Using Machine Learning to Improve Neutron Tagging Efficiency in Water Cherenkov Detectors

by

Matthew Stubbs

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

Master of Science

in the Department of Applied Computer Science

University of Winnipeg

Using Machine Learning to Improve Neutron Tagging Efficiency in Water Cherenkov Detectors

by

Matthew Stubbs

Supervisory Committee

Dr. B. Jamieson, Co-Supervisor
Chair of Physics Department

Dr. S. Ramanna, Co-Supervisor
Department of Applied Computer Science

Dr. C. Henry, Member
Department of Applied Computer Science

Dr. C. Bidinosti, Member
Department of Physics

Abstract

When an anti-neutrino collides with a proton in the atomic nucleus, it yields an anti-lepton and a free neutron. In a water Cherenkov neutrino detector like Super-K or the next generation Hyper-K, the free neutron is captured by a hydrogen or gadolinium nucleus about one hundred microseconds after the collision. The low-energy signal from the neutron capture (ranging from 2-8 MeV of gamma rays) is recorded by only tens of photomultiplier tubes (PMTs), making neutron captures difficult to distinguish from radioactive decay, muon spallation and other background sources. Improved methodologies for neutron tagging can advance understanding and enable new research over a survey of topics in particle physics. In this research, machine learning techniques are employed to optimize the neutron capture detection capability in the new intermediate water Cherenkov detector (IWCD) for Hyper-K. In particular, boosting decision tree (XGBoost) and graph neural network (GCN, DGCNN) models are developed and benchmarked against a statistical likelihood-based approach, achieving up to a 10% increase in classification accuracy. Characteristic features are also engineered from the datasets and analyzed using SHAP (SHapley Additive exPlanations) to provide insight into the pivotal factors influencing event type outcomes. Three main datasets were used for evaluative purposes in this research, each consisting of roughly 1.6 million events in total, divided nearly evenly between neutron capture and a distinct background electron source.

Contents

Abstract **iii**

Acknowledgements **x**

Dedication **xi**

1 Introduction **1**
1.1 Neutrinos 1
1.2 Cherenkov Radiation 6
1.3 Water Cherenkov Detectors 7
1.4 Neutron Tagging 10
1.5 Thesis Outline 12

2 Related Works **13**
2.1 Machine Learning in Particle Physics 13
2.2 Boosting Decision Trees 14
2.3 Deep Learning and Graph Neural Networks 15

3 Machine Learning Theory **17**
3.1 Introduction 17
3.2 XGBoost 18
3.3 Shapley values 22
3.4 Graph Neural Network (GNN) 24

4 Datasets and Likelihood **35**
4.1 Data Simulation 35
4.2 Likelihood Baseline Analysis 41

5 Feature Engineering **46**
5.1 Beta Parameters 47
5.2 Time of Flight 50
5.3 Distance to Wall 53
5.4 Mean Opening Angle 55
5.5 Consecutive Hit Angular RMS 56
5.6 Consecutive Hit Distance 58

6 XGBoost Results Analysis **61**

7 GNN Application **74**
7.1 GCN 74
7.2 DGCNN 79

8 Conclusions **83**

Appendix **96**
8.1 highE Dataset Supplementary 97
8.2 lowE Dataset Supplementary 102
8.3 Other 107

List of Figures

1.1 Beta radioactivity puzzle, observed energies less than predicted. . . 2
1.2 Cherenkov radiation diagram. 7
1.3 Diagram of Cherenkov effect for neutrino detection in water. 8
1.4 Schematic of the intermediate water Cherenkov detector. 10
1.5 Inverse beta decay diagram, showing neutron capture by proton or gadolinium nucleus. 11

3.1 Toy XGBoost example for ensemble of decision trees. 20
3.2 Example of the SHAP additive feature attribution method. 24
3.3 Example of an artificial neural network (ANN) architecture. 26
3.4 Basic graph neural network architecture, alternating filtering and activation layers. 29
3.5 Diagram of the edge convolutional operation from the DGCNN graph network model. 32
3.6 Diagram of an edge convolutional block (EdgeConv) from the DGCNN graph network model. 33

4.1 Representation of the simulated IWCD and its geometry. 37
4.2 Distribution of hit counts for neutron capture and background events, showing higher hits at higher energies. 38
4.3 Distribution of charge sums for the simulated neutron capture and electron background events, showing higher charge sums for higher energies. 38
4.4 Sources of background noise in the Super-Kamiokande detector. . . 40
4.5 Energy distributions of unstable isotopes produced in muon spallation. 41
4.6 Likelihood ratio for neutron and electron particle events as a function of event hits. 43
4.7 Bivariate KDE plots of event hits and charge sums. 44

5.1 Beta parameter calculation diagram. 48
5.2 Distributions of the beta parameters for the neutron capture and spallation background electron dataset. 50
5.3 Distributions of event timing for the spallation and neutron capture dataset. 51
5.4 Distributions of the RMS event times for the neutron capture and spallation dataset. 53
5.5 Distribution of vertex to wall distances for the neutron capture and spallation dataset. 54
5.6 Distributions of mean open angles for the neutron capture and spallation dataset. 56
5.7 Distributions of RMS consecutive hit angles for the neutron capture and spallation dataset. 58
5.8 Event display comparing a sample neutron to background event. . . 59
5.9 Distributions of consecutive hit PMT distances for the neutron capture and spallation dataset. 60

6.1 XGBoost confusion matrix for model trained on the neutron capture and spallation background dataset. 65
6.2 Local SHAP force plots are shown for two sample events, showing individual feature contributions. 66
6.3 Local SHAP decision plots are shown for two sample events, showing the individual feature contributions. 67
6.4 Combined SHAP decision plot for 500 randomly sampled events, showing more global feature impact trends. 68
6.5 SHAP beeswarm plot for the neutron capture and spallation background dataset. 70
6.6 Comparison of engineered features separated by event type for the neutron capture and spallation background dataset. 71
6.7 Feature importance by average absolute SHAP value for the neutron capture and spallation dataset. 73

7.1 DGCNN model training times for varying number of nearest neighbours k in the edge convolution block. 81

8.1 Distributions of beta parameters for the neutron capture and high energy electron background dataset. 97

8.2 Engineered feature distributions for the neutron capture and high energy electron background dataset. 98
8.3 XGBoost confusion matrix for the model trained on neutron capture and high energy electron background dataset. 99
8.4 SHAP beeswarm plot for the neutron capture and high energy electron background dataset. 100
8.5 Feature importance plot for the neutron capture and high energy electron background dataset. 101
8.6 Distributions of beta parameters for the neutron capture and low energy electron background dataset. 102
8.7 Distributions of engineered features for the neutron capture and low energy electron background dataset. 103
8.8 XGBoost confusion matrix for model trained on the neutron capture and low energy electron background dataset. 104
8.9 SHAP beeswarm plot for the neutron capture and low energy electron background dataset. 105
8.10 Feature importance plot for the neutron capture and low energy electron background dataset. 106
8.11 Feature importance plot for the XGBoost model trained on the neutron capture and low energy electron background dataset, sorted by the weight, gain and cover metrics. 107

List of Tables

4.1 Baseline classification accuracies using highest likelihood for the three neutron capture and electron background datasets. 45

6.1 Classification accuracies using XGBoost on the three neutron capture and background datasets. 64

7.1 GCN model applied to fixed input, fully connected and uniformly edge weighted graphs. 76
7.2 Accuracies for GCN model with dynamic, fully connected and uniformly edge weighted graphs. 76
7.3 Accuracies for GCN model applied to fixed input, fully connected graphs with inverse positional edge weighting. 78
7.4 Accuracies for GCN model applied to k-nearest neighbour connected graphs, varying k. 78
7.5 Applied DGCNN model architecture. 80
7.6 DGCNN model classification accuracies for varying number of nearest neighbours k in the edge convolution block. 80
7.7 DGCNN model classification accuracies with $k = 25$ nearest neighbours for the three neutron capture and electron background datasets. 82
7.8 Classification accuracy summary for all methods over all datasets. . 82

Acknowledgements

First, I would like to warmly thank Dr. Jamieson and Dr. Ramanna for their invaluable help and funding over the course of this research. From the time invested into hearing my updates and providing suggestions, helping guide the direction of my inquiry while allowing me freedom to steer in different directions, digging into the technical weeds together when necessary, responsiveness to my varied questions, patience and tireless editing, this writing would absolutely not be possible without their support. I also owe a debt of gratitude to Dr. John Walker for providing the foundation for this research topic and helping me find my feet over the first few months.

I would also like to extend my sincere thanks to the entire WatchMal collaboration, and particularly N. Prouse, W. Fedorko and P. de Perio. I am very grateful for their funding that allowed me the time to pursue my Masters degree, for providing the initial neutron capture dataset and the many tools to produce my own data simulations, for hearing my progress and offering invaluable feedback and for providing amazing opportunities to share my research and connect with others in the particle physics and machine learning world.

I am also very grateful to the University of Winnipeg and the department of Applied Computer Science in general. I had no idea before I began my studies at the UW what a warm, welcoming and supportive environment it would provide, the quality of CS courses it offered for graduate study, or the way in which it genuinely cares about its students and wants them to succeed. I will always recommend the UW for graduate CS study and look back fondly on my experience.

Finally, I would like to thank my fiancée Naomi for her unconditional support and love, my parents for always being there, my cat Freddy for always catching the mice and my dog Mabel for protecting me from the dangerous squirrels and various passers-by out the window during the course of my remote research work at home.

Dedication

To my family of origin
and my family of creation

Chapter 1

Introduction

1.1 Neutrinos

This research is focused on developing methods to better detect and understand the neutrino, through the improved detection of neutrons produced in neutrino interactions. The neutrino is a spin 1/2, neutrally charged elementary particle. At any given second, more than a trillion neutrinos are passing through our bodies. However, the chances of neutrinos interacting with matter is incredibly low. The neutrino, elusive and difficult to detect, is the keeper of many secrets of and beyond the Standard Model. For this reason, many large, sophisticated and expensive particle detectors have been constructed to identify them. The story of the neutrino begins in 1897, with the discovery of radioactivity by Henri Becquerel [1].

Radioactivity is the emission of particles from an unstable nucleus. Beta decay is one form of radioactivity in which an electron is emitted from the nucleus. Initially, this electron emission was thought to be the entire story of beta decay, as

$$ {}^{A}_{Z}\mathrm{X} \rightarrow {}^{A}_{Z+1}\mathrm{X}' + e^{-}. \tag{1.1} $$

However, experimental observations of beta decay did not satisfy conservation of energy (and angular momentum), which created a longstanding puzzle. Energy conservation states that the total energy of any isolated system should remain constant over time, neither increasing nor decreasing. According to this, in beta decay, the energy carried away by the electron would equal the difference of energies between the parent nucleus X and daughter nucleus X' (this difference between nucleus energies may be called Q). Multiple experiments found that the the energy of the electrons produced in beta decay was not, in fact, equal to Q. Instead, the observed electron energies were on a spectrum left of Q to lower energies, as shown

in Fig. 1.1.

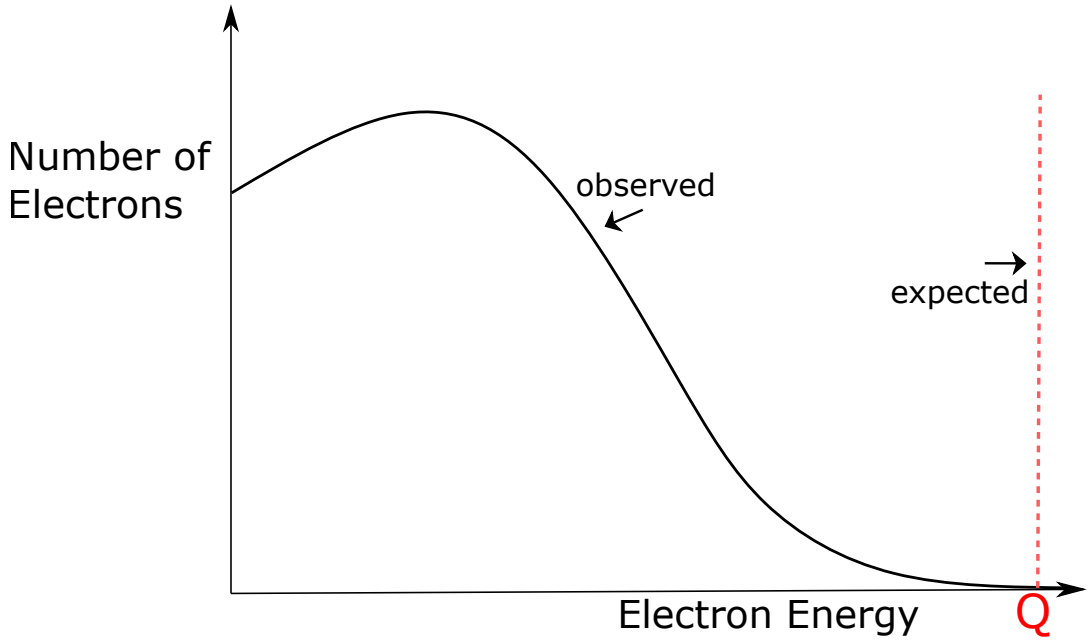


Figure 1.1: Diagram representing the beta radioactivity puzzle, in which the observed energies of electrons generated in beta decay were found to be less than the expected value of Q (as predicted by energy conservation).

Some believed that the beta decay energy spectrum demonstrated an exception to the rule of energy conservation [2]. However, others thought there must be some unknown particle carrying away some of the missing energy. In particular, in a famous letter in 1930, Wolfgang Pauli proposed the existence of a neutrally charged, light particle, which he believed to also be produced in beta decay [3]. He referred to this hypothesis as a 'desperate remedy' to preserve the energy conservation law. Pauli named this hypothetical particle the 'neutron.' Later, in 1932, James Chadwick announced the discovery of a neutral particle in the nucleus of the atom, but this particle was too massive to be the same as Pauli's 'neutron'. Therefore, Enrico Fermi renamed Pauli's 'neutron' to 'neutrino' (little neutron). By 1934, Fermi postulated the production of the neutrino along with the electron to resolve the lack of energy (and angular momentum) conservation in beta decay,

$$ {}^{A}_{Z}\mathrm{X} \rightarrow {}^{A}_{Z+1}\mathrm{X}' + e^{-} + \nu_e. \tag{1.2} $$

Fermi held the neutrino, ν_e, to be massless and neutrally charged. Equation 1.2 may also be rewritten to demonstrate the atomic nuclear decay of the neutron to

a proton, electron and neutrino, as

$$n \rightarrow p^{+} + e^{-} + \nu_{e}. \tag{1.3}$$

Fermi's theory also allowed for a neutrino to pick up charge and reveal itself by colliding with matter. For example, an electron neutrino could collide with a proton to produce a positron (electron antiparticle) and a neutron. This is the inverse beta decay (IBD) process, hypothesized by Bruno Pontecorvo in 1945 [4],

$$\nu_{e} + p^{+} \rightarrow e^{+} + n. \tag{1.4}$$

Armed with Fermi's theory and abundant data from beta decay experiments, two leading theorists of the time, Hans Bethe and Rudolf Peierls, calculated the probability of a neutrino interacting with matter [5]. They estimated the neutrino interaction cross-section in 1934 as roughly 10^{-44}cm^2. This number is so small that they concluded "there is no practically possible way of observing the neutrino." They described how a neutrino can pass throughout Earth without interruption "like a bullet through a bank of fog." However, the 1930s and 1940s saw a rapid development of nuclear fission reactors and fission bombs. These enormous neutrino sources transitioned the prospect of neutrino detection to the realm of possibility.

In 1951, Frederick Reines and Clyde Cowan commenced the 'Poltergeist project,' using fission reactors to try to detect the 'ghostly' neutrinos [6]. The experiment selected inverse beta decay (Eq. 1.4) to look for neutrinos through the delayed signal of positron annihilation (the e^{+} antiparticle will annihilate with the e^{-} particle) followed by neutron capture. The spirit of the design was similar to many modern setups. Large tanks were filled with water to provide the proton targets for the neutrinos. Cadmium chloride was added to the water, yielding cadmium nuclei to capture any free neutrons. The tanks were also surrounded by three detectors lined with 110 light-detecting photomultiplier tubes (PMTs) each to record the signals from the beta decay. In 1956, Reines and Cowan were successful in detecting neutrinos for the first time, an accomplishment which yielded them the Nobel Prize in 1995.

Reines and Cowan knew they had detected neutrinos, but they did not know that neutrinos exist in three possible 'flavour' states. In fact, they had detected electron neutrinos ν_{e}, so-named because they often show up with electrons in interactions. In 1962, Leon Lederman, Melvin Schwartz and Jack Steinberger discovered the muon neutrino at Columbia (ν_{μ}, often shows up with muons). They used a synchrotron to produce a beam of pi mesons and directed the beam at a 5,000 ton steel wall

made of old battleship plates. On the way, the pi mesons decayed to muons and muon neutrinos, but only the muon neutrinos could pass through the steel wall where they could be detected by a spark chamber. This discovery led to the 1998 Nobel Prize in physics [7]. The tau neutrino (ν_τ) was later discovered by Martin Perl and colleagues at SLAC in Stanford.

Aside from neutrino flavours, the production and quantity of solar neutrinos is integral to the story of our understanding of the neutrino. The standard solar model (SSM) posits that nuclear fusion is responsible for powering the sun and heating our planet, following

$$4p \rightarrow {}^4He + 2e^+ + 2\nu_e, \tag{1.5}$$

where (following intermediate interactions) the net effect is the fusion of four protons into a helium atom, releasing two positrons, two electron neutrinos and leftover energy. At the Brookhaven National Laboratory in New York, Raymond Davis Jr., after previous experiments on neutrinos from fission reactors, turned his attention to the detection of solar neutrinos. Nearly a mile underground in the Homestake Gold Mine in South Dakota, he oversaw the construction of a 100,000-gallon tank filled with cleaning fluid. The fluid provided chlorine atoms, which were known to interact with solar neutrinos to produce a countable number of radioactive argon atoms. Meanwhile, the tank was underground to avoid background neutrinos from cosmic rays. After years of refinement to the experimental method, Davis Jr was successful in detecting solar neutrinos [8]. However, the flux detected was only about one-third of the prediction from the SSM! This dilemma become known as the famous 'solar neutrino problem.'

Some believed the astrophysical model of the Sun was wrong. Others questioned the experimentation methodology. A third camp correctly believed the problem was due to a lack of understanding of neutrino physics [1]. Later radiochemical experiments also found a solar neutrino flux lower than expected, from 30% to 50%. One major advance was made in Japan in 1998, when the Super-Kamiokande collaboration, from the study of cosmic ray neutrinos, provided the first experimental evidence of neutrino oscillation [10]. This oscillatory "personality disorder" indicated that neutrinos can actually change from one flavour to another along their paths (for example from a muon neutrino to an electron neutrino)!

The final piece to the solar neutrino puzzle involved a second neutrino interaction mode called the 'neutral current' (NC) interaction. Previously, the only known (and

[1] Sociologist Trevor Pinch made a case study of scientists' responses to the solar neutrino problem [9]

experimentally studied) mode was the charged current (CC) interaction mode, in which the neutrino changes into its lepton counterpart. For example, the change of an electron neutrino into the charged electron,

$$\nu_e + (Z, N) \rightarrow (Z + 1, N - 1) + e^-, \tag{1.6}$$

is a CC interaction. However, in 1973 the NC mode was discovered, in which the neutrino (of any flavour) scatters off a nucleus, causing the nucleus to reach an excited state or disintegrate, but in which the neutrino leaves the interaction unchanged,

$$\nu_l + (Z, N) \rightarrow (Z, N)^* + \nu_l, \tag{1.7}$$

where $\nu_l = \{\nu_e, \nu_\mu, \nu_\tau\}$. Therefore, the possibility existed of a solar electron neutrino changing into a muon or tau neutrino on its path to Earth. After this flavour oscillation, the muon or tau neutrino would need a sufficiently high energy to convert into a muon or tauon through a CC interaction. This is due to the relatively higher masses of muons and taus (respectively 105 MeV and 1777 MeV) and Einstein's famous mass-energy equivalency equation $E = mc^2$. However, the solar neutrino energies were limited to about 14 MeV, too low for this to occur. Thus, it was possible that many solar neutrinos were going undetected because detectors were unable to detect the muon or tau neutrinos they had converted to on their way to Earth. The Sudbury Neutrino Observatory (SNO) in Ontario, Canada was the first to directly measure the solar neutrino flux from NC neutrinos. The measured NC flux was approximately three times higher than the CC solar neutrino flux [11]. This finding in 2001, together with the results from Super-Kamiokande, not only corroborated the phenomenon of neutrino oscillation, but also measured a neutrino flux in agreement with the SSM, thus confirming the nuclear fusion model of solar energy production.

Although great advances have been made in the field of neutrino physics, there are still many unknowns. For example, the Standard Model predicts a zero neutrino mass, in direct opposition to the non-zero mass indicated by experimentally observed neutrino oscillations. Moreover, although the relative neutrino masses are well-known, the absolute masses themselves are unknown. Another outstanding question is whether neutrinos are Dirac particles (the particle (e.g. e^-) differs from its antiparticle (e.g. e^+)) or Majorana particles (the particle is the same as its antiparticle) [12]. The search to resolve this outstanding question is underway

through various 'neutrinoless double beta decay' experiments, but results are inconclusive to date. These and other questions in neutrino physics may lay the framework to a deeper understanding of the physics of our Universe.

1.2 Cherenkov Radiation

As mentioned in Section 1.1, neutrinos are notoriously difficult to detect. However, one proven method utilizes the effect of Cherenkov radiation to make an indirect observation by detection of the charged leptons the neutrinos produce in interactions. This concept is the foundation of several modern neutrino detectors.

Cherenkov radiation, the faint bluish glow common to nuclear reactors, is named after the Soviet physicist Pavel Cherenkov. Although others before him had noticed this effect, he was the first to systematically study it, beginning in 1934. He spent hours in dark rooms, gradually adjusting his vision to detect faint luminescence with the naked eye. This luminescence emanated from certain salt solutions when exposed to radiation from a radium source. The phenomenon was also induced in ordinarily non-luminescent solvents like sulfuric acid and water [13]. Pavel Cherenkov, along with Ilya Frank and Igor Tamm, was jointly awarded the 1958 Nobel Prize in physics "for the discovery and the interpretation of the Cherenkov effect."

In general, although light travels at a constant speed c in vacuum, its wave velocity may be slower in a dielectric medium, due to the polarizability of the medium. Cherenkov radiation is produced when a charged particle (e.g. an electron or muon) moves through such a medium (e.g. water) faster than the wave velocity of light in the medium. This effect is similar to when a jet moves through air faster than the speed of sound, creating a sonic boom. On a quantum mechanical level, when the charged particle moves through the medium, it excites molecules within the medium to more energetic states. When the molecules return to ground level, they emit photons themselves. This electromagnetic radiation travels outwards as spherical waves. If the incoming charged particle moves quickly enough, these waves may add constructively with each other, leading to outward coherent radiation at the Cherenkov angle θ_C. The cosine of θ_C is given by

$$\cos\theta_C = \frac{v_{\text{light}}}{v_{\text{particle}}} = \frac{c}{n} * \frac{1}{v_{\text{particle}}}, \tag{1.8}$$

where v_{light} is the wave velocity of light in the medium, v_{particle} is the velocity of the

charged particle, c is the velocity of light in vacuum and n is the refraction index of the medium (1.333 for water). Cherenkov radiation yields a cone of bluish light in the direction of the charged particle. This process is represented in Fig. 1.2.

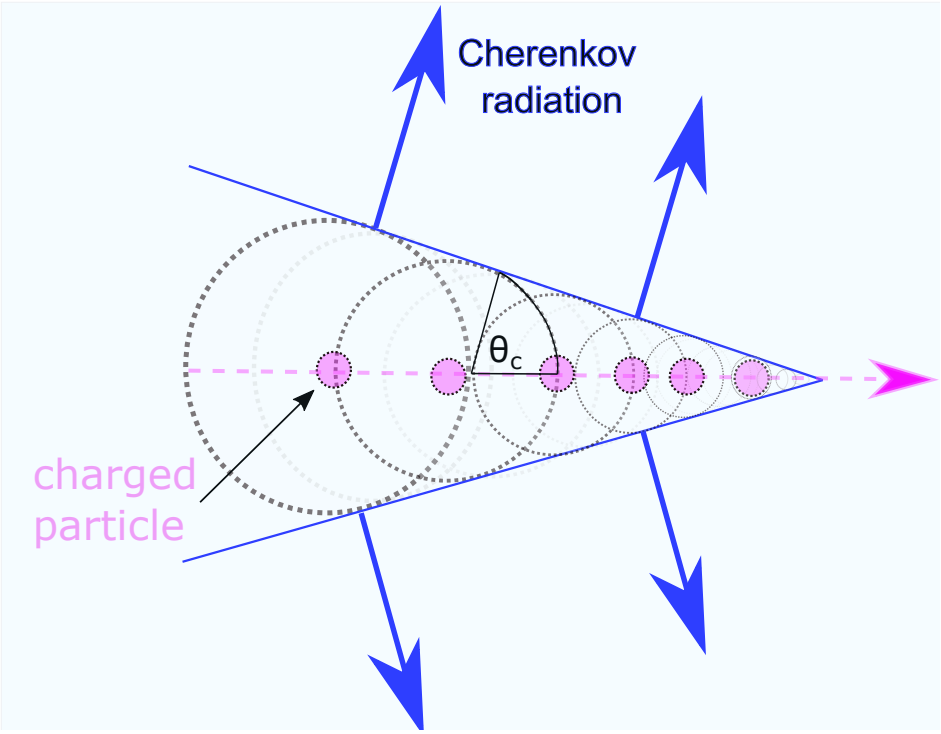


Figure 1.2: Cherenkov radiation is produced when a charged particle moves faster than the wave velocity of light in a medium. If the particle moves quickly enough, molecules in the medium produce spherical electromagnetic waves which may add constructively and radiate outwards at the Cherenkov angle θ_C. Cherenkov radiation is generally a blue hue and is often seen in nuclear reactors.

1.3 Water Cherenkov Detectors

Water Cherenkov (WC) detectors make use of the Cherenkov effect to indirectly identify neutrinos. They generally contain many rows of light-detecting PMT sensors which are capable of recording light flash signals in the detector. When photons from Cherenkov radiation strike the photocathode surface of a PMT, electrons are produced in accordance with the the photoelectric effect [14]. These electrons are accelerated by the high voltage electric field in the PMT and the output charge is then recorded. This charge is in proportion to the energy of the incident radiation.

A water Cherenkov detector is immersed in water to facilitate the Cherenkov effect, described in Section 1.2. When a neutrino interaction occurs and a charged

particle is produced in the water, the subsequent Cherenkov radiation cone is emitted outwards and is recorded by the PMTs lining the walls of the tank. The light cone forms a ring in two dimensions. After the data from the Cherenkov event is recorded from the PMTs, properties of the neutrino interaction, such as the energy, momentum and direction, may be reconstructed. Figure 1.3 shows how PMTs in WC detectors capture the Cherenkov radiation from relativistic charged leptons produced in neutrino interactions.

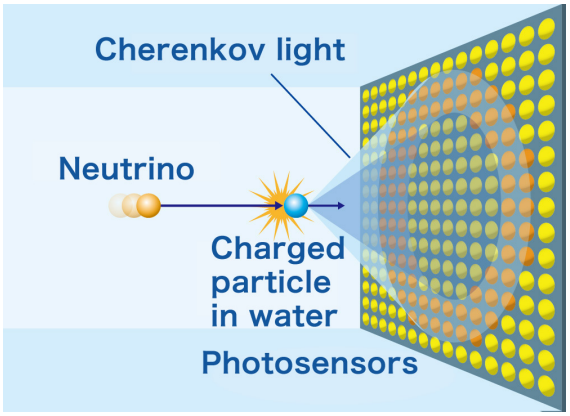


Figure 1.3: In water, a neutrino interaction produces a fast-moving charged particle which generates Cherenkov radiation as a cone of blue light in the direction of the charged particle. This radiation is recorded by rows of photosensors and can be traced back to information about the original neutrino. Retrieved from: `http://physicsopenlab.org/2016/04/24/diy-cherenkov-detector/`

The Irvine–Michigan–Brookhaven (IMB) detector, which began construction in 1979, was the pioneering WC detector. Built in a salt mine in Ohio, U.S.A, IMB began taking data in 1982, primarily with the goal of detecting proton decay. After this, Kamiokande, located in the Kamioka Township, Gifu Prefecture, Japan, was the second kton-scale WC detector to be deployed. Kamiokande began taking data in 1983. Together, IMB-3 (upgraded version of IMB) and Kamiokande achieved renown for successfully detecting neutrinos from the supernova SN1987a [15][16], the first discovery of its kind in neutrino astronomy. The Kamiokande experiment also achieved the first real-time detection of solar neutrinos in 1991 [17], winning Masatoshi Koshiba a share of the 2002 Nobel Prize in physics.

