WATCHMAN: Neutrinos for Non-Proliferation

Application of Neural Networks to Antineutrino Vertex Reconstruction

Stephen Mulligan

August 30, 2021

MSc in Particle and Nuclear Physics

The University of Edinburgh

2021

Abstract

This dissertation looks at the process of antineutrino interaction vertex reconstruction in the WATCHMAN detector in the context of its potential role in ensuring the non-proliferation of nuclear weapons. A novel interaction vertex reconstruction approach is presented using a combination of a traditional reconstruction algorithm and a neural network. The performance of this network, particularly in terms of resolution, is compared with the current standard interaction vertex reconstruction algorithm used by the WATCHMAN collaboration, FRED. It is found that the resolutions achieved by the novel reconstruction approach are on average a factor of 2.00 larger than the equivalent average resolutions of FRED. It is suggested that this performance is indicative of the promise of the application of neural networks to the problem of interaction vertex reconstruction, and that the application of more complex networks should be pursued.

Declaration

I declare that this dissertation was composed entirely by myself.

Chapter 1 introduces and outlines the motivation for the WATCHMAN collaboration and the role of this research within it. Chapters 2 provides the theory necessary for the following chapters, while chapter 3 gives an overview of how the detector functions. They do not contain original research.

Chapters 4 through 6 contain my original work. The work described in Chapter 4 was done in collaboration with Professor Matthew Needham, Dr. Evangelia Drakopoulou, and Dr. Gary Smith. Chapter 5 presents original work done by myself, the basis of which was provided by Dr. Evangelia Drakopoulou.

RATPAC is used for simulation purposes, with FRED used as a benchmark, the libraries for which are produced by the WATCHMAN collaboration. Analysis of outputs from these programs is performed using ROOT.

The QuadFitter scripts used were written by Dr. Evangelina Drakopoulou.

The neural network was written using the Keras library of TensorFlow. Dr. Evangelina Drakopolou provided a basic neural network and analysis structure on top of which I expanded upon.

Personal Statement

The first week of the project were primarily spent familiarising myself with the RATPAC simulation method and output and reading the theory behind neural networks - mainly [1]. The process of analysing the RATPAC outlook also involved re-familiarising myself with the ROOT data analysis framework.

The following two weeks were spent producing satisfactory event data using RATPAC and investigating the function of the QuadFitter scripts by observing their outputs with thsi data. Substantial time was spent editing the QuadFitter script in small ways and observing how this changed the output. My supervisor introduced the idea of the well contained volume cut and I began investigating how best to implement this. I regularly ran RATPAC so as to increase the size of the event dataset. I also began reading about the *DeepSets* architecture in [2].

In the last week of June I began working with the basic neural network Dr. Evangelia Drakopoulou provided me with. My supervisor suggested introducing convolutional layers to improve performance but I ran into difficulty integrating this into the network. At some point I realised that the strange output of the network was the result of having a ReLu activation function on the final layer. Replacing this with a linear activation immediately solved this. The QuadFitter script that only uses hits once is not used after this point due to poor performance.

In the first two weeks of July, in discussion with Dr. Drakopoulou some unnecessary variables were removed from the QuadFitter output that helped improve performance somewhat. As the well contained volume cut had been implemented the inefficiency in the previous simulation method was noticed and simulation of events using the fill shell method replaced this. I began hyperparameter optimisation of the neural network, testing different activation functions etc., which led me to notice the network was stuck in a loss plateau. I spent some time researching methods to combat this and then implemented a cyclical learning rate which solved the issue.

Following this it was noticed that the network began performing better when the time components of quad points were removed from the input features. This was investigated and a number of approaches were tried to solve the issue but in the end the spatial components of the quad points without time components became the default for the input features.

For the next 2 weeks I looked into optimising the network with a convolutional layer and simulated testing datasets at a selection of energy levels. At the end of July I tried to implement the DeepSets network architecture but failed to get the network to produced a reasonable output. I began comparing the output of the network with the equivalent output from FRED.

The first week of August was mainly spent trying to solve the problem of quad point time inclusion as it was thought that leaving this out of the input features may be limiting the performance. I discussed this issue with my supervisor and Liz Kneale. Unfortunately no progress was made. As a result of this I begin writing my dissertation a week later than expected.

In the second week of August I began writing full time. Due to my delay in beginning writing I requested an extension to the project period. At the end of the second week of August, an unexpected personal event arose and I was unable to write for approximately six days. As a result, I extended the deadline for the dissertation further to the 30th of August.

Acknowledgements

I'd first like to thank my supervisor Professor Matthew Needham for the help and understanding throughout this project. The frequency of our meetings throughout the project ensured I was never left alone with a problem long and helped me to work through many challenges.

I must also thank Dr Evangelia Drakopoulou who was always available to answer my questions on the QuadFitter script or the neural network code. Thanks also to Dr. Gary Smith for the help with running RATPAC and FRED, the documentation for which are either scarce or non-existent.

I must of course thank my Personal Tutor, Professor Victoria Martin, for her help and support throughout not only this project but the entirety of this masters programme.

Liz Kneale, 4th year PhD at the University of Sheffield, was also a great help when tackling some problems. I also found speaking at the group meetings organised by Liz to be very helpful in preparing me to give my final presentation.

On a more personal note, I would like to thank my family, good friends, and partner for their unfailing support throughout the entire year. It has been a long and tough year, but it was made brighter by these good people in my life that helped me get through it all.

Go raibh míle maith agaibh go léir.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

In this section, the role of nuclear energy as a power source in the modern world in the context of global warming is discussed before outlining the potential dangers associated with the spread of nuclear technology. An overview of the current non-proliferation framework is then provided and the way in which the WATCHMAN(WATer CHerenkov Monitor for ANtineutrino) Collaboroation seeks to bolster these efforts is explained. Secondary physics goals of the collaboration are also highlighted.

1.1.1 The Role of Nuclear Energy in Addressing Global Warming

Any discussion surrounding the expansion of nuclear energy infrastructure must begin with explaining the role in which this technology will play in the energy transition necessary to combat global warming.

The role of human activity as the main driver of global warming is now well accepted. The recent Intergovernmental Panel on Climate Change(IPCC) Sixth Assessment Report states "It is unequivocal that human influence has warmed the atmosphere, ocean and land". The report also explains that the effects of this human-induced global warming are already being seen around the globe, resulting in weather and climate extremes such as heatwaves, heavy precipitation, droughts and cyclones. The significant increases in greenhouse gases(carbon dioxide, methane etc.) in the atmosphere since the beginning of the industrial revolution (around 1750) are understood to be the main cause of this warming [8].

The need to reduce greenhouse gas emissions is therefore clear and motivates the transition away from energy production methods that result in their creation. Unfortunately, this need to reduce emissions clashes with the still present need to further expand the global energy supply. The International Energy Organisation(IEA), an intergovernmental agency whose work focuses on "the enhancement of the reliability, affordability and sustainability of energy", explains in their World Energy Outlook for 2019 that almost one billion people still do not have access to electricity. Therefore, while it remains essential that energy production continues to increase, it is necessary that this increase must be pursued in a way such that any corresponding increase in greenhouse gas emissions is minimised in order to avoid significant societal and economic damage caused by climate extremes[9].

Nuclear power is well suited to contributing to this essential reduction in emissions. According to the Office of Nuclear Energy at the Department of Energy of the United States of America, nuclear power is an extremely reliable and clean energy source [10]. The plants they discuss run constantly, and are designed to be refueled every 1.5 to 2 years. Furthermore, once a plant is built and the fuel has been mined, the energy generation process of nuclear fission does not emit any carbon dioxide, making nuclear power a significantly cleaner source of energy, in the context of climate change, than traditional sources.

In their most recent World Energy Outlook 2020 report the IEA outlines a scenario for reaching net zero emissions by 2050, a goal of many countries. To achieve this goal, total global carbon dioxide emissions would need to be reduced by 45% from 2010 levels by 2030. This could be achieved by a large reduction in the use of coal power plants without carbon capture technology from 37% to 6% of the global electricity supply. The share of renewables (wind, solar, and hydro power) would increase from 27% to 60%, which would be supplemented by nuclear power contributing 10% of the global energy mix [11].

Nuclear energy technology is clearly an essential part of the global effort to reduce greenhouse gas emissions. However, the technology is not without it's drawbacks. Of particular concern to the WATCHMAN Collaboration is the increased access to fissile material[1] for the production of nuclear weapons that follows the spread of nuclear energy technology (nuclear proliferation).

1.1.2 Danger of Nuclear Proliferation

The same fissile materials used in many nuclear weapons are used as the fuel in nuclear reactors. For example, thermal nuclear reactors can use both uranium-235 and plutonium-239 as fuel sources [12]. The principal isotopes used in nuclear fission based nuclear weapons are the same [13]. Therefore with the spreading of nuclear technology for energy production, access to the material necessary for weapons production increases.

The increased manufacture of nuclear weapons is of great concern. Since the advent of the atomic age, the danger of widespread nuclear weapon proliferation and use has been present. These weapons of mass destruction have immense power, capable of levelling entire cities with a single bomb. According to a relatively recent study, it is thought that a widespread nuclear war would lead to what is known as a “nuclear winter”, a rapid drop in temperature, precipitation and insolation resulting from smoke and dust ejected into the atmosphere after a large nuclear exchange, that would produce catastrophic global consequences [14].

The dilemma therefore exists, where the use of nuclear power is encouraged by it's role in combating climate change, but needs to be expanded in a controlled fashion so as to avoid the aforementioned proliferation of nuclear arms. In order to manage this expansion, it is important that the international community has the ability to monitor how states procure

[1]Material that is capable of undergoing fission after the capture of a low-energy thermal neutron e.g. ^{233}U, ^{255}U, and ^{239}Pu [12].

and utilise fissile material for nuclear reactors to ensure no material is diverted to nuclear weapon production. To this end, a framework of safeguards has been established.

1.1.3 Current Non-Proliferation Safeguards Framework

The control and monitoring of fissile material is the primary focus of the international non-proliferation framework. A combination of a number of international treaties form the basis of this framework and establish the International Atomic Energy Agency(IAEA) as the main international organisation for monitoring the adherence of states to the conditions of these treaties. All stages of the nuclear fuel cycle are subject to monitoring and accounting measures by the IAEA: including the mining of fuel, use of the fuel in a reactor, and how the spent nuclear fuel(SNF) is managed. As well as monitoring the fuel cycle of declared reactors, an important role of the IAEA is the discovery or exclusion of undeclared reactors [15].

The primary non-proliferation aim of the initial phase of WATCHMAN is “to demonstrate high efficiency detection of reactor antineutrinos” [5]. As such it is necessary to outline how the behaviour of known reactors are currently monitored and how the process of undeclared reactor discovery or exclusion proceeds, so as to understand how WATCHMAN could contribute to these activities.

In the case of declared reactors, the safeguards employed by the IAEA are designed to detect any diversions of fissile material from facilities, or detect undeclared processing or production of fissile materials at the declared facilities. Methods such as direct measurement of nuclear material are used to confirm the declared mass(weighing of samples) and composition(gamma-ray spectroscopy) of the material. Environmental sampling within facilities is also employed in order to ascertain whether undeclared materials are present or previously declared materials have been relocated. The remote monitoring of nuclear reactors in real-time can also be performed using satellite and aerial imagery of reactor heat signatures. As the emission of heat can show whether the reactor is on or off this can be correlated to the reactor’s operating power [15].

Detection or exclusion of undeclared reactors has been largely dependent on discovery by intelligence gathering, with some contributions by technological approaches. One method again uses heat output to identify an operating reactor. This can be done using thermal signatures observed by thermal-infrared cameras on satellites or from aircraft. Reactors can also be detected during winter by observing melting of surface ice that could result from a reactor cooling water outlet upstream. Nuclear reactors also produce gases (noble gases and radioactive gases) that can escape into the atmosphere and be detected far from the originating plant. These methods however require intelligence gathering to identify the area that needs to be observed and rely somewhat on weather conditions (presence of ice, clouds, wind). In the case of detecting gases produced by a nuclear reactor, there is also the issue of gases being release from other nuclear facilities nearby that would mask or interfere with the signal from an undeclared reactor [15].

With these methods and their limitations in mind, it is clear to see that the use of technology which could provide detailed information on the operation of a nuclear reactor remotely would be a great asset for both monitoring declared reactors and detecting the presence of undeclared reactors.

1.1.4 Application of Antineutrino Detectors and Role of WATCHMAN

As described in sections 2.1 and 2.2 an antineutrino detector would allow for the measurement of observables such as antinuetrino flux, antineutrino energy spectrum, and the time evolution of both of these observables. From this, the fission rate and reactor power could then be determined. The rate of change of the fission rate is determined by the initial composition of the entire fuel material used in the reactor as the measured flux is the flux from the entire reactor. Unlike the direct measuring of nuclear material described in section 1.1.3, this provides a form of “bulk accountancy”. An example of where this would be particularly useful is in cases where a reactor shutdown occurs and it is impossible to access the reactor to implement safeguards. Fissile material could potentially be removed from the reactor core and traditional safeguard methods are poorly suited for establishing this, but monitoring of the antineutrino flux when the reactor is reactivated would highlight any changes in the core composition [15].

