UNIVERSITY OF THE
WITWATERSRAND,
JOHANNESBURG

UNIVERSITY OF THE WITWATERSRAND
MASTER'S DISSERTATION

Leveraging Machine Learning in the Search for New Bosons at the LHC and Other Resulting Applications

---

Author:
Finn STEVENSON

Supervisor:
Prof. Bruce MELLADO

A dissertation submitted in fulfillment of the requirements for the degree of Master of Science

in the

Wits Institute of Collider Particle Physics
School of Physics

September 11, 2023

Declaration of Authorship

I, Finn STEVENSON, declare that this thesis titled, "Leveraging Machine Learning in the Search for New Bosons at the LHC and Other Resulting Applications" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 09/11/2023

Date: 

"Our sole responsibility is to produce something smarter than we are; any problems beyond that are not ours to solve."

Ray Kurzweil

UNIVERSITY OF THE WITWATERSRAND

Abstract

Science
School of Physics

Master of Science

Leveraging Machine Learning in the Search for New Bosons at the LHC and Other Resulting Applications

by Finn STEVENSON

This dissertation focuses on the use of semi-supervised machine learning for data generation in high-energy physics, specifically to aid in the search for new bosons at the Large Hadron Collider. The overarching physics analysis for this work involves the development of a generative machine learning model to assist in the search for resonances in the $Z\gamma$ final state background data. A number of Variational Auto-encoder (VAE) derivatives are developed and trained to be able to generate a chosen Monte Carlo fast simulated dataset. These VAE derivatives are then evaluated using chosen metrics and plots to assess their performance in data generation. Overall, this work aims to demonstrate the utility of semi-supervised machine learning techniques in the search for new resonances in high-energy physics. Additionally, a resulting application of the use of machine learning in COVID-19 crisis management was also documented.

Acknowledgements

I would like to sincerely thank my supervisor, Prof. Bruce Mellado, for his continued guidance, his sharing of knowledge and the opportunities he has afforded me throughout my master's degree. Prof. Mellado has not only given me invaluable knowledge related to the fields of machine learning and particle physics but has taught me how to work under pressure in a way that elevates the quality of my work. I would also like to thank my fellow colleagues in the Wits physics department, especially Benjamin Lieberman, whose assistance and collaboration have been invaluable in completing this research. It is important to acknowledge and express gratitude to the various agencies from which I have been given funding to complete my research; the Africa-Canada Artificial Intelligence Data Modelling Consortium, the National Research Foundation and SA-CERN. I was able to travel to Geneva, Switzerland to work on my research at CERN as a result of the SA-CERN partnership. The experience of going to work at CERN as a member of the ATLAS experiment, one of the most advanced scientific collaborations in the world is something I will never forget. Lastly, thanks to my parents, Mark and Julie Stevenson for their continued support, sacrifices and encouragement throughout my research, and my grandfather David Stevenson for he has always been my scientific role model. During the COVID-19 pandemic, there were many pressures that made progression in my research difficult, my partner during that time, Tessa Chittenden, was the cornerstone of my sanity, happiness and ability to flourish in an otherwise arduous time.

Research Ouptus

Published Papers

1. Finn Stevenson et al. “Development of an Early Alert System for an Additional Wave of COVID-19 Cases Using a Recurrent Neural Network with Long Short-Term Memory”. In: International Journal of Environmental Research and Public Health 18.14 (2021). ISSN: 1660-4601. DOI: 10.3390/ijerph18147376. URL: https://www.mdpi.com/1660-4601/18/14/7376. [1]

2. Mahnaz Alavinejad et al. “Management of hospital beds and ventilators in the Gauteng province, South Africa, during the COVID-19 pandemic”. In: PLOS Global Public Health 2.11 (Nov. 2022), pp. 1–20. DOI: 10.1371/journal.pgph.0001113. URL: https://doi.org/10.1371/journal.pgph.0001113. [2]

3. Bruce Mellado et al. “Leveraging Artificial Intelligence and Big Data to Optimize COVID-19 Clinical Public Health and Vaccination Roll-Out Strategies in Africa”. In: International Journal of Environmental Research and Public Health 18.15 (2021). ISSN: 1660-4601. DOI: 10 . 3390 / ijerph18157890. URL: https://www.mdpi.com/1660-4601/18/15/7890. [3]

4. Benjamin Lieberman et al., “Big data- and artificial intelligence-based hot-spot analysis of COVID-19: Gauteng, South Africa, as a case study,” BMC Med. Inf. Decis. Making, vol. 23, no. 1, pp. 1–15, Dec. 2023, ISSN: 1472-6947. DOI: 10.1186/s12911-023-02098-3. [4]

Published or Soon to be Published Proceedings

1. Finn Stevenson, Edited by Prof Aletta Prinsloo. ’Evaluation and Optimisation of a Generative-Classification Hybrid Variational Autoencoder in the Search for Resonances at the LHC’ in The Proceedings of SAIP2022, the 66th Annual Conference of the South African Institute of Physics. en. SAIP2022, Dec. 2022, ISBN: 978-0-6397-4426-1[5]

2. Finn Stevenson, ’The Use of a Variational Autoencoder in the Search for Resonances at the LHC’ in the Porceedings of First Pan-African Astro-Particle and Collider Physics Workshop 2022 (to appear)

3. Finn Stevenson, ’Evaluation and Optimisation of a Generative-Classification Hybrid Variational Auto-encoder in the Search for Resonances at the LHC’ in the Proceedings of the International Nuclear Physics Conference (INPC) 2022 (to appear)

Contents

List of Figures

List of Tables

List of Abbreviations

MC	**M**onte **C**arlo
LHC	**L**arge **H**adron **C**ollider
ATLAS	**A** **T**oroidal **L**HC **A**pparatu**S**
CPU	**C**entral **P**rocessing **U**nit
AE	**A**uto **E**ncoder
VAE	**V**ariational **A**uto **E**ncoder
SM	**S**tandard **M**odel of particle physics
BSM	**B**eyond **S**tandard **M**odel
CERN	**C**onseil **E**uropéen pour la **R**echerche **N**ucléaire
ID	**I**nner **D**etector
MS	**M**uon **S**pectrometer
ECal	**E**lectro-magnetic **C**alorimeter
FCal	**F**orward **Cal**orimiter
HCal	**H**adronic **Cal**orimiter
CSC	**C**athode **S**trip **C**hambers
MDT	**M**onitored **D**rift **T**ubes
RPC	**R**esistive-**P**late **C**hambers
TGC	**T**hin-**G**ap **C**hambers
HLT	**H**igh **L**evel **T**rigger
NN	**N**ueral **N**etwork
SGD	**S**torcastic **G**radient **D**ecent
ADAM	**ADA**ptive **M**oment Estimation
RMSProp	**R**oot **M**ean **S**quare Propogation
MSE	**M**ean **S**quare **E**rror
BCE	**B**inary **C**ross **E**ntropy
KL	**K**ullback **L**eibler divergence
DNN	**D**irect **N**ueral **N**etwork
MLP	**M**ultiple **L**ayer **P**erceptron
CNN	**C**onvolutional **N**ueral **N**etwork
RNN	**R**ecurrent **N**ueral **N**etwork
LSTM	**L**ong **S**hort **T**erm **M**emory
GAN	**G**enerative **A**dverserial **N**etwork
VAE+D	**V**ariational **A**uto **E**ncoder + **D**iscriminator
NF	**N**ormalising **F**low
PDF	**P**robability **D**ensity **F**unction
VAE+NF	**V**ariational **A**uto **E**ncoder + **N**ormalising **F**low
IQR	**I**nter **Q**uartile **R**ange
GPU	**G**raphics **P**rocessing **U**nit
RAM	**R**andom **A**ccess **M**emory

COVID-19	**CO**rona **VI**rus **D**isease 20**19**
RIM	**R**isk **I**ndex **M**etric
ROC	**R**eceiver **O**perating **C**haracteristic

This dissertation is dedicated to my parents Mark and Julie Stevenson and my grandfather David Stevenson

Chapter 1

Introduction

Applications of machine learning have resulted in significant progress in numerous industries, including particle physics. Applications in particle physics include but are not limited to classification for jet tagging [6]–[8], regression for pileup mitigation [9]–[11], anomaly detection for new physics searches [12]–[14], and data generation for fast simulation [15]–[17]. This dissertation concentrates on the application of machine learning for data generation in particle physics. Generative machine learning models are often used to mimic data in order to allow for larger-scale research and analysis concerning that data. At the LHC, generative models are valuable resources that have been harnessed to increase the scale and efficiency of analyses and reduce CPU hours required for certain tasks. Monte Carlo (MC) data simulation takes up a large amount of the total ATLAS CPU hours. Pressure on CPU resources could be alleviated by using trained generative machine learning models as an alternative production mechanism to traditional MC production mechanisms. The luminosity of detectors at the LHC will increase over time therefore the need for MC simulations or alternative methodologies will only increase over time. In 2012 the ATLAS and CMS collaborations reported the discovery of the Higgs boson with a mass of 125 GeV [18], [19]. Although the discovery of the Higgs boson completed the Standard Model, SM, of particle physics, there is substantial evidence of phenomena that cannot be explained by the SM. These include Dark Matter, matter-anti-matter asymmetry, the origin of neutrino mass and a number of theoretical problems. Therefore, the search for new bosons is motivated by these experimental discrepancies with the SM. The main concentration of this work is the assessment of different variations of VAEs for data generation to aid in the search for new bosons.

1.1 Standard Model of Particle Physics

The Standard Model (SM) of particle physics is currently the most complete and widely accepted model to best describe the nature and properties of all elementary particles and their interactions [20]. The SM has been developed over many years of collaboration and iterative adjustment through the 20th century. One of the most significant developments in the field of particle physics in recent times has been the discovery of the Higgs boson. The SM guides much of the experimental particle physics research that is undertaken

globally. The SM is a great success in modern physics research and remarkably describes most experimental results. Figure 1.1 shows all the composite quarks, leptons, and bosons of the SM.

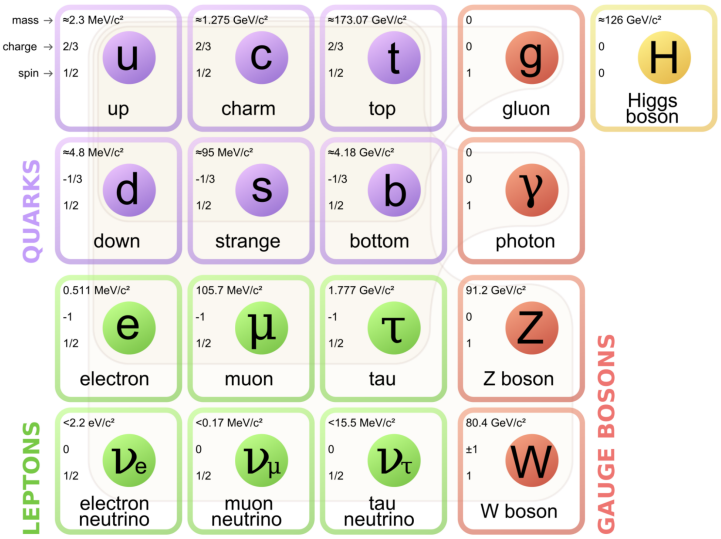


FIGURE 1.1: Diagram showing all of the fundamental building blocks of the Standard Model.

The SM breaks up all currently known particles into categories of either matter particles or force-carrying particles. Quarks (protons and neutrons building blocks) and leptons (including electrons) are matter particles and therefore make up all known ordinary matter in the Universe, excluding dark matter and dark energy. Whilst bosons are forces carrying particles that influence the matter particles, through known interactions. The SM particles are classified based on their spins, masses, and charges. These classifications are used to identify how we understand how the particles interact with each other. Fermions are divided into quarks and leptons and have a spin 1/2. Quarks are more granularly divided into the up quark (charge of 2/3) and the down quark (charge of -1/3), the charm quark (charge of 2/3), strange quark (charge of -1/3), top quark (charge of 2/3), and bottom quark (charge of -1/3). Quarks interact through the strong force and are distinguished using the "colours" classification. Leptons include the electron (charge of -1), muon (charge of -1), and tau (charge of -1). Each lepton has an associated neutrino, which is neutral with no charge: the electron neutrino, muon neutrino, and tau neutrino, respectively. All of these particles have antiparticles, which have opposite charges. Unlike quarks leptons interact through the weak force. Bosons, photons and gluons are force mediators. Vector bosons (spin of 1) mediate electromagnetic, strong, and weak interactions. The photon mediates the electromagnetic force, the gluon mediates the strong force, and the W and Z bosons mediate the weak force. The photon and gluon are massless, while the W and Z bosons are massive. Currently, the SM incorporates and explains three of the four known fundamental forces that govern

the Universe; electromagnetism, the strong force, and the weak force, however, gravity is not yet understood.

1.2 Beyond the Standard Model

Although the SM is profoundly useful in both experimental and theoretical physics analyses, there are still some limitations of the SM, which motivate beyond the SM searches. Beyond the Standard Model (BSM) searches attempt to discover phenomena that cannot be explained by the current Standard Model of particle physics. A number of the phenomenon that the Standard Model cannot explain are the existence of dark matter, the observed matter-antimatter asymmetry in the Universe, and the inability to quantify gravity as a force in the Universe. BSM searches aim to find evidence for new particles, interactions, and phenomena that could help expand our understanding of the fundamental makeup of the Universe. BSM searches are typically carried out using high-energy particle accelerators, such as the Large Hadron Collider at CERN. These accelerators produce high-energy collisions that can produce new particles that are not predicted by the Standard Model. By studying the products of these collisions evidence of new phenomena can be searched for.

1.3 CERN

European Council for Nuclear Research (Conseil Européen pour la Recherche Nucléaire, CERN) is a scientific laboratory located outside of Geneva, Switzerland. At CERN, the fundamental properties of the basic constituents of matter in our Universe are explored in an attempt to deepen our understanding of our physical world. CERN started as a joint scientific collaboration between countries and now has 23 official member states. The main instrumentation at CERN are particle accelerators and detectors used to collide particles with other particles in motion or stationary targets. The detectors are extremely complex pieces of technology that collect data on the various parameters of the sub-atomic particles observed during a collision.

1.3.1 The Large Hadron Collider

The Large Hadron Collider (LHC), located beneath Geneva, is the world's most powerful and biggest particle accelerator, with a circumference of 27 kilometers. The collider consists of a ring of superconducting magnets that aid to correct the track of the accelerated particle and a number of accelerating systems to boost the energy of the particles as required. Along the circumference, there are a number of different experiments that take different approaches in terms of their apparatus design in order to maximize the possibility of uncovering new physics or precise measurement by not relying on solely one methodology for measurement or detection.

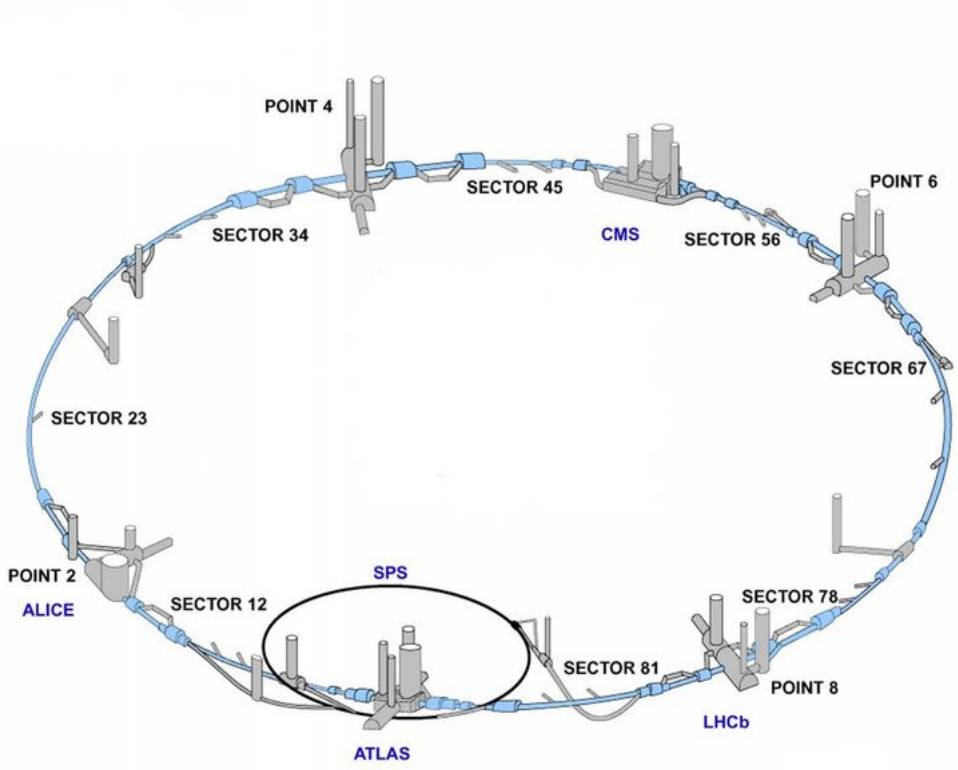


FIGURE 1.2: Diagram of the Large Hadron Collider and existing experiments.

1.3.2 The ATLAS Experiment

The ATLAS experiment is one of the particle physics experiments located at the Large Hadron Collider (LHC) at CERN. The experiment was designed for the purpose of both precise measurement and Beyond Standard Model (BSM) exploration, the experiment aims to exploit the full discovery potential of the LHC. The detector was built to identify almost all of the final state particles produced during high energy pp collisions. The ATLAS detector consists of a wide range of sensory apparatus to measure all valuable parameters during a collision. A diagram of the ALTAS detector sub-systems can be seen in figure 1.3[21]. The ATLAS experiment apparatus is the largest of all the experiments at CERN, and by volume, the largest particle detector ever built. The detector is a composition of multiple different sub-detectors that are arranged concentrically outwards from the collision point. The main components of the ATLAS detector are the Inner Detector (ID), the calorimeters, the Muon Spectrometer (MS), and the magnet system (a central solenoid, a barrel toroid and two end-cap toroids). The design of all the sub-detector systems and their cumulative measurement ability allow for the detection of all established SM particles except neutrinos. Neutrinos have a small interaction with hadronic matter, however, their presence can be determined by looking at the momentum imbalance between all of the observed particles.

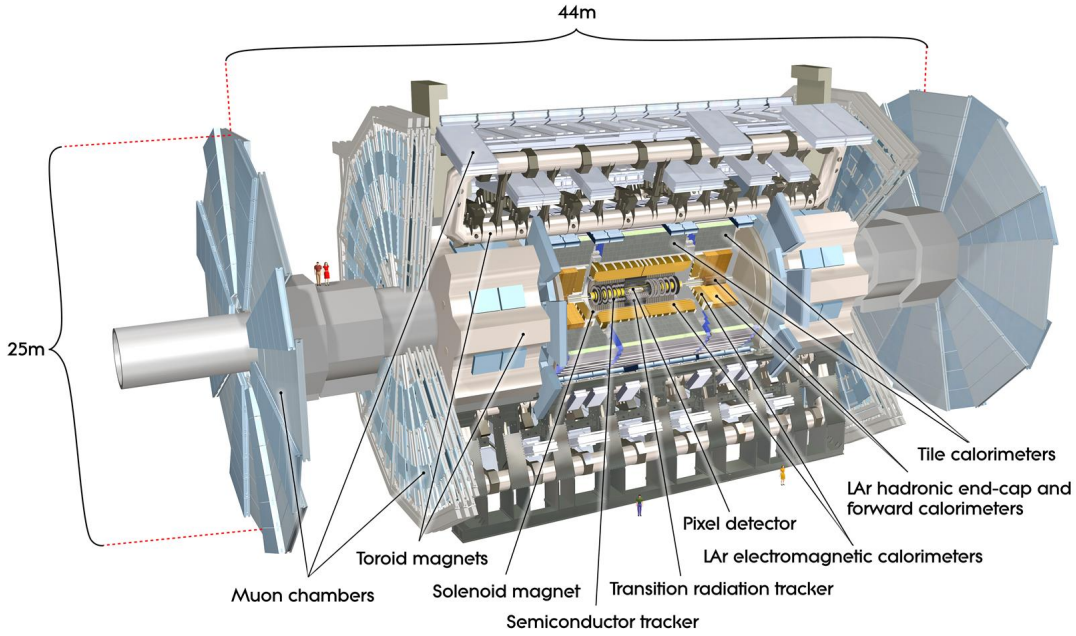


FIGURE 1.3: Diagram of the Atlas Detector.

This plethora of sensory apparatus results in copious amounts of data accumulated during data taking. This data can then be used in analyses for precision measurement or beyond SM searches. This research focuses on analyzing the anomalies between the experimental data produced at the ATLAS experiment and the Monte Carlo simulations of particle physics based on the standard model.

The Inner Detector

The ID, as suggested by the name, is the most inner detector closest to the beam path, and the first part of the detector apparatus that the particle emerging from the collision will interact with. It is composed of several different detector technologies, each of which provides important information about the particles produced in the collisions. The ID functions to track the charged particles by mapping out their interactions with the different layers of the ID. The pixel detector is made up of layers of silicon pixel sensors that provide precise position information for charged particles. The semiconductor tracker consists of layers of silicon strip detectors that provide additional position and momentum information. The transition radiation tracker uses detectors that detect the transition radiation produced when high-energy particles pass through a radiator, providing information on the velocity of charged particles. The track that the charged particles follow outwards from the collision point is reconstructed by using the different interactions of the particles with the different ID layers. The track of the charged particle is helical because of the axial magnetic force that exists in the detector. The ID is able to

detect particles through the generation of ionising radiation when the particles interact with the ID, due to this, only charged particles can be measured using this sub-detector. Figure 1.4 shows a representative diagram of the ID.

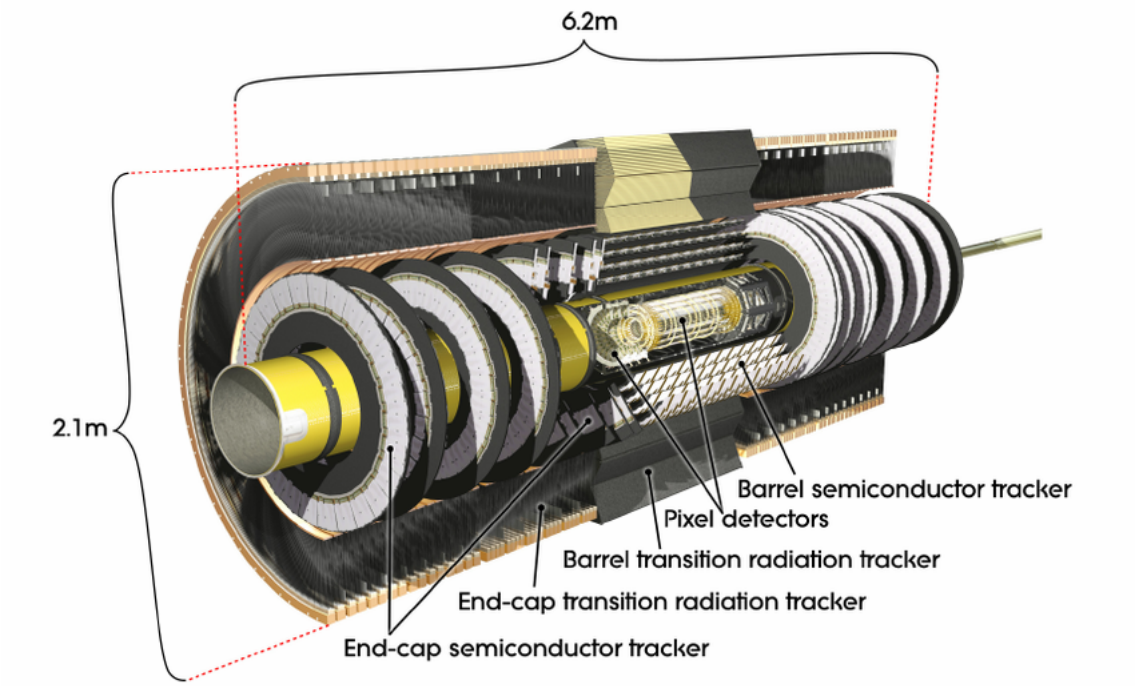


FIGURE 1.4: Diagram of the Inner Detector of ATLAS.

The Calorimeters

The calorimeters are the second array of detectors in the ATLAS apparatus moving radially outwards from the collision point. The calorimeters are used for particle identification and energy measurement of electrons, photons, jets and other hadrons. The overall calorimeter component is made up of three sub-calorimeters, the Electromagnetic calorimeter (ECal), the Hadronic calorimeter (HCal) and the Forward calorimeter (FCal). The three calorimeters work in the same way to measure the energies of incoming particles. The calorimeters contain layers of high-density metal which absorb energy from incoming particles. In between each layer of high-density metal, there are active materials (liquid argon) which are used to record the shape of the particle shower and the total amount of deposited energy. The ECal uses the electromagnetic force as a reference force to measure the energies of the lighter particles. The HCal measures particles which interact via the strong force and pas through the Ecal. Hadronic showers take up much larger volumes because of the larger distance between nuclear interactions. The HCal is designed to make sure that no particle or shower reach the muon spectrometer. Lastly, the FCal is located close to the beam pipe, used to take both electromagnetic and hadronic measurements of particles ejected with η up to 4.9.