Building on the success of the Kamiokande WC detector, the Super-Kamiokande (SK) detector was built and began taking data in April, 1996. A major goal of SK was to look for evidence of neutrino oscillations. SK is also located in Gifu Prefecture in Japan, underground underneath over 3000 feet of rock to minimize background from cosmic rays. The cylindrical detector has a water capacity around 50,000 tons and is 39m in diameter and 42m tall. The walls of SK are lined with Hamamatsu PMTs to detect the Cherenkov radiation from neutrino interactions. SK contains an inner detector (ID) consisting of 11,146 inward-facing PMTs and an outer detector (OD) of 1885 outward-facing PMTs. Overall, the SK experiment boasts a number of impressive accomplishments in neutrino physics, including the

following: confirmation of the deficit of solar neutrinos [18], proof that solar neutrinos really do come from the Sun [18] and the first clear evidence of neutrino oscillation in atmospheric neutrinos [19].

Looking ahead, it has become apparent that greater statistics are necessary to confirm existing hypostheses and enable the experimental study of new physics. For this purpose, the Hyper-Kamiokande detector has begun construction in Japan, just 8 km south of SK. Based on the design of SK but larger and more sensitive to neutrino interactions, this next generation WC detector is planned to begin recording data by 2026 [20]. Amongst its collection of ambitious targets, HK aims to refine and accelerate the search for proton decay, further the study of neutrino mixing parameters and improve the detection of solar and other low energy neutrinos. HK will function as the new far detector to the beam of neutrinos produced at J-PARC, the Japanese Particle Accelerator in Tokai, Japan. However, despite its increased statistical capabilities, the HK project is constrained by the extent to which it can reduce its systematic uncertainties.

The technology in neutrino long baseline experiments has become sophisticated enough that "systematic uncertainties will dominate over statistical uncertainties when measuring neutrino disappearance" [21]. SK itself has an approximate 7% systematic uncertainty event rate prediction for neutrinos launched from the J-PARC particle accelerator. Given its increased size, HK requires lower systematic uncertainty to progress in the new physics it seeks to study. For example, HK requires a 3% systematic uncertainty rate to acheive sufficient certainty on CP (charge parity symmetry) violation measurements [21]. Abe et al. [22] have shown that the addition of a near detector close to (1km or 2km nearby) the neutrino beam source can dramatically lower the event uncertainty on the far detector. For this reason, in preparation to the data-taking launch of HK, an intermediate water Cherenkov detector (IWCD) has been proposed to reduce the systematic uncertainties of HK by making a near measurement to the neutrino beam.

The Canadian collaboration of the international SK/HK neutrino group is spearheading development of the IWCD project. IWCD was proposed as a "10 m diameter by 8 m tall water Cherenkov detector deployed in a 50 m deep pit about 1 km from the J-PARC neutrino source" [23] (latest proposed dimensions are 8 m diameter and 6 m height [24]). The IWCD is designed such that its elevation within the pit can be adjusted. This allows for the detector measurements to be taken over a range of angles relative to the beam source, helping to reconstruct the kinematics of the final state particles at the far detector relative to the incident beam neutrinos.

The IWCD will also include gadolinium doping, described in Section 1.4, making it more sensitive to the detection of neutron capture events. Figure 1.4 shows a diagram of the IWCD [23].

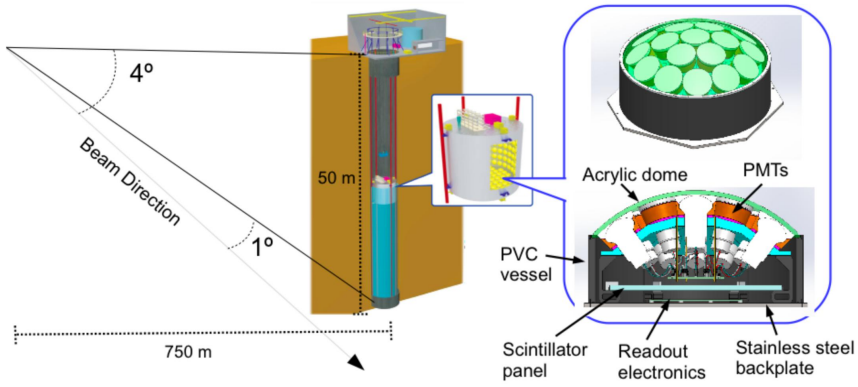


Figure 1.4: Schematic of the proposed intermediate water Cherenkov detector [23]. At 6 m tall and 8 m in diameter, the IWCD would be deployed underground in a 50 m pit. The IWCD would span a range of off-axis angles relative to the incident beam direction, helping constrain the far detector and original neutrino kinematics. The IWCD would include highly sensitive PMTs to record electrical particle signals and includes gadolinium doping to increase the sensitivity to neutron capture events. Figure courtesy of IWCD collaboration.

1.4 Neutron Tagging

One exciting frontier within experimental neutrino physics is the improved identification of neutrons from inverse beta decay reactions (Eq. 1.4). This task, referred to as 'neutron tagging,' is particularly challenging due to the low energy scale and faint signals involved. Progress in this field could lead to a host of advancements in particle physics, including a first detection of diffuse supernova background neutrinos [25], improved precision of neutrino mixing measurements, and even additional insight into black hole formation [26]. However, water Cherenkov detectors have historically been limited in their detection capability of these low energy neutron capture events.

Neutrons are commonly liberated in water due to the inverse beta decay (IBD) process, in which an electron antineutrino collides with a proton to yield a positron and a free neutron (Eq. 1.4). From there, the free neutron undergoes thermalization, colliding with neighbouring molecules and gradually losing energy until it reaches room temperature. Approximately 200 μs after thermalization, the free

neutron is captured by a proton or oxygen nucleus, releasing a gamma particle γ at 2.2 MeV, as per the following [27]:

$$n + p \rightarrow d + \gamma \tag{1.9}$$

where d is deuterium (or 'heavy hydrogen'), the isotope of hydrogen with a proton and neutron in the nucleus. The capture cross section of this neutron capture on a hydrogen nucleus (proton) is only 0.33 barns, and the resulting 2.2 MeV gamma produces such a faint light that it is very difficult to identify by a PMT in a WC detector. Many traditional WC detectors actually have thresholds of 5 MeV, high enough that none of these signals would be recorded at all [27].

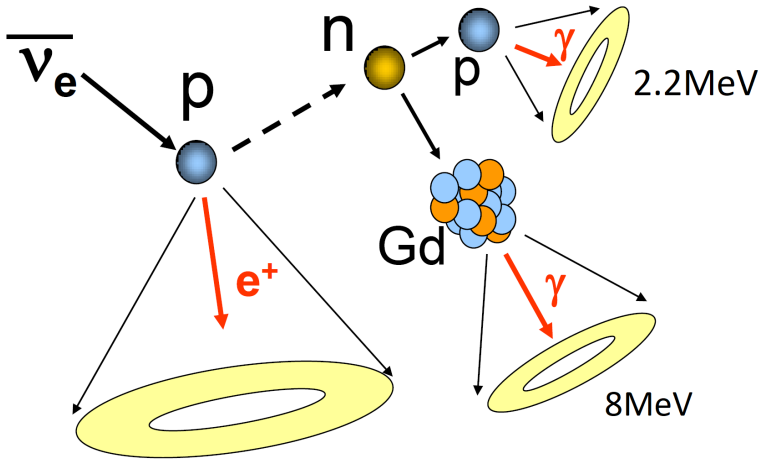


Figure 1.5: Inverse beta decay diagram representing two modes of neutron capture in a water Cherenkov detector. After the electron antineutrino collides with a proton in the water, a prompt positron ring is generated and a free neutron is produced. After thermalization, the neutron may be captured by a proton, yielding a gamma at 2.2 MeV. However, if gadolinium doping is included in the detector, the neutron may be captured by the gadolinium nucleus, leading to a higher 8 MeV gamma cascade. Image retrieved from Ref. [28].

To address this problem, Beacom and Vagins proposed the addition of gadolinium chloride ($GdCl_3$), a light, water soluble-compound, to the SK detector water in 2003 [28]. Gadolinium is known for having the "largest capture cross-section for thermal neutrons among all stable elements" [29]. At approximately 49700 barns, the gadolinium capture cross section is over six orders of magnitude larger than for free protons, leading to faster captures. Neutron capture on gadolinium also leads to an 8 MeV cascade of gammas (7.9 MeV cascade 80.5% of the time and a 8.5 MeV cascade 19.3% of the time [27]), a signal which is far easier to detect due to its relatively higher energy. Beacom and Vagins showed that only a 0.1% addition

of gadolinium by mass leads to at least a 90% probability of neutron capture on gadolinium (the other 10% or less of neutron captures still by hydrogen nuclei). In addition, the neutron capture by gadolinium after thermalization occurs in roughly 20 μs, nearly 10 times more quickly than capture on protons. Figure 1.5 represents the neutron capture following inverse beta decay with gadolinium included [28].

1.5 Thesis Outline

This research seeks to implement and test machine learning methods to improve the efficiency of neutron tagging for simulations of neutrino events within the gadolinium-doped IWCD. Chapter 2 introduces related works in the fields and intersections of particle physics, neutron tagging and machine learning. Chapter 3 provides a brief grounding in machine learning and introduces the relevant machine learning theories and algorithms for this research, including boosting decision trees (XGBoost), SHAP (SHapley Additive exPlanations) and graph neural networks (GCN and DGCNN). Chapter 4 discusses the data simulation process and the datasets used throughout the project. It also shows the implementation of a likelihood analysis based on the event charge and hit numbers to determine an analytic baseline of neutron tagging classification accuracy.

Chapter 5 discusses the process of engineering characteristic and discriminating features from the datasets. The implementation, tuning and results of XGBoost, as well as an analysis of relative feature importances is contained in Chapter 6. Chapter 7 presents the results of the application of the GCN and DGCNN graph network models on the original datasets, and compares various methods of graph construction and hyperparameter tuning. Finally, Chapter 8 concludes on the findings of the previous chapters.

Chapter 2

Related Works

2.1 Machine Learning in Particle Physics

The motivation for machine learning and its historical development in the field of particle physics is discussed in Ref. [30]. Reference [31] also describes how, although the Standard Model (SM) has established the modern foundations of particle physics, the SM is not complete and cannot fully explain such things as dark matter or why neutrinos have mass. In the coming decades, large detectors like Hyper-Kamiokande and the Large Hadron Collider (LHC) at CERN will accumulate an unprecedented volume of data and statistics. In the LHC, for example, Ref. [32] considers that proton beams collide at a frequency of $\sim$40 MHz and each collision may produce a shower of many new particles. With around 10^8 sensors in the detector to detect these particles, the sheer volume of data is beyond the grasp of traditional statistical methods. To unlock the underlying physical insights from these masses of data, new and improved analytical methods must be developed. Unlocking the potential of machine learning in this context will accelerate the pace of discovery and enable innovations in the field of particle physics.

Traditional means of event selection in particle physics are discussed in both Refs. [30] and [32]. These methods generally involve a series of Boolean cuts, or decisions, followed by statistical analyses on the remaining data. For example, only those events may be selected above a certain energy threshold or below a given charge limit. In these cases, the decision cuts and the subsequent analysis were focused on single variables. However, over the past several decades physicists have developed algorithms that employ machine learning to study multiple variables simultaneously. In physics, this is called multivariate analysis (MVA). Reference [32] describes the use of an assortment of machine learning techniques for MVA in the physics context, include support vector machines, kernel density

estimation, random forests, boosting decision trees, etc. Reference [33] provides an overview of applications of machine learning within the physical sciences, including applications to quantum computing, chemistry and cosmology. Reference [33] also discusses applications to particle physics, including jet physics and neutrino signal classification. Machine learning experiments are discussed for a variety of neutrino experiments, including the MicroBooNE collaboration, Deep Underground Neutrino Experiment (DUNE) and the IceCube Observatory at the south pole.

2.2 Boosting Decision Trees

Of the classic machine learning algorithms in particle physics, boosting decision trees (BDTs) are among the most widely employed. There is a substantial body of research discussing the use of boosting decision trees. The theory of boosting decision trees is presented in Section 3.2 for event classification and particle identification. For example, Ref. [34] details the improved performance of particle classification in the MiniBooNE experiment, which searches for neutrino oscillations, using BDTs compared to artificial neural networks. Reference [35] modifies the standard boosting decision tree algorithm to improve high-level triggering in detector data acquisition systems. A general BDT tree usage guidebook is presented in Ref. [36] for the hypothetical identification of the smuon particle and performance is compared to the classic 'cut-and-count' approach.

An example of the widespread usage of BDTs in particle physics is the CMS (Compact Muon Solenoid) experiment at the LHC. Reference [31] describes two use cases of BDTs in the CMS. First, BDTs are used to improve the resolution of the CMS calorimeter. As a proton or electron loses energy in the detector, its energy signal is recorded by multiple sensors. Compared to clustering the recorded energies of the sensors, passing their signals into a multivariate BDT leads to a significantly improved energy reconstruction (mass resolution). The second case, which is a particular highlight, is the contribution of BDTs to the discovery of the Higgs boson in 2012. The Higgs boson is produced only once for every few billion proton-proton collisions, and the products of its decay modes can be difficult to distinguish from background processes. Reference [31] describes how BDTs were used to improve the sensitivity of the detector to various Higgs decay modes, including diphoton decay ($H \to \gamma\gamma$) and antitau-tau pair decay ($H \to \tau^+\tau^-$), by an amount equivalent to adding 50% and 85% more data to the detector respectively.

2.3 Deep Learning and Graph Neural Networks

Beyond statistical cuts and classic machine learning algorithms like BDTs, deep learning has also been an increasingly applied method in particle physics tasks. Deep learning methods utilize neural network architectures with many hidden layers and large numbers of network parameters. Compared with traditional MVA machine learning approaches, deep learning can operate on the raw, low-level features from detector sensor data itself instead of relying on features extracted by a domain expert. The integration of deep learning into particle physics is discussed in Refs. [30], [31], [32] and [37]. In particular, Ref. [31] describes how information may be lost when human-engineered features are used which do not capture the full complexity of the given dataset.

Within the purview of deep learning techniques in particle physics, convolutional neural networks (CNNs), sequence models and graph neural networks (GNNs) comprise the current most frequently applied architectures. The computer vision approach consists of reconstructing the particle events as images and applying CNNs for deep learning. This method has been applied with success in various detector experiments [38, 39, 40]. However, the condensing of particle data into a 2 dimensional grid image format causes an inherent loss of information resulting from irregular detector geometries or the sparsity of the resulting image. For the case of sequence models, methods from natural language processing (RNNs, LSTMs, GRUs, etc.) have been adopted to the particle physics domain by modeling particles and measurement objects in a sequential order. The ordering is determined using insights from physics theory or empirical understanding from the data. Instances of this sequencing approach include tagging of jets containing b-hadrons in the ATLAS experiment [41] and classifying energetic hadronic decays in the CMS experiment [42]. However, the imposed, and somewhat artificial, ordering of objects in the sequence constrains the learning of the model. Indeed, for a given classification task, Ref. [43] shows that a permutation invariant network outperforms the sequence-based RNN benchmark.

The aforementioned limitations to sequence and computer vision models in particle physics are discussed in Ref. [44]. In addition, this thorough review paper provides an excellent survey of the theory and applications of graph neural networks to particle physics data. Graph neural networks (GNNs) encapsulate an emerging class of deep learning architecture which is rapidly gaining traction and momentum in the machine learning community. Reference [44] delineates GNN

particle physics applications into several categories, including graph classification, node classification and regression and edge classification. The graph classification task is most relevant to this research.

The category of graph classification may be further partitioned into jet classification and event classification. In particle physics, 'jets' are collimated sprays, or cascades, of particles which may be initiated by a variety of elementary particles, including gluons, quarks and bosons like the W, Z and Higgs boson [45]. Jet tagging, the task of identifying the the elementary particle initiating the jet, has long been a topic of research. Improved tagging efficiency is necessary to deepen understanding of physical jet interactions and standard model processes. In Ref. [45], the jet is viewed as an unordered structure of particles, analogous to the point cloud representation of shapes in 3D space. The authors propose the 'ParticleNet' method, which adapts the DGCNN architecture (see Section 3.4) for jets. Using this method, which employs the 'EdgeConv' block as an analogue for CNN convolution on 3D point clouds and updates the graph representation dynamically, the authors report state of the art performance on jet tagging tasks. Another approach views the jet particles as nodes on a graph and generalizes the GNN to include a learnable adjacency matrix, applying a variant of a message-passing neural network (MPNN) [46].

While jets represent a part of a particle collision occurrence, an 'event' refers to the full history of the particular physics process. Reference [44] gives an example of an event from astrophysics consisting of the collection of signals from a high energy particle interacting in the atmosphere. The event refers to the physical process at the origin of the ensemble of recorded data. While there are several research articles on the application of GNNs to jet classification, fewer papers have been published on similar approaches for event classification at this time. One excellent example is the application of GNNs for event signal classification in the IceCube neutrino observatory [47]. In this case, the irregular hexagonal geometry of the detector is itself modeled as a graph, where the sensors are the graph nodes and the edges represent their connections. Given the sparsity of activated sensors in an event, every event is considered as a different graph comprised only of the active sensors in the event. Although learning occurs over relatively small sample sizes, the authors report an approximate 3x improvement in signal to noise ratio compared to the physics baseline and the CNN approach.

Chapter 3

Machine Learning Theory

3.1 Introduction

In general, the notion of 'learning' indicates a process of receiving information, recognizing the patterns, then generalizing this newfound knowledge to unseen situations. To use a math analogy, consider a high school student learning about the logarithm function. This student might first study several solved examples, then attempt some questions on their own. After, they might compare their attempts with the solution set and try to learn from their mistakes. In machine learning, there is a remarkably similar process. The machine learning model is given some training examples to learn from (the *training set*), makes predictions, compares the predictions to the actual solutions (*labels*) and adjusts the model parameters to minimize the mistakes made (*loss*). This process repeats iteratively until the parameters have been sufficiently tuned that the model can successively generalize to unseen examples from the *testing set*.

The subject of machine learning has emerged as a subset of the broader domain of artificial intelligence, in which machines are programmed to try to replicate human intelligence. Machine learning attempts to differentiate itself from traditional computer computational systems by improving its performance, i.e. 'learning', autonomously over time, instead of having its parameters manually adjusted again and again by humans.

There are a few main tasks which machine learning systems attempt to solve: *classification*, *regression* and *clustering*. Classification, which is the main focus of this work, is the categorization of a given data example into a specific class. If there are two classes, this task is called *binary classification*. If there are more than two classes, this becomes *multiclass classification*. Another machine learning task is *regression*, which outputs a real number for every instance. For example,

a regression model may predict house prices based on their features (location, number of rooms, etc.). Classification and regression are both *supervised learning* tasks, which means that there is a known label (output) for every instance. In *unsupervised learning*, the label is not known. Clustering, the process of grouping similar data examples together, is an example of an unsupervised learning task, because there is no known target label or predetermined exact clusters.

3.2 XGBoost

Over the last several years, the machine learning model called 'XGBoost' has gained popularity for its performance in classification or regression tasks involving tabular data. For example, XGBoost model was used in the winning solutions of 17 out of 29 challenges issued by Kaggle, a data science website, in 2015 [48]. XGBoost has been successfully employed over a variety of domains and for diverse objectives, including vehicle accident detection [49], cancer diagnostics [50] and network intrusion detection [51]. This model has also been used in the context of physics event classification. After the discovery of the Higgs boson in 2012, attention shifted toward better understanding the physical properties of this new particle. An enormous amount of data was released from the Large Hadron Collider and a competition was held to find the best methodologies to distinguish the extremely rare Higgs boson signal from the abundant background processes in the LHC. Many of the top participants employed the XGBoost algorithm, and Chen and He discuss the advantages of XGBoost in this context [52].

The name XGBoost stands for 'eXtreme Gradient Boosting'. Gradient boosting employs an ensemble, or group, of individual learners to make predictions or produce output scores. For XGBoost, the individual learners are decision trees (discussed later). The gradient boosting concept traces back to the notion of "numerical optimization in function space," introduced by Jerome Friedman in his 2001 paper, "Greedy Function Approximation: A Gradient Boosting Machine" [53]. Given K data inputs $\mathbf{x} = \{x_1, x_2, \ldots, x_K\}$ with corresponding labels $\mathbf{y} = \{y_1, y_2, \ldots, y_K\}$, Friedman considered the task of approximating a mapping function $\mathcal{F}(\mathrm{x})$ from $\mathbf{x}$ to $\mathbf{y}$ which minimizes a given loss function over the input dataset. A classic optimization approach views $\mathcal{F}$ as a parametrized function $(F^*(x)|P)$ and adjusts the parameters P to minimize the loss. However, Friedman considered $\mathcal{F}$ as a set of tunable individual functions,

$$\mathcal{F} = F^*(x) = \sum_{k=0}^{K} f_k, \tag{3.1}$$

where there are K iterations, f_0 represents an initial guess and all subsequent functions $\{f_k(x)\}_1^K$ are 'boosts' or incremental steps from the previous function which are greedily added to minimize the overall loss. A given boosted function f_k is computed by calculating the gradient of the loss of the previous function f_{k-1} and adjusting f_k in the opposite direction, i.e. the path of steepest descent.

Following Friedman's work on gradient boosting, XGBoost was introduced by Chen and Guestrin in their 2016 paper, which considered the case of decision trees as the individual, or weak, learners in the function set [54]. In general, a decision tree applies classification or regression to an example by partitioning the example through a series of splits (decisions) from the root node to a leaf of the tree. The given tree splits are themselves computed by calculating which partition leads to maximum information gain. For XGBoost, the overall mapping function $\mathcal{F}$ is an ensemble of individual decision trees [54]:

$$\mathcal{F} = \{f(x) : w_{q(x)}\}(q : \mathbb{R}^m \to T, w \in \mathbb{R}^T), \tag{3.2}$$

where $\mathcal{F}$ represents the set of decision trees, m is the number of features and, per individual tree, q defines the structure, T is the number of leaves and w denotes the weight of each leaf. Thus, for any specific training example, the overall output is the additive sum of the outputs from every individual tree. A simplified example of this process, in the context of neutron versus electron particle event classification, is shown in Fig. 3.1.

To apply gradient boosting in the context of decision trees, an appropriate objective function (loss) must be defined. Chen and Guestrin define the overall objective function as the sum of a regular loss and a regularization term as

$$Obj = \sum_i L(y_i, \hat{y}_i) + \sum_k \Omega(f_k), \tag{3.3}$$

where L represents any convex differentiable loss function between the given true output y_i and predicted output $\hat{y}_i$, and the $\Omega(f_k)$ term applies regularization to each of the weak learners to prevent overfitting. To reduce model complexity, the regularization function adds a penalty corresponding to the size of the tree T and the magnitude of the weights w^2. The extent of the tree size and weight

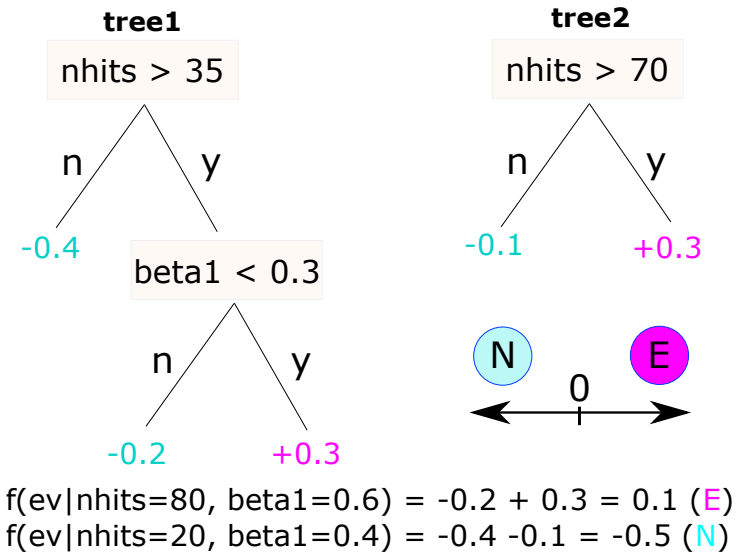


Figure 3.1: Simplified example of an ensemble of individual decision trees for neutron (N) versus electron (E) background event classification. The features are 'nhits' and 'beta1' while 'ev' is the event. The overall output prediction for a specific example is the sum of the regression outputs from each decision tree. For this example, a score above 0 indicates an electron-type event, while a score below 0 indicates a neutron-like event. Every successive tree is built by applying gradient boosting to minimize the objective function (loss). The regression outputs for the two simple event inputs shown here are 0.1 and -0.5 respectively.

penalties may be adjusted manually by the magnitudes of the γ and λ parameters respectively. The regularization term is represented as

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda w^2. \tag{3.4}$$

In the spirit of Freedman's gradient boosting algorithm, after the initial decision tree guess f_0, the next trees f_k are greedily and iteratively added to the tree ensemble to minimize the objective function. Thus, for a learning iteration m and training example i, the objective function may be written in terms of the newly added function f_k and the prediction at the previous iteration $\hat{y}^{(m-1)}$ as

$$Obj^m = \sum_i L(y_i, \hat{y}_i^{(m-1)} + f_k(x_i)) + \sum_k \Omega(f_k). \tag{3.5}$$

Next, the Taylor expansion to second order may be applied to Eq. 3.5 to enable the use of various loss functions L in the overall objective function. Taking this expansion and removing the constant terms yields

$$Obj^m = \sum_i [g_i f_k(x) + \frac{1}{2} h_i f_k(x)^2] + \sum_k \Omega(f_k), \tag{3.6}$$

where g_i and h_i are the first and second order derivatives (gradients) of the loss function L respectively for the instance i, with $g_i = \frac{dL(y_i, \hat{y}_i^{(m-1)})}{d\hat{y}_i^{(m-1)}}$ and $h_i = \frac{d^2 L(y_i, \hat{y}_i^{(m-1)})}{d(\hat{y}_i^{(m-1)})^2}$ [55]. From here, it is necessary to sum up the gradients g_i and h_i for every individual leaf per tree. Allowing I_j to denote the training example set for leaf j, we can also define $G_j = \sum_{i \in I_j} g_i$ and $H_j = \sum_{i \in I_j} h_i$. Then the objective function may be rewritten in terms of the summation over tree leaves,

$$Obj^m = \sum_{j=1}^{T} [G_j w_j + \frac{1}{2}(H_j + \lambda) w_j^2] + \gamma T, \tag{3.7}$$

where the regularization term has been expanded and w_j denotes the weight of each tree leaf. Now the tree weights may be adjusted, every m-th iteration, to minimize the objective function. The optimization problem is

$$\frac{\partial Obj^m}{\partial w_j} = G_j + (H_j + \lambda) w_j^2 = 0, \tag{3.8}$$

and the corresponding leaf weight which minimizes the objective function is $w_j = -\frac{G_j}{H_j + \lambda}$. Plugging this weight back into Eq. 3.7, the resulting objective function may be used to determine the best tree structure for the individual decision tree weak learner at iteration m:

$$Obj^m = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T. \tag{3.9}$$

Practically, when constructing a given decision tree in the XGBoost ensemble, it is too computationally expensive to iterate through all possible tree structures and compute the objective function for each possibility. Instead, a greedy approach is applied where, starting at the tree node, branches are successively added by finding the particular split which leads to maximum gain. At a given tree leaf, the information gain is computed by subtracting the objective function of the leaf by the objective function of a possible split. The objective function for a single leaf is

$$Obj^m = -\frac{1}{2} \frac{G_j^2}{H_j + \lambda} + \gamma T. \tag{3.10}$$

The objective function value resulting from splitting this leaf into a left and right child may also be calculated as

$$Obj^m = -\frac{1}{2}(\frac{G_{jL}^2}{H_{jL}+\lambda} + \frac{G_{jR}^2}{H_{jR}+\lambda}) + 2\gamma, \tag{3.11}$$

where L and R represent the left and right child leaves of this potential leaf split, respectively. Finally, the information gain may be computed as $Gain = Obj_{leaf}$ - Obj_{split}, yielding

$$Gain = \frac{1}{2}[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}] - \gamma. \tag{3.12}$$

Over every training iteration of the XGBoost model, the gain formula (Eq. 3.12) is applied over all possible split points and the split is chosen with maximum gain. Note that the tree grows only while the maximum gain is positive, as negative gain values will not cause a leaf to generate splits. To optimize performance runtime, running tallies of G_L and G_H are stored in memory for each leaf and the split calculation occurs by "scanning left to right through all feature values in a leaf in sorted order" [55]. For n training examples with m features each, the split point computation over all decision trees has a runtime of $\mathcal{O}(nm)$.

3.3 Shapley values

When a machine learning model trains on specific input features, an understanding of the relative importances of the features can help interpret the results. Given that several feature importance metrics exist already (gain, cover, etc.), deriving a measure of goodness for a given metric can help with comparison. In particular, a good feature importance metric should satisfy the properties of *consistency*, *accuracy* and *missingness* [56]. Consistency states that if the model is changed to rely more heavily on a particular feature, the importance attributed to that feature should not decrease (only increase or stay the same). Accuracy requires that the sum of the feature importances should equal the total importance of the model. Finally, missingness states that features with no impact on the model output should have no attributed impact. In the Lundberg and Lee paper on SHAP values [56], Theorem 1 posits that there is only one unique solution which satisfies the above properties. This solution is based on the Shapley value.