In the case of detecting undeclared reactors, an antineutrino detector provides a number of improvements over the methods described in section 1.1.3. An antineutrino detector would not suffer from the same dependence on weather conditions as current remote monitoring techniques. In the case a reactor was discovered, it could then be possible to determine the operational status and from that the thermal power output of the reactor, thus allowing for the determination of the fuel composition (see sections 2.1 and 2.2).

The WATCHMAN detector is planned to be a kiloton scale gadolinium-doped(Gd) Cherenkov antineutrino detector [5]. This is a large cylindrical detector lined with photomultiplier tubes(PMTs - see section 3.2) with a water target. The large volume of the detector(and the water target) is necessary to increase the likelihood of an antineutrino interaction. When antineutrinos passing through the detector interact by inverse beta decay(see section 2.2) a positron is produced. This positron then generates light via the Cherenkov effect (see section 2.3). The PMTs of the detector then detect this light, recording the position of the PMT and the time of the detection (hit data). The Gd-doping of the water is necessary for distinguishing antineutrino events from other background processes(see 2.2).

It is proposed that the WATCHMAN detector would have the ability to exclude the existence of an operating 10 Megawatt thermal(MWt) reactor with high confidence in an approximately 25 kilometer radius [5]. If this can be achieved, a detector of this type could be employed to monitor the operation of declared reactors or confirm the non-operation of reactors within a certain area, on a cooperative basis with the local government. As discussed previously, this antineutrino detection capability would be a great addition to the current safeguards used for monitoring declared reactors and searching for undeclared reactors.

1.1.5 Physics Aims

As well as the non-proliferation aims of the WATCHMAN collaboration, the construction of a detector of this type would allow for the investigation of a number of physics goals.

The detection of neutrinos emitted by supernovae is of great importance to the study of these events. The brightest and one of the most recent supernova was observed in

1987 in Chile, SN1987A. When a star explodes in a supernova, it emits a huge number of neutrinos. SN1987A emitted 10^{58} neutrinos when it went supernova, however a sample of only 20 of these neutrinos were detected on Earth [16]. It is a testament to the value of supernova neutrinos to the study of these phenomena that based on this sample there has been an average of one theoretical paper published every 10 days for the last thirty years (as of 2015). It is then clear how beneficial a larger sample of supernova neutrinos would be [5].

A Gd-doped antineutrino detector is useful in this context as the majority($\sim 88\%$) of a supernova neutrino signal comes from IBD interactions. The Gd-doping would allow for individual tagging of events, which would then allow for the extraction of the electron antineutrino time structure of the supernova neutrino burst. WATCHMAN would also be able to identify the direction of neutrinos, making WATCHMAN the only detector in the world capable of providing both direction and flavour identification of supernova neutrinos. As such, it would be extremely useful to have a detector of this type the next time a supernova occurs, which is estimated to happen about once every thirty years [17]. A detector of this type would be able to instantly alert other experiments such as LIGO and IceCube as the neutrino wave of the supernova is passing through the earth. In a supernova, the neutrinos are generated before any light and so the neutrino wave of a supernova generally precedes the arrival of supernova light by a few hours, giving these experiments time to prepare and allowing them to optimise their observations of the supernova [5].

In recent years a discrepancy in the ratio of the observed antineutrino event rate to the predicted event rate has been discovered. Specifically, improved data and models of reactor antineutrino emissions predict more antineutrinos than are observed. This is referred to as the *Reactor antineutrino anomaly*(RAA) [18]. One possible solution is the existence of another neutrino oscillation channel in the form of a fourth undetectable or *sterile* neutrino [19]. In the event that a compact accelerator was located nearby the detector site, the WATCHMAN detector would be able to perform searches for such sterile neutrinos..

The detector would also be able to perform searches for non-Standard neutrino interactions. This would be achieved by observing large number of events such as antineutrino electron scattering ($\bar{\nu} + e^- \rightarrow \bar{\nu} + e^-$) and looking for discrepancies between the observed and predicted event counts of such interactions [5].

1.2 Project Goal

The process of reconstructing the position of an interaction begins once the hit data has been collected by the detectors PMTs. It is necessary to reconstruct the point where the antineutrino interacted, the interaction vertex, so as to ensure that this is a true antineutrino interaction, and not a background event. The backgrounds experienced within the detector are studied and model so that a certain volume within the detector can be defined as the *fiducial volume.* The fiducial volume is a region within the detector where the background is deemed to be sufficiently low such that any event reconstructed to be within this volume should be considered a signal event.

The process of vertex reconstruction is generally handled by a reconstruction algorithm. Though these traditional algorithms can be very effective, they often suffer from systematic errors. For example, the Quad Fitter algorithm suffers from an issue known as *fitter pull*, which is a systematic shift of the fitted position along the direction of travel of the charged particle [20].

A machine learning approach is thought to be well suited to accounting for the systematic errors in these algorithms. In particular, this project will focus on the use of neural networks. While a neural network may struggle to accurately reconstruct a vertex location when only supplied with raw hit data, it is thought that a promising approach would be to apply a traditional vertex reconstruction algorithm to the hit data, before passing the output to the neural network . The neural network would then be able to learn any systematic errors the algorithm suffers from, and ideally would produce a more accurate result. This network would need to be trained on simulated events as it would be necessary to have the true vertex location as truth values for the network training process.

To summarise, the goal of the project is to simulate antineutrino events in the WATCHMAN detector and investigate how the application of neural networks to the problem of antineutrino interaction vertex reconstruction could improve the accuracy of traditional fitter algorithms.

Chapter 2

Theoretical Framework

2.1 Neutrino Production by Nuclear Fission Processes

Nuclear fission based reactors produce energy through the splitting of the nucleus of a heavy parent nuclei into fission fragments (lighter daughter nuclei and neutrons). The difference in rest mass energy between the parent nucleus and the fission fragments is converted into the kinetic energy of the fission fragments. These fission fragments collide with the atoms in the surrounding medium and lose their kinetic energy as it is converted into thermal energy, which is then used to generate electricity [12].

Antineutrino production does not occur during this fission process, but afterwards as the unstable daughter nuclei decay via β-decay as follows

$$^{A}_{Z}N \rightarrow ^{A}_{Z+1} N' + e^{-} + \bar{\nu}_e \tag{2.1}$$

One fission event generally leads to two fragments, with each fragment undergoing β-decay an average of three times, leading to the production of six antineutrinos for each fission event. The exact number per event and therefore overall reactor anitneutrino flux is dependent on which nuclides undergo fission as a result of differing fission fragment yield. For example, the most commonly used nuclides in both reactors and nuclear weapons (^{235}U, ^{238}U, ^{239}Pu, and ^{241}Pu) all differ in fission fragment yield, and so produce different antineutrino emission rates and spectra. As most reactors use the fission of a combination of nuclides for generating energy the overall neutrino flux from a reactor will generally be made up of contributions from multiple sources [15].

It is then straightforward to see that the reactor antineutrino flux can be written as

$$\phi_{\nu}^{equil}(E_{\nu}, t) = R(t) \sum_{k} \alpha_k(t) S_k(E_{\nu}, t) \tag{2.2}$$

where $R(t)$ is the total fission rate, $\alpha_k(t)$ the fraction of fissions from the k^{th} nuclide, and $S_k(E_\nu, t)$ the flux from the k^{th} fissioning nuclide as a function of antineutrino energy and time. This expression is an equilibrium approximation as the timescale for changes in R and α_k is much longer that that for β-decay lifetimes i.e. the fraction of nuclides and their fission rate can be assumed constant on the time scale of a β-decay. This highlights that the antineutrino flux from a fission reactor would therefore be a proportional sum of contributions from each type of fissile fuel present.

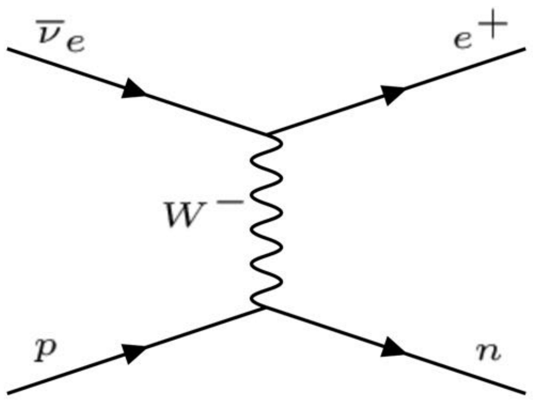


Figure 2.1. Tree level diagram of inverse beta decay process illustrating the scattering by exchange of a W^- boson. Taken from [3].

The antineutrino flux can then be related to the reactor thermal power by the following

$$P_{th} = R\sum_k \alpha_k E_k \tag{2.3}$$

where here E_k is the mean energy per fission of the k^{th} nuclide [15].

2.2 Antineutrino Detection: Inverse Beta Decay and Gadolinium Doping

The inverse beta decay (IBD) interaction that is exploited to detect antineutrinos in the detector proceeds as

$$\bar{\nu}_e + p \rightarrow n + e^+ \tag{2.4}$$

The antineutrino scatters off quasi-free protons in water to produce a positron and a neutron [5]. This process has a low threshold energy, with $E_{\bar{\nu}} = 1.806$ MeV and proceeds via the exchange of a W^- boson [3]. The process is illustrated in figure 2.1.

IBD is an attractive interaction to work with for neutrino detection as it is considered to have a relatively large cross section when compared to other neutrino interactions [21].

The produced positron will then cause the emission of photons by the Cherenkov effect, as outlined in section 2.3. However, there are background processes that can also produce Cherenkov radiation, and so a method of distinguishing and IBD interaction from these background processes (discussed in section 3.1.2) is necessary.

To this end, Gadolinium (Gd) doping of water is employed, which uses the other product of the IBD interaction, the neutron. Neutrons themselves are difficult to detect as they are neutral and so do not interact via the Coulomb force [22]. Fortunately, Gd has the largest known neutron capture cross section of all stable isotopes, and this allows for large neutron tagging efficiency [3]. Even at a level of 0.1% doping, Gd captures approximately 85% of all neutrons produced in the IBD interaction [5].

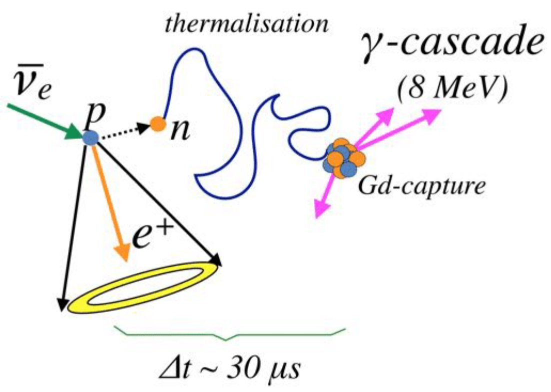


Figure 2.2. Diagram of IBD and subsequent neutron tagging by capture of neutron by Gd isotope [3]

After the neutron is produced, it will interact with the water until it is thermalised. When the neutron has achieved thermal energies, the neutron can be captured by a Gd nucleus, producing an excited intermediate state. When this excited Gd de-excites, it does so by producing a cascade of γ-rays. This reaction, as illustrated in figure 2.2, proceeds as

$$n + {}^{X}\mathrm{Gd} \rightarrow {}^{X+1}\mathrm{Gd}^{*} \rightarrow {}^{X+1}\mathrm{Gd} + m\gamma \tag{2.5}$$

where $3 \leq m \leq 5$ is the number of photons in the cascade. The total energy of the cascade is dependent on the isotope of Gd that is involved in the interaction, but on average can be said to be approximately ~ 8 MeV [3].

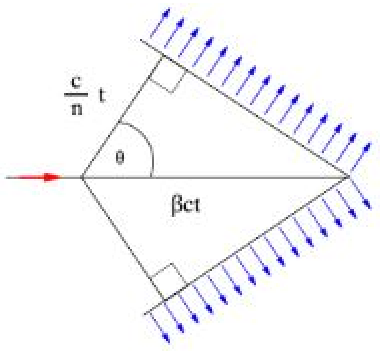
The implementation of this neutron tagging method by Gd doping produces a time correlated signal that allows the IBD interaction to be distinguished from other Chernekov radiation producing processes. This is possible as the mean time between the neutron production and neutron capture is quite short, at $\sim 30\mu s$, which gives a very efficient correlation between the prompt signal of positron Cherenkov radiation and the signal from the γ-ray cascade [3].

The total number of detectable antineutrino events can be written using equations (2.2) and (2.3) and suppressing time and antineutrino energy dependences to find the expression

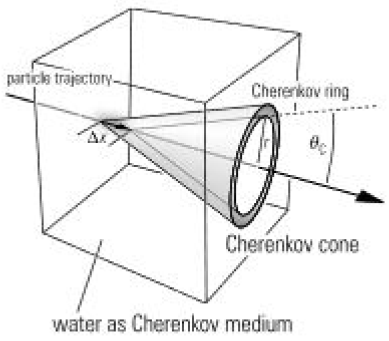
$$N_{det} = \left(\frac{\epsilon N_T \sigma}{4\pi L^2}\right)(P_{e\rightarrow X}(L))\frac{P_{th}}{\sum_k \alpha_k E_k}\sum_k \alpha_k S_k \tag{2.6}$$

where ϵ gives the signal detection efficiency, σ the IBD interaction cross section ($\approx 10^{-43}\mathrm{cm}^2$), N_T the number of interaction targets (i.e. quasi free protons in water), L the distance from the reactor to the detector, and $P_{e\rightarrow X}(L)$ the probability of the neutrino oscillation as a function of the distance L. Neutrino oscillation must be accounted for as the IBD interaction only occurs with electron antineutrinos. Factors ϵ, N_T and σ should all be known, while S_k is well known for the relevant nuclides in nuclear reactors when fissioned by thermal neutrons [15].

