Fast Simulation of Particle Physics Detectors Using Machine Learning Tools: Initial Report

Seyon Sivarajah, Churchill College, ss2165

November 29, 2016

Supervisor: Dr Christopher Lester

Abstract

A key part of experimental particle physics is the simulation of a detector's response to an event. This initial report outlines the background and plans for an investigation into the possibility of using machine learning tools, and neural networks in particular, to learn and generate such simulated data. State of the art accurate simulators are slow and computationally expensive, a successful generative network could lead to a significant performance increase at manageable accuracy costs. This investigation is an attempt at a demonstration of this concept.

1 Introduction

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 the 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. The accuracy-performance imbalance between these two solutions leaves room for other potential avenues. One such route is the use of machine learning tools which have shown notable promise in a large number of fields.

Generative models attempt to learn a given probability distribution via exposure to samples, and thus accurately generate new elements of that distribution. Their 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.

This research project initially focusses on attempting a proof of this concept using Generative Adversarial Networks (GANs) [3]. This method trains two 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.

This investigation would train using simulated data from a FS, aiming to match the accuracy of said simulator while surpassing in performance. A successful demonstration merits further work to attempt closer accuracy to AS. The following section gives the relevant background on particle detectors, simulations and machine learning tools. Section 3 outlines the project plan, including methods, challenges and tasks.

2 Background

2.1 Detectors & Simulations

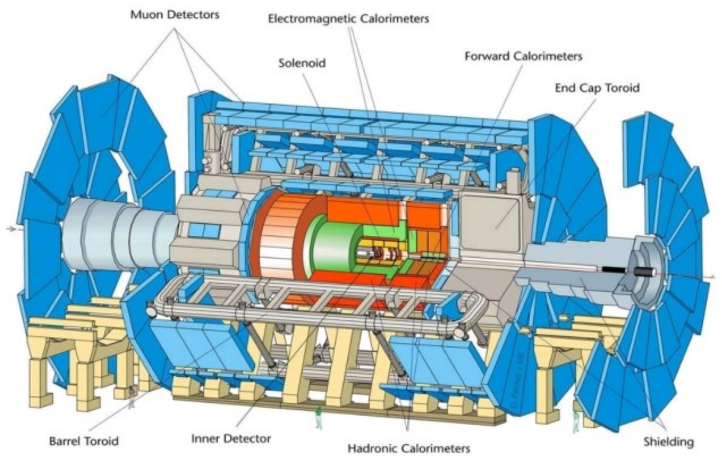

Figure 1: Cutaway diagram showing components of ATLAS detector. Source: [4]

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 gives a measurement of charge and mass respectively. A detailed description of the detector can be found in [5].

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 is a matrix element calculator, which for this investigation is the commonly used MadGraph5 [6]. 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, Pythia for our case, calculates the subsequent interactions, decays parton showers and hadronisation [7].

The results of this are the"truth events", which for our purposes is the input, $\mathbf{x}$, to any simulator under consideration. The simulator performs two crucial 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.

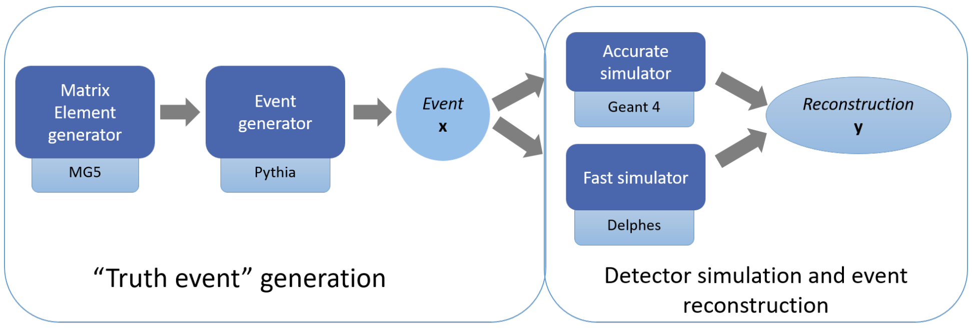

Figure 2: Summary of simulation process, from matrix element generation to final reconstruction from detector response.