The Shapley value traces back to Lloyd Shapley's paper "stochastic games," published in Princeton in 1953 [57]. At the time, Shapley was studying the field of cooperative game theory. Cooperative game theory differs from non-cooperative game theory by focusing on collective actions and coalitions rather than individual player actions and payoffs. Shapley was searching for a mapping from a coalitional single game to a numeric payoff vector. At that time the notion of reaching a single point solution seemed implausible due to the lack of information in the coalitional form game. However, Shapley found an intuitive solution by searching for a set of "reasonable axioms" (efficiency, symmetry, dummy and additivity) [57]. His result, the Shapley value, can be viewed as an "index for measuring the power of players in a game" [58]. In the context of particle physics machine learning classification, the player is analogous to the event feature, the game is analogous to the event and the label is the analogue of the numeric payoff output.

Winter's paper [58] reviews the theoretical framework for the derivation of the Shapley values. In this setting, the value operator $\phi_i(v)$ represents the measure of player i's importance in the coalitional form game v, defined over a finite set of players $\{1, 2, 3, ..., n\}$. Given a permutation π over the player set, $p^i_\pi = \{j : \pi(i) > \pi(j)\}$ represents the set of players before player i in π. The marginal contribution $\phi_i(v)$ of the player i in π is $v(p^i_\pi \cup i) - v(p^i_\pi)$. The average marginal contribution of player i in the game v may be calculated from the sum of marginal contributions over the set of all permutations Π,

$$\phi_i(v) = 1/n! \sum_{\pi \epsilon \Pi} v(p^i_\pi \cup i) - v(p^i_\pi). \tag{3.13}$$

Lundberg and Lee [56] extend this definition (Eq. 3.13), introducing the "SHAP" values as equal to the Shapley values of a "conditional expectation function of the original model." They also introduce the concept of the "explanation model" in which the output prediction of a machine learning model may be viewed as a model itself. Their definition of an '*Additive Feature Attribution Method*' is one in which the explanation model may be represented as a linear function of binary variables,

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i, \tag{3.14}$$

where g is the explanation model, $z' \epsilon \{0, 1\}^M$ is a binary basis, M is the number of simplified input features and $\phi_i \, \epsilon \, \mathbb{R}$ is the marginal contribution of each feature

to the output $g(z')$. Lundberg and Lee review six existing feature attribution methods and derive the result that only methods which utilize the Shapley values can satisfy the aforementioned properties of consistency, accuracy and missingness. They introduce the "SHAP" values as the classic Shapley values (Eq. 3.13) where the difference $v(p_\pi^i \cup i) - v(p_\pi^i)$ represents the difference in expectation values of the model over the permutation of features π with and without inclusion of feature i. As with the regular Shapley values, the SHAP value is computed over the set of all permutations Π.

Upon computation of the SHAP values, each feature is attributed a SHAP value which represents the "change in expected model prediction when conditioning on that feature." For a trained model f and features $(x_1, x_2, \ldots, x_n)$, the overall model output may then be written in terms of Eq. 3.15, where ϕ_i is the SHAP value of the i-th feature, E is the expectation function of the model, $\phi_0 = E[f(z)]$ is the base value of the model when no features are known and M is the number of features. The sample force plot in Fig. 3.2 from [56] gives a representative example of Eq. 3.15.

$$\begin{aligned} g(z) &= \phi_0 + \phi_1 + ... + \phi_M \\ &= E[f(z)] + E[f(z)|z_1 = x_1] + ... + E[f(z)|z_{1,2,...,M} = x_{1,2,...,M}] \end{aligned} \tag{3.15}$$

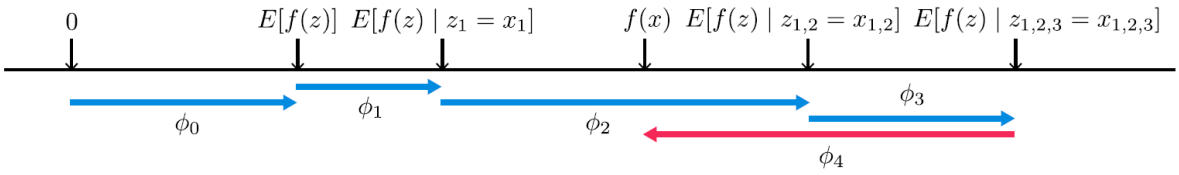


Figure 3.2: Example of the additive feature attribution method applied by SHAP. The overall model prediction is decomposed into the sum of feature attributions (SHAP values) increasing (blue, $\phi_0, \ldots, \phi_3$) or decreasing the output from the base expectation (red, ϕ_4). The SHAP value of feature ϕ_{x_i} is the expectation function of the model when conditioning on features $x_1, \ldots, x_i$. Example retrieved from Ref. [56].

3.4 Graph Neural Network (GNN)

Deep Learning

Traditional machine learning algorithms, such as regression or decision trees, have proven to be particularly effective at learning from tabular data. Examples of tabu-

lar data, which format neatly into rows and columns, include financial spreadsheets, weather data and sport statistics. However, these traditional learning algorithms have historically struggled to learn well from natural data. Natural data, including images, natural language, audio recordings and some particle physics data, does not translate easily into a tabular format. Before the advent of 'deep learning,' it was necessary to preprocess the data first into a tabular format before machine learning could be applied. This manual step requires expertise and domain knowledge of the problem environment to choose relevant, informative features. Even so, it is difficult to impossible to fully extract all the relevant information using manually engineered features. Researchers over the past few decades have actively searched for methods of learning from raw natural data in different contexts.

The solution that has emerged is now known as 'deep learning'. Deep learning arose from the study of artificial neural networks (ANN) in the 20th century. ANNs, largely inspired by the biology of the human brain, consist of an input layer, hidden layers and a final output layer. Each layer consists of a specified number of compute nodes, or *perceptrons*. Information is propagated from the input nodes, through the hidden layers and finally output at the final (output) layer. This kind of arrangement, where data propagates from input to output, is known as a *feedforward* neural network. A feedforward neural network with multiple layers of perceptrons is sometimes called a 'multi-layer perceptron' (MLP) network. The final layer can relay class probabilities for a classification problem. In a fully connected network, every node in each layer (except the input layer) is connected to every node in the previous layer by a vector of weights. Each compute node generates an output *activation* value by calculating the weighted sum of the weights and activations from the connections of the previous layer. These new activations are then propagated forward in the network to the next layer.

In practice, the stochastic gradient descent algorithm (SGD) is commonly used to train the ANN. For supervised learning with SGD, the error is calculated by comparing the output prediction to the true value for every training example. A gradient vector is then computed using a technique called *backpropagation* for all the node weight parameters. The chain rule is used to calculate vector derivatives propagating backward from output to input layer. For a given weight, this gradient determines how much the overall error would change if the weight were increased. Every weight is then adjusted in the opposite direction to decrease the overall error. This process is then repeated until the model has been fully trained over the full training dataset. A physical analogy might be the process of slowly and repeatedly

tuning knobs on an old television to try to get a clear signal.

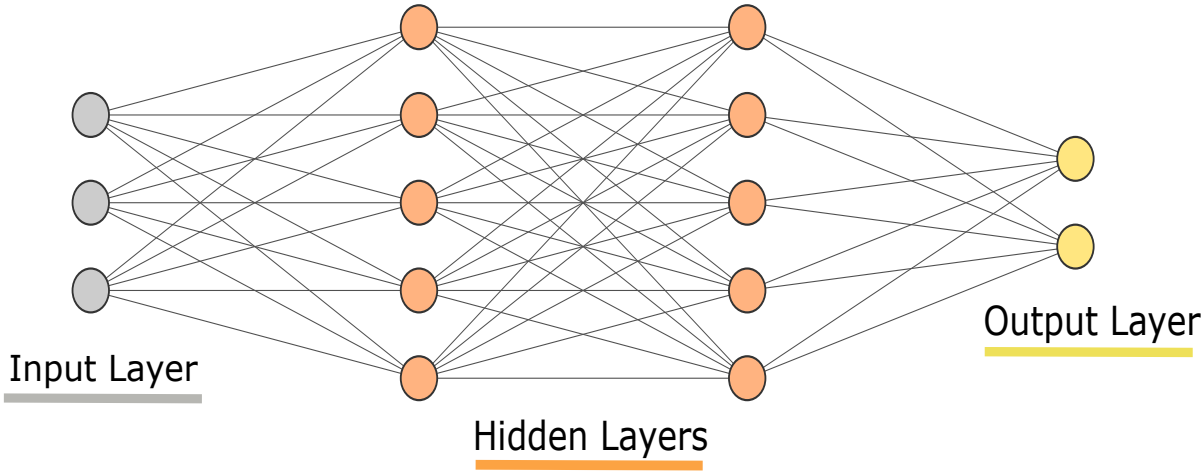


Figure 3.3: Example of an artificial neural network (ANN) architecture. Compute nodes are connected via learnable weight parameters. Information travels from the input layer, through the hidden layers, to the output layer. The loss is computed by comparing the output predictions to the real outputs, then backpropagation updates the weights to train the network. Figure retrieved using tool from Ref. [59].

The ability to train feedforward artificial neural networks using stochastic gradient descent and backpropagation was discovered in the 1970s [60]. While ANNs were shown to be effective with learning on tabular datasets, they were severely limited when presented with the task of learning from natural data like images and natural language. For images, computation time and permutation generalizibility were significant impediments. Mapping pixel values to input neurons in an ANN quickly introduced billions of weighted connections, causing shallow and computationally intractable networks with poor ability to generalize common patterns. With natural language, aside from computational cost, the fixed number of input and output neuron units in the ANN did not translate well to the variable input and output dimensions required for different tasks like translation or name recognition.

Eventually, new network architectures were developed which were better suited to handle natural data. In addition, they proved capable of learning over many hidden layers, which led to the term 'deep learning.' The most revolutionary model was, arguably, the convolution neural network (ConvNet or CNN), used for learning on images. The ConvNet introduced the concept of convolutional and pooling layers, which were added to the classic ANN architecture [61]. These new layers were used alternatingly before the usual fully-connected output layer was applied to yield the output scores. The convolutional layers applied convolutional filters

over the input pixels with learnable filter parameters, reducing the total number of parameters and enabling the learning of repeatable features like edges, contours and specific shapes in the image. The pooling layers functioned to merge and assemble semantically similar features learned from the convolutional filters. By repeatedly stacking the convolution and pooling layers, increasingly more complex shapes became learnable throughout successive layers of the the ConvNet. For example, given an image of a face, earlier layers might learn the generic outline, while later layers could recognize the nose, eyes, chin and eventually the face as a whole. In 2012, researchers from the University of Toronto applied a deep ConvNet to the popular 'ImageNet' dataset of over 14 million images and achieved higher than a 10% accuracy improvement compared to the previous state of the art model [62].

Deep learning network architectures like the CNN and recurrent neural network (RNN) [63] quickly raised the performance bar on learning on images and sequential data. However, *geometric deep learning*, the umbrella term for the task of deep learning on graph data, has taken longer to successfully apply and generalize to different contexts of graph learning. Over the past decade especially, a plethora of research has gone into the development of network architectures for data input in the form of graphs [64]. A given graph G is denoted by its set of vertices and edges G = {V, E}, where the nodes represent objects or concepts and the edges represent their relationships. A variety of situations may be modeled by graphs, including social networks, molecules, transportation routes, Internet traffic, etc.

Learning on Graphs

In particular, particle physics IWCD data may be naturally represented by a graph. For a given physics event, a particular subset of PMTs within the detector may record a signal. For the datasets used in this study, every signal records the time and deposited charge of that particular hit, as well as the 3-dimensional position and orientation of that particular PMT. A graph can model a particular event with the hit PMTs represented by the nodes and the edges as the connections between the hit PMTs.

Without graphs, one way of learning from this data is to preprocess it into a 2D image using an event display and then apply a ConvNet. Indeed, research has already been conducted in this manner [65, 66]. However, this manual step relies on the event display preprocessing and discards geometric topology information. Unlike this process, a *graph neural network* (GNN) is designed to operate directly on

data input as a graph. One of the main purposes of this research is to determine the applicability, performance and feasibility of GNNs on the IWCD particle physics data. In particular, CNNs tend to struggle in the low energy regime where the number of event hits is small and the image is sparsely filled. Therefore, GNNs are studied in the low energy, low hit context of neutron capture events.

The origin of deep learning on graphs traces back to the late 1990s, when RNNs were applied to directed, acyclic graphs (directional edges, no edge with self loops) [64]. With this approach, node feature states are updated in successive layers until equilibrium is reached. This technique was later generalized to cyclic graphs as well in 2008 [67]. Soon after, following the widespread success of ConvNets, significant interest grew in generalizing some of the concepts from ConvNets to learning on graphs. The first successful adaption of the convolution operation to graphs was developed by Bruna et al. in 2013 using Laplacian eigenvectors [68]. The computational complexity of this procedure was later greatly reduced by applying polynomial spectral filters instead of Laplacian eigenvectors [69, 70]. Approaches have also been developed which apply spatial, and not spectral, filters for the convolutional operation [71].

The general formulation of a graph neural network is as follows. To start, every node begins with its unique feature representation from the dataset input. Throughout the layers of the network, every node will update its representation vector repeatedly. The node feature vectors are updated by message passing. Every node both sends messages to and receives messages from its neighbouring connections. The message passing mechanics are specified through the filtering operation. In some GNN models, after the message passing step, every node passes the average of its own values and those of its neighbours through a fully connected layer to update their representation vector.

This alternating sequence of filtering and activation layers is inspired by the alternating convolutional and pooling layers from ConvNets. This process is shown in Fig. 3.4, where $h_1, ..., h_k$ are the filtering layers and $a_1, ..., a_k$ represent the activation updates for layers 1 to k [72]. Once all hidden layers have finished their computations, the output node labels may be used directly in node-focused tasks, or the node outputs may be pooled together to obtain an overall coarsened representation for graph classification. In Fig. 3.4, the pooling layer p at the end shows the aggregation of the node outputs for graph classification.

The output node states are a function of the input features and the structure of the graph, which may be represented by its corresponding adjacency matrix. Thus

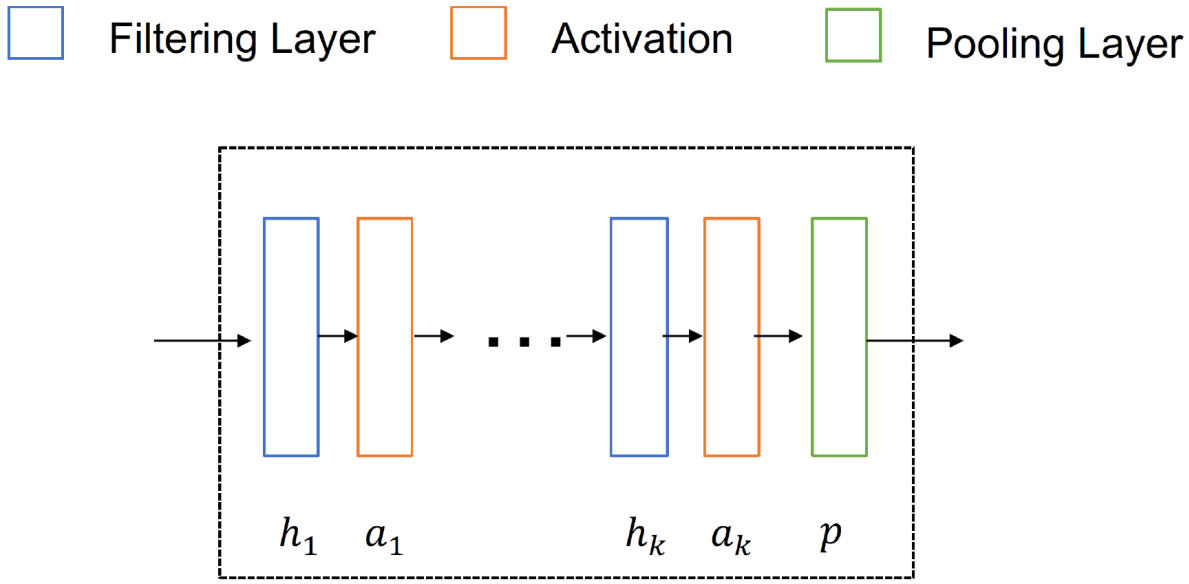


Figure 3.4: Basic structure of a generic graph neural network architecture. Alternating layers of filtering $(h_1, ..., h_k)$ and activation computations $(a_1, ..., a_k)$ are stacked for a network of k hidden layers. After this, the node labels can be computed directly, or a pooling layer p can be applied for graph classification. Retrieved from [72].

the generic mathematic description for a graph neural network may be summarized as

$$\mathbf{F}^{of} = h(\mathbf{A}, \mathbf{F}^{if}), \tag{3.16}$$

where $\mathbf{F}^{of}$ represents the output feature vectors, h denotes the graph filter, $\mathbf{A}$ is the adjacency matrix of the graph and $\mathbf{F}^{if}$ represents the input feature vectors for all the nodes [72]. The node filtering mechanics and other intermediate processes are changeable based on the specific GNN architecture. Two of these architectures are applied and discussed in this research, and their mechanisms are described below.

GCN (Graph Convolution Network)

Kipf and Welling demonstrated the successful approach of using a convolutional architecture to learn on graphs in their paper "Semi-supervised classification with graph convolutional networks" in 2017 [70]. This approach is based on a first-order approximation of spectral graph convolution.

For signal or image processing, the 'spectral' decomposition of a function relates to its decomposition into sine and cosine frequency components. However, the spectral decomposition of a graph denotes the breakdown of the graph's Laplacian

matrix $\mathcal{L}$ into its elementary orthogonal components, i.e. the *eigen-decomposition* of $\mathcal{L}$. The Laplacian $\mathcal{L}$ represents a normalization of the adjacency matrix A of the graph and can be represented as $\mathcal{L} = U\Lambda U^T$ where U are the eigenvectors and Λ are the eigenvalues of $\mathcal{L}$ [70]. Convolution in the spatial domain implies multiplication in the frequency domain. Therefore the spectral convolution of a graph can be written as the multiplication of a signal x (a scalar for every node) with a filter g_Θ (a function of eigenvalues Λ which is parametrized by Θ in the frequency domain):

$$g_\Theta \star x = U g_\Theta \, U^T x \;\; [72]. \tag{3.17}$$

The computation (Eq. 3.17) is expensive and simply computing the eigendecomposition of the Laplacian L might be a computational bottleneck. To simplify the calculation, Hammond et al. [73] proposed a truncation of the function $g_\Theta(\Lambda)$ in terms of the first K Chebyshev polynomials:

$$g_\theta(\Lambda) = \sum_{k=0}^{K} \theta'_k T_k(\tilde{\Lambda}), \tag{3.18}$$

where $\tilde{\Lambda}$ is a rescaled version of Λ [70] and $\tilde{\Lambda} = \frac{2}{\lambda_{max}}\Lambda - I_N$ where λ_{max} denotes the largest eigenvalue of L and I_N is the identity matrix. The convolution of the signal x can then be rewritten as

$$g'_\theta \star x \approx \sum_{k=0}^{K} \theta'_k \; T_k(\tilde{L}) \; x, \tag{3.19}$$

where $\tilde{L}$ is a rescaled version of the Laplacian. A graph convolutional model can be constructed by stacking multiple layers in the form of Eq. 3.19 with each layer followed by a non-linearity calculation. By taking the first-order Chebyshev approximation K=1 and further constraining other parameters, we arrive at the spectral convolution form

$$g_\theta \star x \approx \theta(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}), \tag{3.20}$$

with $\tilde{A} = A + I_N$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. Generalizing to a signal X with C input channels, Θ a matrix of filter parameters and Z the convolved signal matrix, we can express

$$Z = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}} \; \Theta \; X. \tag{3.21}$$

This filtering operation greatly reduces the time complexity of the computation. Finally the multi-layer propagation rule for the GCN model is presented as

$$H^{l+1} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}\ H^l\ W^l), \tag{3.22}$$

where H^l is the node feature matrix at layer l, H^{l+1} represents the feature matrix at the next layer $l+1$, W^l denotes the matrix of weights at layer l and σ is an activation function such as the rectified linear activation unit (ReLU).

Although the GCN model is presented for node classification tasks, it can be converted to graph-focused classification by the use of a pooling operation at the end of the network model. For illustrative purposes, there is the example of a simple 2-layer GCN model from Kipf and Welling's paper,

$$Z = f(X, A) = softmax(\hat{A}\ ReLU(\hat{A}XW^0)W^1) \tag{3.23}$$

where $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ is computed as a preprocessing step [70]. The first layer multiplies $\hat{A}$ by the initial node features $H^0 = X$ and weights W^0. The ReLU non-linearity is applied to this product $(\hat{A}XW^0)$ to compute the output of the first layer and input to the second layer. $\hat{A}$ then multiplies by the second layer input and weights W^1. Finally a softmax function is applied to this product to compute the normalized output probabilities for the node labels. For graph classification, a pooling operation would then be applied over the nodes to calculate the predicted graph label. Afterward, the network weights are updated by computing the loss [70].

DGCNN (Dynamic Graph Convolutional Neural Network)

The dynamic graph convolution neural network (DGCNN), introduced by Wang et al. [74], was designed specifically to learn from point cloud graphs for segmentation or classification tasks. Point clouds are collections of three-dimensional coordinates (points) in Euclidean space. However, the DGCNN model also allows the graph nodes to include other features in addition to the spatial coordinates. The main feature of the DGCNN model is the introduction of the 'EdgeConv' convolutional operator. EdgeConv is designed to learn edge features between node pairs, i.e. a node and its neighbouring connections. The DGCNN model is dynamic because, for every EdgeConv block, the graph representation is updated. This departs from the action of operating on a fixed graph like most other GNN architectures.

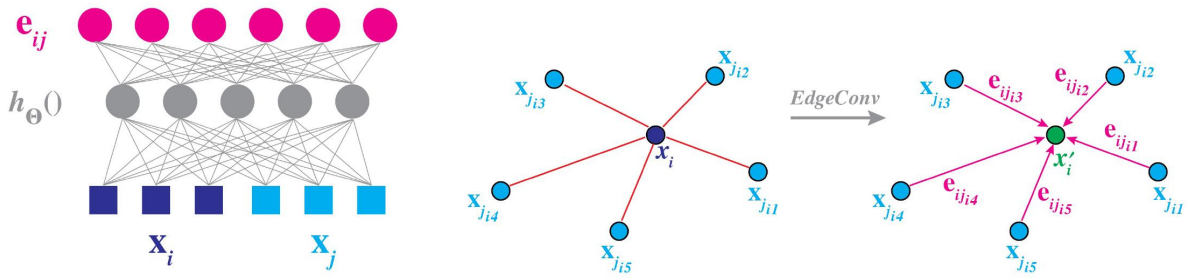


Figure 3.5: The EdgeConv operation is shown for a pair of nodes $\mathbf{x_i}$ and $\mathbf{x_j}$. The node feature vectors are passed through a fully connected layer $h_\Theta()$ with learnable weights Θ to calculate a set of edge features $\mathbf{e}_{ij}$ between the node pair. All nodes then update their vector representations by aggregating these learned edge features. In this example, the new representation of $\mathbf{x}_i$, $\mathbf{x}_i^{'}$ is calculated by aggregating the set of learned edge features $\mathbf{e}_{ij_{i1}}, ..., \mathbf{e}_{ij_{i5}}$. Figure retrieved from Ref. [74].

In the DGCNN model, EdgeConv is applied for every node and its k nearest neighbours in semantic space, where k is a tunable hyperparameter. For any two nodes $\mathbf{x}_i$ and $\mathbf{x}_j$, a fully connected layer $h_\Theta()$ with learnable weights Θ and an adjustable number of compute units is applied to learn the pairwise edge features $\mathbf{e}_{ij}$. The node representations are then updated by aggregating these edge features. For example, as shown in Fig. 3.5, the node representation for $\mathbf{x}_i$ is updated by aggregating the learned edge features $\mathbf{e}_{ij_{i1}}, ..., \mathbf{e}_{ij_{i5}}$ (self loop excluded).

The matrix mechanics of the edge convolution are as follows. For n nodes containing f features each, the EdgeConv block takes as input the corresponding $n * f$-dimensional tensor and computes the k nearest neighbour (k-nn) graph for every node. The activations $a_1, ..., a_n$ are computed for every node by passing its k-nn graph representation through a shared multi-layer perceptron. This generates an $n * k * a_n$ -dimensional tensor overall. Pooling is then applied over the learned features to aggregate and update the node representations, yielding an output $n * a_n$ tensor. This process is demonstrated in Fig. 3.6. Given the updated node representations, future EdgeConv iterations will yield different k-nn, making the model architecture dynamic [74].

The pairwise edge features in an EdgeConv block are defined as $\mathbf{e}_{ij} = h_\Theta(\mathbf{x}_i, \mathbf{x}_j)$, where h_Θ is a nonlinear function with learnable parameters Θ. The node representations are updated by applying an aggregation operation $\square$ along the set of edges j connected to i:

$$\mathbf{x}_i^{'} = \underset{j:(i,j)\,\epsilon\,E}{\square} h_\Theta(\mathbf{x}_i, \mathbf{x}_j), \tag{3.24}$$

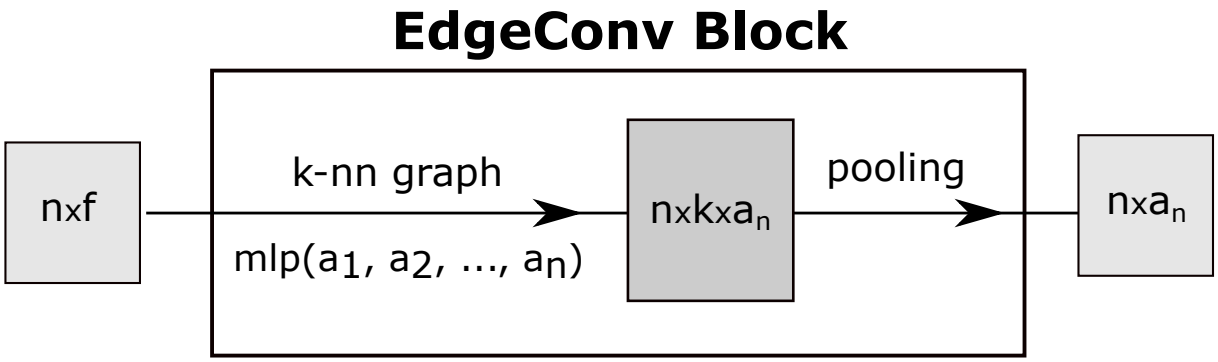


Figure 3.6: Given n f-dimensional input nodes, activations $a_1, ..., a_n$ are computed by applying a shared MLP on the k-nn graph representation of every node, generating an $n * k * a_n$-dimensional tensor. The node representations are updated by pooling the neighbouring learned edge features, generating an output $n * a_n$-dimensional tensor. Figure based on corresponding diagram from Ref. [74].

where E is the set of edges in the graph G: {E, V} [74]. Equation 3.24 is a generic framework which reduces to standard convolution for images when the pixels are defined on a standard grid. In this case, the aggregation scheme is a summation and the filter operation is a convolution over a pixel patch $h_\Theta = \theta_m \cdot \mathbf{x}_j$. The choice of aggregation and filter operations vary for different network models. The filter mechanism in particular determines which kind of information is captured from the graph. For example, the Pointnet model applies an identical operation $h_\Theta = h(\mathbf{x}_i)$ on every node irrespective of its neighbourhood, thus learning global patterns but losing local geometric information [74, 75]. Another choice of filter is $h_\Theta = h(\mathbf{x}_j - \mathbf{x}_i)$, in this case learning local geometric patterns but losing global structural data.

The EdgeConv filter operates over individual nodes and over local node neighbourhoods, as follows:

$$h_\Theta(\mathbf{x}_i, \mathbf{x}_j) = h(\mathbf{x}_i, \mathbf{x}_j - \mathbf{x}_i). \tag{3.25}$$

A single edge feature is then computed according to Eq. 3.26, which introduces two sets of parameters which may be learned using a shared MLP:

$$\mathbf{e}'_{ijm} = RELU(\theta_m \cdot \mathbf{x}_i, \ \ \phi_m \cdot \mathbf{x}_j - \mathbf{x}_i). \tag{3.26}$$

Finally, the edge features may be aggregated together to update the given node representation given an aggregate scheme $\square$ as follows:

$$\mathbf{x}_{im}^{'} = \underset{j:(i,j)\,\epsilon\,E}{\square} \mathbf{e}_{ijm}^{'}. \tag{3.27}$$

A common aggregation choice is the maximum value. By applying the filter on both $\mathbf{x}_i$ and $\mathbf{x}_j - \mathbf{x}_i$, the DGCNN model is able to learn patterns of both the local neighbourhood structure and overall global shape of the graph. In addition, the dynamic recomputation of the graph for every EdgeConv layer allows for groupings of nodes in semantic space compared to the fixed spatial input space. This allows for a diffusion of information throughout the entire graph.

Chapter 4

Datasets and Likelihood

4.1 Data Simulation

The data used in this research was simulated using WCSim (water Cherenkov Simulation) software to generate neutron and background events for the IWCD detector geometry. WCSim was programmed to accurately recreate physics events within large WC detectors [76]. WCSim is based on Geant4 [77] and also depends on ROOT [78]. Geant4 is a robust program for "simulating the passage of particles through matter." Written using object-oriented programming (OOP) in C++, Geant4 has been developed over the past 23 years by an international collaboration of programmers and physicists and is used not only for particle physics but also in medicine, space engineering and other applications. ROOT, also written with OOP in C++, is a toolkit for large scale data analysis and visualization.