Potentially unknown factors are L (for the case of locating undeclared reactors), P_{th}, and α_k. Dependent on which of these factors are known, it could then be possible to remotely infer by studying the reactor antineutrino flux the distance to the reactor, the power level of the reactor, or what fuel the reactor is using [15].



(a) Emission of Cherenkov radiation by charged particles [23]



(b) Production of Cherenkov ring [16]

Figure 2.3. Side and three dimensional views of Cherenkov effect

2.3 Cherenkov Effect

The Cherenkov effect is utilised by the WATCHMAN detector to detect the presence of positrons produced by IBD. When a charged particle moves through a dielectric medium of refractive index n with a velocity greater than $c_n = c/n$ (the speed of light in the medium), electromagnetic radiation is emitted along the path of the charged particle known as *Cherenkov radiation* (see figure 2.3(a)) [16]. The threshold velocity can be used to define a threshold β as follows

$$v \geq \frac{c}{n} \Rightarrow \beta = \frac{v}{c} \geq \frac{1}{n} \tag{2.7}$$

The electromagnetic radiation is emitted in a cone of a well defined radius(r_c) and Cherenkov angle(θ_c)(see figure 2.3(b)).

$$\theta_c = \cos^{-1}\frac{1}{n\beta} \tag{2.8}$$

$$r_c = d\ \tan\ \theta_c \tag{2.9}$$

where d is the distance of the base of the cone from the source.

2.4 Neural Networks

Neural networks are a set of biologically inspired algorithms that attempt to mimic the structure of a brain [24]. They consist of layers of neurons that are connected to one another. A simple neural network with a single hidden layer is shown in figure 2.4, where each layer consists of a number of nodes connected to nodes in each adjacent layer by weighted “synapses”. The first layer is the input layer where the data or features are input. A hidden layer then follows and the network ends in an output layer which outputs the predicted value. Training of the model on data alters these weights in order to improve the accuracy of the network. The number of nodes and layers, as well as types

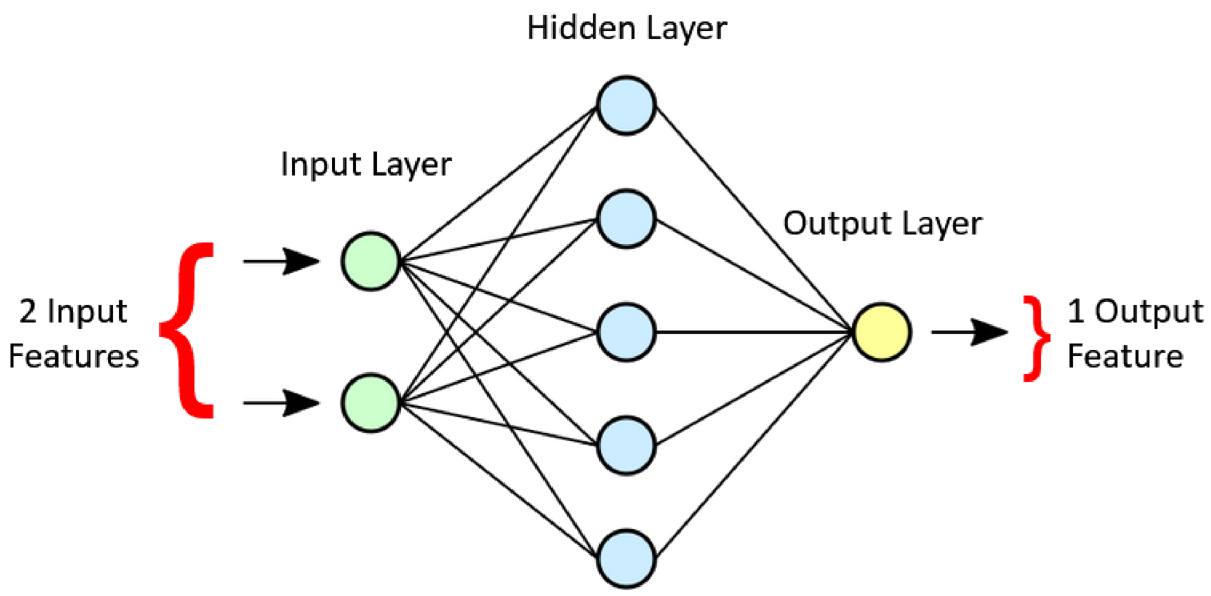


Figure 2.4. Example of single layer neural network. The leftmost layer is the input layer, the central layer is the hidden layer, and the final layer of a single neuron is the output layer.

of layers, can be varied between models to achieve better performance. The basic form of layer described here are generally referred to as *dense layers*.

More specifically, a neural network is a type of adaptive basis function model, which is a model of the form

$$f(\mathbf{x}) = w_0 + \sum_{m=1}^{M} w_m \phi_m(\mathbf{x}) \tag{2.10}$$

The function $\phi_m(\mathbf{x})$ is the m'th basis function, the parameters of which are learned by training the network on the data set [1]. The weights w_n are also updated in this training process.

Training of a network generally proceeds as follows. Training data is input into the network. The output values of the network are compared to the true values associated with the data. A cost function is used to compute how inaccurate the prediction is in comparison with the truth values. The weights of the model are updated and then the procedure is repeated to check for improvement, with the aim of minimising the output of the cost function.

Activation functions are used as basis functions to introduce non-linearity into the system, which allows for backpropagation(training of the model by linear descent) [24]. There are many different types of activation functions with problem specific applications.

A convolutional neural network is a type of neural network in which the nodes have *local receptive fields* in which the weights are shared. This reduces the parameters of the network and allows the network to recognise similar features in different sections of the input data e.g. similar features in different parts of an image. This helps the network to be translation invariant with respect to the ordering of data [1].

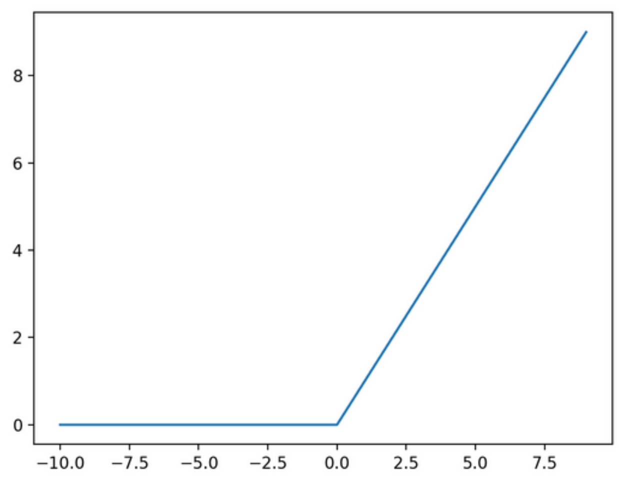


Figure 2.5. A plot of the ReLU activation function from [4]. Inputs along the x-axis do not begin to produce a non-zero output until the values are greater than zero.

2.5 Rectified Linear (ReLU) Activation Function

The primary activation function used in this research is the ReLU activation function. This is a “piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero” [4]. Networks using this activation function are generally easier to train and achieve better performance.

Chapter 3

Detector Design and Backgrounds

3.1 Detector Design

The WATCHMAN detector (see figure 3.1 is a large cylindrical detector filled with Gd doped water. The detector that is used in the simulations for this project has a diameter and height of 20.0 metres. It is divided into an outer *veto region*, which discards any event detected in this region, and an inner *target region*. These regions are separated physically by the PMT support structure of the detector. The support structure is lined with PMTs. The detector geometry used in this project has 20% PMT coverage.

Within the target region, a smaller *fiducial volume* is defined. Fiducial comes from the Latin word *fiducia*, meaning trust. In effect, any event detected within the fiducial volume should be "trusted" as having come from an antineutrino interaction. This volume is based on the positions of the particle interactions as reconstructed from variations in arrival times of Cherenkov light at the PMTs. The region outside the fiducial volume, but still within the target region is referred to as the *buffer region*. As the name suggests, this reduces background from γ-rays and neutrons in the fiducial volume coming from the PMTs and external radiation [5]. The accuracy of the reconstruction process of interaction vertices is therefore essential to distinguishing signal from background.

3.1.1 Location

The WATCHMAN detector will be located at the Boulby Underground Laboratory in Yorkshire, UK . The site is located 25 km from the Hartlepool Nuclear Power Station, which will be used as the source of antineutrinos for the experiment [25].

For neutrino detecting experiments it is important to reduce backgrounds from non-neutrino sources that could potentially imitate the signal of interest as much as possible. In order to minimise backgrounds it is necessary to significantly shield the detector. The most effective way of achieving this is placing the detector deep underground, which helps eliminate a large majority of potential background sources. To this end, the site at Boulby Underground Laboratory for the detector is located 1177 metres underground [26].

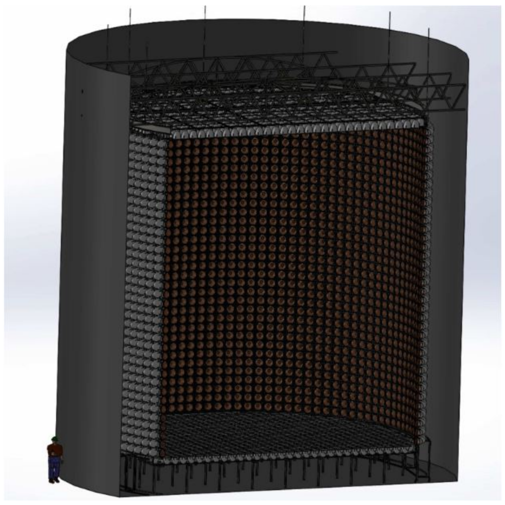


Figure 3.1. Diagram of WATCHMAN detector displaying the Gd-doped water target region, the inward-facing and outward facing PMTs on the PMT support structure, and the veto region. From [5]

3.1.2 Backgrounds

This location deep underground still fails to exclude some sources of background. For a Gd-H2O detector such as WATCHMAN, these backgrounds are as follows:

1. Pairs of accidental coincidences
2. Pairs of muogenic fast neutrons
3. Beta-n decays from long-lived muogenic radionuclide on oxygen in the water

These accidental coincidences occur when two physically independent interactions happen to appear close in time. If these are within the same time interval as the antineutrino prompt signal and γ-cascade signal ($\sim 20 - 50\,\mu s$ depending on Gd loading [5]), they will mimic the correlated signal. This kind of background can be reduced by *fiducialization*(process of defining a fiducial volume) but even after this is implemented, there can still be contributions from ambient radioactivity from PMTs and detector walls [5].

Fast muogenic neutrons (neutrons produced by incident muons) that are not caught in the veto region can result in spallation in the target region, leading to the production of neutron pairs. These neutron pairs will thermalise and produce γ-rays as a result of being captured by Gd atoms. This will happen with a time interval and energy range similar to reactor antineutrinos. This occurs when a muon interacts outside of the detector volume, producing a neutron that then travels into the detector, and so does not produce a veto signal [5].

Muon interactions can also produce long-lived radionuclides. When these decay, they can produce correlated signals mimicking the characteristics of an IBD interaction. The long lifetime of these radionuclides can make the vetoing of the associated background signal difficult [5].

Fortunately, models of the detector have found that even with conservative assumptions about the levels of these backgrounds, the WATCHMAN detector should be capable of

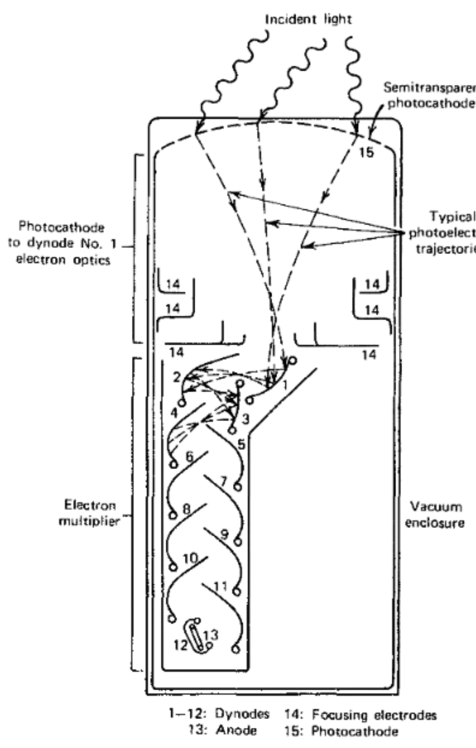

Incident light
Semitransparent photocathode
15
Photocathode to dynode No. 1 electron optics
Typical photoelectron trajectories
14
1
2
3
4
5
6
7
8
9
10
11
12
13
Electron multiplier
Vacuum enclosure
1–12: Dynodes 14: Focusing electrodes
13: Anode 15: Photocathode

Figure 3.2. A simplified PMT structure. From [6]

achieving its goal of high-confidence detection of reactor IBD signal using two 30 day periods of reactor on and reactor off data [5].

3.2 Photomultiplier Tubes (PMTs)

Photomultiplier tubes are employed in order to convert photon signals into usable current pulses. Here a simplified structure, pictured in figure 3.2 for a typical PMT is discussed.

Photons incident on the PMT pass through a transparent casing (necessary to sustain a vacuum) and interact with the *photocathode*. This photocathode is responsible for the photoemission process, which proceeds as: (1) absorption of the photon and transfer of the photons energy to an electron, (2) movement of the electron to the surface of the material, (3) emission of the electron from the material. In order for this to occur the electron must receive enough energy from the incident photon in order to migrate to the surface and overcome the potential barrier, or the *work function*. This work function is usually in the order of electronvolts, and as the threshold for the IBD interaction is 1.8 MeV, with the positron receiving the majority of this energy, photons produced by the positron should have no issue overcoming this potential barrier [6] [15].

The number of electrons produced by the photocathode is generally too low to be used as a signal and so the number of electrons must be increased. The emitted electrons are accelerated across a potential to an electron multiplier structure. This structure consists of a number of *dynodes* that utilise secondary electron emission to increase the number of electrons. These electrons are then reflected onto following dynodes, until they are collected at an anode at the end of the PMT. Though these secondary electrons are lower in energy than the original electrons, the total charge of the electrons is hugely increased

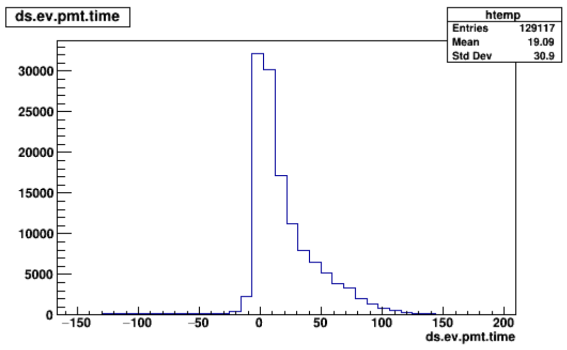


Figure 3.3. PMT hit time distribution (ns) for 4 MeV positrons simulated in RATPAC.

by the time they reach the anode, producing a usable signal [6].

In the WATCHMAN detector, the PMTs will begin recording hits after the sixth hit within a 200 ns window is detected. This sixth hit is considered the *trigger hit* and is taken as the zero for all hit other hit times i.e. all hit times are recorded relative to the trigger hit. This results in negative hit times (though this should not effect the calculations of the quad-fitter described in 5.1) [27]. An example of this PMT hit time distribution for 4 MeV positrons simulated in RATPAC (see section 4.1) is shown in figure 3.3.

Chapter 4

Simulation

4.1 Geant4, RATPAC and ROOT

GEANT4 is a simulation toolkit for studying the passage of particles through matter. It provides precise simulation of electromagnetic interactions of particles with matter, covering a wide range of energies (from 250 eV to the TeV range). This toolkit exploits object-oriented programming. To achieve this, it is implemented in the C++ programming language[28].

The toolkit supports the creation of a geometrical model consisting of different shapes and components. Sections of these models can be defined "sensitive" detector elements that record information (hits) about simulated particles as they pass through these sensitive regions. The toolkit also allows the user to specify the characteristics of the generated primary particles, such as particle type, energy, and momentum and source distribution[28].

For the simulation portion of this project, the program RATPAC will be used, which can be explained as "a layer over Geant4".

RATPAC outputs the results of the simulation in a .root file format so as to aid further processing and analysis, which is carried out in the ROOT data analysis framework. On its website, ROOT is described as "A modular scientific software toolkit. It provides "all the functionalities needed to deal with big data processing, statistical analysis, visualisation and storage"[29]. It is primarily written in C++, with some integration with other languages. This is the program also used to analyse the output of the reconstruction algorithm FRED which is used as the benchmark against which the performance of the neural network is judged (see section 5.4.3).

4.2 Simulation Procedure

4.2.1 Initial Detector Fill Simulation and Well Contained Events

The initial simulation process begins with filling the entire detector geometry with positrons with randomised momentum, imitating IBD events occurring throughout the detector tank. The detectable antineutrino spectrum runs from just below 2 MeV to approximately 10 MeV. In this study a range of energies from 2 to 8 MeV are simulated. The

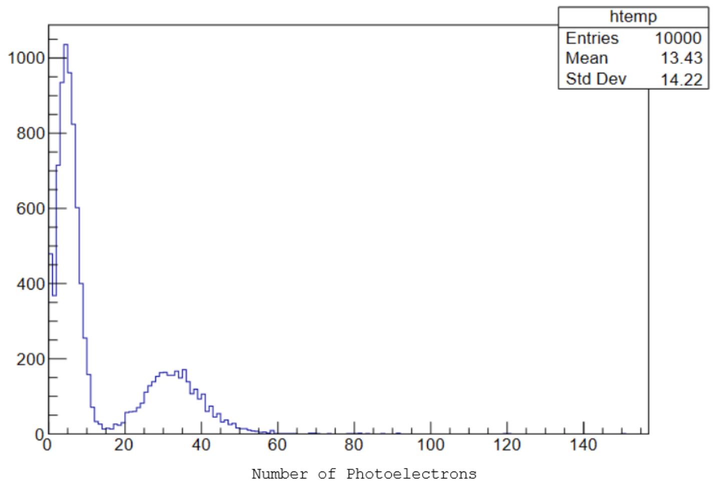


Figure 4.1. Number of photoelectrons generated by each event

initial simulation uses 4 MeV positrons as this is the energy around which the detectability of the antinuetrinos peaks [30].

The detector fill method fills the entire detector geometry, which means events are being simulated in the veto region of the detector. These events would be discarded by the veto PMTs and so it is inefficient to simulate them. To avoid this, a volume where the events are considered to be “well contained”. For this the number of photoelectrons generated by each event can be used. When the photons produced by a positron interact with the PMTs a photoelectron is generated. Events that are well contained within the PMTs will produce photons that can be detected by a larger number of PMTs than those that are not (the photons of an event generated in the veto region would only be able to interact with one side of the PMT support structure) and so will generate a larger number of photoelectrons. In figure 4.1 there is a large peak for events with a low number of photoelectrons (most events are generated in the veto region) followed by a smaller bump of events that have a larger number of photoelectrons. These events corresponding to the latter bump are the well contained events.

A cut is then made taking all events with a number of photoelectrons larger than 14. The volume to be defined should match the geometry of the detector(cylindrical) and so the cut is made in the radial direction(figure 4.2) and z axis(figure 4.3). The exact values this gives us for the well contained volume are taken to be $r = 672$ cm and $z = \pm 669$ cm. Though there are a small number of events that remain outside these values they are clearly separate from the bulk of events and so are not considered when deciding the dimensions of the well contained volume.

4.2.2 Fill Shell Method

With the detector fill simulation approach, the well contained volume cuts drops $\sim 70\%$ of the simulated events. This is clearly inefficient and so another method for simulating the events is chosen.

The simplest solution that could be implemented in RATPAC was defining a suitable shell and simulating the 4 MeV positrons randomly throughout this shell as before [31].

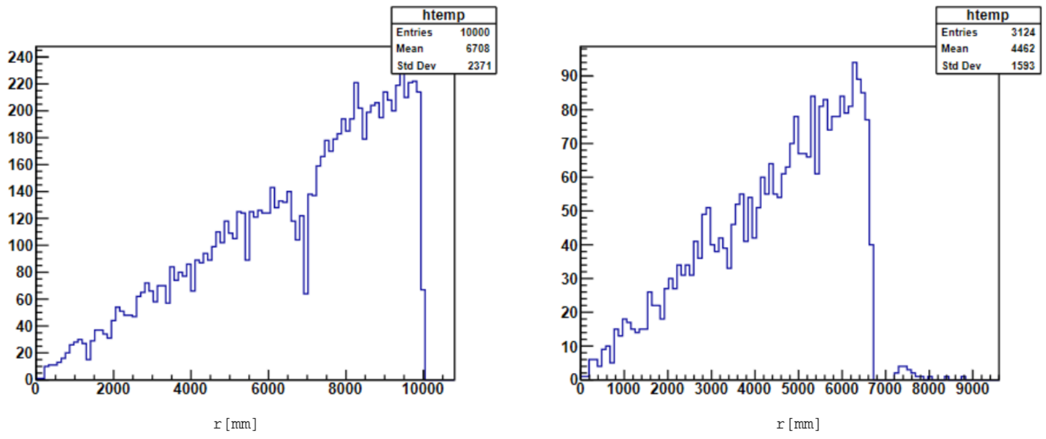


Figure 4.2. Radial event distribution before (left) and after (right) cut on events with more than 14 photoelectrons.

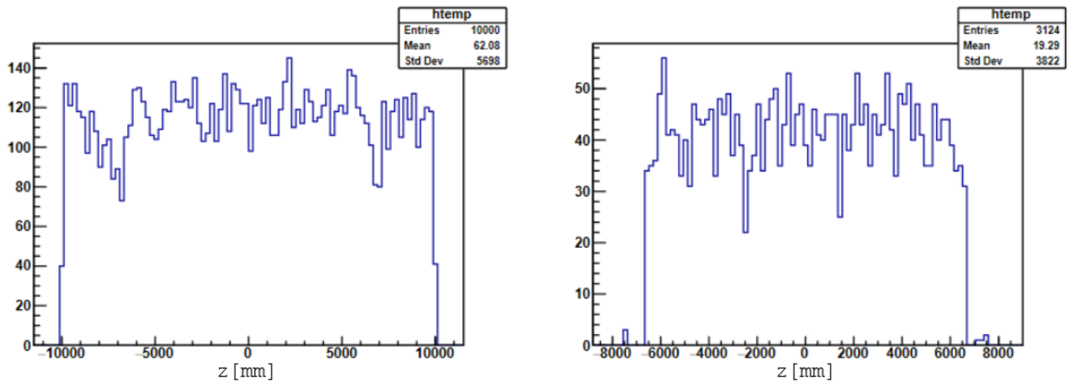


Figure 4.3. Event distribution along z axis before (left) and after (right) cut on events with more than 14 photoelectrons.

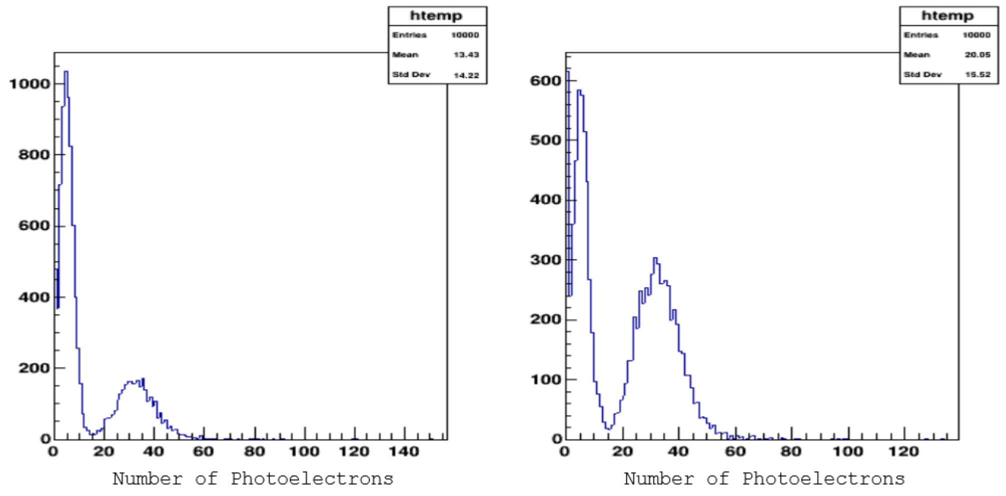


Figure 4.4. Number of photoelectrons with detector fill method (left) and with shell fill method (right).

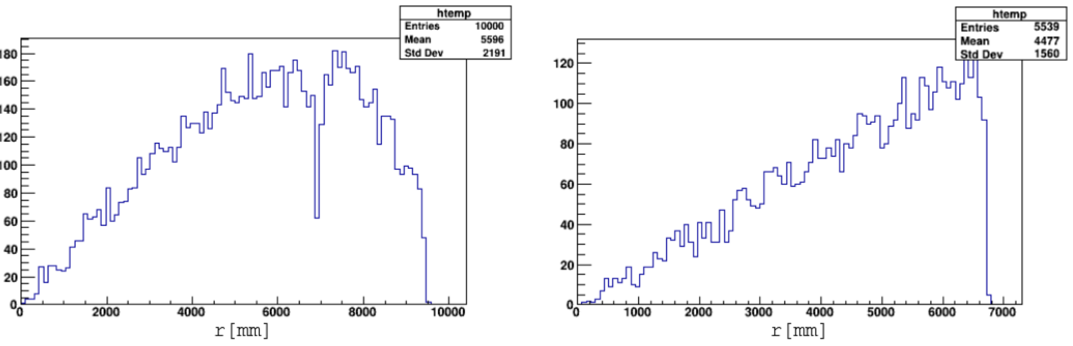


Figure 4.5. Radial event distribution before (left) and after (right) the well contained volume cut is performed.

It should be noted that in this case this shell is essentially a sphere as the inner radius is set to zero. the outer radius of the shell is chosen so as to minimise the number of events dropped when the well contained volume cut is performed. The radius for the smallest possible shell is found simply using

$$R_{shell} = \sqrt{r_{cut}^2 + z_{cut}^2} = \sqrt{672^2 + 669^2} \approx 948 \text{ cm} \tag{4.1}$$

The increase in the number of events that are well contained by the PMTs as a result of this simulation method can be seen by looking at the significant increase in the number of photoelectrons in the latter bump, as seen in figure 4.4.

A significantly smaller portion of the events are lost now when the well contained volume cut is performed. This improvement can be seen in figures 4.5 and 4.6 where the number of events decreases from 10000 to 5539, a decrease of $\sim 45\%$, a significant improvement on the $\sim 70\%$ drop seen with the detector fill simulation method.

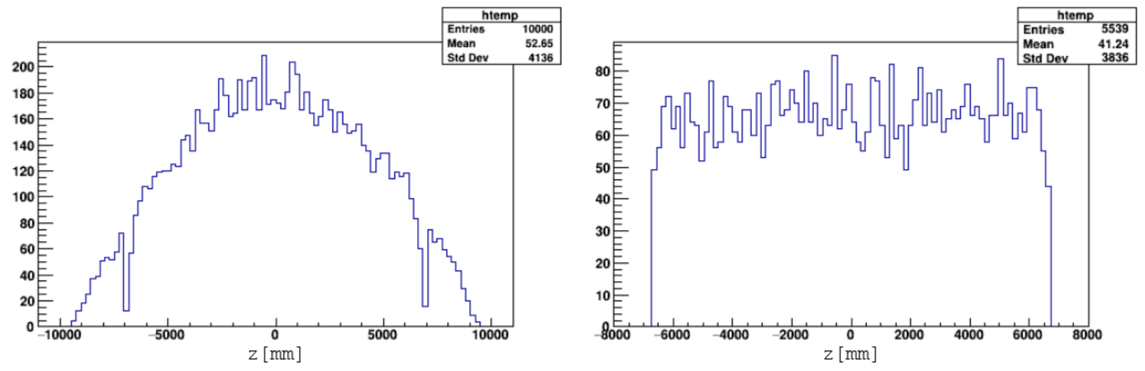


Figure 4.6. Z axis event distribution before (left) and after (right) the well contained volume cut is performed.

4.3 Generating Training and Testing Datasets

When running simulations it was found that RATPAC would occasionally crash. This was an issue for generating a large training dataset. To this end, simulations were performed in batches of 2000 events before being combined into a larger dataset. 70 runs of 4 MeV positrons using the fill shell method were performed, for a total of 140000 events. After the well contained volume cut is performed this gives a total of 76765 events in the training dataset.

Next, testing datasets were produced. These followed the same procedure as generating the training dataset. 5 runs at each energy (3,4,5,6 and 8 MeV) were performed, producing sets of 10000 events each. The number of events left in each training dataset after the well contained volume cut varies but approximately 55% of the events are left in each.

Chapter 5

Vertex Reconstruction

The process of vent reconstruction begins first with the application of the "traditional" algorithm. The output of this algorithm produces good approximations of the interaction vertex. These approximations are then used as the input data for the neural network.

5.1 Vertexing Algorithm - Quad-Fitter

The Quad-Fitter algorithm used here proceeds as follows. Iterating over the dataset of events, this algorithm takes a random sample of four PMT hits(position and timing information $\vec{r}_i$ and t_i) from an event and calculates their residuals R_i as follows

$$R_i = |\vec{r}_i - \vec{r}_{fit}| - v(t_i - t_{fit}) = 0 \tag{5.1}$$

These equations are then solved to find an $\vec{r}_{fit}$ and t_{fit} for these hits, and this is termed a *quad point* [20]. Though the original algorithm in the reference has further steps, in this project we will stop after this step, and take this as an estimate of the reconstructed vertex. In the quad-fitter code the speed of light in water is taken to be $c_{water} = \frac{c_{vacuum}}{n_{water}}$ with $c_{vacuum} = 29.9792458$ cm/ns and $n_{water} = 1.34$.

Two versions of this algorithm are studied. The first takes every combination of hits to form a quad, referred to as the repeat digits Quad-Fitter. The second version only uses a hit once before removing the hit from the set of available hits for forming quads. This is referred to as the digits once Quad-Fitter.

5.2 Tensorflow Keras Library

On the Tensorflow web page, Tensorflow is described as "an end-to-end open source platform for machine learning" [32]. Tensorflow allows for the use of state of the art machine learning in an approachable way.

The Keras library is "is a deep learning API written in Python, running on top of the machine learning platform TensorFlow" [33]. It is this that is used to construct the neural network for this project. Keras is a particularly well suited library for investigating different neural network as it is very simple to change the structure of neural networks.

Functions can be easily written that allow the user to specify the number of layers, the type of layers, and the number of nodes in each layer, to name a few of the characteristics of the network that are easily altered. This functionality is useful for exploring the structure of the neural network that leads to the best performance.

5.3 Pandas Library and Jupyter Notebooks

The analysis of the output of the neural network is primarily handled using the pandas python library. This library implements a dataframe object that helps with easy manipulation and interpretation of data and is highly optimised for performance with large datasets [34].

All python code, including the neural network and the analysis of the networks performance, is written using jupyter notebooks. The Jupyter notebooks app is described as "a server-client application that allows editing and running notebook documents via a web browser" where notebooks are documents used in the app that contain both python code and rich text elements [35].

5.4 Reconstruction Process

5.4.1 Application of Quad-Fitter

Hit data is first passed through the quad-fitter script, written in C, to produce the quad points that will be the input data for the network. This script also implements the well contained volume cut as the script takes as inputs the radial and z axis cut values in mm. When producing the quads for an event, the script first checks whether the event position is within the cut values. If not, the script will not calculate the quads and moves onto the next event. The final output is then a csv file containing the quads for each event and the corresponding true vertex for that event.

5.4.2 Application of Neural Network

The quad points go through necessary preprocessing before being passed to the neural network. First, the dataset is split into features (the input data) and labels (true vertex position). Next, the features are scaled using a min-max scaler, ensuring all features values lie between 0 and 1 [36]. This is necessary as neural networks can sometimes struggle with unscaled values. After this the scaled features and the label data are passed to the network for training or testing. When testing, the position of the vertices output by the neural network are are subtracted from the true interaction vertices. In each case, the the value of $R = \sqrt{x^2 + y^2 + z^2}$ is taken and so this subtraction produces a ΔR distribution. A well performing network has a ΔR distribution centred (mean) on zero and a low standard deviation.

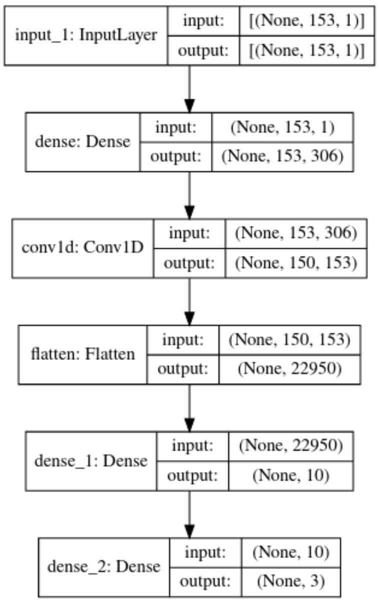


Figure 5.1. Architecture of most successful network.

5.4.3 FRED

The standard reconstruction program currently used by the WATCHMAN collaboration is FRED, which is the standalone implementation of a water cherenkov reconstruction algorithm BONSAI for WATCHMAN [37]. Events are therefore also reconstructed using FRED, which is used as a benchmark against which the performance of the quad-fitter and neural network reconstruction process is judged.

5.5 Neural Network Structure

The structure of the neural network described here (shown in figure 5.1) is the structure that was found to be the best performing model.

The input layer has the same width as 3 times the number of quads of the event with the most hits. In this network the time coordinate of the quad is actually omitted (see discussion). For a network using the time coordinates the width would therefore be 4 times the same quantity. The input layer has a depth of 1, necessary for the later application of a convolutional layer.

The first layer in the network is a dense layer with a number of nodes double the size of the input dimension. It was found through experimentation that increasing the number of nodes after the input layer improved performance.

The following layer is a 1D convolutional layer with the number of filters the same as the input dimension. This is to begin reducing the dimensionality of the network. This conolutional layer has a kernel size of four. After this, it is necessary to flatten the output of the convolutional layer.

The final layer before the output layer is another dense layer with 10 nodes, which aims to significantly reduce the dimensionality before the output (dense) layer with 3 nodes, one node for each coordinate (x,y,z).

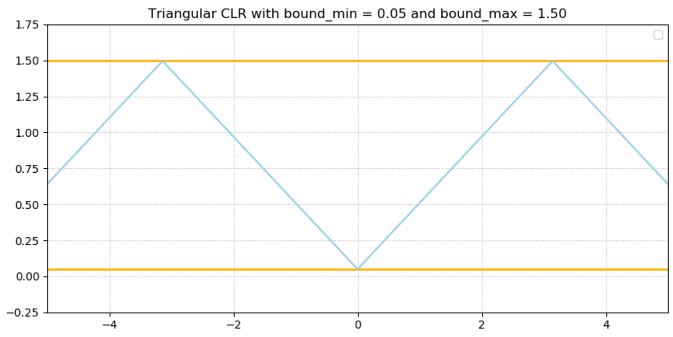


Figure 5.2. Example of triangular window learning rate policy with upper bound of 1.5 and lower bound of 0.05. Taken from [7]

Each layer in the network uses the ReLU activation function except the final output layer, where a linear activation function (function that outputs the input) is necessary in order to predict the negative valued coordinates.

It should be noted that the performance of this network with input data from the repeating quad-fitter far exceeds the digits once quad-fitter (see appendix A.2). As a result, the repeating quad-fitter was used almost exclusively after a point in the project and so unless specified otherwise the reader should assume quad points have been produced by the repeating digits script.

5.6 Neural Network Training

The optimiser used in the training of the neural network is the Keras Adamax optimiser [38]. The Adam optimiser is "an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments". Adamax is a variation of the Adam optimiser algorithm based on "the infinity norm" [39]. This optimiser was chosen after testing a selection of other Keras optimisers and it was found to have the best performance.

Early on in the project, it was found that the network would struggle to avoid local minima or saddle points. To combat this a cyclical learning rate (CLR) is used, which allows the learning rate to vary between boundary values. Increasing the learning rate can have a short term negative effect on the performance of the network, but will allow the network to escape a local minimum, and so will result in a long term improvement in performance. Varying the learning rate of the network in this way has been shown to improve accuracy of neural networks without the need for tuning and can reduce the number of iterations required [40]. The simplest method that achieves this is the triangular window learning rate policy (shown in figure 5.2, where learning rate increases and decreases linearly within the defined bounds. This is implemented as a Keras callback and so is a simple matter to add to an existing network [41].

Appropriate upper and lower bounds for the CLR are found by using a learning rate finder, implemented again as another Keras callback [42]. The neural network is trained over a small number of epochs and a range of learning rates are scanned. For the network presented in section 5.5 a lower bound of 3.18×10^{-5} and an upper bound of 4.90×10^{-3} are used.

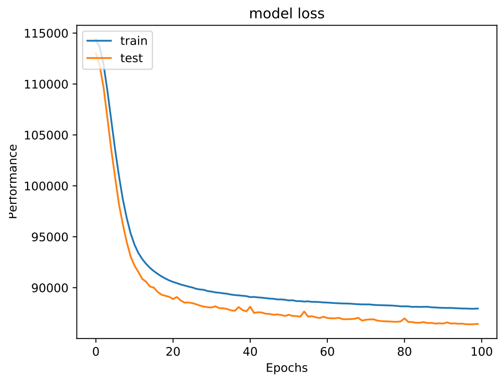


Figure 5.3. Loss plot of a network without CLR.

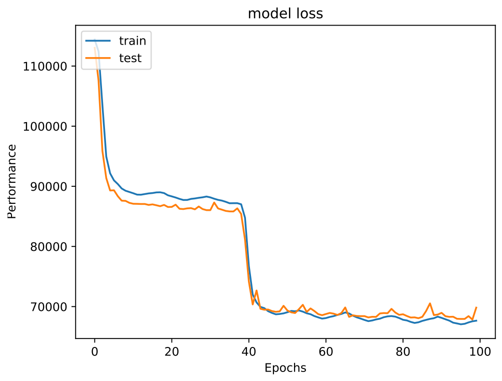


Figure 5.4. Loss plot of a network with CLR.

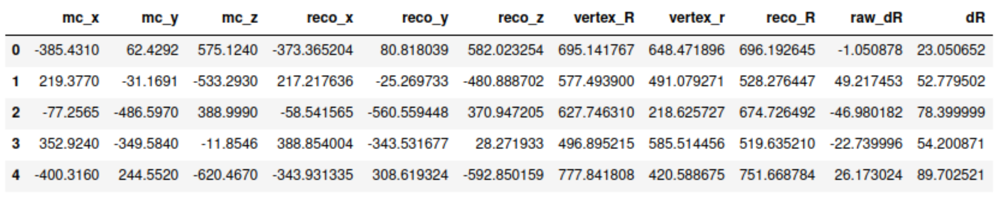
It is also necessary to choose an appropriate stepsize for the CLR, which is defined in [40] as "the number of iterations in half a cycle", or the number of iterations used for the learning rate to change from one bound to the other. It is recommended that this is set to 2-10 times the number of iterations in an epoch. In practice, the number of iterations is the length of the training set divided by the batch size. The batch size used here is 2, with the training data set having a length of 76765. Taking 8 times $\frac{76765}{2}$ is found to produce the best performance.

