertical axis of the image, i.e. $\eta = 0$, as shown in Fig. 11. Central peaks broaden in the $-\eta$ direction, raising the intensity of the inter-lobe region significantly in both cases. While the distinctive W' secondary peak broadens but remains visible, the previously small QCD secondary peak is now no longer separable from the decaying tail of the primary. Strict numerical conclusions are difficult to draw from so few points, further studies could use a higher resolution.

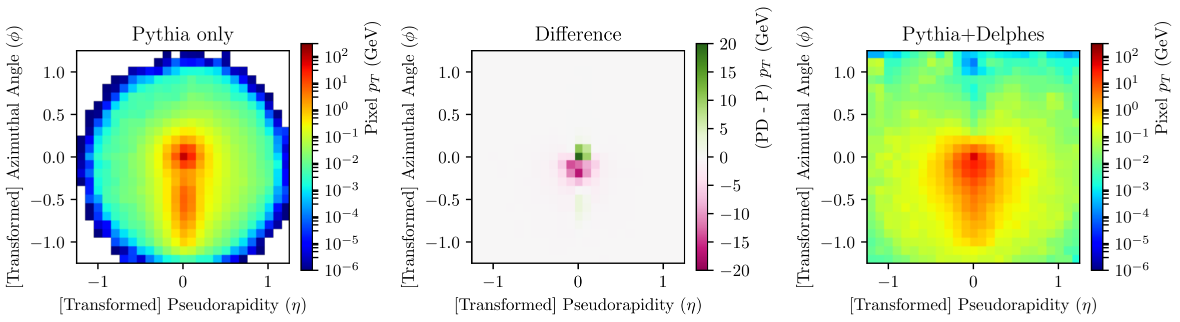
4.2 Generated Images

This section first examines the results of the training performed on the larger 25k dataset, i.e. the more fully trained Generator (G). Figure 12 is a random W' sample from G (c.f. Fig. 8), more samples are given in Appendix B. The generated images display the two key expected properties of sparsity (majority of the pixels are not activated), and the two high intensity regions at the centre and directly below the centre.

[3]Average images correspond to an array of the average value of each pixel over the set.



(a) W' signal



(b) QCD noise

Figure 10: Comparison of average jet-images between those generated from just PYTHIA (P, left) and those also generated with DELPHES (PD, right) on a log scale. The difference between the two images (P - PD on a pixel by pixel basis) is shown in the middle on a linear scale, with the colour map clipped to ± 20 GeV to aid visibility. Images for the W' signal (top) and QCD noise (bottom) are shown. PYTHIA only images from Ref. [17].

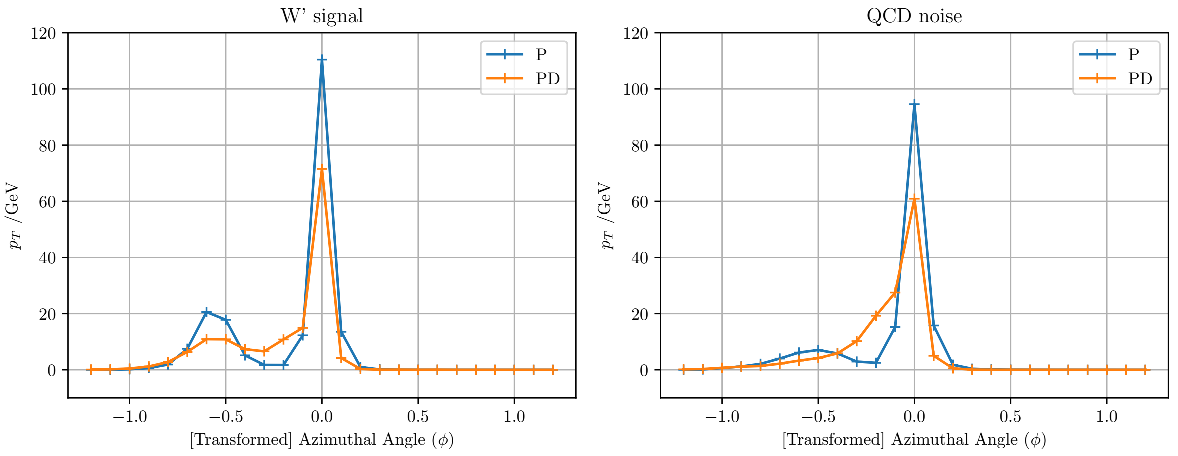


Figure 11: Pixel values for $\eta = 0$ axis from average jet-images (over 12.5k images) for W' signal (left) and QCD noise (right). Each plot compares tracks for P and PD images.