Data simulation for this project was performed remotely using the computing resources on Cedar, Simon Fraser University's supercomputer. Cedar is part of WestGrid, a network of advanced computing resources (ARC) throughout western Canada. WestGrid itself is part of the larger computing organization, Compute Canada [79]. The software versions used for simulation included Python 3.6.3, Geant4.10.01.p03, root_v5.34.38 and the nuprism-v2.0.2 branch of WCSim [80]. To set up the simulations, local versions of ROOT, Geant4 and WCSim were installed locally on Cedar and WCSim was compiled into an executable file. To run a WCSim job, various flags were gathered from the command line, including the number of events, particle type, energy and geometry specifications, etc. These runtime flags were used to create a corresponding macro file that was passed to the WCSim executable to run the simulation. After this, the output files, in .ROOT format, were converted to .npz format. These .npz files were then converted into a single HDF5 file (Hierarchical Data Format 5). HDF5 files are compressed and

heterogeneous (can store different types of data), making it a good choice for storing the large and complex simulation output data. Afterward, a script was run to randomly choose indices for the datasets to be divided into training, validation and testing sets. This was done according to an 80%, 10%, 10% distribution.

Within the simulation specifications, WCSim requires the selection of a particular physics list to specify the particles involved and their physical processes. In this case, the FTFP_BERT_HP physics list was used, which includes the FRITIOF string model, Bertini cascade model and the high precision neutron model (energies below 20MeV) [81]. The simulation also included dark noise current in the PMTs at a rate of 1kHz, recreating the current that is often randomly generated by PMTs without incident photons. The minimum hit threshold was set to eight hits to minimize background events with few hits of exclusively random noise. The water was also doped with gadolinium in the simulation at a rate of 0.1% by mass to generate an approximate 90% thermal neutron capture on gadolinium nuclei (see Section 1.4).

The simulation geometry was set to recreate the IWCD, specifying a cylindrical tank with a height of 6 m and a diameter of 8 m. Included to line the walls of the simulated detector were 525 multi-PMT (mPMT) modules of 19 Hamamatsu PMTs each, for a total of 9,975 PMTs in the detector. The particle direction type was also set to cover a 4π solid angle without the tank. Figure 4.1 plots the 3-dimensional positions of all hits over approximately 1.6 million events to illustrate the geometry of the tank and the positions of the mPMT modules. Within each mPMT, the individual PMTs are also somewhat visible by zooming in to the figure.

Three main datasets were used for evaluative purposes in this research. For each, the dataset consisted of roughly 1.6 million events in total divided nearly evenly between neutron capture and background electron events. While the neutron capture simulation parameters were identical for each dataset, the electron background energy distributions were different each time. With the 'highE' dataset, the background electron radiation energy was set according to a uniform energy distribution from 0-20 MeV. For the 'lowE' dataset, this energy followed a 0-8 MeV uniform distribution and for the 'spallation' dataset, the background electron energies followed a right-skewed distribution from approximately 0 to 16 MeV.

The rationale for varying the background radiation energy scale owes to the outsize effect that total energy has on the event observables. Most notably, this shows up through the number of hits and total charge registered by the PMTs. Higher energy events consistently have greater numbers of hits and charges than

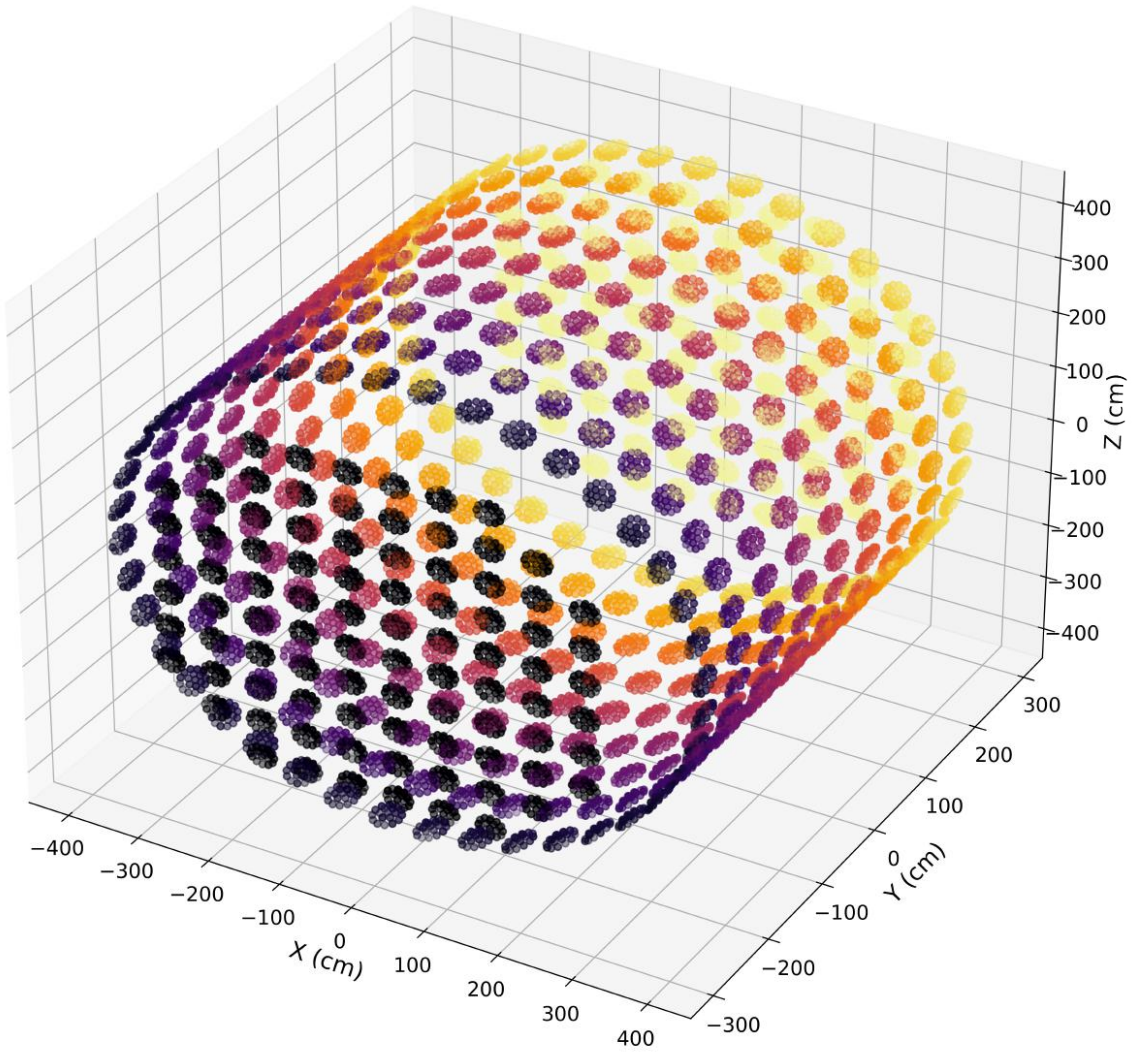
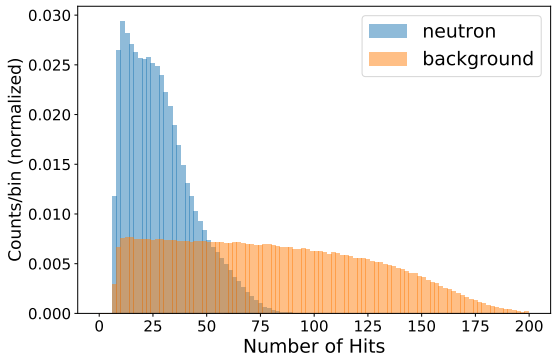


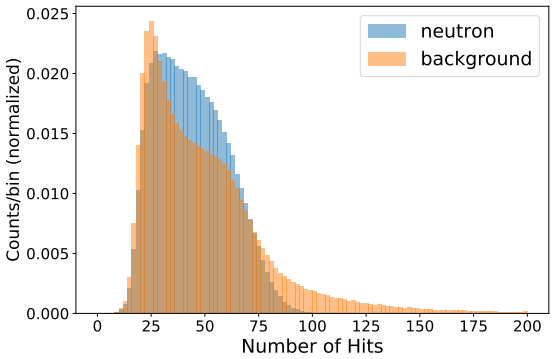
Figure 4.1: IWCD simulation geometry for cylindrical tank of radius 400 cm and height 600 cm, as shown by a scatterplot of the positions of all the individual hit PMTs in an entire dataset of 1.6 million neutron capture and background events. The points are plotted according to Matplotlib's 'inferno' colormap to more clearly outline the geometry. Simulation includes 525 mPMT modules of 19 PMTs each (zoom in to see individual PMTs more clearly).

lower energy events for a given particle interaction type. This can be seen by Figs. 4.2 and 4.3, which show the differences between the highE and lowE datasets for electron background hit totals and charge sums respectively. Since the number of hits and the charge sum of an event is highly correlated, the distribution patterns are quite similar between the two figures. The discrimination extent between hits

and charge sums for the spallation dataset lies somewhere between the lowE and highE datasets (closer to the lowE discrimination) and can be seen in Fig. 6.6 in Section 5.

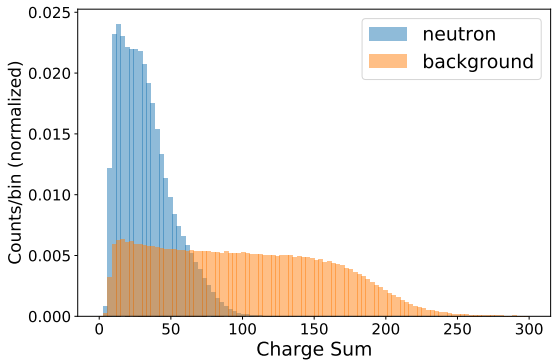


(a) 'highE' dataset, electron background follows uniform 0-20 MeV energy distribution.

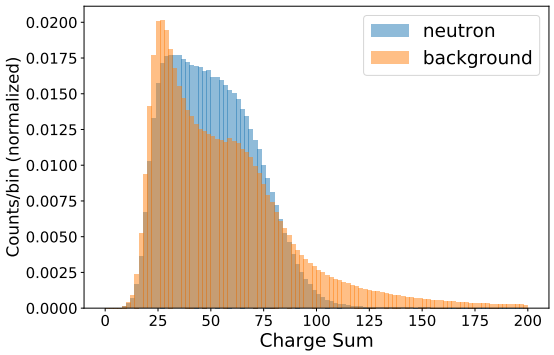


(b) 'lowE' dataset, electron background follows uniform 0-8 MeV energy distribution.

Figure 4.2: The distribution of the number of hits per event (normalized, counts per bin) is shown for the simulated neutron capture (neutron) and electron background (background) radiation events. Higher energies are correlated with higher hit numbers, as can be seen for the electron background generated at higher (a) compared to lower (b) energies.



(a) 'highE' dataset, electron background follows uniform 0-20 MeV energy distribution.



(b) 'lowE' dataset, electron background follows uniform 0-8 MeV energy distribution.

Figure 4.3: The distribution of charge sum per event (normalized, counts per bin) is shown for the simulated neutron capture (neutron) and electron background (background) radiation events. Higher energies are correlated with higher charge sums, as can be seen for the electron background generated at higher (a) compared to lower (b) energies.

The highE dataset, consisting of neutron captures (0-8 MeV) and electron background (0-20 MeV), was the first neutron capture and background dataset available at the start of the research. However, it became clear that good classification performance was already possible for this dataset due to the strong amount of dis-

crimination present between the number of hits and charge sums alone. Section 4.2 evaluates a classification baseline based on hits and charge sums for all datasets. This caused a couple barriers for the development of any machine learning model on the highE dataset. Firstly, it would be difficult to know if the model was mostly learning from the superficial features (number of hits and charge sums), which are not difficult to classify using classic analytic methods, rather than more significant event discriminators like radiation topology and characteristic paths. Secondly, given the strong discrimination already present from hits and charges, it might be difficult to extract significant performance improvements from the baseline analytic approach.

To address these concerns, the 'lowE' background dataset was generated, with the intention of providing a data testbed for which machine learning development can be less influenced by energy differences between particle types, The simulation of background electron events on the same approximate energy range as the neutron captures, 0-8 MeV, led to much more similar distributions in number of hits and charge sums. This may be seen in Figs. 4.2b and 4.3b.

Finally, it was noted that electron background events in real particle detectors do not follow uniform energy distributions, either from 0-8 MeV (like in the lowE dataset) or from 0-20 MeV (as in the highE dataset). Therefore, an effort was made to generate a more realistic approximation of a true background source for the IWCD. In her presentation on muon spallation background in the Super-Kamiokande experiment, Laura Bernard notes that at lower energy scales (tens of MeVs), muon spallation is the dominant source of background [82]. Figure 4.4, retrieved from her talk, demonstrates how the spallation flux (right) is at least an order of magnitude larger than the nuclear radioactive beta decay flux at low energies (left). Figure 4.4 also includes an estimation of counts for diffuse supernova background neutrinos (DSNB). The low relative counts of the DSNB neutrinos, especially compared to the spallation background, helps underscore the necessity of background reduction in WC detectors.

Muon spallation background arises from the production of unstable isotopes through muon interactions. Muons are generated when high energy cosmic rays interact with molecules in Earth's upper atmosphere. Through the breakup, or 'spallation,' of stable nuclei, or through daughter particles produced from secondary muon energy loss processes, muons frequently produce radioactive, unstable isotopes [83]. These unstable isotopes then produce decay products, like electrons, which are a dominant source of background radiation at low energies. Due to the

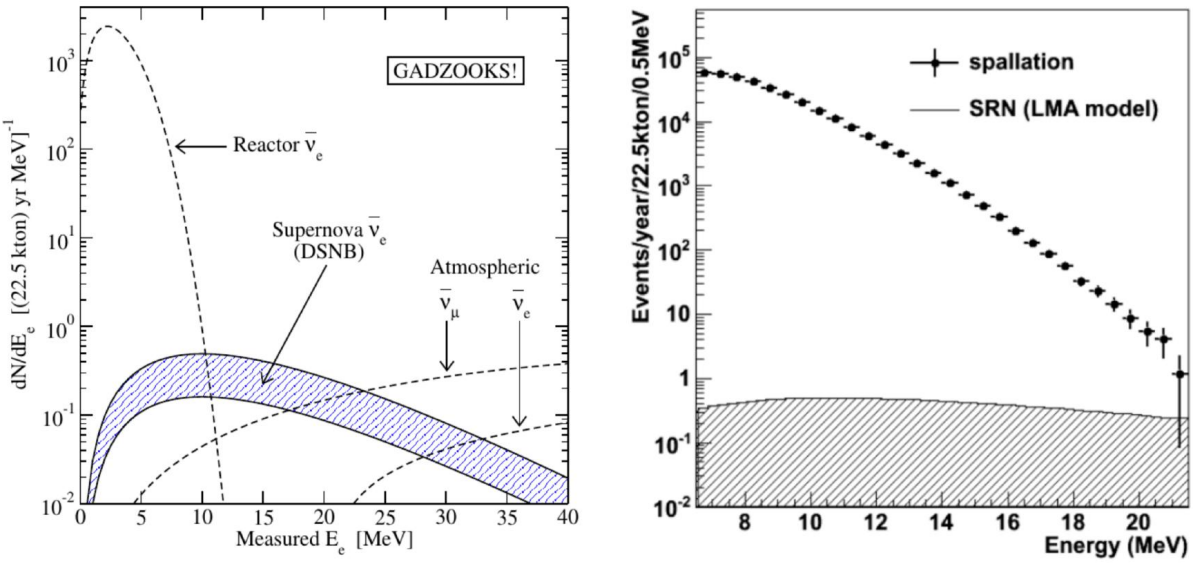


Figure 4.4: Background source flux in the Super-Kamiokande detector compared to the predicted diffuse supernova neutrino background flux (DSNB or SRN, Section 1.4). Reactor and antineutrino background flux (left). Muon spallation background flux (right). Figure retrieved from Ref. [82].

high muon flux at sea level of $6.0 * 10^5$ m^{-2} hr^{-1} [83], the Super-Kamiokande detector was built under 1000 metres of rock. The muons lose energy as they travel through the rock, leading to a far reduced flux rate of 9.6 m^{-2} hr^{-1} at the detector. The IWCD, however, is to be deployed in only a 50 m deep pit (Section 1.3). Therefore, the spallation flux will be greater for IWCD and it is even more important to reduce this background for identifying neutron captures at low energies.

There is a range of radioactive decay patterns from unstable isotopes produced by muon spallation. Every isotope has associated distributions of decay times and energies. Figure 4.5, retrieved from Ref. [82], displays the individual energy spectra for the decay products from muon spallation isotopes like ^{16}N, ^{8}B and ^{8}Li as well as the total energy (blue) for all the decay energies combined. To generate a more realistic background source for the neutron captures, the combined muon spallation energy spectra from Fig. 4.5 (blue) was extracted using the 'WebPlotDigitizer' tool [84] and input to WCSim, replicating the spallation energy distribution for the simulation of electron background radiation events in the IWCD detector. This background, along with the regular neutron capture events generated by WCSim, constituted the third, 'spallation' dataset used in this research. As a final note, although the spallation dataset accounts for more realistic background processes, a true, operational detector background would also include other sources such as "radioactive decays from the surrounding rock, radon contamination in the water,

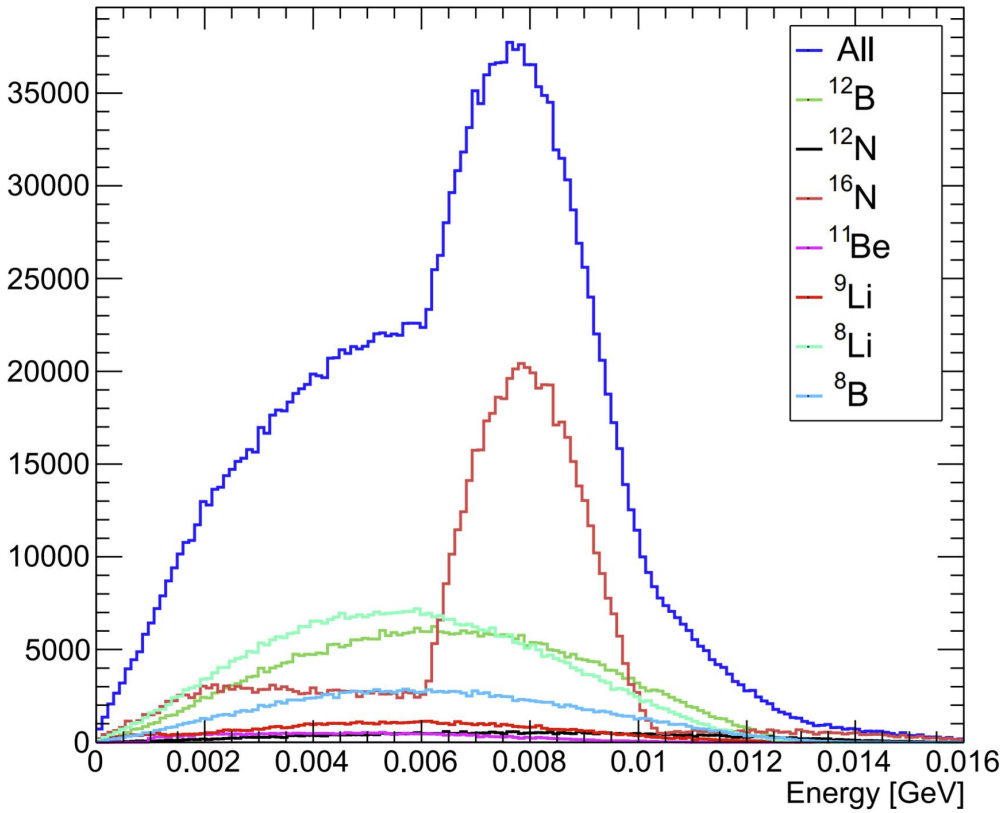


Figure 4.5: Energy distributions for the decay products of unstable isotopes produced in muon spallation processes. The combined energy spectra is shown in blue. Figure retrieved from Ref. [82].

and radioactive contaminants in the tank structure"[85].

Note that in all the simulated datasets, the data is saved in the same 3-dimensional format of (event, hit, features). For every hit in every event, the eight feature values available are the charge, time, 3D position (x, y, z) and 3D orientation (dx, dy, dz) of the hit PMT. Other features may be engineered from these base eight, a topic which is explored in Section 5.

4.2 Likelihood Baseline Analysis

As discussed in the previous section and shown most clearly in Figs. 4.2b and 4.3b, the difference in the total number of hits and charge sums between neutron and background electron events is the most obvious source of separability between these event types. A statistical likelihood analysis based on these features was therefore chosen as a starting point for this project. This likelihood approach aims to set a baseline classification accuracy without using any machine learning tools. The

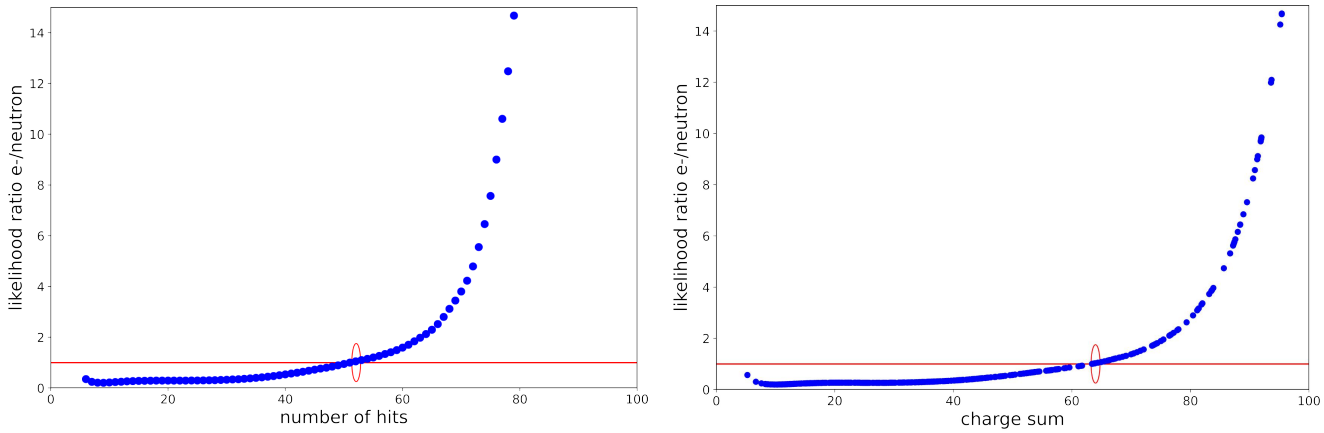
resulting metrics are then used for later comparison against other machine learning models, providing important insight on the true usefulness of the model.

The likelihood baseline classification accuracies for the three datasets were determined by estimating the probability density function (PDF) of the neutron and electron events based on their hit and/or charge sum distributions and then classifying the events based on highest likelihood. Using the 'statsmodels.nonparametric' library in Python, the kernel density estimate (KDE) was calculated using either the 'KDEUnivariate' or 'KDEMultivariate' classes as an estimate of the underlying PDF for the corresponding distribution. The density of the KDE instance, once fit over a distribution of data, was then used to evaluate the event likelihood at a given point.

A univariate and multivariate approach were both applied and the likelihood analysis was carried out depending on the evaluation type. For the 'hits' evaluation type, the univariate KDE was calculated for neutron and electron events over the training set. Then the events in the test set were classified based on highest probability between the neutron hits KDE and the electron hits KDE. For the 'q_sum' (charge sum) evaluation type, an identical process was undertaken, except the univariate KDEs were calculated for the neutron and electron events based on their charge sums over the training events. The final evaluation type, 'q_sum & hits', involved calculation of multivariate KDEs for neutron and electron events on the training set for the combined 2-dimensional distribution of charge sums and number of hits combined. All events in the test set were then classified based on the highest density of the neutron and electron multivariate KDEs for every tuple of event hits and charge sums for that event type.

The above classification process is equivalent to taking the quotient, or ratio, of electron KDE to neutron KDE at a given point, for a given evaluation type, and classifying the event as an electron for any ratio value above one. A ratio below one indicates a neutron event. Figure 4.6 shows the likelihood ratio distributions for univariate classification of events in the 'highE' test dataset (clearest separation of the three datasets) based on event hits (Fig. 4.6a) and charge sums (Fig. 4.6b). The turning point threshold, i.e. the point at which the likelihood ratio passes one and events start to be classified as electron rather than neutron events, occurs around 52 hits for the hits comparison (Fig. 4.6a) and a charge sum of 64 for the charges comparison (Fig. 4.6b). Referencing the corresponding hits and charge sum distributions for the highE dataset in Figs. 4.2a and 4.3a, these threshold values correspond closely to the point at which the electron hit and charge distributions

gain a higher number of event counts. This basic check confirms the likelihood analysis is functioning as expected.



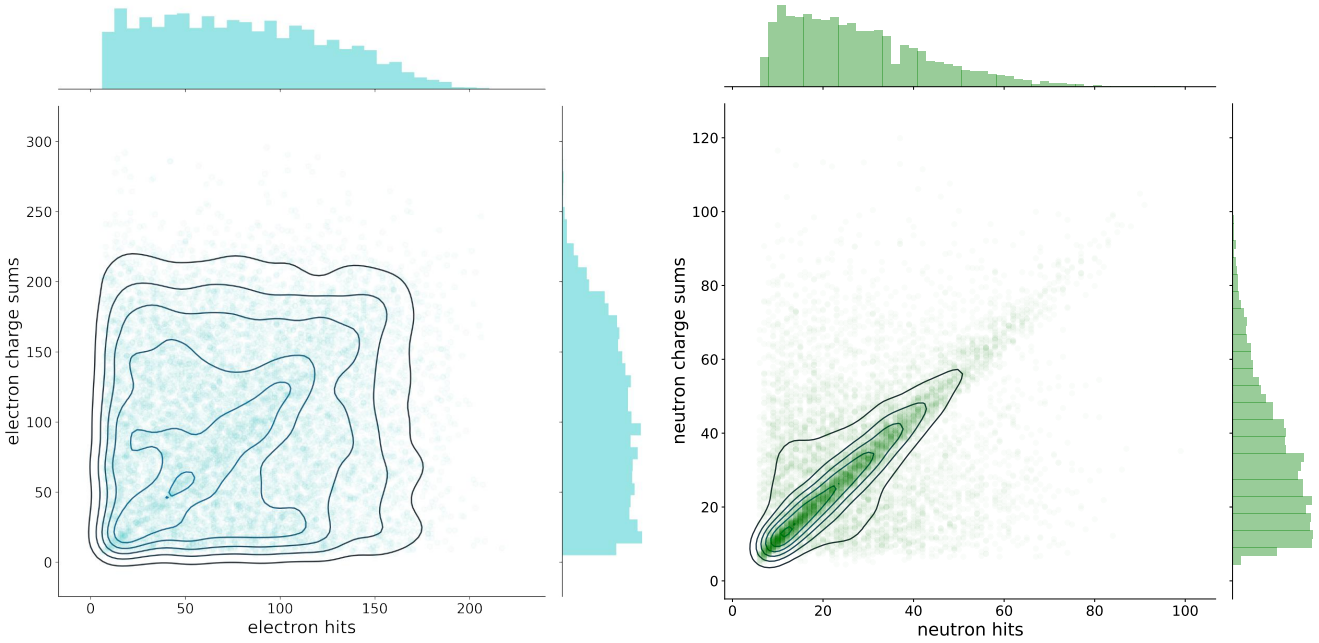
(a) hit likelihood

(b) charge likelihood

Figure 4.6: Likelihood ratio for neutron and electron particle events as a function of event hits (left) and charge sum (right) using highE dataset. Likelihood is computed as the quotient of the electron to neutron KDE probability estimate. Any event with likelihood ratio above one is classified as an electron event, while an event with a ratio value below one is deemed a neutron event. The threshold line y=1 is drawn, indicating the point at which events switch between neutron to electron classification. This occurs at about 52 hits (left) and a charge sum of 64 (right).

Figure 4.7 is presented to help visualize the multivariate likelihood approach. The plots in this figure represent the bivariate hit and charge sum distributions for the electron (Fig. 4.7a) and neutron (Fig. 4.7b) events in the highE dataset. These plots were prepared using the 'kdeplot' function in the Seaborn Python plotting library. Jointly plotted with a large, random subset of the charge and hits data itself (at high transparency), the continuous overlaid lines represent the KDE probability density curves at different density thresholds. In both plots the six foremost density levels, or contours, are plotted. These levels correspond to probability iso-proportions of the data density. For example, the innermost contours outline the areas of greatest probability mass of the data, where data points are likeliest to reside, while the least likely regions of fewest points reside between the outermost contours. Given the significant difference in hit and charge distributions in the highE dataset, it is not surprising that the bivariate plots in Fig. 4.7 are largely dissimilar. The more tightly elliptical shape and linear relationship of the neutron contours show that the neutron events tend to have more similar numbers of hits and charge sums per event, while the electron events have more boxlike contours which indicate many events with few hits and large charge sums and vice versa.