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 measure of the performance of various parts can be seen in Fig. 3. The FastJet jet propagation [8] is by far the slowest, especially at high pile-up (multiple simultaneous proton-proton interactions [9]). Any performance improvements are likely to arise from better execution of this step.

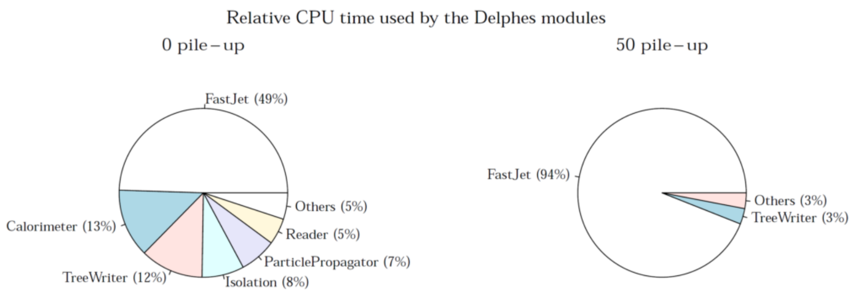

Figure 3: Performance of Delphes modules at 0 and 50 pile-up. FastJet jet propagation algorithm is seen to be most costly, especially at high pile-up. Source: [2]

2.2 Machine Learning

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 possible parameters, θ, for a given model, $\mathcal{M}$, for measurements y. This probability, $p(\theta|y,\mathcal{M})$, is the posterior. It is usually done by maximising the likelihood (probability of data given parameters) [10]:

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

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 [11], 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 [11].

Generative Adversarial Nets (GANs), proposed by Goodfellow et al. [3] have proven very successful in areas such as image generation and text to image synthesis [12, 13]. More recently, it has been shown that GANs can be used to generate particle shower shape data [14], which is promising for our goals.

Basic 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 $V(G,D)$:

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

where $\mathbb{E}_{\mathbf{y}\sim p_{\text{data}}(\mathbf{y})}$ expresses expectation over the data probability distribution.

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. This theoretical guarantee is a key advantage of GANs, as well as the ability to train them using standard back propagation algorithms and the lack of Markov Chains which are often needed in other generative models.

3 Project Plan

3.1 Method

This project aims to apply machine learning methods to simulate particle detectors, beginning with GANs. The first key step is to understand and transform the event into a suitable input for the neural network. Similarly, the reconstructed output of Delphes must be reduced to the crucial parts that G will need to replicate. This is important as the size of the output scales with event number, so could be typically 100s of megabytes, which is large compared to the image file sizes that GANs have mainly been tested on.

A key difference between basic GANs and this case is the requirement that the output $\mathbf{y}$ of G match a given input $\mathbf{x}$. G and D must therefore be conditional on the event input. Such conditional use of GANs has been demonstrated [15], and it can be shown that a simple modification of Eq. (1) is sufficient:

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

Furthermore, appending $\mathbf{x}$ to the input for each net is enough to achieve this. Fig. 4 shows this set up.

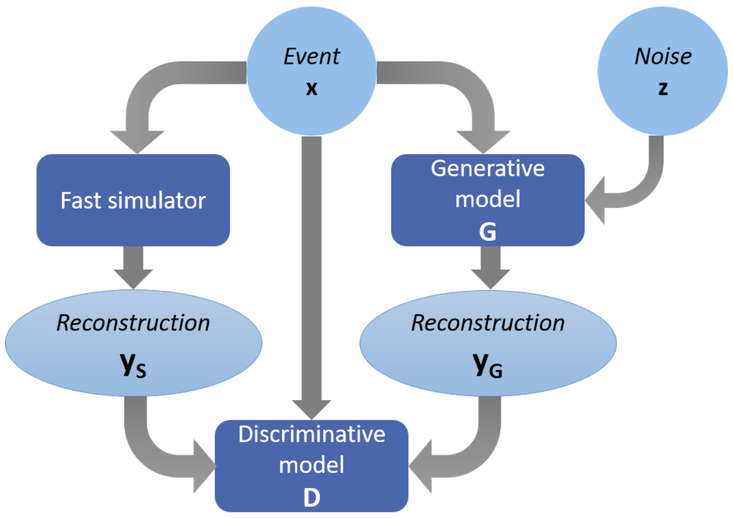

Figure 4: Schematic of simulation and neural net based generation of data.

Initially training will take place with one or a few simple events (e.g. just one or two particles) as input. Eventually the simulation must be made capable being conditional on any event input.

The project will be implemented in Python 2.7, with the associated scientific packages numpy, scipy and matplotlib. The package theano [16] will be used for optimized calculation and symbolic manipulation, primarily due to its compatibility and support for GPUs, which significantly boost neural net performance. Keras [17] is a package that simplifies machine learning actions and works on top of theano; it will be the main tool used for constructing the networks and training. The training algorithm will likely be a form of stochastic gradient descent such as ADAM [18]

3.2 Challenges

Some expected key challenges are:

- The potentially large size of the input, and the large number of inputs which need to be trained for is also an issue. Fortunately, as the training data is simulated, the availability is only limited by computation time.
- The nets based approach will need to be adapted to work with an input size that could vary by orders of magnitude, they typically accept only a fixed length input.
- In the GANs, if G is trained too much without updating D, the "Helvetica scenario" arises, where G will map too many z to the same $\mathbf{y}$. This, and the rate of learning, require fine tuning.

3.3 Tasks

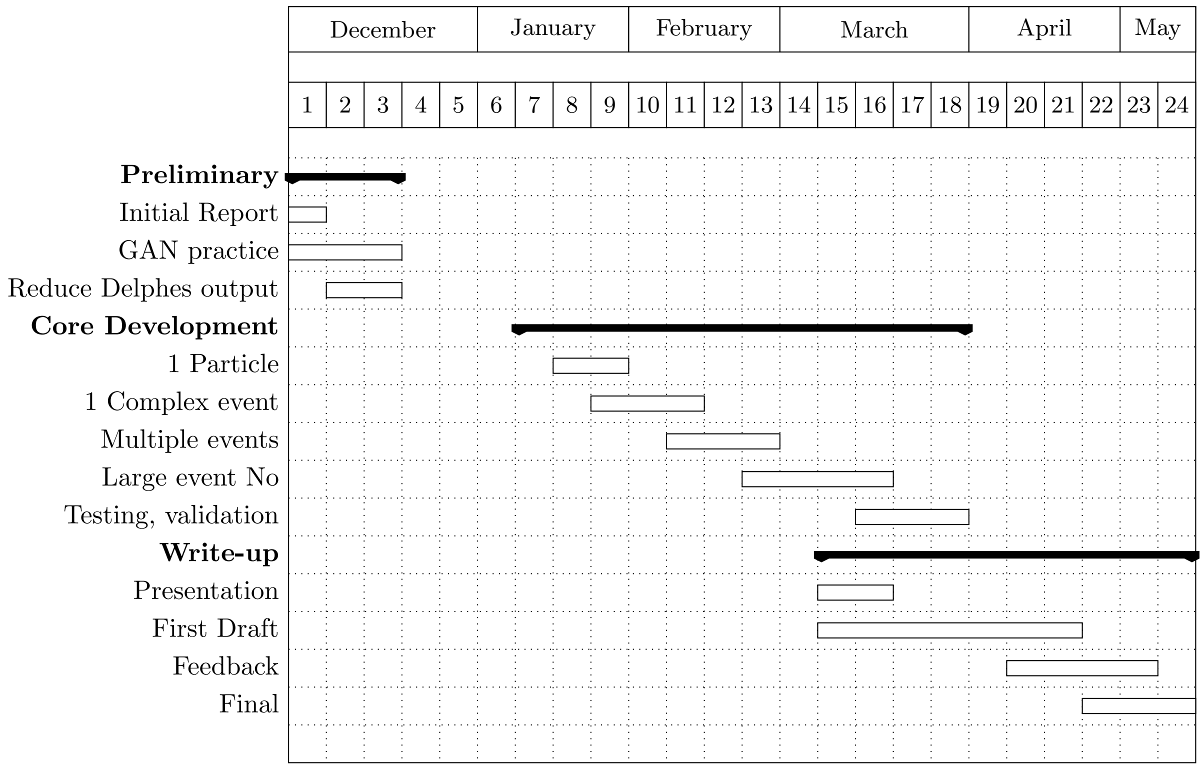

Figure 5: Gantt chart of expected work on this project.

References

[1] J. Allison et al. “Geant4 developments and applications”. In: *IEEE Transactions on Nuclear Science* 53.1 (2006), pp. 270–278. ISSN: 00189499. DOI: 10.1109/TNS.2006.869826. URL: http://ieeexplore.ieee.org/document/1610988/.

[2] J. De Favereau et al. “DELPHES 3: A modular framework for fast simulation of a generic collider experiment”. In: *Journal of High Energy Physics* 2014.2 (2014). ISSN: 11266708. DOI: 10.1007/JHEP02(2014)057. arXiv: arXiv:1307.6346v3. URL: http://arxiv.org/abs/1307.6346http://dx.doi.org/10.1007/JHEP02(2014)057.

[3] Ian Goodfellow et al. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems 27* (2014), pp. 2672–2680. ISSN: 10495258. arXiv: arXiv:1406.2661v1. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[4] Wolfgang Walkowiak. *Project: ATLAS Experiment.* URL: http://www.hep.physik.uni-siegen.de/atlas/atlas{_}en.html (visited on 11/26/2016).

[5] W.W. Armstrong et al. “ATLAS: technical proposal for a general-purpose p p experiment at the large hadron collider at CERN”. In: (1994), pp. 1–289. URL: http://hal.in2p3.fr/in2p3-00011404.

[6] 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: (2014). DOI: 10.1007/JHEP07(2014)079. arXiv: 1405.0301. URL: http://arxiv.org/abs/1405.0301http://dx.doi.org/10.1007/JHEP07(2014)079.

[7] Stefan Gieseke. “Monte Carlo Event Generators”. In: *Nuclear Physics B - Proceedings Supplements* 222-224 (2012), pp. 174–186. ISSN: 09205632. DOI: 10.1016/j.nuclphysbps.2012.03.018. arXiv: 1304.6677.

[8] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. “FastJet user manual”. In: *The European Physical Journal C* 72.3 (2012), p. 1896. ISSN: 1434-6044. DOI: 10.1140/epjc/s10052-012-1896-2. URL: http://www.springerlink.com/index/10.1140/epjc/s10052-012-1896-2.

[9] Zachary Marshall. “Simulation of Pile-up in the ATLAS Experiment”. In: 022024 (2013). ISSN: 17426596. DOI: 10.1088/1742-6596/513/2/022024.

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

[11] Dave Janz and James Requeima. “Deep Generative Models”. In: (2016).

[12] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. In: (2015). arXiv: 1511.06434. URL: http://arxiv.org/abs/1511.06434.

[13] Scott E. Reed et al. “Generative Adversarial Text to Image Synthesis”. In: *CoRR* abs/1605.05396 (2016). URL: http://arxiv.org/abs/1605.05396.

[14] Gilles Louppe. “Learning to generate with adversarial networks”. Aug. 3, 2016. URL: http://indico.cern.ch/event/526308/contributions/2262883/attachments/1319873/1979015/slides.pdf (visited on 11/27/2016).

[15] Mehdi Mirza and Simon Osindero. “Conditional Generative Adversarial Nets”. In: (2014). arXiv: 1411.1784. URL: http://arxiv.org/abs/1411.1784.

[16] Theano Development Team. “Theano: A Python framework for fast computation of mathematical expressions”. In: *arXiv e-prints* abs/1605.02688 (May 2016). URL: http://arxiv.org/abs/1605.02688.

[17] Franois Chollet. *keras.* https://github.com/fchollet/keras. 2015.

[18] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *CoRR* abs/1412.6980 (2014). URL: http://arxiv.org/abs/1412.6980.