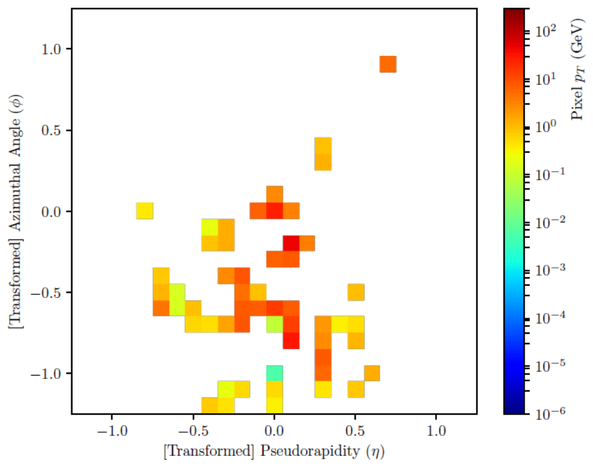


Figure 12: Sample jet-image produced by a LAGAN Generator after being trained on a 25k PD image dataset. Pixel values are plotted on a log scale.

4.2.1 Average Image Comparison

In order to assess the quality of the generated image distribution, a sample set was produced of the same size as the training PD set (12.5k images for each class). Average images are compared to the training data in Fig. 13, as in Fig. 10. In the central and mid-range regions of the image the magnitude of intensities match training data, with a high intensity central region. Moreover, the W' high intensity region also includes a secondary lobe, while QCD does not.

However, deficiencies in G images are readily apparent. The most clear is the empty outer regions of the averages, where negligibly few images have non-zero pixel values. This is largely due to the difficulties in achieving such large range in the output of the Generator. Training to achieve accuracy in the high intensity region comes at the price of a crude low intensity region. The ReLU activation layer, if insufficiently or incorrectly trained could have lead to too many pixels which are never activated.

Examining the difference images shows training images have higher intensity cores, but the generated images match here to within 20%. Notably, the G average image contains certain high intensity pixels in low intensity regions (especially for W'). Similarly, there are also certain pixels in high intensity regions which are never activated. This can be explained by some degree of mode-collapse, wherein G is disproportionately activating some pixels and never activating others. This can likely be overcome to an extent with a larger training set, the effect of dataset size is discussed further in Section 4.3.

4.2.2 Interim Images

Some insight in to the training procedure and evaluation of the hyper parameters can be achieved by looking at intermediate results. For standard machine learning applications we could track training quality using the loss function. The GAN loss function (Eq. (1)), however, is notoriously not a useful measure of training, an issue recent reformulations are attempting to resolve [43].

We may instead examine the loss function for the auxiliary classification task (Eq. (3)). Figure 14 shows the loss varying over the epochs of training (for the 25k dataset), evaluated on training sets of images and a separate test set, for both G and D (the equivalent plot for GAN loss is included in Appendix C for completeness). Also shown are average images from G at intermediate epochs for both classes. The loss variation suggests gains made beyond 20 epochs are marginal. This is somewhat supported by the images, which superficially vary little from this point onwards. The key difference between the two classes, two lobes for signal and one for noise, is seen to develop at around epoch 30.

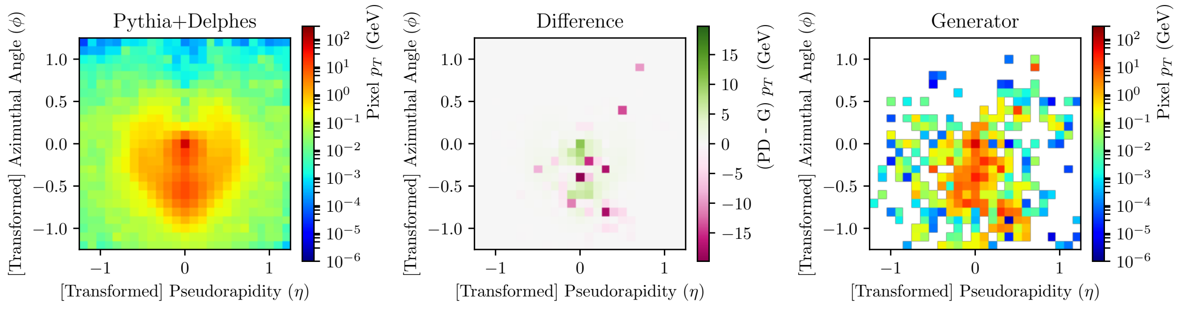
The value in further training may be seen in the sparsity. After beginning as random noise, initial learning produces a highly sparse average image with compact central cores. Subsequent images show more and pixels in the low-intensity regions being activated. While training up to or beyond 50 epochs may be useful in this regard, it is clear that more training data would be far more potent.

4.3 Physical Distributions

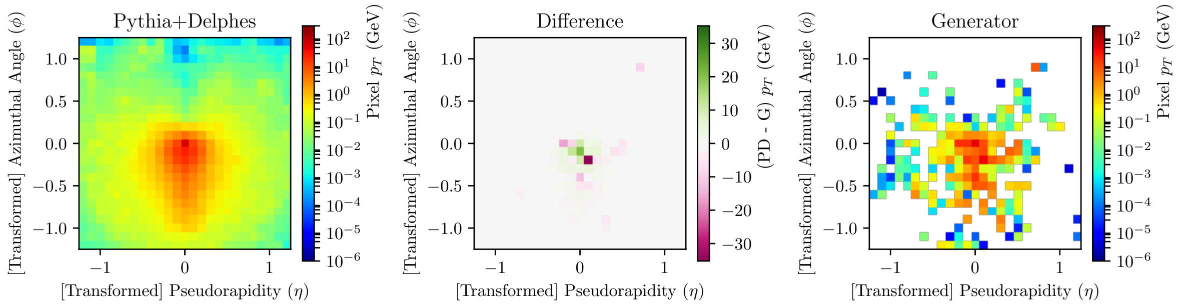
Characterising a distribution of images, a 625 dimension distribution in the case of 25×25 pixels, is difficult to achieve numerically. Mapping images to a single variable, ideally a physically motivated one, gives a one dimensional distribution which could provide insight. Here we calculate two such distributions, the first is the 'discretised' transverse momentum p_T [17], calculated from the pixel values I_i of an image I as

$$p_T^2(I) = \left(\sum_i I_i \cos(\phi_i)\right)^2 + \left(\sum_i I_i \sin(\phi_i)\right)^2$$

where η_i and ϕ_i are the pseudorapidity and azimuthal angle, respectively. The second is *n-subjettiness* τ_n [27], a measure of the extent to which



(a) W' signal



(b) QCD noise

Figure 13: Comparison of average jet-images (over 12 500 images) between training data (PD, left) and those from the trained Generator (G), on a log scale. The difference between the two images (PD - G) is shown in the middle on a linear scale. Images for the W' signal (top) and QCD noise (bottom) are shown.

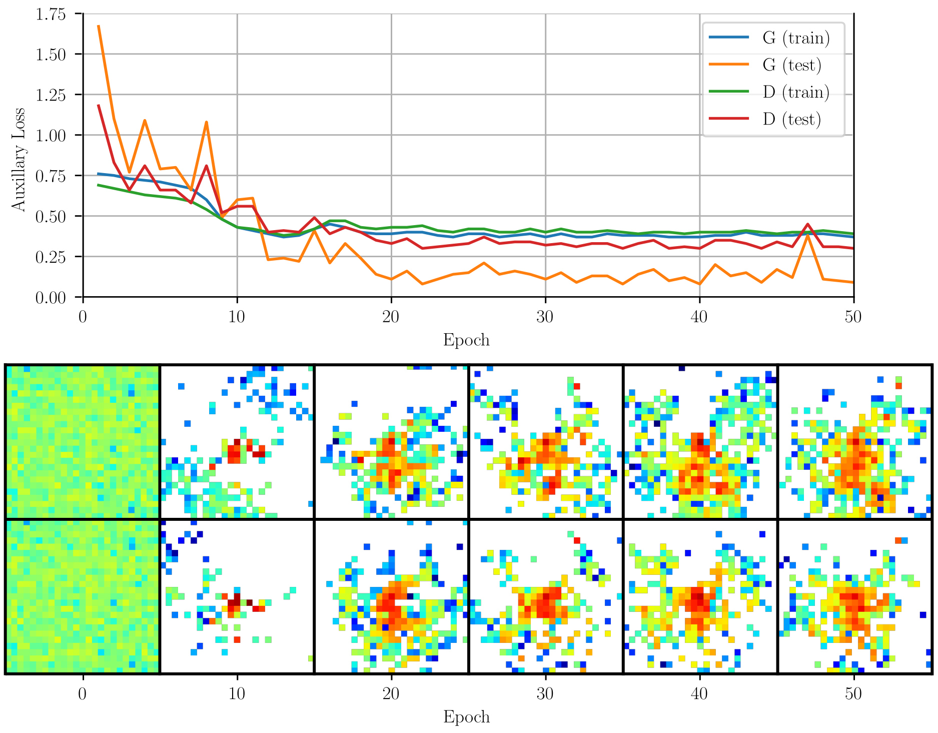


Figure 14: Variation in loss function for auxiliary task with training epoch, evaluated on training and testing data sets for Generator (G) and Discriminator (D) (top). Average images from G at intermediate epochs for W' signal and QCD noise (bottom). The scale for pixel intensities is the same as previous figures, for example Fig. 13.

a jet can be clustered in to n sub-jets (more likely to be n sub-jets for smaller τ_n). We can calculate the ratio τ_{21} using

$$\tau_n(I) \propto \sum_i I_i \min_a \left(\sqrt{(\eta_i - \eta_a)^2 + (\phi_i - \phi_a)^2} \right),$$

$$\tau_{21}(I) = \tau_2(I)/\tau_1(I)$$

where η_a and ϕ_a are axis values determined with the one-pass k_t axis selection using the winner-take-all combination scheme [44]. The ratio τ_{21} when measured for traditional jet data proves to be a useful distribution for discriminating two-pronged jets such as boosted W from QCD background, as shown in Fig. 15.

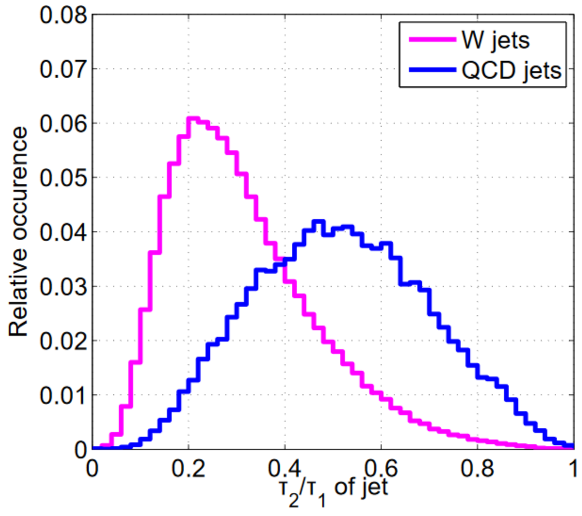


Figure 15: Distribution of τ_{21} for boosted W and QCD jets, from Ref. [27]. An invariant mass window of 65 GeV $< m_{jet} <$ 95 GeV was imposed on jets of R=0.6, $p_T >$ 300 GeV and $|\eta| < 1.3$.

The two distributions are calculated for the PD datasets of size 6k and 25k, and the Generators trained on the respective set. Figure 16 shows the results. The generated transverse momentum distributions are broad compared to the PD distributions in the 6k case, but in the 25k case show a defined peak at approximately the same location as PD. Similarly, in the 6k case generated τ_{21} distributions both peak at approximately 0.6, while the PD W' distribution peaks noticeably lower than QCD (c.f. Fig. 15). However, in the 25k case the peaks show a clear distinction in the correct direction, though the peak locations do not match those of PD.

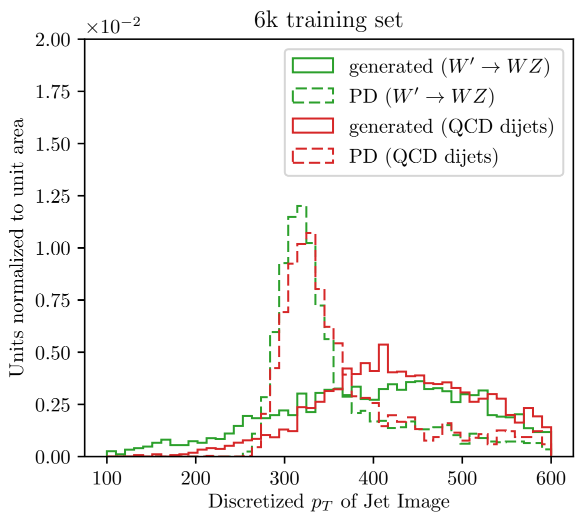
These observations show the value of more training data, and suggest the same scheme repeated with a training set of order 10^6 images (as in the original investigation) could lead to significantly more accurate generated images. The well defined separation in τ_{21} also shows G is producing output conditioned on the process (the difference in average images seen in Fig. 10 also supports this argument). We may also conclude that the Discriminator is showing some effectiveness in distinguishing W' and QCD images.

4.4 Computational Performance

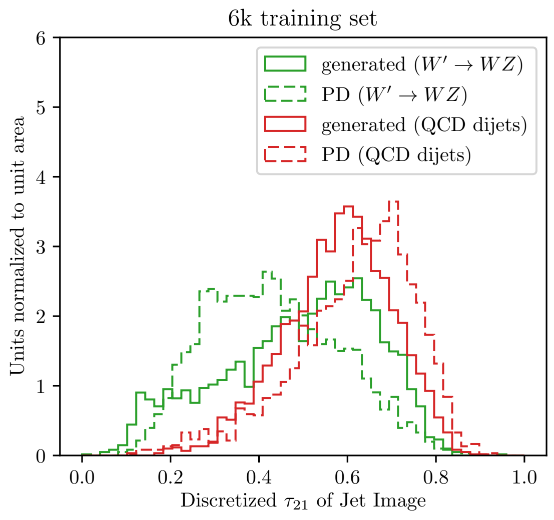
As the initial motivation for using machine learning was computing performance gains, run times were compared for *DelphesPythia8* calls and the Generator producing the same number of jet-images. Both tasks were performed on the same Intel® Core™ i5-3570 @ 3.40 GHz machine.

A logarithmic plot of runtime t against number of events N is shown in Fig. 17. Linear relationships between logarithms of the variables of the form $\log(t/s) = a\log(N) + b$ were fit to the data, the resulting coefficients are given in Table 1. The PD system functions linearly, calculating results for one event after another, so we would expect the runtime to grow as $\mathcal{O}(N)$. The fitted gradient is close to unity, corroborating this prediction. The Generator, however, is highly non-linear in operation thus the relationship is not well modelled by a straight line.

At large event numbers, $N \geq 10^3$, G provides an order of magnitude speed advantage. These results are for a CPU, a GPU would likely facilitate another order of magnitude. As many HEP applications may not have a GPU available for simulation, it is still instructive to consider the CPU only case. Of course, the direct comparison of rates is not entirely justified as a jet-image does not contain all the information generated by PYTHIA and DELPHES, so these numbers are merely indicative.



(a) Discretised transverse momentum.



(b) n-subjettiness

Figure 16: Distribution of discretised image p_T (top) and n-subjettiness τ_{21} (bottom) for 6k (left) and 25k (right) training set size.

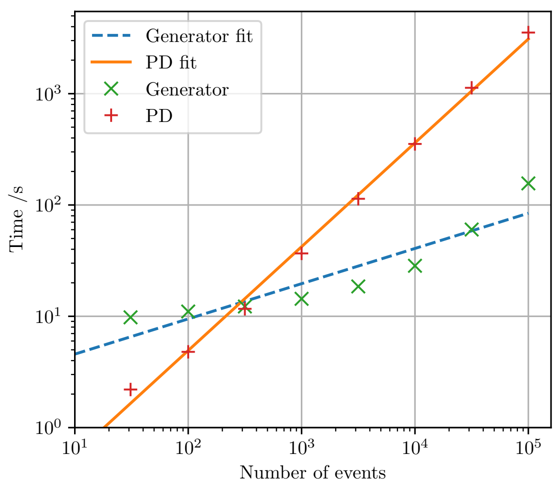


Figure 17: Plot of runtime t against number of events N for the PYTHIA+DELPHES (PD) system and trained LAGAN Generator (G). Linear fits to logarithms of the variables are also shown, see Table 1 for coefficients of the fit.

Method	Gradient, a	Constant, b
PYTHIA+DELPHES	0.32 ± 0.07	0.3 ± 0.2
Generator	0.93 ± 0.03	-1.2 ± 0.1

Table 1: Coefficients for linear fits of the form $\log(t/s) = a \log(N) + b$ fit to data of runtime t against number of events N. Fitted lines are shown in Fig. 17

5 Conclusions

This study aimed to explore the use of machine learning tools to mimic the function of fast, approximate detector simulations. Based on recent work and promise, Generative Adversarial Networks (GANs) were chosen as the generative model. The processes under consideration were jets caused by a boosted W produced from the decay of a hypothetical W' gauge boson, and QCD background jets.

Calorimeter tower energy data from PYTHIA and DELPHES simulations were cast in to 2D images in the form of jet-images. Pre-processing steps of trimming, translation, rotation, and flip were applied to the images to produce training data sets. Compared to images generated without DELPHES, they were observed to have more smeared or broadened regions of high intensity radiation as predicted. A Locally Aware GAN, optimised for generating distinctive local features in the images, was trained using this data.

Generated images were superficially well matched with training data and were seen to reproduce the key features of radiation prongs and sparsity in average image comparisons. Inactive and erroneous pixels were evidence for some degree of *mode-collapse*, alongside a sub-optimally trained rectified activation final layer. Comparing distributions of the physically motivated variables transverse momentum and n-subjettiness showed the benefits of larger training datasets as well as capability to produce two distinguishable classes of output. The predicted increase in computational efficiency for event generation was also measured.

The results of this investigation demonstrate that GAN based emulation of detector simulator behaviour is indeed a viable and productive goal. The presented scheme agrees reasonably well in kinetic distributions, and is computationally more efficient than the original simulation. Future work should aim to explore whether more aspects of DELPHES output could be encoded and generated in image format, whilst also improving the accuracy of images to match existing simulators.

Acknowledgements

I would like to thank Dr. Christopher Lester and Dr. Thomas Gillam for their supervision, guidance and input throughout. I also thank Dr. Gilles Louppe and Dr. Ben Nachman for their help and suggestions.

References

[1] J. Allison et al. "Geant4 developments and applications". In: *IEEE Trans. Nucl. Sci.* 53 (2006), p. 270. DOI: 10.1109/TNS.2006.869826.

[2] J. de Favereau et al. "DELPHES 3, A modular framework for fast simulation of a generic collider experiment". In: *JHEP* 02 (2014). DOI: 10.1007/JHEP02(2014)057. arXiv: 1307.6346 [hep-ex].

[3] P. Demin and M. Selvaggi. "Delphes 3 a modular framework for fast simulation of a generic collider experiment". 2014. URL: https://indico.in2p3.fr/event/10280/contribution/0/material/slides/0.pdf.

[4] I. J. Goodfellow et al. "Generative Adversarial Networks". In: *ArXiv e-prints* (June 2014). arXiv: 1406.2661 [stat.ML].

[5] A. Radford, L. Metz and S. Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *ArXiv e-prints* (Nov. 2015). arXiv: 1511.06434 [cs.LG].

[6] A. Odena, C. Olah and J. Shlens. "Conditional Image Synthesis With Auxiliary Classifier GANs". In: *ArXiv e-prints* (Oct. 2016). arXiv: 1610.09585 [stat.ML].

[7] S. E. Reed et al. "Learning What and Where to Draw". In: *CoRR* abs/1610.02454 (2016). arXiv: 1610.02454 [cs.CV].

[8] S. E. Reed et al. "Generative Adversarial Text to Image Synthesis". In: *CoRR* abs/1605.05396 (2016). URL: http://arxiv.org/abs/1605.05396.

[9] I. J. Goodfellow. "NIPS 2016 Tutorial: Generative Adversarial Networks". In: *CoRR* abs/1701.00160 (2017). URL: http://arxiv.org/abs/1701.00160.

[10] A. Altheimer et al. "Boosted objects and jet substructure at the LHC". In: *The European Physical Journal C* 74 (2014). DOI: 10.1140/epjc/s10052-014-2792-8.

[11] J. Cogan et al. "Jet-Images: Computer Vision Inspired Techniques for Jet Tagging". In: *JHEP* 02 (2015), p. 118. DOI: 10.1007/JHEP02(2015)118. arXiv: 1407.5675 [hep-ph].

[12] L. de Oliveira et al. "Jet-images deep learning edition". In: *JHEP* 07 (2016), p. 069. DOI: 10.1007/JHEP07(2016)069. arXiv: 1511.05190 [hep-ph].

[13] P. T. Komiske, E. M. Metodiev and M. D. Schwartz. "Deep learning in color: towards automated quark/gluon jet discrimination". In: *JHEP* 01 (2017), p. 110. DOI: 10.1007/JHEP01(2017)110. arXiv: 1612.01551 [hep-ph].

[14] L. G. Almeida et al. "Playing Tag with ANN: Boosted Top Identification with Pattern Recognition". In: *JHEP* 07 (2015), p. 086. DOI: 10.1007/JHEP07(2015)086. arXiv: 1501.05968 [hep-ph].

[15] P. Baldi et al. "Jet Substructure Classification in High-Energy Physics with Deep Neural Networks". In: *Phys. Rev.* D93.9 (2016), p. 094034. DOI: 10.1103/PhysRevD.93.094034. arXiv: 1603.09349 [hep-ex].

[16] T. Sjöstrand, S. Mrenna and P. Skands. "A brief introduction to PYTHIA 8.1". In: *Computer Physics Communications* 178.11 (2008), pp. 852–867.

[17] L. de Oliveira, M. Paganini and B. Nachman. "Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis". In: (2017). arXiv: 1701.05927 [stat.ML].

[18] G. Louppe et al. "QCD-Aware Recursive Neural Networks for Jet Physics". In: (2017). arXiv: 1702.00748 [hep-ph].

[19] ATLAS-Saclay. *Le dtecteur ATLAS*. URL: http://atlas-saclay.in2p3.fr/public/detecteur.html (visited on 09/05/2017).

[20] W. W. Armstrong et al. *ATLAS: Technical proposal for a general-purpose p p experiment at the Large Hadron Collider at CERN*. Tech. rep. 1994.

[21] 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". In: *JHEP* 2014.7 (2014), pp. 1–157.

[22] M. H. Seymour and M. Marx. "Monte Carlo Event Generators". In: *Proceedings, 69th Scottish Universities Summer School in Physics : LHC Phenomenology (SUSSP69): St.Andrews, Scotland, August 19-September 1, 2012*. 2013, pp. 287–319. arXiv: 1304.6677 [hep-ph].

[23] P. Demin and M. Selvaggi. "Delphes 3 a modular framework for fast simulation of a generic collider experiment". In: 2014. URL: https://indico.in2p3.fr/event/10280/contribution/0/material/slides/0.pdf.

[24] D. S. Sivia and J. J. Skilling. *Data analysis : a Bayesian tutorial.* Oxford University Press, 2006. ISBN: 0191546704.

[25] D. Janz and J. Requeima. “Deep Generative Models”. 2016.

[26] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *ArXiv e-prints* (Dec. 2014). arXiv: `1412.6980 [cs.LG]`.

[27] J. Thaler and K. Van Tilburg. “Identifying Boosted Objects with N-subjettiness”. In: *JHEP* 03 (2011). arXiv: `1011.2268 [hep-ph]`.

[28] M. Dasgupta, L. Schunk and G. Soyez. “Jet shapes for boosted jet two-prong decays from first-principles”. In: *JHEP* 04 (2016). arXiv: `1512.00516 [hep-ph]`.

[29] J. Beringer et al. “Review of particle physics particle data group”. In: *Physical Review D (Particles, Fields, Gravitation and Cosmology)* 86.1 (2012).

[30] P. Demin. *Delphes Workbook.* 10th Feb. 2017. URL: `https://cp3.irmp.ucl.ac.be/projects/delphes/wiki/WorkBook/Pythia8` (visited on 05/05/2017).

[31] M. Cacciari, G. P. Salam and G. Soyez. “FastJet User Manual”. In: *Eur. Phys. J.* C72 (2012). arXiv: `1111.6097 [hep-ph]`.

[32] M. Cacciari, G. P. Salam and G. Soyez. “The anti- k t jet clustering algorithm”. In: *JHEP* 2008.04 (2008).

[33] D. Krohn, J. Thaler and L.-T. Wang. “Jet Trimming”. In: *JHEP* 02 (2010). arXiv: `0912.1342 [hep-ph]`.

[34] R. Brun and F. Rademakers. “ROOT: An object oriented data analysis framework”. In: *New computing techniques in physics research V. Proceedings, 5th International Workshop, AIHENP '96, Lausanne, Switzerland, September 2-6, 1996.* Vol. A389. 1997, pp. 81–86. DOI: `10.1016/S0168-9002(97)00048-X`.

[35] S. van der Walt, S. C. Colbert and G. Varoquaux. “The NumPy Array: A Structure for Efficient Numerical Computation”. In: *Computing in Science & Engineering* 13.2 (2011), pp. 22–30. DOI: `10.1109/MCSE.2011.37`.

[36] S. van der Walt et al. “Scikit-image: image processing in Python”. In: *PeerJ* 2 (June 2014). DOI: `10.7717/peerj.453`.

[37] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing In Science & Engineering* 9.3 (2007), pp. 90–95. DOI: `10.1109/MCSE.2007.55`.

[38] V. Nair and G. E. Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines”. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10).* Omnipress, 2010, pp. 807–814.

[39] T. Salimans et al. “Improved Techniques for Training GANs”. In: *CoRR* (2016). arXiv: `1606.03498 [cs.LG]`.

[40] M. Mirza and S. Osindero. “Conditional Generative Adversarial Nets”. In: *ArXiv e-prints* (2014). arXiv: `1411.1784 [cs.LG]`.

[41] F. Chollet. *keras.* `https://github.com/fchollet/keras`. 2015.

[42] M. Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.* Software available from tensorflow.org. 2015. URL: `http://tensorflow.org/`.

[43] M. Arjovsky, S. Chintala and L. Bottou. “Wasserstein GAN”. In: *ArXiv e-prints* (Jan. 2017). arXiv: `1701.07875 [stat.ML]`.

[44] A. J. Larkoski, D. Neill and J. Thaler. “Jet Shapes with the Broadening Axis”. In: *JHEP* 04 (2014), p. 017. DOI: `10.1007/JHEP04(2014)017`. arXiv: `1401.2158 [hep-ph]`.

Appendix A Pixel Intensities

Pixel intensities are given in terms of calorimeter cell energy E and pseudorapidity η by

$$I = \frac{E}{\cosh\eta}$$

treating clusters as massless. Using the definition of pseudorapidity in terms of polar angle θ, $\eta \equiv -\ln(\tan\theta/2)$ we calculate

$$\cosh\eta = \frac{e^{\eta} + e^{-\eta}}{2} = \frac{(\tan\theta/2)^{-1} + \tan\theta/2}{2} = \frac{1}{\sin\theta}.$$

Thus $I = E\sin\theta$ and we have shown pixel intensities are equal to the transverse energy. To further show I is invariant under a translation in η of the form $\eta \to \eta' = \eta + \delta$ we note that such a translation is a Lorentz boost. Further noting that in the massless limit pseudorapidity is equivalent to rapidity, we see that E transforms as:

$$E \to E' = E(\cosh\delta + \cos\theta\sinh\delta)$$

Where $E\cos\theta$ is the component of the four-momentum in the direction of the boost. Rewriting $\cos\theta$ as

$$\cos\theta = \sqrt{1 - \sin^2\theta} = \sqrt{1 - \cosh^{-2}\eta} = \tanh\eta$$

We can manipulate the transformed energy according to

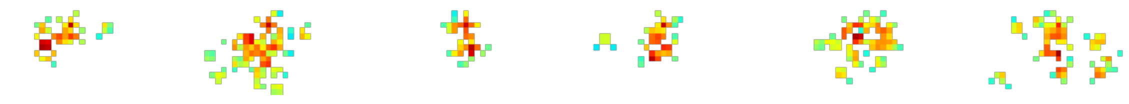
$$E' = \frac{E}{\cosh\eta}(\cosh\delta\cosh\eta + \sinh\delta\sinh\eta) = \frac{E}{\cosh\eta}\cosh(\eta + \delta)$$

Thus the overall intensity transformation,

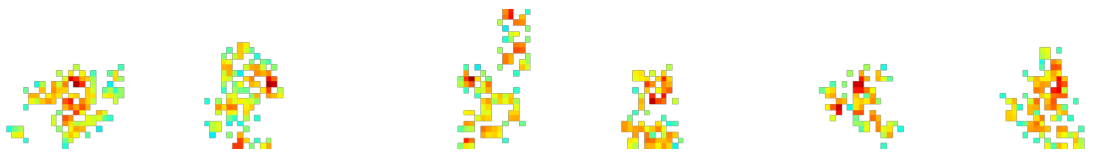
$$I \to I' = \frac{E'}{\cosh\eta'} = \frac{E}{\cosh\eta}\cosh(\eta + \delta)\left(\frac{1}{\cosh(\eta + \delta)}\right) = I,$$

is invariant under translations in η.

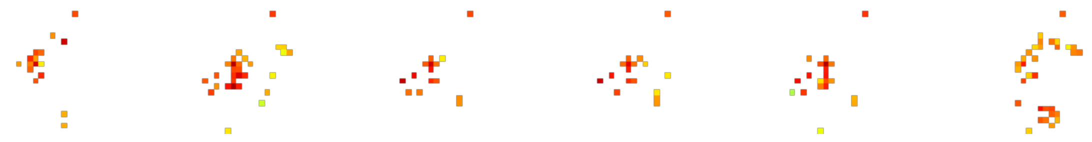
Appendix B Sample Images



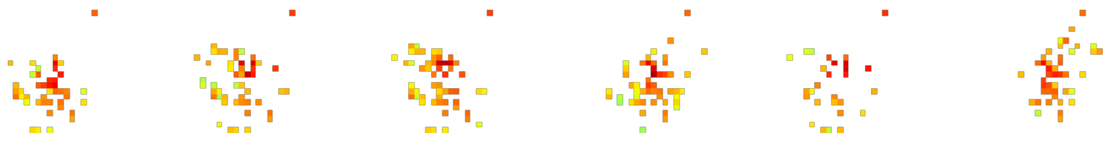
(a) PD W' signal



(b) PD QCD noise



(c) G W' signal



(d) G QCD noise

Figure 18: Radnomly samples jet-images. (a) Pythia+Delphes W' signal, (b) Pythia+Delphes QCD noise, (c) Generator W' signal, (d) Generator QCD noise. Pixel intensities correspond to the same school used in previous figures, see Fig. 8 for example.

Appendix C GAN Loss Function

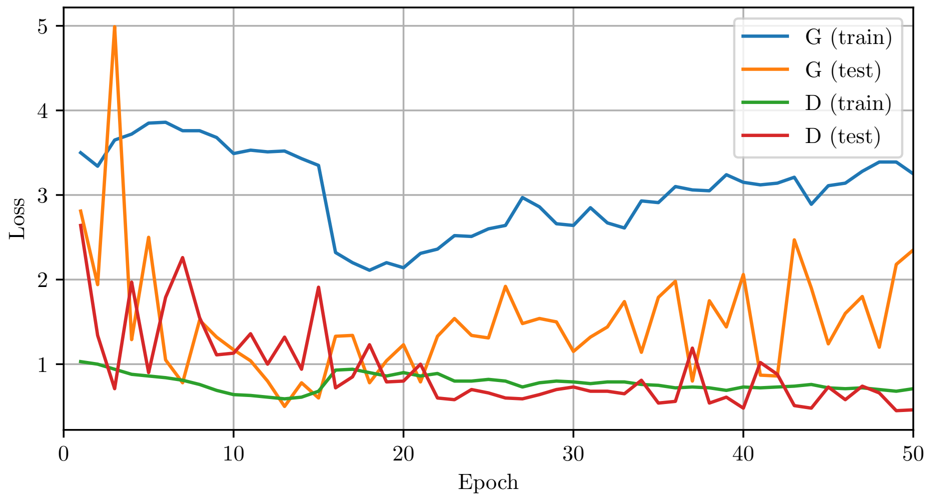


Figure 19: Variation in loss function (as given in Eq. (1)) with training epoch, evaluated on training and testing data sets for Generator (G) and Discriminator (D).