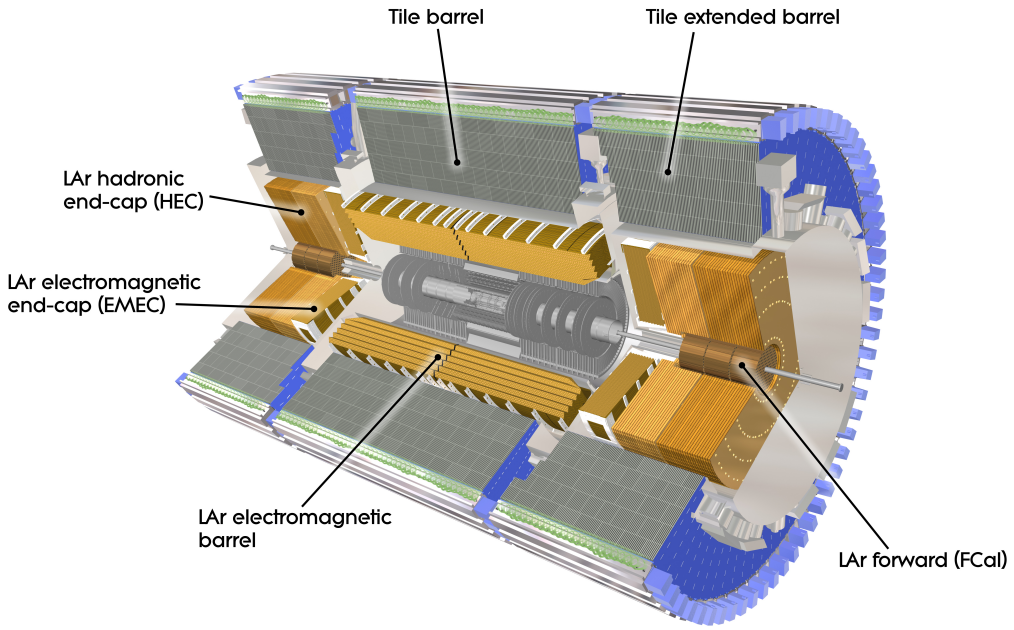


FIGURE 1.5: Flow diagram of ATLAS Calorimeters.

The Muon Spectrometer

The Muon Spectrometer (MS) is the outermost sub-system of the ATLAS detection apparatus. Unlike most of the other stable particles, muons are able to traverse the calorimeters because of their large mass and the fact that they do not emit bremsstrahlung radiation. The MS is therefore used to measure the position of the muons that are emitted from the collision. The tracks of the emitted muons are bent by an array of magnets in which the MS resides. There are four main sub-components of the MS other than the magnets, the Cathode Strip Chambers (CSCs), the Monitored Drift Tubes (MDTs), the Resistive-Plate Chambers (RPCs) and the Thin-gap Chambers (TGCs). The MDTs and the CSCs are used for high-precision position tracking, whilst the RPCs and the TGCs are used for triggering data capture.

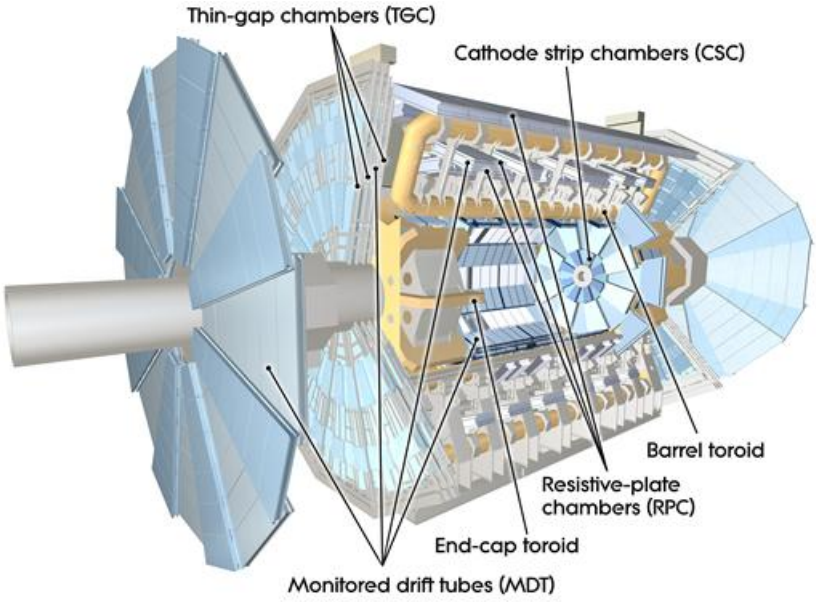


FIGURE 1.6: Diagram of the ATLAS Muon Spectrometer.

The ATLAS Trigger System

The ATLAS trigger system is the sub-system that is responsible for isolating and recording valuable events during a collision. The LHC produces a large number of collisions every second, but only a small fraction of these collisions are interesting from a physics perspective. The total number of recorded events is only a small fraction of the overall collisions that occur. The Trigger System is used to identify these interesting events and send them to the data acquisition system for further analysis. The trigger system has been designed carefully to make efficient and precise decisions regarding the importance of a particular collision. This is a crucial process in the working of the detector as it is physically impossible to retain and store the data from all of the collisions due to long-term storage resource limitations. The Trigger System of ATLAS consists of several levels of triggers, each of which is designed to select events based on different criteria. The level one trigger is a hardware-based trigger taking input from both the muon triggers and information from the calorimeters. The level one trigger functions by filtering events for high p_T electrons, muons, photons and jets. Additionally, events with high missing transverse energy will also pass the level one trigger filter. The High-Level Trigger (HLT) is a software-based trigger and has inputs from all the detector trackers and calorimeters. The HLT uses advanced reconstruction algorithms to decide whether the events that pass the level one trigger should be transferred to permanent storage. Figure 1.7 shows a flow diagram of the ATLAS trigger system, including both the level on the trigger and the HLT. The Trigger System plays a crucial role in the operation of the ATLAS experiment, allowing researchers to efficiently select interesting events for further study.

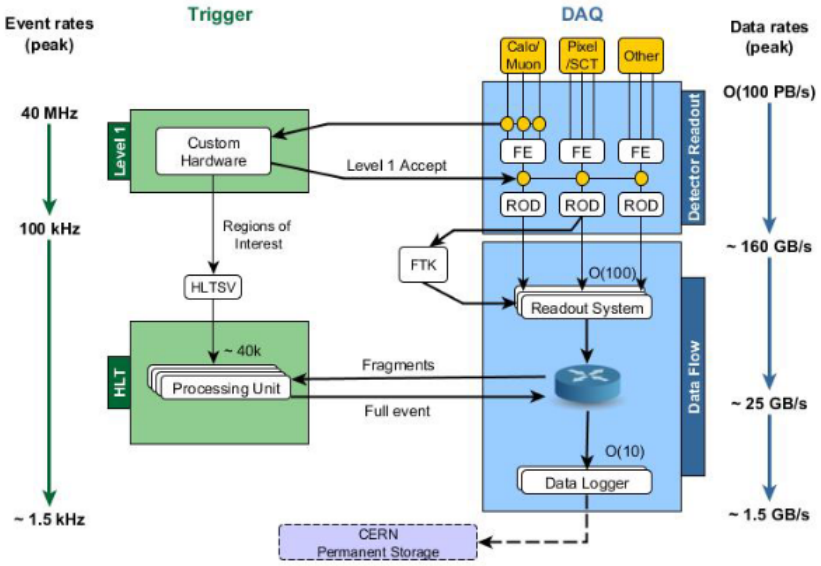


FIGURE 1.7: Flow diagram of the ATLAS data-taking trigger systems.

1.4 Semi-supervised Machine Learning in the Search for New Resonances

ML methods that are fully supervised are often used as binary classifiers in high-energy physics. These models are trained on fully labelled data where each event is labelled either 0 (background event) or 1 (signal event). For a given event, $\vec{x_i}$, the model produces an output response, $\hat{y_i} \in (0,1)$. During training, the loss function is minimized, meaning the difference between the targets and responses is made as small as possible. In the case of semi-supervised learning, models are trained on partially labelled datasets. For example, using a deep neural network as a semi-supervised classifier, the model is only trained on events labelled as 0 (background events). Any events not recognized as background by the model will be labelled as 1 (signal events). Searching for new resonances is crucial in understanding physics beyond the standard model. If there are new resonances hidden in the data, the signals may be subtle due to the absence of inclusive signals. These resonances may be missed in certain unexplored or poorly searched areas of phase space. Semi-supervised machine learning can help address this issue by detecting differences in the phase space where new physics may exist, compared to the baseline model used. Weak supervision does not require any prior knowledge about the topological features of the signal. By training a weakly supervised machine learning model on sideband background data, the model may be able to detect differences in topology between the potential signal region and the sidebands. Although it may be more difficult to train semi-supervised models to classify between known and unknown data, it can provide an advantage in detecting new resonances if trained correctly. Machine Learning based semi-supervised classifiers require a large

amount of background data to be trained on. This means that either Monte Carlo computation is required to generate sufficient data, or as suggested in this work, other machine learning based data generation models can be used for background data generation. There are many different types of semi-supervised machine learning models used in resonance searches, including data generation models and signal classification models.

1.4.1 Machine Learning Data Generation In Particle Physics

Machine learning based data generation has become an important tool in high-energy physics. One of the main avenues of use of machine learning in recent years both in and outside of particle physics has been for data generation tasks. Generative machine learning models are used to mimic chosen data in order to allow for larger-scale research and analysis concerning that data. The LHC produces a vast amount of data, and traditional methods for simulating and generating this data can be computationally expensive and time-consuming. At the LHC, semi-supervised generative machine learning models are a crucial resource that can be fully harnessed to increase the scale and efficiency of analyses and reduce CPU hours required for certain tasks. Machine learning-based data generators can be used in conjunction with traditional Monte Carlo (MC) generation to scale datasets. This synthetic data can then be used to train and validate other machine-learning models, such as those used for particle detection and classification. Advanced deep generative neural network models such as Generative Adversarial Networks (GAN) and Variational Auto-Encoders (VAE) are examples of semi-supervised machine learning methodologies commonly used in industry to scale data. Some promising attempts to generate data for high energy physics with GANs have already been completed, for example for top pair production in the paper 'How to GAN LHC Events' [22]. Once trained these models are able to generate events with sufficient accuracy at scale with minimal required computational resources. Monte Carlo (MC) data simulation takes up a large amount of the total CPU hours at the LHC, see Figure 1.8. This large pressure on CPU resources could be alleviated by using trained generative machine learning models as an alternative production mechanism to traditional MC production mechanisms. By using machine learning-based data generation, the rate of research progression can be sped up drastically. A number of different machine learning and deep learning based models could be assessed for this data generation task. VAEs, GANs, and Transformer models are three examples of deep learning models that could be evaluated for this task. VAEs use autoencoders to learn a probabilistic latent representation of data, whilst GANs use adversarial training to generate data, most commonly used for high-quality image generation. Transformer models utilize self-attention mechanisms for generation tasks, usually for natural language tasks. VAEs are effective with continuous data, while GANs excel in visual fidelity. Transformers are best in language generation but are limited to sequential data tasks. Understanding their strengths and weaknesses aids in selecting the appropriate model for specific applications. The VAE model

will be further evaluated in this work as the architecture lends itself to excel with the continuous data used in this work.

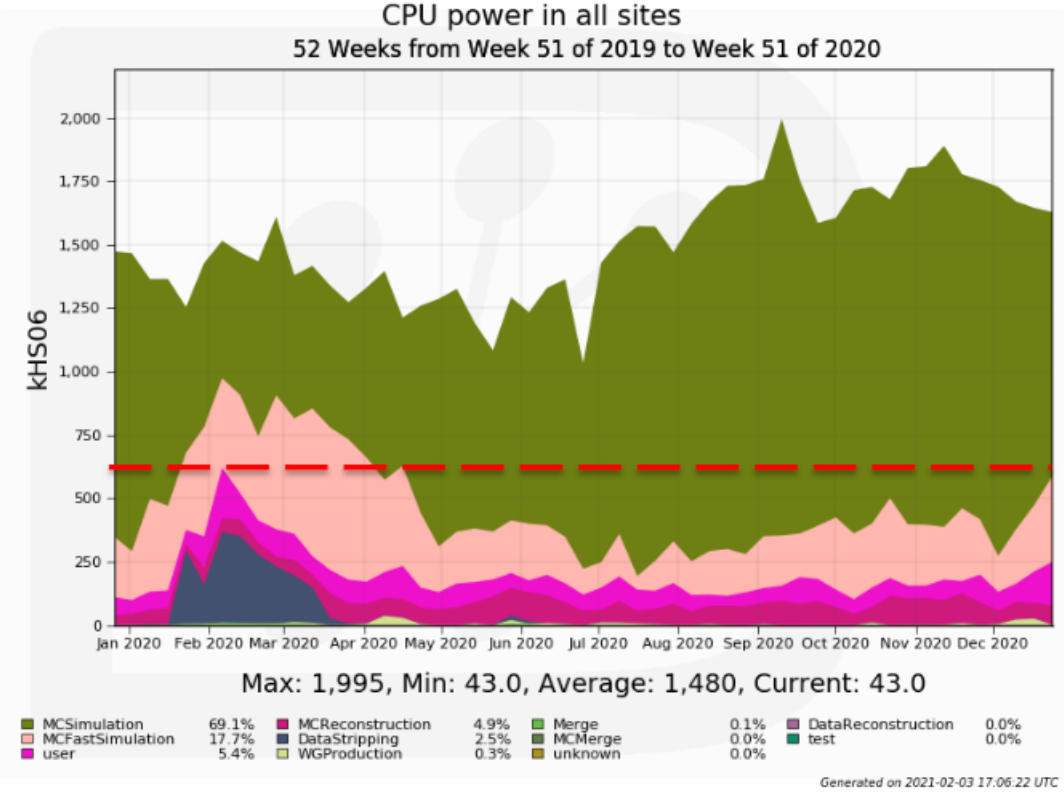


FIGURE 1.8: Graph Showing Usage of CPU Resources for Different Tasks at the LHC in 2022 [23].

The luminosity of detectors at the LHC is continuing to increase over time, this means that the need for MC simulations or alternative-based methodologies will only be increasing over time.

1.4.2 Machine Learning Signal Classification In Particle Physics

Classification and anomaly detection has long been a field heavily influenced by the uptake of machine learning algorithms [24]. The LHC produces a vast amount of data, and accurately identifying and classifying the various signals within this data is a crucial task. In the field of particle physics classification techniques are often used to attempt to identify abstract signals against known background data. This is a crucial process in BSM searches, and in this case, searches for new bosons. Traditional methods for signal classification can be time-consuming and error-prone, but machine learning algorithms, especially semi-supervised, have the potential to quickly and accurately classify the various signals present in the LHC data. By using machine learning for signal classification, the time taken for analysis can be reduced and more accurate insights can be extracted from the data. An example of a machine learning framework for classification used in high energy physics is the CWoLa paradigm, in which common statistical mixtures of classes in collider physics are classified without labels [25].

1.5 Scope and Structure

The scope of this research is concentrated on the evaluation of VAE model derivatives for specifically the data generation task. Therefore, the main scope includes:

- Literature review of Machine Learning
- Selection of the VAE model derivatives to be assessed
- Selection of evaluation metrics and plots for the data generation task
- The detail of the greater $Z\gamma$ physics analysis
- Pre-processing of the $Z\gamma$ background data, including kinematic variable selection
- Training of the VAE derivative models including the hyper-parameter optimisation of each of the models specifically for the data generation task
- Evaluation of the trained models for the data generation task using the selected evaluation metrics and plots
- Showcase of resulting application of using machine learning for crisis mitigation in South Africa during the COVID-19 pandemic
- Discussion and conclusions related to the use of the trained models for data generation
- Conclusions on the use of the semi-supervised VAE model derivatives for use in the greater $Z\gamma$ study

Chapter 2

Physics Analysis

In order to elaborate on some of the currently unexplained phenomena observed in the Run 1 Large Hadron Collider (LHC) data, a 2HDM+S model was used in Ref. [26], [27]. Using this model, the heavy scalar, S, decays into $Z\gamma$, here we consider the rare decay carefully. The full lagraingian including all relevant terms can be found in Ref. [28]. The leading order Feynman diagrams for this process can be seen in Figure 2.1.

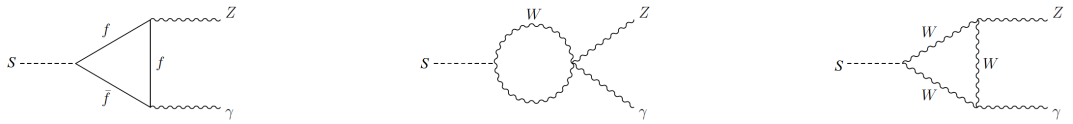


FIGURE 2.1: The leading order Feynman diagrams of massive boson S to $Z\gamma$ final state.

The model exhibits multi-lepton anomalies verified in Refs. [29]–[32] as well as a possible candidate for the singlet S, reported in Ref. [33]. If the model is complemented by a Dark Matter candidate [34], multiple anomalies found in astrophysics can be expressed. Additionally, the model can be further extended to account for various other anomalies, which include the $g-2$ muon experiment reported by Fermilab [35]–[37]. A full review of anomalies can be found in Ref. [38]. The aforementioned model and applications motivate further searches for heavy scalar resonances. In the greater physics analysis related to this dissertation, the search of $S \to Z\gamma$ with $Z \to \ell\ell$ and $\ell = e, \mu$ is investigated. The investigation uses a semi-supervised deep neural network to classify topological features, as suggested in Ref. [39]. The machine learning technique of semi-supervised learning (see more detail in the machine learning section) is used to reduce training biases by training the model on a partially labelled dataset. However one must be cautious when using semi-supervised classifiers due to the fact that background events can be falsely classified as signals due to the possibility of over-training. The extent to which this miss-classification occurs should be quantified. These falsely labelled signals are caused by over-fitting in training semi-supervised models that incorporate side-bands and the signal regions. The methodology of the implementation of the semi-supervised technique, together with the detailed methodology for the investigation of the over-training within semi-supervised machine learning models, is described in detail in Ref. [40]. In the semi-supervised over-training study, a single run can be defined as

the process of calculating the extent of falsely classified signals by the model when trained on a batch of pure background training data. Therefore to complete the study, a frequentist approach must be taken by logging the output of many runs and performing deductions on the overall result. In a high-energy physics analysis, a frequentist study is often used to calculate the statistical significance of an observed phenomenon, such as the discovery of a new particle. This involves comparing the observed frequency of the phenomenon in a sample of data to the expected frequency of the phenomenon under the null hypothesis. The null hypothesis is a statistical model that assumes that the phenomenon is not present in the population. By comparing the observed and expected frequencies, it is possible to calculate a p-value, which is a measure of the statistical significance of the observation. A low p-value indicates that the observation is unlikely to have occurred by chance, and is therefore considered statistically significant. To complete a frequentist study, a dataset with enough events and statistics is required to extract many hundreds of statistically independent batches of data for training the model. A very large dataset is also necessary to overcome the "look elsewhere effect" when analysing local and global resonances [41]. Which can be defined in searches for resonances within a given mass range, as the probability of observing a significant local excess of events, elsewhere within the range. This motivates the use of machine learning-based generative models in combination with traditional Monte Carlo production mechanisms to increase statistics for the successful evaluation of over-training. In the case of this research, VAEs are used as a generative model to expand statistics. A flow diagram of the physics motivation of this work as well as the applications of the trained VAE model in the overarching physics analysis can be seen in Figure 2.2.

2.1 $Z\gamma$ final State Background Data Monte Carlo (MC) Production

The Monte Carlo method is a computational technique that uses random sampling to solve problems. In the context of high-energy physics, a Monte Carlo method can be used to simulate events according to the Standard Model. This involves generating random numbers that represent the energies, momenta, and other properties of the particles involved in the event. These random numbers are then used to calculate the probability of the event occurring according to the rules of the Standard Model. This process can be repeated many times to produce a large sample of events that can be analyzed to study the properties of the particles involved. Monte Carlo methods are commonly used in high-energy physics to simulate the effects of detector resolutions and other experimental effects on the data. The MC events corresponding to the $Z\gamma$ sample in this analysis have been generated using `Madgraph5` [42] with the `NNPDF3.0` parton distribution functions [43]. In the electron (muon) channel, Z + makes up 0.90 (0.91) of the total background events [44]. In this work, since fast simulation data is utilised, the Z + jets background is approximated using data driven methods as it is difficult to accurately simulate the

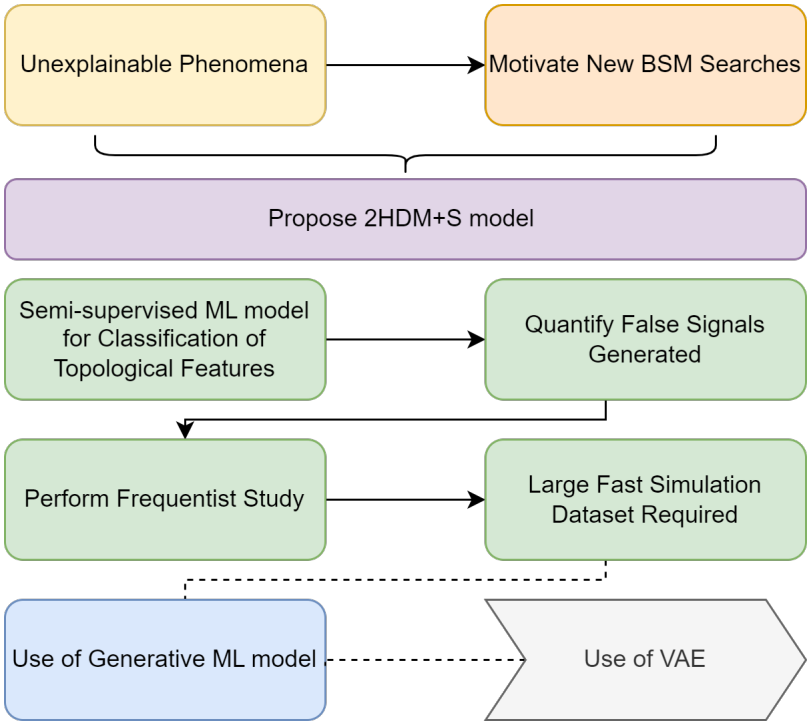


FIGURE 2.2: Research Necessity Flow Diagram.

jets as they can be miss-classified as photons. Here we have utilized the Standard Model (SM) of particle physics for which the UFO model files required by the Madgraph are obtained from `FeynRules` [45]. The parton level generation is followed by the parton showering and hadronization by `Pythia` [46] and then the detector level simulation is performed using `Delphes`(v3) [47]. The jets at this level has been constructed using Fastjet [48] which involves the anti-K_T jet algorithm with $P_T > 20$ GeV and radius $R = 0.5$. While generating the sample we decayed the Z boson to leptons. We also have applied some baseline cuts on the leptons and photons at the Madgraph level to enhance the statistics.

2.2 $Z\gamma$ Background Data Kinematic Variable Selection

The $H \to Z\gamma$ with $Z \to \ell\ell$ and $\ell = e, \mu$ Monte Carlo data produced has been further processed by selecting kinematic variables for the study. The analysis focuses around the centre of mass of 150GeV (132GeV$< m_{\ell\ell\gamma} <$ 168GeV). The kinematic features used in the study are $Z\gamma$ invariant mass, $m_{\ell\ell\gamma}$; the transverse momentum, azimuthal angle, pseudo-rapidity and energy of the leading lepton, sub-leading lepton and photon respectively, $Pt_{\ell_1|\ell_2|\gamma}$, $\Phi_{\ell_1|\ell_2|\gamma}$, $\eta_{\ell_1|\ell_2|\gamma}$ and $E_{\ell_1|\ell_2|\gamma}$; missing transverse energy E_T^{miss} and it's azimuthal angle, $\Phi_{E_T^{miss}}$; the number of jets, N_j, the number of central jets, N_{cj}; and $\Delta R_{\ell\ell}$ ($\Delta R \equiv \sqrt{(\Delta\eta_{ll})^2 + (\Delta\phi_{ll})^2}$), $Pt_{\ell\ell}/m_{\ell\ell\gamma}$, $\Delta\Phi_{\ell\ell}$ and $\Delta\Phi(E_T^{miss}, Z\gamma)$. The following table

shows the kinematic variables selected accompanied by a short description of each variable.

Kinematic Variable	**Variable Description**
$m_{\ell\ell\gamma}$	Invariant mass of di-lepton with a photon.
Pt_{l1}	Transverse momentum of leading lepton.
η_{l1}	Pseudo-rapidity of leading lepton.
Φ_{l1}	Azimuthal angle of leading lepton.
E_{l1}	Energy of the leading lepton.
Pt_{l2}	Transverse momentum of the sub-leading lepton.
η_{l2}	Pseudo-rapidity of the sub-leading lepton.
Φ_{l2}	Azimuthal angle of the sub-leading lepton.
E_{l2}	Energy of the sub-leading lepton.
Pt_{γ}	Transverse momentum of the photon.
η_{γ}	Pseudo-rapidity of the photon.
Φ_{γ}	Azimuthal angle of the photon.
E_{γ}	Energy of the the photon.
$\Delta R_{\ell\ell}$	$\sqrt{(\Delta\eta_{ll})^2 + (\Delta\phi_{ll})^2}$.
E_T^{miss}	Missing transverse energy.
$\Phi_{E_T^{miss}}$	Azimuthal angle of missing transverse energy.
Nj	Number of jets
Ncj	Number of central jets

TABLE 2.1: Kinematic Feature Selection.

Figure 2.3 shows the distribution graphs of the chosen kinematic variables associated with the leading lepton.