The axis scales also indicate the larger number of hits and charge sums for the electron events in general.



(a) Electron bivariate KDE plot, highE dataset.

(b) Neutron bivariate KDE plot, highE dataset.

Figure 4.7: Bivariate KDE plots of event hits and charge sums for electron and neutron events in the highE dataset. Six density level contours are drawn for each plot. Neutron events tend to have a more linear relationship between hits and charge sums than the electron events, as shown by the more boxlike contours of the electron distribution. The electron events have higher numbers of hits and charge sums in general.

Table 4.1 shows the results of the likelihood classification approach using univariate and multivariate KDEs. Overall, lowest to greatest accuracy was obtained on the low energy, spallation and high energy datasets respectively. This was expected as these datasets had increasing levels of separation between number of hits and charge sums. Evaluation based on the univariate q_sum KDE yielded the top accuracies for each dataset. From here, the goal of the research was to see if machine learning approaches could yield better classification performance than this analytic approach. Of note, the runtime cost of classifying events from highest KDE likelihood was approximately one hour and twenty minutes on average for the approximately 160,000 events of the 10% testing set for either of the three aforementioned neutron capture and electron background datasets. Conversely, fitting the univariate or bivariate KDEs to either of the three training datasets only took a couple minutes.

Background Source	hits (1D)	q_sum (1D)	q_sum & hits (2D)
Spallation	62.4	62.5	62.5
Low Energy	57.4	57.4	54.9
High Energy	78.4	79.8	79.3

Table 4.1: Classification accuracy of neutron capture and background electron events by comparison of highest likelihood for the spallation, low energy and high energy datasets. Evaluation types included highest likelihood of univariate hits KDE (hits), univariate charge sum KDE (q_sum), and multivariate KDE on charge sums and hits combined (q_sum & hits). KDE distributions were fit on the training sets and applied on the test sets.

Chapter 5

Feature Engineering

In machine learning, feature engineering is the process of applying domain knowledge to extract useful features from the original dataset. These features are often more useful than the raw data itself for predictive or analytic tasks. For all three datasets (low energy, high energy and spallation background), application of the XGBoost model on the raw time, charge, hit PMT position and orientation data did not yield significant improvement over the likelihood baseline method.

The application of XGBoost on the original dataset structure requires learning on $nhits * 8$ features (8 features for the time, charge, position (3D) and orientation (3D) of each hit in the event). This many input features (up to 4000) overcomplicated training and did not yield any improvements in performance outcome. This is because the position, orientation and timing information of an individual hit PMT (one of up to 500 hit PMTs in an event) are not by themselves informative in determining the outcome of the event. Rather, it is the relationship of hits within an event, or the aggregation of individual features which is helpful in differentiating between neutron capture and background. For example, given an event with 120 hits, rather than creating 120 features for the charge of each hit, a feature may be constructed representing the aggregate charge of the event.

With this philosophy, a search was conducted for useful features in the domain of neutron capture and subatomic physics. Relevant features were selected to aggregate information from each event, reducing the complexity of the dataset and extracting it into a more useful format. It was found that the classification performance of the XGBoost models significantly improved upon application to the aggregated features compared to the original dataset. The engineered feature plots will be shown for the spallation background dataset, as this is the most realistic of the three datasets, while later plots (this section and appendix) will show the feature differences for all of the datasets.

5.1 Beta Parameters

Useful features in the context of particle classification are those whose numeric distributions vary demonstrably between different types of events. One discriminating physical quantity is the event topology, i.e. the positional distribution of hits throughout the detector for a given event. The event topology varies depending on the particles involved, the incident kinematics and the amount of scattering and reflection through the detector. One way the topology can be numerically quantified is by the amount of anisotropy within the event with respect to the event vertex. For a completely isotropic Cherenkov event, the radiant emission is uniformly distributed over all angles from the position of the vertex. However, for an anisotropic event the direction of radiation is not uniform and there is an inherent directionality. In reality, no single Cherenkov event is perfectly isotropic and the amount of isotropy exists over a range of possible values.

In the case of neutron capture signal to background events, isotropy is a discriminating factor because there is typically more scattering and reflections in background noise compared to neutron capture events. The amount of isotropy difference between signal and background of course depends on the background source. The calculations were performed on the three datasets consisting of neutron captures and electron backgrounds. As described in Section 4.1, the background sources consisted of a low energy source (uniform electron distribution 0-8 MeV), high energy source (uniform electron distribution 0-20 MeV) and a spallation energy distribution ($\sim$0-14 MeV). In all cases, the background source is not entirely typical of realistic background sources in a Cherenkov detector. More realistic sources would include multiple sources of background noise and probably a higher rate of dark noise (random noise hits in the PMTs due to the detectors themselves) than the 0.1% rate used in the simulations. Therefore, the actual isotropy differences would likely vary for real datasets in the IWCD. This factor nonwithstanding, the following results are illustrative of the methodology used for isotropy parameter event discrimination.

There are several possible parameters measuring isotropy that were considered for use in this study. In Ref. [86], Wilson presents the following options: Θ_{ij}, "the average of the angles between each pair of PMT hits in an event with respect to the fitted vertex position," the correlation function ring inner product (CFRIP), which compares the angular correlation of the event to that of a perfect ring, and the beta parameters $\beta(l)$, which are defined similarly to Θ_{ij} but make use of Legendre

polynomials.

Both Wilson [86] and Dunmore [87] find the beta parameters to yield the most isotropic discrimination between different types of subatomic particle events. Following this result, the beta parameters were chosen as the measure of isotropy in this project. The definition for the l-th beta parameter $\beta(l)$ is

$$\beta(l) = \langle P(l)(\cos\theta_{ik})\rangle_{i\neq k}, \tag{5.1}$$

where $\beta(l)$ is equal to the average of the l-th Legendre polynomial $P(l)$ of the cosine of the angle θ_{ik} between every pair of hit PMTs in the event ($i \neq k$) with respect to the event vertex. For any of the beta parameters, a value of 0 indicates perfects isotropy while higher absolute values indicate directionality and lower isotropy. Figure 5.1 shows a representative diagram of the geometry involved in calculating the beta parameters. In practice, this event vertex would need to be calculated using an existing vertex reconstruction method. For the purpose of this study, the truth information is used for the exact event vertex position.

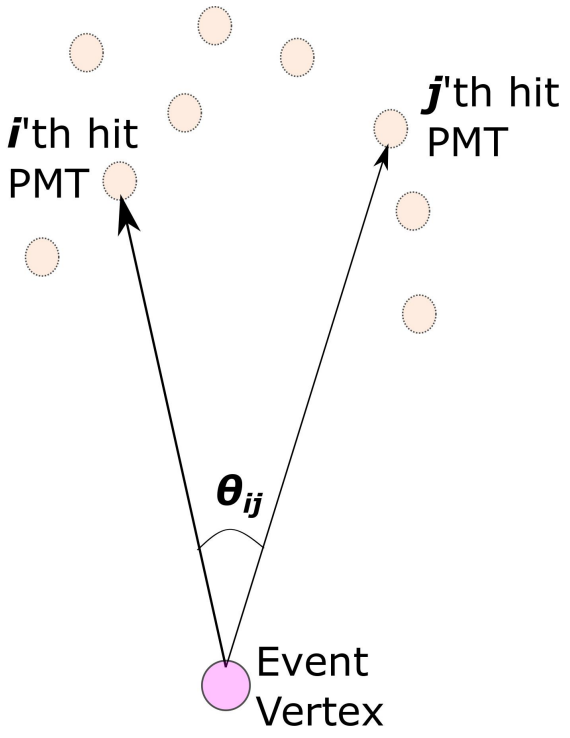


Figure 5.1: A given angle θ_{ij} is presented between the i-th and j-th hit PMTs in an event, relative to the event vertex. A given beta parameter is calculated by averaging the values of the angles between all pairs of hit PMTs within the event and applying the corresponding Legendre polynomial.

The calculation of the beta isotropy parameters is computationally very expen-

sive. A simulated neutron capture event may have over 500 hit PMTs, which adds up to roughly 250,000 pairs of hit PMT combinations. For every combination there are five beta parameters to calculate. Each beta parameter requires the computation of the cosine of the angle between hit PMTs, a Legendre polynomial calculation of that output, and the addition of that value to a running tally. Given approximately 1.6 million events in a full dataset, this translates to several trillion basic mathematic operations in total. The time needed to calculate the total beta parameter set by looping over every event in sequence was estimated to take around 100 days. An effort was made to parallelize a given event using multithreading in CUDA C++, but the many read/write operations and pairwise conditional calculations proved difficult to implement efficiently. Instead, multiprocessing pools were created using the Python multiprocessing library to execute multithreading on the CPU. By utilizing 32 CPU cores on Cedar (see Section 4.1) to calculate batches of 32 events in parallel, and dividing the full dataset computation into twelve simultaneous batch jobs, the full dataset isotropy parameter computation time was reduced to just over ten hours.

The results of the beta parameter computation for the spallation background dataset are shown in Fig. 5.2. The amount of overlap between neutron and background distributions was also calculated for every beta parameter by summing over the minimum of the normalized histogram distribution count between both signal and background distributions over every bin. This overlap number is included in the top-right of every subplot. A higher number indicates more overlap, whereas a lower number indicates greater separability between the distributions. For the spallation dataset, the overlap is highest for the lower order beta parameters $\beta 1$, $\beta 2$ and $\beta 3$, with values of 91.8%, 89.7% and 89.4% respectively. $\beta 4$ and $\beta 5$ show the most discrimination with overlaps of only 77.3% and 78.8% respectively. The isotropy discrimination is also shown for the neutron capture dataset with high energy and low energy backgrounds in the appendix in Figs. 8.1 and 8.6 respectively. For the low energy background dataset, a similar pattern is found to the spallation background dataset, where $\beta 4$ and $\beta 5$ have lower overlaps than $\beta 1$, $\beta 2$ and $\beta 3$, although the magnitude of discrimination is lower due to the nearly identical range between signal and background. The high energy dataset shows least overlap and corresponding greatest discrimination of all the datasets, with $\beta 4$ and $\beta 3$ showing the least overlaps of 66.8% and 68.9% respectively. This indicates the dependence of the isotropy (and topology) of event hits of a given particle type depending on the energy distribution of the events.

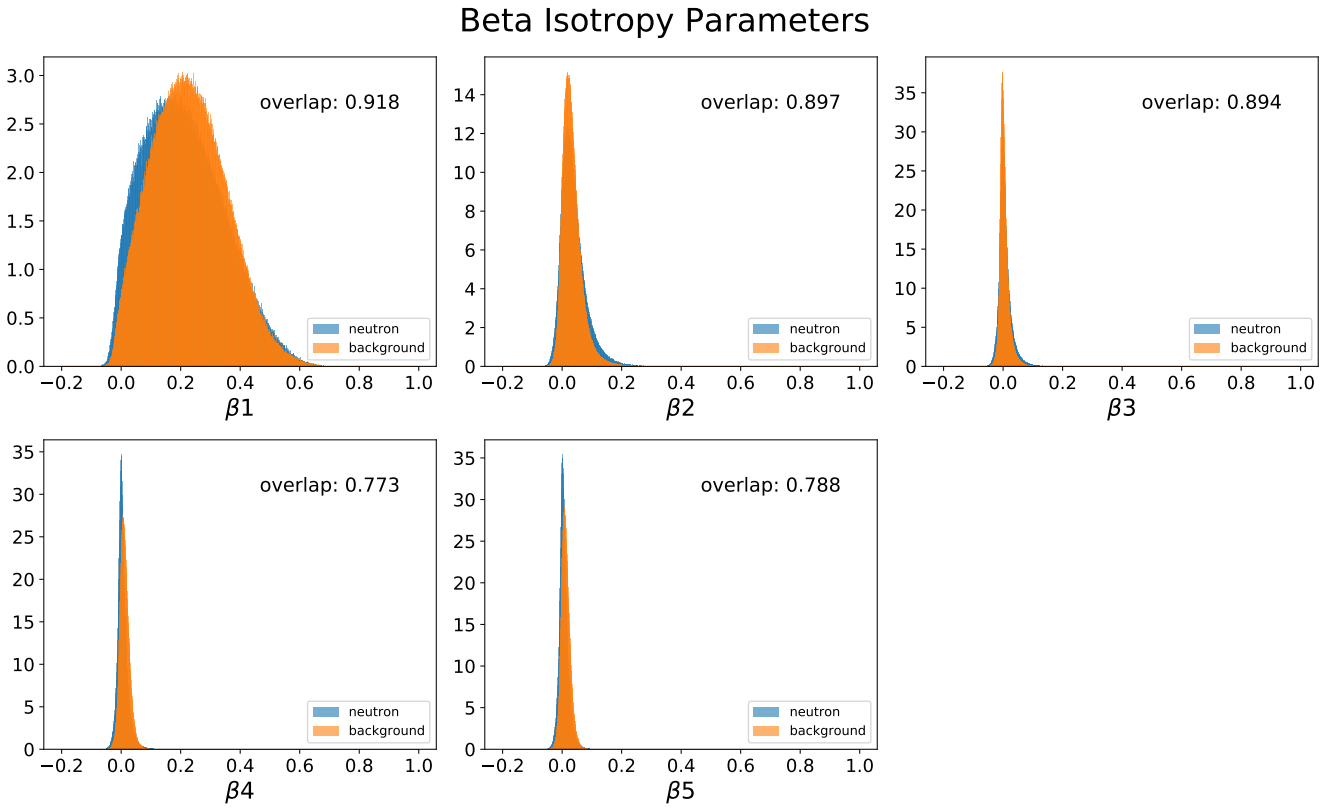


Figure 5.2: Distributions of the beta parameters $\beta 1$ to $\beta 5$ are shown for the neutron capture and spallation background dataset. The amount of overlap between the histograms is shown by the number in the top-right corner of every subplot. $\beta 1$ through $\beta 3$ have the highest overlaps, around 90%, while $\beta 4$ and $\beta 5$ have the lowest overlaps of 77.3% and 78.8% respectively.

5.2 Time of Flight

Timing information within an event contains valuable information. Figure 5.3 represents the recorded times of the first PMT hit (left), final PMT hit (middle) and difference between the first and last hits (right) for the full IWCD dataset of neutron capture and spallation electron background events. The data is separated by event type, where the neutron capture events are shown in magenta and the background is cyan.

Previously in the research, the trained XGBoost model had learned to predict events with over 99% accuracy. When this was investigated further, it was seen that the trigger time of the electrons had a consistently earlier trigger time than the neutron capture events. Therefore, the start time of the electron events had a clear and consistent leftward shift to the neutron capture start time distributions. After this was fixed, the start and end time plots in Fig. 5.3 show that the start

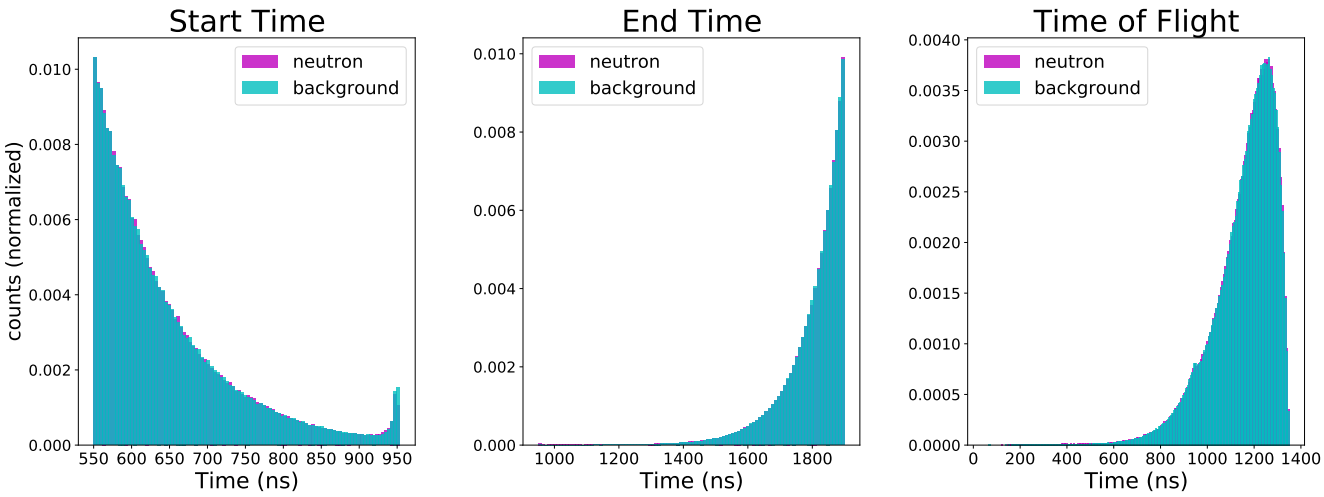


Figure 5.3: Distributions of the timing information within an event are shown for the neutron capture dataset with spallation electron background. The timing is shown for the distributions of first PMT hits (left), final PMT hits (middle) and overall time of flight (right) per event. The neutron capture events are shown in magenta while background events are cyan. The overlap is nearly complete for these distributions, showing these distributions are potentially not useful for discrimination tasks between event types. Moreover, these metrics are highly susceptible to dark noise hits misrepresenting the timing of the first and last hits.

or end time is not a clear discriminating feature. These plots show the absence of a timing simulation artifact.

The raw timing information should not be helpful for event discrimination for the methodology of improving machine learning neutron tagging methods. Rather, the models should learn from characteristics of the event types themselves. The differences between signal and background timing are minute for the distributions of start time, end time and time of flight. This is also seen for the other two datasets of low and high energy electron background. Moreover, these timing metrics are also highly sensitive to dark noise. In reality, false recorded hits before or after the true event due to dark noise are likely to occur, skewing the statistics.

To account for this, the RMS (root-mean-square) time can be used as an engineered feature to measure the average timing residuals of PMT hits from the mean time for a given event. In this case, the RMS time is calculated for a given event as the square root of the sum of the squared differences of every hit time from the average hit time per event, averaged over the number of hits for that event:

$$t_{RMS}(x) = \sqrt{\frac{\sum_{i=1}^{N(x)}(t_i(x) - t_{\mu(x)})^2}{N(x)}}, \tag{5.2}$$

where i is an individual hit within the event x, $N(x)$ is the number of hits in event x, t_i is the recorded time of hit i and $t_{\mu(x)}$ is the average hit time for the event.

In addition to having greater resistance to dark noise fluctuations, Fig. 5.4 shows how the RMS event time metric has greater discrimination between signal and background for the dataset of neutron capture and spallation events. The distribution of neutron capture events has a smaller kurtosis than the background distribution, showing a greater frequency of RMS event times skewed toward the lowest and highest values. This behaviour may be attributed to the differing nature of the particle interactions. For neutron capture events, the photon that emerges from the capture vertex produces an electron positron pair at opposite angles (180 degrees of separation). This particle pair will travel in opposite directions, generating Cherenkov radiation, until hitting the PMTs lining the detector walls and exiting the detector. Depending on the event vertex location, there is significant variation between the time required for both particles to strike the wall. In general, this variation causes the RMS event time to assume a flatter distribution and wider range than the single-particle electron background events. For the spallation dataset, the RMS event time distribution was peaked at approximately 200 ns for background electron events and 225 ns for neutron events.

For the high energy dataset (feature differences shown in Fig. 8.2) the neutron capture events also have longer RMS event times on average than the background events. However, the neutron events themselves still have lower RMS times and a different RMS timing distribution than both the spallation and low energy background events. This is due to a small change in the simulation process. For the spallation and low energy background datasets, the gamma products of the neutron capture are re-simulated at the centre of the detector. This was done to avoid too many events being thrown out due to gamma byproducts being generated too close to the detector walls and then exiting the tank. Although the distributions look different, the similar overlap pattern to the spallation dataset leads to minimal difference in training. For the dataset of neutron captures and low energy background events (Fig. 8.7), the electron background distribution has a more widespread distribution. It generally shows more overlap with the neutron capture RMS timing distribution, indicating that the particle RMS timing distributions are more sharply peaked at higher energies.

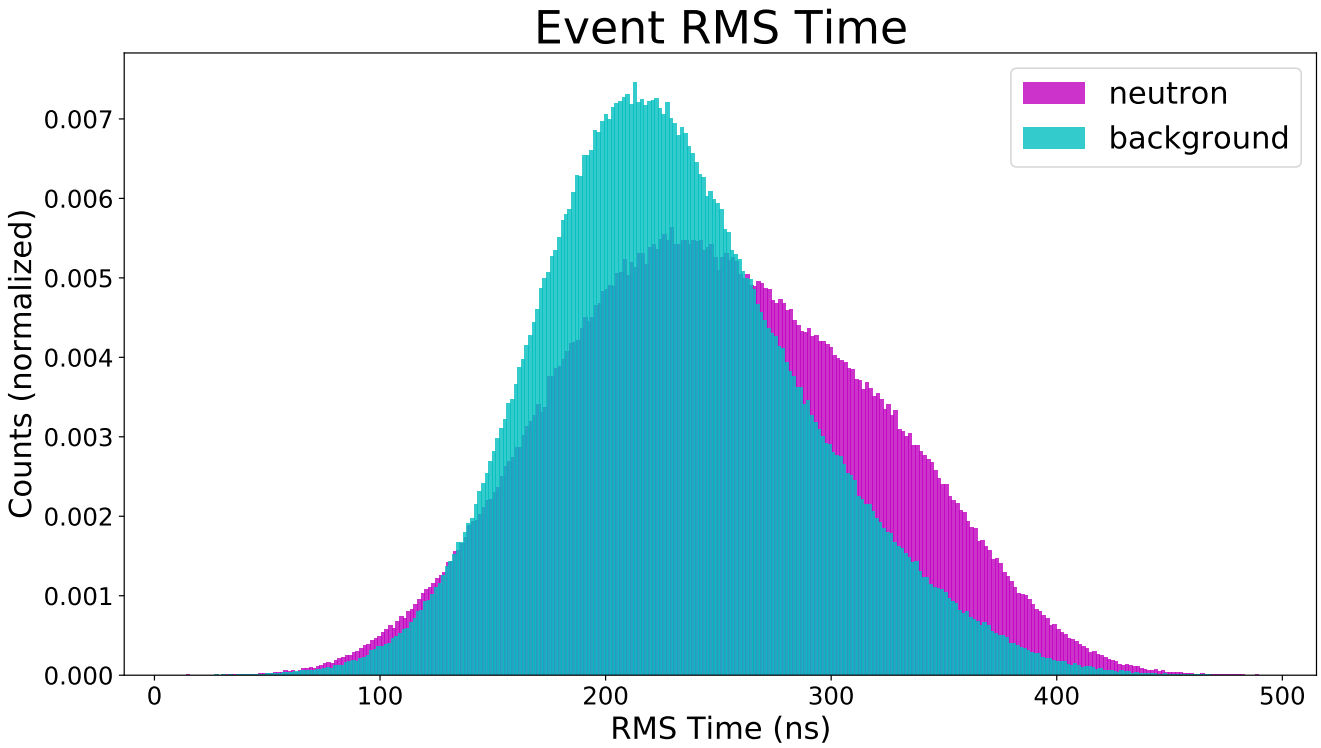


Figure 5.4: Distributions of the RMS flight times are shown per event for both neutron captures (magenta) and electron background (cyan) for the simulated IWCD dataset of neutron capture and spallation electron background events. The RMS time, calculated per event as the square root of the sum of the squared differences of every hit time from the average hit time per event, averaged over the number of hits for that event, shows a greater discrimination between the event types than the previously shown timing metrics. In addition, the RMS timing is more resistant to timing bias due to a dark noise hit before or after the event occurs.

5.3 Distance to Wall

The distribution of event vertex distance to the IWCD cylindrical tank wall was also explored as a potential discriminating feature between neutron capture and background events. For an underground water Cherenkov detector such as Super-Kamiokande, which is located approximately one kilometer underground, a greater number of background events originate at positions nearer the detector walls due to radiation from the surrounding cave. While this effect would be less pronounced for the IWCD, which is not nearly so far underground as SK, there are other effects which may skew the distribution of vertices throughout the detector. For example, dark noise current in the PMTs may be misconstrued as background events.

Figure 5.5 represents the distributions of distances from event vertex to tank wall for the IWCD neutron capture and spallation electron simulated dataset of

approximately 1.6 million events. A nearly identical pattern is observable for both the datasets consisting of low and high energy background (appendix Figs. 8.2 and 8.7). The true vertex position is used for these calculations. Although the distributions overlap greatly from approximately 50 cm to 300 cm, there are noticeably more background than neutron events in the region of 0 to 50 cm from the detector wall. Given that the present simulation included a 0.1 kHz frequency of dark noise, the higher rate of background events generated at lesser distances to the tank wall could possibly be attributed to the dark noise current in the PMTs generating electrons that travel short distances. Conversely, there is a slightly greater occurrence of neutron capture events in the region of 50 cm to 300 cm from the tank wall.

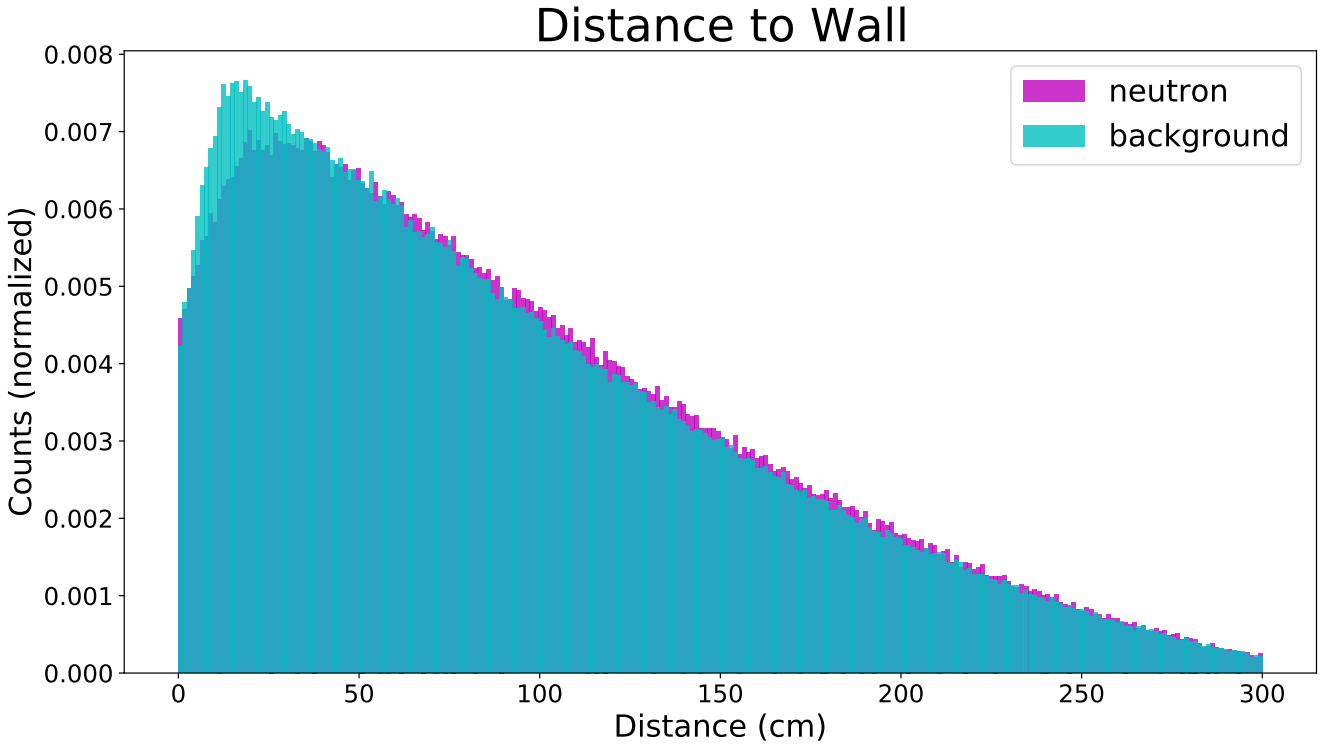


Figure 5.5: Distributions are shown for the distances from event vertex to the IWCD tank wall for the IWCD simulated dataset of neutron capture (magenta) and spallation background events (cyan). The higher incidence of background events generated more closely to the distance wall is likely due to the presence of dark noise in the dataset simulation which misconstrues the random current as true hit PMTs occurring close to the tank wall. This effect causes a slightly greater frequency of neutron capture events generated 50 cm to 300 cm from the tank wall.

Another possibility involves the conversion length of the neutron captures. After the simulated neutron is captured by the hydrogen or gadolinium nucleus, the resultant gamma particles travel some distance ('conversion length') before conversion into an electron positron pair. If these photons escape the outer detector

before conversion, the event is discarded by the simulation software. This effect would lead to slightly more electron than neutron events counted with vertices close to the detector wall. To account for this potential bias in the dataset, the simulation was re-calibrated such that after the neutron capture occurs, the gamma byproducts were re-simulated in the center of the tank.

5.4 Mean Opening Angle

The Cherenkov emission from relativistic photons in water is emitted on a cone with respect to the origin of radiation. The angle of emission is dependent on the kinematic properties of the incident charged particles. The mean opening angle from the event vertex varies on average for different types of particle interactions, making this metric another possible discriminant to improve neutron tagging performance. Following the definition in Ref. [88], this mean opening angle is calculated as the vector sum of the angles between every hit PMT and the true vertex position within the given event:

$$\Theta_\mu(x) = \frac{\sum_{i=1}^{N(x)} \Theta(\vec{p_i}, \vec{p_0})}{N(x)} = \frac{\sum_{i=1}^{N(x)} (\frac{\vec{p_i} \cdot \vec{p_0}}{|\vec{p_i}| \cdot ||\vec{p_0}|})}{N(x)}, \tag{5.3}$$

where $\Theta_\mu(x)$ is the mean opening angle for the event x, $N(x)$ is the number of hits, $\vec{p_i}$ is the (x, y, z) position of the i-th hit, $\vec{p_0}$ is the (x, y, z) position of the event vertex and $\Theta(\vec{p_i}, \vec{p_0})$ is the angle between $\vec{p_i}$ and $\vec{p_0}$, computed as the quotient of the dot product by the product of their magnitudes. The mean opening angle distributions for the neutron capture and spallation electron background events are represented in Fig. 5.6. Refer to Figs. 8.2 and 8.7 for the mean opening angle distribution for the datasets including high energy and low energy background, respectively.

The mean opening angle metric is largely influenced by the event energy. For the dataset with low energy background, the distributions are very similar for the mean opening angle. However, for the spallation background and high energy electron background datasets, a greater amount of discrimination is observed between the distributions. This is due to a combination between the event energy and topological distribution of the hits throughout the event. For example, although the electron events in the high energy background dataset have higher energies, they tend to have more hits closely together, leading to a reduced mean opening angle. Comparatively, the electron events in the spallation background dataset have lower energies but the hits are more sparsely distributed, leading to a higher peak mean

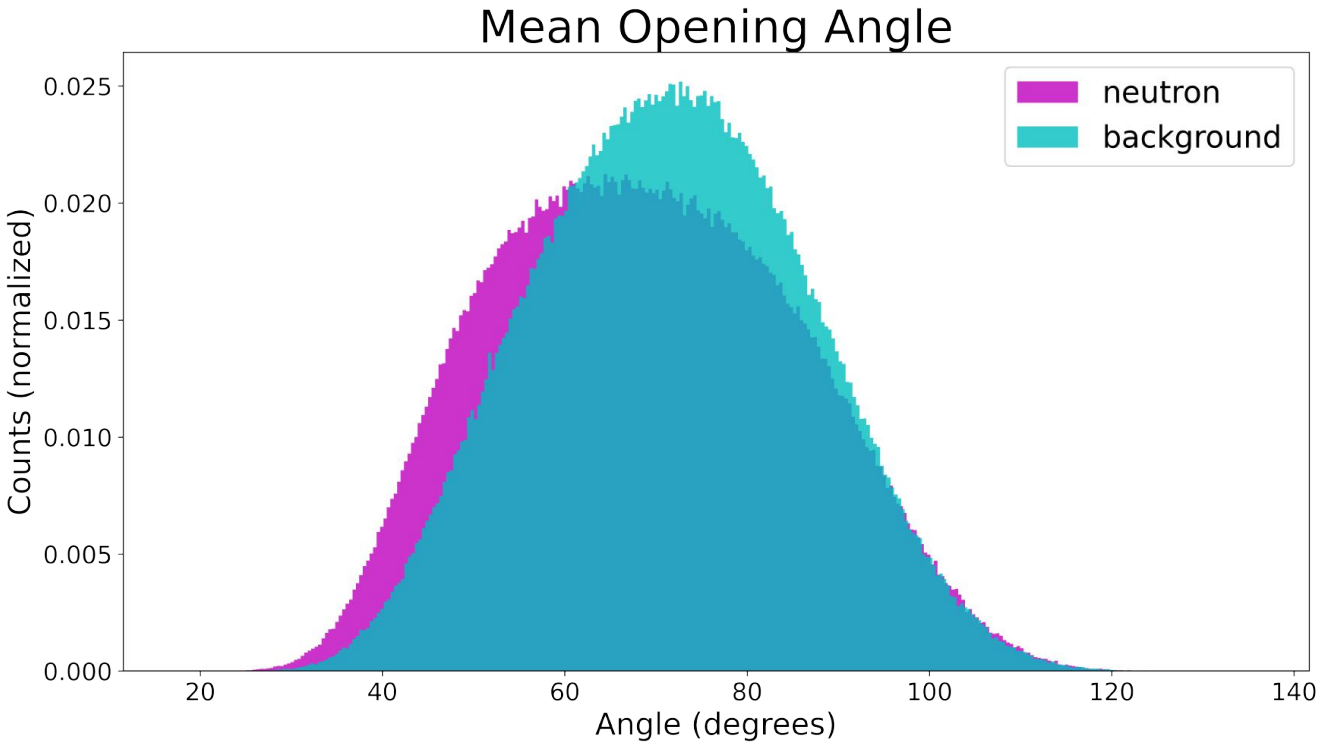


Figure 5.6: Mean opening angle distributions are shown for the IWCD simulated dataset of neutron capture (magenta) and background spallation electron events (cyan). The mean opening angle is dependent on the energy and the topological distribution of hits throughout the event.

opening angle than the neutron events in the dataset, on average. Another factor is the extent of random PMT hits due to true background noise. These dark noise hits tend to cluster in specific regions of the detector tank, causing a lower mean opening angle for such events.

Overall, the ability of the mean opening angle metric to discriminate somewhat between similar particle event types makes it a good candidate to pass to machine learning models as an input feature in neutron tagging tasks.

5.5 Consecutive Hit Angular RMS

Another metric for improving neutron tagging performance in machine learning, again inspired by Abe et al. [88], is the root-mean-squared consecutive angle of an event. True background hits, for example from radioactive background sources, often contain spatially compact clusters of hits. On the other hand, Cherenkov photons from neutron capture events would be expected to propagate more uniformly within the average opening angle of the radiation emission cone. The RMS

difference of angle between temporally consecutive hits with respect to the event vertex position can extract information on these kinds of angular differences between event types.

The RMS angle is calculated by first sorting all PMT hits chronologically within a given event, then computing the sum of the squared differences of the angles between consecutive events from the mean consecutive angular difference, averaged over the number of hits for the event and square rooted, as

$$\Theta_{RMS}(x) = \sqrt{\frac{\sum_{i=1}^{N(x)-1}(\Theta(\vec{p_i}, \vec{p_{i+1}}) - \Theta_\mu)^2}{N(x)}} = \sqrt{\frac{\sum_{i=1}^{N(x)-1}(\frac{\vec{p_i}\cdot\vec{p_{i+1}}}{|\vec{p_i}|\cdot|\vec{p_{i+1}}|} - \Theta_\mu)^2}{N(x)}}, \quad (5.4)$$

where $\Theta_{RMS}(x)$ is the RMS consecutive angle for the event x, $N(x)$ is the number of hits, $\vec{p_i}$ is the (x, y, z) position of the i-th hit, $\vec{p_{i+1}}$ is the (x, y, z) position of next consecutive hit in time order i+1, Θ_μ is the average angle between consecutive hits in the event and $\Theta(\vec{p_i}, \vec{p_{i+1}})$ is the angle between $\vec{p_i}$ and $\vec{p_{i+1}}$, computed as the quotient of the dot product by the product of their magnitudes. The consecutive RMS provides a benchmark on the amount of angular variation between consecutive hits within an event. For events with more scattering, clustering and reflections, the distributions of RMS consecutive angles will be higher on average, and vice versa.

The similar angular distribution patterns between the particles involved in the neutron capture and electron background events lead to a similar distribution of RMS angles for the spallation (Fig. 5.7), high (Fig. 8.2) and low energy background (Fig. 8.7) and neutron capture datasets. This is because the WCSim simulation software treats the kinematics of the electron and position pair from the gamma produced by the neutron capture similarly to the electron produced on its own. Moreover, as mentioned previously in Section 5.4, the low dark noise rate of the simulated background event is not representative of radioactive decays and other more realistic sources of WC background. These sources would lead to higher frequencies of dense hit clustering within the detector. For a background source more inclusive of clustering, the discrimination extent is expected to be greater for the RMS metric, meriting inclusion into training as a potentially discriminating feature.

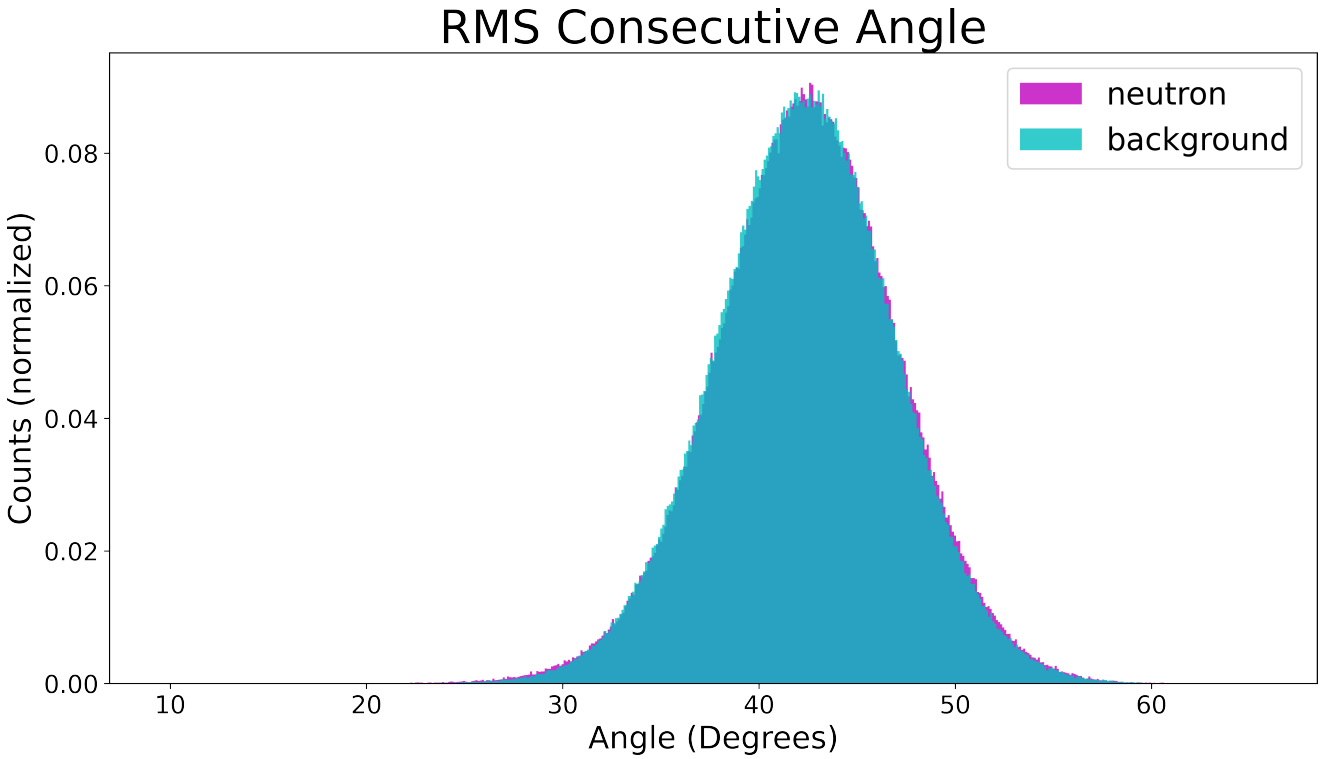


Figure 5.7: Distributions of the RMS angle between consecutive hits within an event are shown for neutron capture events (magenta) and background electron events (cyan) for the full IWCD neutron capture and spallation electron background dataset. The high amount of overlap between distributions is caused by the similar kinematics involved in the WCSim simulation software for the neutron capture and electron events. Background events with greater rates of dark noise and clustering will increase the RMS angle distribution, leading to greater discrimination between neutron capture events for this metric.

5.6 Consecutive Hit Distance

In studying the event displays of the neutron capture and background events for the IWCD dataset of higher energy (0-20MeV), the pattern was observed that the positional distributions of hits tended to be more widespread in neutron capture events. Given two events of different type with similar numbers of hits, the neutron capture event could be reasonably well differentiated by eye by selecting the event with greater average distance between hits. This effect was most pronounced using the high energy electron background dataset. An example of two such events is shown in Fig. 5.8.

Motivated by such examples as shown in Fig. 5.8, the average consecutive hit distance metric was created to capture this effect in a discriminant feature. The hits within a given event were first sorted chronologically in time, then the Euclidean

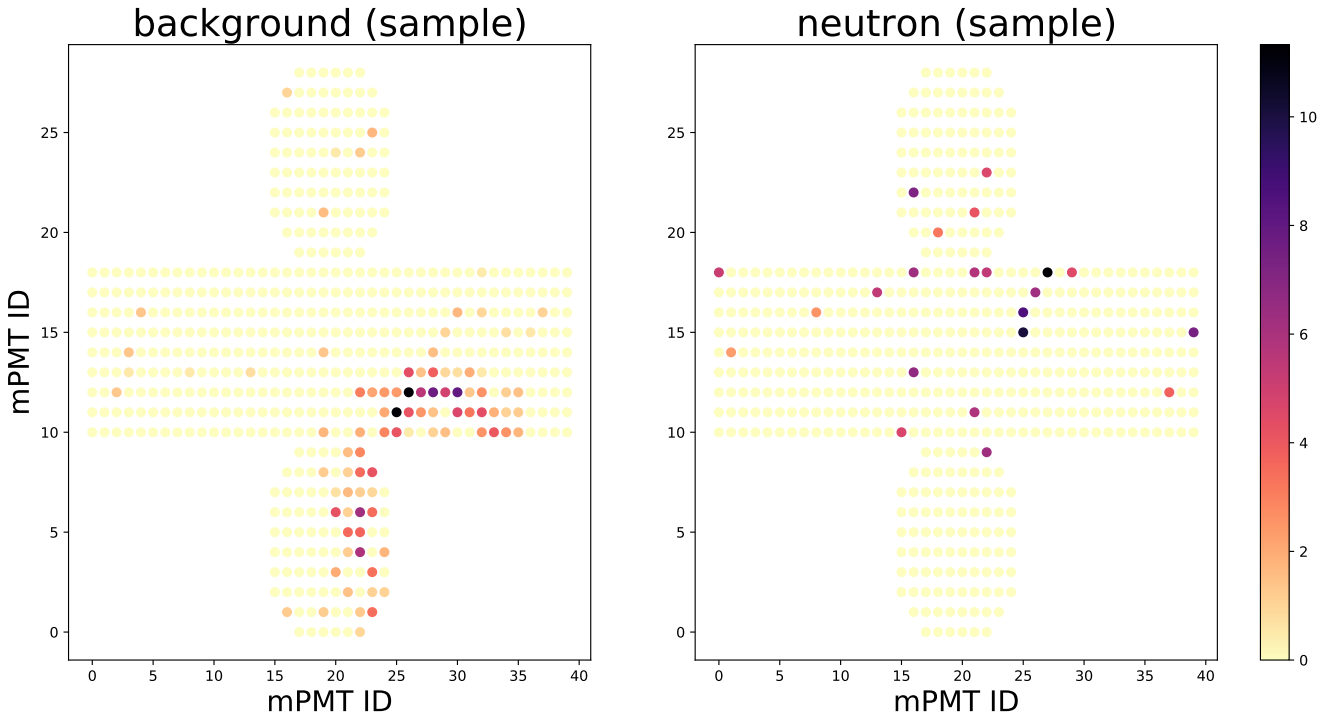


Figure 5.8: Two events are shown, sampled from the full IWCD high energy background (0-20MeV) neutron capture dataset. The background electron event (left) is representative of the closer spacing of electron background hits compared to the generally more diffuse hit patterns of neutron capture events (right).

distance between consecutive hits were summed and averaged over the number of hits within the event as

$$hd_{\mu}(x) = \frac{\sum_{i=1}^{N(x)-1} dist(\vec{p_i}, \vec{p_{i+1}})}{N(x)} = \frac{\sum_{i=1}^{N(x)-1} (p_{x(i)} - p_{x(i+1)})^2 + (p_{y(i)} - p_{y(i+1)})^2 + (p_{z(i)} - p_{z(i+1)})^2}{N(x)}, \tag{5.5}$$

where $hd_{\mu}(x)$ is the average consecutive hit distance for the event x, $N(x)$ is the number of hits, $\vec{p_i}$ is the (x, y, z) position of the i-th hit, $\vec{p_{i+1}}$ is the (x, y, z) position of the next consecutive hit in time order i+1 and $dist(\vec{p_i}, \vec{p_{i+1}})$ is the Euclidean distance between consecutive hits.

The difference of consecutive hit distance was greatest for the spallation background (Fig. 5.9) and high energy background (Fig. 8.2) neutron capture datasets. Compared to these, the difference was less for the dataset including low energy electron background (Fig. 8.7). Therefore it seems that the electron hits are more

clustered at higher energies. However, even with the nearly identical energy range in the low energy dataset, the average hit distance was still slightly higher for the neutron captures. This difference in hit diffusion extent is likely due to due to the differing nature of the particle interactions, in which the pair production of the electron positron pair from the gamma(s) in the neutron capture events leads to greater spatial separation of hits throughout the detector, on average, when compared to the electron background hits.

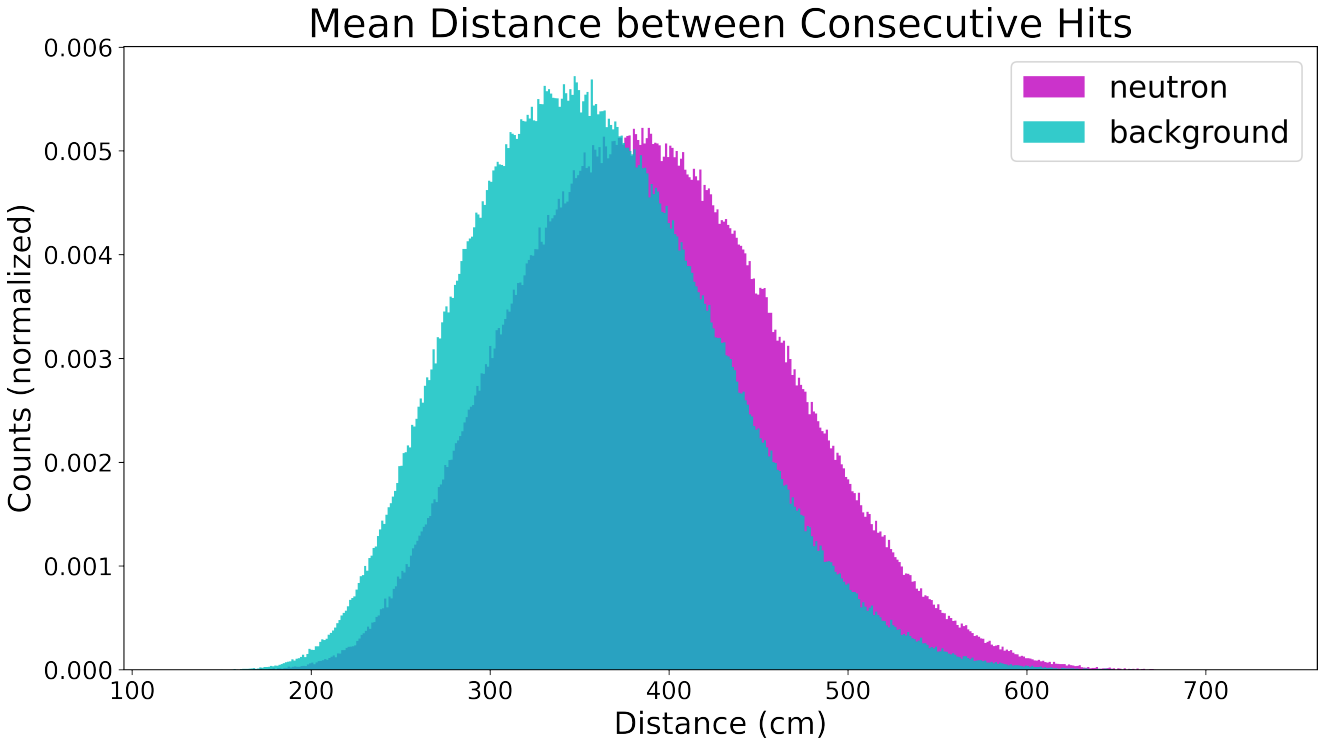


Figure 5.9: Distributions of the mean consecutive hit PMTs distances are shown for neutron capture events (magenta) and background electron events (cyan) for the full IWCD dataset of neutron capture and spallation electron background events. The spacing is more diffuse for the neutron events than the background electron events on average, leading to a distribution peak shifted about 50 cm higher for the average hit distance between consecutive hits for the neutron events.

Chapter 6

XGBoost Results Analysis

The previous section describes the feature engineering process of the neutron capture and background datasets. The compute time for feature engineering is constrained by the calculation of the beta parameters, which takes approximately 10 hours for 10 simultaneous batch jobs submitted to Cedar. However, adding more simultaneous batch jobs would reduce this compute time. The other aggregate features each take from 30 to 45 minutes each to calculate. The resulting features are then loaded from memory and normalized to range from zero to one so that no single feature overpowers another in training due to its distribution having a larger relative numeric scale. After normalization, the features are combined into a single dataframe using the 'pandas' library in python. XGBoost, a gradient boosting decision tree model, was applied to learn from this data. Other models tested include lightGBM (another gradient-boosted decision tree) and a multi-layer perceptron model. In all cases, the relevant parameters and hyperparameters were tuned to reduce bias and variance to try to obtain the most accurate and generalizable model.

The XGBoost gradient boosting decision tree model was applied to the task of learning from the features engineered in Section 5 for all three of the neutron capture and electron background datasets. This model contain a set of parameters which effect the bias and variance of the models. In general, some parameters tend to increase training accuracy while decreasing generalizability to new data, and vice versa. In both cases, the parameters chosen for tuning were heuristically selected by the greatest impact on the bias and variance of the model.

For the XGBoost model, the training parameters selected for tuning include the maximum tree depth *max_depth*, the minimum tree node weight *min_child_weight*, the training data subsampling ratio *subsample*, the tree column subsampling ratio *colsample_bytree* and the learning rate *eta*. The *max_depth* parameter limits the

maximum depth of a given decision tree, denoting the greatest number of nodes from the root node to any leaf following a downward path through the tree. Increasing the maximum tree depth will lower the model bias, i.e. fit the training data more accurately, but may cause higher variance, i.e. overfitting to the training set and losing ability to generalize to unseen datasets. The *min_child_weight* parameter sets the minimum weight needed by a given leaf in a tree instance to further branch and create new child leaves from that node. A larger value of *min_child_weight* will lead to simpler trees, favouring low variance over high bias.

A strategy for increasing generalizability of the XGBoost model is to randomly leave out part of the data for the training instances. The value of the *subsample* parameter sets the fraction of data subsampling for every instance. A value close to one (all data included) leads to lower bias while a lower subsampling ratio may increase model generalizability. Another approach is to randomly exclude certain features, or columns, from training. This way, the model is not overly dependent on any given feature. The *colsample_bytree* parameter sets the subsampling ratio of columns, or features, randomly selected when constructing every tree. A *colsample_bytree* value of one will apply all columns to every tree, while the ratio of features used per tree decreases quadratically as the *colsample_bytree* parameter value is decreased.

Finally, the parameter *eta*, also known as the learning rate, controls the rate at which the model parameters update after every step of gradient descent. The model will tend to learn quickly with large values of *eta* but is likely to miss the best set of parameters associated with the absolute minima of the loss function, whereas a lower learning rate will lead to slower learning which requires more iterations to reach the loss minima.

For both models, a grid search was applied by sequentially iterating over several relating parameters in pairs to find the best combination. This iterative method must be employed as there is no exact protocol to finding the optimal solution of hyperparameters in machine learning. For the grid search, while keeping the learning rate and sampling ratios fixed, the maximum tree depth *max_depth* and minimum tree child weight *min_child_weight* were first adjusted together. Since both hyperparameters control the tree complexity, they may be tuned as an ensemble. XGBoost model instances were trained with *max_depth* ranging from five to fifteen and *min_child_weight* ranging from one to ten. For each hyperparameter combination, XGBoost's native cross-validation function was used to train the model with four folds over a maximum of 1250 boosting rounds. In this way, each

model was trained on a unique set of three dataset folds and reports the average result from holding out a different fourth fold for testing. Early stopping was used to cancel model training if performance did not improve over twenty consecutive rounds.

Applying this technique on the full IWCD dataset of nearly 1.6 million neutron captures and spallation background events, the best parameters for tree complexity were found to have a *max_depth* of 11 and a *min_child_weight* of 1, with the best mean log loss over the four folds of the testing set equal to 0.593 over 443 rounds. This indicates that at least moderately complicated trees are necessary to classify the events in this dataset. These hyperparameters were updated and fixed in the XGBoost model parameter dictionary. Next, the *subsample* and *colsample_bytree* ratio parameters were tuned in tandem, with *subsample* and *colsample_bytree* both ranging from 0.5 to 1.0. In this case, the optimal combination was found to have a *subsample* ratio of 0.7 and a *colsample_bytree* ratio of 1.0, combining to produce the lowest mean log loss of 0.591 over 499 rounds. The subsample ratio indicates that holding out 30% of the data for each training instance helps the model to generalize and avoid overfitting, improving performance. The 1.0 column sample ratio indicates there is little to no redundancy in the twelve engineered features and that all twelve are helpful for training the XGBoost model. Finally, a learning rate of 0.007 was found to lead to highest test accuracies.

The same hyperparameter tuning process was also applied on the other two datasets including low energy and high energy electron background events. After training the XGBoost tree model on these corresponding combinations of parameters obtained through the grid search process, the models were tested against their own 10% test set. The resulting metrics are displayed in the following Table 6.1. Although the training accuracies are generally slightly higher than the test accuracy, the extent of overfitting is not too severe and may be decreased by using a smaller number for early stopping.

Along with the accuracies on the train, validation and test set, the area under the curve (AUC) of the receiver operating characteristic (ROC) is also shown. The ROC curve, in general, is a plot of the true positive rate (the probability that an actual positive, i.e. neutron, will be correctly identified) against the false positive rate (the probability that the model incorrectly predicts a positive). It is useful for evaluating performance at various false positive rate thresholds. The AUC of the ROC plot is another useful metric of separability between the neutron and background classes. An ROC AUC of 0 indicates no predictive ability at all, 0.5

represents a half and half guess and 1.0 indicates perfect predictive separability.

Dataset Background Source	Train	Validation	Test	ROC AUC
Spallation	73.0	71.5	71.4	0.784
Low Energy Electron	68.7	66.3	66.3	0.734
High Energy Electron	84.3	83.8	83.6	0.915

Table 6.1: Results of application of trained XGBoost model on three neutron capture datasets with background sources of spallation, low energy electron and high energy electron events, for a total of nearly 1.6 million events each.

The XGBoost models attained the highest test accuracy of 83.6% on the high energy background dataset, a middle test accuracy of 71.4% on the spallation background dataset and the lower accuracy of 66.3% on the low energy background dataset. The corresponding ROC AUC metrics are 0.915, 0.784 and 0.734. These results are commensurate with the extent of discrimination of the features represented in Figs. 6.6, 8.2 and 8.7 respectively. The more the engineered features vary between event type, the more accuracy the XGBoost model is able to obtain, and vice versa. At the learning rate of 0.007, the XGBoost model construction for any of the 90% training sets was found to take approximately 45 to 60 minutes, depending on the number of trees constructed before early stopping. Evaluation on any of the 10% testing sets took less than a minute.

Figure 6.1 displays the confusion matrix for the XGBoost model trained on the spallation background and neutron capture dataset. This matrix shows that for this model, the true positive rate (neutron recall) is significantly lower than the true negative rate (neutron specificity). This indicates that for the XGBoost model on the spallation background dataset, the electron events are easier to classify than the neutron captures. The same pattern holds for the XGBoost model applied to the low energy dataset, where the electron recall is 67.5% and the neutron recall is only 65.1% (Fig. 8.8). This pattern shifts when the XGBoost model is trained on the high energy dataset. In this last case, it recalls 89.7% of the true neutron events but only recalls 78.1% of the electron background events.

To further analyse the results of the XGBoost model, it is desirable to get a sense of the relative importances of the dozen features contributing to the tree ensemble. The XGBoost python interface provides weight, gain and cover as the three methods available to visualize the feature importances. *Weight* indicates how many times a feature is used to split data across all the trees, *gain* represents the information gained (equivalently, the reduction in training loss) by splitting a tree on the feature and the *cover* shows the weight combined with how many data points

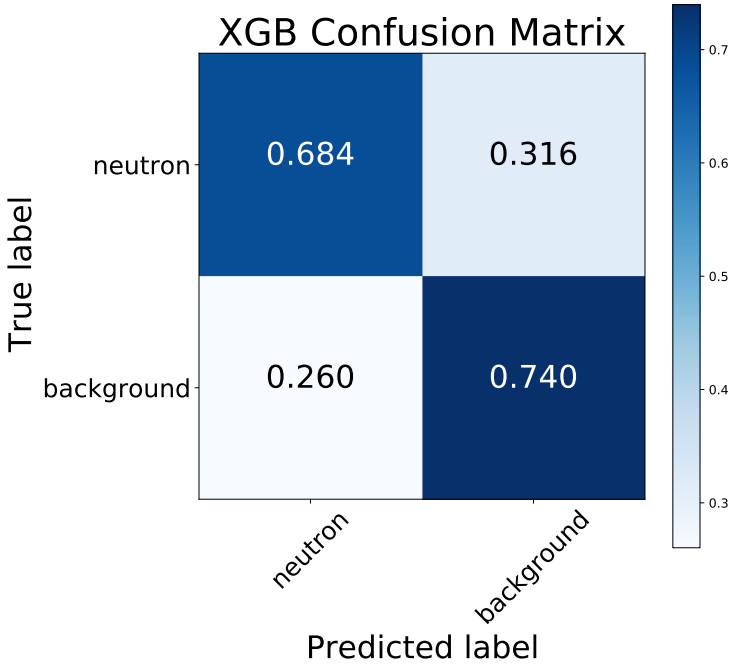
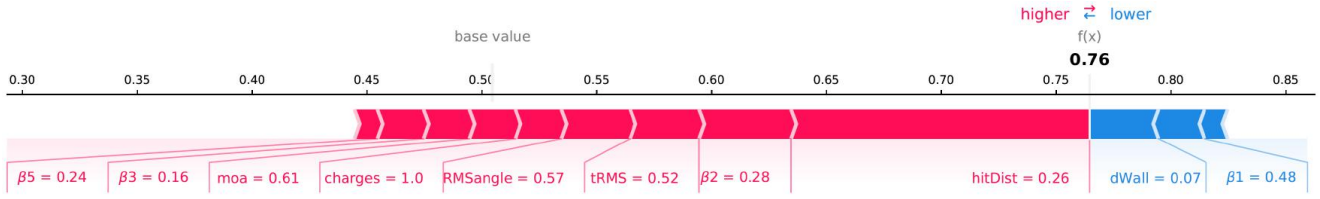


Figure 6.1: Confusion matrix for the XGBoost model trained on the dataset of neutron capture and spallation background electron events. The electron recall rate exceeds the neutron recall rate, indicating the neutron capture events are more difficult to identify correctly.

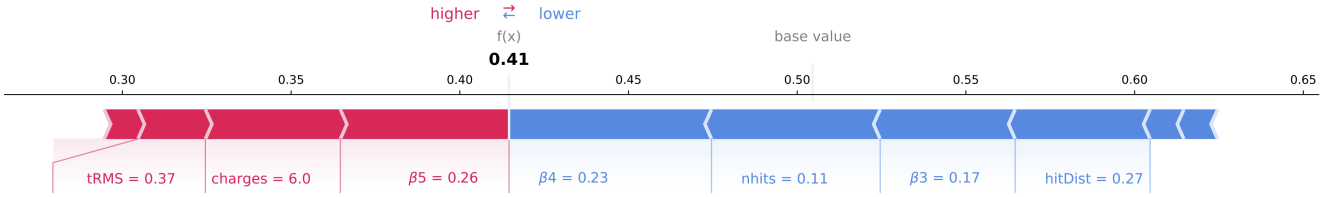
actually travel through those feature splits. Figure 8.11 in the appendix represents the above feature importance metrics for the trained boosted decision tree model for the low energy background neutron capture dataset. The figure, as an example, shows a lack of consistency across the different metrics. For example, the distance to the wall feature *dwall* is the most important feature by the weight metric, but is among the least important under both the cover and gain metrics. Conversely, while the event number of hits *nhits* is the most important feature according to gain, *nhits* is fourth least important with respect to the weight and cover metrics.

Given the inconsistency of the weight, cover and gain metrics, another metric may be applied more reliably to draw conclusions about the relative feature importances of the trained XGBoost models. This metric is the SHAP value, as introduced and discussed in Subsection 3.3. An attractive feature of the SHAP values is that they are applicable both locally, to a single event, and globally, to a conglomerate of events. Figure 6.2 displays the SHAP values for two events chosen at random, demonstrating the local effect (single event) of the various features on 'pushing' the model output from the base value. For the low energy dataset, the base value of 0.504 represents the average output label. It is slightly higher than 0.5

because there are marginally more electron events in the dataset (neutron capture and spallation background). A label below the 0.5 threshold represents a neutron event and a label above 0.5 stands for an electron background event. The feature value is shown to two decimal places below the SHAP value bar.



(a) Local SHAP force plot for a sample predicted electron event.

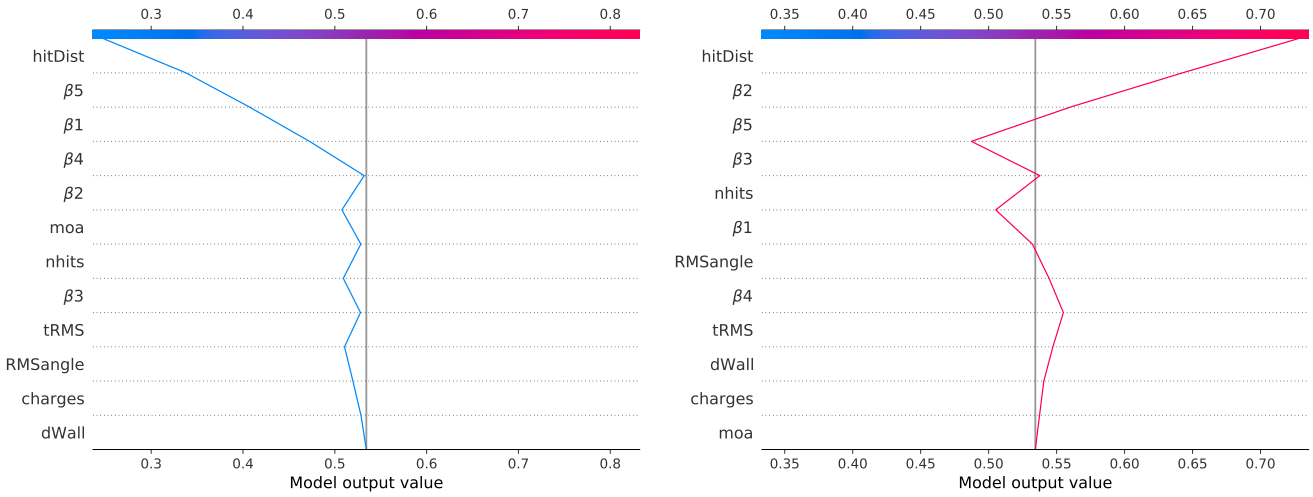


(b) Local SHAP force plot for a sample predicted neutron event.

Figure 6.2: Local force plots are shown for a randomly chosen predicted electron background event (top) and neutron capture event (bottom). The red bars represent features with positive SHAP values driving the local model output toward an electron label and the blue bars represent features with negative SHAP values driving the model output toward a neutron label. The bar width corresponds to the absolute magnitude of the SHAP value. The model output is shown in bold (0.76 for the predicted electron event and 0.41 for the predicted neutron event).

For the sample events in Fig. 6.2, 6.2a (top) shows a predicted electron event and Fig. 6.2b (bottom) shows a predicted neutron event. The model output is 0.76 for the electron event and 0.41 for the neutron event (shown in bold). For the electron event in Fig. 6.2a, the large positive SHAP value of the mean hit distance, as well as other positive SHAP values from $\beta 2$, rms time, and other values (shown in red) drive the output up to 0.76 from the base expectation value. Conversely, the cause of the output value being driven down from expectation toward 0.41 for the neutron event in Fig. 6.2b may be attributed to features with corresponding negative SHAP values for this event (shown in blue), such as the $\beta 4$ value and number of hits. The width of the SHAP bars correspond to the magnitude of the Shapley value for the feature in the event.

An alternative way to visualize the SHAP values locally to understand the marginal feature contributions for a given event is through decision plots. Figure 6.3 displays local decision plots for two more single events, chosen at random.



(a) Decision plot for sample predicted electron event. (b) Decision plot for sample predicted neutron event.

Figure 6.3: Local decision plots are shown for a randomly chosen predicted neutron capture event (left) and background electron event (right). Starting at the bottom from the base expectation model output of 0.504, the model output changes upward in the line plot according to the SHAP value of every feature. The marginal contributions of the features adjust the model output until the final output at the top of the plot.

The decision plot again orders the features by the magnitude of their SHAP value, and is meant to be read from the bottom up. The horizontal axis represents the model output value. At the bottom of the plot, the model output is the base expectation of the model. As the line plot ascends upward, the output adjustment impact of each feature may be viewed through the horizontal change in the output value. For example, in Fig. 6.3a, the output value remains relatively close to 0.5 until the hit distance, $\beta 5$ and $\beta 4$ combine to change the model output to distinctly electron-like. In Fig. 6.3b, the charge and $\beta 5$ value tend to push the model output toward an electron-like event, but features such as $\beta 3$, the number of hits and the $\beta 4$ value for this event cause the model output to be neutron-like.

Figure 6.3 provides another local visual representation of SHAP values. However, the SHAP values may also be visualized globally for many events. This helps give a sense of the overall patterns influencing the feature contributions for large numbers of particle events. One method of representing the SHAP values for many events is to superimpose the individual decision plots of several events onto the same graph. Figure 6.4 shows the decision plots for 500 events chosen at random (for the combined decision plot, thousands of events tends to clutter the graph and the marginal contribution patterns are more difficult to discern).

Such global visualization methods help reveal patterns in the feature contribu-

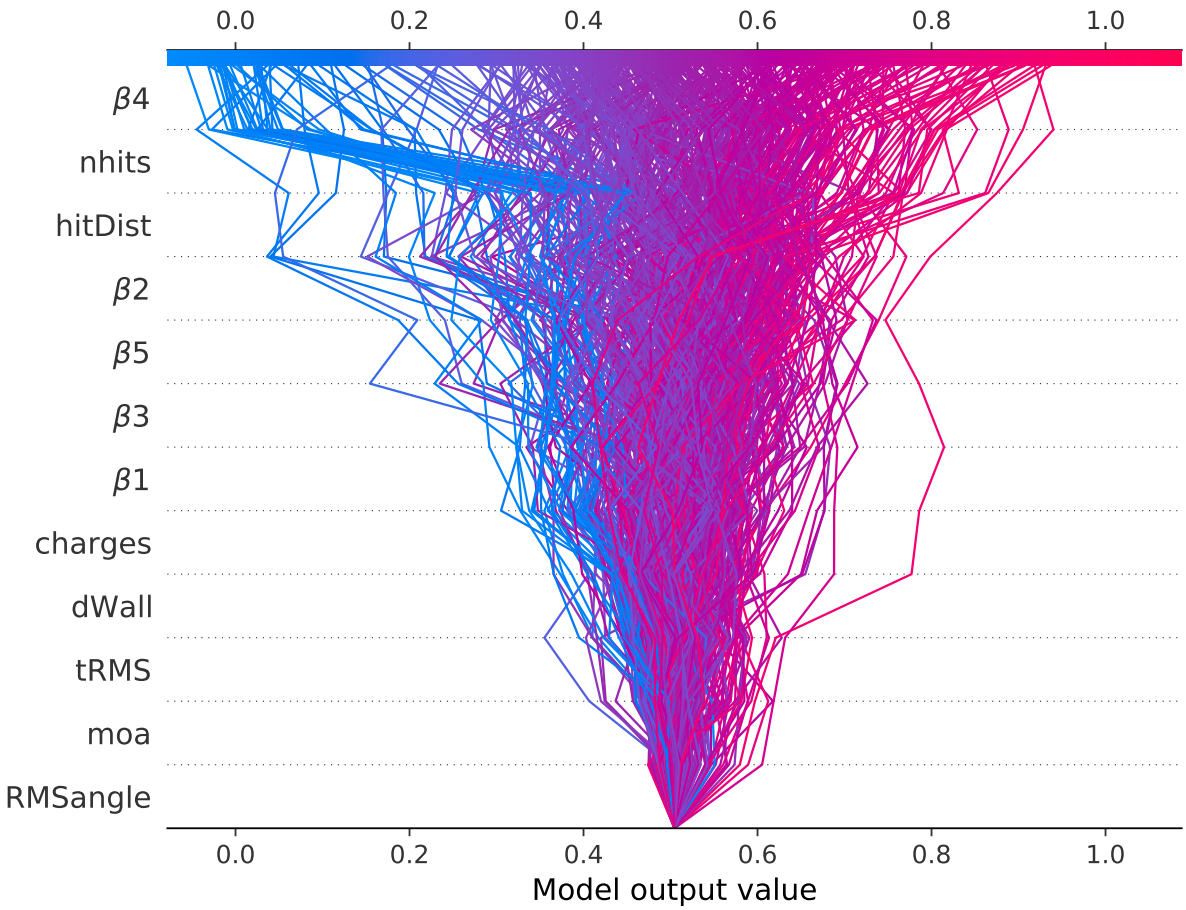


Figure 6.4: Combined decision plots are shown for 500 events sampled at random from the neutron capture and spallation background dataset simulated using WCSim for the IWCD tank geometry. Starting from the bottom base expectation model output of 0.504, the model output changes according to the Shapley value marginal contribution of every feature. This continues until the final output of the model, shown at the top of the plot. Electron events have model outputs above 0.5 and neutron electrons have model outputs below 0.5.

tions. In Fig. 6.4, some global trends may be observed: the number of hits often is impactful in lowering the model output value, the average hit distance between successive PMTs is often used to identify an event as electron-like, the isotropy parameters $\beta 1$, $\beta 2$, $\beta 3$ and $\beta 5$ tend to have less impact on the model output than $\beta 4$, which has the largest SHAP values and often can be affect the model output to be highly neutron-like. The RMS angle between consecutive events with respect to the event vertex appears to have the least marginal contribution on model output, and the mean opening angle, RMS event time and distance to wall also have low marginal contributions. This plot also gives a sense of the model output distribution for this sample of 500 events. Most event outputs are clustered between

the output values of 0.4 and 0.6. Of the electron-like events, there are a number of 'mostly' electron-like events with probabilities in the range of 0.6 to 0.9, but no events with softmax probabilities above 0.9. This contrasts to the neutron-like events, of which there are relatively fewer 'mostly' neutron-like events in the range of 0.1 to 0.4 and more 'definitely' neutron-like outputs with softmax probabilities around 0.0 to 0.1.

While the combined decision plot is useful in understanding trends in feature contributions and model outputs across many events, it is possible to obtain higher granularity visualizations across a much larger number of events. Another type of visualization, the 'beeswarm' plot, shows the range and density of SHAP values for individual features on the colorbar scale of the feature values themselves. Again, the feature are arranged on the vertical axis by feature importance, with the most important features (by average SHAP value) on the top and the least important features on the bottom. This type of visualization is more amenable to representing a higher magnitude of events. Figure 6.5 shows the beeswarm plot for all events in the neutron capture and spallation electron background dataset. This plot allows for the discernment of additional patterns in the dataset. In addition to the beeswarm plot in Fig. 6.5, Fig. 6.6 shows the feature differences for this spallation dataset. Referencing between the beeswarm and the feature differences plot provides insight into what is influencing the decision making patterns of the XGBoost model.

The most clear patterns from Fig. 6.5 are shown for number of hits and distance to the wall. For high number of hits, the SHAP value is uniformly negative. Correspondingly, in Fig. 6.6, it is clear that events with more than approximately 100 hits are uniformly neutron events (top-left plot). For the wall distance, it is clear from Fig. 6.6 that there are is a slight over-representation of background events at distances close to the wall. The XGBoost model clearly notices this difference, as events with lower wall distances (Fig. 6.6, blue) mostly have higher SHAP values, meaning the model output value is pushed higher to 1, for which the event is classified as an electron event. This pattern holds for the neutron capture and low energy background dataset (beeswarm plot in Fig. 8.9 and feature difference plot in Fig. 8.7) and is similar to a reduced extent for the high energy background dataset (beeswarm plot in Fig. 8.4 and feature difference plot in Fig. 8.2). In both cases, for the number of hits and wall distance features, the relative number of events with these extremes (very high number of hits, very low distance to wall) is low. This is seen by the thin lines in the beeswarm plot. Every dot in the plot

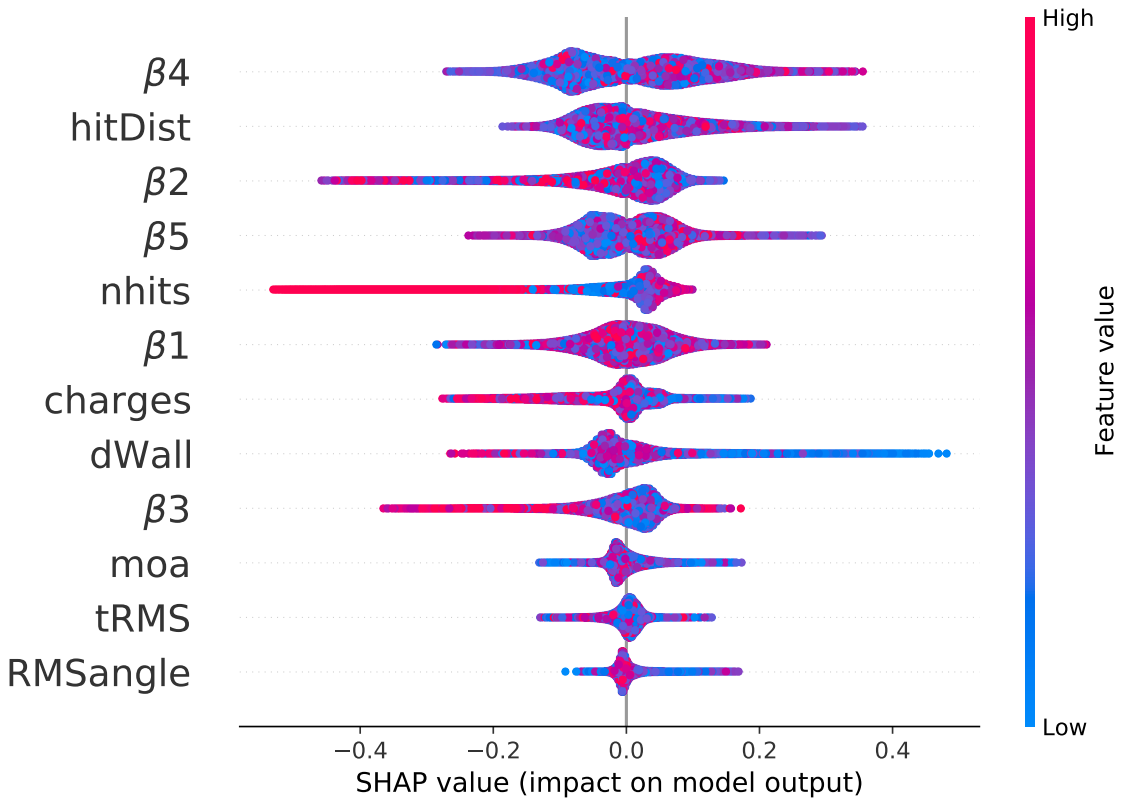


Figure 6.5: Beeswarm plot of SHAP values for the neutron capture and spallation background dataset simulated using WCSim for the IWCD tank geometry. The SHAP value for each feature in every event is plotted as a dot in the plot, where the x axis position corresponds to the SHAP value and the colorbar shows the feature value (blue is low, red is high). High SHAP values influence the model output towards 1 (electron-like event) and low SHAP values (negative) influence the model outputs toward 0 (neutron-like event).

represents a single event, and bulges in a feature row represent a greater density of events at that position. The thinness for number of hits and wall distance indicates that, while there are relatively fewer events with very high number of hits or very low distance to wall, these kinds of events are highly influential in determining the event output, as shown by the correspondingly high SHAP values.

The beeswarm Fig. 6.5 also reveals a notable difference between the lower order ($\beta1$, $\beta2$, $\beta3$) and higher-order ($\beta4$, $\beta5$) isotropy parameters. $\beta1$, $\beta2$ and $\beta3$ both have single mode representations in the beeswarm plot, in which there is a single bulge. Higher values for these parameters also attribute the output toward a neutron classification, on average. This correspondence may be seen by the feature difference plot in Fig. 6.6. Alternately, $\beta4$ and $\beta5$ have two main modes (bulges) in the SHAP value beeswarm plot, indicating two main regions with SHAP values

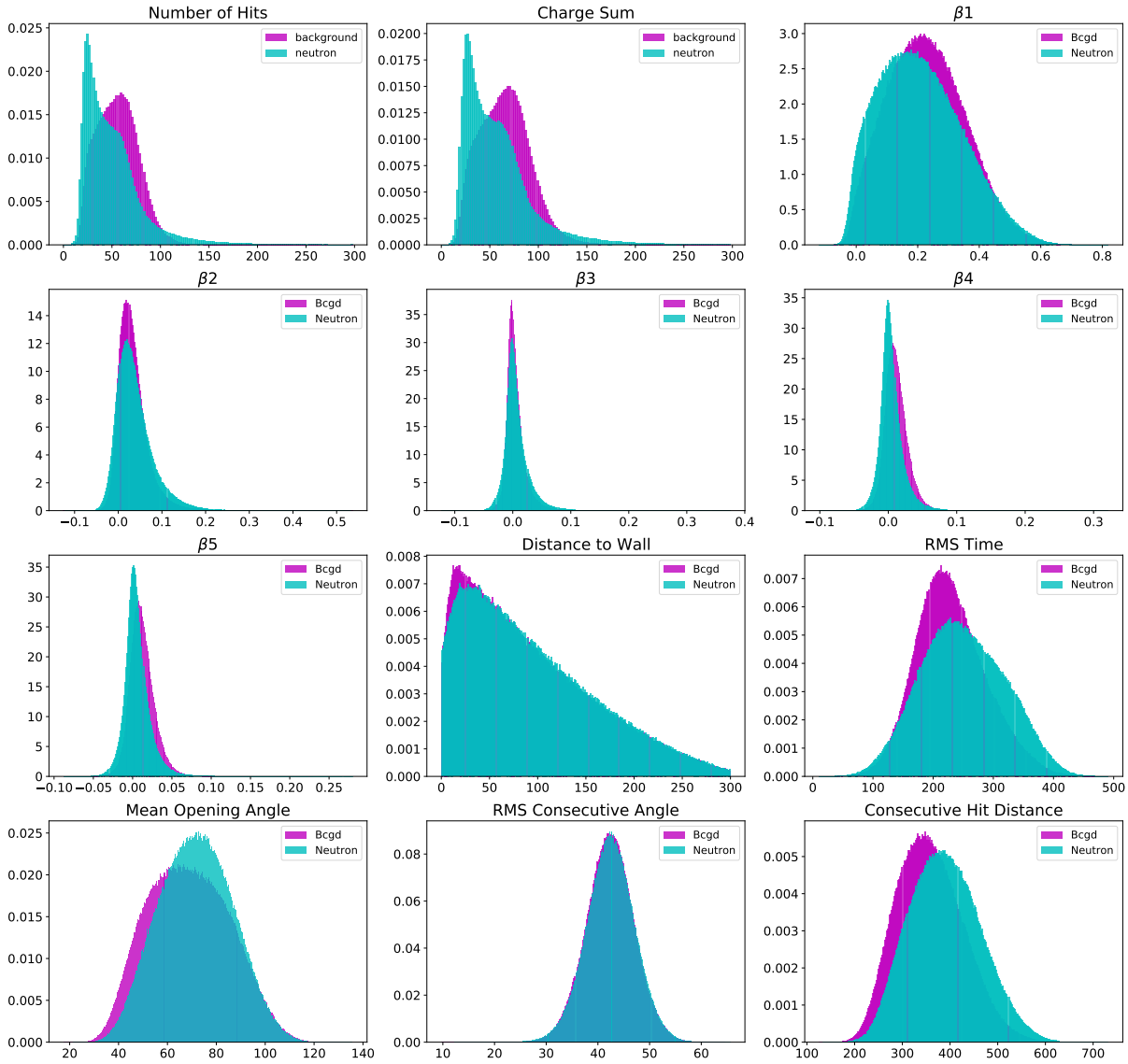


Figure 6.6: Comparison of twelve engineered features separated by neutron capture and spallation electron background events. The data consists of nearly 1.6 million events, generated by WCSim for the IWCD detector geometry.

of a similar range. For $\beta 5$, a clear distinction is seen between lower values of $\beta 5$, attributed toward neutron events, and higher values of $\beta 5$, attributed toward electron events. While this difference is clear, the SHAP values themselves are lower, showing a smaller output impact. This small difference is observable in Fig. 6.6 as well for $\beta 5$.

This double-moded pattern is visible to a lesser extent for the neutron capture and low energy dataset, shown in Fig. 8.9. In this case, the lowest order isotropy parameter $\beta 1$ has such overlap that the beeswarm plot shows little asymmetry

around the SHAP value of 0. Both $\beta 2$ and $\beta 3$ have SHAP values which tend toward an electron output for lower isotropy values, similar to the spallation electron dataset, and vice versa. Both $\beta 4$ and $\beta 5$ show a similar double moded pattern in the beeswarm plot with $\beta 4$ in particular tending to have negative SHAP values for low $\beta 4$ isotropy values. The beta parameters are generally less important in the beeswarm plot of the high energy background dataset (appendix, Fig. 8.4), although it can be clearly seen that high $\beta 2$, high $\beta 3$ and low $\beta 4$ values correspond to neutron outputs (negative SHAP values), which corresponds exactly to the feature difference plot (Fig. 8.2).

For the spallation dataset, $\beta 4$ has a similar double-moded pattern in the beeswarm plot, but the attributed difference is smaller for lower and higher values of the parameter. However, $\beta 4$ still has the greatest average absolute SHAP value, and therefore the greatest average impact on the model output. This is why $\beta 4$ is shown as the most important feature. The $\beta 4$ parameter is also the most important feature for the low energy dataset. While the beeswarm plots list the feature importance rankings of the various features, it does not show the relative importance quantities. The relative feature importance quantities are represented in the bar plot in Fig. 6.7 for the spallation dataset, and in the appendix Figs. 8.5 and 8.10 for the high and low energy background datasets, respectively. In this plot, the features are ranked by the mean absolute SHAP value, with the most important feature at the top and least important feature at the bottom. The bar widths on the x axis show the value of the mean absolute SHAP value for that feature. This plot helps to visualize the relative importance of the features.

For the spallation and low energy background datasets, the relative feature importances decrease gradually from $\beta 4$ down to the RMS consecutive hit PMT angle. This is an indicator of more complicated decision making processes in the boosted decision trees. This is in contrast with the feature importance plot for the high energy background dataset (Fig. 8.5) for which the charge sum is the clearly most important feature, followed by a steep dropoff for the number of hits, $\beta 1$, etc. down to the RMS event time. This importance barplot indicates a more simplistic decision making process for XGBoost for this dataset, based mainly on the charge sums and only secondarily on the other features in the event.

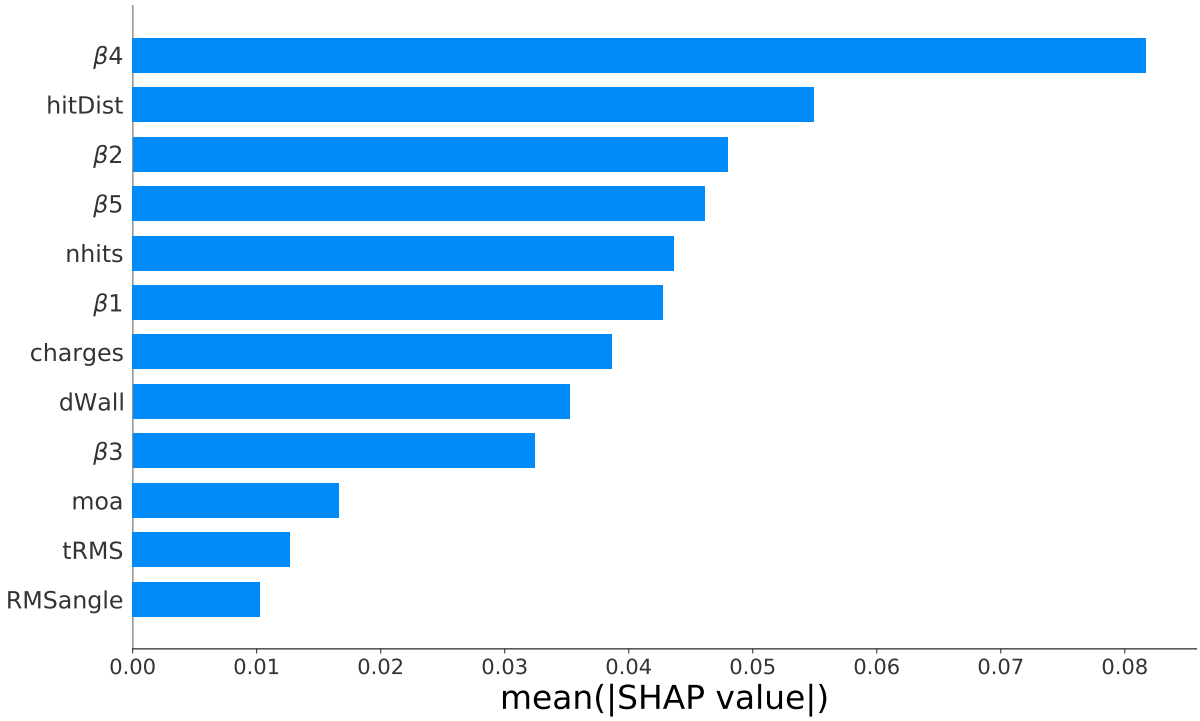


Figure 6.7: The relative feature importances are shown for the neutron capture and spallation electron background dataset, ranked by average absolute SHAP value. These results show that the beta parameters (mostly $\beta 4$), the event number of hits, and the average distance between consecutive PMT hits in a given event are the most importance features for the XGBoost model to predict event outcomes.

Chapter 7

GNN Application

In this study, the PyTorch Geometric (PyG) library was used to apply graph neural network models to the IWCD dataset [89]. This particular library was chosen for its ease of use, breadth of graph network models available, data loading tools and GPU support. The method developed for graph model training using PyG follows a common framework. The code reads in the configuration hyperparameters (GNN model architecture, number of epochs, batch size, etc.), the PyG DataLoader objects are constructed for the train, test and validation datasets (the DataLoaders receive instructions on graph construction) and then the model is trained by iterating over the training dataset for the predetermined number of epochs. At regular intervals, the model is applied to the validation dataset to check for underfitting or overfitting. After the model has been trained, it is applied on the test dataset and evaluation metrics are computed. For all models the data was partitioned in a 80%, 10%, 10% split of training, validation and testing data, consisting of approximately 1.2, 0.2 and 0.2 million data examples respectively. During training, the model parameters were updated using Adam optimization [90] with cross-entropy loss. Training was carried out on a Quadro P2000 GPU.

7.1 GCN

Before a graph network model can be trained on the simulated IWCD particle datasets, the graph construction procedure must be specified. Every event in the IWCD data consists of a given number of hits, with each hit containing the time, deposited charge and the 3-dimensional position and 3-dimensional orientation of the hit PMT. This data format translates naturally to a graph structure, for which each hit may be represented by a node consisting of the eight features mentioned

above.

However, several factors complicate the question of graph construction. It is not immediately obvious which nodes should be connected or, for the connected nodes, what edge weightings they should have. Additionally, the number of hits varies for every event. Therefore, the corresponding graphs could either vary in size (dynamic graph) or zero padding could be added to those events with less than the maximum number of hits to fix the graph input dimension (fixed graph). To manually create a new 'dataloader' object in PyG, the $\_\_get()\_\_$ and $\_\_len()\_\_$ methods from the default PyG dataloader class must be overridden to determine how to retrieve an individual graph and its corresponding length. The features x, labels y and adjacency matrix of the graph must be specified in the $\_\_get()\_\_$ method. To accommodate the various graph construction options for this problem domain, the manual WCH5Dataset class was written. WCH5Datasets inherits from the PyG base Dataloader class, and tunable runtime parameters were created that turn on, turn off or adjust the graph construction implementation within WCH5Dataset.

The graph convolutional network (GCN) model described in Section 3.4 was the first GNN model applied on the IWCD particle data. Three graph construction methods were compared within the GCN framework to test performance. This includes using a static (fixed size) versus dynamic graph, edge weighting (inversely proportional to distance) versus uniform weights, and a fully connected graph versus the k nearest neighbour graph. This testing was done in a similar fashion to the hyperparameter tuning of XGBoost in Section 6. After one graph construction choice was found to work better (e.g. static graph vs dynamic graph), that particular method was then employed in the later tests.

To begin, the GCN model was tested on graphs constructed using a static (zero padding, fixed input size), fully connected (every node connected to every other node) graph representation, with all edge weightings set to a value of one. Given 450 maximum hits per event, the corresponding fully connected, static network consisted of 202,500 node connections in total. The parameters of the GCN model architecture itself, including the number of filtering layers and activation nodes, were also tested and fixed for these tests. Consistently good performance was obtained from the configuration of two alternating layers of GCN convolutional filtering and activation computation, with 24 and 8 compute nodes in the first and second hidden layers respectively. This was followed by max pooling across all graph nodes and then the log softmax of a fully connected layer with two activations

at the output layer. All results were obtained by training with a batch size of 32, learning rate of 0.0003 and learning rate decay of 0.001%. The model was tested on the validation dataset every 30,000 iterations and the model parameters were saved if the result improved on the previous best validation score.

The first graph construction comparison tested whether the GCN model learned better on static, padded graphs or dynamic graphs (no padding) with a variable number of nodes per event. The results of training the GCN model for the three neutron capture datasets with spallation, low energy, and high energy electron backgrounds on *static*, fully connected event graphs with uniform edge weightings are shown in Table 7.1. Corresponding results for GCN results on *dynamic*, fully connected graphs with uniform edge weightings graphs are shown in Table 7.2. The performance is higher for the static graph construction method than using a dynamic graph across all datasets and metrics. The average test accuracy improvement is 9.85% for the static compared to the dynamic graphs. The difference is most notable on the high energy background dataset for which the static graph represents a 17.8% improvement.

Background Source	Train Accuracy	Validation Accuracy	Test Accuracy	ROC AUC
Spallation	61.3	63.1	63.1	0.667
Low Energy	56.7	57.5	57.4	0.603
High Energy	79.2	79.7	79.6	0.860

Table 7.1: GCN model applied to fixed input (zero padding added), fully connected and uniformly edge weighted graphs for the simulated IWCD neutron capture datasets with backgrounds of spallation, low energy and high energy electron radiation.

Background Source	Train Accuracy	Validation Accuracy	Test Accuracy	ROC AUC
Spallation	58.5	59.8	59.9	0.628
Low Energy	53.3	55.2	55.1	0.572
High Energy	62.6	65.4	65.4	0.704

Table 7.2: GCN model applied to dynamic (number of nodes varies per event), fully connected and uniformly edge weighted graphs for the simulated IWCD neutron capture datasets with backgrounds of spallation, low energy and high energy electron radiation.

While accuracy was higher for the static, padded graphs, the runtime was also considerably longer than for the dynamic graphs. This was due to the significantly greater number of connections and message passing operations in the padded graphs. Runtime over 5 training epochs for the fixed size, fully connected graphs took an average of 30 hours. This runtime of approximately 6 hours per epoch

contrasts to only 14 *minutes* per epoch with the dynamic, fully connected graphs, which is over 25 times faster. Table 7.2 shows the results of dynamic GCN training over 75 training epochs, which took approximately 17 hours, while the results in Table 7.1 are from the fixed input GCN model trained over 5 epochs. A higher number of epochs was not found to improve the performance for either model.

The results in Tables 7.1 and 7.2 indicate the GCN model learns better on input graphs of a fixed size for this problem domain. The GCN model essentially implements a low-pass filter, or smoothing operation over the graph nodes, followed by node activation updates through fully-connected layers. The fixed size graph representation could have the effect of 'leveling the playing field' and enabling the model to have greater generalizibility and consistency across different events. The variability of the graph representation is significant especially for events with few hits, and this difference could trick the model, or cause it to learn more slowly, compared with a fixed size input representation.

After setting the graph representation to static (fixed input size with zero padding added), the effect of distance weighting on the node edges was investigated to see if in the fully connected graph, where every node is connected to every other node, edge weightings related to physical distance could provide a learning advantage over uniform edge weightings set to a tensor of ones. A function 'dist_pos_matrix' was written to return the edge weights for a given event. This function extracts the position information for the PMTs in the event, then calls on the 'distance_matrix' function from the SciPy spatial Python library to construct the matrix of distances from every node to every other node in the event. The reciprocal function is applied to the matrix elements such that closer hit pairs have larger weights and sparser node pairs lower weights. The matrix is then normalized and converted to compressed sparse row format, from which it may be converted into PyTorch Geometric edge indices and edge attributes (weights) using the corresponding function in 'torch_geometric.utils.convert.'

The results of training GCN on fully connected, static, inverse distance edge weighted graphs is shown in Table 7.3. Runtimes are nearly identical to the same model with fixed edge weights. While all test accuracies and ROC AUC scores are higher for the static, edge weighted graphs than the dynamic graphs (Table 7.2), the scores are lower than the static, padded, uniform edge weighted graphs in Table 7.1. Indeed, with weighted edges the test accuracies are 5.05% lower and the ROC AUCs are 4.76% on average than with the uniform edge weights on the static graphs. This somewhat unintuitive result is perhaps representative of the fact that,

on the scale of hundred of thousands of node connections for IWCD particle events, edge weightings might overcomplicate the GCN parameter learning.

Background Source	Train Accuracy	Validation Accuracy	Test Accuracy	ROC AUC
Spallation	59.7	61.3	61.4	0.632
Low Energy	53.1	57.0	56.9	0.600
High Energy	75.6	71.7	71.7	0.849

Table 7.3: GCN model applied to fixed input (static), fully connected graphs with edge weights between nodes corresponding to their positional inverse squared distance for the simulated IWCD neutron capture datasets with backgrounds of spallation, low energy and high energy electron radiation.

The GCN model with uniform edge weights was also tested on graphs for which every node was only connected to its k positional neighbours. Compared to the fully connected graphs, this lowered the number of node connections to $450 * k$. For $k = 20$, for example, this reduced the number of edges to 9000, less than the fully connected graph by a factor of 22.5. On average, this led to a runtime of approximately 50 minutes per epoch, faster than the fixed size, fully connected graphs by over 700%, but still considerably slower than dynamic sized graphs. The nearest neighbour edge index tensor was calculated for every node using the 'knn_graph' function from the 'torch_cluster' library given the 'k_neighbours' parameters k. Results are shown in Table 7.4 for GCN models trained over 25 epochs on the neutron capture and high energy background dataset for the nearest neighbour graphs with k ranging from 10 to 30 in increments of 5. In general, the accuracy and ROC AUC scores for the k nearest neighbour GCN models are similar for all k values between 15 and 30, and consistently slightly underperform the static, fully connected, uniform edge weighted graphs. Training was not found to improve beyond 25 epochs.

k neighbours	Train Accuracy	Validation Accuracy	Test Accuracy	ROC AUC
10	76.8	78.2	78.1	0.839
15	77.4	79.0	78.8	0.846
20	77.8	78.9	78.7	0.847
25	78.1	79.0	78.9	0.850
30	78.1	79.0	78.8	0.849

Table 7.4: GCN model applied to event graphs with each node connected to its k-nearest positional neighbours, with uniform edge weights between nodes, for the simulated IWCD neutron capture datasets with high energy electron background radiation. The k number of nearest neighbours was varied from 10 to 30 in increments of 5.

Overall, the GCN model was found to perform best on static, fully connected, uniform edge weighted graphs. The results are shown in Table 7.1. This GCN configuration has comparable metrics to the highest likelihood baseline (see Table 4.1, q_sum), with 0.6% higher accuracy on the spallation set, no difference on the low energy dataset, on a 0.2% lower accuracy on the high energy dataset. Therefore, it may be concluded that the GCN model was largely learning to classify events based on the trivial number of hits, and that it failed to significantly learn from the geometric differences of neutron capture to electron background hit patterns. Adding any additional network layers to the GCN models was also found to worsen performance, presumably as the extra filtering step oversmoothes the node representations. Overall, the GCN model is perhaps better suited to contexts with fewer and more meaningful node connections and edge weightings, such as social networks and credit found identification, rather than the 'point cloud' representation of the particle events.

7.2 DGCNN

The DGCNN model was the next graph network model applied to the particle classification task. Described in Section 3.4, the DGCNN model was selected for its ability to learn from point cloud data specifically. The network architecture configuration was set to the default from the PyTorch Geometric example documentation, which consisted of the following: two dynamic edge convolution layers followed by a fully connected layer, a global max pooling layer and a final MLP to yield the output class probabilities. The first edge convolution layer applied an MLP on input node features with three layers of 64 compute units each. The second edge convolution layer took the output of the first as input and applied an MLP with a single layer and 128 output units. In both cases, the MLP was applied to every node pair ($n * 2$ pairs for n nodes in an event) over the k-nn (k nearest neighbour) graph representation of each node, and the representations were updated by pooling the learned edge features (Fig. 3.6). The penultimate fully connected layer concatenated the 64 and 128 unit features from the first two dynamic edge convolution layers, passing this input to a 1024 unit output. Global max pooling was applied over the n nodes to reduce the representation from $n*1024$ to only 1024. The final MLP then passed this information into final layers of 512, 256 and 2 activation nodes respectively and the softmax of the output was applied to calculate the output the binary classification probabilities. Note that for every

fully connected layer throughout the network, the activations were calculated using the RELU activation function and batch normalization [91] to reduce overfitting. The model description is represented in Table 7.5.

Layer	Input Features	Output Features
EdgeConv1 (MLP1)	n ∗ 2 ∗ 8	n ∗ 64
	n ∗ 64	n ∗ 64
	n ∗ 64	n ∗ 64
EdgeConv2 (MLP2)	n ∗ 2 ∗ 64	n ∗ 128
FC	n ∗ (64 + 128)	n ∗ 1024
Global Max Pooling	n ∗ 1024	1024
MLP3	1024	512
	512	256
	256	2

Table 7.5: Applied DGCNN architecture for neutron capture and electron background event discrimination. Two dynamic edge convolutional blocks were applied, followed by a fully connected layer, global max pooling, and a final multi-layer perceptron layer.

With the fixed architecture as described in Table 7.5, the number of nearest neighbours k in the DGCNN dynamic edge convolution blocks were adjusted over multiple runs to compare performance. Table 7.6 shows the results of applying the DGCNN model on the spallation background dataset with the k hyperparameter varying from 10 to 30 in increments of 5. The resulting accuracies were largely the same for $k = 15$ to $k = 30$, while $k = 10$ neighbours led to a markedly lower accuracy. Among the range of $k = 15$ to $k = 30$, $k = 25$ yielded the highest ROC AUC score of 0.797, although the 0.001 difference compared to the other k values in the range was not necessarily statistically significant.

k neighbours	Train Accuracy	Validation Accuracy	Test Accuracy	ROC AUC
10	70.9	72.0	71.9	0.792
15	71.8	72.2	72.3	0.796
20	71.7	72.3	72.3	0.796
25	71.8	72.4	72.4	0.797
30	71.4	72.4	72.4	0.796

Table 7.6: DGCNN model classification accuracies for variations of the number of nearest neighbours k in the DGCNN dynamic edge convolution blocks from 10 to 30 in increments of 5.

While there was minimal performance difference for the range of $k = 15$ to $k = 30$, there was however a large variance in the training times. This was expected as, for n f-dimensional input nodes, an $n * k * a_n$ -dimensional tensor is

generated before pooling across the neighbouring edge features (Fig. 3.6) for every dynamic edge convolution block. Therefore the total number of training parameters increases significantly for every increment of nearest neighbours k. Figure 7.1 shows a barplot of training times for the DGCNN model for k from 10 to 30. This figure demonstrates a sharp increase in training time after about $k = 20$ and more than a doubling in overall training time from $k = 10$ to $k = 30$.

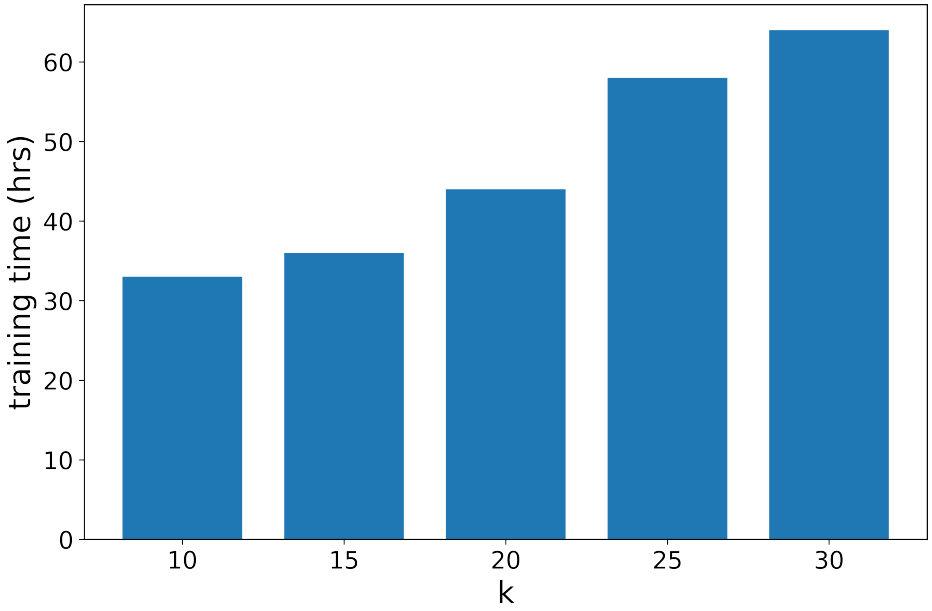


Figure 7.1: DGCNN model training times (in hours) for varying number of nearest neighbours k for the construction of the k-nn graph in the dynamic edge convolutional block.

Given the results in Table 7.6 and Fig. 7.1, $k = 20$ appears to be a good compromise between training time and classification accuracy. However, when training time is not a significant impediment, $k = 25$ might be used to optimize results. This $k = 20$ value was applied with the DGCNN model architecture as described in Table 7.5 for comparison across the other datasets, including the highE and lowE background neutron capture datasets. Table 7.7 shows the results for this configuration.

Compared to the likelihood statistical baseline, the DGCNN model results in Table 7.7 showed accuracy improvements of 9.9%, 9.6% and 2.4% for the neutron capture datasets with background of spallation, low energy and high energy respectively. Where the classification accuracy by likelihood was already high ($\sim 80\%$ on highE dataset), the improvement was lowest. The $\sim 10\%$ classification accuracy improvements on the spallation and lowE datasets strongly indicates the capability of the DGCNN model to learn from event topology and other, more subtle factors

Background Source	Train Accuracy	Validation Accuracy	Test Accuracy	ROC AUC
Spallation	71.8	72.4	72.4	0.797
Low Energy	67.0	67.1	67.0	0.740
High Energy	82.3	82.3	82.2	0.904

Table 7.7: DGCNN model with $k = 25$ nearest neighbours in the edge convolution block and model configuration described in Table 7.5 applied to the neutron capture datasets with backgrounds of spallation, low energy and high energy electron radiation for the simulated IWCD geometry.

than the number of hits and overall sum of charges within the event. The dynamic method of graph construction with the DGCNN model, which shuffles the groupings of every node with its other nearest neighbour nodes in semantic space, allows the diffusion of nonlocal information throughout the graph. This ostensibly allows the DGCNN model to learn global event topology in a way which the GCN model, restricted to operating over fixed input graphs, was not able to.

Overall, the DGCNN also slightly outperformed the best XGBoost model for the more difficult lowE and highE background datasets, representing improvements in accuracy of 1% and 0.7% and ROC AUC score of 0.013 and 0.007 respectively. However, the XGBoost model outperformed DGCNN for the less difficult highE background classification task, yielding a 1.4% higher classification accuracy and 0.015 higher ROC AUC.

The overall test accuracy results for all approaches undertaken in this study, including the likelihood baseline analysis, XGBoost with feature engineering and the GCN and DGCNN models are presented in Table 7.8. The best accuracies for each of the neutron capture and spallation, lowE and highE electron background datasets are shown in bold as 72.4% (DGCNN), 67.0% (DGCNN) and 83.6% (XGBoost) respectively.

Dataset Background Source	Likelihood	XGBoost	GCN	DGCNN
Spallation	62.5	71.4	63.1	**72.4**
Low Energy Electron	57.4	66.3	57.4	**67.0**
High Energy Electron	79.8	**83.6**	79.6	82.2

Table 7.8: Overall accuracies for neutron capture versus electron background classification for the likelihood analysis (Likelihood), XGBoost, GCN and DGCNN methods over the datasets consisting of neutron capture and spallation (Spallation), low energy (Low Energy Electron) and high energy (High Energy Electron) electron background events. The best accuracies were found to be 72.4% for the spallation dataset (DGCNN), 67.0% for the low energy background dataset (DGCNN) and 83.6% for the high energy background dataset (XGBoost).

Chapter 8

Conclusions

This thesis has presented a search to improve the classification performance of neutron capture identification in water Cherenkov detectors using techniques in machine learning. Beyond the original dataset of neutron captures and high energy electron background ('highE'), two new datasets were synthesized using WCSim simulation software that vary the electron background energy scale to lower energies ('lowE') and that mimic a muon spallation background source ('spallation'). To provide a performance baseline, a statistical model was applied to classify events using maximum likelihood of kernel density estimates of the main event type discriminants, namely the number of hits and charge sums. The highest baseline accuracies were found to be 62.5%, 57.4% and 79.8% for the spallation, lowE and highE datasets respectively. Of note, an informal procedure of randomizing event displays and manually classifying by eye was undertaken over several hundred events. The neutron capture signal and background events overlapped visually to the extent that no improvement was obtained by this human-level performance estimation over the likelihood baseline approach.

Next, a series of features were engineered from the datasets. Besides number of hits and charge sums, the beta parameters $\beta 1$-$\beta 5$ were created to capture event isotropy. The mean opening angle, event vertex distance to wall, RMS consecutive hit angle and mean consecutive hit distance were also computed to summarize event topology and the RMS event time was added to capture timing discrimination. Gradient boosting decision trees were applied on these engineered features using the XGBoost algorithm. The XGBoost model hyperparameters were tuned for each dataset to optimize performance, yielding test accuracies of 71.4%, 66.3% and 83.6% for the spallation, lowE and highE datasets, representing improvements of 8.9%, 8.9% and 3.8% over the baseline approach respectively. SHAP analysis of these model outputs revealed useful information. For the more challenging lowE

and spallation datasets, the $\beta 2$, $\beta 4$, $\beta 5$, number of hits and consecutive hit distance parameters were consistently rated most important, as measured by the mean absolute SHAP value, while the number of hits and charge sums were the overwhelmingly most important by far for the highE dataset. This emphasized the relevance of energy distribution comparison. Given particle events over similar energy ranges, where hit statistics are more closely related, classification improvements were most accessible from accessing the topological and geometric discrimination information. The SHAP beeswarm plots also helped to understand at a granular level how XGBoost made its decisions by presenting the features and feature values which tended to influence classification towards neutron capture or background.

Drawbacks to the XGBoost and the feature engineering approach include preprocessing time to calculate the feature values and the fact that the calculation of several features relies on the event vertex position. For this research, the true vertex position was taken from the simulation information, but in reality a vertex reconstruction algorithm would need to be used, introducing some uncertainties into the equations. As an alternative approach, deep learning was introduced to the neutron tagging classification problem via graph neural networks. Graph neural networks can operate on the original particle data directly, and the representation of hit PMTs to nodes is natural and fits to the cylindrical IWCD geometry without loss of information.

The graph convolutional network (GCN) model was the first graph network implemented, and a variety of graph construction approaches were tested. This included static, fully connected graphs with zero padding, dynamic graphs with hit PMTs constituting the only graph nodes, graphs with uniform edge weights, edge weighting scaled to inverse positional PMT distance, and graphs with each node connected only to the k nearest neighbour nodes in position space. Of all these cases, the best test accuracies obtained were 63.1%, 57.4% and 79.6% for the spallation, lowE and highE datasets respectively, using the fully connected, zero padded, uniform edge weighted graphs. These accuracies were nearly identical to the baseline likelihood accuracies, providing a strong indication that the graph networks were learning mainly from the number of hits in the event, or the number of non-zero nodes. At the least, there was little evidence that any learning was accomplished from the characteristic hit pattern geometry differences between the event types. Besides the GCN, other models like the AGNN, [92], SG [93] and GAT [94] were applied and all results were similar to, or somewhat poorer than, the results from the GCN. Notwithstanding significant efforts in hyperparameter opti-

mization and graph construction, performance hit a threshold near the likelihood baselines. For example, adding additional layers in the GCN was found to decrease performance as the filtering operations oversmoothed the node representations.

The dynamical graph convolutional neural network (DGCNN) was the only GNN model found to have significantly improved neutron tagging performance above baseline metrics in this research. Compared to the other GNN models, which pass their feature states as messages between nodes, the DGCNN model learns edge features between node pairs. The 16 input features between node pairs (charge, time, 3D position and 3D orientation of each node) are passed through an MLP with learnable parameters. This allows the model to learn both global features, since the MLP has shared parameters over all node pairs, and local features, since the weights between input features are learnable. This allows the model to determine the relevance between, for example, the y-coordinate PMT orientation of one node and the recorded charge of the other. This imbues the DGCNN model a capacity for granularity of learning that surpasses the node neighbour feature vector averaging mechanism of other GNN models.

In addition, although the other GNN models consisted of fully connected networks capable of diffusing information throughout the entire graph, the magnitude of connections could have led to oversmoothing in the network. This is similar to the effect of a low-pass filter, especially in cases where there are many 0-valued nodes and relatively few hit PMTs. With the DGCNN model, every node is connected to its k most relevant neighbouring nodes in terms of semantic similarity. Therefore, nodes are only connected to their key neighbours, while the dynamic recomputation of graphs in successive model layers also enables the nonlocal diffusion of information throughout the network. This overall computational structure is hypothesized to be an important factor for the improved neutron tagging performance of the DGCNN model compared to the other GNN models that were implemented.

With the DGCNN model, the hyperparameter k was tuned and reported accuracies were found to be 72.4%, 67.0%, and 82.2% for the spallation, lowE and highE datasets respectively, representing improvements of 9.9%, 9.6% and 2.4% over the baseline metrics. Thus, DGCNN slightly outperformed XGBoost on the more challenging lowE and spallation datasets, while slightly underperforming XGBoost on the less challenging highE dataset. DGCNN, however, retains the advantage of not requiring any preprocessing or prior knowledge. On the other hand, XGBoost provides a much greater level of model interpretability. Furthermore, once the engi-

neered features have been computed, the training time of XGBoost for the datasets used in this study was within the range of only 45 minutes to one hour, much faster than the DGCNN model which took from 30 to over 60 hours, depending on the value of k. However, DGCNN was trained over only a single GPU, and using multiple GPUs could reduce the runtime significantly. Table 7.8 shows the overall results of XGBoost, GCN and DGCNN compared to the likelihood baseline for all datasets.

Of note, the GCN model might work better for a graph setup where the nodes include all PMTs in the geometry of the WC detector. However, the sheer magnitude of computation would certainly be a large bottleneck. The fixed input GCN model, with 500 nodes, had a runtime of approximately 6 hours per epoch. Expanding this up to tens of thousands of nodes to cover the full range of a WC detector could easily introduce over 100 times more connections, potentially causing a runtime on the order of 1000 hours for only a single training epoch.

Overall, both XGBoost with feature engineering and DGCNN show promise in improving neutron tagging efficiency in water Cherenkov detectors. In particular, the application of these methods in the IWCD might help reduce systematic uncertainties for the Hyper-Kamiokande detector, which it turn could advance our understanding of neutrino physics and the Standard Model itself. In future, the network architecture of the DGCNN model could be optimized. While the k value and learning rates were tuned, optimization of hyperparameters like the number of units in the shared MLP or number of model layers could improve performance. Also, the models developed in this study could be applied in other contexts besides neutron tagging. In particular, it would be interesting to see if these models could be applied to classification of various particles in the high energy regime.

For practical purposes, given that these models were developed for data simulation, another reasonable next step would include the deployment of these models for neutron tagging in active water Cherenkov detectors. This would test if the models are transferable for real use cases. Also, these models could be incorporated into a pipeline which tests for the coincidence of neutron capture and positron rings within a timescale indicative of neutrino inverse beta decay. While the development of improved neutron tagging is desirable, the ultimate goal is to trace back to the originating neutrino to probe deeper into the unknowns of neutrino physics. An end-to-end network could thus be deployed using the neutron tagging models developed in this research to better identify the neutrinos themselves in the overarching process of the neutrino inverse beta decay.

Bibliography

[1] Hernández, P. (2016). IFIC, Universidad de València and CSIC, E-46071 Valencia, Spain. 2015 CERN–Latin-American School of High-Energy Physics, 85.

[2] Ecker, G. (2020). James Chadwick: ahead of his time. arXiv preprint arXiv:2007.06926.

[3] Bilenky, S. M. (2013). Neutrino. History of a unique particle. The European Physical Journal H, 38(3), 345-404.

[4] Pontecorvo, B. (1946). Inverse beta decay. Chalk River Report PD-205, 15.

[5] BETHE, H., PEIERLS, R. The Neutrino. Nature 133, 689–690 (1934). https://doi.org/10.1038/133689b0

[6] Anderson, E. C. The Reines-Cowan Experiments.

[7] The hunt for the muon neutrino. NobelPrize.org. Nobel Media AB 2021. Sun. 21 Feb 2021. https://www.nobelprize.org/prizes/physics/1988/9557-the-hunt-for-the-muon-neutrino/

[8] Davis, R., Jr. (2003). The Nobel Prize in PHYSICS 2002. https://www.nobelprize.org/prizes/physics/2002/davis/biographical/

[9] Pinch, T. (1985). Theory testing in science—the case of solar neutrinos: Do crucial experiments test theories or theorists?. Philosophy of the Social Sciences, 15(2), 167-187.

[10] Fukuda, Y., Hayakawa, T., Ichihara, E., Inoue, K., Ishihara, K., Ishino, H., ... & Super-Kamiokande Collaboration. (1998). Evidence for oscillation of atmospheric neutrinos. Physical Review Letters, 81(8), 1562.

[11] Ahmad, Q. R., Allen, R. C., Andersen, T. C., Anglin, J. D., Barton, J. C., