Investigations of Calorimeter Clustering in Atlas using Machine Learning

by

Graeme Niedermayer
B.Sc., University of Ottawa, 2013

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Physics and Astronomy

Investigations of Calorimeter Clustering in Atlas using Machine Learning

by

Graeme Niedermayer
B.Sc., University of Ottawa, 2013

Supervisory Committee

Dr. Robert V. Kowalewski, Supervisor
(Department of Physics and Astronomy)

Dr. J. Michael Roney, Departmental Member
(Department of Physics and Astronomy)

ABSTRACT

The Large Hadron Collider (LHC) at CERN is designed to search for new physics by colliding protons with a center-of-mass energy of 13 TeV. The ATLAS detector is a multipurpose particle detector built to record these proton-proton collisions. In order to improve sensitivity to new physics at the LHC, luminosity increases are planned for 2018 and beyond. With this greater luminosity comes an increase in the number of simultaneous proton-proton collisions per bunch crossing (pile-up). This extra pile-up has adverse effects on algorithms for clustering the ATLAS detector's calorimeter cells. These adverse effects stem from overlapping energy deposits originating from distinct particles and could lead to difficulties in accurately reconstructing events. Machine learning algorithms provide a new tool that has potential to improve clustering performance. Recent developments in computer science have given rise to new set of machine learning algorithms that, in many circumstances, out-perform more conventional algorithms. One of these algorithms, convolutional neural networks, has been shown to have impressive performance when identifying objects in 2d or 3d arrays. This thesis will develop a convolutional neural network model for calorimeter cell clustering and compare it to the standard ATLAS clustering algorithm.

Contents

List of Tables

List of Figures

DECLARATION

Contributions from this thesis include:

- constructing a toy model and convolutional neural network (using Keras) for proof of concept.
- designing convolutional neural network structures (using Keras) for training on ATLAS simulations.
- incorporating a matching algorithm into the objective function of these networks. To the knowledge of the author, an objective function that contains a matching algorithm has not previously been explored.
- specifying parameters for ATLAS Monte Carlo simulations (using Athena) for these network structures.
- performing a hyper-parameter scan of 30 network structures.
- programming scripts that organize calo-cells into images for Keras.
- programming scripts that form CalHits into Truth-Clusters for Keras.
- comparing these convolutional neural networks with the Topo-Cluster algorithm.

A specific timeline can be found in Figure A.1 as a Gannt chart.

This work relied on pre-existing software. The software includes:

- Keras

  This is a high-level API that wraps around Theano.

- Theano

  This library implements the background material covered in Chapter 3.

- Athena

  This is the framework used for ATLAS-specific software.

Other libraries and computational resources used are listed in Appendix C.

ACKNOWLEDGEMENTS

I would like to thank my family: **Kay Niedermayer**, **Colette Forbes**, and **Daryle Niedermayer**. You have all been unyielding in your support and love.

Countless thanks to **Colby Richardson** who helped start me on this path by starting debates about the speed of light in grade-7. You may have taught me more about creativity and curiosity than anyone else.

Special thanks to:

- **Julie Ramsey** and **Jèrèmie Harris**, my past lab partners, for helping me with my first formal experimental research.
- **Saskatchewan Science Center**, for the opportunity to share my enthusiasm of science. While working there, the staff[1] was amazing and expanded my appreciation of the sciences.
- **Mia Crewe** and crew[2], for the sublime underground caving trips organized through the **UVic Caving Club**. These trips truly helped balance my thesis work.
- My **graduate school cohort**[3], for have all been so supportive.
- Professor **Michel Lefevbre** and **Kate Taylor**, for help with ATLAS Master-Class as well as unrelenting positivity and boundless energy.
- **Allison, Tony, Justin, Savino,** and **Kayla**, my office mates, for a great work environment, particle physics help, and useful advice. Your skills with software is undeniable.
- **ATLAS** and the **ATLAS machine learning group**, both provided amazing resources. A large number of members provided helpful insight. Some of these members include: **Michael Kagan, David Rousseau, Gilles Louppe, Daniel Guest, Kenji Hamano,** and Professor **Richard Keeler**.

[1] An incomplete list of coworkers and friends: Tiffany Vass, Merissa Scarlett, Justin Schneider, Rebecca Hay, Ali Ryan, Emily Putz, Casey Sakires, and Jesse Searcy.

[2] An incomplete list of cavers: Michael Brinton, Lauren Eckert, Peter Chiba, Stuart Taylor, Chelsea Power, Tristan Crosby, Kirsten Mathison, Buzas Levente, Josiah Macleod, Ethan Dinnen, Marisa Davy, Cameron Gregory, and Stuart de Haas.

[3] An incomplete list of the cohort: Andrew, Matthias, Ildara, Astara, Zoey, Mannix, Eva, Anne-Marie, Michael, and Noa.

- **Kacie Williams**, for friendship, square-circles, and emotional support.
- **Juan Hernandez**, for many long conversations about theoretical particle physics which really helped solidify some of my foundational understanding.
- **Pramodh Yapa**, for friendship, support, and some of the most useful conversations about linear algebra. You are truly one of the best people to bounce ideas off.
- **Chelsea Dunning**, for python hints, friendship, computer cluster hints, being a great roommate, and your general positivity.
- **Jannicke Pearkes**, for doing a great co-op term that helped initiate this project. Your suggestions of libraries helped give me a sturdy foundation to build off.
- **Matt LeBlanc** and **Jennifer Roloff**, for helping with a particularly difficult roadblock related to CalHits.
- Professor **J. Micheal Roney** and Professor **George Tzanetakis** for corrections, edits, and comments made for this thesis.

Finally thank you Professor **Bob Kowalewski** for all the experience, knowledge, and wisdom you brought to this project.

This is a truly non-exhaustive list.

DEDICATION

For Rita:

Before I was looking up at the stars,
I was looking up to you.

Chapter 1

Introduction

The goal of this project is to explore the replacement of a sub-algorithm linked to reconstructing particle momenta within the ATLAS detector using recent discoveries from the field of Machine Learning. Under higher luminosity conditions, these machine learning algorithms could provide performance improvements.

1.1 Motivation

Many of the recent results from particle detectors have searched for evidence of new physics. These papers largely place more stringent constraints on the allowable parameter space for new physics[1, 2]. One of the limitations to these papers is the number of rare events. Some of these papers search for processes that correspond to rates of once in ten billion events. A method for improving the sensitivity of searches for new physics is to increase the number of events. For many experiments this increase in statistics corresponds to increasing an experimental variable known as luminosity. For the particle experiments located along the Large Hadron Collider (LHC) ring the project to increase this luminosity has been named the High-Luminosity Large Hadron Collider (HL-LHC)[3] and corresponds to an increase by a factor of ten. The primary method for increasing luminosity and the number of these rare events is to increase the number of simultaneous proton-proton collisions. This comes with a trade-off of also increasing a particularly difficult form of noise within the detector. This noise comes from the unavoidable and undesirable simultaneous proton-proton interactions per bunch crossing referred to as pile-up. Pile-up deposits particles in the ATLAS detector that are not associated with the interesting physics processes

being sought.

The improvement in sensitivity to new physics from the HL-LHC is dependent on maintaining the quality of data despite the rise in pile-up[4].

Many of the current algorithms are set up to scale well with random noise; however, it is less clear how they will interact with pile-up, which deposits correlated particles in the detector. There are some signs that pile-up might cause difficulty within the calorimeter[5, 6].

Concurrently, in the discipline of computer science there has been a renaissance of technologies related to artificial neural networks. Particularly in computer vision problems, these approaches have begun solving problems at levels comparable to human categorization[7]. These machine learning algorithms promise high performance and have already been implemented successfully in a few areas in physics such as categorization, tagging, and simulation[8, 9, 10, 11, 12, 13, 14]. These algorithms seem to be very robust to structured noise.

The negative aspects of this increase in pileup could be mitigated by a machine learning approach and this project will use machine learning to combine calorimeter cells into larger cluster objects in a meaningful way[1].

1.2 Challenges

This objective gives rise to significant challenges[2] which need to be overcome over the course of this project. These include:

[1] A recent article used these methods for a jet groomer for pile-up mitigation[15](these results are not through peer review and may be excessively optimistic).

[2] A few of these challenges are common when approaching problems with machine learning.

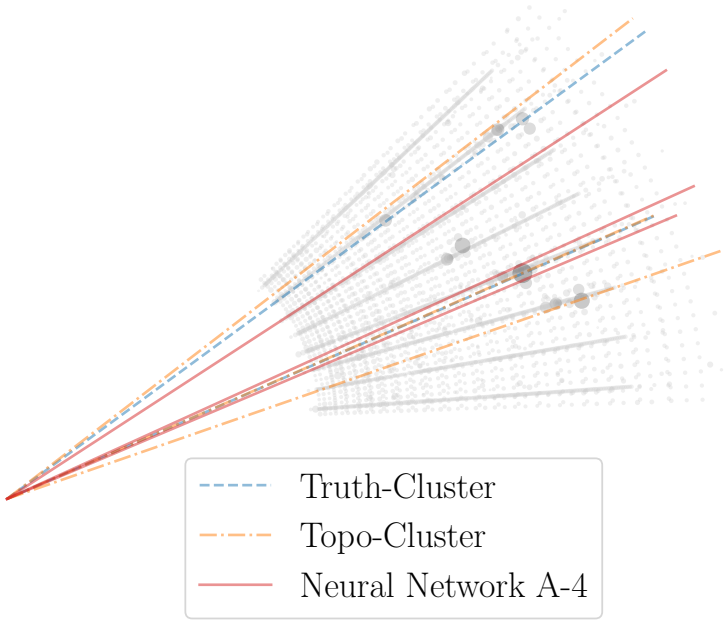
1. Particle detectors have complex geometries which do not translate immediately to rectangular images or videos.
2. Comparing sets of particles can lead to ambiguities (a combinatorics problem).
3. Particle physics phenomena leads to complications that are domain specific such as: shower shapes, extra particle interactions, and pile-up.
4. There are a large amount of hyper-parameters and artificial neural network architectures to be investigated.
5. We would like to get numerical values for quantities such as the magnitude and location of energy depositions. This problem is known as regression. Regression is generally considered more challenging than categorization for machine learning algorithms.

All of these challenges will be tackled with varying degrees of success.

1.3 Overview of Results

An algorithm will be demonstrated that can estimate the location and energy of incident particles, from a set of calorimeter cells. This algorithm will converge towards constructed target clusters with training. The performance of this machine learning algorithm will compare unfavorably to the current ATLAS algorithm (known as "topo-clusters") which came from many FTE-years of development effort. However, it is one of the first implementations of machine learning for calorimeter clustering in such a highly granular detector and contains many avenues for improvements. A preview of this energy-deposition-finding machine learning algorithm is shown as Neural Network A-4 in Figure 1.1.

Figure 1.1: **Machine Learning Example**: An example that uses calorimeter cells as inputs and energy depositions as outputs. The calorimeter cells are represented by circles where their sizes represent how much energy they contain. The lines reflect a projection to the location of the energy deposits.



1.4 Agenda

Here we'll outline an agenda to use for the rest of the thesis:

Chapter 2 focuses on experimental particle physics. This outlines key ideas from particle physics then gives an overview of the LHC and ATLAS experiment. Finally it outlines core aspects of the detector and the topo-cluster algorithm.

Chapter 3 elaborates on the machine learning algorithms being used. This chapter will discuss the reasons for using convolutional neural networks.

Chapter 4 shows the datasets and the pre-processing used. The formations of truth clusters will be discussed. A hyper-parameter scan is done to optimize the structure of the networks.

Chapter 5 compares truth targets to both the topo-cluster algorithm and the neural networks. This chapter will conclude by comparing the topo-clusters with the clusters formed from neural networks.

Chapter 6 concludes with remarks about machine learning and clustering. It will return to the challenges introduced above and discuss other routes forward.

Appendix A contains supplementary material for future researchers.

Appendix B contains supplementary material on machine learning and artificial neural networks.

Appendix C contains supplementary data, the software used, the hardware, and software suggestions.

Chapter 2

Experimental Particle Physics

2.1 Particle Physics

Particle physics explores the most basic objects and their interactions within our universe.

Because of the complexity that arises at the length and energy scales of particle physics we'll begin with an analogy with classical physics resonances. Additional background for this next section can be found in Chapter 4 of Ref. [16], Chapter 48 of Ref. [17], and Ref. [18].

2.1.1 Classical Resonance Analogy

Let's start our exploration of particle physics by making an analogy with classical physics through a thought experiment (Fig. 2.1). We build this experimental setup with a tuning fork, a speaker and a sound recorder with the goal of finding the resonance frequency. Finally we'll place the limitation of not being able to directly strike the tuning fork. There are a number methodologies of reaching this goal but we'll focus on a particular method that will work as an analogy to particle physics. We can play a broadband sound source and beam it directionally towards the frictionless tuning fork. This sound will be absorbed preferentially by frequency and then re-emitted in a scattered direction. By strategically placing a recording device we might be able to record this re-emitted sound. If we can isolate the re-emitted sound from the original sound we will have the output spectrum of the tuning fork. Now we'll either need to open a textbook[19], do some experiments, or do some math to remember

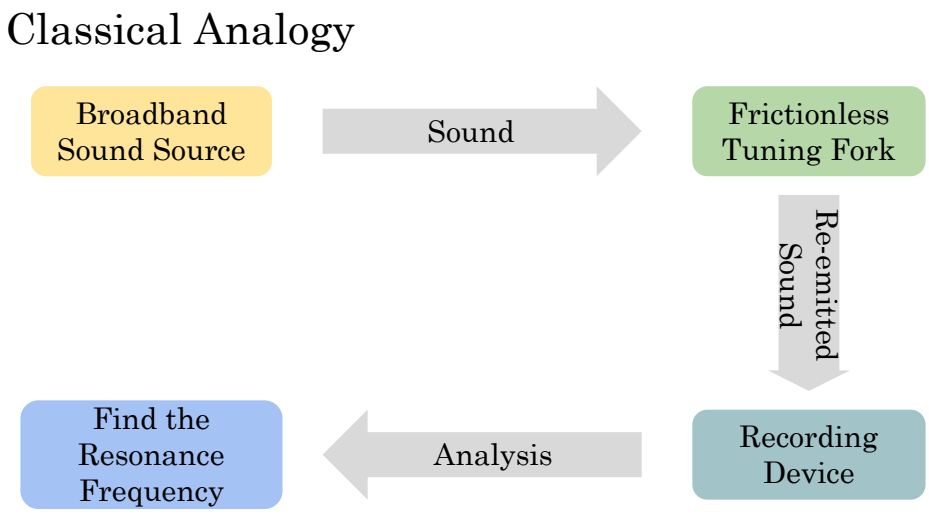


Figure 2.1: **Classical Thought Experiment**: The setup used to obtain resonance frequency of tuning fork. Can be used as an analogy for a simplified model of experimental particle physics shown in Fig. 2.1.

the power emitted from the tuning fork[1].

This is a solution of the sinusoidally driven damped simple harmonic oscillator near the resonance frequency. It has the form shown in Fig. 2.2 and can be written as:

$$P(\omega) = \frac{A}{(B-\omega)^2 + C}$$

By fitting this function we can obtain the resonance frequency without ever having to directly strike the tuning fork.

Now let's look at Fig. 2.3 which contains the 2016 experimental data from the Z boson.

From this image a particular line-shape should jump-out: the resonance line-shape. This is one of the reasons that unstable particles within particle physics are often referred to as resonances.

Let's build Table 2.1 which lists some of the similarities between particle physics experiments (ATLAS and CMS) and the thought experiment above.

To start with we need something to carry the resonance. Similar to light we can

[1]In this thought experiment all of the power 'consumed' is perfectly re-emitted meaning some sort of frictionless tuning fork that does not dissipate energy. This analogy is more accurate in atomic physics with photons but is less relate-able.

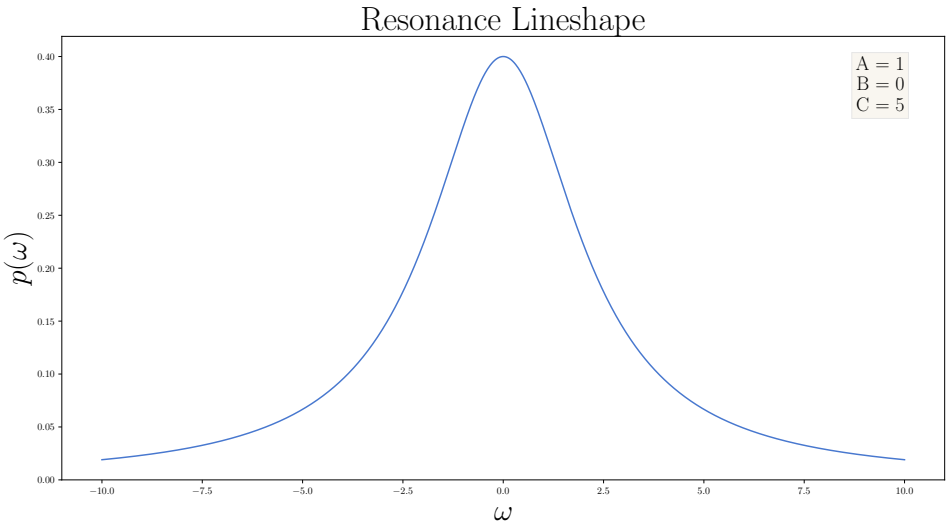


Figure 2.2: **Resonance Line-Shape**: The absorption/re-emission line-shape of a sinusoidally driven damped harmonic oscillator near resonance.

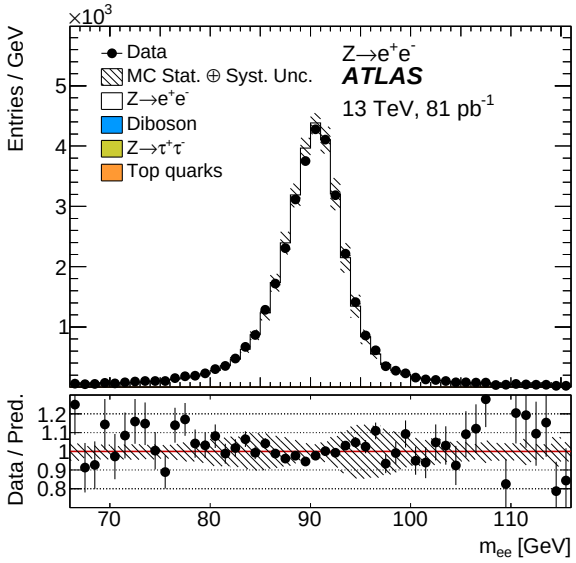


Figure 2.3: **ATLAS Z boson resonance**: Note the resonance. The line-shape is explored more in Appendix A.1.2. Taken from an auxiliary figure of Ref. [20]

introduce a field as a medium[2]. These fields will be termed quantum fields. Many of these fields are for most purposes invisible, which means we need a procedure to find them. The procedure utilized at the ATLAS and CMS can heuristically be seen as

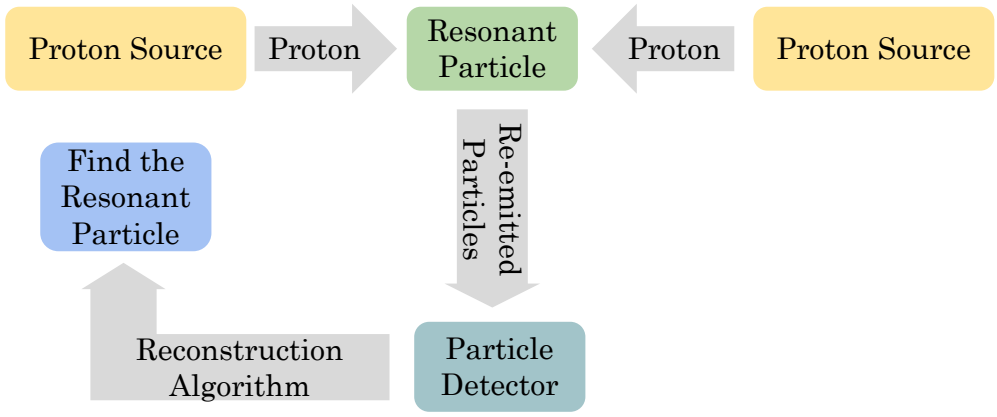
[2]These fields have manifest Lorentz invariance that distinguishes them from the erroneous aether theories of the past

Table 2.1: **Comparison of Classical Analogy and Particle Physics**: This table shows some of the analogies that can be drawn from the comparison of particle physics with the classical thought experiment.

Commonality	Classical Analogy	Particle Physics
Resonance	Acoustic Excitations	Short-lived Particles
Medium for Resonance	Tuning Fork	Quantum Field
Method for causing Resonances	Sound	Collisions
Method for detection	Re-emitted Sound	Outgoing particles
Width of the Spectrum	Damping term	Particle's lifetime

making proton-proton collision which cause resonances that decay and lead to very specific sets of outgoing particles. This can be seen in Fig. 2.4.

Figure 2.4: **Analogous Model of Particle Physics**: this model shares many similarities to the thought experiment shown in Fig. 2.1.



We can introduce a new field for each of new resonance. However something that quickly becomes clear is there is a very large number of resonances with many different amplitudes. Fortunately there are some clean ways of organizing this "particle zoo". We'll organize them by a smaller set of fields and allow these fields to interact with each other and have multiple levels of excitations or modes. This model with a reduced number of fields along with the study of these fields is commonly referred to as the Standard Model.

The systematic and formal rules of the coupling and interaction of these fields and their particles is quite complicated[3] and are known as quantum field theories. The Standard Model can be seen as a specific set of quantum field theories.

With this in mind let's use the analogy to gain some understanding:

- What's one thing physicists are searching for at ATLAS? When physicists are looking for new physics, one search they do is for a new statistically significant resonance lineshape.
- Why might physicists be colliding protons? They provide a high energy broad-band source.
- Why are many of the particles that particle physicists discuss never seen on a daily basis? Particles are only seen at high enough energies to excite the quantum fields. These excited fields are "damped" into other more stable quantum fields.

Most critically this analogy underlines the importance of analysis and reconstruction, particularly in regards to frequency and energy resolution. With a clear enough line-shape, particle physicists can find the mass, lifetime, and infer interaction strengths. However if the experimental setup has poor energy resolution this will smear out the line-shape and this information will be lost or reduced. This thesis will focus on a promising method for maintaining resolution with the increasing number of collisions.

2.1.2 The Standard Model

The Standard Model of Particle Physics has had a little bit of a moving definition over the years. We'll be referring to the most recent one as of writing this document.[4] This theory is a set of quantum field theories with seventeen fields, one for each particle, and a set of specific parameters for these fields. Breaking it down by the type of quantum field theory: there are twelve fermionic, four spin-one bosonic, and one spin-zero bosonic field. Excitations in these fields give rise to leptons and quarks; gauge bosons; and the Higgs bosons respectively. In the Standard Model, quantum

[3] The analogy above should not be taken too literally.

[4] At the time of writing this document: the Higgs boson and mechanism has been added although some of its traits are still being experimentally verified; and the method for including neutrinos oscillations/masses have not been finalized/agreed upon.

fields can undergo two types of excitations which give us: particles and anti-particles. The anti-particles have many inverse traits to their counterpart particle (such as charge), but also share many traits (such as mass). There are a variety of couplings between the fields however most often these fields interact predominantly[5] through the four spin-one bosonic fields so these bosonic fields are referred to as "Force Carriers". The Standard Model does not include a mediator for gravity[6]. Gravity is omitted from the Standard Model due to the difficult of creating a self-consistent theory that agrees with experiments and observations.

The Standard Model provides an accurate description of the phenomena observed to date, but there are good reasons for modifying the Standard Model. Models that add new features are usually referred to as Beyond the Standard Model and some of the issues they address include: dark matter[21], dark energy[22], gravity, topics in cosmology, and aesthetic issues with the Standard Model (fine-tuning). Modern experimental particle physics can be thought of as primarily testing and refining the Standard Model.

2.2 Modern Particle Physics Experiments: LHC and ATLAS

In order to study fundamental particles and their interactions most experiments require two things: a source of particles and a detector.

2.2.1 The Large Hadron Collider Overview

Particle sources can be anything from cosmic rays, radioactive samples, solar particles, or a particle collider. The Large Hadron Collider (LHC) is both a particle source and a particle collider [24, 25]. The LHC forms a massive ring that contains two accelerators that accelerate beams of protons in opposite directions. The beams are organized into *bunches* of protons which are separated by gaps of 25 ns. Sometimes the whole beam is referred to as a *bunch train* giving a visual to this structure. This bunching is done because of the acceleration method which requires radio-frequency cavities. Particle colliders have an advantage of producing large numbers of particles with specific

[5] Two fermions can act similar to a boson.

[6] A quick note: the Higgs field is related more closely to the concept of inertial mass than gravitational mass and as of writing this shouldn't be seen as a gravitational mediator.

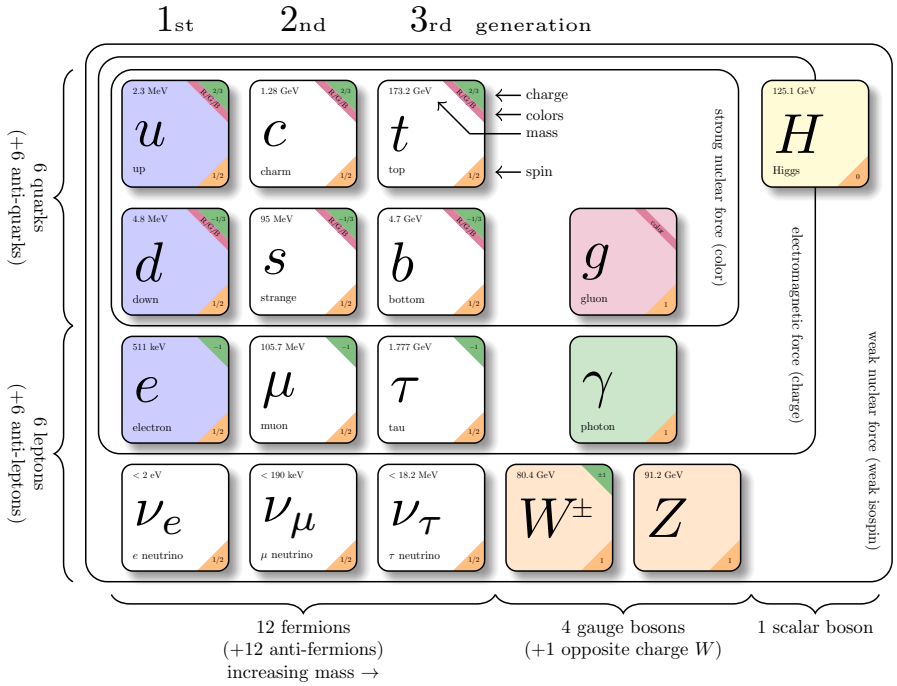


Figure 2.5: **The Standard Model of Particle Physics**: taken from Ref. [23] updated with values from Particle Data Groups[17].

qualities (such as energy) allowing us to create a large sample of collisions under statistically identical conditions. This brings up an important measure for the rate at which we acquire information, referred to as luminosity. Luminosity depends on the particular experiment, but for two beams with Gaussian profiles colliding head-on repeatedly we can define it as[26]:

$$L = \frac{n_{\text{bunches}} f_{\text{rev}}}{4\pi} \frac{N_1 N_2}{\varepsilon_x \varepsilon_y} \tag{2.1}$$

where f_{rev} is the revolution frequency, n_{bunches} is the number of bunches, N_1 & N_2 are the number of particles in each bunch, and ε_x & ε_y are the standard deviations that specify the transverse size of the beam. Because this luminosity rate is an instantaneous quantity. we can define *integrated luminosity* by integrating luminosity over the time:

$$L_{\text{integrated}} = \int_{t_o}^{t} L(t) dt \tag{2.2}$$

The LHC is the highest center of mass energy collider currently in operation with

13 TeV collisions. At this center of mass energy, the LHC has produced approximately $L_{\text{integrated}} = 30\text{fb}^{-1}$ worth of data. By comparison HL-LHC is projected to produce $L_{\text{integrated}} = 3000\text{fb}^{-1}$ by the end of it's operation in 2038[27]. This allows the production of high energy particles and the exploration of new physics at high rates.

Final States

The LHC is designed to slam two protons together creating a large variety of initial conditions for collisions[7]. This great number of initial conditions will create an even larger variety of *final states* after the collision. Depending on the particles produced, these final states can be observed by components of the detector through: direct interaction, child particle interaction, or missing interactions. Reconstruction of the final states is the primary role of particle detectors. Figure 2.6 outlines the two 'directions' of particles physics. Physics causes final state particles to produce electronic signals which are digitized and read out. Meanwhile, reconstruction is the process by which those instrument readouts are used to rebuild the final state particles.

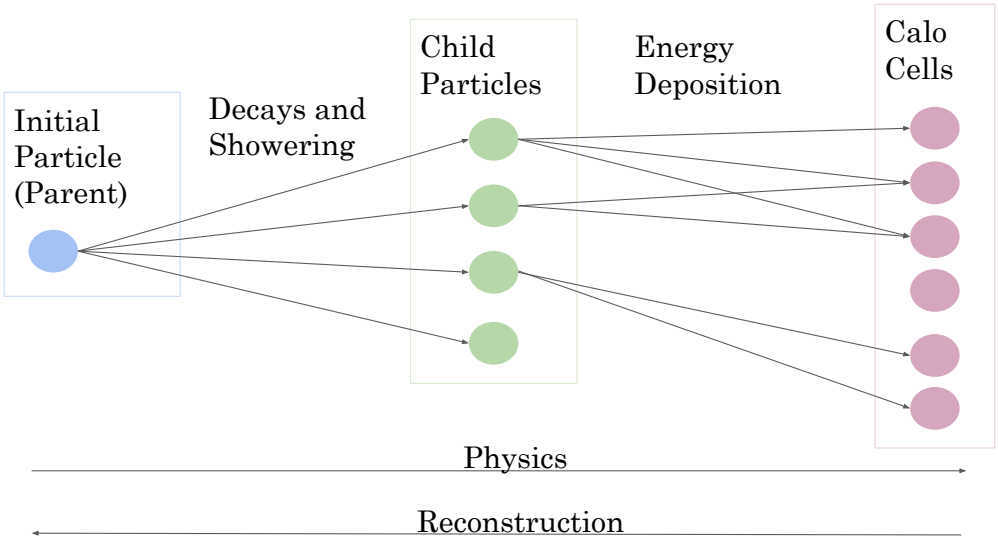


Figure 2.6: **Schematic of Physics and Reconstruction**: This shows a simplified description of how final states deposit energy in the calorimeter.

It would be useful to know the rate of creating a particular final state. These rates can be found by multiplying a quantity related to the strength of an interaction,

[7]These initial conditions has to do with both kinematics and the composite nature of protons.

known as the *cross-section*, by the luminosity:

$$R^{i\to f} = \sigma^{i\to f} L \tag{2.3}$$

where σ is the cross-section and R is the rate; both of which are between an initial state i and a final state f. Only a very small fraction of these finals states are of interest to modern particle physics. As an example the probability of creating a higgs boson at ATLAS is of the order $\sigma_{Higgs}/\sigma_{total} \approx 10^{-10}$. This highlights the technological challenge of the ATLAS experiment.

2.2.2 ATLAS Overview

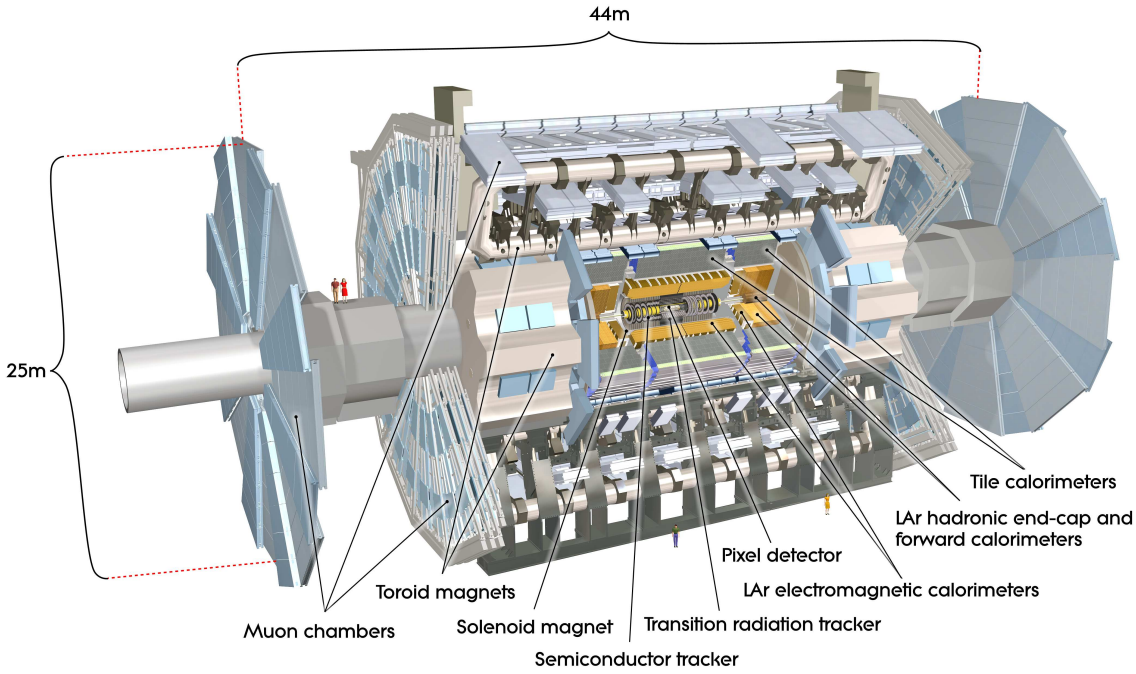
The ATLAS detector[8] is a multi-purpose detector located along the LHC ring. ATLAS is shown in Figure 4.1. Its goal is to observe information from the collisions by essentially acting as an eye.

This observed information is commonly summarized by the integrated luminosity variable. Once the maximum number of available bunches is used (currently the case at LHC), the only way to increase the luminosity is to increase the number of protons per bunch or to decrease the cross-sectional area of the bunches (see Eqn. 2.1). This increase in luminosity per bunch has the consequence of increasing the number of simultaneous proton-proton collisions. Each collision will independently deposit energy into ATLAS, which can lead to a degradation in reconstructing a single proton-proton collision of interest. Returning to the eye analogy this can *roughly* be thought of trying to find a picture of interest by looking at many pictures simultaneously, with some of the pictures overlaid.

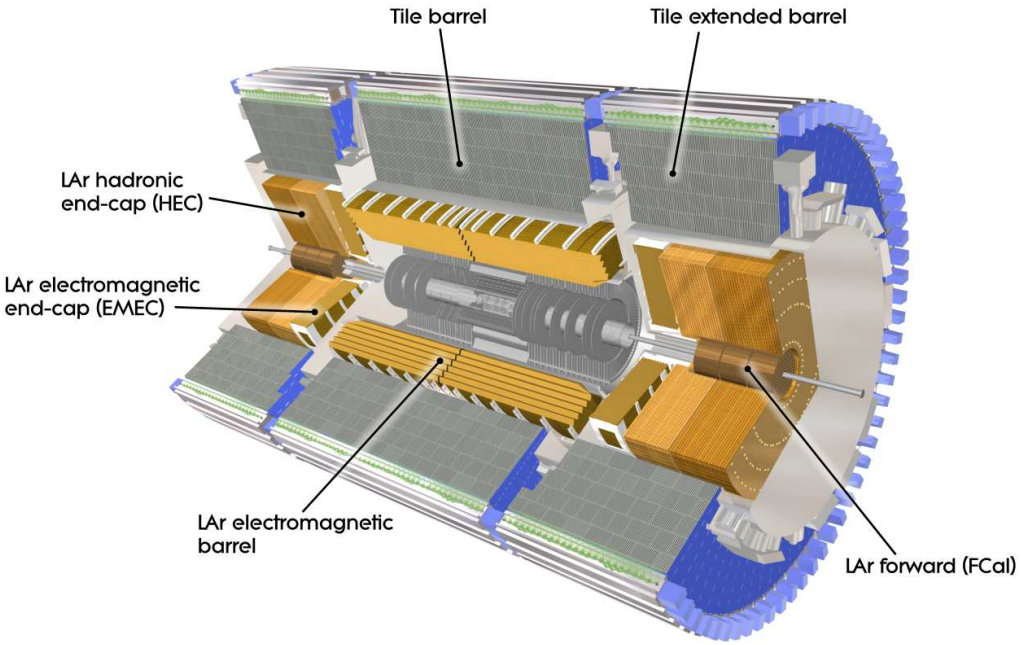
Similar to a human eye ATLAS contains a large number of smaller detectors used for reconstructing events. These detectors at ATLAS can be divided into three categories and are geometrically located in increasing distance from the interaction point: the inner detector, the calorimeter, and muon spectrometer.

[8]Two important references are [28, 29]. Ref. [28] largely deals with the schematics of the detector and subsystems while Ref. [29] largely focuses on expected particle physics interactions with the detector.

Figure 2.7: **The ATLAS Detector**: On top is the full ATLAS detector. Near the center of it is the calorimeter which has been blown up and displayed on the bottom.



The ATLAS Calorimeter: The majority of this project relates to a subsection the electromagnetic and hadronic endcaps (bronze cylinders on both sides of the calorimeter).



2.3 ATLAS Detector

The ATLAS detector is often subdivided into three subsections from smallest to largest: the inner detector, calorimeter, and muon spectrometer. Each of these sections can be seen as being designed for a particular goal. As such each section has their own sub-systems, design choices, and geometry. This thesis is interested primarily in the calorimeter which will be discussed in more detail.

2.3.1 Coordinate Systems

The ATLAS detector uses a couple of coordinate systems. The Cartesian coordinate system has its origin at the center of the detector with the x-axis pointing towards the center of the LHC ring, y-axis pointing towards the surface, and the z-axis running along the beam direction such that it follows the right-hand rule. *Transverse* values are projected into the x-y plane whereas the z-axis is referred to as *longitudinal* or *beam* direction. In particle physics a directional coordinate system is defined at the center of the detector with coordinates of: pseudo-rapidity, azimuthal angle, and a proxy for depth—such as a radial or momentum coordinate. In this space particle production distributions are approximately uniform and differences between directions (i.e. solid angles) are Lorentz invariant[9]. The polar angle pseudo-rapidity or η is defined by:

$$\eta = -\ln\left(\tan\left(\frac{\arccos\left(\frac{z}{\sqrt{x^2+y^2+z^2}}\right)}{2}\right)\right)$$

pseudo-rapidity is discussed more the Appendix A.1.1. Energies are calibrated assuming electromagnetic particles caused the deposits unless otherwise mentioned.

Inner Detector

The inner detector is designed to track the motion and measure the momentum of charged particles. This is done by using a variety of sensors including pixel detectors and strip detectors. A large solenoid coil surrounds the inner detector making the tracks of charged particles bend from the magnetic field. We can explore how these

[9] more precisely rapidity is invariant to Lorentz boosts in the longitudinal-direction (z-axis). Pseudo-rapidity is a proxy for rapidity for high energy particles.

magnetic fields affect charged particles by starting with the Lorentz force[10]:

$$\mathbf{F} = q(\mathbf{v} \times \mathbf{B} + \mathbf{E})$$

Within the detector the magnetic fields are partially perpendicular to the motion of particles. This gives us $\mathbf{F} = |\mathbf{v}||\mathbf{B}| \sin\theta\hat{\phi} = |\mathbf{v}_T||\mathbf{B}|\hat{\phi}$ so the particles will be curved in the $\hat{\phi}$ direction. The magnetic field in the inner detector travels down the cylinder in the $\hat{z}$ direction and is uniform. Metal yokes help the magnetic field lines return to the other end of the solenoid without producing a magnetic field within the calorimeter.

Calorimeter

The calorimeter is designed primarily to find the energy and position of incident particles. This requires high resolutions of both energy deposits and transverse positioning of the energy deposit. This will be discussed in more detail in the Section 2.3.2.

Magnet Systems

Magnet systems are used to measure momentum through the Lorentz forces. The inner detector has a solenoid around it giving an approximately constant magnetic field in the $\hat{z}$ direction. Metal guides return the magnetic fields back through the calorimeter structure causing the magnetic field to be approximately zero in the calorimeter. The second magnetic system is the toroidal magnetic field which is formed by 16 superconducting race tracks that gives a donut shaped magnetic field that surrounds the calorimeter in the $\hat{\phi}$ direction.

Because the magnetic fields are designed to be kept out of the calorimeter their impact on charged particles takes place before and after the calorimeter.

Muon Spectrometer

Muons are involved in some of the most relevant decays in particle physics so measuring them accurately is very important. However with their small production cross-section and their penetration depth, they require a different strategy to detect. This is why the largest volume portion of the detector is designed and dedicated to measuring the momentum of muons.

[10] the covariant form $\frac{\mathrm{d}p^\alpha}{\mathrm{d}\tau} = qF^{\alpha\beta}U_\beta$ goes to $\mathbf{F} = q(\mathbf{v}\times\mathbf{B}+\mathbf{E})$ when $\frac{\mathrm{d}p^0}{\mathrm{d}\tau} = 0$ (Energy is conserved)

2.3.2 Calorimetry Details

The calorimeter can be divided into three geometric regions: the central barrel region, the two end-caps, and forward regions. The forward region lies near the beam-line.

Materials and Pulse Shape

The ATLAS calorimeter can be categorized as a *sampling calorimeter*[11]. This means that the calorimeter can be split into active regions and inactive regions as shown in Fig. 2.8. The high energy particles traversing active regions cause ionization that results in charged particles that drift to anodes. The current these drifting particles produce is measured and converted into an energy measurement. The ionization drifts to the anode creating a current that has a triangular shape versus time as shown in Figure 2.9. This is electronically reshaped into a pulse that integrates to zero. The reason for this will be explored more in Section 2.3.4. In the inactive materials energy is not recorded but the energy deposited can be empirically estimated. These materials are summarized in Table 2.2.

Table 2.2: **The Calorimeter Materials**: This table summarizes material used throughout the calorimeter to find energy deposits.

Calorimeter Section	Active Material	Inactive Material
EM calorimeter (both barrel and end-cap)	liquid argon	lead
Hadronic end-cap calorimeter	liquid argon	copper
Hadronic barrel calorimeter	scintillator tiles	steel
Forward region calorimeter	liquid argon	copper and steel

Dead material are regions where energy can be deposited that are not as uniform as inactive materials. The energy loss in these regions can fluctuate leading to a degradation in resolution. These include areas such as mechanical support structure or wiring.

[11] As opposed to a solid crystal calorimeter

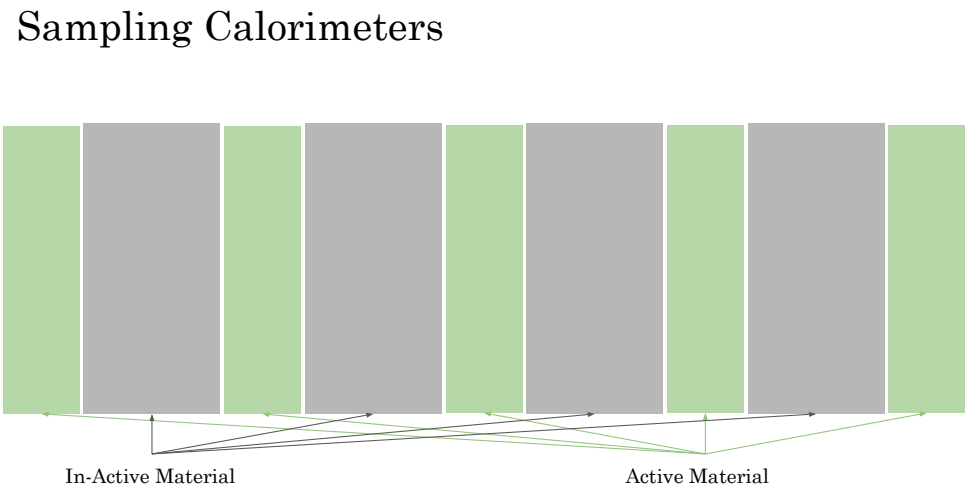


Figure 2.8: **Sampling Calorimeter**: Each region of calorimeter can be divided into inactive and active materials. Energy is deposited in both regions but only measured in active regions. Energy deposits in inactive regions can be estimated empirically with the active regions.

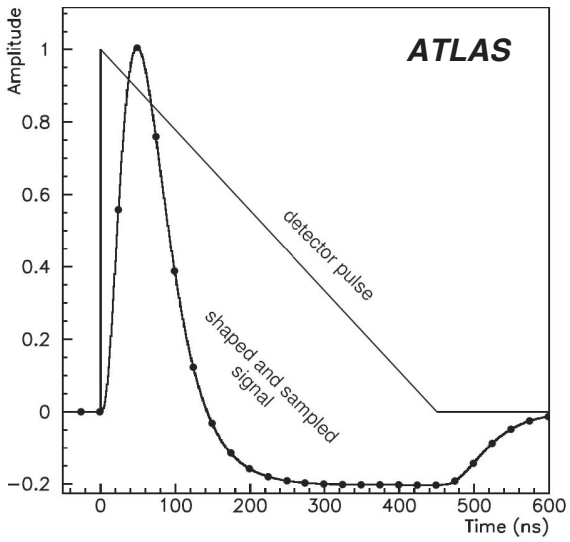


Figure 2.9: **Pulse Shape**: The raw detector pulse is shown as the triangle. This shape is reshaped so that the pulse will integrate to zero over time. This is done to help deal with the pile-up that is discussed in Section 2.3.4.

Granularities

It is important to discuss the structure of calorimeter cells in ATLAS. While they form a cylinder-like structure in x-y-z space, the cells form layers of image-like rectangular structures in η and ϕ (except for the forward region calorimeters). These can be organized into about 30 images. An example can be seen in two layers of the EM end-cap calorimeter in Figure 2.10.

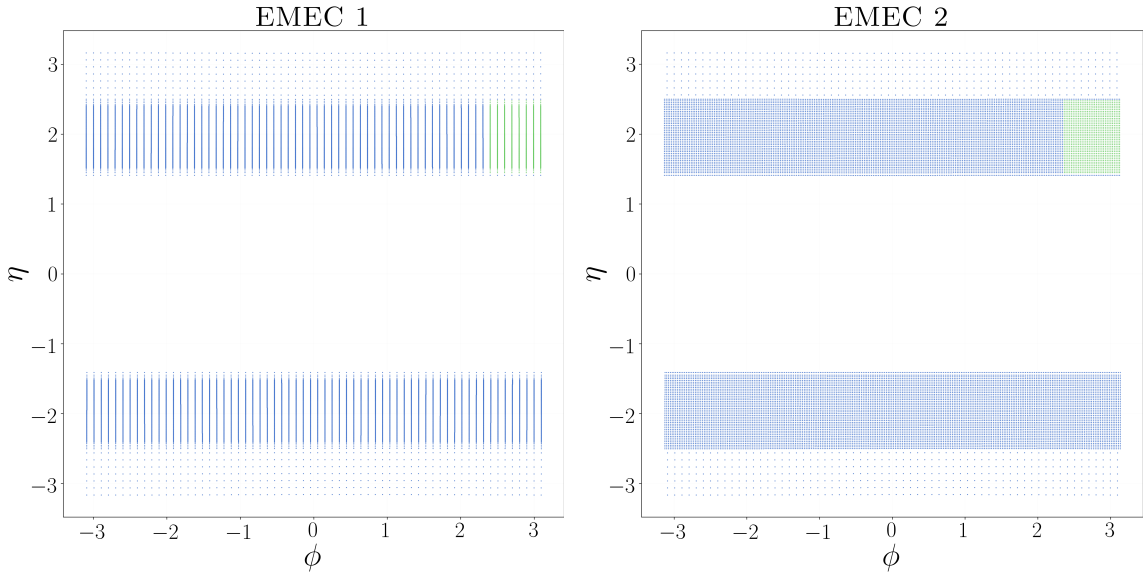


Figure 2.10: **Important Granularities**: The ATLAS Calorimeter is highly granulated for reconstruction. Here two layers, EMEC1 and EMEC2, from the electromagnetic endcap are shown. A visible difference is the granularity of ϕ. The granularity of η also differs although this isn't as visible. These layers are associated with the largest energy deposition for electrons and photons. This granularity varies greatly with depth and η. This highlights one of the challenges of this project.

Showers

The reconstruction is complicated by the impact of high energy particles on the sampling material. This interaction will often create more particles leading to a cascade or *shower*. These showers can be represented as a directed tree[12] split into infinitesimal time steps (as shown in Fig. 2.11). In this case each node in this graph represents the creation of at least one new particle. Let's take note of some important traits of the nodes and shower:

[12]Sometimes this is referred to as a *forward-directed rooted tree*.

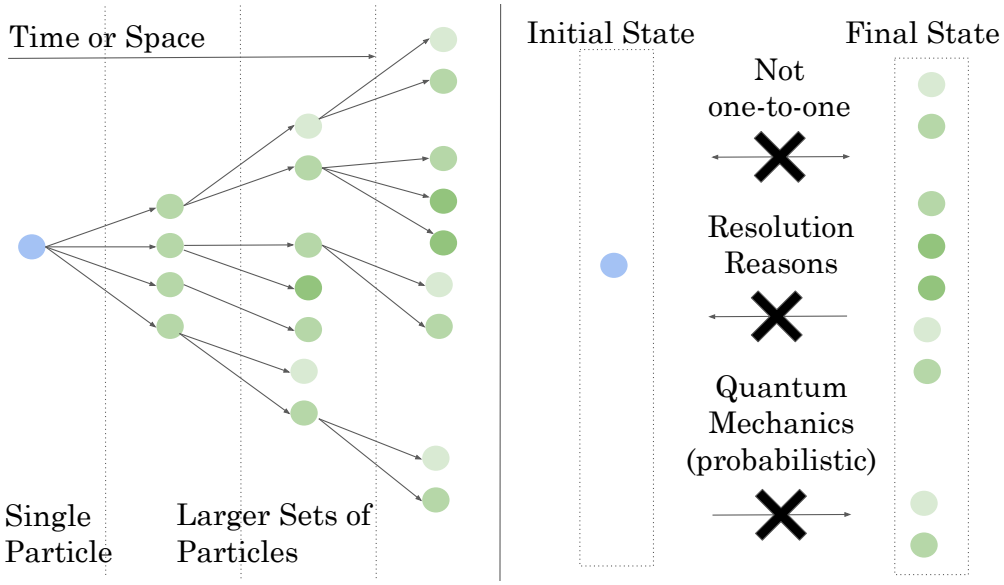


Figure 2.11: **Initial Particle to Final Cascade**: This graphic outlines how initial shower information is smeared into the final pieces of information. Because of this smearing, one final state can represent the outcome from many distinct initial states.

1. Each node can be thought of as having a probability of occurring. This means that one initial state will not deterministically lead to one final state.

2. A great number of the particles in the cascade have energy below a "critical threshold". This threshold refers to particles that will not produce/excite new particles because they do not have enough energy.

3. The number of final particles greatly exceeds the number of initial particles.

4. Generally the final states can be correlated with the initial particle and vice-versa.

5. The longitudinal energy deposition shape of the shower *roughly* resembles a Gamma distribution[13] with a strong skew towards the initial collision point.

6. The lateral energy deposition shape of the shower is highly dependent on the depth. The lateral shape generally has a sharper peak closer to the initial collision point that broadens with depth[14].

[13] This is a asymmetric bell-curve-like function. This is less true of hadrons where the tails deviate.
[14] By combining the longitudinal and lateral shape, we can see why early calorimeter layers have significantly higher granularity.

Nevertheless, using these final outputs we need to be able to infer an initial state. Fortunately these showers can easily be sub-divided into two types: electro-magnetic showers caused by photons and electrons, and hadronic showers caused by hadrons notably: pions, kaons, protons, and neutrons.

Electromagnetic Showers

At higher energies (above 1 GeV) electrons, positrons, and photons all interact in a similar fashion. Photons interact through pair production (the production of electrons and positrons) while electrons and positrons interact via bremsstrahlung (creating photons). These created particles will undergo more reactions creating a larger shower of electrons, positrons, and photons. These showers tend to have shallow penetration-depth which explains why the EM calorimeter is on the interior of the hadronic calorimeter. The lateral spread of EM showers is more collimated than hardronic showers. For EM showers this collimation is quantified by the Molire radius which contains 90% of the energy.

Hadronic Showers

Hadrons (particles made up of quarks) tend to shower in a very different way. They tend to produce a great number of pions which further shower. These showers tend to be deeper. They tend to have an EM shower component and hadronic-dominated component. The hadronic component takes longer to develop and deposits energy further into the calorimeter.

Jets

When very unstable particles hadronize[15] they produce many particles before reaching the calorimeter. These groups of particles are known as a *jet*. Energetic jets are often of physics interest because they tend to be created by heavy final states.

Resolution

Resolution is a critical subject when discussing any experiment. In this context it can be broken down into three separate sections: time, energy, and spatial resolution.

[15]This is a process where high energy hadrons split forming more hadrons.

This project is especially concerned with spatial and energy resolution and the arrival time of the calorimeter signal will be ignored[16].

Energy resolution is particularly crucial for a calorimeter as it affects the observable quantities and their uncertainties such as the peaks and width of the resonances. Energy resolution has contributions from three different sources: sampling, noise, and geometry. This can be described as:

$$\frac{\sigma}{E} = \frac{a}{\sqrt{E}} \oplus \frac{b}{E} \oplus c$$

with $\oplus$ as a quadrature sum. Breaking this down term by term we find that:

$\frac{a}{\sqrt{E}}$ The number of ionized particles is proportional to the energy. This leads to a fluctuation in ionization collected and gives rise to a Poisson term: $\sigma_{sampling}/E \propto 1/\sqrt{E}$.

$\frac{b}{E}$ This term is from electrical noise within this system and gives rise to: $\sigma_{noise}/E \propto 1/E$.

c This term is associated with leakage, non-uniform geometry, and mis-calibrations. This source of noise has the form: $\sigma_{constant}/E \propto$ constant.

specific values for the ATLAS electromagnetic calorimeter (for electromagnetic showers) depend on the state of the detector but example values are[17]:

$$[a, b, c] = [0.1 \text{ GeV}^{\frac{1}{2}}, 0.3 \text{ GeV}, 0.004]$$

2.3.3 The Topo-cluster Algorithm

The ATLAS topo-cluster algorithm is a method for gathering the 188000 calorimeter cells into a more manageable number of math objects for further reconstruction steps[28][17]. To start off with the topo-cluster primarily uses cell signal significance, which is defined as:

$$\zeta_{\text{cell}}^{\text{EM}} = \left| \frac{E_{\text{cell}}}{\sigma_{\text{noise,cell}}} \right| \tag{2.4}$$

[16] Although most of this thesis could be extended to time that would add extra complexities.

[17] It should be mentioned that electrons are often reconstructed using a sliding-window cluster algorithm[30] although topo-cluster algorithm can be used. In the future there is talk of using another method known as super-clusters.

and the σ_{noise} is defined:

$$\sigma_{\text{noise}} = \sqrt{(\sigma_{\text{noise}}^{\text{electronic}})^2 + (\sigma_{\text{noise}}^{\text{pile-up}})^2}$$

where $\sigma_{\text{noise}}^{\text{electronic}}$ is found experimentally and $\sigma_{\text{noise}}^{\text{pile-up}}$ is defined through

$$(\sigma_{\text{noise}}^{\text{pile-up}})^2 = (\sigma_E \sqrt{N_{\text{minbias}}})^2$$

where σ_E is the root-mean-square of the energy in 1 minimum bias collision[18] and N_{minbias} is the number of minimum bias events

Topo-Cluster Algorithm

1. We place seeds at locations where cell significance is above a seed threshold ($\zeta_{\text{cell}}^{\text{EM}} > 4$). These are referred to as *proto-clusters*. If any proto-clusters touch we combine them.

2. Next we look at the nearest neighbours (currently this is defined as the cells overlapping in η, ϕ, and adjacent layers) and add neighboring cells above the neighbouring threshold ($\zeta_{\text{cell}}^{\text{EM}} > 2$) to the proto-cluster. Cluster growth will stop once all the outer cells are below the neighbour threshold. The cluster is completed by adding neighbouring cells above a final threshold, which is set to zero.

3. If, during this expansion, a proto-cluster becomes nearest neighbour to another proto-cluster, they are combined into one proto-cluster.

4. There is a splitting step that splits local maxima that have a energy above a threshold of 500 MeV into two clusters. There are some extra details to splitting such as the ordering in which clusters are split. These details (all in the topo-cluster paper[28]) become important when multiple splits are necessary.

5. Finally these clusters are calibrated using 4 separate steps.

One extra detail: Negative clusters are produced because of the absolute symbol in Equation 2.4. These negative clusters are used to evaluate the prevalence of noise clusters.

[18] Minimum bias event refers to a collision where minimal filtering or triggering was used.

This summarizes the five steps for the formation and calibration[19]:

1. **cluster formation**
    - implicitly suppresses both electronic and pile-up cell noise by using cell significance
    - includes both positive and negative cells to avoid bias from energy fluctuations
2. **classification**
    - Each cell in a topocluster is split into EM and Hadronic components.
    - This classification is based on a likelihood model from pions based on: cluster depth, η, cluster energy, and cluster density.
3. **calibration (hadronic scale)**
    - the calorimeter uses the EM scale so the EM calibration factor is 1
    - hadronic calibrations are done for individual cells using a lookup table created from pion simulations.
4. **out-of-cluster**
    - This calibration tries to capture nearby smaller energy clusters that were missed in the topo-cluster.
    - This could be affected by pile-up.
5. **dead material**
    - corrects for material of the detector that is inactive and not calibrated for. Applied when the solid angle[20] of cluster overlaps regions of dead material.
    - leakage of cluster energy is calculated here in the last calorimeter cells and is based on cluster η, energy, and depth.

One thing to notice is that these calibrations can be influenced in a number of ways by pile-up.

[19] A detailed impact of these correction can be found in the thesis[31] of topo-clusters.

[20] More precisely based on $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2}$ overlap of cluster and dead material region.

2.3.4 Pile-up And Upgrade

One way to increase instantaneous luminosity leads to an increase in simultaneous proton-proton interactions per bunch-crossing, also referred to as in-time pile-up (μ). This can be seen from the luminosity formula[32]:

$$\mu = \frac{L_{\text{instant}}\sigma_{inelastic}}{n_{\text{bunch-bunch}}f_{rev}}$$

Where $n_{\text{bunch-bunch}}$ is the number of colliding bunch pairs, f_{rev} is the revolution frequency, and $\sigma_{inelastic}$ is the cross section of inelastic processes or total cross section. However, pile-up leads to some extra difficulties within the detector. These difficulties stem from drift-time of the ionization (out-of-time pile-up) and overlapping energy deposits (in-time pile-up). The pulse shape described in Fig. 2.9 helps remove extra energy deposits by canceling out-of-time pile-up with in-time pileup. This will lead to the total energy in the calorimeter not increasing with pileup, but fluctuations in the deposited energy, i.e. noise, do increase. The noise introduced is more complicated as nearby cells will be correlated causing noise signals that appear similar to the signals associated with particles from the proton-proton collision of interest.

The LHC and ATLAS detector[3] are undergoing upgrades that will increase the luminosity by a factor of ten; this will increase the pile-up. With the current topo-cluster implementation energy and position will be more difficult to resolve. This might be solvable by altering parameters within the topo-cluster algorithm. However the topo-cluster algorithm is dependent on the total signal energy deposition within the calorimeter. This gives an opportunity to look back and discuss whether in this new environment a new algorithm should be used. Returning to the human eye analogy perhaps an algorithm based on the human eye can be helpful for this problem.

The convolutional neural networks are algorithms used in visual identification tasks and we will be testing them out as an alternative. The next chapter will outline how these networks work.

Chapter 3

Machine Learning Methodology

3.1 Supervised Machine Learning

Supervised machine learning is a form of machine learning in which all of the information is labeled information, this to contrast to unsupervised machine learning.[1] Artificial Neural Networks (ANNs) are based on an early understanding of the mammalian brain. It was thought that a large number of neurons that are connected by synapses could become a general problem solver. This thesis will focus exclusively on a branch of ANNs named forward-feeding supervised training networks. We will focus on their usage as a parameterized transformation. These can be viewed as a rapid application of a simplified scientific method:

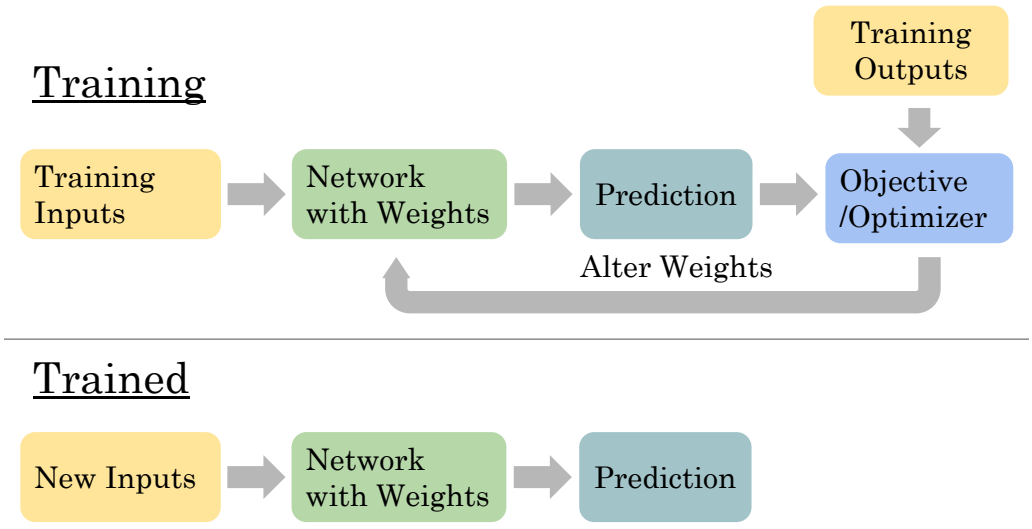
Heuristic neural network algorithm:

1. Create hypothesis
2. Use data and hypothesis to form a prediction
3. Compare prediction with target
4. Alter hypothesis to improve the prediction
5. Return to step 2

[1] In practice, unsupervised machine learning often refers to a system where the training set isn't clearly labeled or defined.

This next chapter will be breaking down this heuristic idea into a formal algorithm. We will start by breaking down the heuristic idea above into three separate topics: training sets (data), memory structure (hypothesis), and improvements (altering the hypothesis). This will be followed by the specific implementation[2].

Figure 3.1: **Schematic Overview of Supervised Learning**: This highlights some of the common stages of machine learning and highlights the difference between the training and trained stages. A key note is that once trained an input to the network will have exactly one output (non-random).



3.1.1 Training Sets

Training sets are a set of pairs.[3] These pairs represent an answer key with an input-output that is considered correct. If the input was the question, the output value would be the correct answer. The simplest form of a training set could be two lists of n values. Using a vector Y as outputs and a vector X as inputs we can define our training set:

$$\text{Training Set } = (Y_i, X_i)$$

For clarity individual samples can have a different dimensionality:

$$\text{1st Training Example } = (y_k, x_j)$$

[2]This chapter is covered in [33, 34, 35, 36] although I would strongly suggest [33] for learners because recent discoveries invalidate certains sections of older references.

[3]a notation aside: x are scalars, X are vectors, and $\mathbf{X}$ are rank 2 or higher matrices

As an example a training example might be given a picture of a cat in the form of an array as its input component and the word 'cat' for its output component.

A few key ideas are that the training set is usually a subset of a more complete set

$$\text{Training Set} \subseteq \text{Complete Set}$$

and there is the expectation that they are approximately equivalent in the region of interest

$$\text{Training Set} \approx \text{Complete Set}_{\text{Within region of interest}}$$

If a large subset of the complete set is missing, it may bias the learning.

Pre-Processing

Preprocessing of the target and input is a step that is generally taken prior to training. While it isn't always a requirement there are some papers suggesting that if the target and input are preprocessed to stay inside of a unit square or sphere this helps avoid potentially divergent networks.

3.1.2 Memory Structure

Neural networks have memory structures set up for efficiency and flexibility.

1. In recent times matrix operations and tensor operations have been highly optimized for machine learning with GPUs. This means that an efficiency is gained by using a matrix/tensor structure.

2. Tensors and matrices are commonly used to transform between geometric spaces so if our target and input set don't lie in a space with equal dimensions tensors and matrices allow for this flexibility.

For these two reasons the memory structure is usually represented as a set of arrays/tensors separated by non-linear functions[4]. It is customary to call the values of the arrays/tensors: *weights* or parameters, and the non-linear functions: *activation functions*. Activation functions generally treat input values above and below zero as being active and inactive respectively. These structures are separated into layers each

[4]the reason for non-linear functions we'll be addressed.

with a set of weights[5] and a non-linear function that can be written as

$$\text{output}_j = \text{activation function}\left(\sum_{i=1}^{n}(\text{weight}_{ji})(\text{input}_i)\right) \tag{3.1}$$

Translating this to more concise variables we get:

$$z_j = \Theta\left(\sum_{i=1}^{n} w_{ji}x_i\right) = \Theta\left(t_j\right) \tag{3.2}$$

Where Θ is an activation function, z_j are outputs of the layer,[6] x_i are inputs, w_{ij} are weights, and $t_j = \sum_{i=1}^{n} w_{ji}x_i$. An example of an activation function is the leaky ReLu function:

$$\Theta(t_j) = \begin{cases} t_j, & \text{if } t_j \geq 0, \\ at_j, & \text{if } t_j < 0. \end{cases}$$

With $a < 1$. There is no reason that the output of this layer cannot become the input

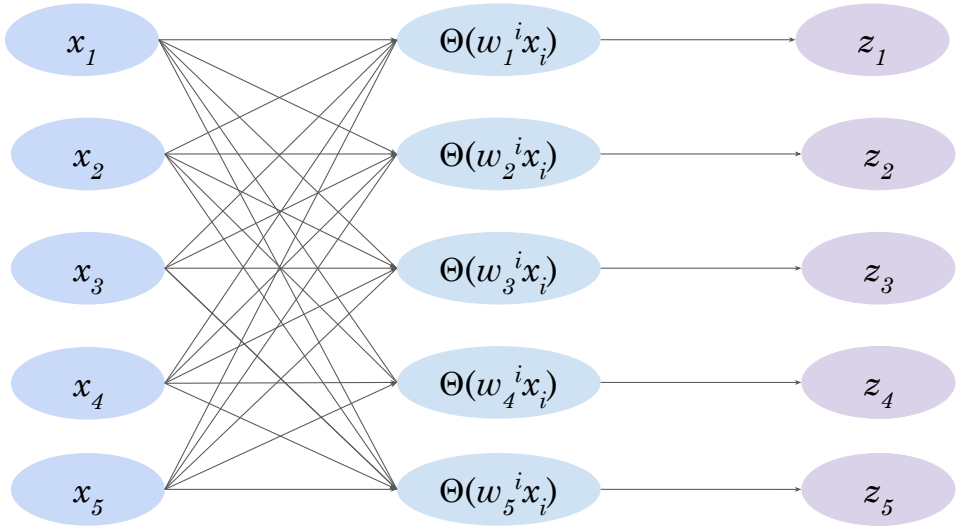


Figure 3.2: **Artificial Neural Network Single Layer**: Single layer neural networks share many common traits with linear regression models.

to a new layer. This may even be preferable due to layers acting as intermediary transformations. An analogy can be made with infinitesimal transformations being

[5]There is often a single weight added before the activation function. This term is known as the *bias* term.

[6]This is not the same as the training sets output.

able to represent more generic transforms. Put another way: $z_j = x_j$.

$$z_k = \Theta_1 \left(\sum_{j=1}^{n} w_{kj} \left(\Theta_0 (\sum_{i=1}^{n} w_{ji} x_i) \right) \right) \tag{3.3}$$

These equations will start to become very bulky so we will change to Einstein notation.[7]

$$z_k = \Theta_1 \left(w_k^j \left(\Theta_0 (w_j^i x_i) \right) \right) \tag{3.4}$$

It should be clear now that an arbitrary number of these layers can be connected. This leaves us with a flexible structure that can be easily stored.

Why non-linear functions are used: Non-linear functions are used to 'break' linear maps which would allow the tensors and array to be contracted to a single tensor. Let's look at this using two layers and linear functions.

1. $z_k = \Theta_1 \left(w_k^j \left(\Theta_0 (w_j^i x_i) \right) \right)$
2. set $\Theta_1 = \Theta_0 = \mathbf{1}$
3. $z_k = w_k^j (w_j^i x_i)$
4. Notice that $w_k^j w_j^i$ is just matrix multiplication that could be represented as w_k^i
5. Therefore two layers becomes one large layer when using linear functions.

Another thing to quickly discuss is why multiple layers are shown. It has been shown that single layer neural nets can solve linear classifier problems; however, larger networks have been shown to solve both: more complicated problems and linear classifier problems more accurately[37, 38, 39].

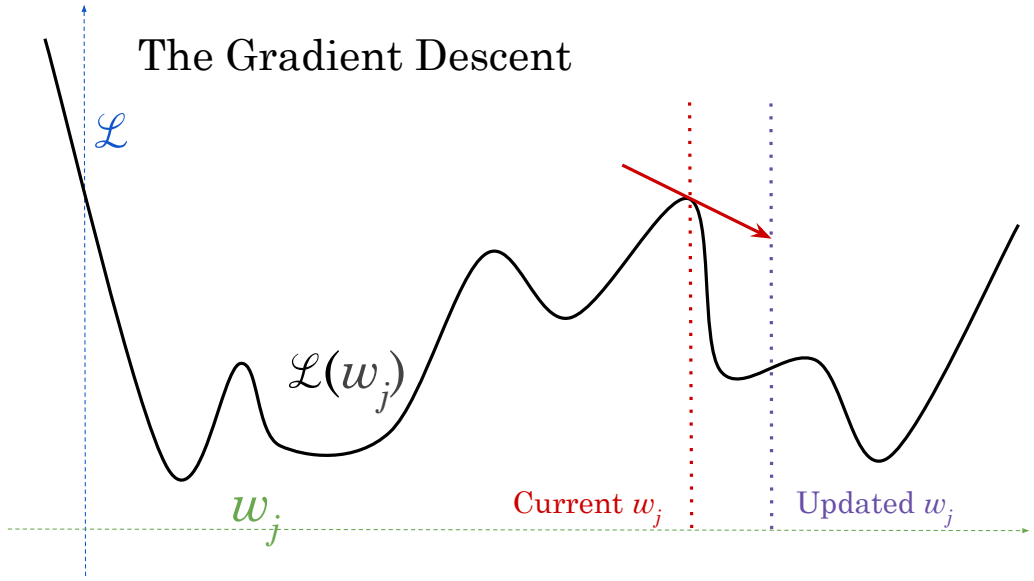
3.1.3 Improvements

Now we need to find a method for improving our structure so that the output of our neural network more closely resembles the training output. This step can be separated into two pieces: a loss function and an optimizer.

A **loss function** (or objective function) is a function needed to quantify how well

[7] If you are unfamiliar with Einstein notation, we are short-handing $\sum_{i=1}^{n} w_{ji} x_i$ as $w_j^i x_i$.

Figure 3.3: **Illustration of Gradient Descent:** This diagram demonstrates how networks are improved. The gradient of the loss is calculated for the current w_j and is used to update the w_j towards the minimum. These jumps will converge towards local minimums. In actuality there are millions of weights ($x-$axis) and some of these weights are not clearly independent when using multiple-layers.



the network performed. The loss function should be a function of the training set's output and the output of the neural network. Another formulation of representing this is as a function of the weights and the training set. This highlights that the loss function itself is changed when the training set is altered.

$$\mathscr{L}(z_j, y_i) = \mathscr{L}(x_i, w_{jk}, y_i)$$

The loss function is often a simple function. An example of a commonly used function is the absolute difference function which can be written as:

$$\mathscr{L}(z_j, y_i) = |z_j - y_i|$$

An **optimizer** gives a set of rules on how this loss function should be used to update the weights. The simplest method is known as *gradient descent*. The gradient descent starts by calculating the slope of the loss function with respect to the weights, then the weights are altered proportionally. This is done by altering weights by a specific value α, known as the learning rate, multiplied by this slope and can be

represented by[8]:

$$\Delta w_{ji} = -\alpha \frac{\partial \mathscr{L}}{\partial w_{ji}} \tag{3.5}$$

It likely isn't immediately obvious how this term is calculated. A first step for clarity would be to break it down with **the Chain Rule**:

$$\frac{\partial \mathscr{L}}{\partial w_{ji}} = \sum_k \frac{\partial \mathscr{L}}{\partial z_k} \frac{\partial z_k}{\partial t_k} \frac{\partial t_k}{\partial w_{ji}} \tag{3.6}$$

Let's look at each of these terms separately:

$$\frac{\partial t_k}{\partial w_{ji}} = \frac{\partial (\sum_l w_{kl} x_l)}{\partial w_{ji}} = \sum_l \delta^j_k \delta^l_i x_l = x_i \delta^j_k$$

where δ symbols represent kronecker deltas. These symbols are 0 when their indices differ and 1 when their indices are the same. This effective "kills" the sum over l. This term represents the fact that larger values entering into the layer will alter the weights more significantly the small values.

For the next term we'll use the leaky ReLu activation function, we will define the derivative[9] as:

$$\frac{\partial z_k}{\partial t_k} = \frac{\partial z_k}{\partial (\sum_l w_{kl} x_l)} = \frac{\partial (\Theta(\sum_l w_{kl} x_l))}{\partial (\sum_l w_{kl} x_l)} = \begin{cases} 1, & \text{if } \sum_l w_{kl} x_l \geq 0, \\ a, & \text{if } \sum_l w_{kl} x_l < 0. \end{cases}$$

This gives us a distinct weight change for values that 'activate' the neuron.

To get further we'll have to look at a particular loss function, the absolute difference loss function:

$$\frac{\partial \mathscr{L}}{\partial z_k} = \frac{\partial (|z_k - y_k|)}{\partial z_k} = \begin{cases} 1, & \text{if } z_k - y_k \geq 0, \\ -1, & \text{if } z_k - y_k < 0. \end{cases}$$

How to calculate this last term is not as straight forward for multi-layered neural nets. This problem has a known solution called backward propagation of errors or *back-propagation* which will give us $\frac{\partial \mathscr{L}}{\partial w_{ij}}$ term for every w_{ij}. We will start by finding a recursive relation. This is done by looking at the loss function as a function of the weight values feeding into the next layer, which we will label with the ordered set

[8] Further discussion of optimizers can be found in the Appendix B.1.

[9] Let's just ignore that discontinuity at zero.

$L = (a, b, ..., z)$:

$$\frac{\partial \mathscr{L}(z_j)}{\partial z_j} = \frac{\partial \mathscr{L}(t_a, t_b, ..., t_z)}{\partial z_j} = \beta_j$$

By taking the total derivative of both sides of this equation we get

$$\frac{\partial \mathscr{L}(t_a, t_b, ..., t_z)}{\partial z_j} = \sum_{l \in L} \frac{\partial \mathscr{L}(t_l)}{\partial t_l} \frac{dt_l}{dz_j}$$

now remembering that $t_l = \sum_j w_{lj} z_j$ (reminder the derivative kills the sum and picks it out the same index)

$$\sum_{l \in L} \frac{\partial \mathscr{L}(t_l)}{\partial t_l} w_{lj} = \sum_{l \in L} \frac{\partial \mathscr{L}(z_l)}{\partial z_l} \frac{\partial z_l}{\partial t_l} w_{lj}$$

This gives us a recursive rule between layers:

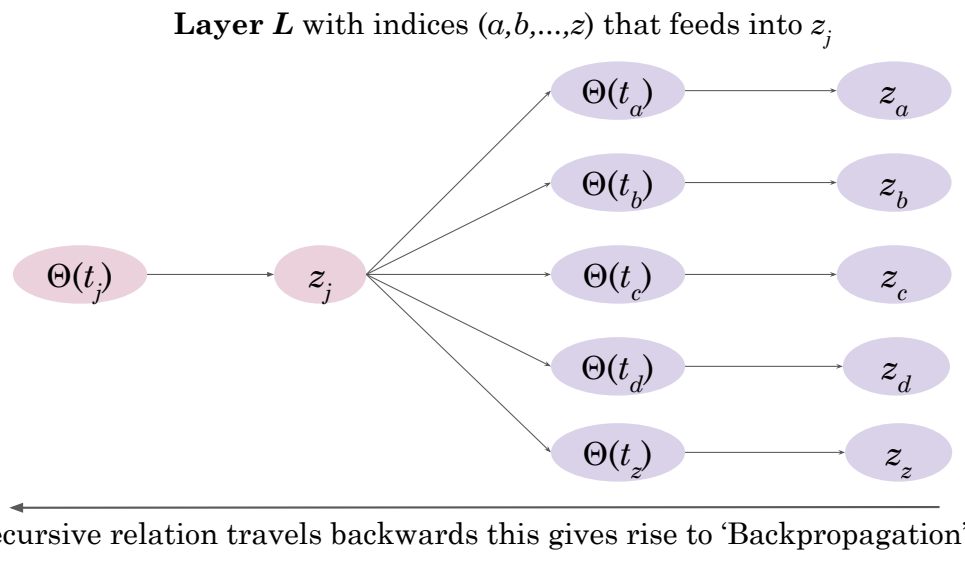
$$\frac{\partial \mathscr{L}(z_j)}{\partial z_j} = \sum_{l \in L} \frac{\partial \mathscr{L}(z_l)}{\partial z_l} \frac{\partial z_l}{\partial t_l} w_{lj} \tag{3.7}$$

We can rewrite this as:

$$\beta_j = \sum_{l \in L} \beta_l \frac{\partial z_l}{\partial t_l} w_{lj} \tag{3.8}$$

Let's put some thought into what this result means.

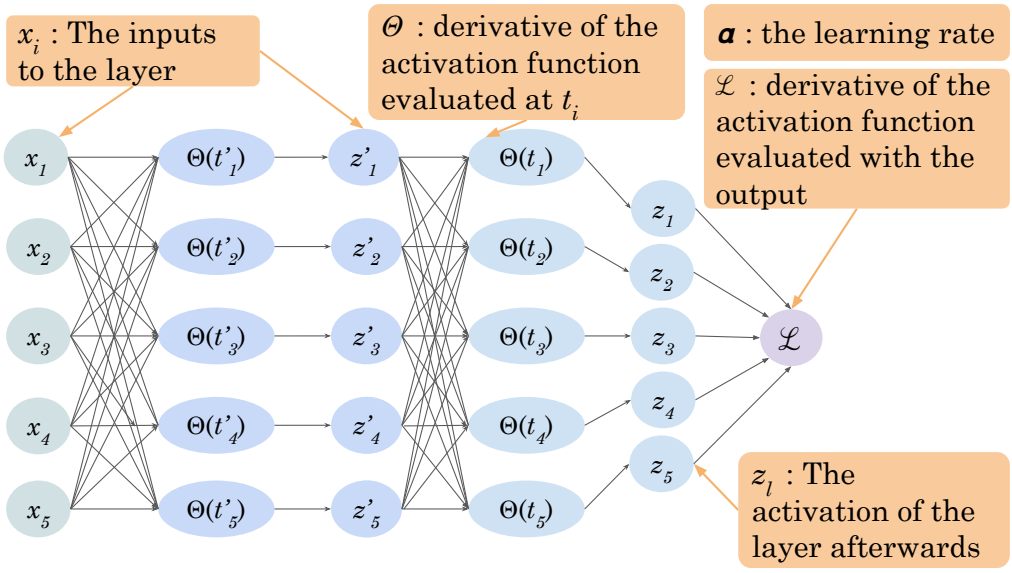
Figure 3.4: **Recursive aspect of Back-propagation**



The change in weights have four different sources:

$$\text{Change in weights} \longleftarrow \begin{cases} \alpha & \text{(learning rate)} \\ \frac{\partial z_k}{\partial y_k} & \text{(derivative of activation function)} \\ \frac{\partial t_k}{\partial w_{ij}} & \text{(original input into weights)} \\ \frac{\partial \mathscr{L}}{\partial z_j} & \text{(back-propagation term)} \end{cases}$$

Figure 3.5: **Decomposition of Weight Change**: Here we look at different aspects of back-propagation and how they impact the networks weight changes.



Let's go over each one of these sources separately:

$\boldsymbol{\alpha}$ So first off the weights will be altered more significantly if the learning rate is larger and change less if the learning rate is slower. It makes sense why this parameter is called the learning rate.

$\frac{\partial z_k}{\partial y_k}$ This project used primarily leaky-ReLu activation functions. This activation function causes active neurons to be altered significantly more than inactive neurons. Most activation functions monotonically increase so this number is generally positive.

$\frac{\partial t_k}{\partial w_{ij}}$ This term means that weights that have a larger input will be more likely to significantly change. This is useful because these weights are more significant to the network.

$\frac{\partial \mathscr{L}}{\partial z_j}$ This term largely means that weights that add more to the incorrect solutions will be altered more. This also give the sign of the change when the weight change should be positive or negative. The last layer remains the same.

This gives an idea of how each of the terms will affect the weight changes. There are a few additional common practices when working using back-propagation such as: preprocessing and adaptive learning[10].

Putting this all together we can get update rules for the weights using a Heaviside function, H, where the value at zero is defined as 0.

$$\Delta w_{kj} = \begin{cases} -\alpha x_k H\left(\sum_m w_{km} x_j\right) \sum_{l \in L} \beta_l \frac{\partial z_l}{\partial t_l} w_{lj}, & \text{if it is an inner neuron.} \\ -\alpha x_k H\left(\sum_m w_{km} x_j\right), & \text{if it is an outside neuron.} \end{cases}$$

This gives a straightforward although computationally expensive method to updating all of the weights.

3.1.4 Repetition and Batches

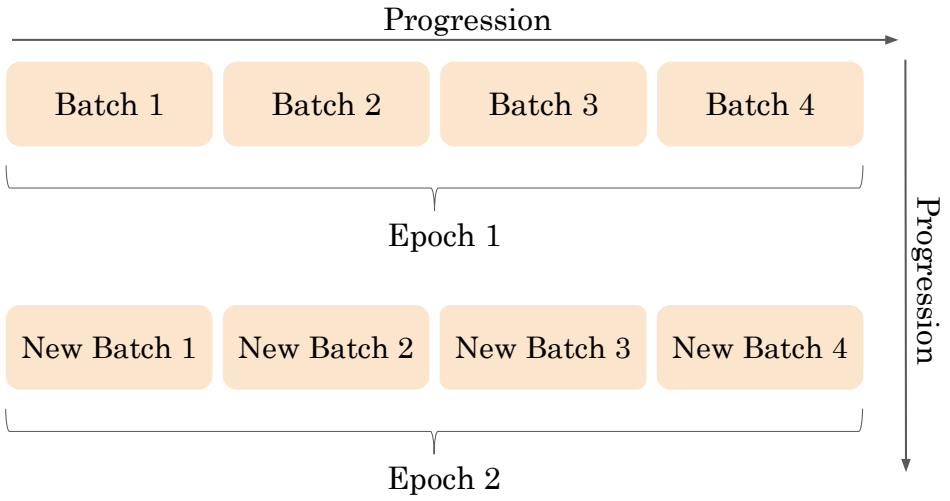
As a quick mention, so far we have discussed how to update the weights after a single training set. This is continuously repeated to improve our network for different training examples. We usually do training in batches so that we can avoid being biased by outliers examples; this can be seen as doing a simple average $\mathscr{L} = \frac{1}{n} \sum_i^n (\mathscr{L}_i)$. This can be seen in Fig. 3.6.

3.2 Convolutional Neural Networks

Convolutional neural networks have a different structure from the networks we just discussed. The inputs are organized in some fashion, let's use a 2d matrix as an example. The weights are organized into the same shape (or rank) but contain significantly fewer elements. These restricted weights known as kernels or filters are run across the input layers and then the kernels/filters are altered to improve the solution.

[10] This is explored in more detail in Appendix B.

Figure 3.6: **Training Schedule**: This outlines some of the common terms for training. Batches are a small set of examples. Epochs represent one entire cycle on the training set. The batches are mixed randomly after each Epoch. This is where randomness is introduced to the training of the network.



A convolutional layer with a kernel of size (r', s') can be represented by:

$$z_{kl} = \Theta\left(\sum_{i,j}\left(\sum_{s,r=0}^{r',s'} w_{rs}\delta_r^j\delta_s^i x_{ji}\right)\right) = \Theta\left(t_{ij}\right) \tag{3.9}$$

This might look more complicated but this greatly reduced the number of parameters which will lead to quicker training times. This also implies locally connected neurons because of the δs.

Usually multiple kernels are trained simultaneously to allow different convolutional shapes to propagate through the layers. Convolutional neural networks are invariant to spatial translations. This fact means that learning discoveries in one area of the image will translate to every area.

3.3 Residual Convolutional Networks and Mergers

One of the issues with back propagation and this pre-processing step is once enough layers are stacked the gradients near the start of the network can become vanishingly small. This reduction is due to the recursive nature of back-propagation[37]. There are a number of methods of overcoming these vanishing gradients. One method is

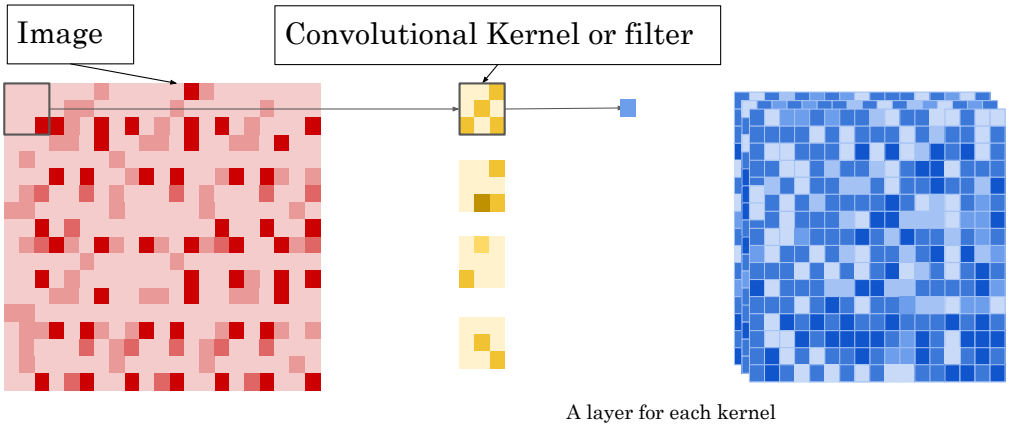


Figure 3.7: **Convolutional Neural Networks Structure**: The weights that are trained are the convolutional filters (shown in yellow). This represents a much smaller number of trainable weights than a full matrix transformation.

altering the structure of the network.

A common strategy is to add an identity layer that runs parallel to the convolution neural network layer. These layers are then merged through either concatenation or averaging. This allows larger values to travel through the identity layers and 'resolves' the vanishing gradients problem. In recent years, this has been shown to give a large improvement over standard convolutional networks[37, 38, 39]. This is one of the styles of networks that will be explored in this thesis.

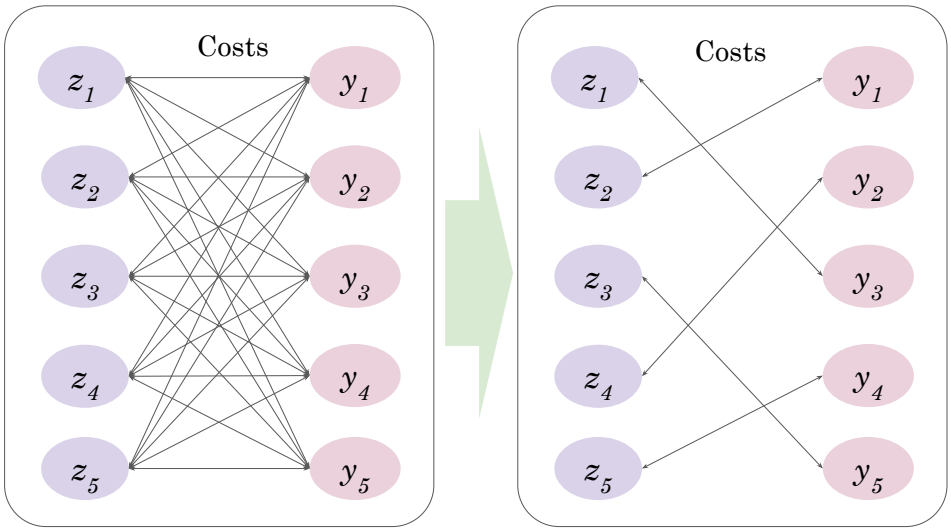
3.4 The Objective Function

Matching is an important subject that comes up because we are looking for multiple energy deposits. Because we have two different lists—one of predicted clusters and one of target or truth clusters[11]—both situated in (E, η, ϕ)-space[12], we need a method for matching them. There is an ambiguity when comparing two different unordered lists of uncertain locations. The assignment problem is a solved problem in the field of combinatorial optimization. It can be phrased as: "Given two equal sets

[11]Truth clusters are clusters produced from Monte Carlo simulation.

[12]This space contains the useful variables of a cluster: the energy and the location in coordinates described in Section 2.3.1.

Figure 3.8: **The Assignment Problem**: The assignment problem is the problem of optimally assigning one list to another list given an cost function.



of different objects and a cost that relates the two sets can you find the optimum one-to-one relation between the sets?"[13] There are a few known solutions to this assignment problem; one is the Hungarian/Munkres algorithm. This algorithm has a guaranteed convergence of $O(n^3)$ time. We'll define some notation before discussing this algorithm.

The cost between these two lists can be presented by a matrix:

$$\mathbf{C} = \begin{array}{c} \\ Z_1 \\ \dots \\ Z_\nu \end{array} \begin{array}{c} \begin{array}{ccc} Y_1 & \dots & Y_\nu \end{array} \\ \begin{pmatrix} c_{11} & \dots & c_{1\nu} \\ \dots & \dots & \dots \\ c_{\nu 1} & \dots & c_{\nu\nu} \end{pmatrix} \end{array}$$

So the problem would become finding the minimum cost list of length ν where each Y and Z are only used once.

A nice method of representing the solution is to define another matrix, M which we can call the matching matrix. We can write the solution as an element-wise multiplication of matrices (Hadamard Product):

$$\text{Solution} = \mathbf{C} \circ \mathbf{M}$$

[13] This can be seen as an exact matching to minimize a *weighted bipartite graph*.

Where the matching matrix will be an identity-like matrix with its rows and columns scrambled resembling:

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

This can then also be seen as a set of edges in graph theory. It also clarifies that the goal is to find an algorithm that can find a matching matrix that minimizes the the cost/objective function.

3.4.1 The Hungarian Algorithm

Hungarian Algorithm: This is a known solution for solving the Assignment Problem[40]. The solution can be broken down into steps:

1. Remove the minimum from each row by finding the minimum and subtracting that from each element of that row. The problem has been solved if there is a single zero in each column and row. $C'_{\mu\nu} = C_{\mu\nu} - \min_{\mu}(C_{\mu\nu})$
2. The step should be repeated for columns. $C''_{\mu\nu} = C'_{\mu\nu} - \min_{\nu}(C'_{\mu\nu})$
3. Cover all elements with a minimum number of lines.
4. Find the minimum uncovered element. Add it to every element that is covered. Elements that are covered twice get added to twice.
5. Subtract the global minimum from each element this should give a new set of zeros (note: previous line crossings won't be zero).
6. Covered again with the minimum number of lines. Now if the number of rows equals the number of lines continue to step 7, otherwise return to step 4.
7. Select matches by finding a zero element such that there is only one for each column/row. This gives M.

3.4.2 The Partial Hungarian Algorithm

There are a few small issues with the Hungarian algorithm. While it is $O(n^3)$ fast it isn't so quick at large n. Additionally matches made after a few recursive iterations are more "ambiguous" in the sense that these matches do not yield large cost minimizations. We can use a partial algorithm to only match the most significant matches to help speed this step up.

Partial-Hungarian Algorithm: The solution can be broken down into steps:

1. Use first two steps of the Hungarian algorithm.
2. Do as many matches as possible.
3. Finally rather than repeat we will simply connect the remaining clusters one-to-one match by their current ordering. This leads to a worse match early in training; however, because training is improving matching, this should lead matches that are comparable to Hungarian algorithm late in training.

This algorithm only relies primarily on basic linear algebra so it can be optimized to be quick.

3.4.3 The Cost Matrix

For our particular usage we will define a cost function for the matching algorithm. The simplest method would be to form a χ^2 type function[14]. We need to represent a variable number of clusters. We will represent this variable number by saying there are four maximum clusters and any fewer cluster will be labeled a missing or 'ghost' cluster. We will 'flag' these missing clusters by setting these clusters energy to zero and adding the maximum distance. This additional distance means that these 'ghost' clusters will be matched afterwards and because these matches do not depend on η or ϕ they will not be biased by the term $\frac{\partial \mathscr{L}}{\partial z_k}$:

$$c_{\text{true,pred}} = \begin{cases} E_{\text{true}} > 0 & \left(\frac{E_{\text{true}} - E_{\text{pred}}}{\sigma_E}\right)^2 + \left(\frac{\eta_{\text{true}} - \eta_{\text{pred}}}{\sigma_\eta}\right)^2 + \left(\frac{\phi_{\text{true}} - \phi_{\text{pred}}}{\sigma_\phi}\right)^2 \\ E_{\text{true}} = 0 & \left(\frac{E_{\text{true}} - E_{\text{pred}}}{\sigma_E}\right)^2 + \text{ Max Distance} \end{cases} \tag{3.10}$$

[14] when used for distance this is sometimes referred to as Mahalanobis distance

This function as well as the matching step introduces divergences to the gradients of this function because of the discrete jumps in the cost. The divergences should disappear later into training and they haven't caused noticeable problems with the analysis.

The specific values used in this thesis where:

$$\text{Max Distance} = 3$$

and

$$[\sigma_\eta, \sigma_\phi, \sigma_E] = [1, 1, 1 \text{ MeV}]$$

While not currently implemented some extra things we may want to add to the objective function are:

1. We are using (ϕ, η) over a restricted range. When expanding to the full range, ϕ should be projected the shortest direction around the circle.

2. Very low energy clusters could be treated differently.

3. Logarithmic or other functional forms could be used instead of squares. The square-root could be added to Equation 3.10 for distances.

Chapter 4

ATLAS Calorimeter Network Model

This chapter will start by describing the method of producing the training samples used by the neural networks. These samples[1] were produced using a pre-existing framework known as ATHENA. This section will highlight unique traits of this particular training set. This chapter will end with a discussion of the architectures of the networks used in Chapter 5.

4.1 Monte Carlo Samples

The ATLAS Monte Carlo is incredibly complex so this section will take some liberties for simplicity.

The production of ATLAS Monte Carlo samples[41] can be separated into 4 steps:

1. Generation
2. Simulation
3. Digitization
4. Reconstruction

[1] The traits of these samples were designed by the author to be simpler for machine learning Algorithms. They inluded a simpler underlying event (PARTICLEGUN and extra truth information (*CalHits*).

Generation

This step represents particles being produced in an idealized theoretical environment. This could be seen as the underlying Feynman diagram and is often referred to as the truth event.

We use samples that were generated starting with 4 electrons generated within the endcap region. More precisely:

- E uniform distribution between (1 GeV, 100 GeV)
- η uniform distribution between (1.425, 2.5)
  - We restrict ourselves in η for two reasons: the granularity varies significantly across η and we are particularly interested in the endcap—a section of the calorimeter that is expected to be more sensitive to pile-up. So naturally if we are interested in creating an algorithm that functions well at higher pile-up this is an ideal region.
- ϕ uniform distribution between ($\frac{3}{4}\pi$, π)
  - Our restriction in ϕ is for computational reasons. While prototyping, it was helpful to restrict ourselves to a smaller region[2].

Electrons are negatively charged and once they are simulated in the detector they will have a deflection in the -$\hat{\phi}$ direction in the forward endcap.

For clarity this doesn't represent a physics event. This was used to more clearly understand the networks reaction to the inputs. This was done with an internal ATLAS generator known as Particle Gun.

Simulations

This step propagates the generated particles traveling through the detector. This simulates and records the results of the interactions of the particles with the physical model of the ATLAS detector; this step includes particle showers, magnetic fields, energy depositions, and secondary particles produced. The energy depositions including noise deposits are referred to as HITs. In addition to the default settings,

[2]This also helped connect this model with earlier toy models.

the exact amount of energy deposited by each truth particle was kept. These energy depositions are referred to as calibration hits or *CalHits*. This was done using an ATLAS internal model and GEANT4[42]. This data excluded bunch structure variation over time because of this the pileup was 'constant' across samples (for this particular Monte Carlo data):

$$\langle \mu \rangle = \mu$$

Digitization

This step represents how the experimental setup actually records the energy deposits created from the Simulation step. This gives the first, low-level, readouts the detector would have. This is also where pile-up events can be added on-top of the underlying truth event[43] (in analogy to adding noise). Pile-up events are added from a large repository of LHC data samples with either high transverse momentum events or low transverse momentum events at a particular ratio (this ratio is a measured quantity). Four different pile-up sets where used $\mu = [0, 40, 120, 200]$. *CalHits* were also kept and are independent of pileup. This is done with custom ATLAS internal software[3].

Reconstruction

This step represents the procedure for reconstruction of truth events using only information from the experimental setup (i.e. trackers, calorimeter, muon Spectrometer, magnetic field). This step both builds final particle objects—such as electrons and photons—with detector noise removed and builds a set of intermediate objects such as topo-clusters and hadronic clusters. Because most truth information is from either the Generation or Simulation step they are not dependent on pileup whereas most of the objects created by the Reconstruction are highly dependent on pileup. This also uses custom ATLAS software. These reconstruction algorithms are the exact-same as those used on ATLAS data.

---

[3]This software is often referred to as ATHENA which is ATLAS's software framework. This is built on the older but open software framework GAUDI[44].

The Reconstruction and Simulation are generally the most expensive steps both computationally and time-wise. In contrast, the Generation step is the quickest step. The Reconstruction and Digitization steps are highly sensitive to pile-up.

The data was split into three sections: Training, Testing, and Validation. Testing samples where used to optimize networks whereas validation samples were blinded and only used during the final validation[4]. This is seen in Table 4.1. A few of the

Table 4.1: **Division of Data Samples:** Notice that because the samples where not produced with a pile-up structure each event can be produced in parallel.

Pileup(μ)	Training	Test	Validation	Reco Time per Event (approx)
0	9000	1000	1000	5 Sec
40	9000	1000	1000	30 Sec
120	9000	N/A	1000	1 Min
200	8997	N/A	1000	2-5 Min

$\mu = 200$ samples are missing due to processing failures.

4.1.1 Training Inputs: Calorimeter Cell Arrays

We are going to start by looking at the input data. In ($\phi = [3\pi/4, \pi]$, $\eta = [1.425, 2.5]$) space of the calorimeter can be described by a set of 7 rank 2 arrays, one for each layer of the calorimeter. The different granularities or image sizes do add some extra complexities. The information above and our knowledge of the calorimeters gives us three reasons why the calorimeter cannot immediately be thought of as a 3d image:

1. The layers are not the same size.

2. When we generalize to a full calorimeter/ detector there are about 64 granularities throughout the detector. The reason for these granularities comes from a constrained optimization problem of:

    - maximizing the sensitivity to hadronic and EM shower responses (which tend to spread laterally as they penetrate).

[4]This helps avoid overfitting the validation examples by altering hyper-parameters.

Table 4.2: **Calorimeter Image Sizes**: This table gives an idea of the number of calorimeter cells used in the networks. The Topo-cluster algorithm used all ≈188000 cells; however, all of the truth cluster were contained in the ($\phi = [3\pi/4, \pi]$, $\eta = [1.425, 2.5]$) region.

Layers	Image Size (η,ϕ)	Number of Pixels
Pre-Sampler EMEC	(11, 8)	88
EMEC1	(213, 8)	1704
EMEC2	(41, 32)	1312
EMEC3	(19, 32)	608
HEC1	(9, 8)	72
HEC2	(9, 8)	72
HEC3	(8, 8)	64
Total	N/A	3920

- while being constrained by a budget (high granularity calorimeter cells are expensive).

3. The calorimeter cells in different regions have different statistics.

There are a variety of machine learning approaches to deal with the non-uniform segmentation of the calorimeter. We'll start by treating each of our 7 granularities as 7 separate images.

Let's first look at some summary graphs of layer specific statistics so we can understand our networks inputs a little better. We'll start by looking at the mean cell energy per event and make a histogram with this set of events. The constancy of energy versus pile-up shows that the pulse shape shown in Figure 2.9 is functioning as expected. Next we'll look at standard-devation. We see that this STD rises with pile-up as expected. It is worth noting that the signal is being included which is why both the mean and the STD of the EMECs are significantly larger than the HEC. The HEC shows a larger sensitivity to pile-up; this is because the signal electrons don't leave much signal there.

4.1.2 Training Outputs: Truth-Clusters

Truth-Clusters are clusters made from the calibration hits.

1. The four initial particles are each considered separate roots to the directed tree that is created by the Monte Carlo simulation.

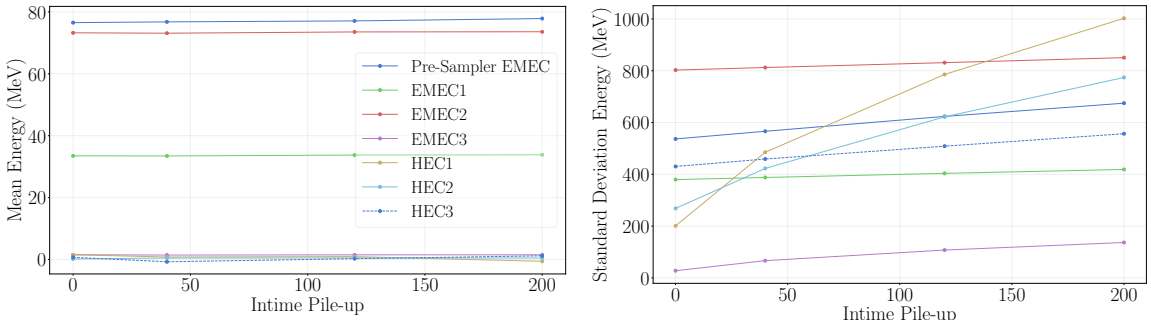


Figure 4.1: **Histograms of Mean and STD Energy per Event versus Pileup**: From top right to bottom left $\mu = [0, 40, 120, 200]$. These graphs include the signal electrons and the indicated amount of pile-up.

2. Each of the four directed trees have leaves that are calo-cells (energy loss in dead material is thrown out).

3. A center-of-energy type average is taken to get the ϕ and η of the truth-cluster.

This can be precisely described by:

$$E_{\text{true}} = \sum_{\text{cells}_i}^{m} E_{\text{cells}_i} \tag{4.1}$$

$$\eta_{\text{true}} = \frac{1}{E_{\text{true}}} \sum_{\text{cells}_i}^{m} \eta_{\text{cells}_i} E_{\text{cells}_i} \tag{4.2}$$

$$\phi_{\text{true}} = \frac{1}{E_{\text{true}}} \sum_{\text{cells}_i}^{m} \phi_{\text{cells}_i} E_{\text{cells}_i} \tag{4.3}$$

Because of the magnetic field and material within the physical detector, the uniform distributions created by the particle generators will no longer be valid for objects created from the calorimeter cells. We also maintained the same box described in Section 4.1.1 and set the truth values for any cluster outside this box to $(E_{\text{true}}, \eta_{\text{true}}, \phi_{\text{true}}) = (0, 1.9625, \frac{7}{8}\pi)$.

4.1.3 Network-Preprocessing

All seven of the input arrays are pre-processed identically. Events were converted into a geometric space more closely representing a unitary space for the networks. This

is to avoid both 'dying' and 'exploding' networks[5], which are caused by the recursive nature of back-propagation. The pre-processing attempted to fit all values into a sphere smaller than 1. Looking at Equation 3.8, we can see this when there is a large number of recursions[6] ($n \to \infty$):

$$\beta^n = \begin{cases} \text{if } \beta > 0 & \text{then } \beta^n \to \infty \\ \text{if } \beta < 0 & \text{then } \beta^n \to 0 \end{cases}$$

Table 4.3: **Coordinate Transformation for Neural Network**: We transform into a coordinate system more suitable for neural nets. This transformed our data to be mostly smaller than 1 and our positional data to be centered about zero as described in Section 4.1.3.

Experiment Space	Transformation	Network Space
ϕ	$\phi' = 8/\pi(\phi - 7/8\pi)$	ϕ'
η	$\eta' = 2/1.075(\eta - 1.9625)$	η'
E	$E' = \sqrt{E/10000 \text{ MeV}}$	E'

The energy transformation is to help with the Poisson term in the resolution described in Section 2.3.2.

[5]These terms refers to networks that deteriorate because of to many zero values ('dying') or extremely large values ('exploding').

[6]This is a little exaggerated as β values vary from layer-to-layer; however, a component comes from the layer above.

Energy Transformation: One useful thing to remember is that doing transformations on your space before training actually does have a different impact on the network than doing the transformation explicitly in the objective. This is because the derivatives in each space are not the same. Using an example where $\sqrt{E} = E'$:

E Space	$\sqrt{E}$ Space
$\frac{\partial}{\partial E_{\text{pred}}}(\sqrt{E}_{\text{true}} - \sqrt{E}_{\text{pred}})^2$	$\frac{\partial}{\partial E'_{\text{pred}}}(E'_{\text{true}} - E'_{\text{pred}})^2$
$= -2(\sqrt{E}_{\text{true}} - \sqrt{E}_{\text{pred}})\frac{1}{2\sqrt{E}_{\text{pred}}}$	$= -2(E'_{\text{true}} - E'_{\text{pred}})$
	Converting back into E space
$= \frac{\sqrt{E}_{\text{true}}}{\sqrt{E}_{\text{pred}}}$	$= 2(-\sqrt{E}_{\text{true}} + \sqrt{E}_{\text{pred}})$

Interestingly one of these functions diverges as $\sqrt{E}_{\text{pred}}$ it approaches zero and the other does not. For clarity our networks work in E' space.

No other form of preprocessing or regularization were used.

4.2 Hyper Parameter Scan

Because 3d computer vision with multiple resolutions for multiple objects regression is not a perfectly solved problem, it is good practice to try a variety of hyper parameters to cover different styles of networks. While we can use knowledge from 2d computer vision problems, it is important to retest some of our intuitions.

This was done by preparing 50 different architectures with different merging and mapping styles. A python library known as Keras [45] was used which is a wrapper that was using the machine-learning library Theano[46] as a backend. Twenty of these networks didn't make it past 5 epochs dues to timeouts or memory errors. From the twenty, fifteen didn't seem to converge and were dropped. The other five networks showed some promise although due to slowness they are impractical. This left 30 networks which finished training within reasonable time and memory limits.

Of these 30 networks, four networks will be used: one with the lowest training loss (Network D), one with the lowest test loss (Network B), and two were chosen (Network A and Network C). Network A was chosen because it gave small loss and trains very quickly. Network C was chosen because of its structural similarities to Network B while having a very different loss values. These networks are shown in Table 4.4. Figures 4.6 explore the 'depth' of these networks which is more difficult to define due to the number of merging layers.

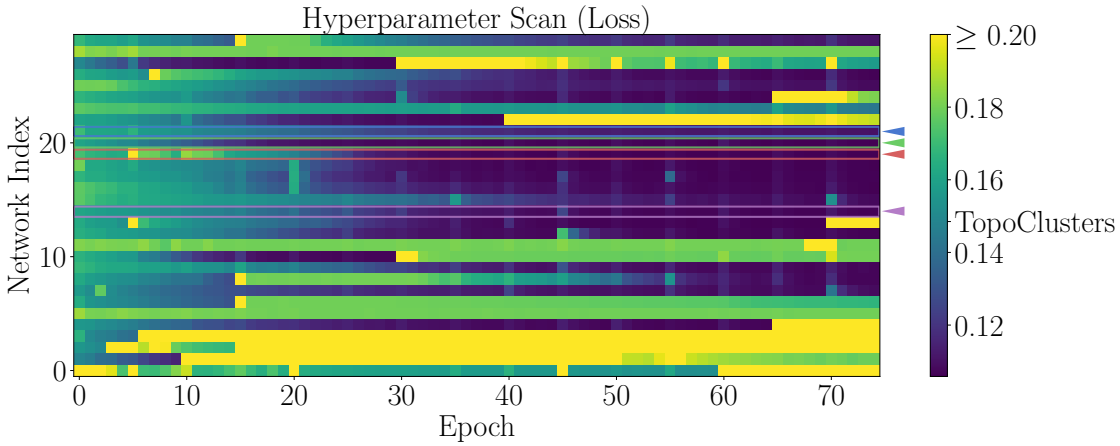


Figure 4.2: **Hyper-parameter Scan (Training Set)**: Scan of 30 network structures. This shows the loss at the end of each epoch. Four networks were picked from this scan; they are highlighted by coloured boxes and arrows to the right. From top to bottom these networks are named: Network A, Network B, Network C, and Network D. The yellow streaks (≥ 0.2) represent either: a network that exploded (this relates to the ∞ in Section 4.1.3) or a network that failed due to time or memory constraints. Both of these conditions are disqualifying for a network.

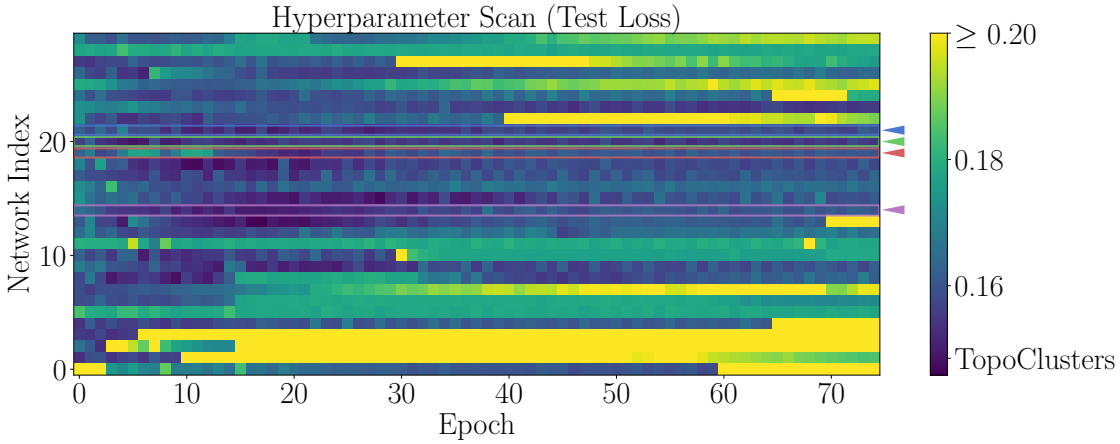


Figure 4.3: **Hyper-parameter Scan (Testing Set)**: Scan of 30 networks structures. This shows the test loss at the end of each epoch. Boxes highlight the networks picked. There is a dip in test loss values around epoch 15 for the network indexes 12-20. This implies that the minimum of test loss happens around 15 epochs for most networks.

4.2.1 The Finalized Networks

Let's discuss some of the networks that had the lowest loss and their structure. These networks range from the simplest Network-A shown in Figure 4.4 to complex Network-B shown in Figure C.8. They all contain comparable numbers of weights.

Table 4.4: **Network Specifics**: Notice that training time is not entirely dependent on the number of parameters. More merges and convolutional neurons are more computationally expensive. Training time are approximate (based on CPUs). This was accomplished with running on around 100 CPU cores with the Cedar Cluster. Extra information about the computation resources used can be found in Appendix C.

Network	Number of Parameters	Approx. Training Time (80 Epochs)
Network A	18026001	12 Hours
Network B	17070037	26 Hours
Network C	13056037	22 Hours
Network D	11271697	36 Hours

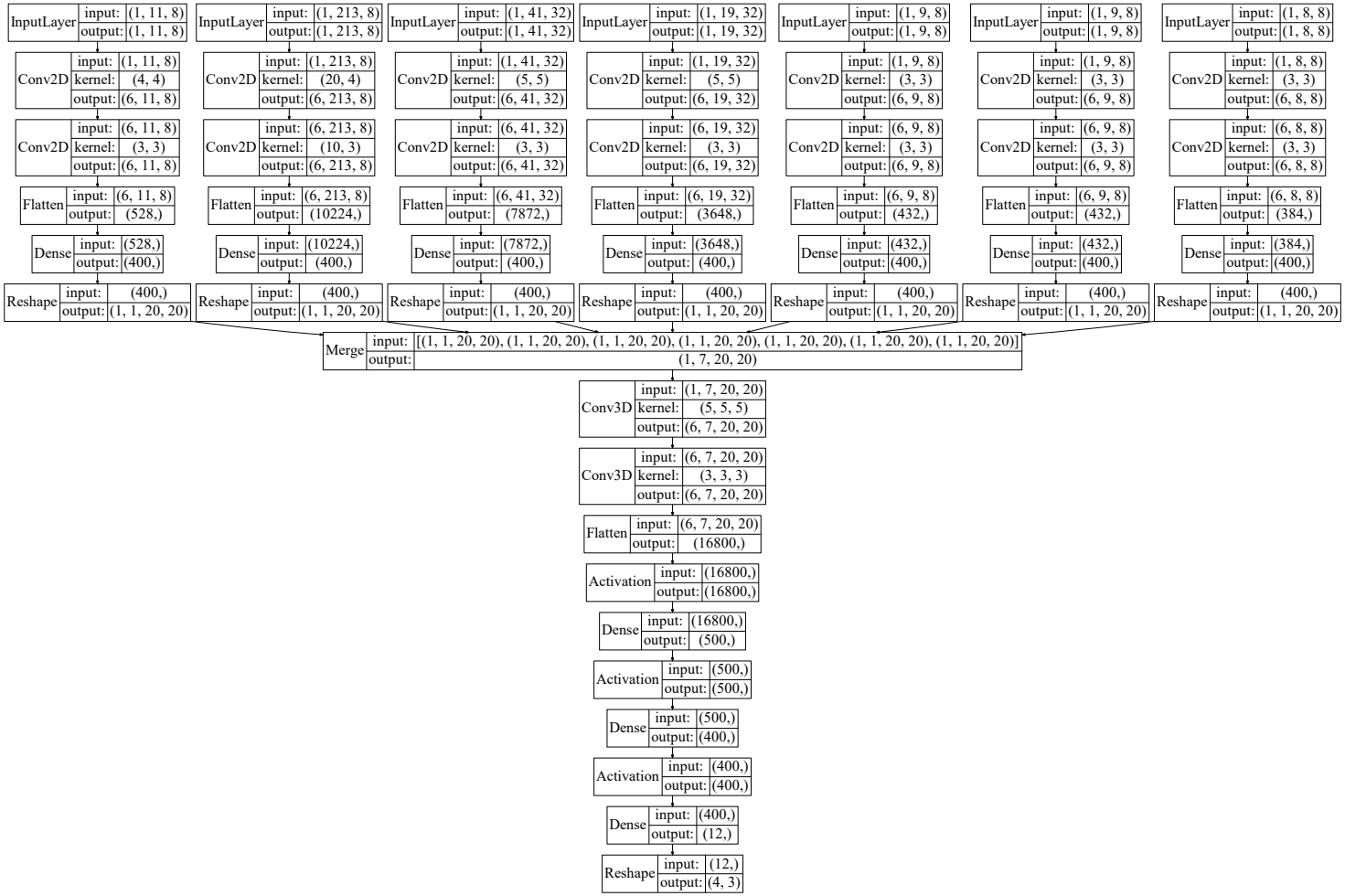


Figure 4.4: **Network-A** is the simplest network in terms of layers. The 2d convolutional layers are created to mirror the dimensionality of the input. Convolutional Layers are followed by a *Leaky-ReLu* activation function with negative slope of 0.1. Bias terms were used for all layers. Padding is used to maintain the dimensionality of layers. This network's weights were initialized with Glorot uniform distributions[47]. The *Nadam*[48] optimizer was used. This graph is structured using the standards defined in the KERAS Documentation[45].

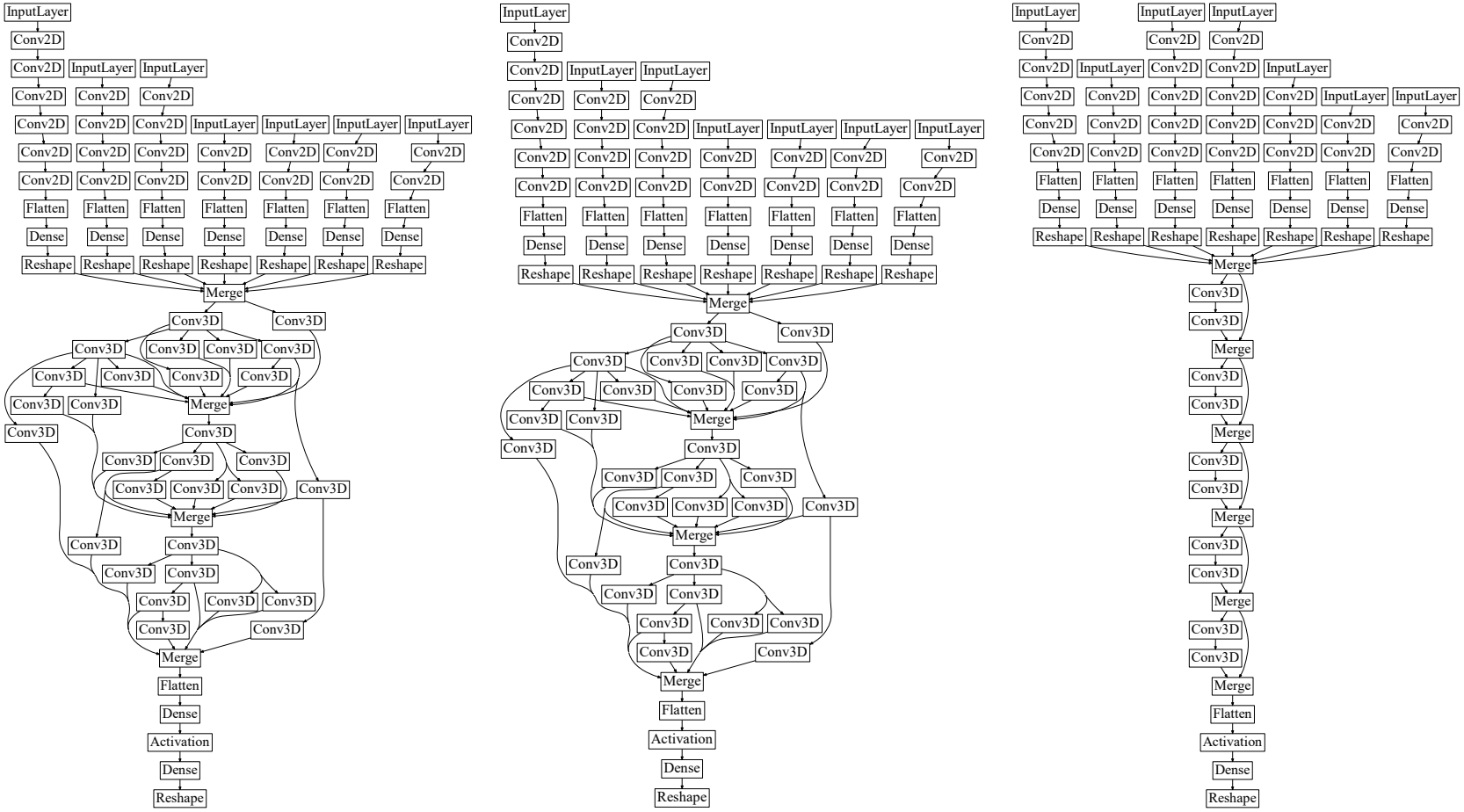


Figure 4.5: **Network-B, Network-C and Network-D** all contain more complex layerings. They all make use of merger layers which causes them to have more paths through the network. Because of this merging all of them require more memory and more time than network-A. Network-C is Network-B with the last very large dense layer removed from the end. Network-D contains a residual-like structure that has become popular in recent years. A more detailed description of Network-B, Network-C, and Network-D in the appendices as Figure C.8, Figure C.9, and Figure C.10 respectively. This graph is structured using the standards defined in the KERAS Documentation[45].

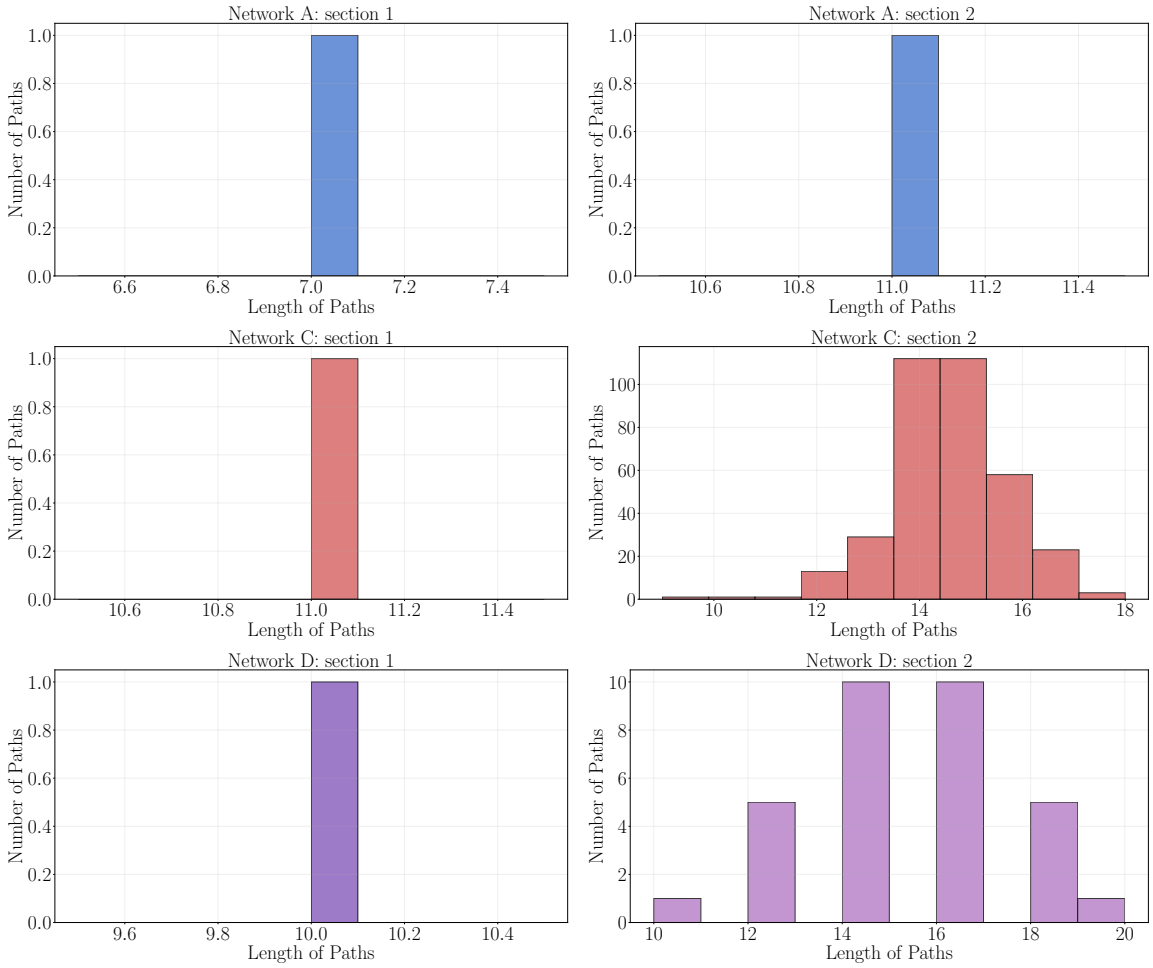


Figure 4.6: **Network's Path and Path Lengths**: A large difference between these chosen networks is the layer structure. This graph highlights both the number of paths through and the 'depth' of each network. Each length has been associated with a different number of back-propagation iterations. These graphs are divide into the 2d section (section 1) and the 3d section (section 2) of the network. Recent networks have a tendency to favor more paths through the network. This shows that Network C and Network B have the most paths.

4.2.2 Commonalities of Poor Networks

Poor performance[7] was associated with the following:

- Networks that had a very large number (30-100) of convolution filters tended to perform poorly initially.

[7]Poor performance is defined by large training loss and large testing loss.

- Networks that had 80-100 layers of depth. This *may* have been caused by a lack of regularization.
- Networks that had many merge layers before the main 2d to 3d merge.

  This could be due to a very large number of paths for back propagation.
- Networks that had a single convolutional filter size.

  This doesn't support VGG-style networks[8]

Here are some of the ways these networks *might* be improved:

- Intermediary layers should be regularized.
  - particularly of the $|L|^2$ variety. This pushes the weights towards a unit sphere and has been suggested for networks that are deep.
- Merging layers should not be product merged.
- Networks with large number of filters should be trained longer.
- Divergent networks could be eliminated by using a different optimizer.

The most significant lesson is that very deep networks require significantly much more data.

[8]These are networks that only use 3×3 filters, with the hypothesis that enough layered 3×3 filters effectively form any filter size.

Chapter 5

Evaluation, Analysis and Comparisons

5.1 Comparison Method

The comparison is rather complex (summarized in Fig. 5.1) and must be done for both the validation and training sets. The end goal is to arrive at a comparison of the topo-clusters and neural-networks versus pile-up. We'll start by comparing both topo-clusters and neural-networks against truth-clusters:

1. The two sets of clusters—one of which is truth-clusters—are matched event by event. The matching is done by minimizing the root-mean-squared distance in the space $(E/50\text{ GeV}, \eta, \phi)$. This matching is done using the full Hungarian method. With these cluster matches the differences between clusters $(\Delta E, \Delta\eta, \Delta\phi)$ are recorded.

    - For the topo-clusters only the 4[1] most energetic clusters are compared. This helps deal with the multiplicity of clusters below 25 GeV.
    - For the Neural Network clusters there only are 4 clusters by construction.

2. The events are combined together and binned into histograms for ΔE, $\Delta\eta$, and $\Delta\phi$. From these data sets—ΔE, $\Delta\eta$, and $\Delta\phi$—the mean and standard deviation were found.

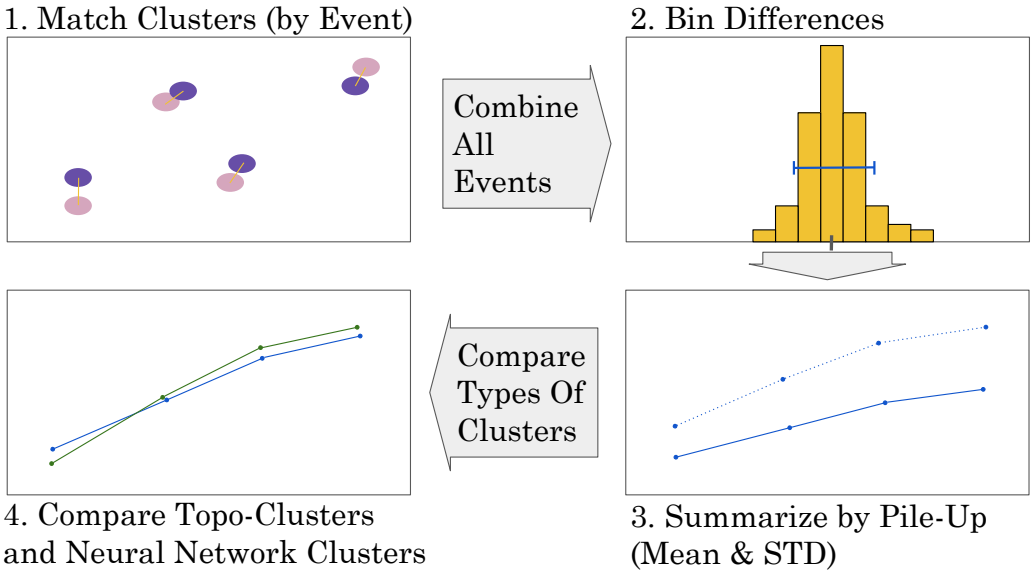
[1] This restriction of four clusters that was made for algorithm development and was intended to eventual relax. Four cluster was chosen because many important physics channels the produce four clusters—such as the 4 lepton channel.

- Matchings where the truth clusters are below 25 GeV are filtered out. Very low energy clusters are not particularly meaningful because of the ghost clusters discussed in Section 3.4.3.
- The complete set[2] (without filters) has 36000 truth-clusters for the training sets and 4000 truth-clusters for the validations sets .
- Improvements in the algorithms will cause the the standard deviation and mean to tend towards zero, although there is a minimum value for the standard deviation set by imperfections in the detectors and ultimately quantum mechanics.

3. The mean and standard deviations are plotted versus pile-up. The mean is referred to as the calibration and the standard deviation is referred to as the resolution.
   - It would be expected that the standard deviation should increase with pileup and the mean should remain relatively constant (for both the neural-networks and the topo-clusters).
4. Finally the results of previous steps will be compared for the topo-cluster and neural networks clusters.
   - Better performance is indicated by a smaller standard deviation and smaller absolute mean in ΔE, $\Delta\eta$, and $\Delta\phi$.

[2]Combining all pile-up samples $\mu = [0, 40, 120, 200]$

Figure 5.1: **Summary of the comparison process**: This figure shows a summary of the comparison process and does not represent actual results.



5.2 Topo-Cluster versus Truth-Cluster Comparison

Here our set of topo-clusters are compared directly against truth-clusters. Topo-clusters are spit into three sets by energy filters[3]: values less than 25 GeV, values greater then 25 GeV, and all energies. These differences are made as:

$$\Delta\text{Values} = \text{Topo-Cluster Values} - \text{Truth-Cluster Values}$$

These comparisons are summarized in Figures 5.3–5.5.

Comparison of topo-clusters with previous studies

Our histograms in Figure 5.2 have long tails. These tails originate from a number of effects[4] but predominately they come from the simplicity of the definition of truth clusters in Section 4.1.2. This truth cluster definition treats special cases identically. For instance, the truth-cluster's location might end up between a hard radiated photon and the electron. Improving the definition of truth clusters is a possible area

[3]These filters were chosen to split a clear population difference if Figures C.5.
[4]such as hard radiative photons and pair production.

for development in the future. Returning to the comparison, because of these tails as well as tails in other experiments, these distributions are difficult to compare to previous measurements using the standard deviation.

A simple method for removing tails is to take the Full-Width-Half-Maximum (FWHM) and then treat it as if it has a Gaussian shape to calculate its Gaussian standard deviation:

$$\text{FWHM}_{\text{with Gaussian shape}} = 2\sqrt{2\ln 2}\sigma \approx 2.355\sigma$$

Tables 5.1 shows a comparison of topo-cluster resolution based on FWHM with results from previous studies. The rest of this chapter's comparison was done by comparing

Table 5.1: **Comparison with Previous Results**: These are **rough** comparisons as the data sets being compared vary in a number of ways. This comparison is done with results from *Electron Performance*[49] and *Expected Performance*[29]. *Electron Performance* used 2010 data and applies for a range from $1.52 < |\eta| < 2.47$ and *Expected Performance* are Monte Carlo where energy resolution was taken at $\eta = 1.075$ and position resolution was taken at 100 GeV. Our data also included the infamous "gap" in the detector where performance is degraded by dead material (cables, mechanical supports) between the endcap and barrel.

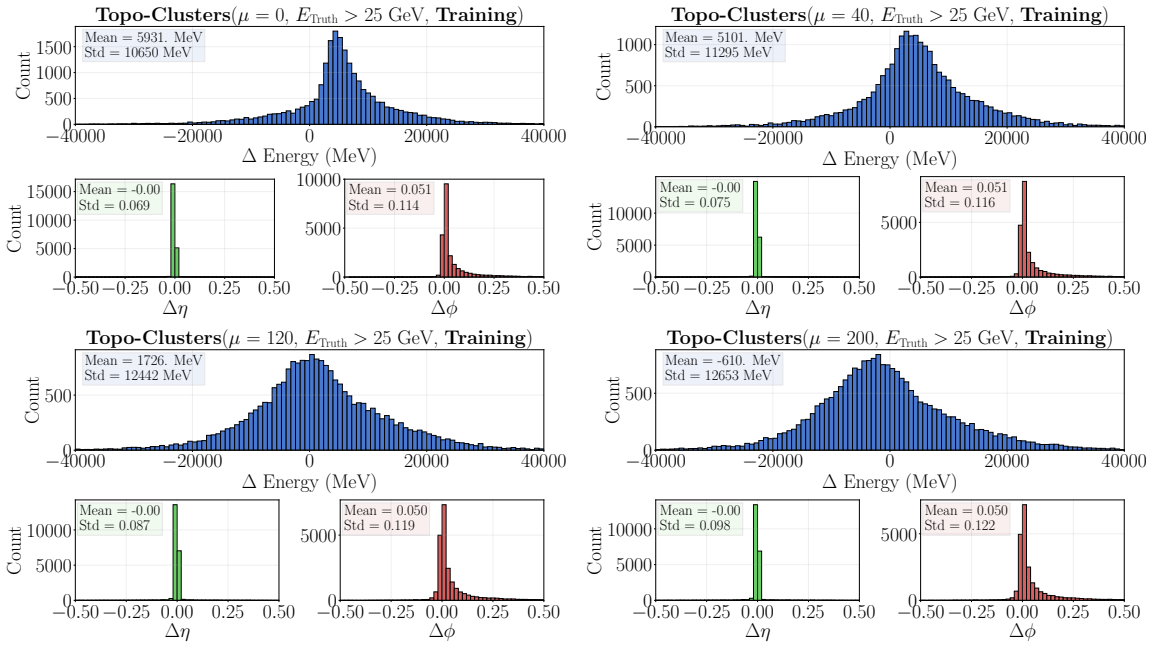
	Our Topo-Clusters ($\mu = 0$)			
Variable	FWHM	σ_E/E[a]	Monte Carlo(%)[29]	Data(%)[49]
E	(80GeV)12/200	2.77%	1.66±0.08	1.8±0.4(stat)±0.4(sys)
Variable	FWHM	σ_{variable}	Monte Carlo	Data
ϕ	7/1600	0.0019	0.0016	0.0025[b]
η	3/1600	0.0008	0.0006	0.001[b]

[a] setting $E = 75$ GeV

[b] These are track-cluster differences so this shouldn't be taken too literally.

standard deviation rather than FWHM. This was done as the cost function discussed Section 3.4.3 is minimizing a term similar to the standard deviation. At the outset it was unclear whether the FWHM of neural networks resolutions would vary significantly from topo-clusters; however, they do. This can be seen in Figure C.7. By removing these tails from the truth-clusters we *may* improve the network clusters as well.

Figure 5.2: **Resolution graphs of Training Data** From left to right and top to bottom $\mu = [0, 40, 120, 200]$. All these values include an energy cut of 25 GeV or greater. These histograms are summarized in a linear graph below. The subsequent linear graphs throughout this thesis summarize these Gaussian-like shapes.



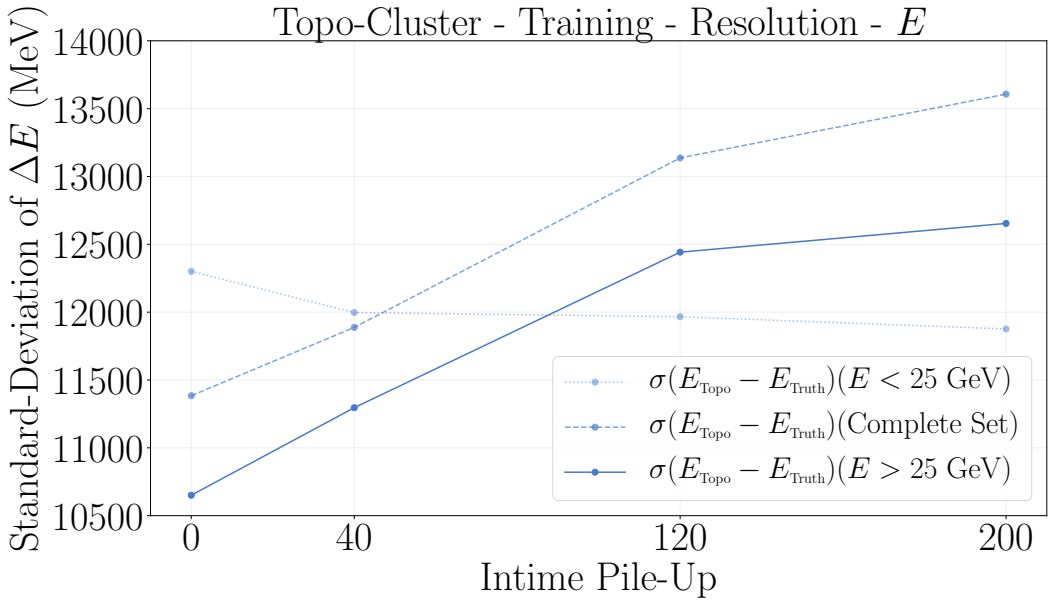
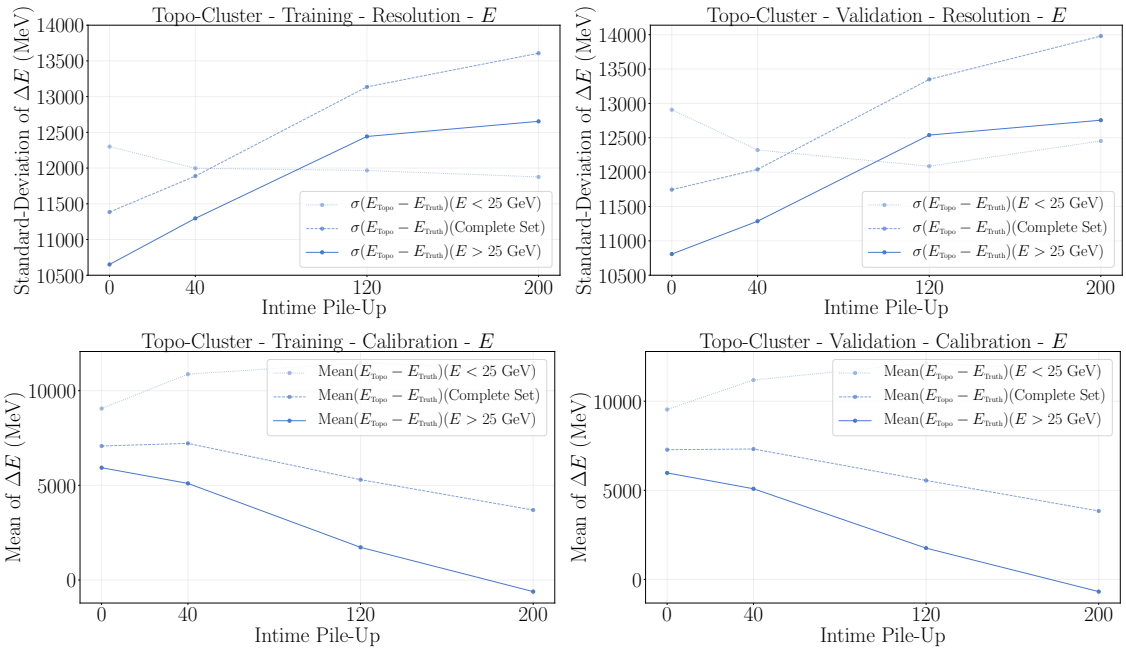
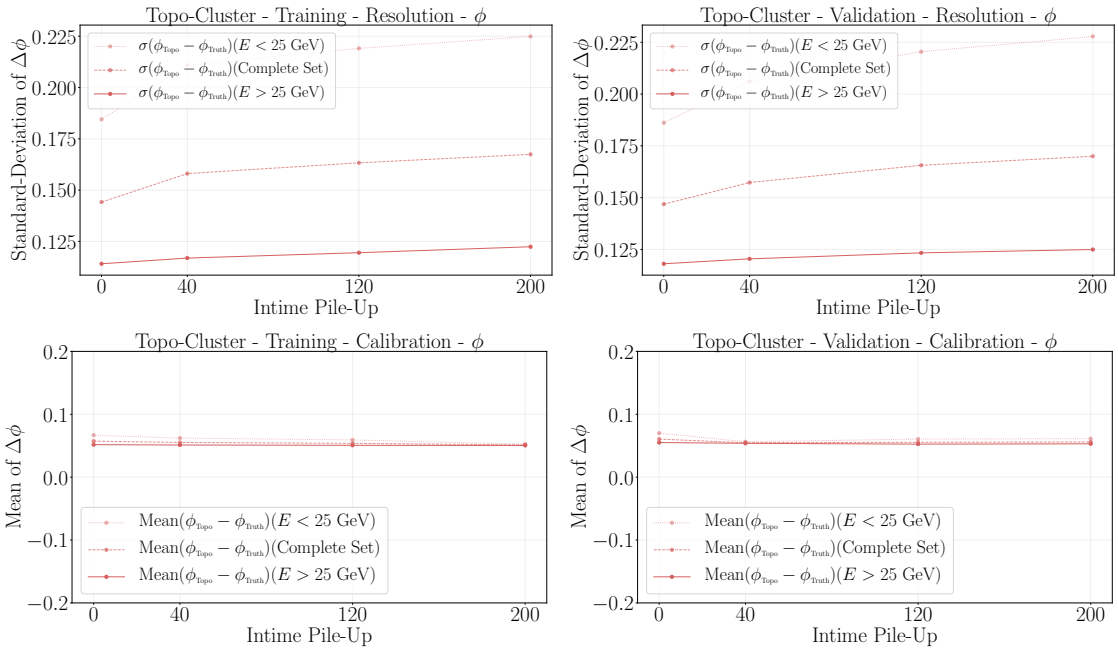


Figure 5.3: ΔE **Results (Topo-Clusters)**: This shows that the differences between the topo and truth-clusters. The resolutions worsens with pile-up as expected. The calibration graphs have values that are non-zero. These non-zeros originate from the topo-cluster calibration for *dead-material* whereas truth-clusters does not include dead material[a]. Because topo-cluster thresholds increase with pile-up, it is reasonable that the energy mean decreases with pile-up, although the topo-cluster calibrations complicate an exact explanation. The training set is the set the networks will be trained on whereas the validation set is blinded to the networks. These two sets should be approximately equivalent for the topo-cluster algorithm.



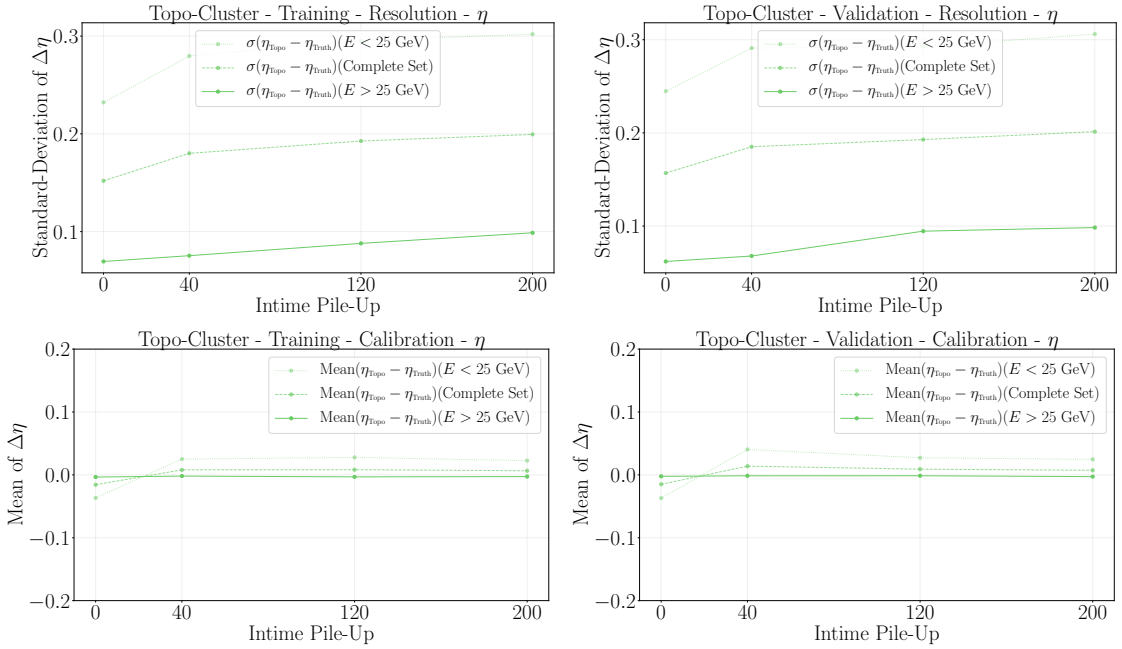
[a]Combining *dead-material* into the truth-clusters does rectify the E mean by shifting the curve down but it increases stand deviation in $\Delta\phi$,$\Delta\eta$, and ΔE. A modification in the definition of truth clusters, in section 4.1.2, *may* resolve this increase but it adds complexity.

Figure 5.4: $\Delta\phi$ **Results (Topo-Clusters)**: Electrons are bent in the $-\hat{\phi}$-direction because of the inner detector's magnetic field. All radiative photons, from the electrons, are accounted for in the truth cluster[a]; by contrast, the topo-clusters only accounts for radiative photons overlapping in the calorimeter. This gives a mechanism for the skew in the ϕ calibration. As expected resolution also deteriorates with pile-up.



[a]It *might* be reasonable to exclude distant radiation from the ϕ discussed in Section 4.1.2.

Figure 5.5: $\Delta\eta$ **Results (Topo-Clusters)**: These resolution graphs also shows deterioration with pile-up similar to graphs above. The calibration graphs below also highlights very stable η calibration with increasing pile-up.



5.3 Neural Network versus Truth-Cluster Comparison

Networks were trained **five** times with four networks at four different levels of pile-up leading to 80 networks. The loss of these networks is summarized in Figure 5.6. The five trials differ by two traits: initialization and batch randomization. For each setup, the networks starts with different initializations: all Glorot uniform (Gaussian-like)[5][47]. The trainings samples order is randomized at the end of each epoch this will also lead to a random spread between the trials. These multiple trials allows the consistency of training to be measured.

In the Figures 5.7–5.10 below, the calibration (mean) and resolution (STD) of $\Delta\eta$,$\Delta\phi$, and ΔE—for network clusters—are summarized. These plots are produced with the same methodology as Section 5.2. For neural networks we will use a similar order of values as topo-clusters:

$$\Delta\text{Values} = \text{Network-Cluster Values} - \text{Truth-Cluster Values}$$

We'll also look at 4 different training phase slices: epoch-10, epoch-20, epoch-45, and epoch-80. This will give us an idea of whether we are over-fitting[6] as well as give some idea to the training evolution of networks.

The Figures 5.7-5.10 have the error-bands and error-bars that correspond to the standard deviation of the five trials[7]. The mean of the five trials is shown by a small point. The lines and larger points are formed by the optimistic value (values closest to zero). This is akin to highlighting the network that reached the lowest loss.

Overall from the subsequent graphs we find that training examples converge towards the exact values while the validation set begins to pull away from the training set. Many cluster variables shown signs of the validation set worsening at 45-epochs and 80-epochs. Let's go over some observed trends seen by each network:

Network-A gives some of the most consistent results (smallest standard deviation between trials) as well as some of the most optimal results (nearest to zero). It

[5]Glorot uniform refers to: parameters that are Gaussian distributions centered at zero with their variance dependent on the number of inputs and output to their layer. Layers with more inputs and outputs will have smaller spread of initializations.

[6]Over-fitting is when the network fits the training set very well but does not optimally fit the test or validation set.

[7]A few trials failed due to memory/timeouts and were not included.

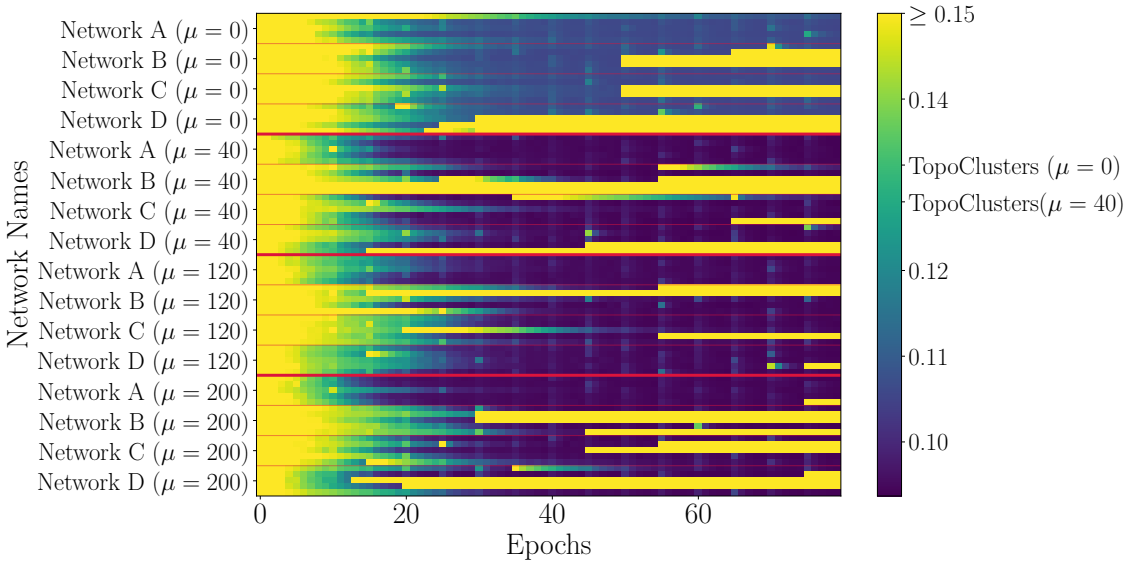
tends to exhibit the best results at late epochs. It also trains in the shortest time. The blue bands highlight the standard deviation of trials for this particular network.

Network-B had comparable performance to Net-A. It also performed very well in the validation set of η and epoch-20.

Network-C also had comparable results to Net-A and Net-B. Net-C consistently had the smallest resolution values for the training set at epoch-80. This implies it is good at 'recording' information. Which is surprising as network-B had a similar structure but approximately 4 million more parameters. This network also had some of the best energy resolutions at epoch-10.

Network-D gives very different results depending on the trial. Because of this it can be seen as unreliable.

Figure 5.6: **Multi-Network Loss**: This figure shows a combination of five trials, four networks, and at four pile-ups. The large lines separate the pile-up samples. The small lines separate the network structures. The five 'pixels' within each of these lines show the five separate trials. This gives rise to 80 networks. The loss converges[a] to 0.095. Most of yellow lines in this figure refer to values that grew beyond 0.15 and only a few actually failed to finish[b].



[a] This is from the "max distance" variable in Section 3.4.3 that adds a constant to the loss function. Networks can still improve as it approaches 0.095.

[b] Network D had a few memory/timeout failures.

Figure 5.7: E **Resolution Results**: To the left is the training set to the right is the validation set. The error bars represent different trials whose initialization and training sample ordering differed. The training set converges very cleanly to zero whereas the validation set shows signs of over-training by epoch 45. This figure contains the results for Network-A, Network-B, Network-C, and Network-D. The lines show the "best networks" (values closest to zero). The blue band highlights Network-A's standard deviation of the multiple trials.

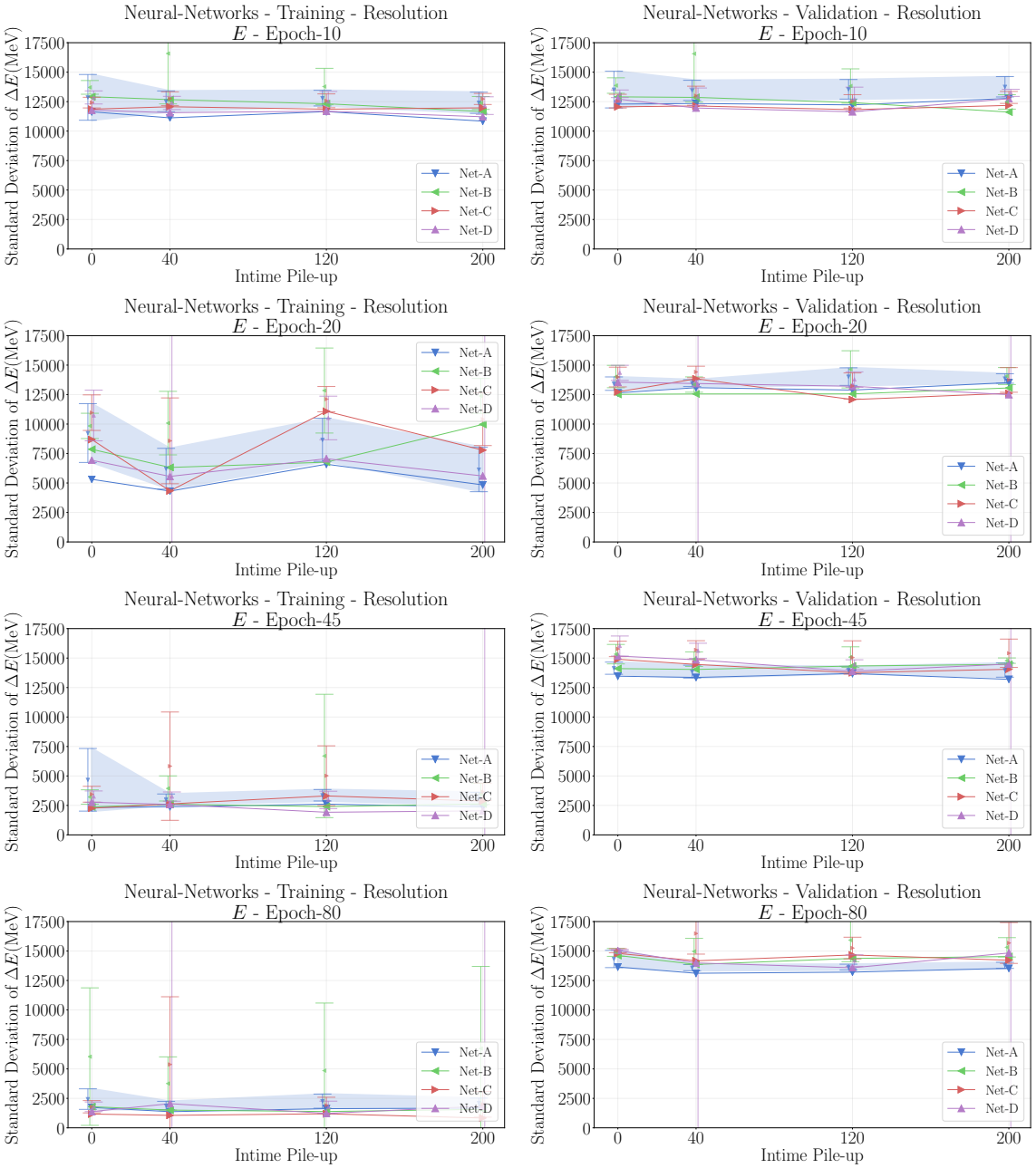


Figure 5.8: E **Calibration Results**: These graphs show signs of over-training by epoch 45. The error bars decreases with epochs suggesting that solutions are converging. This figure contains the results for Network-A, Network-B, Network-C, and Network-D. The lines show the "best networks" (values closest to zero). The blue band highlights Network-A's standard deviation of the multiple trials.

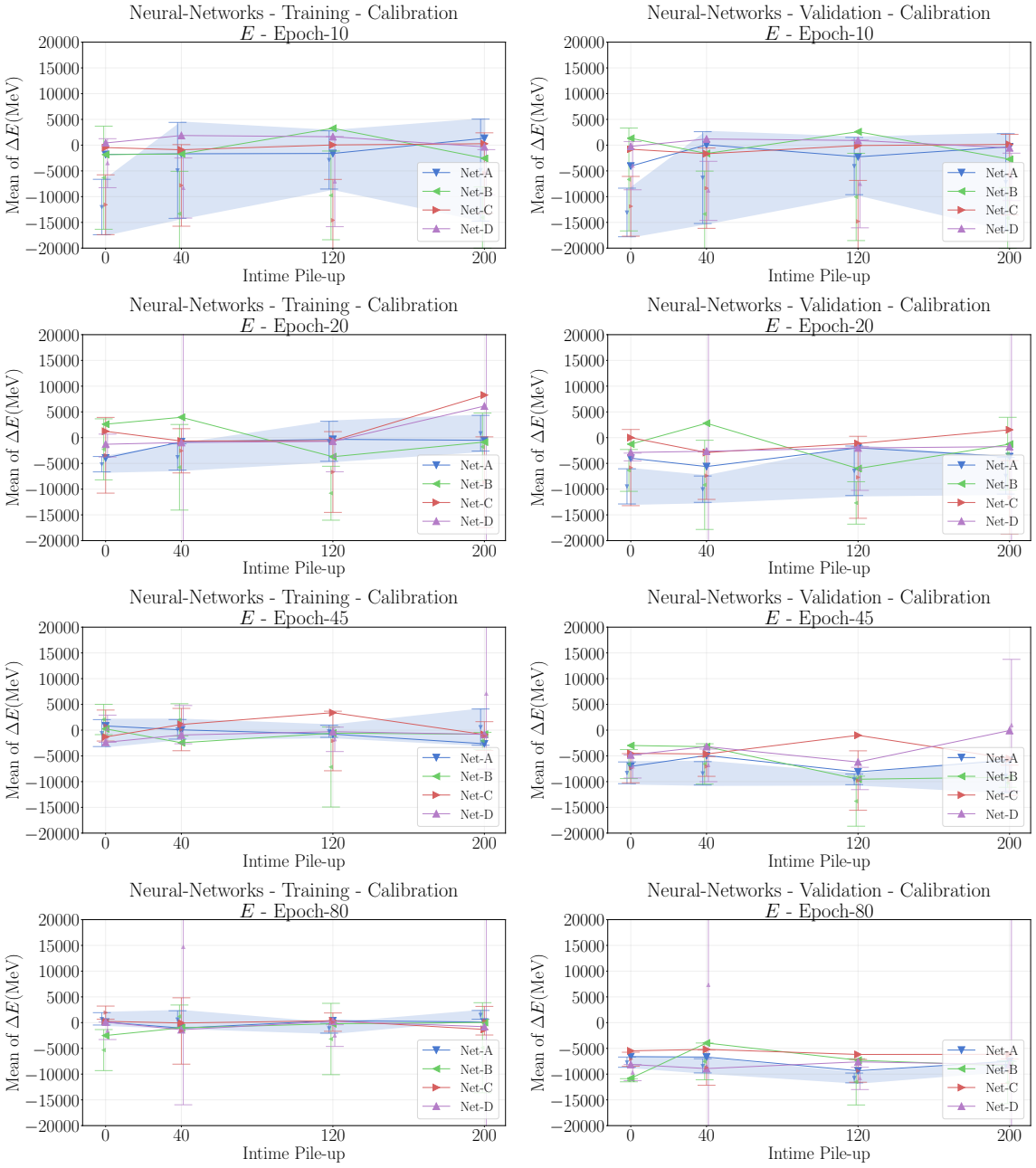


Figure 5.9: ϕ **Resolution Results**: These results are very stable both over epochs and trials. This figure contains the results for Network-A, Network-B, Network-C, and Network-D. The lines show the "best networks" (values closest to zero). The blue band highlights Network-A's standard deviation of the multiple trials.

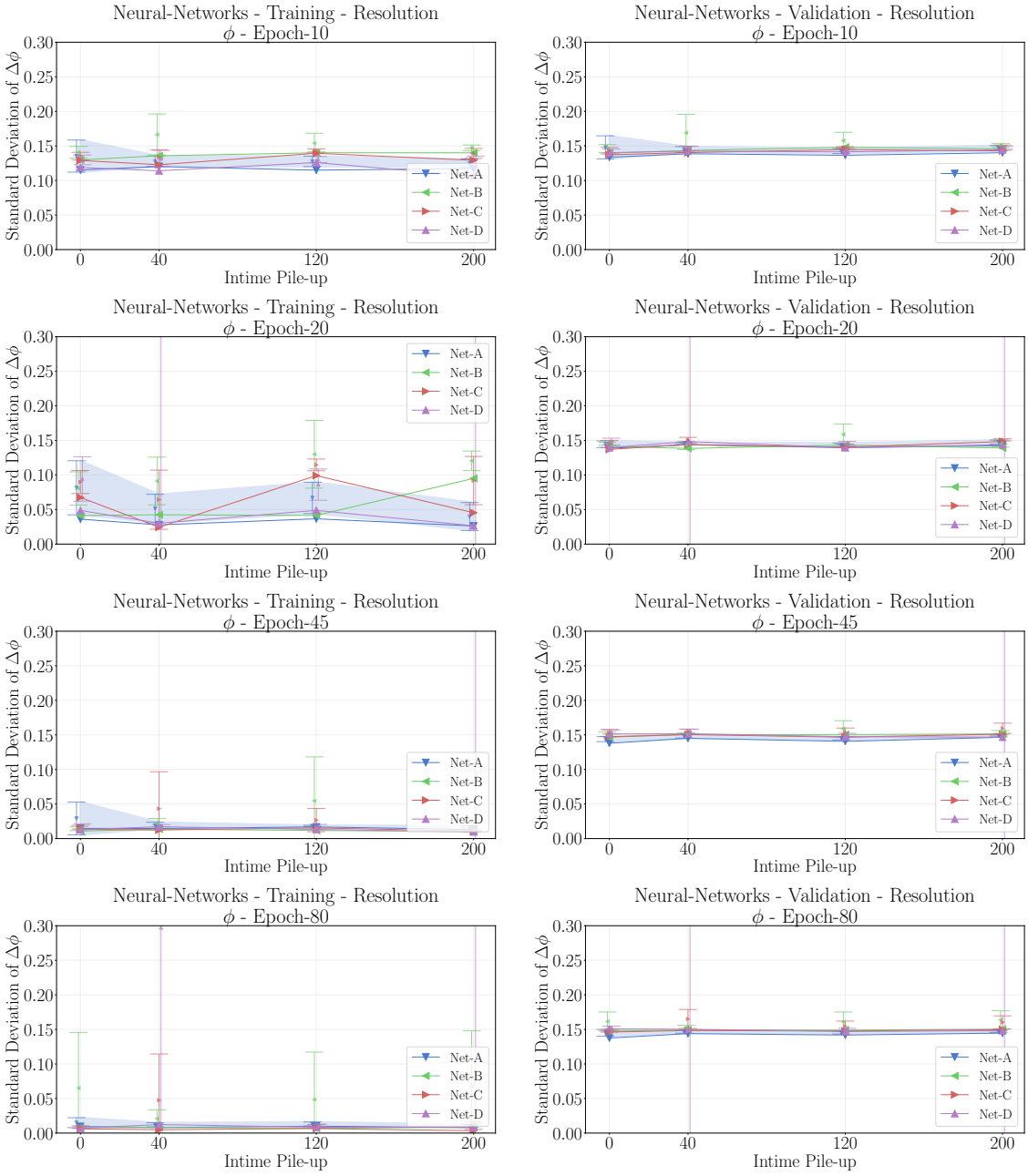


Figure 5.10: ϕ **Calibration Results**: These results are very stable both over epochs and trials. This figure contains the results for Network-A, Network-B, Network-C, and Network-D. The lines show the "best networks" (values closest to zero). The blue band highlights Network-A's standard deviation of the multiple trials.

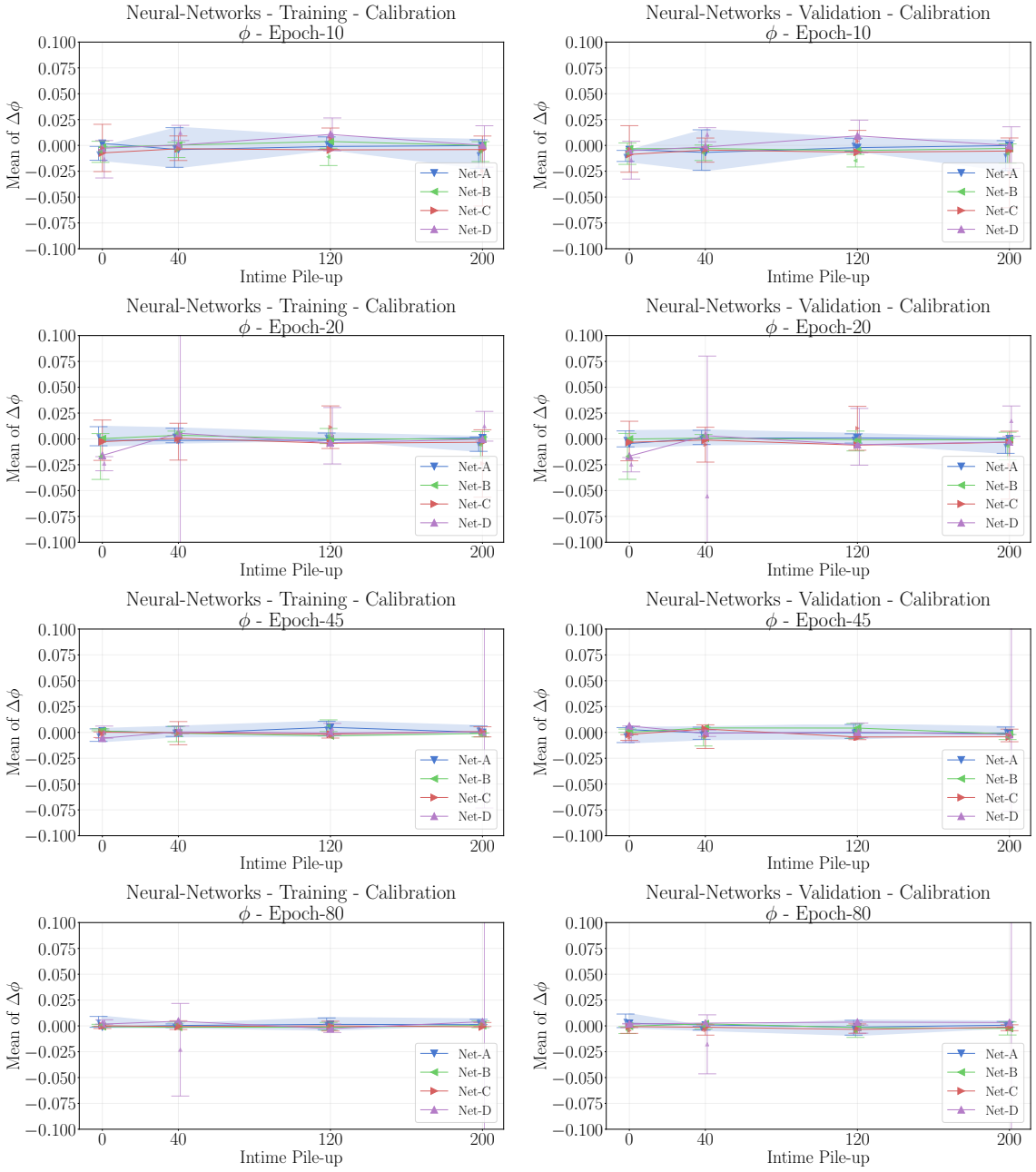


Figure 5.11: η **Resolution Results**: These graphs show similar results to ϕ resolutions. These graphs and Figure 5.12 suggest that the cost function in Section 3.4.3 should have had a lower σ_η values. This is discussed more in Section 5.4. This figure contains the results for Network-A, Network-B, Network-C, and Network-D. The lines show the "best networks" (values closest to zero). The blue band highlights Network-A's standard deviation of the multiple trials.

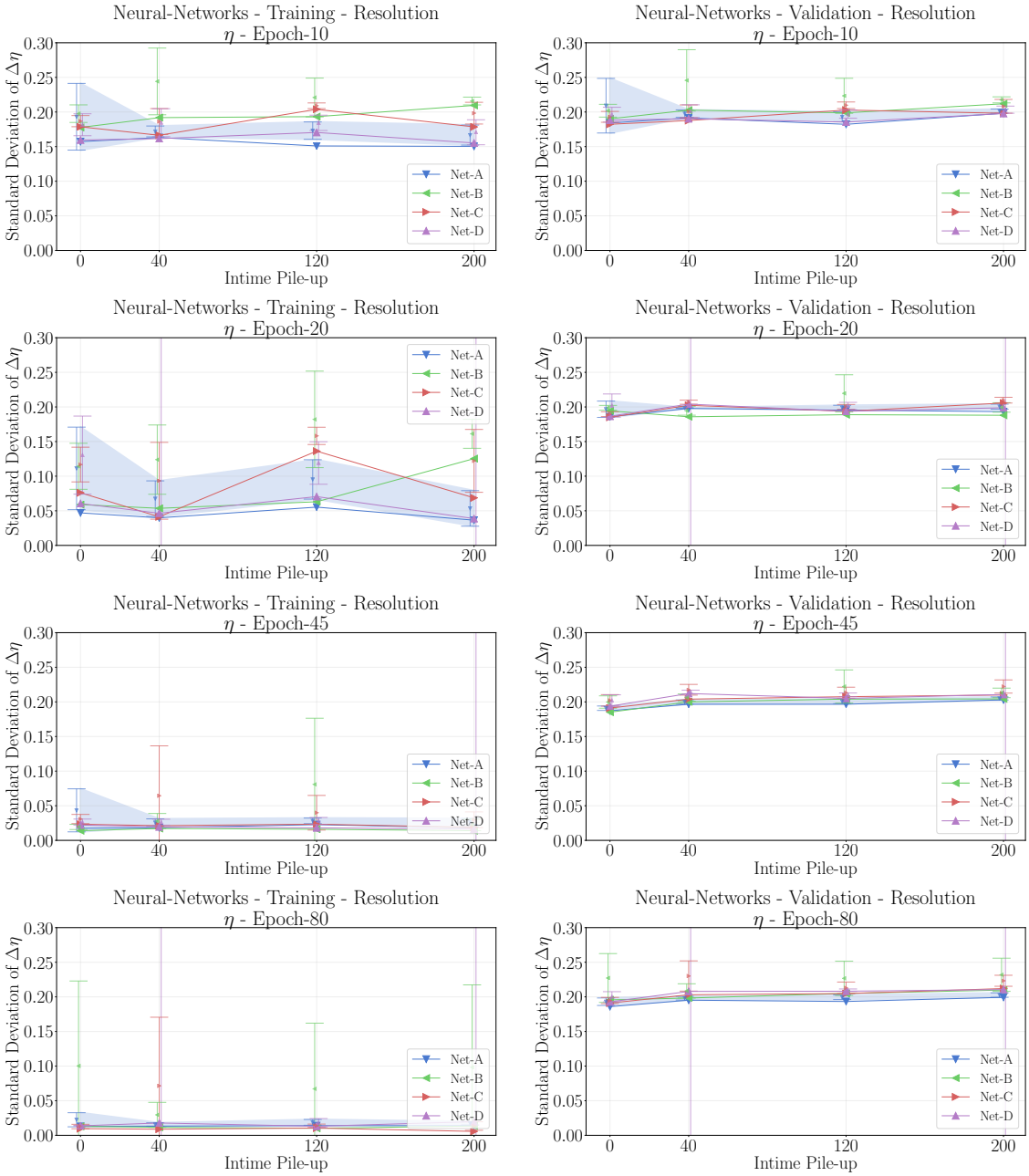
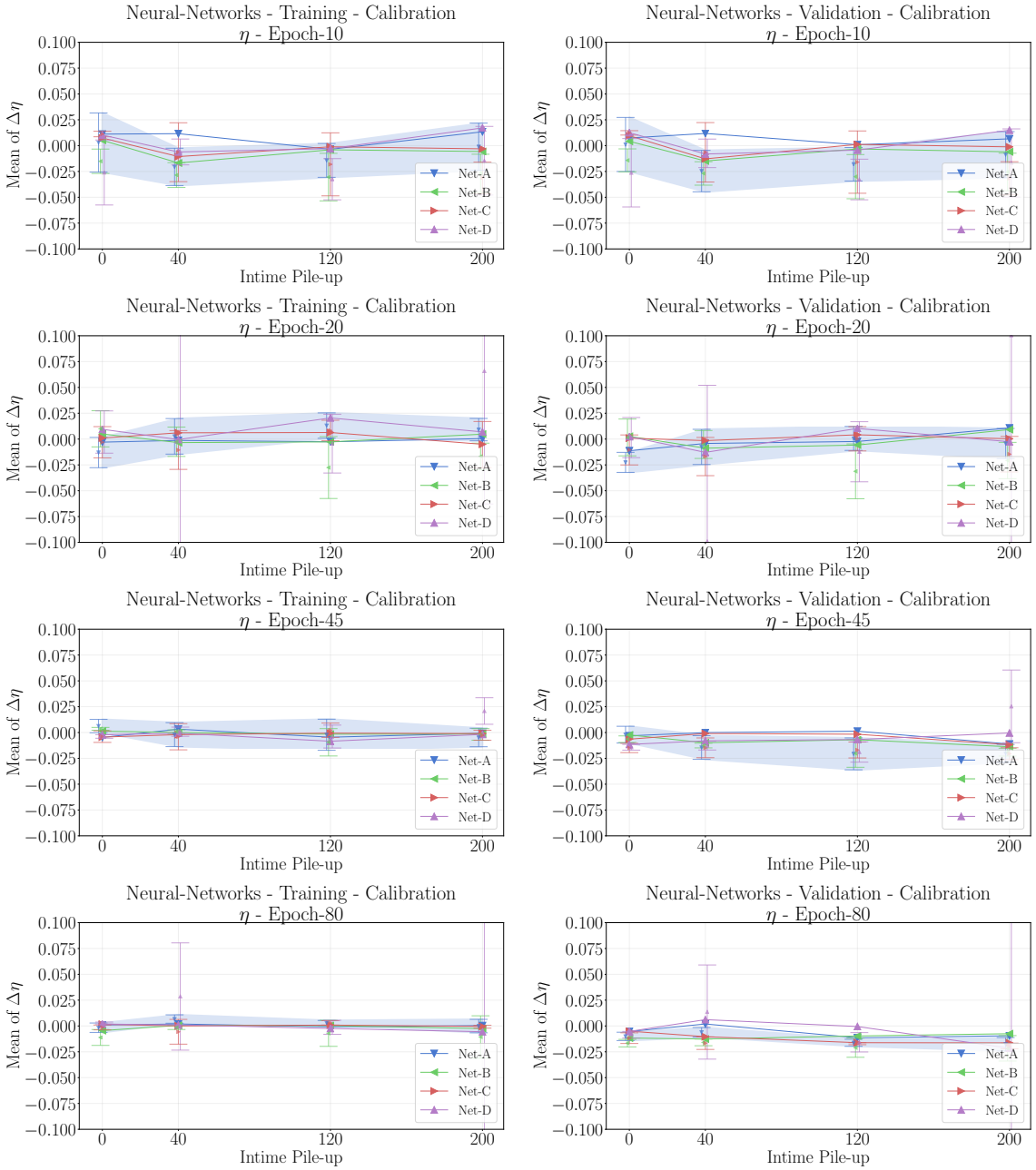


Figure 5.12: η **Calibration Results**: These graphs show a larger spread than ϕ. The validation set appears to be off center (below zero). This figure contains the results for Network-A, Network-B, Network-C, and Network-D. The lines show the "best networks" (values closest to zero). The blue band highlights Network-A's standard deviation of the multiple trials.



5.4 Final Comparison: Neural Network versus Topo-Cluster Comparison

Out of the four networks trained, the Network-A appears to be the most consistently trained. The best results of Network-A are from epoch 10 so this is what will be used for the final comparison.

Table 5.2 summarizes the key results from Figures 5.13–5.15. Generally the networks do better at calibrations whereas the topo-cluster algorithm does better at resolutions. Elaborating on the resolution results, in the validation set, the topo-cluster algorithm almost always outperforms the network. The exception is energy resolution at high pile-up where the network is comparable to topo-clusters[8]. The η results are surprisingly poor for the networks although some of this might be explainable by the equal values of σ_ϕ and σ_η in Section 3.4.3. For electrons in particular bending from the magnetic field along the -$\hat{\phi}$-direction along with EM showering particles leads to a larger σ_ϕ[9]. The non-zero calibrations for the topo-clusters are explained in the figures and offer a potential area for improvements for the training examples.

The networks show no significant sensitivity to the change in pile-up[10]; however, there *may* be a more dominant error—that is pile-up invariant—overshadowing noise from pile-up. This invariance to pile-up does offer a *possible* advantage for the networks. If the resolution results of the networks were comparable at low pile-up, this claim of an advantage would be strengthened.

A final note, the networks reached a *lower* test loss than the topo-cluster algorithm (from Fig. 4.3) which implies that it is doing a better job at minimizing the objective function from Section 3.4.3. This highlights one of the difficulties of machine learning, creating a differentiable objective function that converges towards the "ideal" solution. While our algorithm offers improvements with regards to particular variables, overall the topo-cluster algorithm offers better resolution results.

[8] While the topo-cluster algorithm has not been optimized for this range of pile-up, its thresholds do compensate for pile-up.

[9] This can be seen in the comparison Table 5.1.

[10] It's worth remembering that we are training networks at different pile-ups.

Table 5.2: **Summary of Final Results**: This summarizes which algorithm performed better for particular variables. The ratios (σ or mean) used the trial with the nearest value (σ or mean) to zero, Network-A, epoch-10, and intime pile-up of 120. The σ-ratio is calculated using σ_{Network} divided by $\sigma_{\text{Topo-Cluster}}$. The |mean-ratio| is calculated by taking the absolute value of $\text{mean}_{\text{Network}}$ divided by $\text{mean}_{\text{Topo-Cluster}}$.

Variable	Training Set	σ-ratio	Validation Set	σ-ratio
ΔE Resolution	Depends[a]	0.94	Depends[a]	0.98
$\Delta\phi$ Resolution	Tie	0.97	Topo-Cluster	1.15
$\Delta\eta$ Resolution	Topo-Cluster	1.73	Topo-Cluster	2.09
Variable	**Training Set**	**|mean-ratio|**	**Validation Set**	**|mean-ratio|**
ΔE Calibration	Network	0.94	Topo-Cluster	1.13
$\Delta\phi$ Calibration	Network	0.001/0.05	Network	0.002/0.05
$\Delta\eta$ Calibration	Tie	$\approx 0/0$	Topo-Cluster	$\approx 0.002/0$

[a]Topo-clusters are better at low pile-up. The networks are better at high pile-up.

Figure 5.13: **Comparison of Neural Networks and Topo-Clusters Energy**: On top we show the energy resolution (STD) graph and to the bottom is the energy calibration (mean). While some sections of the neural networks are closer to zero than the topo-cluster algorithm, these are in areas where the topo-cluster algorithm has not been completely optimized.

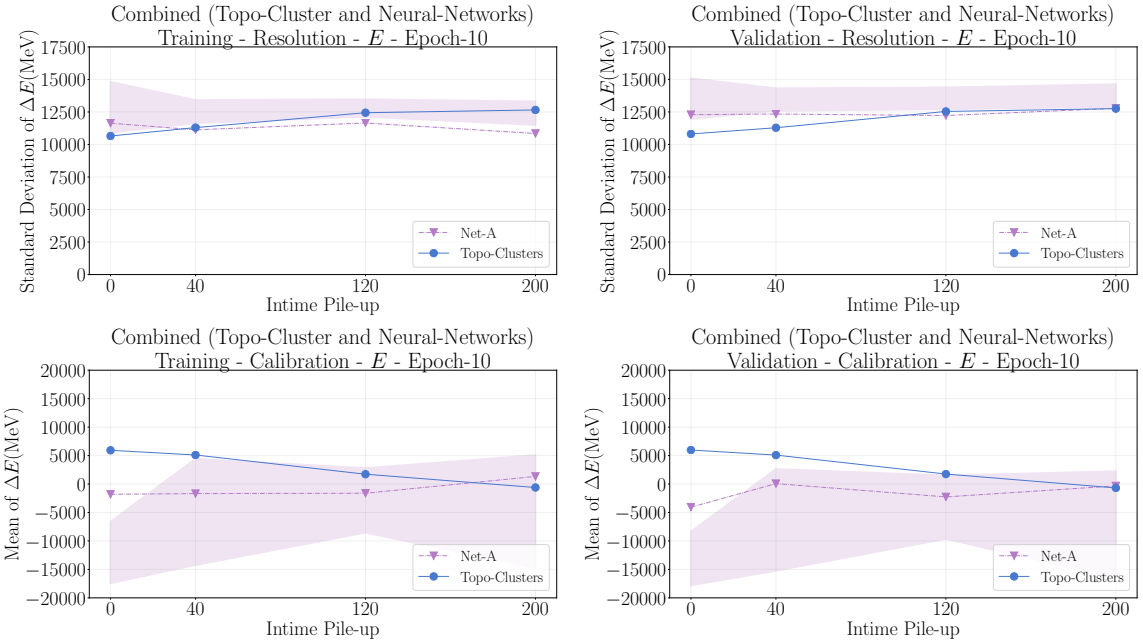


Figure 5.14: **Comparison of Neural Networks and Topo-Clusters** ϕ: On top we show the ϕ resolution(STD) graph and to the bottom is the ϕ calibration (mean). The ϕ discrepancy is discussed in Figure 5.4.

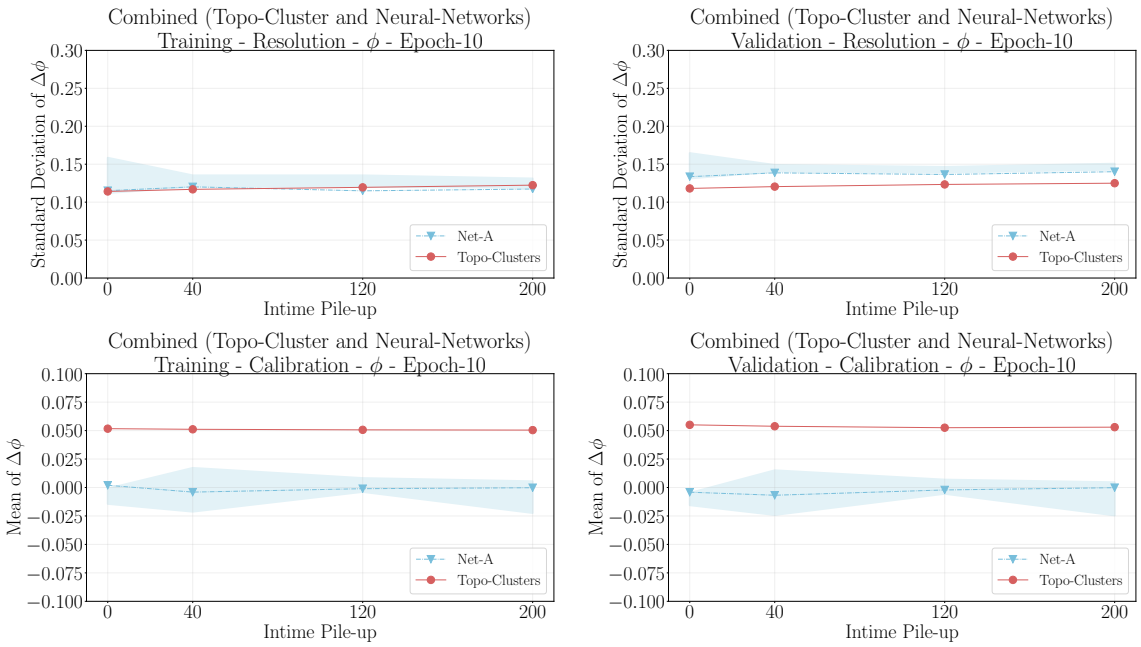
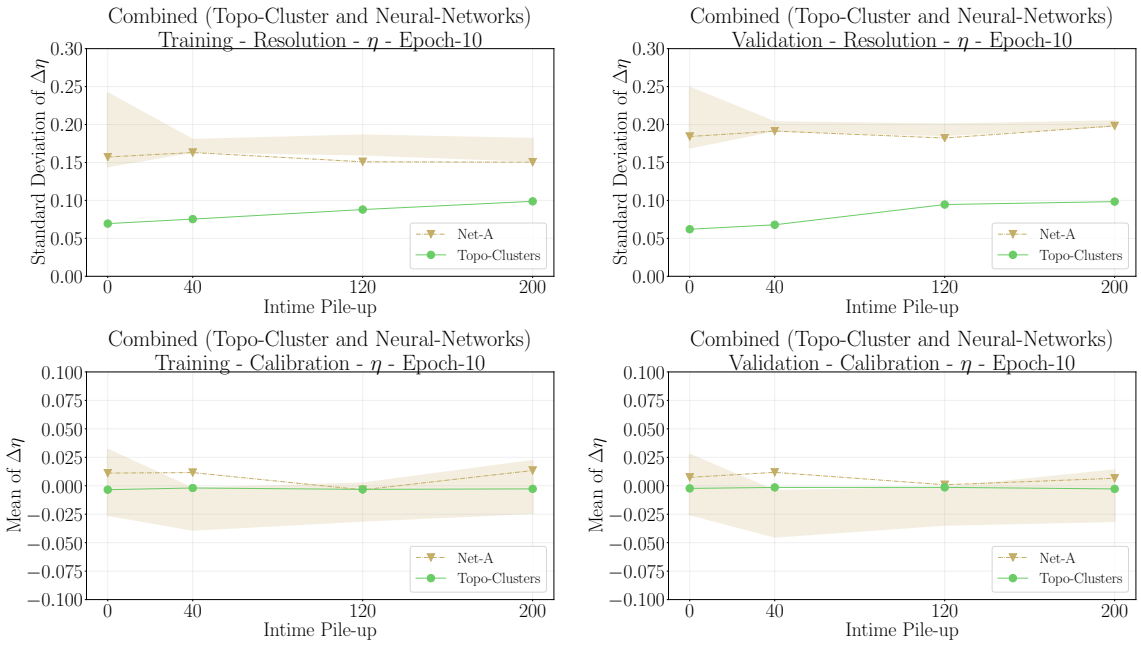


Figure 5.15: **Comparison of Neural Networks and Topo-Clusters** η: On top we show the η resolution (STD) graph and to the bottom is the η calibration (mean). η graphs show the networks need to prioritize this could be reflected by altering the σ_η value in the objective function in Section 3.4.3.



5.5 Follow-Up Analysis

All analysis after this point was done after the validation was revealed meaning that any further changes can bias the validation set. However, we'll treat this more as a warning for the next set of results rather than a blockade.

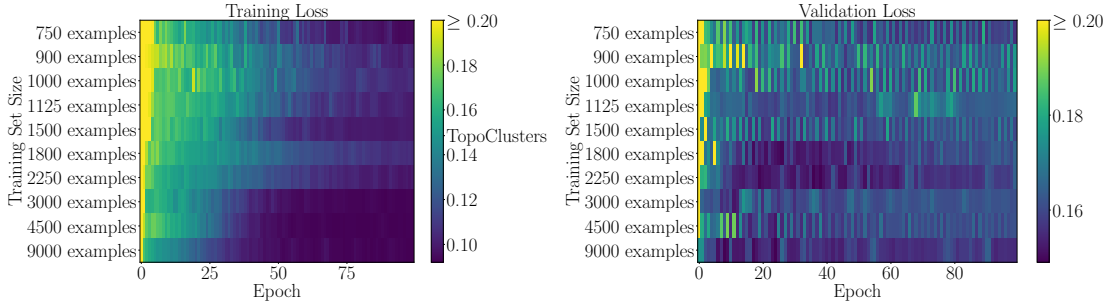
Data-sets: same update-interactions different training sizes

An often cited solution to over-training is increasing the size of the data set. Often it is expected that as the training set increases it should converge towards the validation set. In a hand-wavy fashion we can use this hypothesis to estimate the amount of training samples needed to reach this region by using an extrapolation. Keeping a fixed number of update iterations,

$$\text{Number Of Update Iterations} = \text{Epoch} \times \text{Training Size}$$

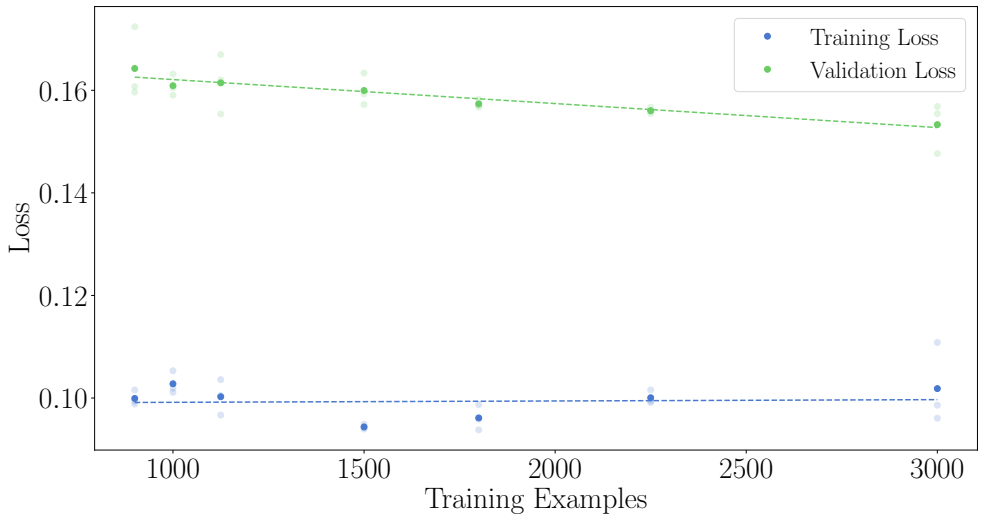
we can change the training size, and compare epochs where the number of update iterations are equal. It would be helpful to look at the results after the decrease in loss has saturated because this is the point where no more learning is happening. By

Figure 5.16: **Loss scans varying Training set Size**: Demonstrates how loss functions react to change in training examples. Equal numbers of updates should be compared. For instance, after 9 epochs "1000 Examples" has updated the same number of times as 3 epoch of "3000 Examples". The validation set is kept as a constant complete size (1000 events).



following this process for both the training set and validation (Fig 5.16) we get two curves. The method will allow to evaluate how the validation loss and training loss vary with the training set size. We'll do this three times to ensure similar training. This is shown in Figure 5.17. The two linear fits are loss scan of validation sets:

Figure 5.17: **Estimation of Training Examples Needed**: The lines all have equal updates meaning that they were updated on the same number of examples. Networks with smaller training sets reach higher validation loss with the same number of weight updates. We can use this to estimate the number of samples needed to reach training-validation equivalent. This is done with a simple linear fit.



$$y = -4.679 \times 10^{-6} x + 0.16677$$

and loss of training set:

$$y = 2.6234 \times 10^{-7} x + 0.0989$$

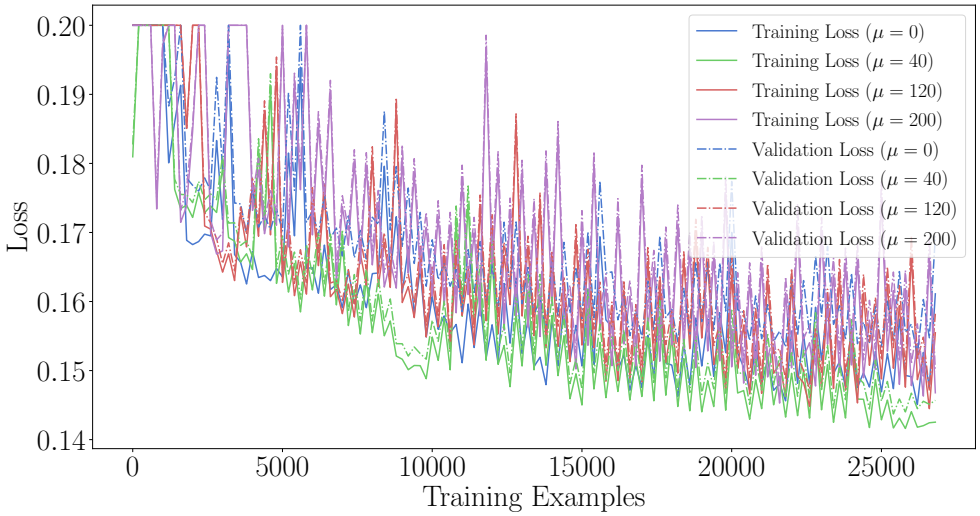
These two curves overlap at 13734 events. It is worth noticing that this is a very distant extrapolation and is very sensitive to the fitted slopes; the crossing point may be much further. Additionally the relationship is likely not linear over this range. In fact if we add a point at 9000 training examples and 0.149 validation loss this increases the cross over to 33931 training examples. This suggests that these slopes are not linear. A suggested training samples size for futures studies would be between 50000 and 100000 training samples.

Data-sets: Loss and Validation by batches

This was a follow-up to the test loss in the hyper-parameter scan appearing saturated (shown in Fig. 4.3). It could be seen as a concern that the test set did not appear

to decrease after the first epoch. This section demonstrates that the loss of both samples decreases during the first couple epochs. However; this does appear to imply retraining on the same information does not improve the training significantly. This might be due to using a fixed set of underlying pile-up events. This may give each example too strong of an identifier leading to over-training. A *possible* solution is separating the pile-up from the signal samples and randomizing the pairing of pile-up and signal events during training.

Figure 5.18: **Loss Scans by batch size**: A scan over the first three epochs[a] using batches of 200. Notice that there is significant learning in both training and validation sets in the first two epochs but this is decreases over the third training epoch. Loss here is calculated on the entire training set and the entire validation set.



[a] epoch 1 is 0-9000, epoch 2 is 9000-18000, and epoch 3 is 18000-27000.

Chapter 6

Conclusions

This thesis showed methods for using neural networks within a particle detector. The claim made in Section 1.3 has been accomplished. Let's digest some of the lessons:

6.1 Evaluations of Challenge and Solutions

At the beginning we outlined some of the challenges of this project. Now we have solved all of these challenges (albeit some solutions could be improved) so let's return to the list in Section 1.2:

Table 6.1: **Solutions of Challenges**: This gives a summary of the challenges and their solutions. Each topic will be expanded below.

Challenges	Solutions
Geometries	Treating the Detector of a set of images [Section 4.1.1]
Unordered Lists	Hungarian Algorithm and Ghost Clusters [Section 3.4]
Energy Depositions[a]	Truth Clusters [Section 4.1.2]
Hyper-Parameters	Using scans of different hyper parameters [Section 4.2]
Regression	Preprocessing and Matching [Section 4.1.3,Section 3.4.3]

[a]Complexities of energy deposition in material.

Geometries This seemed to work well. There are other methods using RNNs but so far there does not seem to be a more obvious approach for this problem.

Unordered Lists The Hungarian algorithm (that was used) *possibly* isn't the most optimal solution in this particular context. It remains *unclear* whether other

choices would be better. Other choices include using a moving window or a network version of the particle flow algorithm used by the Compact Muon Solenoid experiment[50].

Energy Depositions There are some reasons that one may want to be define truth clusters differently. For example, one might exclude large radiation from the electron or ignore energy deposits far from the core of the shower. This is especially true for location variables such as η and ϕ. This also is something that could affect the training significantly.

Hyper-Parameters Looking at the neural network results, different hyper parameters converged to relatively[1] similar places. After completing this project I think it would have been wiser to focus on the traits—such as the size and truth-cluster definition—in the training set earlier and look at network structure later on. 100 layer networks appear to require significantly more fine-tuning/domain knowledge.

Regression Regression is still a challenging problem. Small changes in preprocessing data can lead to much larger impacts on final results than expected such as an energy transformation or different scaling. Small preprocessing changes can lead to a divergence or significant improvements. Because machine learning in particle physics in a new field, this is an area where there is room for significant advancements.

6.2 Next Steps

There are many potential directions that could be pursued to try to improve on these results. I list a few examples:

1. **Including Pions**: A large portion of energy in the Calorimeter is deposited through pions. Because of this all general clustering algorithms should be able to cluster pion depositions. Pions also deposit in the EMEC and HEC which would be more sensitive to the deeper layers of the Calorimeter.

2. **Separating In-time Pile-up**: Each event was fixed throughout training. By re-mixing pileup and signal events after each epoch, the 'effective' number of

[1] changes in the data sets leads to much larger differences.

training samples can be increased. This separation seems necessary because of the saturation in validation loss.

3. **Separating Particle Events**: Similar to the suggest above the truth clusters could be reshuffled to increase the 'effective' number of events. This is also easier to implement as the ATLAS EGamma group does maintain large set of single electrons and single pions samples with *CalHits* (although this still requires a significant amount of development).

4. **Revisiting Hungarian Alg**: It is possible that there is a better method such as: using a moving window or the particle flow algorithm used by the Compact Muon Solenoid experiment[50]. Because different clusters share common network weights there *could* be correlated cluster-quantities when using the Hungarian algorithm. The Hungarian algorithm *seems* to be weighted towards the average of the truth-clusters (although this needs more study).

5. **Regularization**: regularization is a concept in machine learning that was not used. It *may* have helped larger networks perform better. This is a blind spot to the hyper parameter as regularization is common in many modern architectures. This concept *should* be included into future hyper-parameter scans.

6. **Dreams**: dreams act as a method for 'inverting' networks, and to understand what the network does they could be helpful. Dreams are further discussed in the Appendix B.2.1. Also with the appropriate constraints this may allow simulated clusters to be made with the clustering algorithm. This would require a significant amount of development.

6.3 Two Big Lessons

Both of the lessons are very elementary but sometimes get forgotten in very complex implementations:

Experimentation usually triumphs Theorizing: with regard to machine learning and neural networks it is necessary to **test** ideas. Networks often optimize and operate in non-intuitive and highly non-linear ways. Because of this aspect, ideas and optimizations don't always carry simply from context-to-context. For this reason a end-to-end algorithm should be formed as early as possible. This

gives a more appropriate context for experimentation. Others have put forward this statement[33].

Datasets: Problems often arise due to the data set. During the development of these algorithms preprocessing data often affected the algorithms more than network structure. If the dataset does not contain enough information to solve the problem or the function between target and input is not contained within the data set machine learning cannot help. This is very elementary but often new comers (including me initially) are more interested in network structures. However the largest gains can usually be found in the data. Network differences[2] give very similar results whereas pre-processing and filters on training data gave significant differences. Network structure should be seen more as a fine-tuning step rather than primary step.

6.4 Concluding

Returning to Figure 1.1 we should be able to identify every aspect. We have accomplished our goals, although more work is needed to attempt to improve upon previous solutions. The prospects of artificial neural nets at ATLAS look bright although there is a great deal of growth needed.

[2] see Figure 4.4 and C.8

Appendix A

Additional Information

A.1 Notes for future students

Thesis Used

Some of the quality experimental physics dissertations/theses used by this thesis are: [51, 52, 53, 54, 55].

Useful Git Repositories

External:

- repository of resources: https://github.com/iml-wg/HEP-ML-Resources
- light weight neural nets: https://github.com/lwtnn/lwtnn
- B-tagging with dense neural nets: https://github.com/Marie89/BTagging_DL1

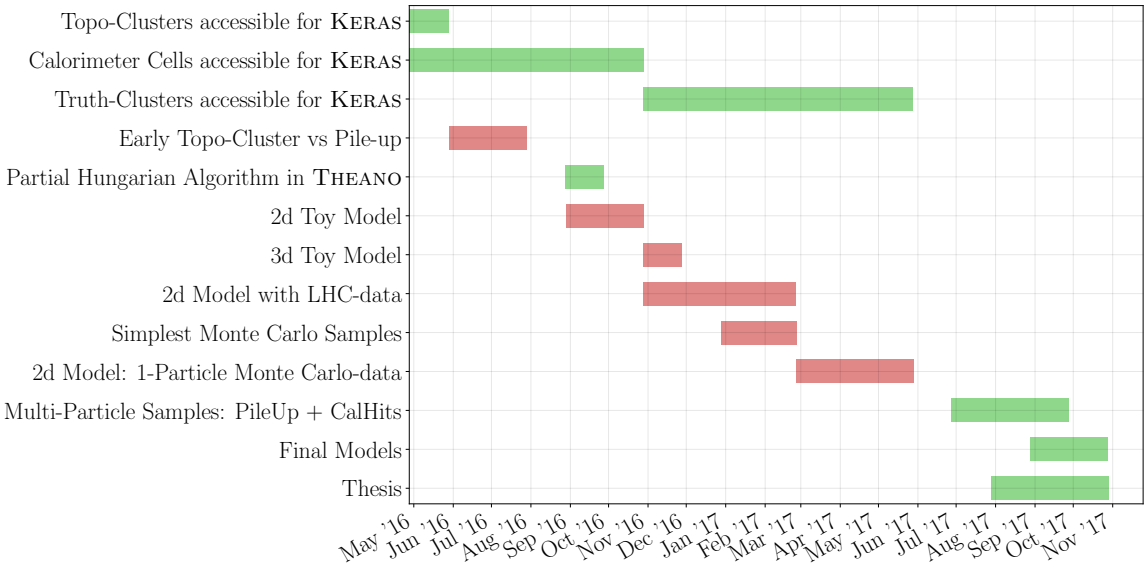
ATLAS Internal (ATLAS member password protected):

- Code used for thesis: https://gitlab.cern.ch/gniederm/Retia
- Toy models used in development of thesis: https://gitlab.cern.ch/gniederm/cellCNNs
- ATLAS main: https://gitlab.cern.ch/atlas/athena

Gannt Chart

The project's rough timeline is highlighted in Figure A.1. It doesn't include smaller projects.

Figure A.1: **Gannt Chart**: This highlights the time-lines of the project to give readers an idea of development lengths and how many intermediary subjects. All green entries are or red entries are items that were not directly discussed in this thesis. This was done from memory.



A.1.1 Rapidity and Pseudo-rapidity

To new students is might seem strange to use the coordinate η (pseudo-rapidity) rather than θ or z. This coordinate gives z directional lorentz boost invariant differences when relativistic velocities are involved. Consequentially this means the production of particles will be more uniformly distributed through η than θ. However both of these traits are difficult to see without doing some mathematics, so let's start by looking at a variable known as rapidity, y [1]. Rapidity is defined as

[1] it is normally used in concert with azimuthal angle ϕ which avoids the ambiguity with Cartesian coordinates

$$y \equiv \frac{1}{2}\ln\left(\frac{E+p_z c}{E-p_z c}\right)$$

Looking quickly at this we can see this is dependent on E and p_z. When the transverse momentum is significantly higher than the momentum along z ($E \gg p_z$) the value goes to zero and when the transverse momentum is significantly lower than the momentum along z ($E \ll p_z$) the value goes to infinity. This can be written in a few other ways:

$$y = \ln\left(\sqrt{\frac{E+p_z c}{E-p_z c}}\right) = \ln\left(\frac{E+p_z c}{\sqrt{E^2-p_z^2c^2}}\right) = \ln\left(\frac{E+p_z c}{M_T c^2}\right)$$

Let's apply a lorentz boost in the z direction to this last definition of rapidity. As a reminder $p'_z = \gamma p_z - \beta\gamma E/c$ and $E'/c = \gamma E/c - \beta\gamma p_z$.

$$y' = \frac{1}{2}\ln\left(\frac{E'+p'_z c}{E'-p'_z c}\right) = \frac{1}{2}\ln\left(\frac{\gamma E/c - \beta\gamma E/c + \gamma p_z - \beta\gamma p_z}{\gamma E/c + \beta\gamma E/c - \gamma p_z - \beta\gamma p_z}\right)$$

$$y' = \frac{1}{2}\ln\left(\frac{E/c(1-\beta)+p_z(1-\beta)}{E/c(1+\beta)-p_z(1+\beta)}\right) = \frac{1}{2}\ln\left(\left(\frac{E+p_z c}{E-p_z c}\right)\left(\frac{1-\beta}{1+\beta}\right)\right)$$

This gives a very simple transformation law:

$$y' = y + \frac{1}{2}\ln\left(\frac{1-\beta}{1+\beta}\right)$$

and this also shows that differences between rapidity are Lorentz invariant to boosts in the $\hat{z}$-direction:

$$y'_1 - y'_2 = y_1 + \frac{1}{2}\ln\left(\frac{1-\beta}{1+\beta}\right) - y_2 - \frac{1}{2}\ln\left(\frac{1-\beta}{1+\beta}\right) = y_1 - y_2$$

Now there is a very useful approximation we can make when momentum is much large than that the rest mass which can be written as $mc^2 \ll pc$ giving us $E \approx pc$ together this gives:

$$y = \frac{1}{2}\ln\left(\frac{E+p_z c}{E-p_z c}\right) \approx \frac{1}{2}\ln\left(\frac{1+p_z/p}{1-p_z/p}\right) = \frac{1}{2}\ln\left(\frac{1+\cos\theta}{1-\cos\theta}\right) = \frac{1}{2}\ln\left(\frac{\cos^2\theta/2}{\sin^2\theta/2}\right)$$

Notice that this last term independent of momentum and energy meaning it is very easy to calculate and it has the same traits as rapidity at large momentum scales.

This term is called pseudo-rapidity and is the common variable of choice for particle colliders like the LHC:

$$\eta = -\ln\left(\tan\frac{\theta}{2}\right)$$

It is worth noting that this is a monotomically increasing function in the range of θ. So we can form a basis in the position coordinates from η, ϕ, and r instead of θ, ϕ, and r.

A.1.2 Extra Notes on Lorentz/Cauchy Distributions

Because Gaussians and Lorentz distibutions look very similar on a linear scale it important to highlight how they are different. Predominately because the tails of the lorentz/cauchy distribution directly affect physics searchs. This can easily be seen on the log-scale where the tails of the Lorentz distribution is still quite significant $(1/x)$. This means that large resonances such as the Z boson resonance will overlay smaller resonances (higgs) causing a fairly difficult data processing. The data processing would be much simpler if these resonances where Gaussian shaped. Figure A.2 also does highlight that for smaller resonances, the peaks are comparable.

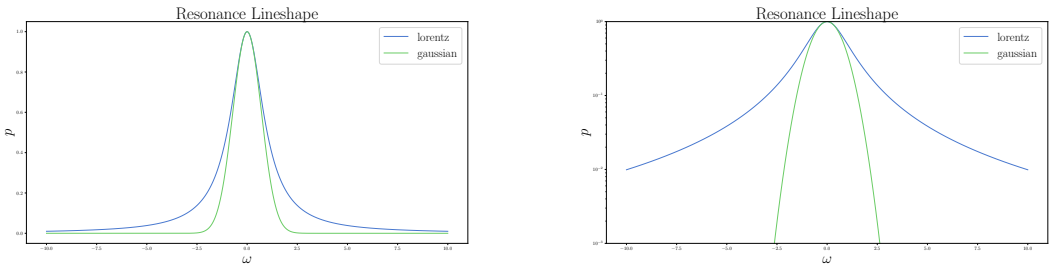


Figure A.2: **Lorentz vs Gaussian Distribution**: A comparison of Gaussian and Lorentz/Cauchy distribution at two different scales (not normalized).

Lorentz distrubitions are stable in the sense that multiple random variables following the distribution will converge to a Lorentz distribution. It also has some other strange traits such as mathematically undefined expectation value and variance. Often in experimental physics a convolution of Lorentz/Cauchy and Gaussain distribution happens this gives the Voigt distribution which isn't often seen due to it's complexity shared with the Lorentz/Cauchy and because often only the peak or tails are of interesting.

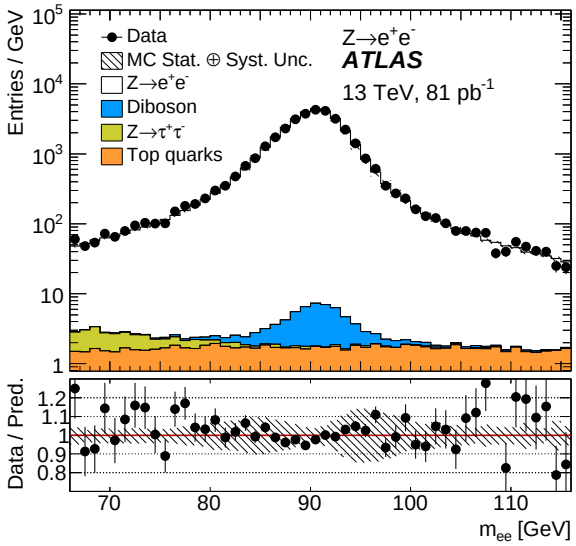


Figure A.3: **ATLAS Z boson resonance log-scale**: On the log scale the tails can be seen more clearly. Taken from [20]

Appendix B

Additional Neural Nets Information

In this appendix, I'd like to review some extra concepts from the field of artificial neural network. There is a large variety of different tools to use and I'd like to try to explain some of the design choices and give the reader an idea about potential. This section is largely based on a review article [56].

Quite a few of these topics were not implemented in the thesis. The topics that were used includes: *dropout*, *automatic differentiation* (done by Theano), and the *Nadam* optimizer.

B.1 Optimizers

B.1.1 Mini-Batched Stochastic Gradient Descent

Often in neural net literature mini-batched stochastic gradient descent is used interchangeably with gradient descent. mini-batched stochastic gradient descent is usually seen as being the standard candle of optimizers. This splits the full training set into a set of mini-batches which are then trained on. Because the training set is split up it will be quicker to solve and yet it should be large enough that it adequately represents the loss function($\mathscr{L}_{\text{mini-batch}} = \mathscr{L}_{\text{actual}}$). Splitting the total set of n into smaller sets of m. This can be written as:

$$\Delta w_{ij} = -\frac{1}{m_0} \sum_{i=0}^{m_0} \frac{\partial \mathscr{L}(x_i, y_i)}{\partial w_{ij}}$$

This can be evaluated using *back-propagation* described in Section 3.1.3.

B.1.2 Decay, Momentum, and Nestrov Momentum

There are a few common add-ons to the mini-batched stochastic gradient descent. These improve the rate of convergence and accuracy of the convergence area. The three most common are decay, momentum, and nestrov momentum.

Decay initially it makes sense for the learning rate to be high to find the approximate area; however, if it stays large the descent will likely only oscillate around of the exact solution. The usually method for accomplishing this is just multiplying the weights by the decay rate.

Momentum is used to both improve the speed of convergence and also to reduce the impact of saddle points. To a regular gradient descent multi-dimensional saddle points will resemble local minimum to most of the weights. For this reason, it is helpful to add a portion of the last update to the current update. This is analogous to building up 'momentum'. This *momentum* will build up in saddle points allowing the descent to escape these regions. When referring to standard *momentum* the momentum term is added after the gradient is calculated.

Nestrov Momentum is an method that is used to improve the convergence with momentum. It uses the accumulated gradient ("momentum") to correct the network before calculating the gradient loss. This method has been shown to converge quicker in many situations.

B.1.3 Parameter-based Adaptive Learning

There are a number of methods for improving the standard gradient descent many of where revolve around using a set of learning parameters. I'd like to quickily summarize some of the popular methods.

Adagrad uses the concept of updating each parameters learning rate independently based on a matrix G:

$$w_{t+1,i} = w_{t,i} - \frac{\alpha}{\sqrt{G_{t,ii} + \epsilon}} \nabla_{w_i} \mathscr{L}(w_{t,i})$$

If we write this in a similar method to the gradient descent it becomes:

$$\Delta w_i = - \sum_{j,i=0}^{m_o, m_w} \frac{\alpha}{\sqrt{G_{ii} + \epsilon}} \frac{\partial \mathscr{L}(x_j, y_j)}{\partial w_i}$$

ϵ avoids any divisions by zero and G_{ii} is a diagonal matrix that is summed of the square of the past gradients of w_i. The square and square root is required for performance reasons. using a element-wise multiplication we can find a solution. This is an improvement because it automatically updates the learning rate for each parameter. The issue with this algorithm comes from the fact that all of the previous gradients are added.

Adadelta and *RMSprops* both try to limit the number of previous gradients used in different fashions.

Nadam was the optimizer used in this thesis. Nadam stands for Adadelta and nestrov momentum. It incorporates all the topics above.

B.1.4 Which Optimizer to use

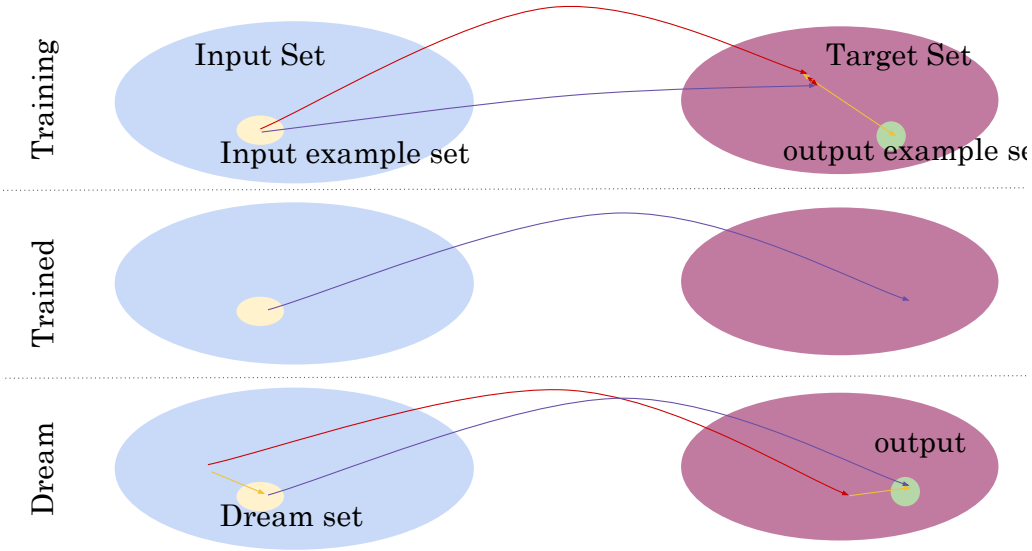
As for application generally for troubleshooting simpler optimizers should be used while Nadam has a higher potential to converge the quickest. If the objective function is non-standard it is best to be a little wary of the more complex optimizers. There are a great deal of articles describing there effects in different fields. A reference that can help with the choice of optimizer is Ref. [56].

B.2 Select Topics

B.2.1 Dreams

An interesting topic that could be useful in the future is a concept known as *dreams*[57, 58, 59]. These 'dreams' could help understand what the networks are learning. They work by fixing a network's weights then 'unfolding' or 'inverting' a network through a root-finding algorithms. This can be seen in Figure B.1.

Figure B.1: **Dream Schematic**: this diagram shows that during the dream stages the network (arrow) is fixed and the input is altered in order to try and reach the output that has been given.



This is a method for opening the "black box"[1]. An early **rough** example can be seen in Figure B.2.

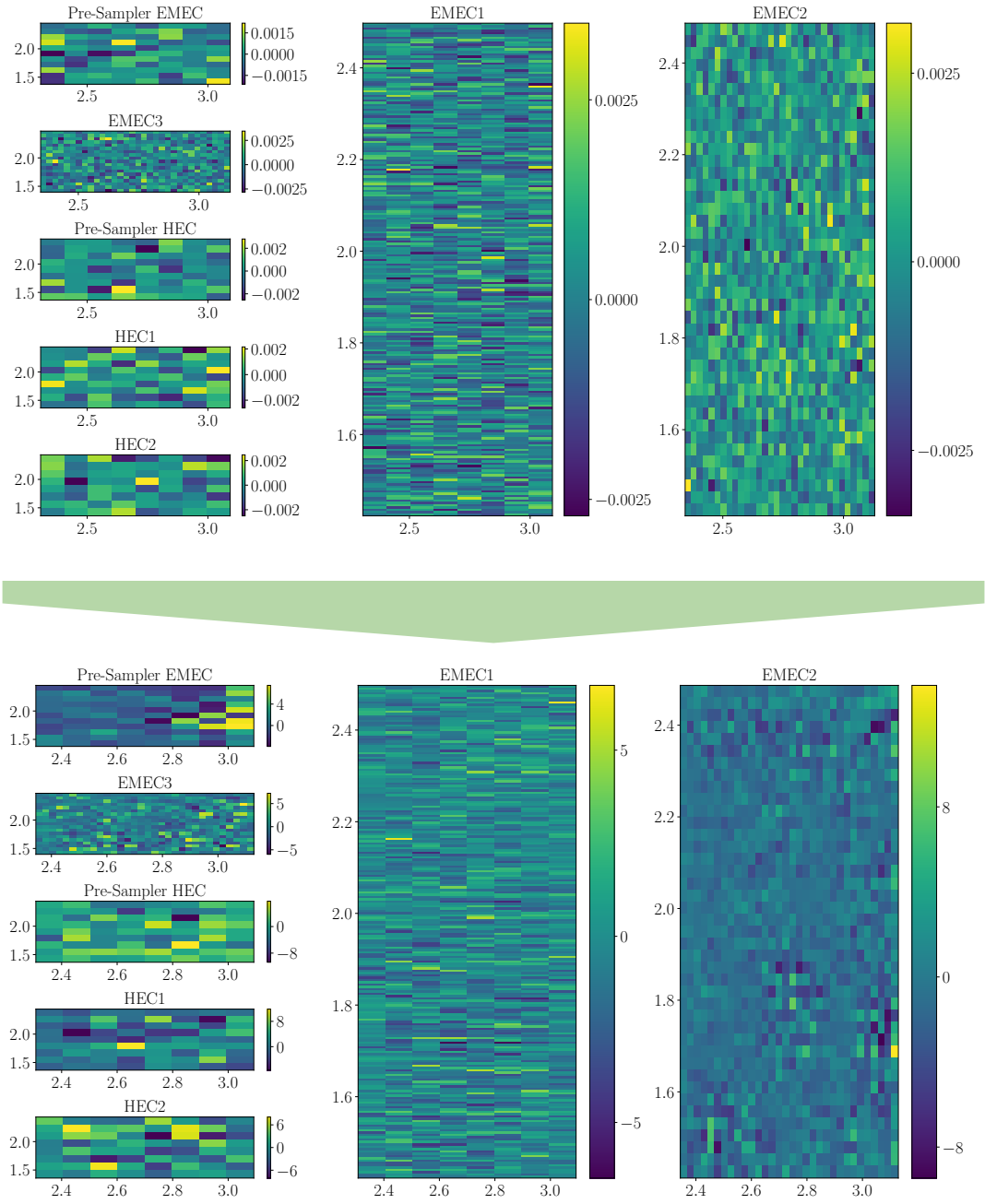
B.2.2 Recurrent Neural Networks

Recurrent neural networks are known for being able to expand to time-varying or dimensional-varying information[2]. These networks could have been used to solve the variable amount of clusters problem; however, their prospects didn't look great for

[1] Although in principle the box is always open and it would be more apt to call it a grey or "murky" box.

[2] A little bit of a confusion recurrent neural networks and recursive neural networks are both called RNNs. For the purpose of this work we will be using RNN for recurrent

Figure B.2: **Dream Example**: this is **rough** example of dreaming a network. The input (top image) was a Gaussian distributed with σ=0.001. After the dream process (lower image), clusters of "energy"[a] clearly appear in EMEC2 and the Pre-sampler. The lower image also shows signs of clusters in the HEC whereas it doesn't appear to show signals in EMEC1. The Network-A was used with $\mu = 120$.[b]



[a] the units are in network space.

[b] This was done with a modified KERAS deep-dream code from https://github.com/fchollet/keras/blob/master/examples/deep_dream.py.

this particular problem. Particularly within the realm of particle physics. Although this may change in the near future.

B.2.3 Dropout

Often neural networks can over fit due to the large numbers of parameters which often rival or eclipse the size of the training set. An interesting "solution" has been found for this problem. Random neurons are dropped from each layers so called dropout is formed co-adaptation. The dropped out neurons are switched after each batch. This can be seen as training a set of different models with different structure and then combining them together.

One final thing to mention has to do with the post-training stage. In order to get the proper weight all the layers should be multiplied by the dropout rate.

$$\text{weights}_{\text{final}} = p \times \text{weights}_{\text{trained}}$$

More information can be found at[60].

B.2.4 Dying ReLu Problem

Looking back to Section 3.1.3 where we discussed backprogapation the ReLu function behaves in an interesting way when its input is negative values. In this state the gradient of the ReLu function is zero meaning any update to the weight is canceled. If entire set of inputs doesn't turn this number positive, the neuron likely will stay negative and remain unchangeable, i.e. "die". While the network could change bringing this neuron back to "life" in practice the opposite tend to happen.

In particular situations this is useful because it introduces a permanent dropout that prevents over-fitting. However if too many neurons "die", there is a lack of trainable parameters and you end up with the *dying-ReLu problem*. This tend to be of larger concern in long training sessions and when many trainable parameters are need to properly represent the problem.

The solution to this problem is to introduce a activation function with non-zero gradients, e.g. the Leaky ReLu function.

B.2.5 Global and Local Minimums of the Loss Function

It is worth discussing for a few moments global and local minimum. Neural nets have highly dimensional loss functions which are often not convex. This fact leads to the question of how likely it is to find the global minimum rather than a local minimum? Before tackling this question it is worth remember some of the approximations we have implicitly made. First because we are only using a subset of the possible training examples we are using an approximate loss function:

$$\mathscr{L}_{\text{complete set}} \approx \mathscr{L}_{\text{training set}}$$

This means that the global solution is the solution to the training set and not necessarily the solution to the complete set. The argument that is commonly made is that the goal is not actually to reach the global minimum of training set. The goal is to reach a minimum that is similar to the global minimum of the complete set. A local minimum of the training set may actually be preferred, if it performs better on independent validation sets.

The paper that makes this argument is Ref. [60].

B.2.6 Automatic Differentiation

This is one of the tools that makes neural nets computationally feasible. It is important for completeness but is well outside of the scope or interests of this thesis. In order to do this efficiently a gradient graphical map can be computed. This is more completely explained in TensorFlow or Theano documentation.

B.2.7 Path-Summation Formulation

For theoretical consideration it is helpful to realize that ANNs can be represented by a summation of all paths through the network. This can be helpful for more fully understanding drop-out[60].

B.2.8 Non-Euclidean Geometry

Another topic to mention is the non-Euclidean geometry. Neural networks have also been written but these or non-computationally optimized algorithms[61].

B.2.9 Trainable Activation functions

These were not used for this work. But it is useful to be aware that there are activation functions that contain learn-able weights such as PreLu (parameteric ReLu).

Appendix C

Computational Resources

C.1 Software

Most of the base code is written largely in C++ and python. Any omission beyond those listed below is accidental.

Machine Learning Python Libraries

- theano [46]
- keras [45]
- CUDA (testing purposes)

General Python Libraries

- matplotlib [62]
- numpy [63]
- h5py [64]
- NetworkX [65]

ATLAS specific

- Athena [41]
- Root [66]

- pyAthena
- numpyRoot
- Rucio [67]
- GEANT4 [42]
- rootpy [68]

Latex

- Overleaf
- hyperlinks

Operating system/Misc

- Windows 8, Scientific Linux, and CentOS
- Git and SVN[1]
- GitHub and GitLab
- Javascript and ThreeJS

Software to be aware of for future use

- Caffe [69]

  A quick convolutional neural net platform.

- TensorFlow [70]

  Googles version of theano, in the future it is worth looking into in more depth.

- PyTorch

  An easy to use python machine learning library uses reverse-mode autodifferentation and helpful for Recursive neural networks.

[1] use Git

- LTNN

  Light neural networks. A C++ library for converting Keras models from python. This might be the easiest way of switching out of the python regime and very helpful for any implementation of neural nets in current physics analysis.

C.2 Hardware

This contains acknowledgment of computational resources uses from most to least used.

Cedar Cluster – One of Canada's new National Computing Sites located at Simone Fraser University

this is where most of the final neural network results were done. Only CPU ended up being used as the CPU-time was very available and the CPU trained the networks quicker than anticipated. The work-horse of this thesis

Symmetry Cluster – Small Cluster located at the University of Victoria

Majority of toy models and experimenting were done with this cluster as well as most of the Monte Carlo samples were produced here.

Parallel Cluster – Cluster that contains GPUs located at University of Calgary

Some GPU tests were performed on Parallel

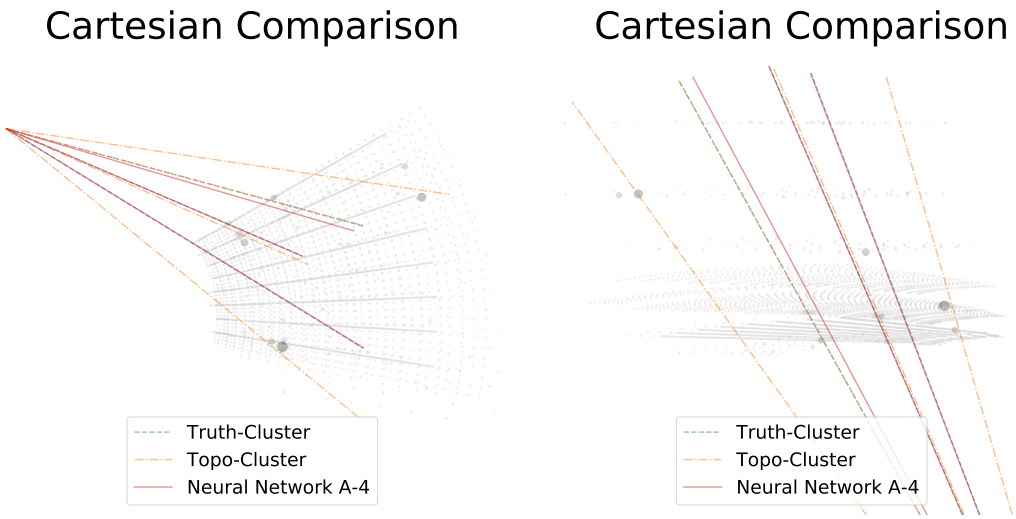
lxplus Cluster – Cluster Located in Geneva, Switzerland

Used for learning purposes related to ATLAS-specific software. Useful for prototyping.

C.3 Detailed Data

These are vector images and zoom-able. These are the average energy across events. It includes both the signal electrons as well as the electronic noise and pileup. It can be thought of as a heat map for areas of sampling.

Figure C.1: **Single Events In Cartesian Space - Training Sample**: Epoch 80 and $\mu = 120$. Size of cells is proportional to energy lines represent a projection from the center of the collider. Notice that training samples begin converge while there are ignored energy deposits. This extra categorization between pile-up and event may make the neural network worse.



Validation Sample - Small Number of Clusters

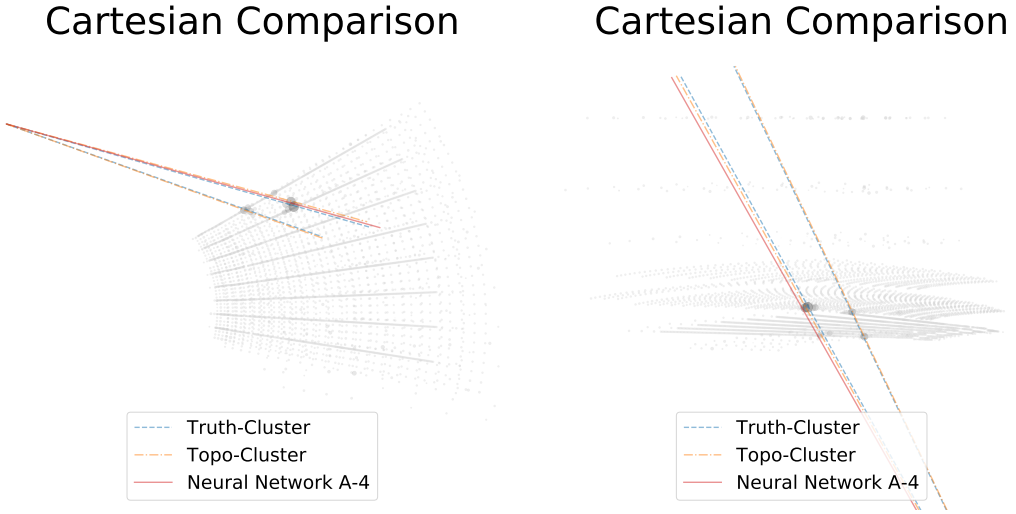


Figure C.2: **Validation Sample - Multiple Clusters**: Notice that the neural network clusters are on the inside. There seems to be a component of the network clusters linked to the center of mass of all clusters within the validation samples. This might be an issues caused by not removing hard radiative electrons.

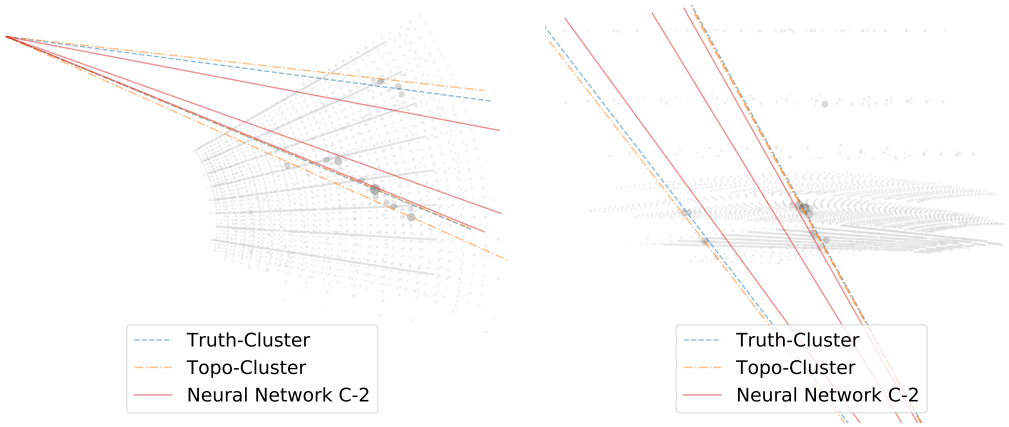


Figure C.3: **Input Images - Mean Heatmap**: to the right is the training set and to the left is the validation set. From top to bottom μ increases ($\mu = [0, 40, 120, 200]$). The mean is taken with respect to samples.

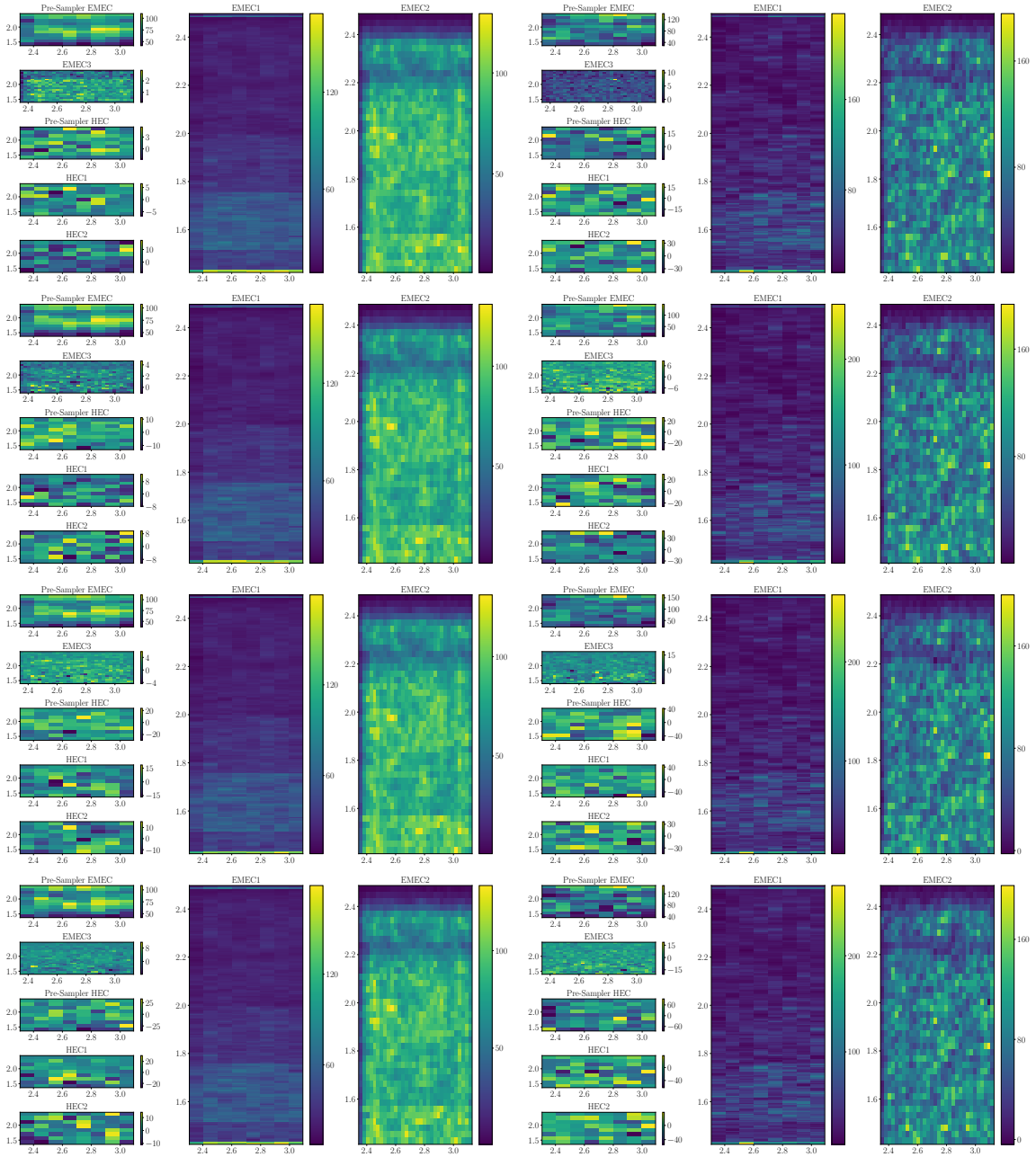


Figure C.4: **Input Images - Stand Deviation Heatmap**: to the right is the training set and to the left is the validation set. From top to bottom μ increases ($\mu = [0, 40, 120, 200]$). The mean is taken with respect to samples.

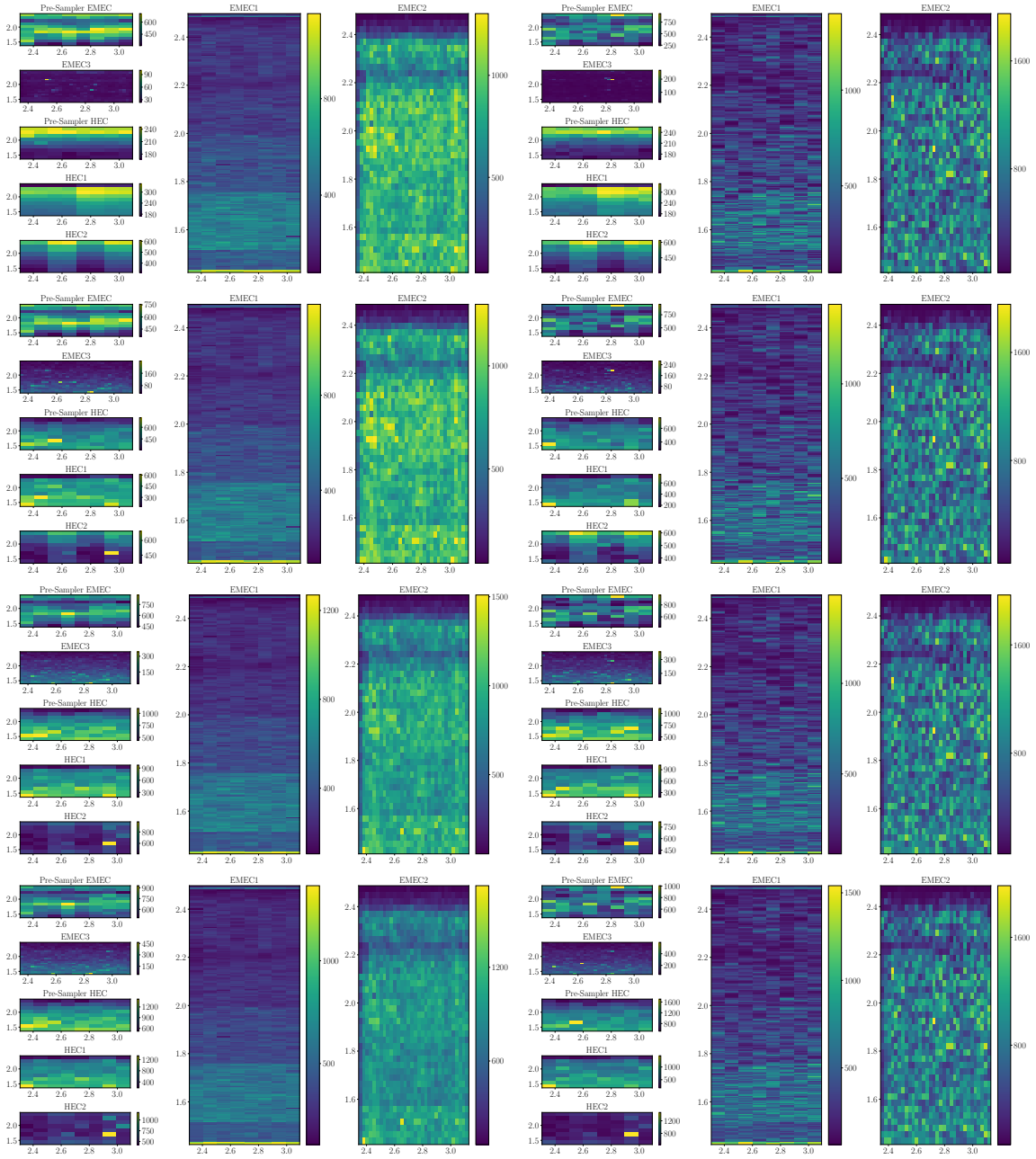


Figure C.5: **TopoCluster-TruthCluster Comparison for different pileup** (rows) and with different energy filters (columns) [Complete, Energy>25 GeV, Energy<25 GeV]

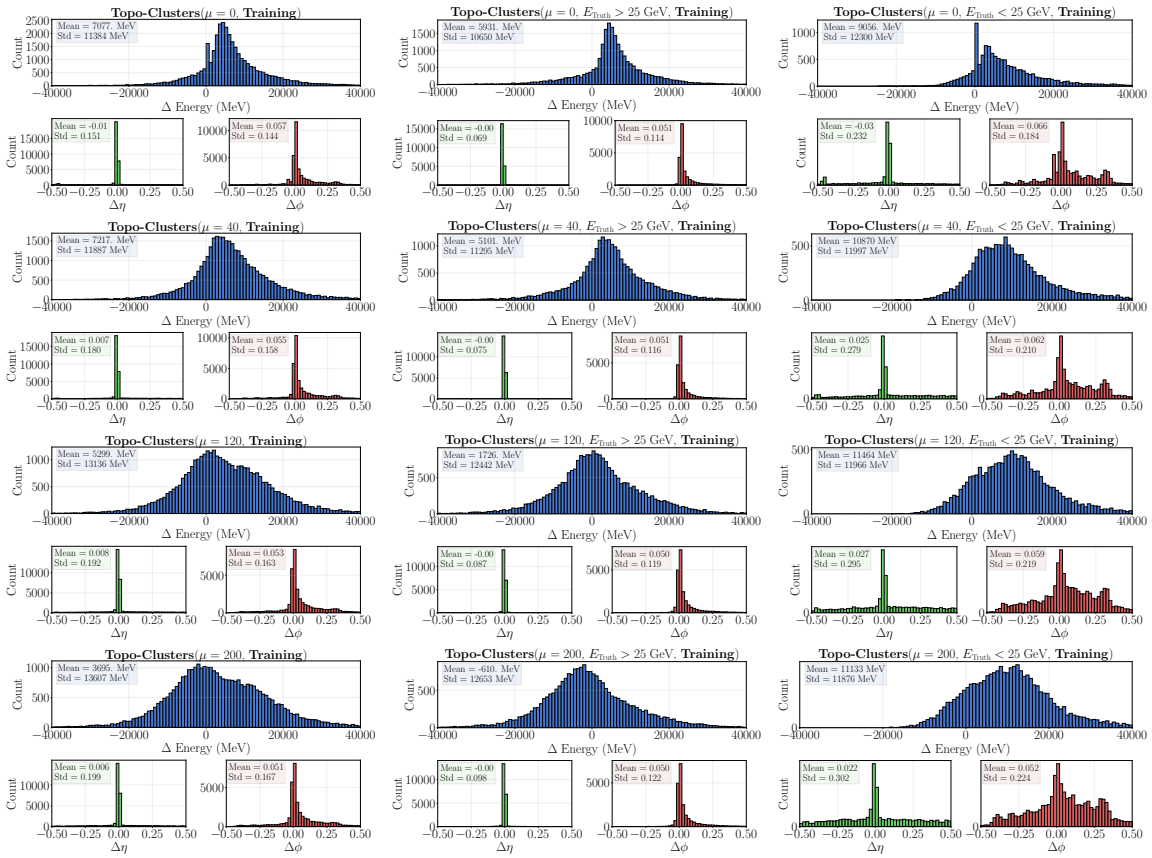


Figure C.6: **Neural Nets Histograms - Validation**: This plot gives an idea of the FWHM of the validation set. This contrasts strongly against Figure 5.2 where even though the STD is similar the FWHM vary greatly. This suggests the definition of truth clusters could be altered especially with regards to η and ϕ. It uses epoch 10, network-A, and validation set.

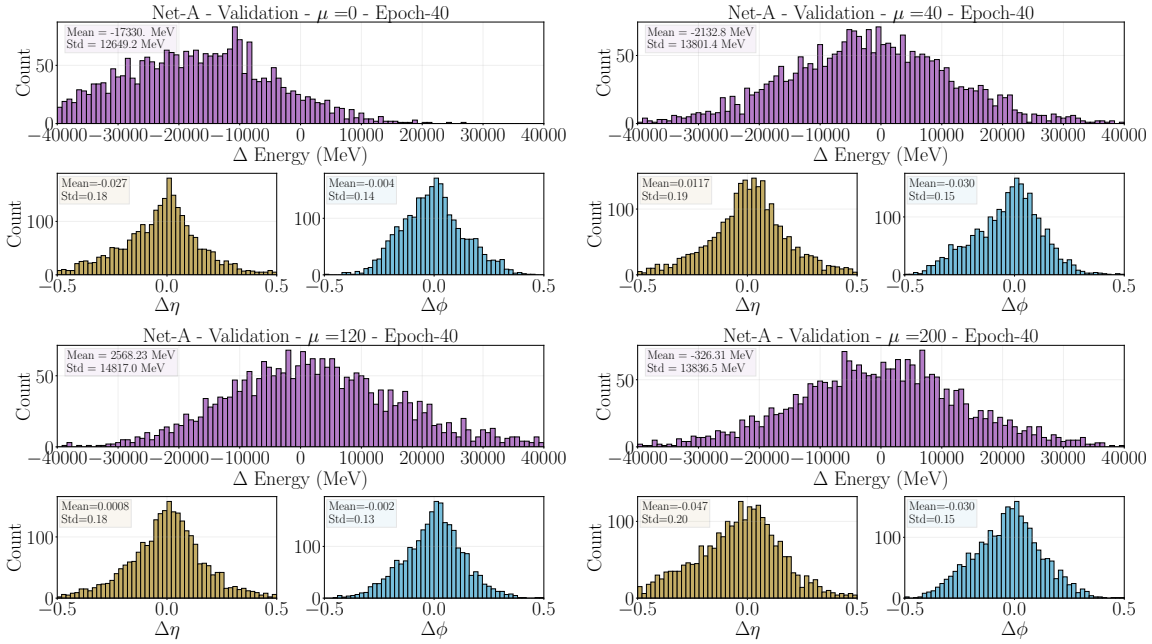


Figure C.7: **Neural Nets Histograms - Training**: This is shows that our objective is properly minimizing regardless of the loss saturating shown in Figure 5.6. It uses epoch 80, network-C, 120 pileup, and training set.

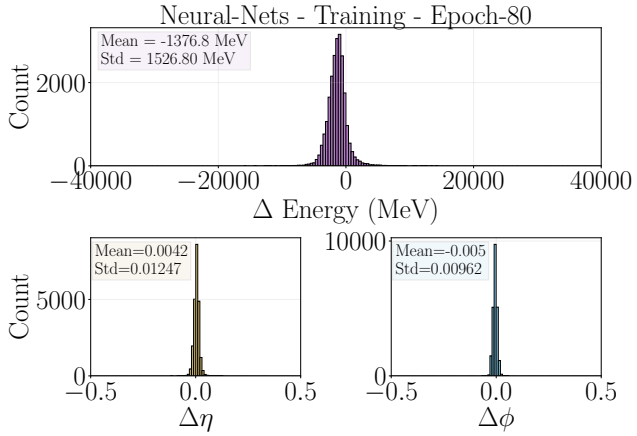


Figure C.8: **Network-B** is a much more complex network. It uses merge layers to create a greater number of paths through the network and a wider variety of path-lengths. Larger layers are given more 2D Convolutional Layers. This network also demonstrates the resiliences of *back-propagation* optimization.

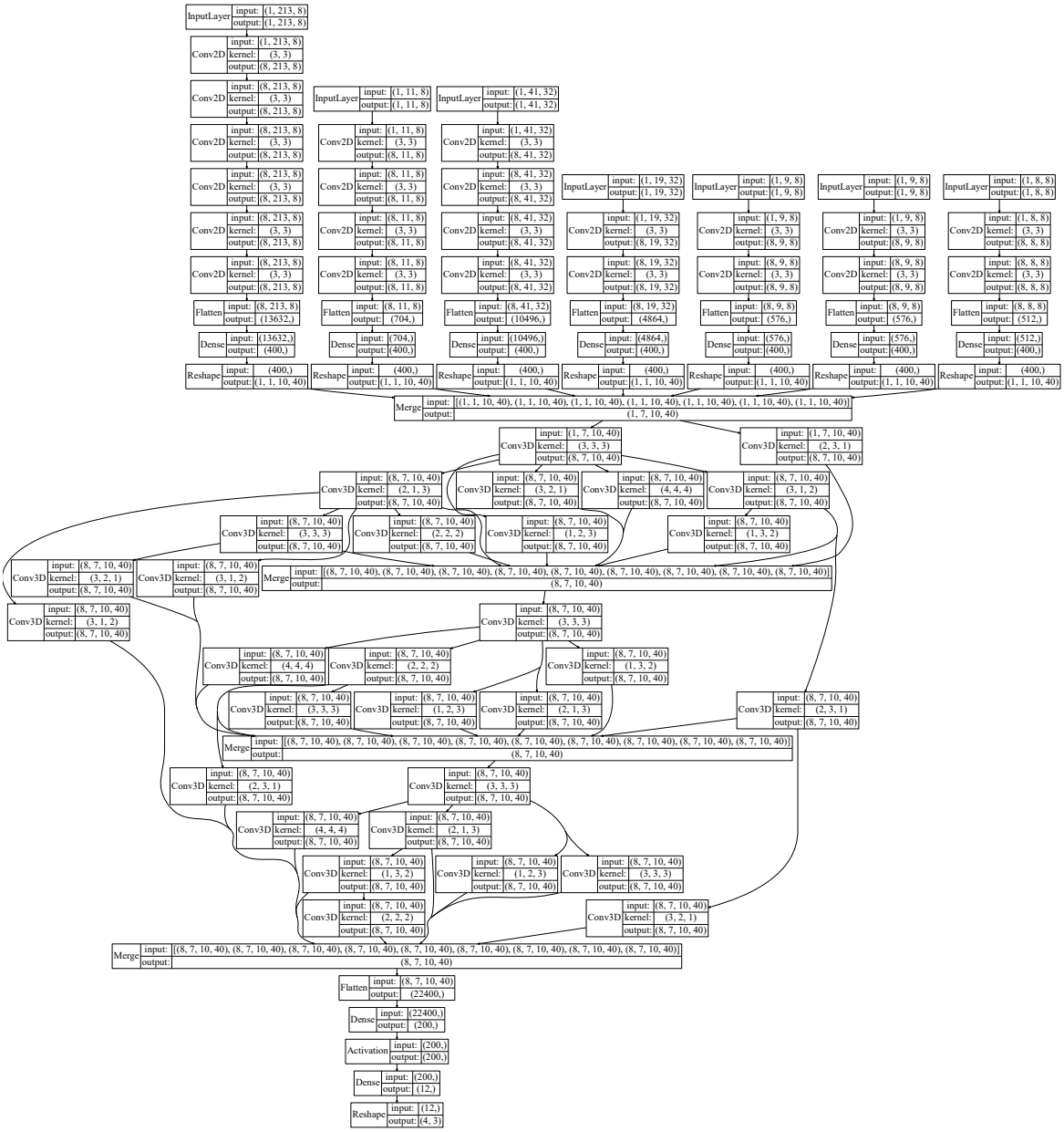


Figure C.9: **Network-C** quite similar to Network-B with the exception of the final layer.

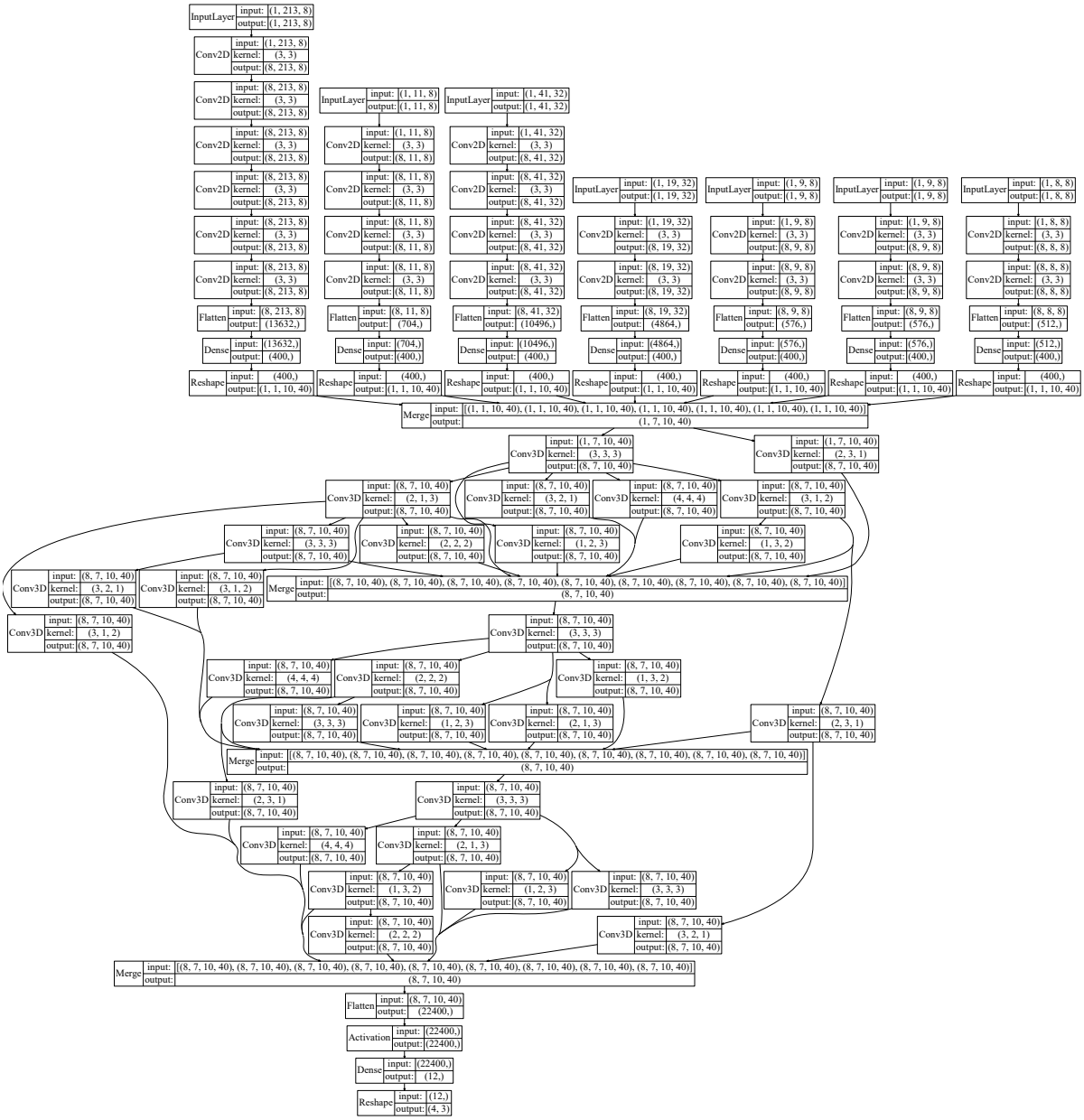
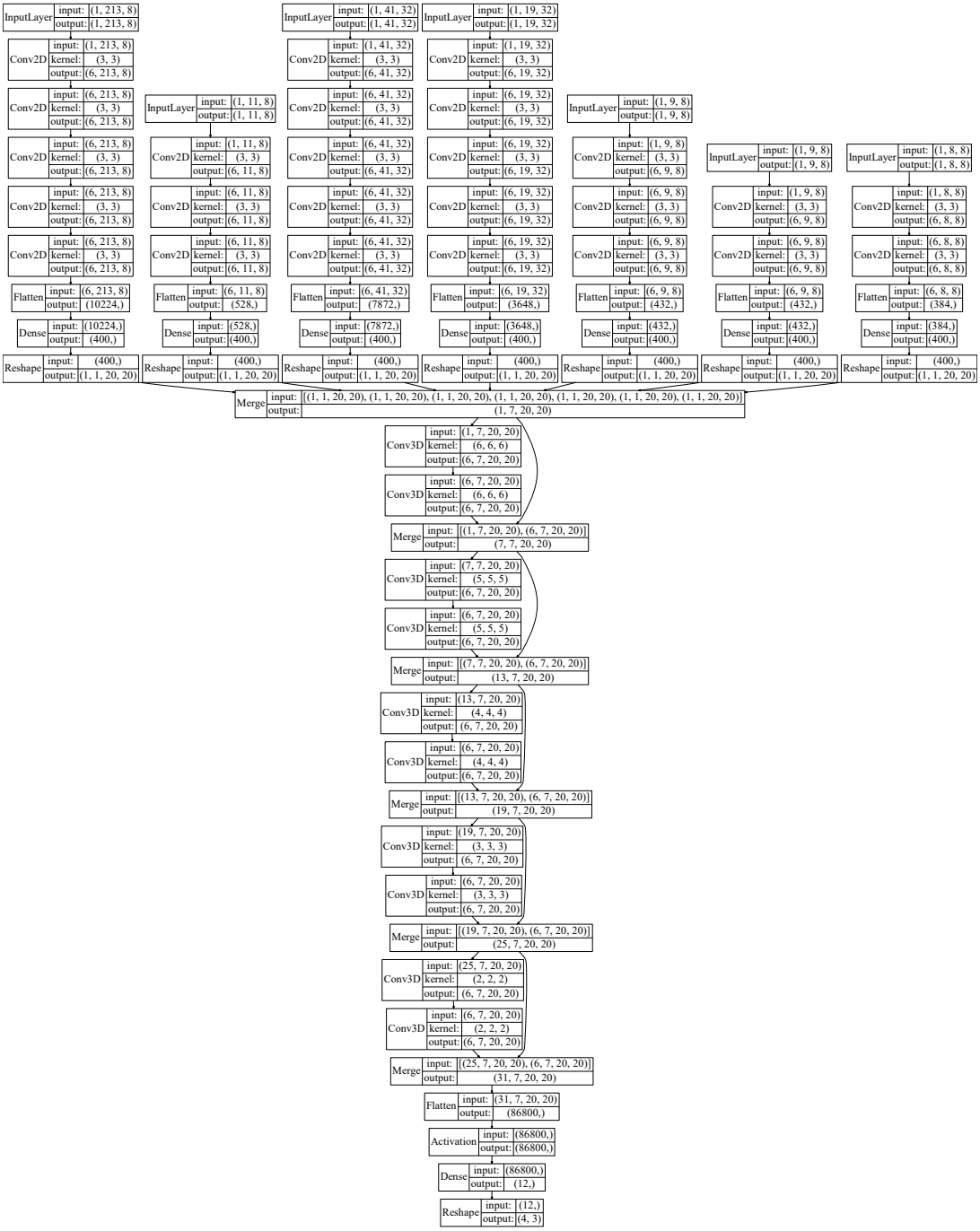


Figure C.10: **Network-D** has a residual 3D end layers. These also form a cone-like shape that decreases in kernel size as the layers progress. This that was suggested by the ATLAS machine learning group as well as something commonly used in papers.



References

[1] ATLAS Collaboration, *Searches for heavy ZZ and ZW resonances in the $\ell\ell qq$ and $\nu\nu qq$ final states in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector*, *ArXiv e-prints* (Aug., 2017) , [1708.09638]. (Cited on page 1.)

[2] ATLAS Collaboration, *Search for new phenomena in high-mass final states with a photon and a jet from pp collisions at $\sqrt{s}$ = 13 TeV with the ATLAS detector*, *ArXiv e-prints* (Sept., 2017) , [1709.10440]. (Cited on page 1.)

[3] G. Apollinari, I. Bjar Alonso, O. Brning, M. Lamont and L. Rossi, *High-Luminosity Large Hadron Collider (HL-LHC): Preliminary Design Report*, http://cds.cern.ch/record/2116337. (Cited on pages 1 and 26.)

[4] G. Arduini, J. Barranco, A. Bertarelli, N. Biancacci, R. Bruce, O. Brning et al., *High luminosity lhc: challenges and plans*, *Journal of Instrumentation* **11** (2016) C12081, http://stacks.iop.org/1748-0221/11/i=12/a=C12081. (Cited on page 2.)

[5] J. P. Araque Espinosa, *Noise dependence with pile-up in the ATLAS Tile calorimeter*, https://cds.cern.ch/record/2017323. (Cited on page 2.)

[6] M. Aaboud, G. Aad, B. Abbott, J. Abdallah, O. Abdinov, B. Abeloos et al., *Identification and rejection of pile-up jets at high pseudorapidity with the ATLAS detector*, *European Physical Journal C* **77** (Sept., 2017) #580, [1705.02211], DOI. (Cited on page 2.)

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma et al., *ImageNet Large Scale Visual Recognition Challenge*, *ArXiv e-prints* (Sept., 2014) , [1409.0575]. (Cited on page 2.)

[8] G. Louppe, K. Cho, C. Becot and K. Cranmer, *QCD-Aware Recursive Neural Networks for Jet Physics*, *ArXiv e-prints* (Feb., 2017) , [1702.00748]. (Cited on page 2.)

[9] J. Pearkes, W. Fedorko, A. Lister and C. Gay, *Jet Constituents for Deep Neural Network Based Top Quark Tagging*, *ArXiv e-prints* (Apr., 2017) , [1704.02124]. (Cited on page 2.)

[10] M. Paganini, L. de Oliveira and B. Nachman, *CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks*, *ArXiv e-prints* (May, 2017) , [1705.02355]. (Cited on page 2.)

[11] L. de Oliveira, M. Paganini and B. Nachman, *Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis*, *ArXiv e-prints* (Jan., 2017) , [1701.05927]. (Cited on page 2.)

[12] J. Cogan, M. Kagan, E. Strauss and A. Schwartzman, *Jet-images: computer vision inspired techniques for jet tagging*, *Journal of High Energy Physics* **2015** (Feb, 2015) 118, DOI. (Cited on page 2.)

[13] L. de Oliveira, M. Kagan, L. Mackey, B. Nachman and A. Schwartzman, *Jet-images — deep learning edition*, *Journal of High Energy Physics* **2016** (Jul, 2016) 69, DOI. (Cited on page 2.)

[14] L. G. Almeida, M. Backović, M. Cliche, S. J. Lee and M. Perelstein, *Playing tag with ann: boosted top identification with pattern recognition*, *Journal of High Energy Physics* **2015** (Jul, 2015) 86, DOI. (Cited on page 2.)

[15] P. T. Komiske, E. M. Metodiev, B. Nachman and M. D. Schwartz, *Pileup Mitigation with Machine Learning (PUMML)*, *ArXiv e-prints* (July, 2017) , [1707.08600]. (Cited on page 2.)

[16] M. E. Peskin and D. V. Schroeder, *An introduction to quantum field theory; 1995 ed.* Westview, Boulder, CO, 1995. (Cited on page 6.)

[17] K. Nakamura and P. D. Group, *Review of particle physics*, *Journal of Physics G: Nuclear and Particle Physics* **37** (2010) 075021, http://stacks.iop.org/0954-3899/37/i=7A/a=075021. (Cited on pages 6, 12, and 23.)

[18] D. J. Griffiths, *Introduction to elementary particles; 2nd rev. version*. Physics textbook. Wiley, New York, NY, 2008. (Cited on page 6.)

[19] S. Thornton and J. Marion, *Classical Dynamics of Particles and Systems*. Brooks/Cole, 2004. (Cited on page 6.)

[20] G. Aad, B. Abbott, J. Abdallah, O. Abdinov, B. Abeloos, R. Aben et al., *Measurement of $W^{\pm}$ and Z-boson production cross sections in pp collisions at $\sqrt{\{}\ s\}$ = 13 TeV with the ATLAS detector*, *Physics Letters B* **759** (Aug., 2016) 601–621, [1603.09222], DOI. (Cited on pages 8 and 88.)

[21] J. L. Feng, *Dark matter candidates from particle physics and methods of detection*, *Annual Review of Astronomy and Astrophysics* **48** (2010) 495–545, DOI. (Cited on page 11.)

[22] J. A. Frieman, M. S. Turner and D. Huterer, *Dark energy and the accelerating universe*, *Annual Review of Astronomy and Astrophysics* **46** (2008) 385–432, DOI. (Cited on page 11.)

[23] D. Galbraith and C. Burgard. http://davidgalbraith.org/portfolio/ux-standard-model-of-the-standard-model/, 2012. (Cited on page 12.)

[24] L. Evans and P. Bryant, *Lhc machine*, *Journal of Instrumentation* **3** (2008) S08001, http://stacks.iop.org/1748-0221/3/i=08/a=S08001. (Cited on page 11.)

[25] M. Lamont, *Status of the lhc*, *Journal of Physics: Conference Series* **455** (2013) 012001, http://stacks.iop.org/1742-6596/455/i=1/a=012001. (Cited on page 11.)

[26] W. Herr and B. Muratori, *Concept of luminosity*, https://cds.cern.ch/record/941318. (Cited on page 12.)

[27] B. Schmidt, *The high-luminosity upgrade of the lhc: Physics and technology challenges for the accelerator and the experiments*, *Journal of Physics: Conference Series* **706** (2016) 022002, http://stacks.iop.org/1742-6596/706/i=2/a=022002. (Cited on page 13.)

[28] ATLAS collaboration, G. Aad et al., *The ATLAS Experiment at the CERN Large Hadron Collider*, *JINST* **3** (2008) S08003, DOI. (Cited on pages 14, 23, and 24.)

[29] ATLAS collaboration, G. Aad et al., *Expected Performance of the ATLAS Experiment - Detector, Trigger and Physics*, 0901.0512. (Cited on pages 14 and 60.)

[30] W. Lampl, S. Laplace, D. Lelas, P. Loch, H. Ma, S. Menke et al., *Calorimeter Clustering Algorithms: Description and Performance*, https://cds.cern.ch/record/1099735. (Cited on page 23.)

[31] P. Speckmayer, T. Carli and C. W. Fabjan, *Energy Measurement of Hadrons with the CERN ATLAS Calorimeter*, Ph.D. thesis, 2008. https://cds.cern.ch/record/1112036. (Cited on page 25.)

[32] S. Mttig, *The online luminosity calculator of atlas*, *Journal of Physics: Conference Series* **331** (2011) 022035, http://stacks.iop.org/1742-6596/331/i=2/a=022035. (Cited on page 26.)

[33] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016. (Cited on pages 28 and 83.)

[34] G. Cowan, *Statistical Data Analysis*. Clarendon Press, 1998. (Cited on page 28.)

[35] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. (Cited on page 28.)

[36] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning*. Springer, 2001. (Cited on page 28.)

[37] K. He, X. Zhang, S. Ren and J. Sun, *Deep Residual Learning for Image Recognition*, *ArXiv e-prints* (Dec., 2015) , [1512.03385]. (Cited on pages 31, 37, and 38.)

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov et al., *Going Deeper with Convolutions*, *ArXiv e-prints* (Sept., 2014) , [1409.4842]. (Cited on pages 31 and 38.)

[39] C. Szegedy, S. Ioffe, V. Vanhoucke and A. Alemi, *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*, *ArXiv e-prints* (Feb., 2016) , [1602.07261]. (Cited on pages 31 and 38.)

[40] W. J. Cook, C. William H, W. R. Pulleyblank and A. Schrijver, *Combinatorial Optimization*. Wiley-Interscience, 1998. (Cited on page 40.)

[41] ATLAS Collaboration collaboration, *ATLAS Computing: technical design report*. Technical Design Report ATLAS. CERN, Geneva, 2005. (Cited on pages 43 and 97.)

[42] GEANT4 collaboration, S. Agostinelli et al., *GEANT4: A Simulation toolkit*, *Nucl. Instrum. Meth.* **A506** (2003) 250–303, DOI. (Cited on pages 45 and 98.)

[43] Z. Marshall and the Atlas Collaboration, *Simulation of pile-up in the atlas experiment*, *Journal of Physics: Conference Series* **513** (2014) 022024, http://stacks.iop.org/1742-6596/513/i=2/a=022024. (Cited on page 45.)

[44] G. Barrand, I. Belyaev, P. Binko, M. Cattaneo, R. Chytracek, G. Corti et al., *Gaudi a software architecture and framework for building hep data processing applications*, *Computer Physics Communications* **140** (2001) 45 – 55, DOI. (Cited on page 45.)

[45] F. Chollet et al., "Keras." https://github.com/fchollet/keras, 2015. (Cited on pages 50, 53, 54, and 97.)

[46] Theano Development Team, *Theano: A Python framework for fast computation of mathematical expressions*, *ArXiv e-prints* (May, 2016) , [1605.02688]. (Cited on pages 50 and 97.)

[47] X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, http://proceedings.mlr.press/v9/glorot10a.html. (Cited on pages 53 and 65.)

[48] T. Dozat, *Incorporating nesterov momentum into adam*, https://openreview.net/pdf?id=OM0jvwB8jIp57ZJjtNEZ. (Cited on page 53.)

[49] ATLAS collaboration, G. Aad et al., *Electron performance measurements with the ATLAS detector using the 2010 LHC proton-proton collision data*, *Eur. Phys. J.* **C72** (2012) 1909, [1110.3174], DOI. (Cited on page 60.)

[50] A. M. Sirunyan, A. Tumasyan, W. Adam, E. Asilar, T. Bergauer, J. Brandstetter et al., *Particle-flow reconstruction and global event description with the CMS detector*, *Journal of Instrumentation* **12** (Oct., 2017) P10003, [1706.04965], DOI. (Cited on pages 81 and 82.)

[51] A. Elliot, *Search for dark matter in association with a leptonically decaying Z boson in the ATLAS detector at the Large Hadron Collider*, Ph.D. thesis, University of Victoria, 3800 Finnerty Rd, Victoria, BC, Aug., 2017. https://dspace.library.uvic.ca//handle/1828/8494. (Cited on page 84.)

[52] T. Kwan, *Measurement of neutral current Drell-Yan production at 8 TeV with the ATLAS detector*, Ph.D. thesis, University of Victoria, 3800 Finnerty Rd, Victoria, BC, Aug., 2017. https://dspace.library.uvic.ca//handle/1828/8426. (Cited on page 84.)

[53] M. LeBlanc, *Characterising the decays of high-pt top quarks and addressing naturalness with jet substructure in ATLAS runs I and II*, Ph.D. thesis, University of Victoria, 3800 Finnerty Rd, Victoria, BC, May, 2017. http://hdl.handle.net/1828/8102. (Cited on page 84.)

[54] J. Chiu, *The performance of the ATLAS missing transverse momentum high-level trigger in 2015 pp collisions at 13 TeV*, Ph.D. thesis, University of Victoria, 3800 Finnerty Rd, Victoria, BC, Sept., 2016. http://hdl.handle.net/1828/7535. (Cited on page 84.)

[55] S. Longo, *Measurements of the Radiation Hardness of CsI(Tl) Scintillation Crystals and Comparison Studies with Pure CsI for the Belle II Electromagnetic Calorimeter*, Ph.D. thesis, University of Victoria, 3800 Finnerty Rd, Victoria, BC, Sept., 2015. http://hdl.handle.net/1828/6707. (Cited on page 84.)

[56] S. Ruder, *An overview of gradient descent optimization algorithms*, *ArXiv e-prints* (Sept., 2016) , [1609.04747]. (Cited on pages 89 and 91.)

[57] A. Mahendran and A. Vedaldi, *Understanding Deep Image Representations by Inverting Them*, *ArXiv e-prints* (Nov., 2014) , [1412.0035]. (Cited on page 92.)

[58] A. Dosovitskiy and T. Brox, *Inverting Visual Representations with Convolutional Networks*, *ArXiv e-prints* (June, 2015) , [1506.02753]. (Cited on page 92.)

[59] S. Arora, Y. Liang and T. Ma, *Why are deep nets reversible: A simple theory, with implications for training*, *ArXiv e-prints* (Nov., 2015) , [1511.05653]. (Cited on page 92.)

[60] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous and Y. LeCun, *The loss surface of multilayer networks*, *CoRR* **abs/1412.0233** (2014) , [http://arxiv.org/abs/1412.0233]. (Cited on pages 94 and 95.)

[61] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam and P. Vandergheynst, *Geometric Deep Learning: Going beyond Euclidean data*, *IEEE Signal Processing Magazine* **34** (July, 2017) 18–42, [1611.08097], DOI. (Cited on page 95.)

[62] J. D. Hunter, *Matplotlib: A 2d graphics environment*, *Computing In Science & Engineering* **9** (2007) 90–95, DOI. (Cited on page 97.)

[63] S. v. d. Walt, S. C. Colbert and G. Varoquaux, *The numpy array: A structure for efficient numerical computation*, *Computing in Science and Engg.* **13** (Mar., 2011) 22–30, DOI. (Cited on page 97.)

[64] A. Collette, *Python and HDF5*. O'Reilly, 2013. (Cited on page 97.)

[65] A. A. Hagberg, D. A. Schult and P. J. Swart, *Exploring network structure, dynamics, and function using NetworkX*, in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, (Pasadena, CA USA), pp. 11–15, Aug., 2008. (Cited on page 97.)

[66] R. Brun and F. Rademakers, *Root - an object oriented data analysis framework*, in *AIHENP'96 Workshop, Lausane*, vol. 389, pp. 81–86, 1996. (Cited on page 97.)

[67] V. Garonne, R. Vigne, G. Stewart, M. Barisits, T. B. eermann, M. Lassnig et al., *Rucio the next generation of large scale distributed system for atlas data management*, *Journal of Physics: Conference Series* **513** (2014) 042021, http://stacks.iop.org/1742-6596/513/i=4/a=042021. (Cited on page 98.)

[68] N. Dawe, P. Waller, E. K. Friis, C. Deil, schmitts, R. Turra et al., *rootpy: 0.8.0*, June, 2015. 10.5281/zenodo.18897. (Cited on page 98.)

[69] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick et al., *Caffe: Convolutional architecture for fast feature embedding*, in *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, (New York, NY, USA), pp. 675–678, ACM, 2014, DOI. (Cited on page 98.)

[70] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro et al., *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015. (Cited on page 98.)

[71] G. Rossum, *Python reference manual*, tech. rep., Amsterdam, The Netherlands, The Netherlands, 1995. (Not cited.)