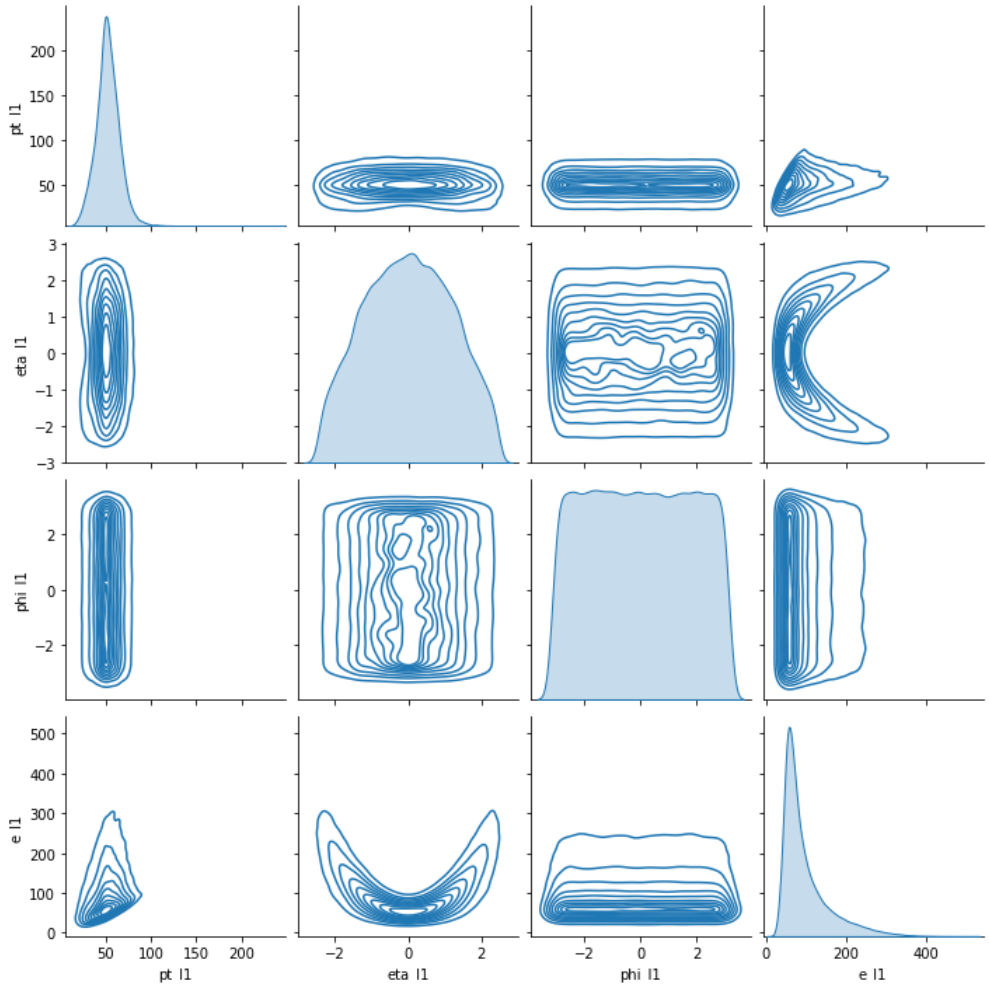


FIGURE 2.3: Distribution graphs of the selected kinematic features of the 1st lepton.

Figure 2.4 shows the distribution graphs of the chosen kinematic variables associated with the sub-leading lepton.

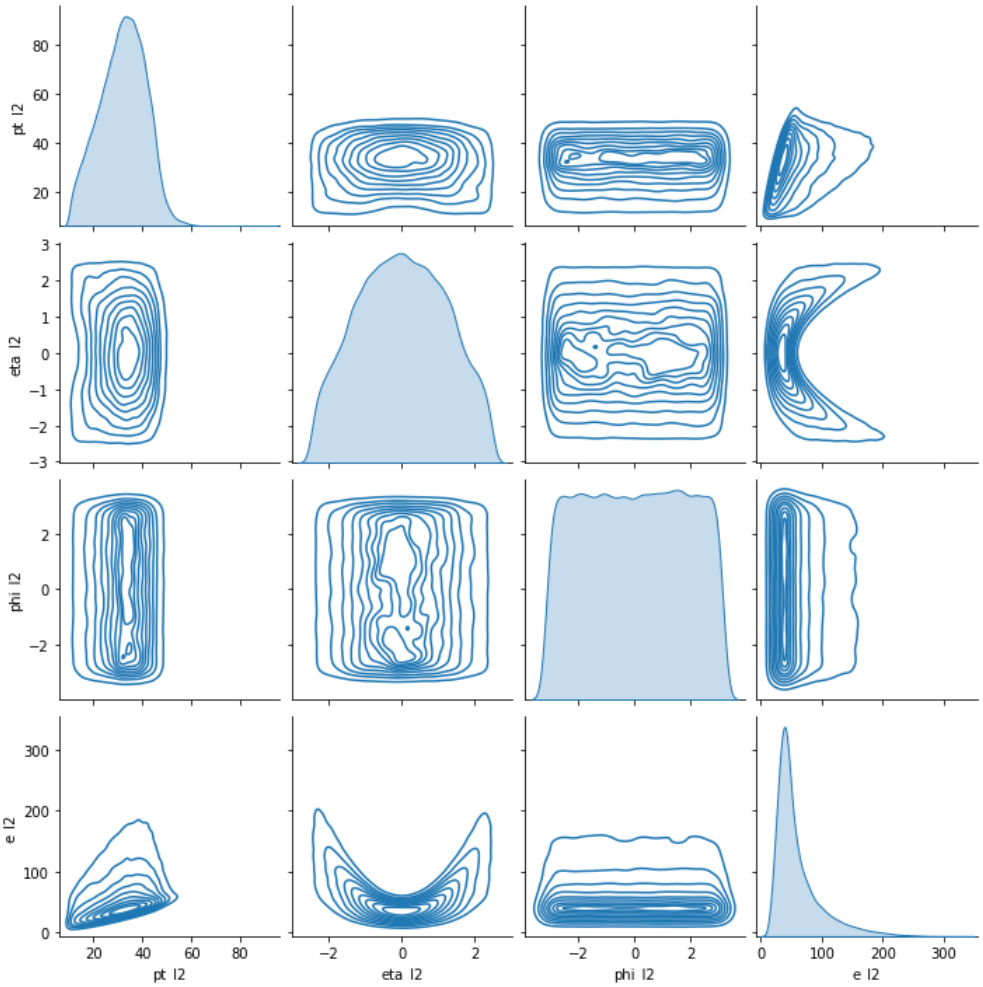


FIGURE 2.4: Distribution graphs of the selected kinematic features of the 2nd lepton.

Figure 2.5 shows the distribution graphs of the chosen kinematic variables associated with the photon.

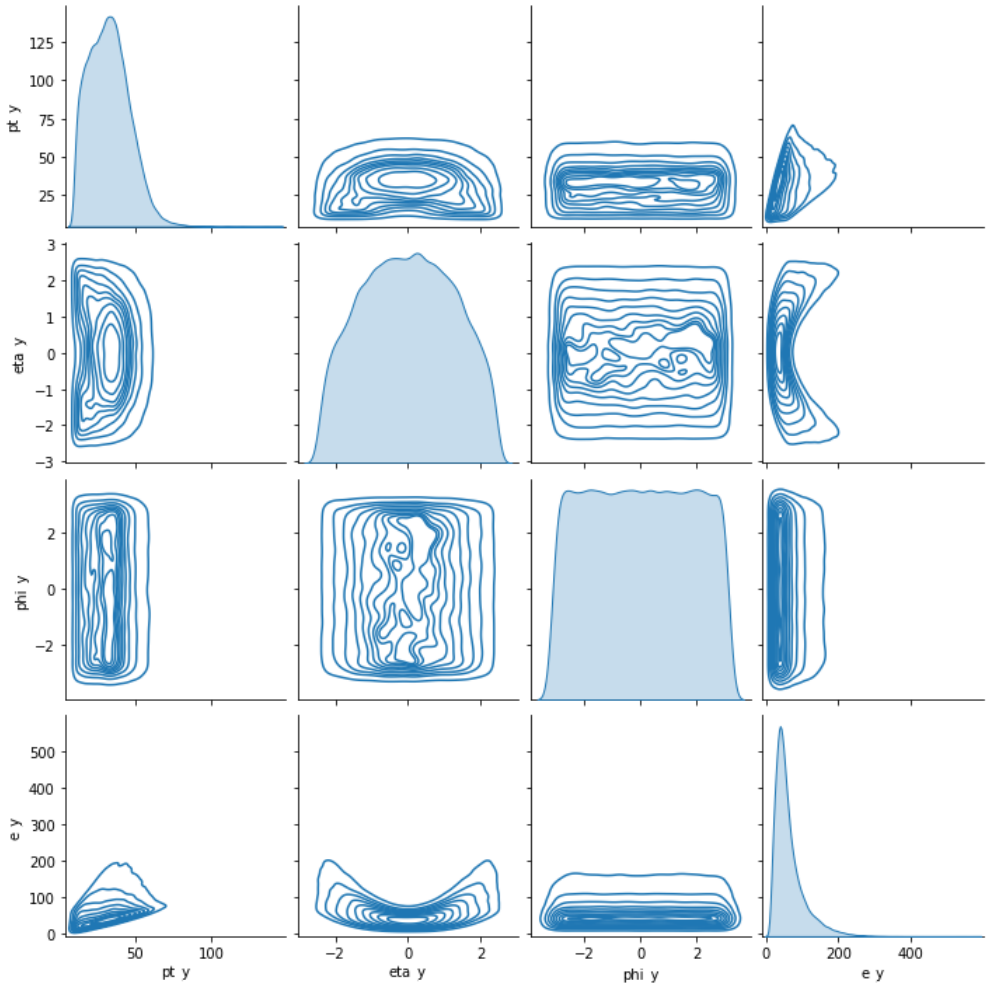


FIGURE 2.5: Distribution graphs of the selected kinematic features of the photon.

Figure 2.6 shows the distribution graphs of the following chosen kinematic variables, $m_{\ell\ell\gamma}$, $\Delta R_{\ell\ell}$, E_T^{miss} and $\Phi_{E_T^{miss}}$.

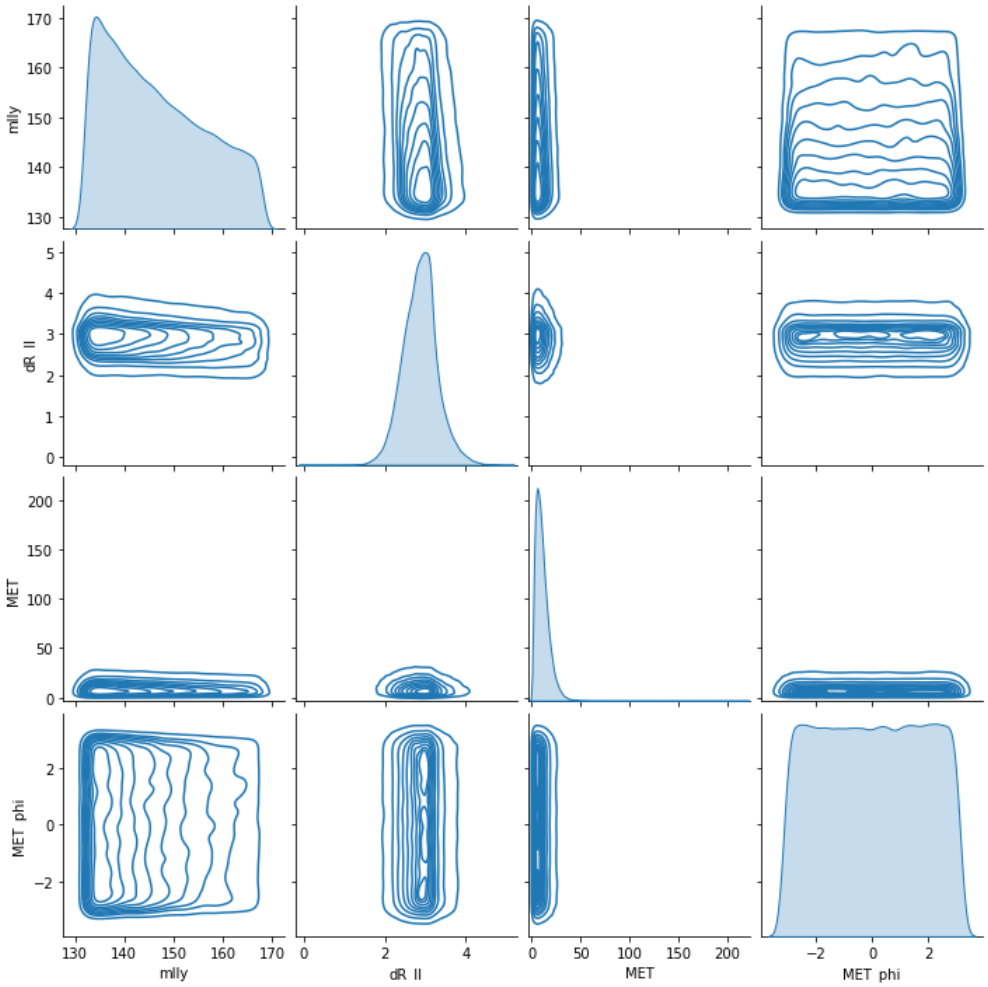


FIGURE 2.6: Distribution graphs of the selected kinematic features.

Figure 2.7 shows the distribution graphs of the chosen kinematic variables associated with the number of jets.

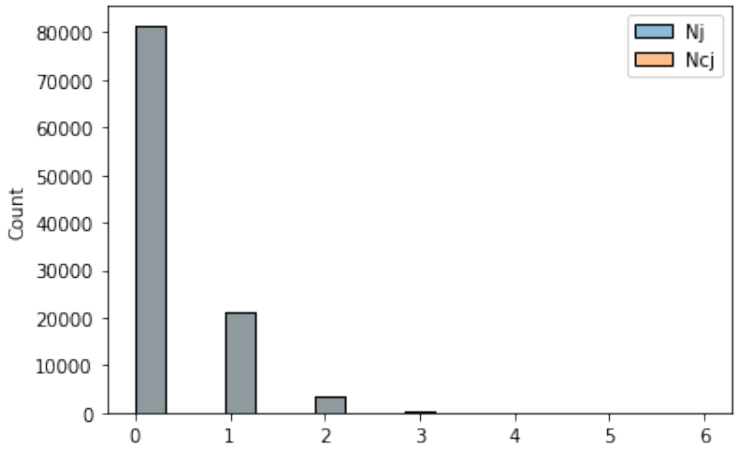


FIGURE 2.7: Distribution graphs of the selected kinematic features related to the number of jets.

From the above distribution graphs, it can be seen that there are many different types of distributions that the selected variables follow. This selection of types of distributions will serve well to test the generative capability of the chosen model.

2.3 $Z\gamma$ Final State Resonance Searches

Resonance searches are a type of search performed in high-energy physics experiments to look for evidence of new particles or phenomena. The specific search related to this work involves looking for resonances in MC fast simulation data in which a Z boson and a photon are produced in the final state. During a collision at the LHC, new particles that are not predicted by the Standard Model of particle physics could be created and the evidence of their presence can be shown by an apparent resonance. By using machine learning to study the products of these collisions, searches for evidence of new particles and phenomena can be completed. Resonance searches have led to many discoveries in recent years. For example, the discovery of the Higgs boson at the LHC in 2012 provided evidence for the existence of a new type of particle that decays into a Z boson and a photon. Further $Z/gamma$ final state resonance searches have can be focused on looking for evidence of new particles that could help explain the nature of dark matter, the matter-antimatter asymmetry of the universe, and other unexplained phenomena.

Chapter 3

Machine Learning

Machine learning was conceived in an attempt to give machines the ability to learn like the human brain. The origin of ML as we know it today comes from the work of psychologist Frank Rosenblatt who led a research group that initially built a machine to serve as a classifier to recognise letters of the alphabet, known as the perception [49]. The perceptron became the foundation of Artificial Neural Networks (ANNs) as they exist today. The original perceptron machine worked using both analogue and digital signals and a threshold to convert the analogue signals to discrete digital ones. The perceptron was the first example of a machine that was able to learn similarly to that of human or animal learning. Although the perceptron machine was a foundational step forward in control engineering, the original machine did not prove as successful in identifying many different classes of patterns. Single-layer perceptrons, like the initial machine, are only capable of learning linearly separable patterns. However, the addition of more layers and nodes was the next step in the progression of the perception creating the multi-layer perceptron, able to classify more complex patterns.

3.1 The Perception

In modern applications, the perceptron can be used as a threshold function binary classifier, where an input value is mapped to a binary output based on an imposed threshold:

$$f(x) = \begin{cases} 1 & w \cdot x + b > 0 \\ 0 & \text{else} \end{cases} \tag{3.1}$$

Where w is a vector of weights and $x \cdot w$ is the dot product $\sum_{i=1}^{m} w_i x_i$, m is the number of inputs going into the perceptron node and b is the bias.

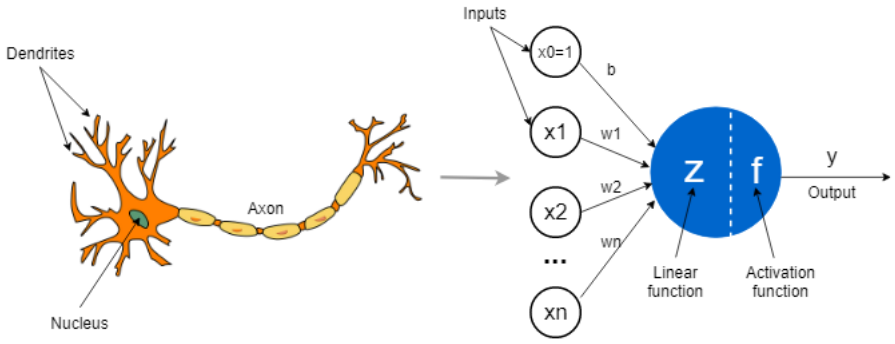


FIGURE 3.1: Diagram of the Perceptron compared to a human brain element from [50].

3.2 Artificial Neural Networks (ANNs)

Artificial neural networks are a further advancement in the progression of the original perceptron machine, also developed to mimic the biological brain. An ANN is a collection of interconnected nodes or neurons, similar to a synapse in the brain. Signals are transferred through the ANN between the neurons. The output of each of the neurons as the signal is passed forward through the network is computed as some non-linear function of the sum of its inputs. The connection between each of the connected neurons has a weight attached to it. This weight value is found during the training of the neural network and it allows for the weighting of the importance of the output of that specific node. Neurons also might have a threshold or activation attached to the output so that the output can be further tuned and non-linearised according to its value (see activation function section below). The training of a neural network is the updating of weights based on the information that the network can derive from the combination of input and desired output pairs and a chosen loss function that describes the relationship between the achieved network output and the desired output.

3.2.1 Activation Functions

Activation functions in machine learning models are used to define the output of a node based on the input. The purpose of having activation functions is to help decide how important the input value at that node is to the overall result of the model. There are many different types of activation functions that can be used, and most of them include a piece-wise type function. The following section will outline the main activation functions used in modern machine learning, and more specifically in this research.

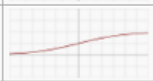
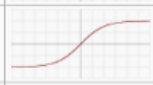
Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1+e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

FIGURE 3.2: Diagram of Activation Functions [51].

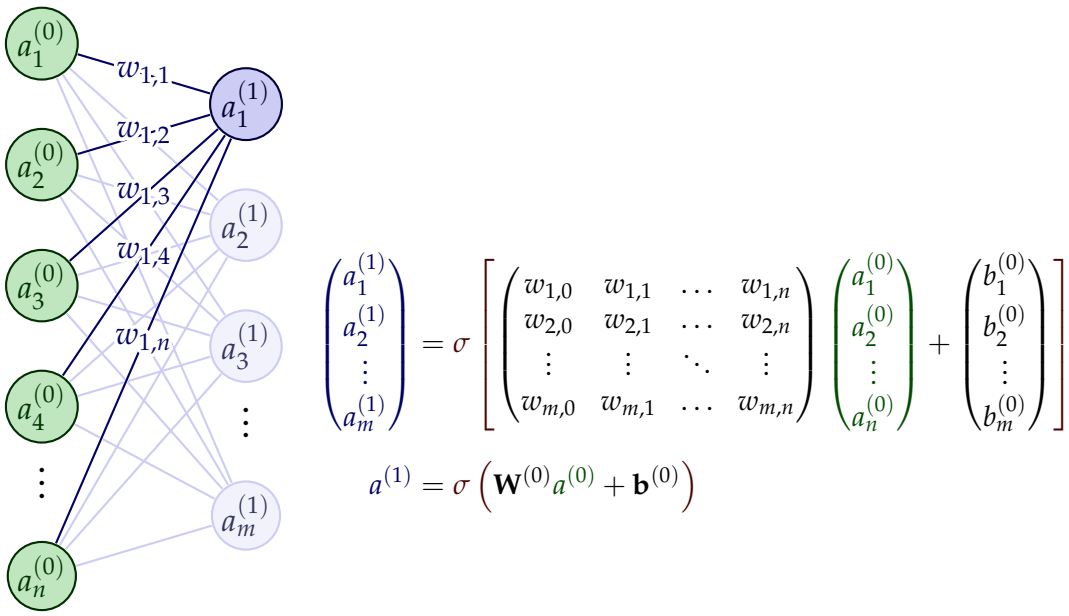


FIGURE 3.3: Diagram showing the formulae for the activation of a single node in the first hidden layer

Figure 3.3 demonstrates how the activation function σ is applied to the output of the first node of the first hidden layer of a neural network.

3.2.2 NN Training

The training of a neural network has been designed to mimic the learning of the human brain. In practice, this is undertaken through the use of three main components of the network, the loss function, backpropagation and the optimisation algorithm. The loss function is a formula that can be calculated after each forward pass of the network that will quantify the error of the results of that forward pass. This loss function can then be used in conjunction with the optimizer and back-propagation to calculate the gradient of the weights of the neural network and then update them appropriately to edge towards convergence. There are several important considerations when training a neural network, including the quality and quantity of the training data, the choice of network architecture and hyper-parameters, the use of regularization techniques to prevent over-fitting, and the selection of an appropriate loss function. It is also important to have a good understanding of the problem being solved and the desired outcome in order to choose the most appropriate neural network for the task. Additionally, ensuring that the training process is efficient and effective is crucial to the success of the model. This may involve the use of techniques such as batch normalization and gradient descent with momentum.

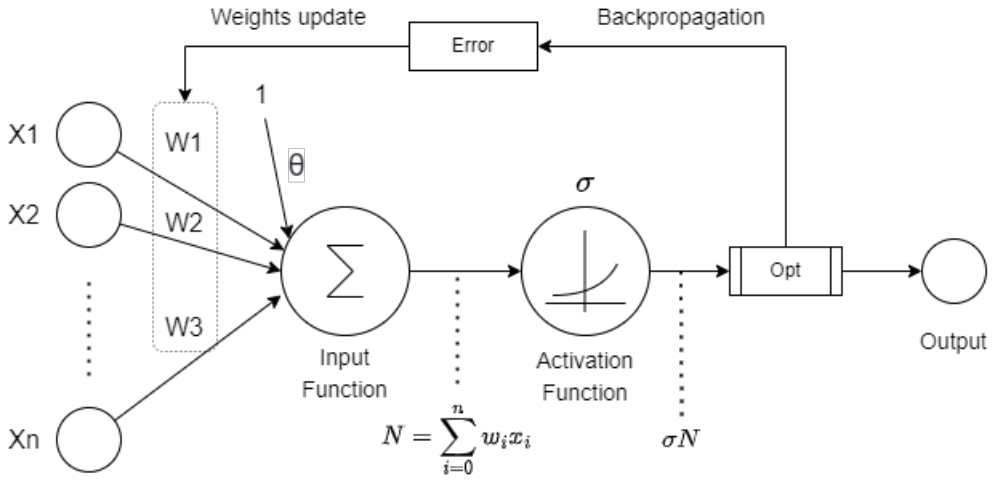


FIGURE 3.4: Diagram of the Training of a Neural Network: A look at a single node in the final layer.

Back-propagation

Back-propagation is an algorithm for calculating the gradient of a loss function with respect to the learned variables of a model, the weights, using the chain rule. Backpropagation calculates the gradient of the weights of the model with respect to the loss function using partial derivatives and the chain rule, working backwards from the output to the respective weights. How back-propagation works can be demonstrated by zooming into a single node in an output layer and understanding how the gradients of the input

weights to that node are calculated in relation to the output of the node, as seen in Figure 3.5.

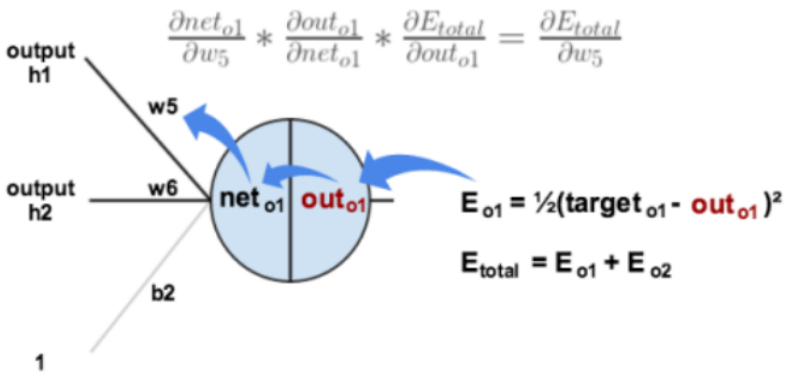


FIGURE 3.5: Diagram of Back-propagation on single node in output layer [52].

In Figure 3.5, output $h1$ and output $h2$ are the outputs from nodes in the previous hidden layer, $w5$, $w6$ are the weights associated with the connection between the node undergoing back-propagation and the nodes of the previous layer, and $b2$ is the bias associated with this specific node. net_{o1} is the output of the node before activation and out_{o1} is the output of the node after activation. In the formula in Figure 3.5, it can be seen that the chain rule is used to determine the partial derivative of the loss function E_{total} relative to the $w5$ weight. This is done for all the weights with respect to the E_{total} to calculate the overall gradient matrix of all the weights.

Optimization Algorithms

Once the weight gradients have been calculated using back-propagation or other automatic gradient calculation algorithms, an appropriate optimizer is then used to adjust weights in a manner that attempts to edge the model towards convergence. One of the most popular optimizers in machine learning is the stochastic gradient descent algorithm. The algorithm works by iteratively searching for local minima in the gradient space. SGD minimises the loss function $L(\theta)$ parameterised by the model's weights and biases, R, by updating the parameters in the reverse direction to the gradient of the loss function with respect to the parameters, θ. The learning rate is used to determine how big the steps are that the model takes down the gradient function towards the minima. The stochastic gradient descent algorithm can be described by the following steps:

1. Obtain gradients from back-propagation
2. Pick random initial values of parameters
3. Update gradient with new parameters

4. Calculate the step size down the gradient function using the learning rate multiplied by the gradient.
5. Calculate the new parameters by subtracting the step size from the old parameters
6. Redo steps 3-5 until local minima reached

Figure 3.6 is a diagram showing a visual representation of the descent of a gradient function using SGD.

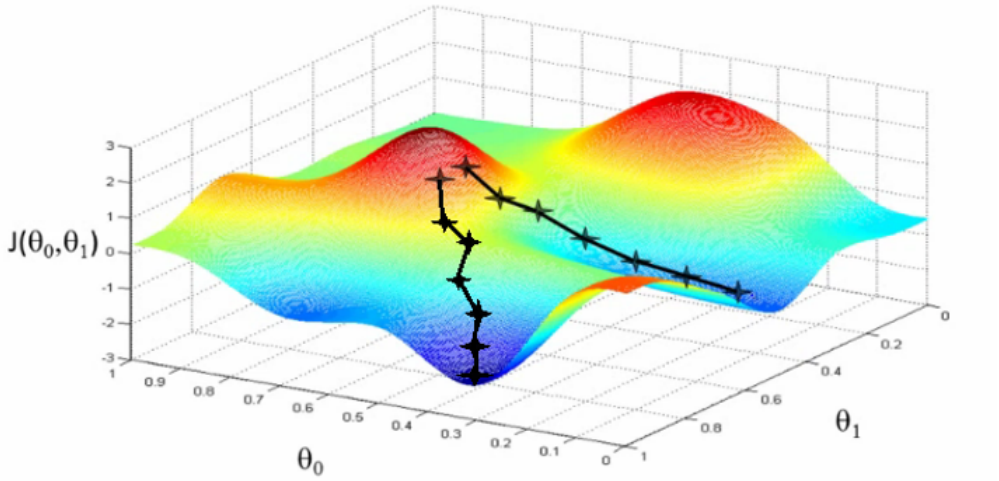


FIGURE 3.6: Diagram of Visual Representation of Descent of Gradient Function using SGD [53].

ADAM (Adaptive Moment Estimation) optimizer is another optimisation algorithm that is based on the foundation of stochastic gradient descent. ADAM is an extension of the stochastic gradient descent (SGD) algorithm, and it can be seen as a combination of two other algorithms: RMSProp and momentum. One of the main benefits of ADAM is its ability to adapt the learning rates of the model parameters independently, based on the past gradient values, unlike SGD where the same learning rate is used for all model parameters. This can make ADAM more effective than SGD because it can automatically tune the learning rates to be well-suited for the problem at hand. ADAM also makes use of momentum to accelerate the convergence of the optimization process. Momentum is a technique that helps the optimization algorithm to "pass over" local minima and avoid getting stuck in sub-optimal minima. Overall, the ADAM optimization algorithm is a powerful tool for training machine learning models. It can be used in a wide range of applications, and it can often provide faster and more stable convergence than other optimization algorithms.

Loss Functions

The loss function of a neural network is also known as the objective function. It is the mathematical expression that results in a metric that can quantify the

success of the model. This metric, the loss, is often some form of a measure of the difference between the expected output value of the model versus the actual value obtained. The loss function is used during the training of the model to evaluate the success of weight and bias parameters during a forward pass. There are many different loss functions that are used for different outcomes, models and training configurations. A number of the more common loss function components and more relevant loss functions to this work will be explained in this section.

Mean Square Error:

$$L_{MSE} = \frac{\sum_{i=1}^{D}(x_i - y_i)^2}{n}. \tag{3.2}$$

L_{MSE} is a loss function used often in regression-based tasks and is the average of the squared distances between the predicted values and the actual values. The squaring utilised within the mathematical formulation penalizes values heavily that are further away from the actual value. The L_{MSE} loss function also has good mathematical properties in terms of the continuity of the function which makes gradients easier to compute.

Binary Cross Entropy:

$$L_{BCE} = -(y \log(p) + (1 - y) \log(1 - p)). \tag{3.3}$$

L_{BCE} is a type of loss function used in classification tasks, where the function will output as a classification value indicating one of two classes. This loss function is used in conjunction with the sigmoid activation function.

Kullback–Leibler Divergence:

$$L_{KL} = KL(\hat{y}||y) = \sum_{c=1}^{M} \hat{y}_c \log \frac{\hat{y}_c}{y_c}. \tag{3.4}$$

The L_{KL} loss function is useful for estimating the difference between two given distributions, often referred to as the statistical distance. L_{KL} can be understood as the negative sum of the probability of each event in y multiplied by the log of the probability of the event in $\hat{y}$ over the probability of the event in y.

3.2.3 Machine Learning Supervision

When training a machine learning model there are a number of different strategies in which the machine learning model can be supervised to learn. Traditionally the most common form of supervision is full supervision. Full supervision refers to the scenario where the model is trained on a dataset that consists entirely of labelled examples. Each unique example passed to the model during training has both input features and a corresponding correct output label. The goal of the model is to make accurate predictions for new, unseen examples by learning the patterns in the training data. In unsupervised learning, the input data that the model is trained on is completely unlabeled and the model works by uncovering patterns and relationships

that exist in the input data and using these findings to output useful information about the data, for example, clusters of similar data points. Lastly, in semi-supervised learning portions of labelled data and unlabeled data can be used during training and inference of the model. This strategy helps to provide the model with a foundation of knowledge, and then use the unlabeled examples to help the model generalize and make better predictions.

3.3 Deep Neural Networks (DNNs)

A deep neural network can be simply defined as an ANN with multiple hidden layers between the input and output layers. The term 'deep' refers to the added complexity in the model in terms of the number of hidden layers and the number of nodes in each layer. Deep neural networks are often optimal for large datasets as the added complexity of the network affords the network the ability to capture higher levels of patterns that exist in the data. There are many different neural network architectures that exist, each used for different purposes and types of datasets. A brief overview of some of the main DNN derivatives will be detailed in this section.

3.3.1 Multi-Layer Perceptron (MLP)

Multi-layer perceptrons are the most basic form of NNs. MLP are types of feed-forward network that have a number of fully connected hidden layers between the input and the output layers. Fully connected is a term that is used to describe that all of the nodes in consecutive layers are connected to each other. Feed-forward is a term used to describe that the input data is only propagated forward through the network and not cycled back. The architecture of the MLP network forms the foundation of many of the more complex neural network architectures.

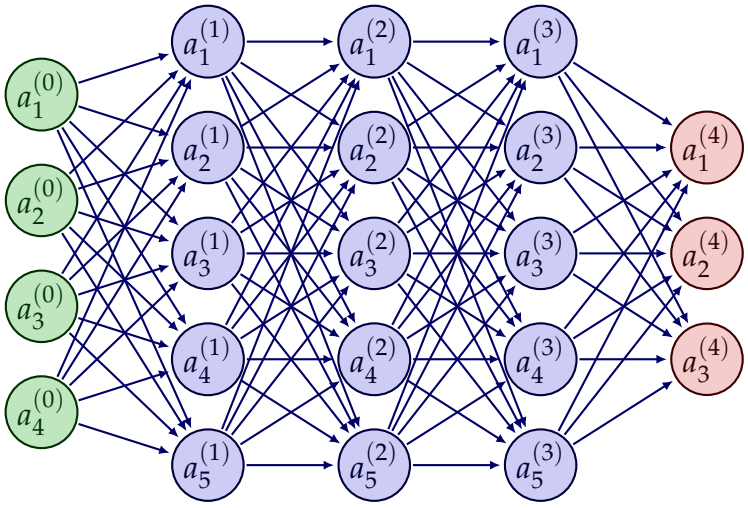


FIGURE 3.7: Diagram Showing Neural Network Structure of MLP.

3.3.2 Convolutional Neural Network (CNN)

CNNs are a class of NN most frequently used with image-based data. The main ability of a CNN that differentiates the architecture from a standard DNN is the convolution that is implemented during the forward pass. A convolution effectively is a systematic mask or crop on the data entering the network as to only concentrate on a smaller subset of the input data/image for the particular forward pass. This gives the network the ability to identify which sections or patterns in the input data are most relevant to obtaining the desired prediction. This convolutional functionality is what makes CNNs so successful with image data.

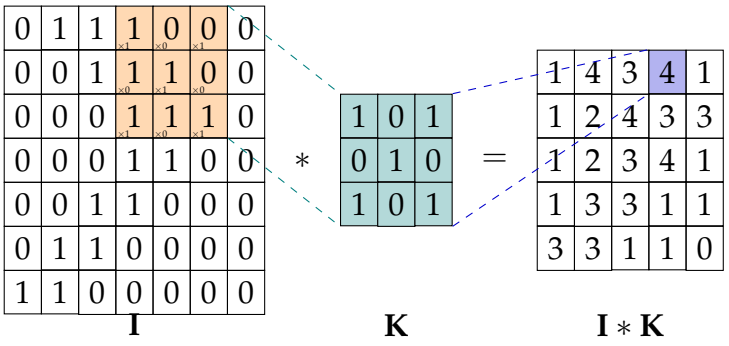


FIGURE 3.8: Diagram shows how the convolution in a CNN works, showing a visual representation o the matrix multiplication that occurs.

3.3.3 Recurrent Neural Network (RNN)

RNNs are NNs where the output of a specific layer is fed back into the input to help predict the next output. This methodology was developed for problems where data is sequential, this includes time-series data as well as natural language processing-based text data. This configuration allows information to be passed between adjacent nodes, as some information is retained from the previous time step. Though standard RNNs are often used in time series prediction, the standard architecture suffers from the problem of vanishing gradients, which hinders the learning of long-period relationships and patterns in data sequences. RNNs with LSTM solve the issue by deliberately adding long-term memory [54], [55]. RNNs with LSTM have two memory cells, one for long-term memory and another for short-term memory to solve the problem. The equations below describe the LSTM RNN block. C_t is responsible for long-term memory and h_t for short-term memory. The introduction of the Forget gate vector, proposed by Felix A. Gers et al. in 1999, has also improved the accuracy of the prediction by allowing the adjustment of long-term memory [56]. More in detail, the Forget gate vector, f_t, controls how much information is discarded from long-term memory, C_{t-1}. The new long-term memory cell, C_t, is created by adding information from the input gate vector, i_t, and the new short-term memory cell, h_t is decided by the output vector, o_t and the long-term memory, C_t (Figures 3.9–3.11).

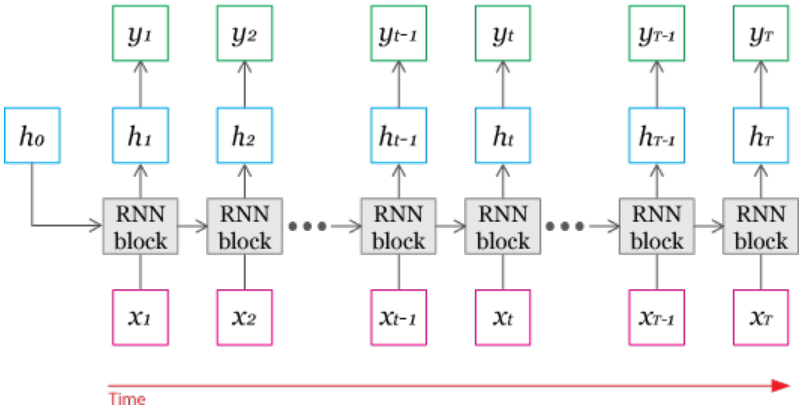


FIGURE 3.9: Recurrent Neural Network (RNN) structure.

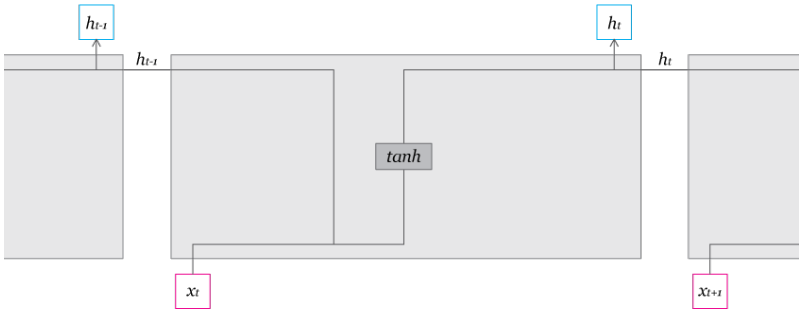


FIGURE 3.10: Details of the Simple Recurrent Neural Network (RNN) Block.

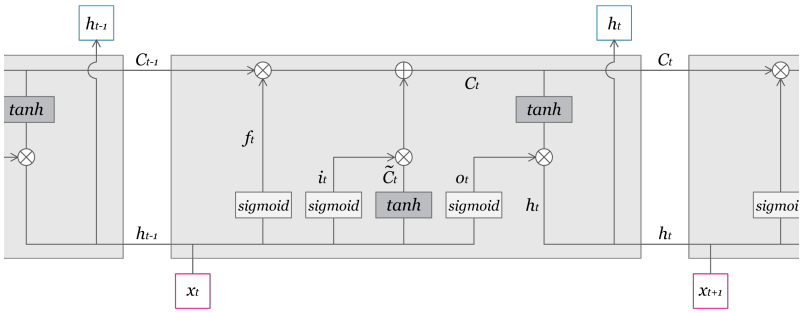


FIGURE 3.11: Details of the Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) Block.

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) && (3.5)\\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) && (3.6)\\
\widetilde{C}_t &= \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) && (3.7)\\
C_t &= f_t \odot C_{t-1} + i_t \odot \widetilde{C}_t && (3.8)\\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_0) && (3.9)\\
h_t &= o_t \odot tanh(C_t), && (3.10)
\end{aligned}
$$

where $\odot$ represents the Hadamard product.

3.3.4 Auto-encoders (AEs)

Variational autoencoders (VAEs) are a type of generative model that was first introduced in 2013 by Diederik Kingma and Max Welling. VAEs combine the strengths of both generative models and autoencoders, allowing them to generate new data samples that are similar to the training data, while also being able to compress the data into a lower-dimensional latent space. VAEs are based on the idea of using a neural network to approximate the complex distribution of the data, and they have become a popular tool in many areas of machine learning, such as image generation, speech recognition, and natural language processing. In recent years, VAEs have been further developed and improved, and they continue to be an active area of research in the field of deep learning. A Variational Auto-encoder (VAE) is a type of neural network that is similar to a classic autoencoder (AE) but with slight architectural changes and an additional component added to the loss function. These modifications allow for regularized training to prevent overfitting and improve the model's generative capabilities by ensuring appropriate latent space properties. The VAE architecture consists of two main networks: an encoder and a decoder. It is trained to minimize the loss between the input data (kinematic variable events) and the encoded-decoded output (reconstructed event). However, unlike an AE, which encodes an input event as a single vector, the VAE encodes the input as a distribution over the latent space. This regularizes the latent space. Variational inference is a mathematical technique used to approximate complex probability distributions using simpler distributions. In the context of a VAE, it is used to approximate the complex data distribution using a simpler distribution that can be learned by the model. The KL divergence loss term is a measure of the difference between two probability distributions. In a VAE, it is used to measure the difference between the approximate distribution learned by the model and the true data distribution. This loss term is then used to train the model to minimize the difference between these distributions, improving its ability to generate data. By combining variational inference and the KL divergence loss term, a VAE can learn a compact representation of the data distribution that can be used to generate new samples similar to the original data. This makes the VAE a useful tool for data generation and unsupervised learning tasks.

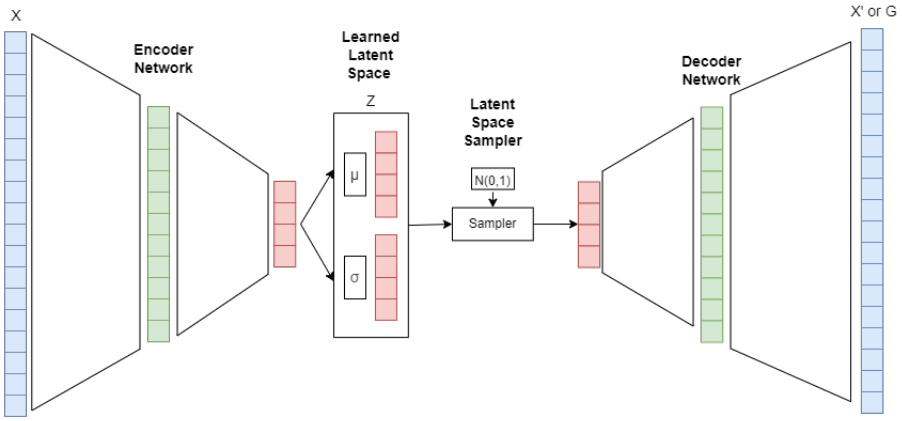


FIGURE 3.12: Diagram of VAE Base Architecture, showing encoder network, decoder network and learned latent space.

These latent space distributions are normal so that the encoder can be configured to return latent space vectors that represent the mean and the covariance of the normal Gaussian distributions. As a result of encoding an input event to a distribution rather than a single vector, it is possible to express the latent space regularisation. The loss function that is minimised when training a VAE is composed of a reconstruction loss component that is responsible for forcing the output of the decoder to be as close to the input, and secondly, a regularisation loss component, that serves to regularise the organisation of the latent space by making the distributions returned by the encoder close to a standard normal distribution. The loss function also contains a coefficient for the KL-Divergence loss term. This can be used during optimisation to weight the importance of the KL-Divergnce loss term against that of the reconstruction loss term. The VAE loss function can be written as:

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}q_\phi(z|x)\left[\log p_\theta(x|z)\right] - \beta_V * D_{\text{KL}}\left(q_\phi(z|x)||p_\theta(z)\right)$$

.

The first term in the VAE loss function, $\mathbb{E}q\phi(z|x)\left[\log p_\theta(x|z)\right]$, is the reconstruction loss, which measures how well the VAE is able to reconstruct the input data. The second term, $D_{\text{KL}}\left(q_\phi(z|x)||p_\theta(z)\right)$, is the regularization loss, which encourages the VAE to learn a compact and meaningful latent space representation. More generally as:

$$\mathcal{L}_{VAE} = \mathcal{L}_R + \beta_V * \mathcal{L}_{KL}, \tag{3.11}$$

and the loss function sub-components can also be rewritten in terms of the in-code format as follows:

$$\mathcal{L}_R = \overline{((X' - X)^2)}, \tag{3.12}$$

where X is the input event and X' is the reconstructed event.

$$\mathcal{L}_{KL} = \sum(\sigma^2 + \mu^2 - \log\sigma - 1) \tag{3.13}$$

Where μ is the mean of the latent space variable and σ is the variance of the latent space variable. The Kullback-Leibler divergence between two Gaussian distributions has a form that can be directly expressed in terms of the means and the covariance matrices of the two distributions.

Chapter 4

VAE Model Development Methodology

The following section will specifically concentrate on the machine learning methodology. This includes the detail of the model selection, overall model and model component descriptions, model hyper-parameter optimisation, machine learning library detail and additional software development detail.

4.1 Model Description

A VAE architecture has been selected as the base architecture because of its proven capacity to be used as a generative model for complex data. A VAE can be viewed as a modified classic autoencoder (AE) with slight architectural changes and an additional component added to the loss function that allows for regularised training and introduces the generative capability of the model by ensuring appropriate latent space properties.

4.1.1 Variational Auto-encoder (VAE)

The encoder network of a VAE serves to translate input data from its original dimensionality into a different feature representation, usually of a lower dimensionality called the latent space. The decoder network is used to decode a latent space representation back into its original representation. The latent space is the representation of the input data once the data has been fed-forward through the encoder. Different types of AEs have different latent space formats. The VAE has a latent space that is forced to be normal and Gaussian, whilst the latent space of an AE has no such constraint and is rather a single vector representation.

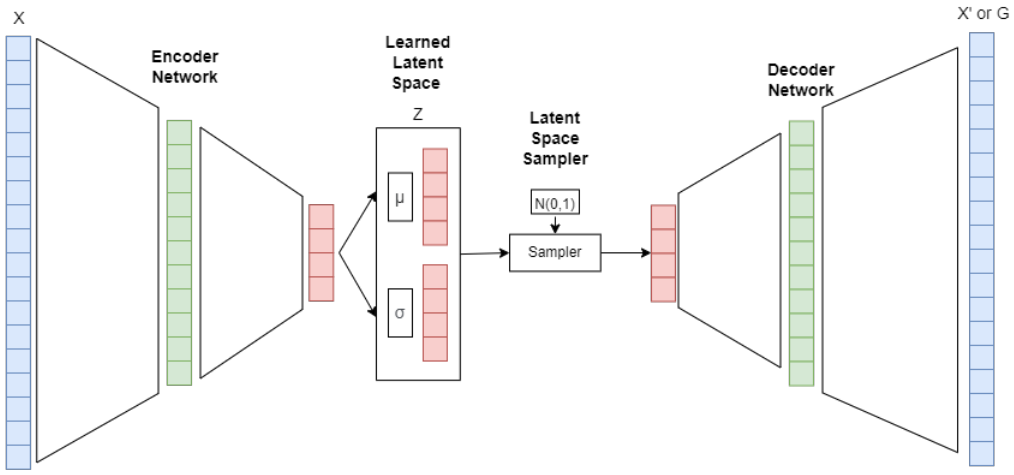


FIGURE 4.1: Diagram VAE architecture.

4.1.2 Variational Auto-encoder + Discriminator (VAE+D)

A Variational Autoencoder plus discriminator has been used as an alternative option to the classic VAE in this research. The addition of a discriminator network helps in the training of the VAE encoder-decoder network and provides an additional method for completing classification tasks. Figure 4.2 shows the architecture diagram of the VAE+D model. The addition of the discriminator network was inspired by the use of such a network in the training of a Generative Adversarial Network (GAN). During the training of the network, there is a balance that needs to be achieved between the training of the discriminator to be able to differentiate MC events from reconstructed events and generated events, and the training of the VAE to be able to reconstruct and generate valid events. This makes the training of the VAE+D somewhat more complex than the already infamously difficult-to-train VAE. The discriminator network is a classic MLP network with a single output node. The output value of this final node is activated with a sigmoid activation function as to achieve a classification score. The VAE+D loss functions are slightly different to the VAE, whilst still including the main loss components of the original VAE. Unlike the VAE, the VAE+D has a loss function for each individual network, the encoder, decoder and discriminator and each network's weights are updated individually at different times during a forward pass of the overall VAE+D. Similar to a GAN, the discriminator and the VAE are trained simultaneously, with the discriminator learning to differentiate fake events from real events and the VAE learning to reproduce real events accurately. The VAE+D loss functions and components are as follows:

$$L_{disc} = BCE_{real} + BCE_{recon} + BCE_{gen} \tag{4.1}$$

$$L_{dec} = \gamma * Loss_R - Loss_{disc} \tag{4.2}$$

$$L_{enc} = Loss_{KL} + \gamma * Loss_R \tag{4.3}$$

Where $Loss_R$ and $Loss_{KL}$ are as defined in 3.3.4. BCE is the binary cross entropy between either the actual data against actual data, reconstructed data or generated data. The final discriminator loss function, L_{disc} is obtained as the sum of the three BCE based losses. γ is a similar variable to the variational beta in the standard VAE, however, instead is a coefficient of the reconstruction loss instead of the KL divergence like in the VAE. More detail on the training iteration of this model configuration can be seen in 4.3.

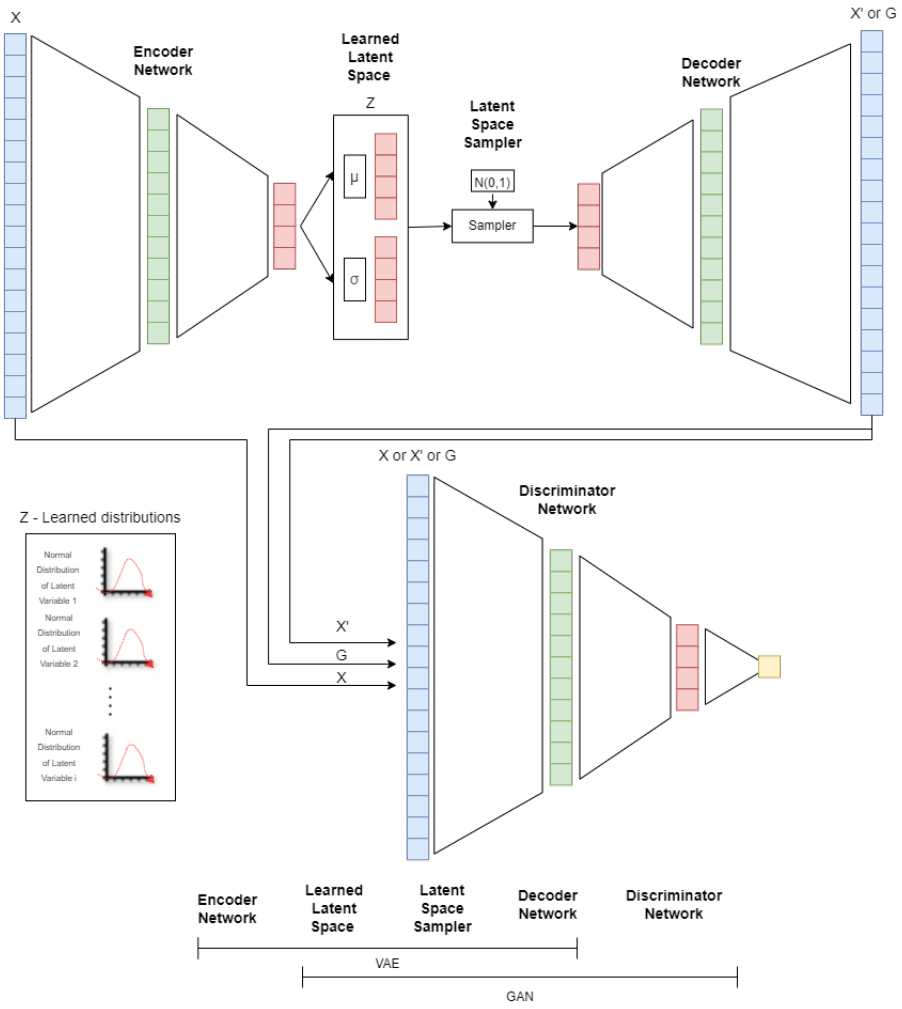


FIGURE 4.2: Diagram VAE+D architecture.

4.1.3 Normalising Flows (NF)

An NF model is a type of ML model that is used to learn complex probability distributions. Similarly to VAEs, NFs are used in generative modelling tasks, where the goal is to generate new data that is similar to the original data. Normalising Flow (NF) models are likelihood-based generative models and consist of a series of bijector, invertible transformations that can be applied to an input distribution to change the distribution to be more complex. Normalising flows can be applied on top of the Gaussian latent spaces of a VAE to make the learned latent space distributions more complex and therefore able

to retain more complex patterns and relationships that exist in the data. An NF model builds a probability distribution of u by applying bijective functions to a known distribution z. These bijective functions keep the probability mass normalised and can be used to transform and inverse transform as they have well-defined inverses [57]. The change of variables formula is used to find the probability distribution of x, this requires a function inverse and jacobian. Deep flow models can exist by chaining multiple bijectors together. These bijector functions have trainable parameters. Once trained, a normalizing flow model can be used to generate new samples by sampling from the learned distribution and then applying the inverse transformations to the samples to transform them back into the original space. This allows the model to generate new samples that are similar to the original data.

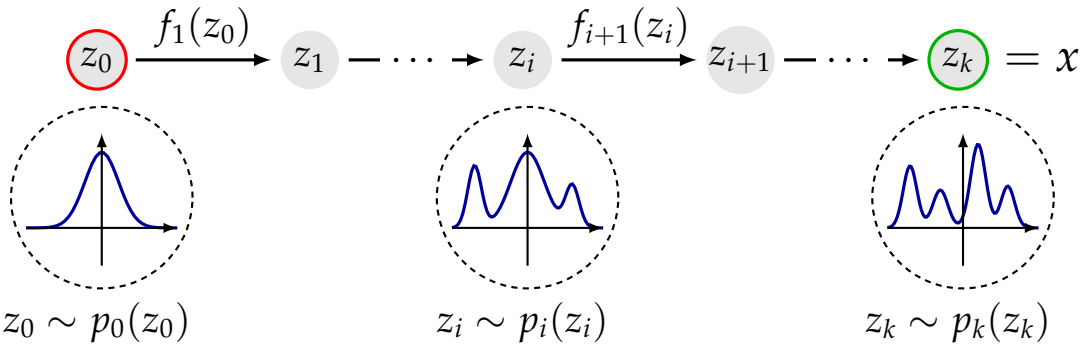


FIGURE 4.3: Diagram of Normalising Flows.

The key to normalizing flow models is the use of the change of variable formula, which allows us to transform random variables in a way that preserves the probability distribution. Specifically, if we have a random variable X with a probability density function (PDF) $p(x)$, and we apply a bijective transformation $f(x)$ to X, the resulting random variable Y will have a PDF given by:

$$p(y) = p(x)\,|\det\left(\mathrm{Jac}(f)(x)\right)|$$

,

where $\mathrm{Jac}(f)(x)$ is the Jacobian matrix of the transformation $f(x)$ at x. This formula allows us to transform random variables in a way that preserves the probability distribution, which is important for generating samples from the target distribution.

Types of Normalising Flow Algorithms Used

In this work, previously developed NF algorithms have been utilised instead of developing novel algorithms. The following NF algorithms were utilised, planar, radial, householder and nice. Each of these NF models work slightly differently and have advantages and disadvantages. The type of NF model and the flow length (number of bijector transformations applied sequentially) are included as hyper-parameters in our model development. Transformational forms of the chosen NF models can be seen below:

1. Planar flow transformation form:

$$y = x + u \tanh(w^T x + b)$$

, where $x \in \mathbb{R}^d$ is the input, $y \in \mathbb{R}^d$ is the output, $u \in \mathbb{R}^d$, $w \in \mathbb{R}^d$, and $b \in \mathbb{R}$ are learnable parameters.

2. Householder flow transformation form:

$$y = x - 2uu^T x$$

, where $x \in \mathbb{R}^d$ is the input, $y \in \mathbb{R}^d$ is the output, and $u \in \mathbb{R}^d$ is a learnable parameter.

3. Radial flow transformation form:

$$y = x + \frac{u}{|x|}$$

, where $x \in \mathbb{R}^d$ is the input, $y \in \mathbb{R}^d$ is the output, and $u \in \mathbb{R}^d$ is a learnable parameter.

4. Nice flow transformation form:

$$y = x + u \odot \left(\sin\left(\frac{\pi}{2}x\right)\right)$$

, where $x \in \mathbb{R}^d$ is the input, $y \in \mathbb{R}^d$ is the output, and $u \in \mathbb{R}^d$ is a learnable parameter.

4.1.4 Variational Auto-encoder + Discriminator + Normalising Flows (VAE+D+NF)

Another VAE model variation that was evaluated in this research was the composite model achieved as the result of the addition of a normalising flow model to the VAE+D model. The NF model is inserted just after the gaussian latent space of the VAE. This transforms the stored latent space distributions using the bijectors of the NF. This allows for more complex distributions to store latent space information, whilst still incorporating the training benefits related to the addition of the discriminator network.

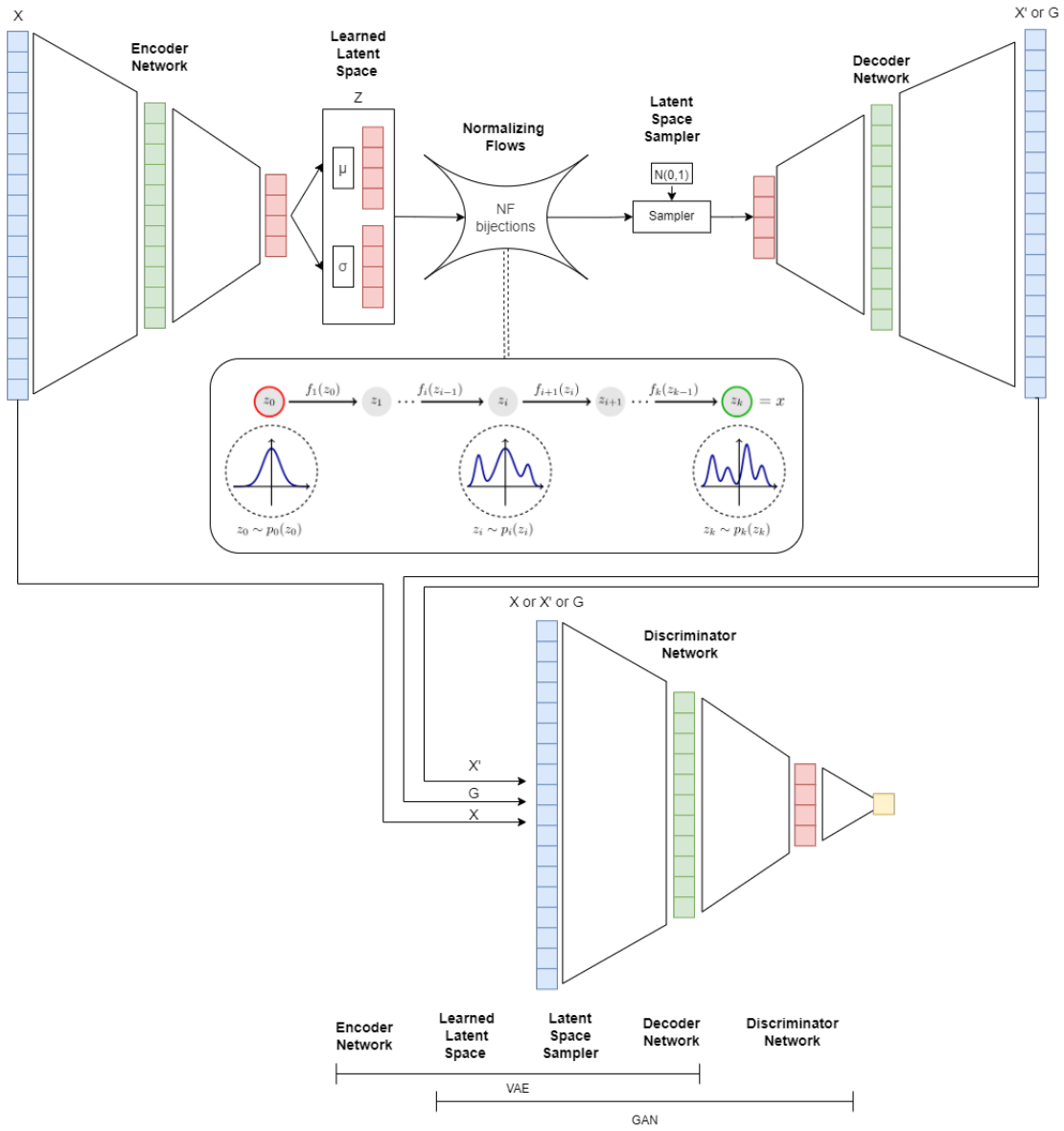


FIGURE 4.4: Diagram VAE+NF+D architecture.

4.2 Data Pre-processing

Data preprocessing is a step in the machine learning pipeline that involves cleaning and transforming the input data before it is used to train a machine learning model. Data preprocessing is an crucial step when implementing machine learning models, and it can have a significant impact on the performance and generalization of the model. The goal of data preprocessing is to make the input data more suitable for training the specific model and to improve the performance of the model. There are several common techniques used in data preprocessing for machine learning models, including:

- Data cleaning: This involves removing or correcting errors and inconsistencies in the input data, such as missing values, outliers, and duplicate records.

- Data transformation: This involves applying mathematical transformations to the input data to make it more suitable for training the model, such as scaling or normalizing the data, or applying transformations to extract features and patterns from the data.
- Data splitting: This involves dividing the input data into separate training and test sets, which are used to train and evaluate the performance of the model.

The sklearn library data transformations that were assessed for this work can be seen in Table 4.1.

Scalar/Transformation	Transformation Description
StandardScalar	Standardize features by removing the mean and scaling to unit variance.
MinMaxScalar	Scales feature ranges to desired range (default [0,1]).
RobustScalar	Removes the median and scales the data according to the quantile range (defaults to IQR: Interquartile Range)
PowerTransformer	Apply a power transform featurewise to make data more Gaussian-like.
QuantileTransformer	Transform features using quantiles information.

TABLE 4.1: Types of Sklearn Transformations Assessed

It was found that the best scalar for the data preparation is the MinMaxScalar.

4.3 Model Training

The VAE is trained on pure $Z\gamma$ final state background data on batches delivered by a custom data loader. During this training, the VAE learns to reconstruct the input events and furthermore regularises the latent space by minimising the loss function.

4.3.1 Important Training Parameters

Learning rate:
The learning rate is a standard machine learning hyper-parameter that determines how drastically the model changes its weights each iteration in an attempt to minimise the loss function and achieve convergence. The learning rate always has a drastic effect on the ability of the model to converge and achieve a favourable final model. A learning rate that is too large will cause the model to pass over a local minima in the gradient function, however,

a learning rate that is not large enough could take long to converge or converge in a local minimum that is not the overall actual minima of the gradient function. It is therefore essential to tune the learning rate for the model.

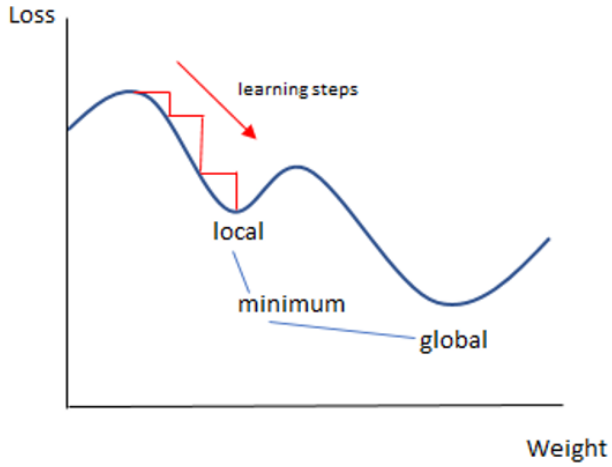


FIGURE 4.5: Local minima in gradient function.

Batch size:
The batch size used when training a VAE refers to the number of training examples used in one iteration of the model. In this use case, a training example is a single event with all the respective features. For example, if the batch size was 1, a single event is passed through the model before the weights are updated through the use of the optimizer. In contrast, if the batch size was 512, 512 events would be passed through the VAE before updating the weights using the average loss value for all the events passed through in that batch.

Batch Normalisation:
Batch normalization is a technique used to normalize the activations of hidden units in a neural network, improving performance and stability. It normalizes the activations of each hidden unit across a mini-batch by subtracting the mean activation and dividing by the standard deviation, reducing the internal co-variate shift during training. It is commonly used in modern deep-learning models and is a key component of many state-of-the-art architectures. The use of batch normalization in a VAE can help to improve the performance and stability of the model.

Variational Beta:
As previously mentioned in the VAE model description section the loss function contains a coefficient for the KL-Divergence loss term, β_V. This can be used during training to weight the importance of the KL-Divergnce loss term against that of the reconstruction loss term. The effect of different values of β_V during training will prioritise the reconstruction of the events fed through the VAE against regularising the latent dimension of the VAE, which is important for generating realistic events.

Parameter γ:
The γ parameter is similar to the variational beta parameter, except is the

coefficient of the reconstruction loss term instead of the KL divergence loss term.

4.3.2 Model Specific Loss Functions and Training Iteration

The following section outlines the specific loss functions of each model variation as well as the structure of the training iterations.

VAE

During training a VAE's weights are adjusted by the optimization function to minimise the loss function, Equation 3.11, after each iteration of the training loop. VAEs have hyper-parameters that need to be optimised in order to achieve a satisfactory model. Algorithm 5 shows the flow of a single forward pass of the VAE network during training.

Algorithm 2 Algorithmic representation showing forward pass of VAE.

$eventbatch \rightarrow VAE \rightarrow \mu, \sigma^2, reconstructed$
$z_p \rightarrow Decoder \rightarrow generated$
$Loss_{recon} \rightarrow ((reconstructed - eventbatch)^2).mean()$
$Loss_{KL} \rightarrow 1 + \sigma - \mu^2 - e^{\sigma}$
$Loss_{VAE} \rightarrow Loss_{recon} + \beta_v Loss_{KL}$
VAE backward step

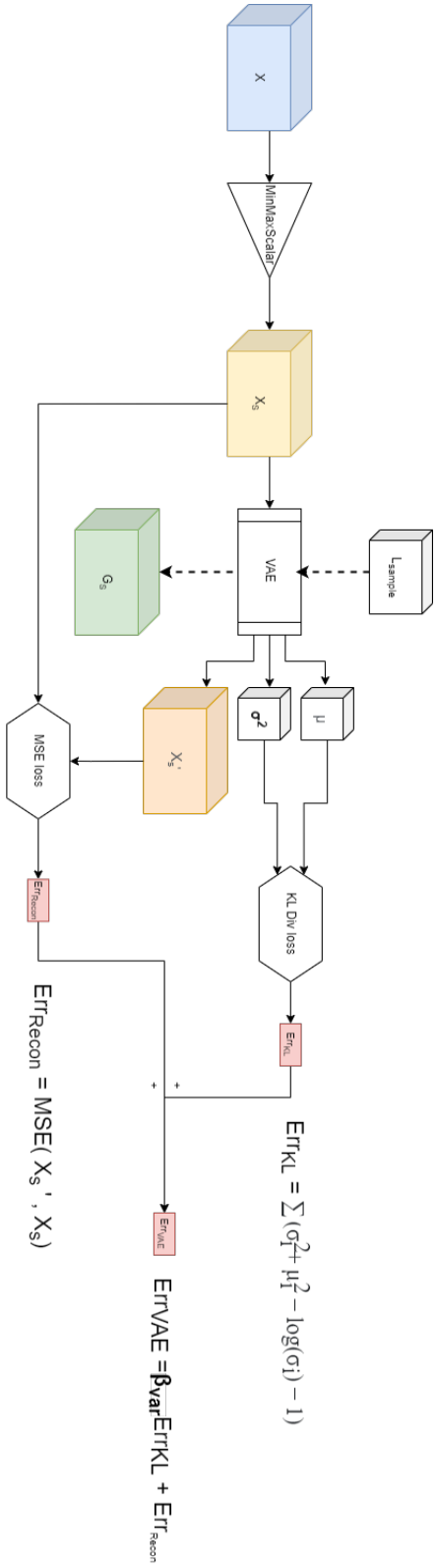


FIGURE 4.6: Flow diagram of VAE training iteration with loss functions.

VAE+D

The VAE+D loss functions are slightly different to the VAE, whilst still including the main loss components of the original VAE. Figure 4.7 shows the flow diagram of the input event batch during a forward pass and the resulting loss parameters and backward passes.

Algorithm 4 Algorithmic representation showing forward pass of VAE+D.

$eventbatch \rightarrow VAE \rightarrow \mu, \sigma^2, reconstructed$
$z_p \rightarrow Decoder \rightarrow generated$
$eventbatch \rightarrow Diss \rightarrow diss_{real}$
$diss_{real} vs. 1 slabel \rightarrow criterion_{BCE} \rightarrow Err_{D-real}$
$reconstructed \rightarrow Diss \rightarrow diss_{recon}$
$diss_{recon} vs. 0 slabel \rightarrow criterion_{BCE} \rightarrow Err_{D-recon}$
$generated \rightarrow Diss \rightarrow diss_{gen}$
$diss_{gen} vs. 0 slabel \rightarrow criterion_{BCE} \rightarrow Err_{D-gen}$
$Loss_{disc} \rightarrow Err_{D-gen} + Err_{D-recon} + Err_{D-real}$
Discriminator backward step
$Loss_{recon} \rightarrow ((diss_{recon} - diss_{real})^2).mean()$
$Err_{Decoder} \rightarrow \gamma * Loss_{recon} - Loss_{disc}$
Decoder backward step
$Loss_{KL} \rightarrow 1 + \sigma - \mu^2 - e^{\sigma}$
$Err_{Encoder} \rightarrow Loss_{KL} + \gamma * Loss_{recon}$
Encoder backward step

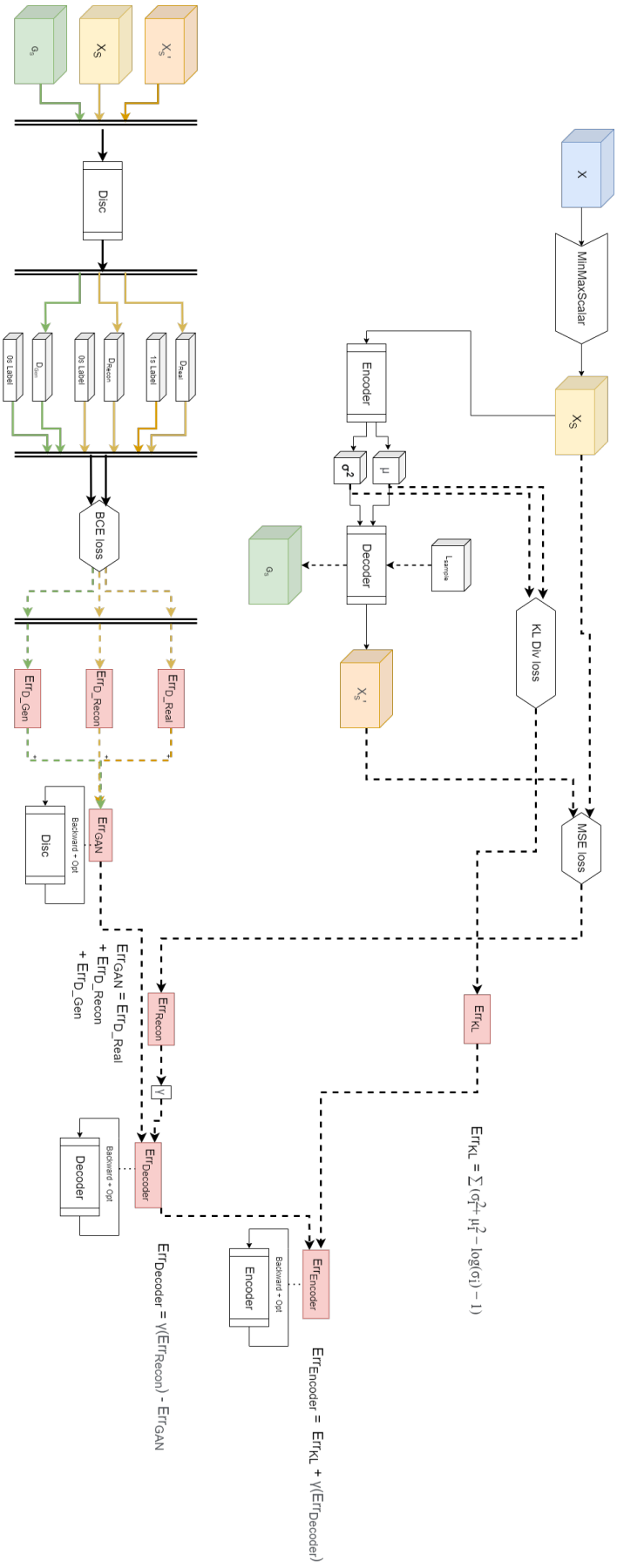


FIGURE 4.7: Flow diagram of VAE+D training iteration with loss functions.

VAE+NF

The VAE+NF forward pass algorithm is similar to that of the VAE forward pass, except the NF model is incorporated in between the traditional VAE latent space and the decoder network.

Algorithm 6 Algorithmic representation showing forward pass of VAE+NF.

$eventbatch \rightarrow VAE \rightarrow \mu, \sigma^2, reconstructed$
$\mu, \sigma^2 \rightarrow NF \rightarrow z_s$
$z_p \rightarrow Decoder \rightarrow generated$
$Loss_{recon} \rightarrow ((reconstructed - eventbatch)^2).mean()$
$Loss_{KL} \rightarrow 1 + \sigma - \mu^2 - e^{\sigma}$
$Loss_{VAE} \rightarrow Loss_{recon} + \beta_v Loss_{KL}$
VAE+NF backward step

VAE+NF+D

The VAE+NF+D forward pass algorithm is similar to that of the VAE+D forward pass, except the NF model is incorporated in between the VAE+D traditional latent space and the decoder network.

Algorithm 8 Algorithmic representation showing forward pass of VAE+NF+D.

$eventbatch \rightarrow VAE \rightarrow \mu, \sigma^2, reconstructed$
$\mu, \sigma^2 \rightarrow NF \rightarrow z_s$
$z_p \rightarrow Decoder \rightarrow generated$
$eventbatch \rightarrow Diss \rightarrow diss_{real}$
$diss_{real} vs. 1 slabel \rightarrow criterion_{BCE} \rightarrow Err_{D-real}$
$reconstructed \rightarrow Diss \rightarrow diss_{recon}$
$diss_{recon} vs. 0 slabel \rightarrow criterion_{BCE} \rightarrow Err_{D-recon}$
$generated \rightarrow Diss \rightarrow diss_{gen}$
$diss_{gen} vs. 0 slabel \rightarrow criterion_{BCE} \rightarrow Err_{D-gen}$
$Loss_{disc} \rightarrow Err_{D-gen} + Err_{D-recon} + Err_{D-real}$
Discriminator backward step
$Loss_{recon} \rightarrow ((diss_{recon} - diss_{real})^2).mean()$
$Err_{Decoder} \rightarrow \gamma * Loss_{recon} - Loss_{disc}$
Decoder backward step
$Loss_{KL} \rightarrow 1 + \sigma - \mu^2 - e^{\sigma}$
$Err_{Encoder} \rightarrow Loss_{KL} + \gamma * Loss_{recon}$
Encoder backward step

4.3.3 Hyper-parameter Optimization

VAEs are particularly sensitive to hyper-parameter, and in order to achieve favourable results, optimisation of hyper-parameters is crucial. A number of phases of hyper-parameter optimisation were completed throughout the

research. The first iteration of hyper-parameter optimisation was somewhat more manual than the latter iterations because the possible parameter values had large variances in order to understand the correct rough ranges of specific hyper-parameters. Within the manual optimisation loops, which randomly loop through chosen values of each hyper-parameter, classes and functions are called to; build the VAE with the parameters of the current loop iteration, train the VAE, evaluated the results and save relevant log-able information and metrics and create plots. The created log files can then be referenced for the performance of each combination of hyper-parameters. The information of the runs can be sorted by the loss function and other performance metrics to obtain the best parameters. Plots can then be referenced for confirmation of performance. After the initial exploration of hyper-parameters, a second, more refined selection of hyper-parameter values were selected in order to have a selection of possible values that were more plausible in achieving better results. Table 4.2 shows the secondary hyper-parameter optimisation value options chosen.

VAE+D Hyper-parameter Optimisation		
Hyper-parameter	Value Options	Applicable Model
Learning Rate	[0.01, 0.001, 0.0001]	All
Batch Size	[64, 256, 512, 1024]	All
Activation function	[elu, relu, sigmoid]	All
Optimiser	[ADAM, RMSprop]	All
Latent Dimension Size	[8, 12, 16, 18, 32]	All
Number of Hidden Layers in VAE	[2, 3]	All
Number of Hidden Layers in Discriminator	[2, 3]	VAE+D, VAE+NF+D, CVAE+D
Number of Nodes in the Hidden Layers of VAE	[128, 256, 512]	All
Number of Nodes in the Hidden Layers of Discriminator	[128, 256, 512]	VAE+D, VAE+NF+D, CVAE+D
γ	[1, 10, 100, 500, 1000, 2000]	VAE+D, VAE+NF+D, CVAE+D
Variational Beta	[1, 0.1, 0.01, 0.001, 0.0001]	VAE, VAE+NF
NF Type	[planar, radial, householder, nice]	VAE+NF, VAE+NF+D

TABLE 4.2: Table showing VAE Selected Hyper-parameters and value options.

It must be noted that the hyper-parameter optimisation process took an extensive amount of time as the training of a single model for evaluation

takes roughly 8 hours. That means that if all the possible options of hyper-parameter combinations for the base VAE were assessed it would be impossible to complete due to a lack of time and training resources. This is why the random selection of hyper-parameter values for each iteration was necessary instead of iteratively looping through every value in each hyper-parameter array. Values of specific hyper-parameters that produced obviously bad results when used were also removed from the value options as soon as this could be concluded. For example, it became clear that the learning rate used with the VAE+D model was definitely best around the 0.0001 range and therefore the other options were removed from further optimisation tests.

4.4 Use of the trained decoder and latent space for $Z\gamma$ Data Generation

Once the VAE has been trained on the $Z\gamma$ final state background data, we can use the latent space and decoder components to generate new events that mimic the $Z\gamma$ input data. Figure 4.8 shows a graphical representation of how the latent space normal distributions can be sampled from in order to generate new events.

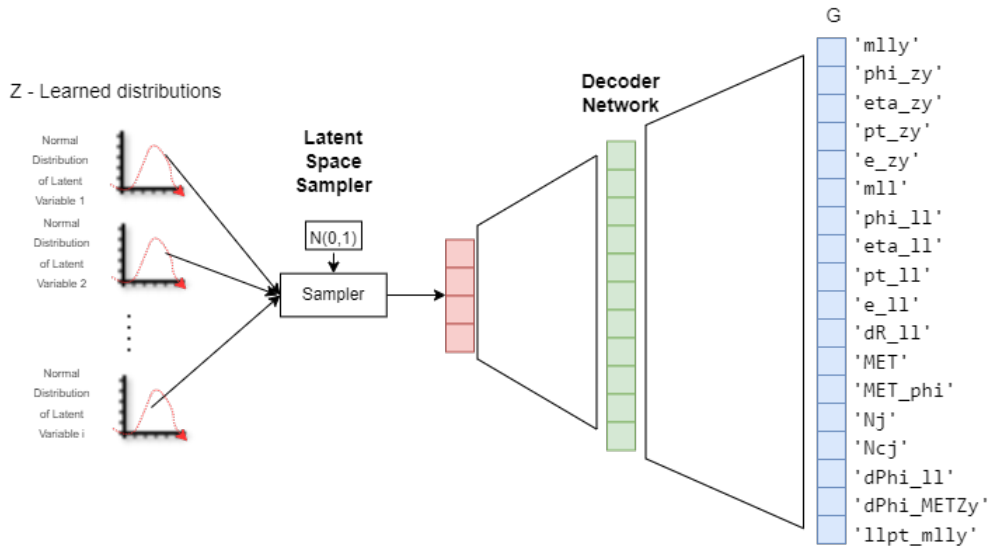


FIGURE 4.8: Diagram showing VAE components during Data Generation.

4.5 Use of the trained model for Classification

Although out of the scope of this research, the VAE derivative models trained for the data generation task in this research could also be used in a number of ways for signal/event classification. The following table outlines the main ways each model could be used for signal classification.

Model	Primary classification	Secondary classification
VAE Derivatives	Monitoring of reconstruction loss	Latent space proximity analysis
VAE+D Derivatives	Use of trained discriminator network	Monitoring of reconstruction loss, Latent space proximity analysis

TABLE 4.3: Mechanisms of signal classification.

A VAE trained for generative purposes can potentially be used as a classification system for classifying signal events against known background events by monitoring the reconstruction loss term against a previously determined threshold. If the signal data is significantly different enough to the background data on which the VAE was trained, the reconstruction loss can be monitored and a threshold can be applied to indicate a reconstruction loss value high enough to identify an input event as a signal event rather than the known background events. This threshold can be determined by feeding a test set of background data through the trained VAE, storing the reconstruction loss values for each of the events fed through the network from the test set, and finally, choosing a threshold value that separates the two reconstruction loss distributions created by the background data and the signal data. Additionally, plots of the latent space can be used as a secondary measure to asses the classification using the reconstruction loss. Distribution of each latent space variable plots and principle component plots of the latent space can also be used to identify 'outliers' (signal). A signal event might not be in close proximity to other points within the latent space. The discriminator in the VAE+D could be used for classification purposes as it is trained to recognise background data. When an event that is not background is fed through the trained discriminator network, the binary output will indicate that it is not a background event. However, as stated, the classification capabilities of the VAE derivative models will not be assessed in this work because of the chosen scope of the research and the fact that the chosen variables for the research are not variables that are valuable for the successful separation of background and signal events at the centre of mass of 150 GeV.

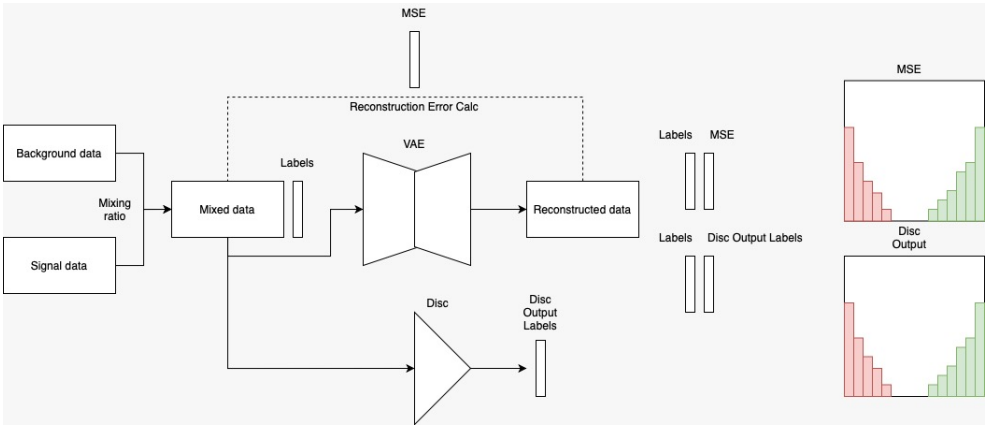


FIGURE 4.9: Diagram showing the signal classification mechanisms of the VAE derivative models.

4.6 Software Development Methodology

The following section will outline the development of the software required for this research.

4.6.1 Machine Learning Frameworks and Libraries

TensorFlow and PyTorch are two popular open-source machine-learning frameworks that are commonly used for developing and training machine-learning models. Both frameworks provide a rich set of tools and libraries that make it easier to develop, train, and deploy machine learning models. One of the main differences between TensorFlow and PyTorch is the way they handle computation. TensorFlow uses a static computational graph, where the model is defined and compiled before training, and the computations are performed on the graph. PyTorch, on the other hand, uses a dynamic computational graph, where the model is defined and compiled on the fly, and the computations are performed on the fly as well. This makes PyTorch more flexible and easier to use for some types of models and tasks. Another difference between TensorFlow and PyTorch is the way they handle data. TensorFlow uses a dataflow-based programming model, where the data is transformed and passed through the computational graph in a series of steps. PyTorch, on the other hand, uses a tensor-based programming model, where the data is represented as a set of tensors, which are manipulated directly by the model. This makes PyTorch more intuitive and easier to use for some types of data. Overall, TensorFlow and PyTorch are both powerful and popular machine learning frameworks, and they each have their own strengths and weaknesses. Which framework is best for a particular task will depend on the specific requirements and goals of the project.

4.6.2 Operational Software Development

The following section details the development of additional software tools that were required in the training, evaluation, and inference of the various VAE derivative models.

Data Loader

A machine learning data loader is a software component that is used to load and prepare the input data for a machine learning model. The data loader is typically responsible for reading the input data from a storage device, such as a disk or a database, and then transforming and organizing the data in a way that is suitable for training the model. A machine learning data loader typically performs a number of tasks, including:

- Reading the input data from a storage device and loading it into memory

- Transforming the input data, such as by scaling or normalizing the data, or applying transformations to extract features and patterns from the data
- Organizing the input data into batches, which are small subsets of the data that are used to train the model
- Shuffling the data, which is the process of randomly rearranging the data to prevent the model from overfitting to any particular patterns in the data

A machine learning data loader is an important component of a machine learning pipeline, and it can have a significant impact on the performance and generalization of the model. Properly designing and implementing the data loader can help ensure that the model is trained on clean, high-quality data, and can improve its ability to make accurate predictions on new data.

Model Evaluation

It is crucial to choose appropriate evaluation processes to assess how successful the VAE and other model derivatives are for data generation and secondly signal classification. The evaluation strategy involves the selection on various metrics, on top of the loss metrics, related to distribution and correlation comparison between the generated and the MC data. The selected metrics for the data generation evaluation can be seen below.

Evaluation Metrics		
Evaluation Metric	Evaluation Type	Brief Description
Kolmogorov-Smirnov	Distribution	A non-parametric test of the equality of continuous or discontinuous one-dimensional probability distributions that can be used to compare a sample with a reference probability distribution.
Scaled Difference	Distribution	(Real data binned frequency for each bin - Generated data binned frequency for each bin)/(Real data binned frequency for each bin)^2
Absolute Correlation Difference	Correlation	Absolute difference between the correlation matrices of the MC data and the generated data
Pearson Correlation	Correlation	A test statistics that measures the statistical association between two variables.
Pearson p value	Correlation	Test statistic indicating the probability that you would have found the current result if the correlation coefficient were in fact zero (null hypothesis).

TABLE 4.4: Table showing selected evaluation metrics.

In addition to the evaluation metrics, distribution plots and correlation comparison plots are used to compare the generated distributions and variable correlations against the MC distributions and variable correlations.

Model saving, performance logging and training continuation

In order to undertake the model evaluation, it is crucial to have appropriate model saving, metric logging and model training continuation code. TensorBoard was utilised to log loss function values and chosen performance metrics for each epoch of each run. The values of these logged metrics and loss values can then be viewed in the Tensorboard dashboard. An example of what the Tensorboard dashboard looks like is shown in Figure A.1 in Appendix A. Additionally, every 20 epochs, the PyTroch library's *.save* functionality was utilised to save model state dictionaries, loss function values, metric values, optimizer state dictionaries, run hyper-parameters and the epoch number. This saved file was then utilised for later inference of the model as the required point where convergence was achieved and for the continuation of training should the training have been interrupted.

GPU Usage

The use of a GPU in order to train the VAE models was essential in order to complete this research in a reasonable time frame. GPU resources on Google Colaboratory were harnessed to reduce training time. A Google Colab pro account was acquired in order to be able to have a powerful enough GPU and remove some time limitations on the usage of the GPU. The Google Colab computational resources available with the pro subscription are up to 32Gb of RAM and a Nvidia Tesla P100 or a Nvidia V100 Tensor Core GPU depending on the availability of resources, however, the resource availability is subject to the demand created by all Google Colab subscribers and might be cut off due to overload.

Code Structure Framework Design

Figure 4.10 shows the structural design of the overarching python classes developed and the connections between them.

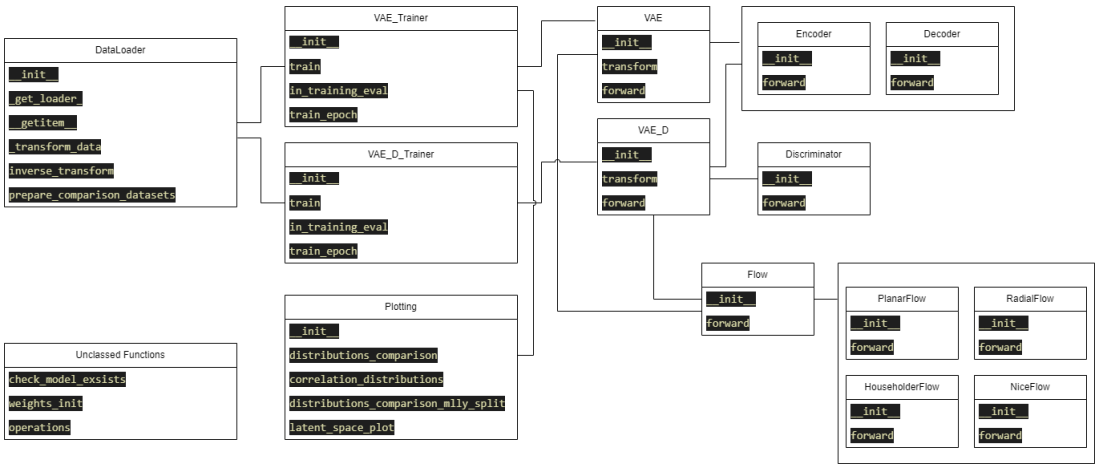


FIGURE 4.10: Diagram showing developed classes and functions.

Chapter 5

Results

The following section will consider the results of the data generation of the $Z\gamma$ final state background data, shown in the form of feature distribution and correlation graphs. The following section shows the results for each chosen VAE model derivative for the data generation task. For each model type, the best model hyper-parameters were chosen on the basis of the hyper-parameter optimisation loop that was completed. The use of different VAE derivatives with different training mechanisms and latent space properties helped in determining the most appropriate VAE derivative for generating the $Z\gamma$ final state dataset. The VAE showed decent generation results in terms of the distribution comparison plots for the variables that were normally distributed, or the right skewed variable distributions. The VAE seemingly struggled with generating variables with uniform distributions, likely due to the fact that its latent space is Gaussian, making it challenging to transform Gaussian variables to uniform distributions. The VAE+NF showed somewhat of an improvement in the distribution results, however, the generated distributions for the uniformly distributed variables had multi-modal spikes which varied from the $Z\gamma$ final state dataset. The VAE+NF+D derivative was found to be the best overall performer, as it allowed for a more flexible and complex latent space through the addition of the NF model and improved training convergence with the discriminator network. The hyper-parameter optimised VAE proved to be a decent base model for the research. In terms of the correlation comparison, the VAE+NF+D far out performed the other models in terms of the max correlation difference with an achieved value of 0.12, with the others in the range $[0.2 - 0.5]$.

TABLE 5.1: Comparison of models using evaluation metrics.

Metric	VAE	VAE+D	VAE+NF	VAE+NF+D
Relative Difference	0.010701	0.004422	0.009670	0.003230
Kolmogorov-Smirnov score	0.067512	0.051637	0.055487	0.034702
Spearman correlation coefficient	0.333721	0.557210	0.623028	0.674031
Correlation difference max	0.511256	0.072731	0.026247	0.012071

5.0.1 VAE

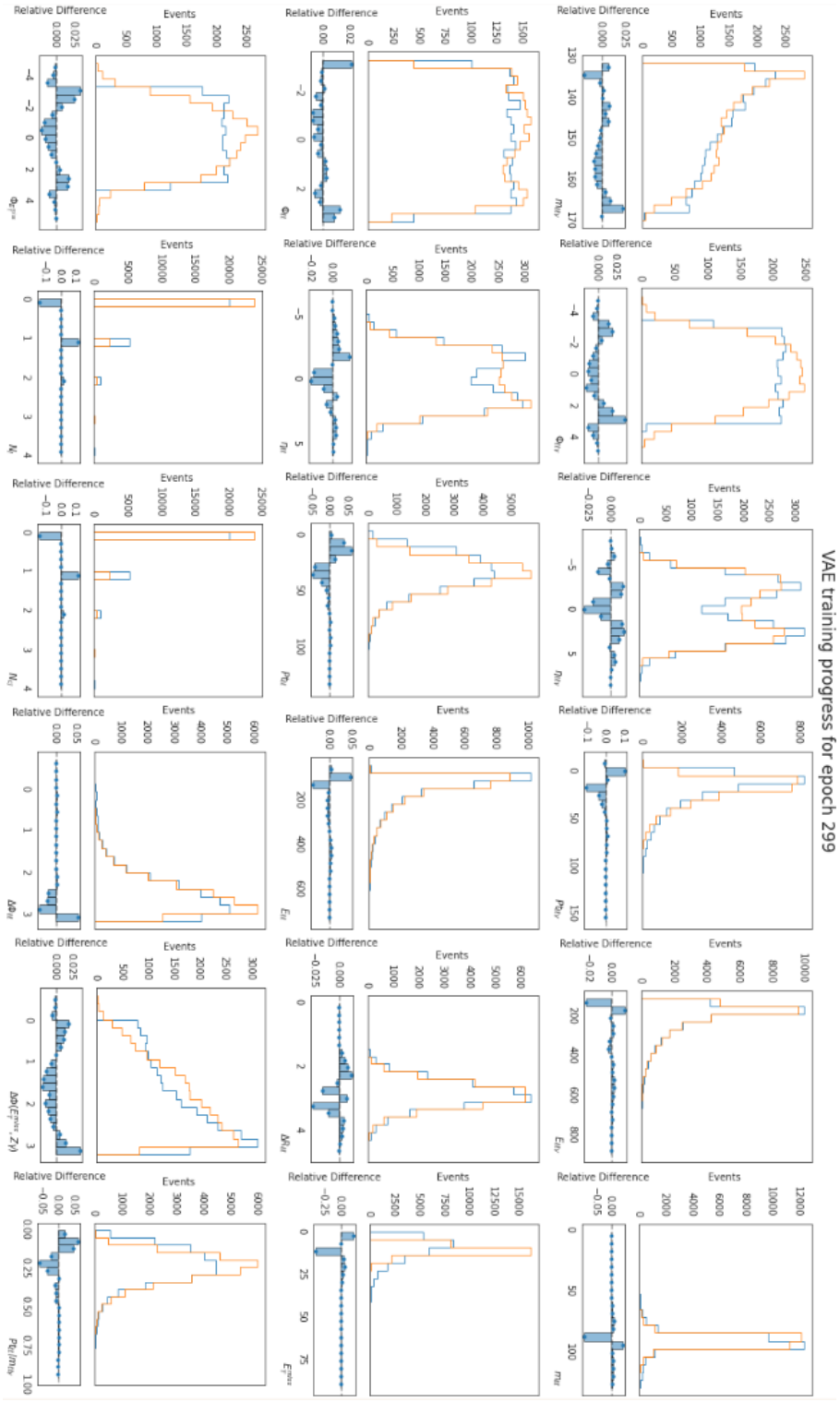


FIGURE 5.1: Distribution Plots of the Generated Kinematic Features Against MC Kinematic Feature Data

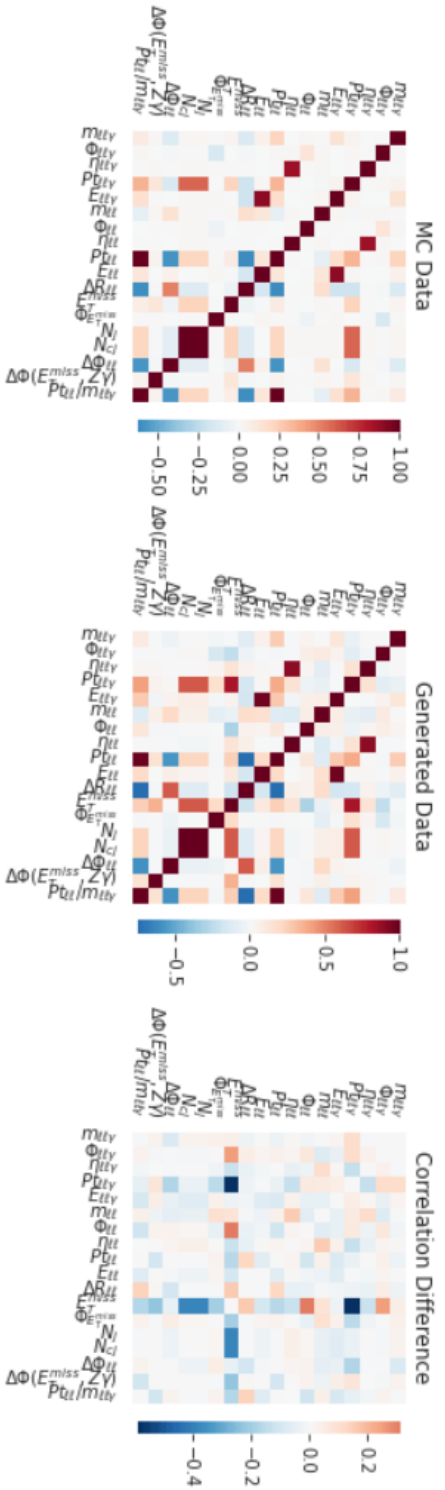


FIGURE 5.2: Correlation Plots of the Generated Kinematic Features Against MC Kinematic Feature Data

5.0.2 VAE+D

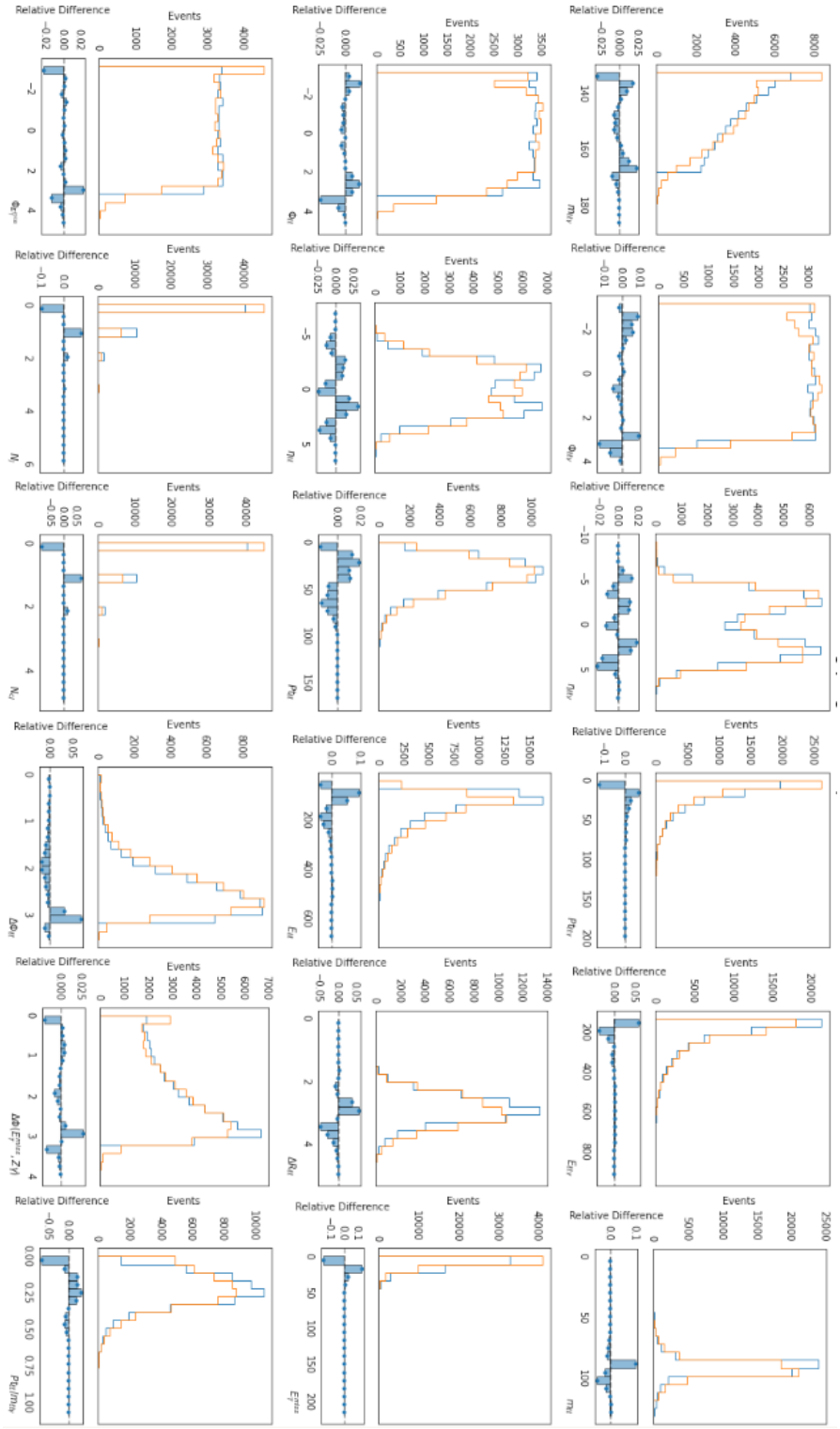


FIGURE 5.3: Distribution Plots of the Generated Kinematic Features Against MC Kinematic Feature Data

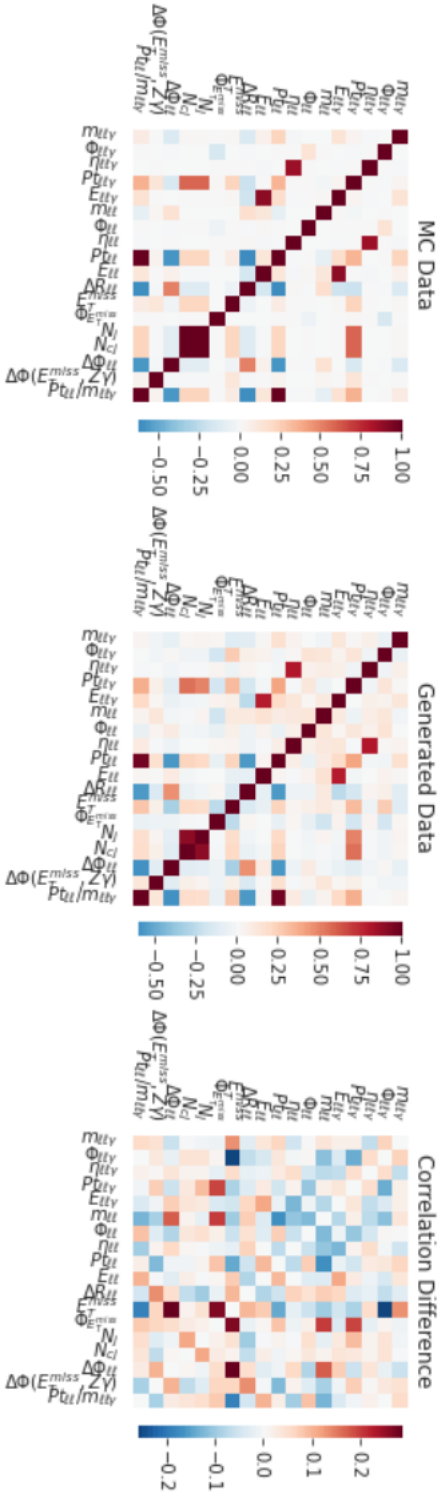


FIGURE 5.4: Correlation Plots of the Generated Kinematic Features Against MC Kinematic Feature Data

5.0.3 VAE+NF

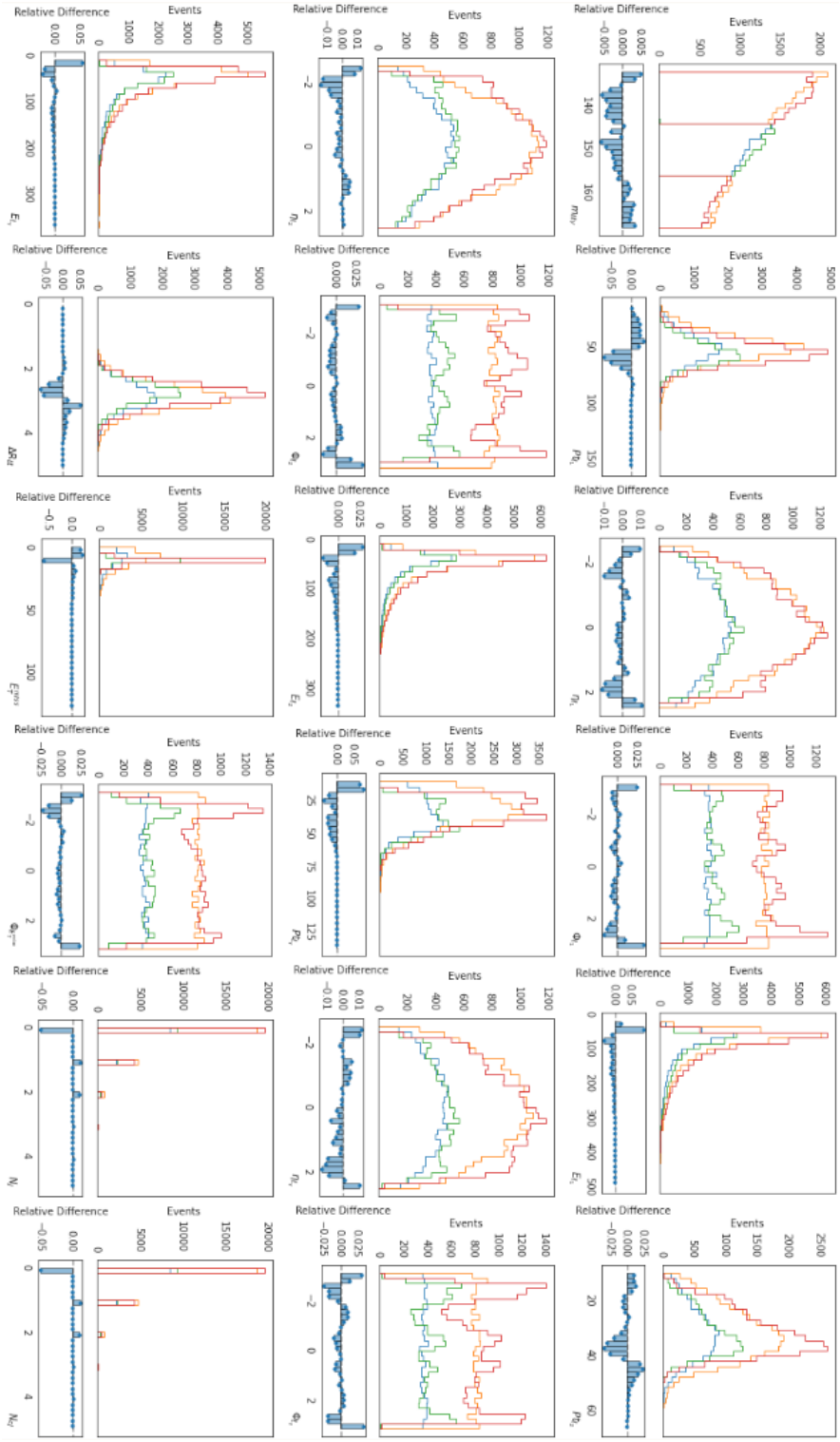


FIGURE 5.5: Distribution Plots of the Generated Kinematic Features Against MC Kinematic Feature Data

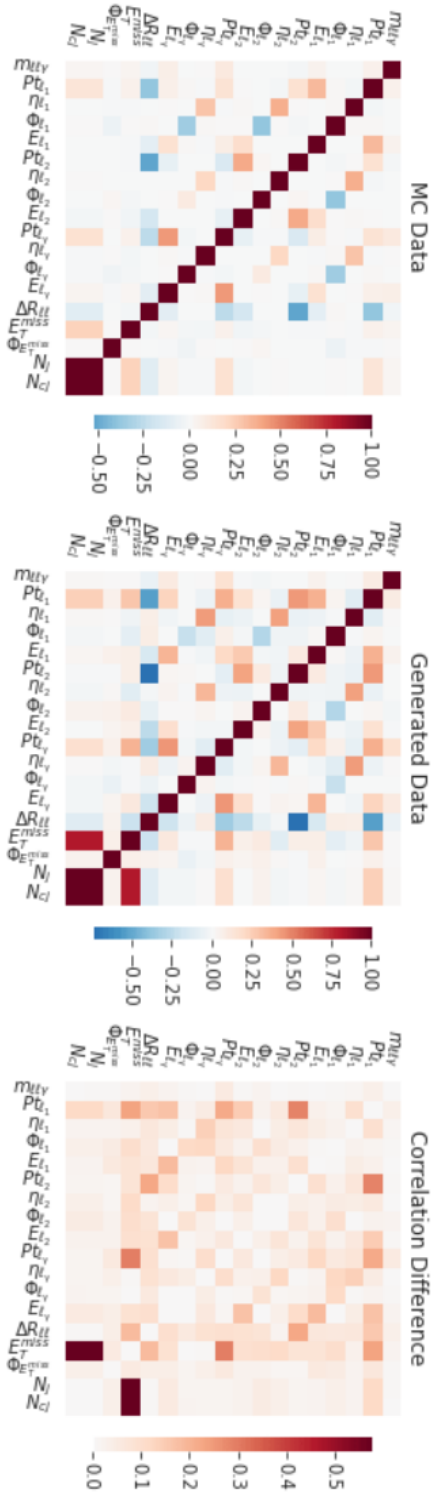


FIGURE 5.6: Correlation Plots of the Generated Kinematic Features Against MC Kinematic Feature Data

5.0.4 VAE+NF+D

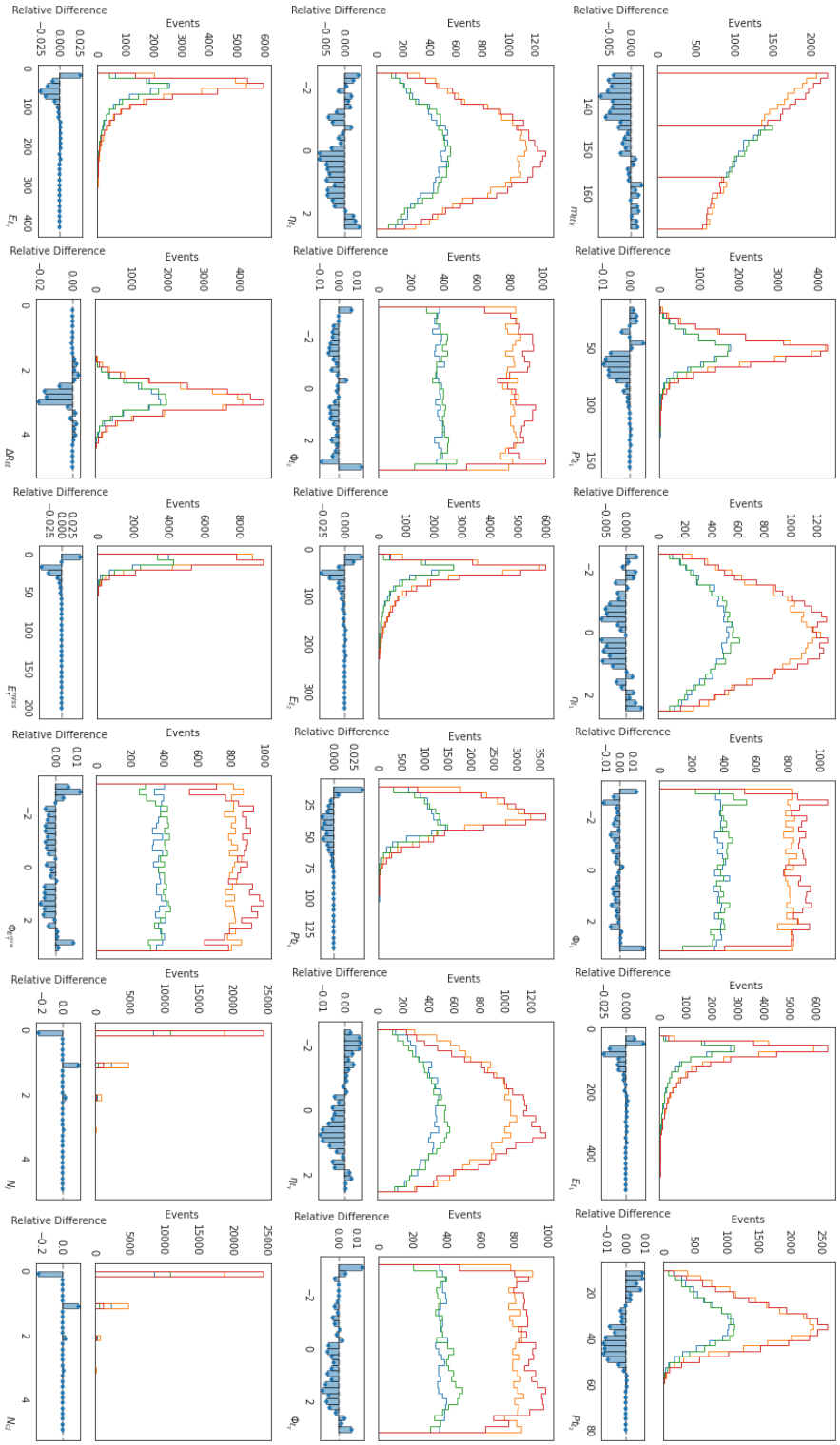


FIGURE 5.7: Distribution Plots of the Generated Kinematic Features Against MC Kinematic Feature Data

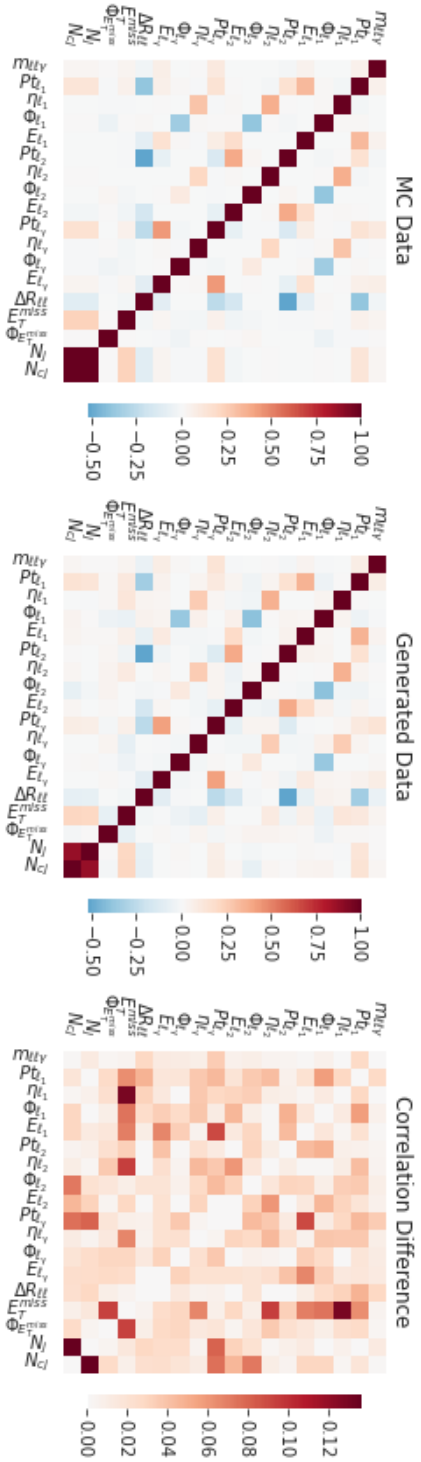


FIGURE 5.8: Correlation Plots of the Generated Kinematic Features Against MC Kinematic Feature Data

Chapter 6

Applications beyond Particle Physics

6.1 Applications beyond Particle Physics

The following section outlines the methodology of research outside of the particle physics realm that resulted from the expertise gained, in combating COVID-19 in South Africa.

6.1.1 COVID-19 Management in South Africa Resulting Applications

This section of the dissertation documents the work carried out using machine learning to developed an early alert system for an additional wave of COVID-19 cases in South Africa for the Gauteng government, full detail of this work can be found in the published paper [1]. A neural network-based early alert/detection system was developed to predict the daily confirmed cases of COVID-19 in South Africa by utilizing Big Data and Artificial Intelligence. The system is based on an LSTM RNN model that is trained on mobility indices, stringency indices, and epidemiological parameters for all provinces in South Africa during the period between the first and second waves of COVID-19 cases. The chosen prediction parameter for the model was the daily change in cases, $dTCt$. While the model was able to accurately predict daily cases during the interim period, it struggled to recognize the behavior of the features in relation to the prediction parameter $dTCt$ during a COVID-19 case peak. This weakness was intentionally built into the model to create an early detection system for additional waves of COVID-19 cases. When the relative difference between the actual and predicted daily cases exceeds a province-specific threshold value, known as the Risk Index Metric (RIM), a warning is issued to notify the government and the public. The model was calibrated using data from the first and second waves of COVID-19 cases in South Africa and was verified using data from the second wave. The model was then used to monitor the third wave in South Africa, with the aim of developing a functional alert system for additional waves of COVID-19 in specific regions. The main constraints for this early detection algorithm are related to the availability of mobility and epidemiological data, with reports from Google and Facebook only being updated on Sundays with

a week's worth of data. Figure 6.1 shows the total period used for training the model between the first and second peaks for the Gauteng province.

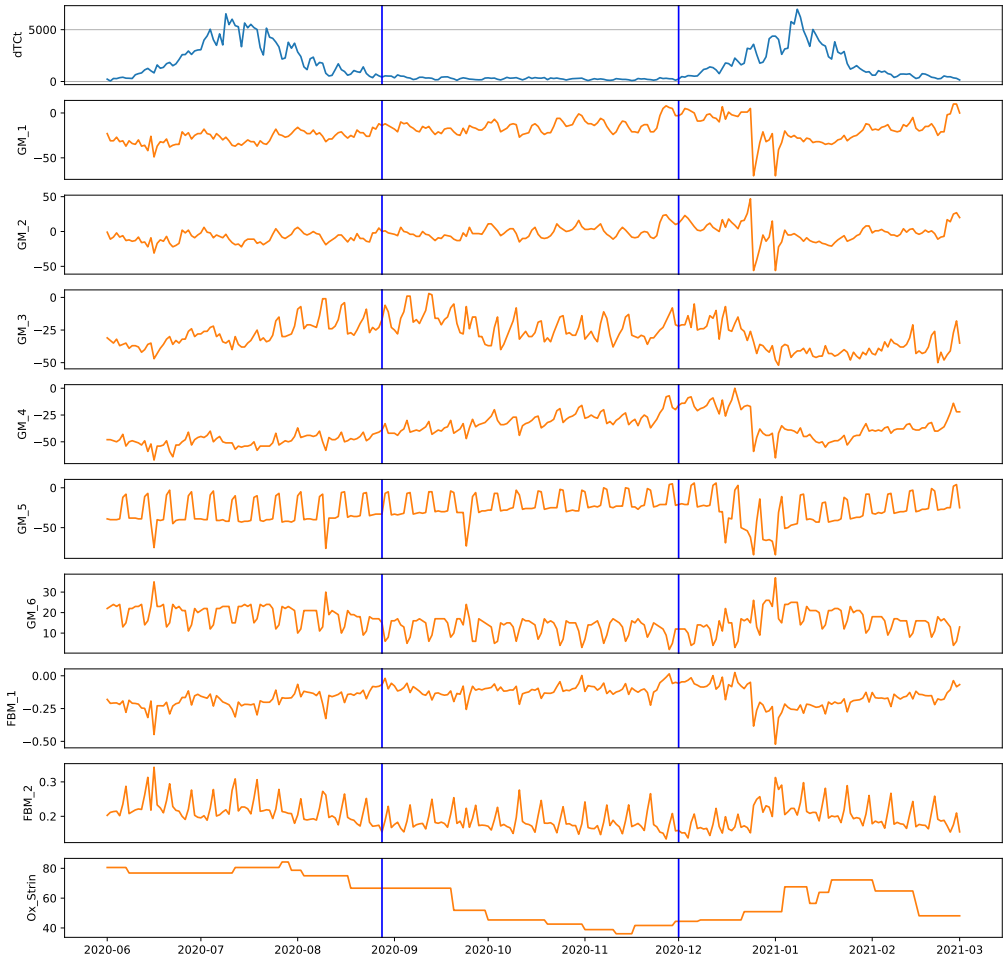


FIGURE 6.1: Appropriate test-train period for Gauteng Province, South Africa.

The output of the trained RNN model with LSTM is a 14-day prediction of new daily cases, $dTCt$. The first day of this prediction is a Monday, which is 6 days earlier than the actual date that the model is run due to data availability constraints from external sources. As a result, the prediction only covers the next 7 days from the date it is generated. The model also provides a secondary output on a daily basis, which is the relative difference between the prediction and the actual recorded value. This relative difference is calculated using the formula:

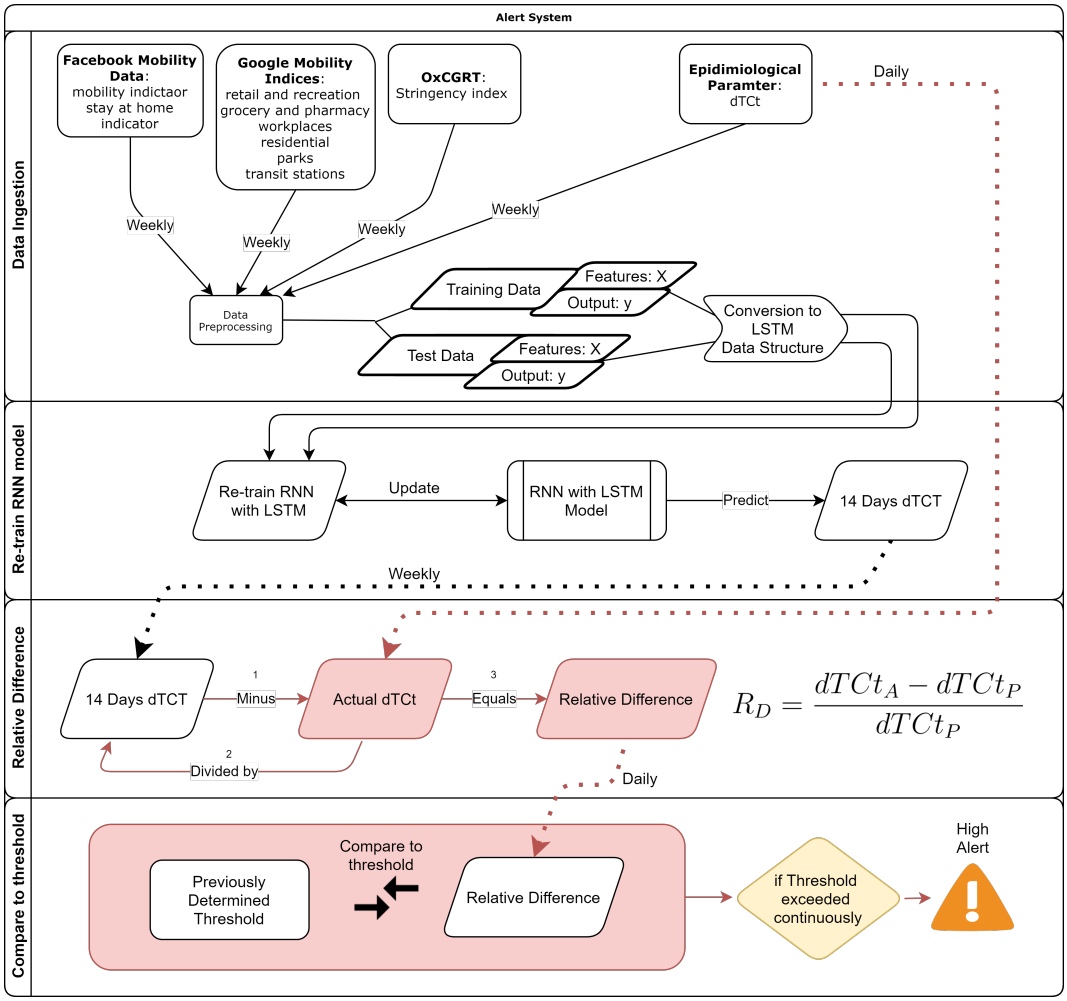


FIGURE 6.2: AI Early Alert System Flow Diagram.

$$R_D = \frac{(dTCt_A - dTCt_P)}{dTCt_P}, \tag{6.1}$$

where $dTCt_A$ is the daily change in actual total cases and $dTCt_P$ is the daily change in predicted total cases. The relative difference is used as the Risk Index Metric (RIM) for an alert of an additional wave of COVID-19 cases. The model is recalibrated and a new 14-day $dTCt$ prediction is generated every Sunday, covering the dates from the Monday before to the Monday after. The data from the second wave of COVID-19 cases was excluded from the training data for the model. It is important to note that the alert system functioned as a surveillance system for the third wave of COVID-19 in South Africa, aiding in the management of the pandemic in the country. A flow diagram of the functioning of the AI powered early alert system developed can be seen in Figure 6.2.

6.1.2 Example Prediction Result During Non-peak Period

Figure 6.3 shows the final LSTM RNN model's ability to predict $dTCt$ during non-peak times.

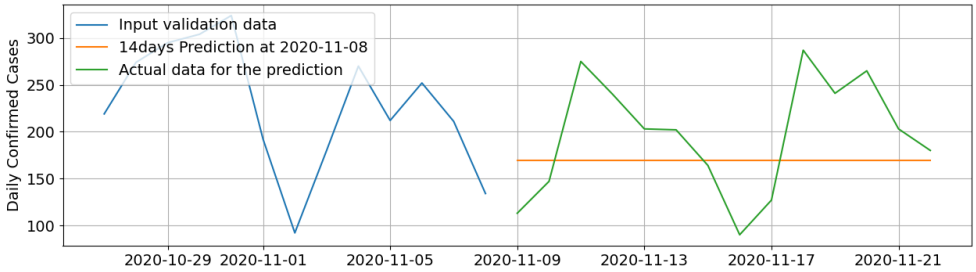


FIGURE 6.3: Graph showing 14-day prediction of $dTCt$ during a non-peak period.

6.1.3 Verification of the Alert System Using Second Wave Data

Using the appropriate threshold discovered by comparing non-peak and peak RIM distributions for each province in South Africa, the alert system was tested by comparing the system's predicted start date of the second wave against the actual case data. For each province, the model alerted the start of the additional wave at an appropriate date.

TABLE 6.1: Province Specific Second Wave Start Date.

Province	2^{nd} **Second Wave Start Date**
Gauteng	2020-12-07
Western Cape	2020-11-11
Eastern Cape	2020-10-21
KwaZulu-Natal	2020-12-01
Free State	2020-12-19
Mpumalanga	2020-12-15
Limpopo	2020-12-01
North West	2020-12-23
Northern Cape	2020-12-23

Figure 6.4 shows all the last halves (last week of the 2 week prediction period) of the R_D values generated from each week's prediction joined for a date period that extends into the second wave period for Gauteng province. The blue line indicates the date when the model identified a wave starting for Gauteng.

6.1.4 Third Wave Surveillance

Below is a screenshot of the final output of the model for third-wave surveillance and was available on a COVID-19 monitoring website. The model

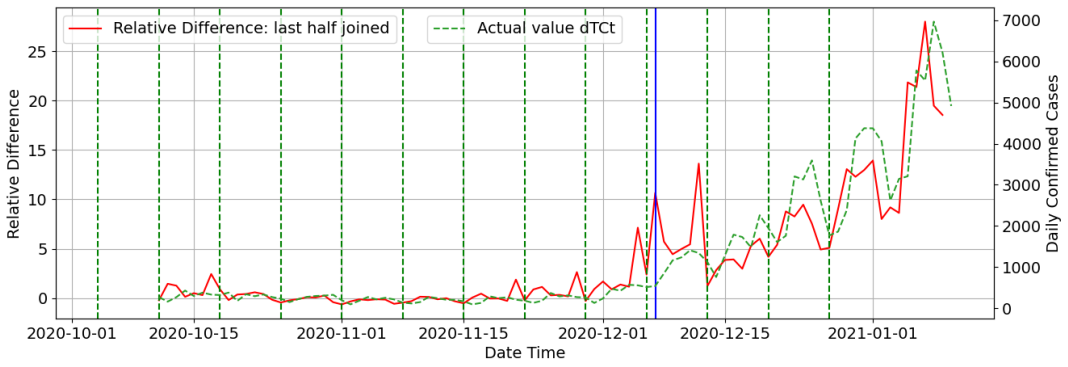


FIGURE 6.4: Graph showing all last halves of R_D values joined: Gauteng.

shown on the site was updated weekly and the relative difference value was calculated automatically as new daily case data became available (Fig. 6.5). Notably, at the time of writing, the system was successful in detecting the beginning of the 3rd wave in the provinces in South Africa. During the COVID-19 pandemic South African policymakers engaged with the created RIM on a weekly basis during the Gauteng government COVID-19 command council meetings.

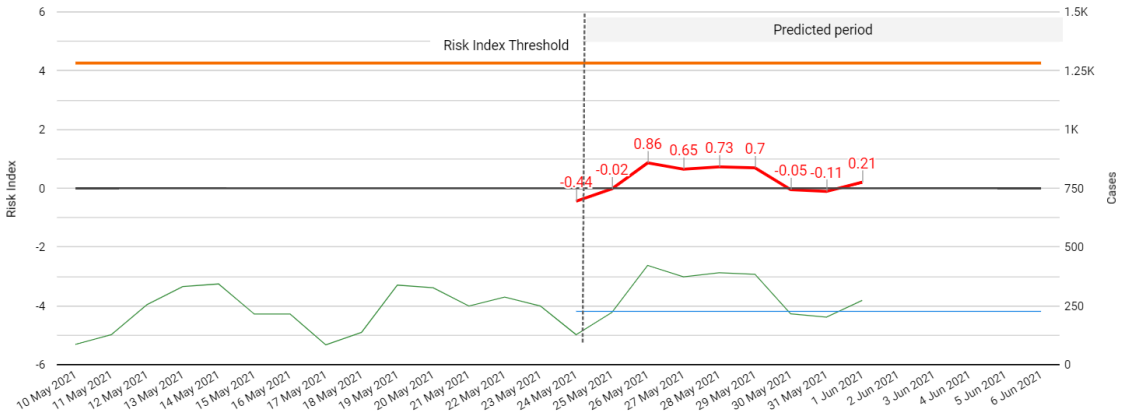


FIGURE 6.5: Graph showing the 14-day prediction of *dTCt* and *rd* values for surveillance of the 3rd wave.

6.1.5 Resulting Applications - COVID-19: Discussion and Conclusions

This research exploits the multivariate, multi-time-step time-series predictive capabilities of an RNN with LSTM to predict daily change in cases *dTCt* in South African provinces. Ten features were chosen as inputs to the RNN model. These features include mobility measures, stringency indicators and epidemiological parameters. The model was trained over the interim period between COVID-19 case waves within each province. This configuration caused the model to perform well over the interim period, however, when

another COVID-19 case wave is reached, the system is unable to predict the $dTCt$ values accurately. The intentional pitfall of the model to predict $dTCt$ during a peak has been taken advantage of to create an alert system by monitoring the relative difference R_D between the prediction and the actual value on a daily basis. When the R_D value is consistently above a calculated threshold for at least 2 days, the probability of an additional wave is high. The thresholds for each province are calculated by analysing the distributions of R_D values generated as a result of the predictions overtime during peak and non-peak times. The threshold was chosen by selecting an R_D value that encapsulated the whole non-peak distribution. This work demonstrates that incorporating Google-outputted mobility data resulted in a significantly higher predictive power of COVID-19 cases. Using this methodology, the dates of the start of the second wave of COVID-19 cases in South African provinces were accurately estimated. Noteworthy, the dates generated by the model would not have been able to be achieved confidently by simply monitoring the daily change in cases only. Furthermore, the model has been successful in identifying the start of the third wave of COVID-19 cases in South African provinces and has proved a valuable tool to South African policymakers.

Chapter 7

Discussion and Conclusion

7.1 Data Generation Discussion and Conclusions

The VAE architecture performed well as a base generative architecture. Although the training of VAEs can be temperamental, with appropriate hyper-parameter optimisation it is possible to achieve the desired model training convergence. The incorporation of a number of VAE model derivatives that incorporate different training mechanisms and latent space properties proved to be a useful strategy in assessing what VAE derivative is most appropriate for the generation of the $Z\gamma$ final state dataset. The best overall performing VAE derivative was found to be the VAE+NF+D. The addition of the NF model allowed for a more flexible and complex latent space to be achieved and the addition of the discriminator network, with optimised hyper-parameters, improved the training convergence specifically of the VAE component, less so of the discriminator network. It should be noted that the VAE base model and VAE derivatives struggled the most with the generation of the variables with uniform distributions. This is could be due to the fact that the latent space of a VAE is Gaussian and the transformation from a variable that is Gaussian to a uniform distribution through the decoder layers is more challenging to learn than a transformation from Gaussian to a non-uniformly distributed variable.

There were a number of model configurations and hyper-parameter choices that resulted in significant improvement in the results in terms of the data generation task.

1. The value of the β_{var} in the standard VAE based models (VAE, VAE+NF), and similarly the value of the γ parameter in the VAE+D models (VAE+D, VAE+NF+D). Both these parameters are related to the weighting of each of the loss terms against each other that make up the overall loss function. The optimisation of these parameters was somewhat more simple than that of other parameters such as number of nodes in hidden layers because of the large effect that the value had on the data generation results. It was trivial to conclude that the best value of the β_{var} parameter was around the magnitude of 0.0001 and the best value for the γ parameter was around 500.

2. The addition of the batch normalisation layers in between hidden layers of the constituent networks has a significant in the time it took the

model to get to reasonably close to the final convergence value (i.e. the loss function curves dropped quicker initially).

3. The use of a sigmoid activation function at the end of the deocder instead of a relu activation had a large effect on the results in terms of the distribution. With the use of the relu function, there was often a peak at the beginning of the distribution as a result of the relu function always being 0 if the value before the activation is less than 0.

4. The chosen learning rate also as expected had a significant effect on the overall convergence of the model. This is due to the fact that choosing a learning rate too large would cause the optimiser to 'jump over' the desired minima in the gradient function and choosing a learning rate too small would cause the optimizer to settle on a local minima in the gradient function that is not actually the desired minima.

5. There were a number of architectural configurations that also caused great improvements in the results. The choice of a architecture for the encoder that expands from the 18 feature input dimension to a large number of nodes in the hidden layers (256/512) and then dropped back down to a latent dimensionality lower or equal to the input dimension was found to be optimal for this generation task.

6. The use of the MinMax data transformer before the data is inputted to the model improved the results drastically. The MinMax scalar is used to scale the data to a range between 0-1.

Although significant progress was made to achieve the best VAE-based data generative models, which produce sufficient generated data in terms of the comparison plots and the chosen metrics, the models will need to be further evaluated by using the generated data to train a semi-supervised classifier and a frequentist study completed.

Additionally, in terms of the resulting applications of machine learning in COVID-19 crisis management for the Gauteng government, this work demonstrated the successful use of a recurrent neural network (RNN) with long short-term memory (LSTM) to develop an early alert system for an additional COVDI-19 case waves in South African provinces. This work demonstrates that incorporating Google-outputted mobility data as input features to the model, resulted in a significantly higher predictive power of COVID-19 cases. The resulting model has been successful in identifying the start of the third wave of COVID-19 cases in South African provinces and proved a valuable tool to South African policymakers.

Appendix A

Additional Figures

This appendix contains additional figures related to the training configuration and metric logging.

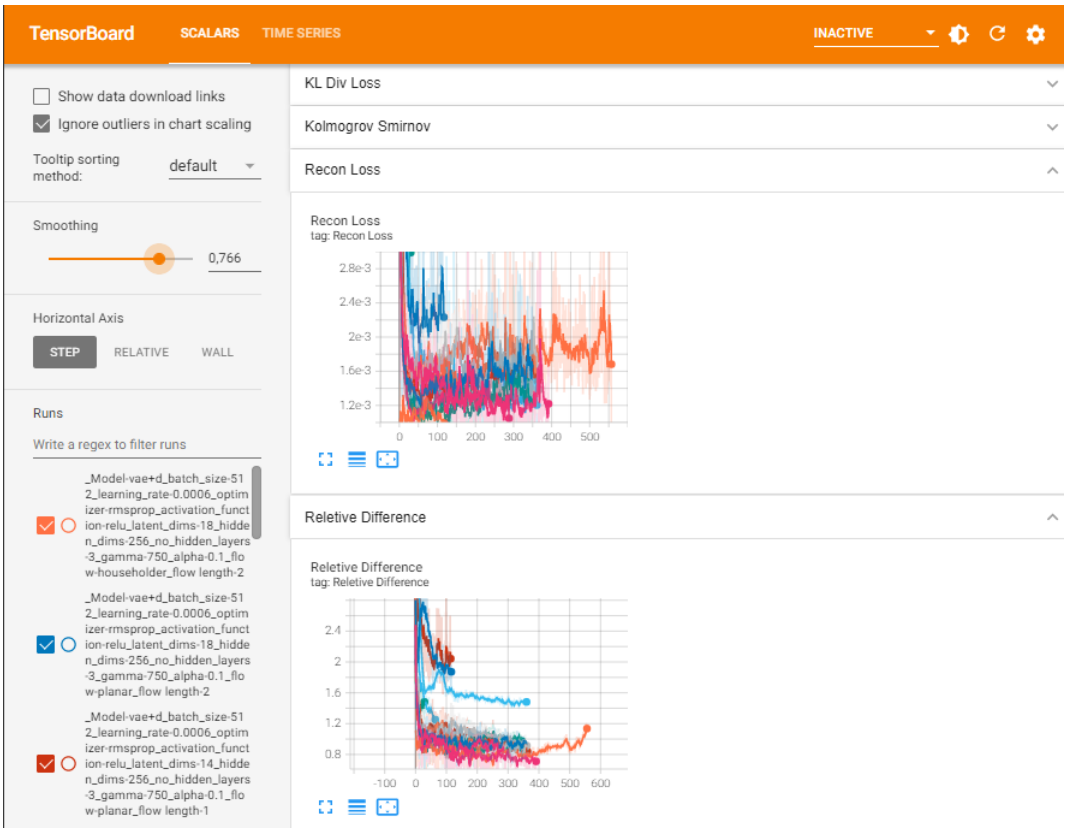


FIGURE A.1: Snapshot of Tensorboard Dashboard.

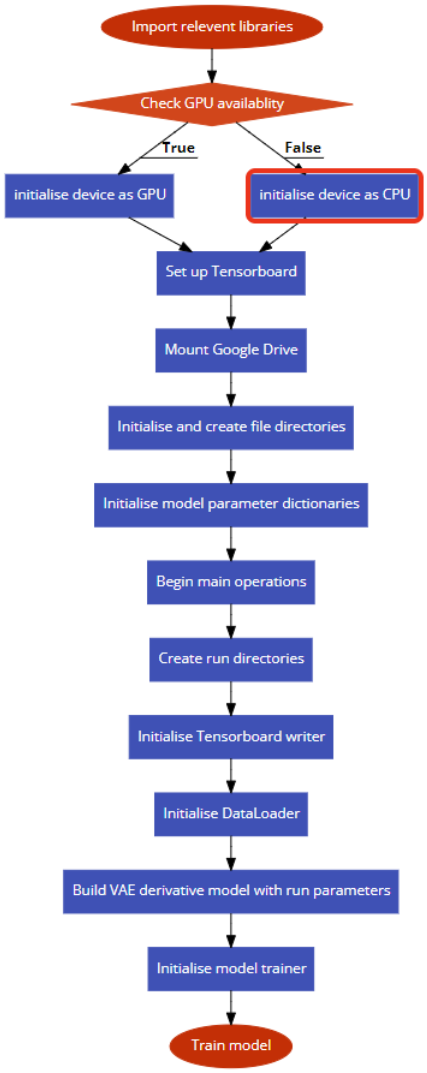


FIGURE A.2: Flow diagram of steps to initiate model training.

Bibliography

[1] F. Stevenson, K. Hayasi, N. L. Bragazzi, *et al.*, "Development of an early alert system for an additional wave of covid-19 cases using a recurrent neural network with long short-term memory," *International Journal of Environmental Research and Public Health*, vol. 18, no. 14, 2021, ISSN: 1660-4601. DOI: 10.3390/ijerph18147376. [Online]. Available: https://www.mdpi.com/1660-4601/18/14/7376.

[2] M. Alavinejad, B. Mellado, A. Asgary, *et al.*, "Management of hospital beds and ventilators in the gauteng province, south africa, during the covid-19 pandemic," *PLOS Global Public Health*, vol. 2, no. 11, pp. 1–20, Nov. 2022. DOI: 10.1371/journal.pgph.0001113. [Online]. Available: https://doi.org/10.1371/journal.pgph.0001113.

[3] B. Mellado, J. Wu, J. D. Kong, *et al.*, "Leveraging artificial intelligence and big data to optimize covid-19 clinical public health and vaccination roll-out strategies in africa," *International Journal of Environmental Research and Public Health*, vol. 18, no. 15, 2021, ISSN: 1660-4601. DOI: 10.3390/ijerph18157890. [Online]. Available: https://www.mdpi.com/1660-4601/18/15/7890.

[4] B. Lieberman, J. D. Kong, R. Gusinow, *et al.*, "Big data- and artificial intelligence-based hot-spot analysis of COVID-19: Gauteng, South Africa, as a case study," *BMC Med. Inf. Decis. Making*, vol. 23, no. 1, pp. 1–15, Dec. 2023, ISSN: 1472-6947. DOI: 10.1186/s12911-023-02098-3.

[5] F. Stevenson, *Evaluation and Optimisation of a Generative-Classification Hybrid Variational Autoencoder in the Search for Resonances at the LHC in The Proceedings of SAIP2022, the 66th Annual Conference of the South African Institute of Physics*, en, P. A. Prinsloo, Ed. SAIP2022, 2022, ISBN: 978-0-6397-4426-1.

[6] J. Cogan, M. Kagan, E. Strauss, and A. Schwartzman, "Jet-images: Computer vision inspired techniques for jet tagging," *Journal of High Energy Physics*, vol. 2015, no. 2, 2015. DOI: 10.1007/jhep02(2015)118. [Online]. Available: https://doi.org/10.1007%2Fjhep02%282015%29118.

[7] "Quark versus Gluon Jet Tagging Using Jet Images with the ATLAS Detector," CERN, Geneva, Tech. Rep., 2017. [Online]. Available: http://cds.cern.ch/record/2275641.

[8] G. Kasieczka, T. Plehn, M. Russell, and T. Schell, "Deep-learning top taggers or the end of QCD?" *Journal of High Energy Physics*, vol. 2017, no. 5, 2017. DOI: 10.1007/jhep05(2017)006. [Online]. Available: https://doi.org/10.1007%2Fjhep05%282017%29006.

[9] P. T. Komiske, E. M. Metodiev, B. Nachman, and M. D. Schwartz, "Pileup mitigation with machine learning (PUMML)," *Journal of High Energy Physics*, vol. 2017, no. 12, 2017. DOI: 10.1007/jhep12(2017)051. [Online]. Available: https://doi.org/10.1007%2Fjhep12%282017%29051.

[10] "Convolutional Neural Networks with Event Images for Pileup Mitigation with the ATLAS Detector," CERN, Geneva, Tech. Rep., 2019. [Online]. Available: http://cds.cern.ch/record/2684070.

[11] J. A. Martinez, O. Cerri, M. Pierini, M. Spiropulu, and J.-R. Vlimant, *Pileup mitigation at the large hadron collider with graph neural networks*, 2018. DOI: 10.48550/ARXIV.1810.07988. [Online]. Available: https://arxiv.org/abs/1810.07988.

[12] R. T. D'Agnolo and A. Wulzer, "Learning new physics from a machine," *Physical Review D*, vol. 99, no. 1, 2019. DOI: 10.1103/physrevd.99.015014. [Online]. Available: https://doi.org/10.1103%2Fphysrevd.99.015014.

[13] J. H. Collins, K. Howe, and B. Nachman, "Extending the search for new resonances with machine learning," *Physical Review D*, vol. 99, no. 1, 2019. DOI: 10.1103/physrevd.99.014038. [Online]. Available: https://doi.org/10.1103%2Fphysrevd.99.014038.

[14] O. Cerri, T. Q. Nguyen, M. Pierini, M. Spiropulu, and J.-R. Vlimant, "Variational autoencoders for new physics mining at the large hadron collider," *Journal of High Energy Physics*, vol. 2019, no. 5, 2019. DOI: 10.1007/jhep05(2019)036. [Online]. Available: https://doi.org/10.1007%2Fjhep05%282019%29036.

[15] J. A. Martí nez, T. Q. Nguyen, M. Pierini, M. Spiropulu, and J.-R. Vlimant, "Particle generative adversarial networks for full-event simulation at the LHC and their application to pileup description," *Journal of Physics: Conference Series*, vol. 1525, no. 1, p. 012 081, 2020. DOI: 10.1088/1742-6596/1525/1/012081. [Online]. Available: https://doi.org/10.1088%2F1742-6596%2F1525%2F1%2F012081.

[16] Vallecorsa, Sofia, Carminati, Federico, and Khattak, Gulrukh, "3d convolutional gan for fast simulation," *EPJ Web Conf.*, vol. 214, p. 02 010, 2019. DOI: 10.1051/epjconf/201921402010. [Online]. Available: https://doi.org/10.1051/epjconf/201921402010.

[17] "Deep generative models for fast shower simulation in ATLAS," CERN, Geneva, Tech. Rep., 2018. [Online]. Available: http://cds.cern.ch/record/2630433.

[18] G. Aad *et al.*, "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC," *Phys. Lett. B*, vol. 716, pp. 1–29, 2012. DOI: 10.1016/j.physletb.2012.08.020. arXiv: 1207.7214 [hep-ex].

[19] S. Chatrchyan *et al.*, "Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC," *Phys. Lett. B*, vol. 716, pp. 30–61, 2012. DOI: 10.1016/j.physletb.2012.08.021. arXiv: 1207.7235 [hep-ex].

[20] M. Thomson, *Modern particle physics*. New York: Cambridge University Press, 2013, ISBN: 978-1-107-03426-6.

[21] *Atlas experiment website*, 2022. [Online]. Available: https://atlas.cern/about.

[22] A. Butter, T. Plehn, and R. Winterhalder, "How to gan lhc events," vol. 7, no. 6, 2019. DOI: https://doi.org/10.21468/scipostphys.7.6.075. [Online]. Available: https://arxiv.org/abs/1907.03764.

[23] C. Bozzi, "LHCb Computing Resource usage in 2020," CERN, Geneva, Tech. Rep., 2021. [Online]. Available: https://cds.cern.ch/record/2752155.

[24] R. Wang, K. Nie, Y.-J. Chang, *et al.*, "Deep learning for anomaly detection," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, ser. KDD '20, Virtual Event, CA, USA: Association for Computing Machinery, 2020, 3569–3570, ISBN: 9781450379984. DOI: 10.1145/3394486.3406481. [Online]. Available: https://doi.org/10.1145/3394486.3406481.

[25] E. M. Metodiev, B. P. Nachman, and J. Thaler, "Classification without labels: Learning from mixed samples in high energy physics," vol. 2017, no. 10, 2017. DOI: https://doi.org/10.1007/jhep10(2017)174. [Online]. Available: https://arxiv.org/abs/1708.02949.

[26] S. von Buddenbrock, N. Chakrabarty, A. S. Cornell, *et al.*, "The compatibility of LHC Run 1 data with a heavy scalar of mass around 270\,GeV," Jun. 2015. arXiv: 1506.00612 [hep-ph].

[27] S. von Buddenbrock, N. Chakrabarty, A. S. Cornell, *et al.*, "Phenomenological signatures of additional scalar bosons at the lhc," *Eur. Phys. J. C*, vol. 76, no. 10, p. 580, 2016. DOI: 10.1140/epjc/s10052-016-4435-8. arXiv: 1606.01674 [hep-ph].

[28] M. Herrero and R. A. Morales, "Anatomy of higgs boson decays into and z within the electroweak chiral lagrangian in the r gauges," vol. 102, no. 7, 2020. DOI: https://doi.org/10.1103/physrevd.102.075040.

[29] S. von Buddenbrock, A. S. Cornell, A. Fadol, M. Kumar, B. Mellado, and X. Ruan, "Multi-lepton signatures of additional scalar bosons beyond the Standard Model at the LHC," *J. Phys. G*, vol. 45, no. 11, p. 115 003, 2018. DOI: 10.1088/1361-6471/aae3d6. arXiv: 1711.07874 [hep-ph].

[30] S. Buddenbrock, A. S. Cornell, Y. Fang, *et al.*, "The emergence of multi-lepton anomalies at the LHC and their compatibility with new physics at the EW scale," *JHEP*, vol. 10, p. 157, 2019. DOI: 10.1007/JHEP10(2019)157. arXiv: 1901.05300 [hep-ph].

[31] S. von Buddenbrock, R. Ruiz, and B. Mellado, "Anatomy of inclusive $t\bar{t}W$ production at hadron colliders," *Phys. Lett. B*, vol. 811, p. 135 964, 2020. DOI: 10.1016/j.physletb.2020.135964. arXiv: 2009.00032 [hep-ph].

[32] Y. Hernandez, M. Kumar, A. S. Cornell, *et al.*, "The anomalous production of multi-lepton and its impact on the measurement of Wh production at the LHC," *Eur. Phys. J. C*, vol. 81, no. 4, p. 365, 2021. DOI: 10.1140/epjc/s10052-021-09137-1. arXiv: 1912.00699 [hep-ph].

[33] A. Crivellin, Y. Fang, O. Fischer, *et al.*, "Accumulating Evidence for the Associate Production of a Neutral Scalar with Mass around 151 GeV," Sep. 2021. arXiv: 2109.02650 [hep-ph].

[34] G. Beck, M. Kumar, E. Malwa, B. Mellado, and R. Temo, "Connecting multi-lepton anomalies at the LHC and in Astrophysics and the prospects of MeerKAT/SKA," Feb. 2021. arXiv: 2102.10596 [astro-ph.HE].

[35] D. Sabatta, A. S. Cornell, A. Goyal, M. Kumar, B. Mellado, and X. Ruan, "Connecting muon anomalous magnetic moment and multi-lepton anomalies at LHC," *Chin. Phys. C*, vol. 44, no. 6, p. 063 103, 2020. DOI: 10.1088/1674-1137/44/6/063103. arXiv: 1909.03969 [hep-ph].

[36] B. Abi *et al.*, "Measurement of the Positive Muon Anomalous Magnetic Moment to 0.46 ppm," *Phys. Rev. Lett.*, vol. 126, no. 14, p. 141 801, 2021. DOI: 10.1103/PhysRevLett.126.141801. arXiv: 2104.03281 [hep-ex].

[37] T. Aoyama *et al.*, "The anomalous magnetic moment of the muon in the Standard Model," *Phys. Rept.*, vol. 887, pp. 1–166, 2020. DOI: 10.1016/j.physrep.2020.07.006. arXiv: 2006.04822 [hep-ph].

[38] O. Fischer, B. Mellado, S. Antusch, *et al.*, "Unveiling hidden physics at the LHC," *Eur. Phys. J. C*, vol. 82, no. 8, pp. 1–58, Aug. 2022, ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-022-10541-4.

[39] S.-E. Dahbi, J. Choma, G. Mokgatitswane, *et al.*, "Machine learning approach for the search of resonances with topological features at the large hadron collider," *International Journal of Modern Physics A*, 2021, ISSN: 1793-656X. DOI: 10.1142/s0217751x21502419. [Online]. Available: http://dx.doi.org/10.1142/S0217751X21502419.

[40] B. Lieberman, J. Choma, S.-E. Dahbi, B. Mellado, and X. Ruan, "An investigation of over-training within semi-supervised machine learning models in the search for heavy resonances at the LHC," in *65th Annual Conference of the South African Institute of Physics*, Sep. 2021. arXiv: 2109.07287 [hep-ex].

[41] E. Gross and O. Vitells, "Trial factors for the look elsewhere effect in high energy physics," *The European Physical Journal C*, vol. 70, no. 1-2, pp. 525–530, 2010. DOI: 10.1140/epjc/s10052-010-1470-8. [Online]. Available: https://doi.org/10.1140%2Fepjc%2Fs10052-010-1470-8.

[42] J. Alwall, R. Frederix, S. Frixione, *et al.*, "The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations," *JHEP*, vol. 07, p. 079, 2014. DOI: 10.1007/JHEP07(2014)079. arXiv: 1405.0301 [hep-ph].

[43] R. D. Ball *et al.*, "Parton distributions for the LHC Run II," *JHEP*, vol. 04, p. 040, 2015. DOI: 10.1007/JHEP04(2015)040. arXiv: 1410.8849 [hep-ph].

[44] Y. Huang, M. Garcia, S.-E. Dahbi, *et al.*, "Search for heavy resonances in di-lepton plus photon final states in 139 fb-1 of pp collisions at $\sqrt{s}$ = 13 TeV with the ATLAS detector," ATLAS Collaboration, Geneva, Tech. Rep., 2022. [Online]. Available: https://cds.cern.ch/record/2815366/.

[45] A. Alloul, N. D. Christensen, C. Degrande, C. Duhr, and B. Fuks, "FeynRules 2.0 - A complete toolbox for tree-level phenomenology," *Comput. Phys. Commun.*, vol. 185, pp. 2250–2300, 2014. DOI: 10.1016/j.cpc.2014.04.012. arXiv: 1310.1921 [hep-ph].

[46] T. Sjostrand, S. Mrenna, and P. Z. Skands, "PYTHIA 6.4 Physics and Manual," *JHEP*, vol. 05, p. 026, 2006. DOI: 10.1088/1126-6708/2006/05/026. arXiv: hep-ph/0603175 [hep-ph].

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

[48] M. Cacciari, G. P. Salam, and G. Soyez, "FastJet User Manual," *Eur. Phys. J.*, vol. C72, p. 1896, 2012. DOI: 10.1140/epjc/s10052-012-1896-2. arXiv: 1111.6097 [hep-ph].

[49] 2019. [Online]. Available: https://news.cornell.edu/stories/2019/09/professors-perceptron-paved-way-ai-60-years-too-soon.

[50] R. Pramoditha, *The concept of artificial neurons (perceptrons) in neural networks*, 2021. [Online]. Available: https://bit.ly/3WoSBLD.

[51] S. SHARMA, *Activation functions in neural networks - towards data science*, 2017. [Online]. Available: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.

[52] Mazur, *A step by step backpropagation example*, 2015. [Online]. Available: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/.

[53] R. D, *Gradient descent algorithm - raghunath d - medium*, 2019. [Online]. Available: https://medium.com/@rndayala/gradient-descent-algorithm-2553ccc79750.

[54] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[55] A. Graves, "Long short-term memory," in *Supervised Sequence Labelling with Recurrent Neural Networks*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 37–45, ISBN: 978-3-642-24797-2. DOI: 10.1007/978-3-642-24797-2_4. [Online]. Available: https://doi.org/10.1007/978-3-642-24797-2_4.

[56] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," in *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, vol. 2, 1999, 850–855 vol.2. DOI: 10.1049/cp:19991218.

[57] I. Kobyzev, S. J. Prince, and M. A. Brubaker, "Normalizing flows: An introduction and review of current methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 11, pp. 3964–3979, 2021. DOI: 10.1109/tpami.2020.2992934. [Online]. Available: https://doi.org/10.1109%2Ftpami.2020.2992934.