The benefits of the use of the CLR is shown in figures 5.3 and 5.4 which show the loss plots of an early version of the network. The network is trained without and then with the CLR for the same number of epochs. When using the CLR, the network escapes a local minimum after 40 epochs, whereas when trained without it it never escapes. The CLR is clearly useful for escaping these local minima, and so the use of the CLR is retained for all subsequent versions of the network.

In the final version of the network presented in section 5.5 the network is trained for 25 epochs as training any longer was not found to improve performance significantly.

5.7 Resolution Calculations

An important metric used to compare the performance of the network with FRED is the resolution. The definition of resolution used by the WATCHMAN collaboration is the range within which the lowest 68% of values lie when the absolute value of the ΔR distribution is plotted. An example of this type of plot can be seen in figure 5.5.



	mc_x	mc_y	mc_z	reco_x	reco_y	reco_z	vertex_R	vertex_r	reco_R	raw_dR	dR
0	-385.4310	62.4292	575.1240	-373.365204	80.818039	582.023254	695.141767	648.471896	696.192645	-1.050878	23.050652
1	219.3770	-31.1691	-533.2930	217.217636	-25.269733	-480.888702	577.493900	491.079271	528.276447	49.217453	52.779502
2	-77.2565	-486.5970	388.9990	-58.541565	-560.559448	370.947205	627.746310	218.625727	674.726492	-46.980182	78.399999
3	352.9240	-349.5840	-11.8546	388.854004	-343.531677	28.271933	496.895215	585.514456	519.635210	-22.739996	54.200871
4	-400.3160	244.5520	-620.4670	-343.931335	308.619324	-592.850159	777.841808	420.588675	751.668784	26.173024	89.702521

Figure 5.6. The head of the dataframe of the network output for the 6 MeV positron data.

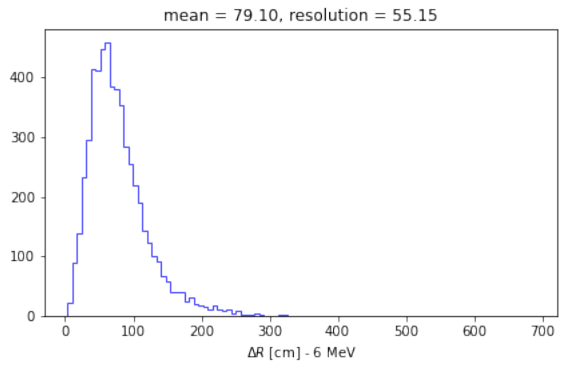


Figure 5.5. Example of absolute value ΔR distribution plot for network output with 6 MeV positrons.

As a result of the different analysis tools used for the output of the network and FRED (pandas library in python and ROOT respectively) this calculation proceeds in a different manner in each case.

5.7.1 Resolution Calculation for Network with Pandas

The dataframe object used in pandas makes the calculation of the resolution very straightforward. First, the head of the relevant dataframe is printed (shown in figure 5.6) using

```
df_6.head()
```

The eleventh column can then be identified as the column of interest (here dR refers to the absolute value of the difference between the true and reconstructed vertices here). The following function is then called to calculate the resolution, which takes the relevant dataframe and column as inputs.

```
def res(dataFrame, col):
    one_std = round(dataFrame.shape[0] * 0.68)
    df_sorted = dataFrame.sort_values(by=['dR'])
    std_cutoff = 0
    count = 0
    for i in df_sorted.itertuples():
        if count == one_std:
            std_cutoff = i[col]
        count +=1
        return std_cutoff
```

This function first calculates the number of events equivalent to 68% the total number of events by multiplying the length of the dataframe (`dataFrame.shape[0]`) by 0.68 and rounding this to the nearest whole number. The dataframe is then sorted by dR in ascending order. The rows of the dataframe are then looped over, incrementing a count variable for each row. An if statement checks whether the count is equal to 68% of the events. If this is satisfied it stores the dR value of that row. This value is then returned and is the value taken as the resolution of the network for that energy.

5.7.2 Resolution Calculation for FRED with ROOT

The output of FRED is in the form of a ROOT file and so finding the resolution is slightly more involved than with the network output. First, the number of events equivalent to 68% of the total number of events is calculated as before. The distribution is then plotted as shown in figure 5.7. This is done using

```
data4->Draw("sqrt((mcx-x)*(mcx-x) + (mcy-y)*(mcy-y) + (mcz-z)*(mcz-z))
>> dRPos4(1600, 0., 12800.)", "sqrt((mcx*mcx)+(mcy*mcy)) < 6720 &&
abs(mcz) < 6690 && good_pos > 0.01")
```

This draws the histogram using the variables stored in the tree `data4` and stores it in the histogram `dRPos4` which is set to have 1600 bins over the range 0.0 mm to 12600.0 mm. It is important that the bin width must be set suitably small for this method ot be implemented correctly. It also performs the well contained volume cuts and a cut on the `good_pos` variable, which is discussed in section 6.2. To calculate the resolution, the following code is run.

```
int ibin = 0;
double val = 0;
double res = 2932;
while(val < res){
      val = dRPos4->Integral(0, ibin);
      ++ibin;
};
dRPos4->GetBinCenter(ibin);
```

The `double res` variable is set to the value of 68% of the events. The while loop checks if the `double val` variable is less than this. If this is true, `val` is set equal to the integral of the histogram from zero to the bin denoted by `int ibin`, which is incremented each loop. In this way, the loop proceeds until the bin that returns the integral with a value greater than `res` is reached. Then `dRPos4->GetBinCenter(ibin)` can be called which returns the value for the centre of the bin. This is value is then taken as the resolution value for FRED at this energy.

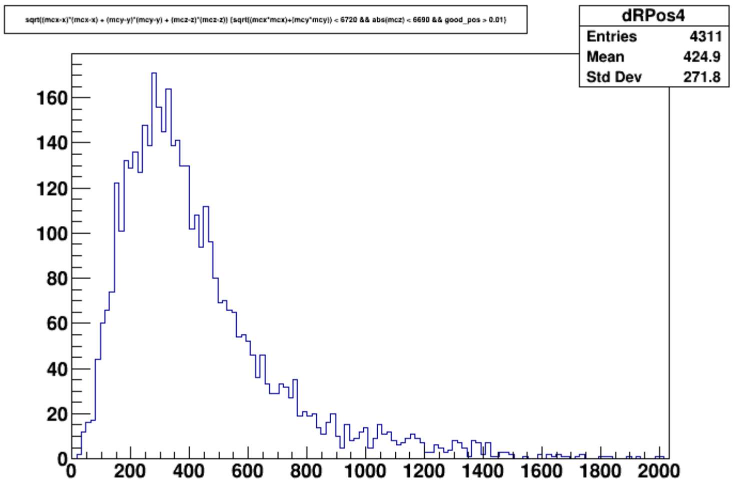


Figure 5.7. Example of an absolute valued ΔR distribution (mm) for the 4 MeV positron as reconstructed by FRED.

Chapter 6

Results and Analysis

6.1 Network Performance with and without Quad Point Time

An interesting and counter-intuitive finding from the project was the disimprovement of the network performance when the predicted interaction times in the quad points were included. Though in both cases the loss function achieves similar values, when quad point time values are included in the input features the ΔR distribution becomes heavily skewed with a large negative mean (-49.85 cm), and a large standard deviation (59.17 cm), which can be seen in figure 6.1.

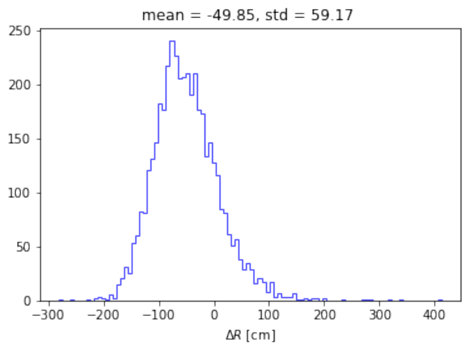


Figure 6.1. ΔR distribution of network with quad point time for 4 MeV positron.

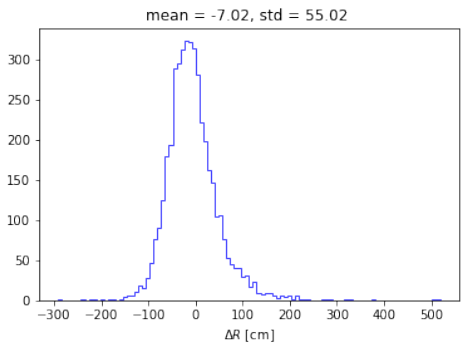


Figure 6.2. ΔR distribution of network without quad point time for 4 MeV positron.

This issue was explored and a number of different approaches were tried to find a solution to this but all were unsuccessful (see appendix section B). When quad point time values are not given to the network as input features, the ΔR distribution produced is in much better agreement with the "good performance" standard of a mean near zero (-7.02 cm) and a lower standard deviation (55.02 cm), which can be seen in figure 6.2.

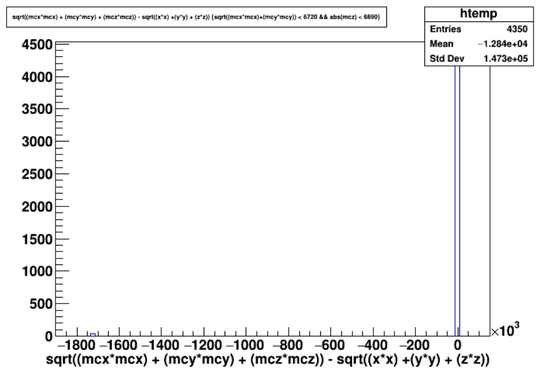


Figure 6.3. ΔR distribution (mm) of 4 MeV positron vertices reconstructed by FRED before good_pos cut has been applied.

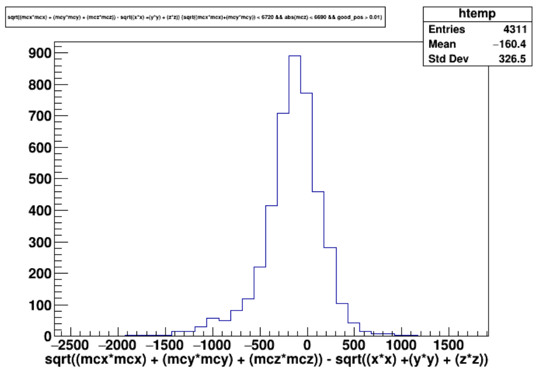


Figure 6.4. ΔR distribution (mm) of 4 MeV positron vertices reconstructed by FRED after good_pos cut has been applied.

Table 6.2. Mean values of reconstructed ΔR distributions at each energy from the network and FRED

Energy(MeV)	Network Mean (cm)	FRED Mean(cm)
3	-2.62	-16.95
4	-7.02	-16.04
5	-5.90	-15.42
6	-5.91	-15.27
8	-3.17	-15.44

Table 6.1. Mean and standard deviation values of ΔR distributions for the network with and without quad point time in the input features.

Input Features	Mean (cm)	Standard Deviation (cm)
With Time	-49.85	59.17
Without Time	-7.02	55.02

6.2 Network Performance vs FRED

The events used in the distribution in figure 6.4 and in the data from FRED generally are cut using the same well contained volume cuts as previously defined. There is also a third cut on the `good_pos` variable, which is a measure of how well FRED believes the vertex has been reconstructed. A cut of `good_pos` $>$ `0.01` is applied which is the smallest cut that removes the large outliers (see figure 6.3). Only a small number of outliers are present and so this cut only removes 39 events in the distribution shown. A similarly small number of outliers are removed at other energy levels.

The network performance is compared to FRED data cut in this way. First the ΔR distributions of the network and FRED are compared in terms of their mean values and standard deviation. The mean and standard deviation values of both the network and FRED are presented for each energy in tables 6.1 and 6.3 respectively.

The resolution values as is defined in section 5.7 are next compared. The resolution values

Table 6.3. Standard deviation values of reconstructed ΔR distributions at each energy from the network and FRED

Energy(MeV)	Network Standard Deviation (cm)	FRED Standard Deviation(cm)
3	61.36	47.14
4	55.02	32.65
5	51.53	29.68
6	53.23	28.10
8	51.00	26.06

Table 6.4. Resolution values of reconstructed ΔR distributions at each energy from the network and FRED

Energy(MeV)	Network Resolution(cm)	FRED Resolution(cm)
3	95.72	47.6
4	92.06	58.0
5	90.54	43.6
6	87.65	40.4
8	88.17	38.0

are presented in table 6.4. The average resolution value for the network is 90.83 cm while the average for FRED is 45.52 cm.

6.3 Network Performance with Larger Event Dataset

The size of the event dataset was doubled when the poor performance of the network relative to FRED was seen. The ΔR distribution of the network trained on this dataset for 4 MeV positrons is shown in figure 6.5. There is little improvement in terms of mean and standard deviation when compared with the distribution in figure 6.2.

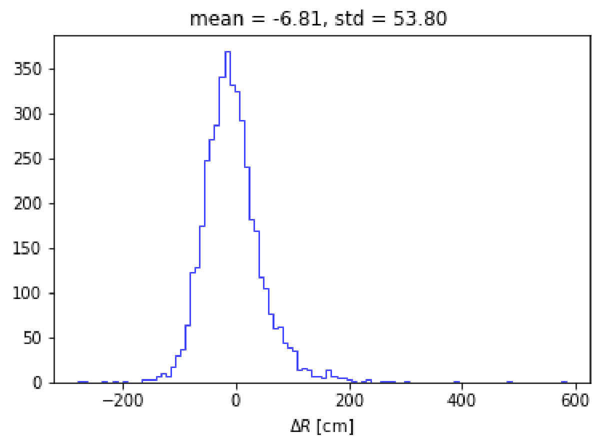


Figure 6.5. Network ΔR distribution for 4 MeV positrons trained on 140000 event dataset.

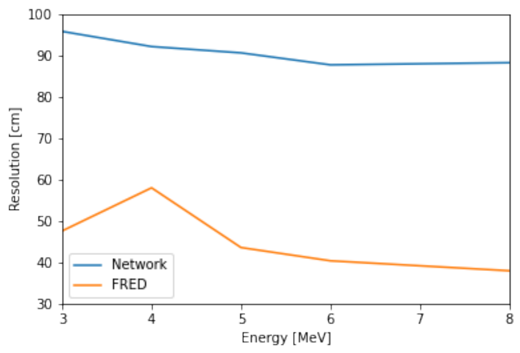


Figure 6.6. Resolution(cm) vs Energy(MeV) for the network and FRED

6.4 Discussion and Analysis

The aim of this project was to apply combine the use of the Quadfitter algorithm and a neural network for the purpose of reconstructing well interaction vertices in the WATCHMAN detector. In this section the results presented in the previous section are discussed in tandem with the presentation of any analysis that was performed, and the extent to which the project achieved its aims are outlined.

6.4.1 Discussion of Network Performance in Comparison with FRED

The network output ΔR distributions have mean values significantly nearer zero than the corresponding distributions of FRED, but the resolution of the absolute value ΔR distributions for the network is poorer the FRED. Taking the ratio of the average resolution values from section 6.2 we see that the resolution of the network is on average 2.00 times worse than FRED.

The difference mean values of the ΔR distributions are interesting to discuss in the context of the decision to not include quad point time values in the input features for the network due to the large negative mean. Though the mean value of the distribution in figure 6.1 is by far the largest negatively skewed of any of the distributions, a negative skew is seen in the ΔR distribution of every reconstruction method. The large negative skew of the ΔR distribution when using quad point time was explained as the network getting stuck in a local minimum of some sort during training [30], but the presence of the negative skew in all distributions could suggest another source, or at least the local minimum was not the only contributing factor. Unfortunately, due to a lack of knowledge on the reconstruction process used by FRED, it is hard to identify what this common source could be.

The most relevant metric when comparing the network to FRED is the resolution values of the absolute value ΔR distributions. These values, plotted in figure 6.6, show the resolution of the network displays the decreasing resolution as a function of energy that would have been expected and is also present in the plot of FRED's resolution (bar 4 MeV which appears to be an outlier) [30]. The poor performance of the network has a number of potential explanations.

The absence of timing information from the quad points could potentially lead to the network being unable to achieve the better resolution from spatial information alone as this may be limited by the geometry of the PMTs. This would put a hard limit on the performance of the network. However, the time of the PMT hit is used to calculate the quad in the quad fitter, and so the timing information is in some sense accounted for in the spatial part of the quad point.

The network structure itself may just be too simple to learn enough of the complex behaviour of the data necessary to achieve higher resolutions. The very limited increase in performance seen when increasing the size of the dataset discussed in section **??** may be supportive of this. To this end, more complicated neural network structures should be investigated.

One area of interest that the network appears to perform better than FRED is in the lack of large outliers. Though these outliers are sufficiently small in number that they do not significantly effect the statistics of FRED's performance (mentioned in 6.2), they are still present. The output of the network does not suffer from this in the same way.

Ultimately, while the neural network reconstruction approach shows promise in some areas, FRED still significantly outperforms the network in terms of resolution.

6.4.2 Spatial Analysis of Resolution Values

It is interesting to see if the resolution of the network output has any spatial dependence. For this, the well contained volume $V = \pi r^2 h = \pi(r_{cut}^2)(2 \times z_{cut})$ is divided into three equal volumes. The corresponding cylindrical radius of each of these volumes is then found, which are used to define the three radial cuts used in the analysis. The radial intervals used for the cuts are shown in table 6.6 along with their corresponding midpoint values used to plot the radial resolution dependence of each energy.

Table 6.5. Resolution values of network output for each energy and radial cut.

Energy(MeV)	**r $\leq$ 388 cm**	**388 cm <r $\leq$ 548 cm**	**548 cm <r**
3	109.59	94.10	89.80
4	104.83	88.79	84.06
5	90.42	90.93	93.21
6	88.74	88.03	89.43
8	88.35	88.01	89.86

Table 6.6. Radial intervals and corresponding midpoint values used for plotting.

Interval (cm)	**Midpoint Value (cm)**
0 - 388	194
388 - 548	468
548 - 672	610

The plot of resolution as a function of position is displayed in figure 6.7. Qualitatively the energies can be divided into two groups: decreasing resolution with increasing radial position (3 and 4 MeV), and increasing resolution with radial position (5, 6 and 8 MeV).

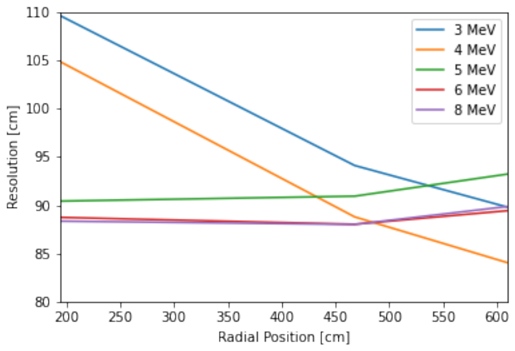


Figure 6.7. Resolution (cm) vs Radial Cut Midpoint (cm) for the network at each energy (MeV).

The manner in which the resolution of the first grouping decreases with corresponding increase in radial position is much more dramatic than the increases in resolution observed in the second grouping. The resolution of the 4 MeV events improves from being the second worst resolution in the innermost cut to the best resolution in the outermost cut. The 3 MeV dependence follows a similar pattern, from its innermost events being the worst resolution wise, to achieving parity with the outermost events of 6 and 8 MeV.

The resolution dependence of the 5, 6 and 8 MeV events appears roughly constant until the outermost events where a small increase in resolution is observed. The plots of 6 and * MEV are almost near identical.

This behaviour suggests the light of the 3 and 4 MeV interactions in the inner detector regions is more strongly scattered by a significant amount than that of the second grouping, leading to a much poorer resolution. However if this was the case, it is strange that there is no intermediate energy at the transition from low to higher energy. The difference in behaviour between the 4 and 5 MeV plots is distinct, with the 5 MeV plot clearly belonging to the second grouping.

The increase in resolution in the outermost events of the second grouping could be explained by lying closer to the PMT support structure. This is known to be an issue that FRED suffers from, and perhaps the reconstruction process here has similar issues [43]. The fact that the resolution plots roughly converge near the same values in this region suggest that energy dependence could lessen closer to the PMT support structure. This does not explain why the 4 MeV resolution is so much better than the other energies in this region however.

Ultimately, the fact that there exists two very clear and distinct groupings in the resolution behaviour as a function of radial event position is suggestive of some underlying mechanism, but it is difficult to provide a satisfactory solution that explains the entirety of the data.

6.4.3 Network Performance on Unseen Energies

It is interesting to note that the network was only ever trained on a dataset of 4 MeV positron events, and yet the network is able to predict reconstruct the interaction vertices of events at different energies to a comparable standard with the 4 MeV positron events

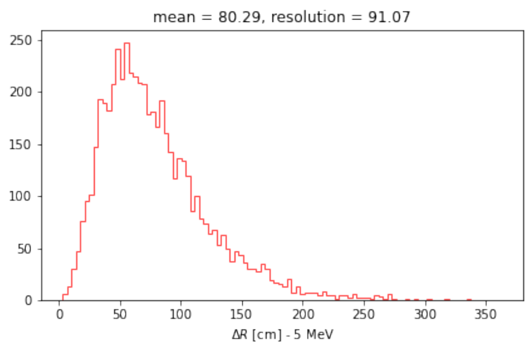


Figure 6.8. Absolute value ΔR distribution of 5 MeV positron vertices reconstructed network.

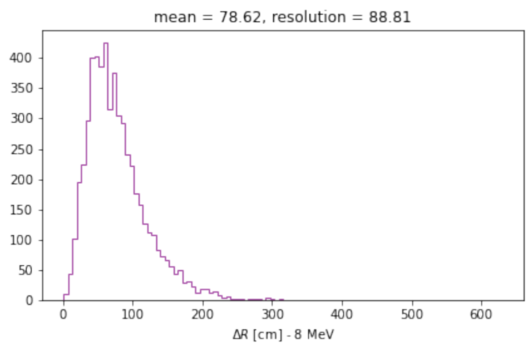


Figure 6.9. Absolute value of ΔR distribution of 8 MeV positron vertices reconstructed network.

the network was trained on. Some examples of this are shown in figures 6.8 and 6.9. Though the resolution achieved is still outperformed by FRED, this ability to extrapolate to unseen data is very important if a more advanced network was to be applied to vertex reconstruction in the future.

Chapter 7

Conclusions

The quad-fitter and neural network approach studied in this project ultimately fails to perform as well as the benchmark FRED reconstruction algorithm used in WATCHMAN. Though the difference in resolution is significant, a factor of 2.00 increase in performance is not out of the realm of possibility for a more complex neural network architecture. The neural network is able to make reasonable predictions for interaction vertices and extrapolates to unseen energy ranges with a comparable performance to training energy with no issues. The extrapolation to unseen energies is a particularly important feature if the network was ever to be deployed in a detector, as the energy of the events may be unknown in a real experimental setting, and it may not be feasible to train a network at all potential energies. A particular advantage that the network would have over FRED if the resolution could be improved to comparable levels would be the lack of very large outliers needing to be cut from the final results.

The performance of the network benefits little from increases in size of the training dataset, and, taken in combination with the networks struggle to utilise the timing data of the quad points, this may suggest the need for a more complex network design.

Two more complex network architectures that were considered for implementation in this project and could possibly be used as the basis for further work were *recurrent* neural networks and the *DeepSets* architecture.

Recurrent neural networks are neural networks where feedback connections between layers are allowed. In particular the Long Short Term Memory (LSTM) variation of the recurrent neural network structure was investigated. This is an architecture that allows the network to evaluate whether or not it should "remember" or "forget" previous predictions. The remembered predictions can then be used to make further predictions. The quad-fitter algorithm used in this project forms all possible combinations of quads. Inevitably, a fraction of these reconstructed quads are not good estimates of the true vertex. A LSTM network could potentially learn to remember good quads, and forget bad quads. In this way it may be able to improve the quality of the reconstructed interaction vertex by only using these good quads [44].

The DeepSets architecture results from treating the input features to the network as a set or collection of objects, and then looking for a network structure that is invariant to the ordering of objects in the set. The deepsets network derived from this was shown to outperform some more traditional neural networks on number of tasks [2]. As the input features used in this project, the quad points, should be invariant with respect to their ordering, it is thought that a network of this type would be well suited to the problem of vertex reconstruction. An attempt was made during the course of this project to design a suitable network using this deepsets architecture but there was unfortunately not enough time to make this network functional.

The issue of why the network performance disimproves when quad point time values are included in input features remains unresolved. This issue is important as if the spatial components of the quad points do not fully account for the timing information this puts a limit on the maximum resolution that can be achieved, as without time information the resolution is limited by the geometry of the PMTs [30]. Any future work on interaction vertex reconstruction with more complex networks should therefore attempt to discern what the cause of this issue is, and whether or not time values from the quad points can be omitted form the input features without limiting the performance of the network.

On the whole, though the interaction vertex reconstruction method explored in this project is not currently competitive with the benchmark FRED algorithm, the performance of the network can be considered a proof-of-concept for the application of neural networks to the problem of interaction vertex reconstruction and should serve as motivation to investigate the application of more complex networks to the problem.

Appendix A

Development of Neural Network

This appendix seeks to highlight the improvement in network performance brought about by various additions to the network. All the networks in this section are using the testing dataset of 4 MeV positron events. It should be noted that the plots presented here have mean and standard deviation values not resolution values as when these plots were produced the definition of resolution used by the WATCHMAN collaboration was unknown.

A.1 General Network Improvement

Here some general improvements to the network are presented to demonstrate how the performance of the network has greatly improved over the course of the project.

The first example in figure A.1 is from a very early version of the network using only dense layers with ReLu activation, no CLR and still including quad point time data. The training data set was also only around 10000 events at this point.

The next plot presented in figure A.2 is trained on 70000 event dataset and uses a CLR.

The plot shown in figure A.3 is the output from a network trained on the 70000 event dataset with a CLR, and using a linear activation function in the output layer. Quad point time is also no longer included in the input features.

Figure A.4 shows the effect the introduction of the convolutional layer to the network had on the performance of the network of figure A.3.

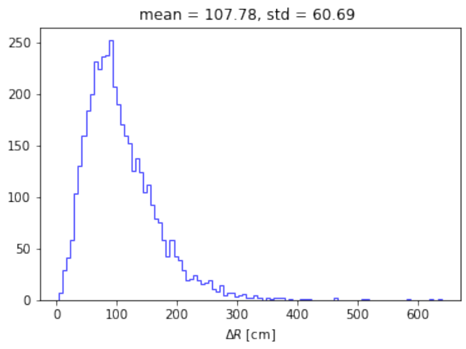


Figure A.3. Absolute value ΔR distribution highlighting the effect of the introduction of the linear activation function in the output layer.

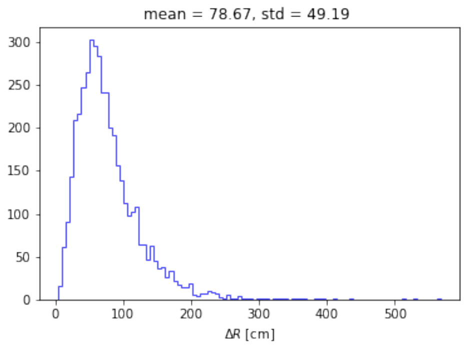


Figure A.4. Absolute value ΔR distribution with the 1D convolutional layer introduced between the dense layers.

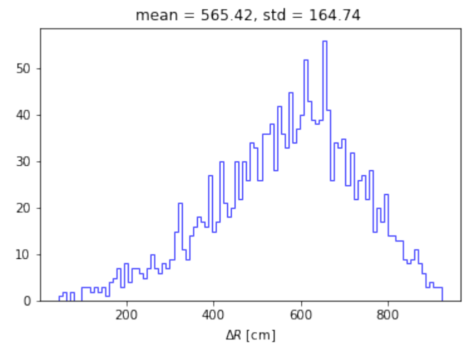


Figure A.1. Early absolute value ΔR distribution.

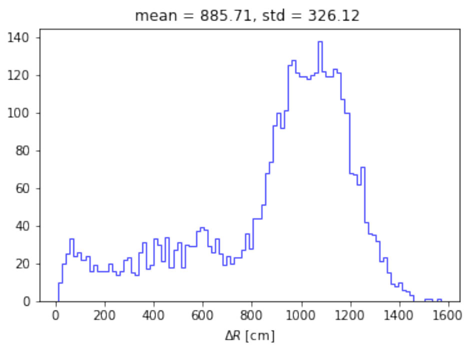


Figure A.2. Absolute value ΔR distribution of network trained on 70000 event dataset using CLR.

A.2 Repeat Digits versus Digits Once

In this section plots of the network output using data from the repeating digits quadfitter and the digits once quadfitter are compared to highlight how poorly the network performs with only the digits once data. The plot for the digits once is shown in figure A.5 and for the repeat digits in figure A.6.

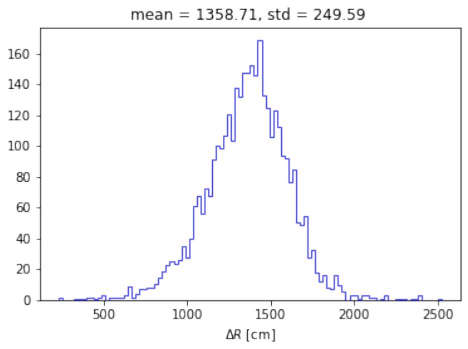


Figure A.5. Absolute value ΔR (cm) distribution using input data from the digits once quad-fitter.

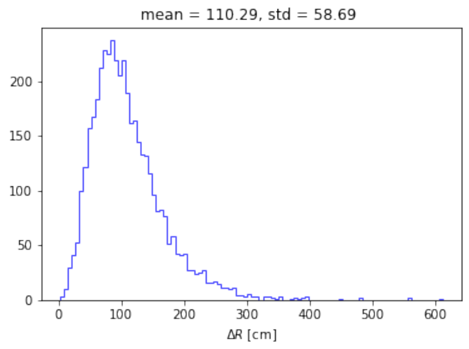


Figure A.6. Absolute value ΔR (cm) distribution using input data from the repeat digits quad-fitter.

Appendix B

Attempts to Improve Performance of Network with Quad Point Time

In this appendix some of the attempted solutions to the problem of disimproving network performance with quad point time are discussed.

It is first important to understand the structure of the csv file output by the QuadFitter script. Each row of the csv file corresponds to an event. The quads fill the row one by one, and so moving left to right you encounter the x, y, z, and t components of the first quad, in that order. This repeats for each quad produced for that event.

The first method trialed was scaling the position and time of the quad components separately. It was thought that because these are in a way different types of data, that scaling them together could lead to issues for the network. To implement this, a function for extracting time from the dataset was written.

```
def timeExtract(in_df):

    pos_df = in_df

    height = in_df.shape[0]
    width = in_df.shape[1]
    pos_width = (width/4) * 3
    time_width = (width/4)

    index = np.arange(0, height, dtype=int)
    pos_cols = np.arange(0, pos_width, dtype=int)
    time_cols = np.arange(0, time_width, dtype=int)

    time_df = pd.DataFrame(index=index, columns=time_cols)
    time_df = time_df.fillna(0)
    count = 0
    col_count = 0

    for col in in_df:
        count += 1
        if count%4 == 0:
            time_df[col_count] = in_df[count+2]
            col_count +=1
            pos_df = pos_df.drop(count+2, axis = 1)
    return time_df, pos_df
```

This function takes the dataframe the csv file is loaded into as an input, scans across the columns in the dataframe, and takes every fourth column and adds it too a new dataframe. Separate quad point position and time dataframes are then returned. These can then be scaled separately. To input the scaled data into the network, it was necessary to recombine these dataframes. This was done with the following function.

```
    def recombine(posArray, timeArray):
        if posArray.shape[0] != timeArray.shape[0]:
            print("Fail")
            return 0
        height = timeArray.shape[0]
        width = posArray.shape[1] + timeArray.shape[1]
        newArray = np.zeros((height,width))

        timeCols = []
        for i in range(width):
            if i%4==0:
                timeCols.append(i)
        posCols = []
        for i in range(width):
            if i not in timeCols:
                posCols.append(i)
        posCount = 0
        timeCount = 0
        for i in range(width):

            if i in posCols:
                newArray[:, i] = posArray[:, posCount]
                posCount += 1
            elif i in timeCols:
                newArray[:,i] = timeArray[:, timeCount]
                timeCount += 1

        return newArray
```

The dataframes are converted to arrays for scaling. This function takes these arrays as its inputs and recombines them in the same column structure as the original csv file. The outputted array is then used as an input to the network.

This method saw no improvement in performance when compared with scaling both position and time together, eliminating this as the issue.

It was thought that negative quad point time values, arising from the negative times recorded by the PMT, could be interfering with the network. In retrospect this could not be the case as the scaling process re-scales negative values to positive values or zero. Nevertheless, the quad point times were extracted and the value of the lowest time value was added to all values so as to ensure that all values were zero or positive. This did not produce any improvement.

Another approach was introducing a fourth output node to the network (for event time) and adding truth values of zero for time values (all events are generated at $t = 0$). It was thought that by having to predict the time value the network would learn to integrate the time data better, but no improvement was seen.

The final approach was to introduce another convolutional layer as the first layer in the network. The filter size was set to 4 and the number of filters was set to equal the number of quads in an event. It was hoped that this would integrate the time and spatial components of the quad point better but this saw no improvement in performance either.

Bibliography

[1] K. P. Murphy, *Machine learning : a probabilistic perspective.* MIT Press, Cambridge, Mass. [u.a.], 2013.

[2] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov and A. J. Smola, *Deep sets*, *CoRR* **abs/1703.06114** (2017) [1703.06114].

[3] P. Fernandez, *Neutrino Physics in Present and Future Kamioka Water-Cherenkov Detectors with Neutron Tagging*, Ph.D. thesis, 02, 2017. 10.13140/RG.2.2.19649.56169.

[4] J. Brownlee, *A gentle introduction to the rectified linear unit (relu)*, 2020.

[5] M. Askins, M. Bergevin, A. Bernstein, S. Dazeley, S. T. Dye, T. Handler et al., *The physics and nuclear nonproliferation goals of watchman: A water cherenkov monitor for antineutrinos*, 2015.

[6] G. F. Knoll, *Radiation Detection and Measurement.* John Wiley and Sons Inc., New York/Chichester/Weinheim/ Brisbane/Toronto/Singapore, 2000.

[7] C. Versloot, "Training your neural network wiht cyclical learning rates." https://www.machinecurve.com/index.php/2020/02/25/training-your-neural-network-with-cyclical-learning-rates/.

[8] V. Masson-Delmotte, P. Zhai, A. Pirani, S. L. Connors, C. Péan, S. Berger et al., *Ipcc, 2021: Summary for policymakers. in: Climate change 2021: The physical science basis. contribution of working group i to the sixth assessment report of the intergovernmental panel on climate change*, 2021.

[9] International Energy Agency, *World energy outlook 2020*, 2020.

[10] Office of Nuclear Energy, "Advantages and challenges of nuclear energy." https://www.energy.gov/ne/articles/advantages-and-challenges-nuclear-energy.

[11] International Energy Agency, *World energy outlook 2019*, 2019.

[12] E. De Sanctis, S. Monti and M. Ripani, *Nuclear Reactions and Fission*, pp. 89–143. Springer International Publishing, Cham, 2016. 10.1007/978-3-319-30651-3_3.

[13] K. K. David Albright, *Neptunium 237 and americium: World inventories and proliferation concerns*, 2005.

[14] A. Robock, L. Oman and G. L. Stenchikov, *Nuclear winter revisited with a modern climate model and current nuclear arsenals : Still catastrophic consequences*, *Journal of Geophysical Research* **112** (2007) .

[15] A. Bernstein, N. Bowden, B. L. Goldblum, P. Huber, I. Jovanovic and J. Mattingly, *Colloquium : Neutrino detectors as tools for nuclear security*, *Reviews of Modern Physics* **92** (2020) .

[16] C. Grupen, *Astroparticle physics; 1st ed.* Springer, Berlin, 2005, 10.1007/3-540-27670-X.

[17] S. M. Adams, C. S. Kochanek, J. F. Beacom, M. R. Vagins and K. Z. Stanek, *Observing the next galactic supernova*, *The Astrophysical Journal* **778** (2013) 164.

[18] G. Mention, M. Fechner, T. Lasserre, T. A. Mueller, D. Lhuillier, M. Cribier et al., *Reactor antineutrino anomaly*, *Phys. Rev. D* **83** (2011) 073006.

[19] K. N. Abazajian, M. A. Acero, S. K. Agarwalla, A. A. Aguilar-Arevalo, C. H. Albright, S. Antusch et al., *Light sterile neutrinos: A white paper*, 2012.

[20] S. Brice, *Monte Carlo and Analysis Techniques for the Sudbury Neutrino Observatory.* Queen's University, 1996.

[21] A. Oralbaev, M. Skorokhvatov and O. Titov, *The inverse beta decay: a study of cross section*, *Journal of Physics: Conference Series* **675** (2016) 012003.

[22] G. F. Knoll, *Radiation Detection and Measurement.* John Wiley and Sons Inc., New York/Chichester/Weinheim/
Brisbane/Toronto/Singapore, 2000.

[23] Y. Gao, "Detectors in Particle and Nuclear Physics Lecture 2: Interactions of Particles with Matter ." Lecture Slides, 2020.

[24] S. Palazzo, *Introduction to Neural Networks and Deep Learning.* University of Edinburgh, 2020.

[25] UC Davis Neutrino Group, "WATCHMAN."
`http://svoboda.ucdavis.edu/experiments/watchman/`.

[26] S. Paling, "AIT/WATCHMAN @ Boulby Lab: Overview of Boulby Lab our Science. Progress considerations for the new AIT/WATCHMAN project ."
`https://indico.cern.ch/event/704343/contributions/2940919/attachments/1632575/2603437/Boulby_THEIA_April18_Paling.pdf`, 2018.

[27] Private communication with Liz Kneale, 4th year PhD at University of Sheffield.

[28] GEANT4 collaboration, S. Agostinelli et al., *GEANT4: A Simulation toolkit*, *Nucl. Instrum. Meth. A* **506** (2003) 250.

[29] Cern, "Root data analysis framework." `https://root.cern.ch/`.

[30] Private communication with Dr. Matthew Needham.

[31] S. S. et al., "Gsim generators."
`https://rat.readthedocs.io/en/latest/generators.html`, 2014.

[32] Tensorflow, "Tensorflow." https://www.tensorflow.org/.

[33] Keras, "Keras." https://keras.io/about/.

[34] pandas, "About pandas." https://pandas.pydata.org/about/.

[35] A. Ingargiola and contributors, "What is the jupyter notebook?." https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html, 2015.

[36] S. learn developers, "sklearn.preprocessing.minmax_scale." https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html, 2019.

[37] WATCHMAN Collaboration, "Fred." https://github.com/AIT-WATCHMAN/FRED.

[38] Keras, "Adamax." https://keras.io/api/optimizers/adamax/.

[39] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017.

[40] L. N. Smith, *Cyclical learning rates for training neural networks*, in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 464–472, 2017, DOI.

[41] B. Kenstler, "Cylical learning rate." https://github.com/bckenstler/CLR.

[42] F. Wittman, "Learning rate finder as keras callback." https://gist.github.com/WittmannF/c55ed82d27248d18799e2be324a79473.

[43] Private communication with Dr. Gary Smith.

[44] C. Maklin, "Lstm recurrent neural network keras example." https://towardsdatascience.com/machine-learning-recurrent-neural-networks-and-long-short-term-\\memory-lstm-python-keras-example-86001ceaaebc.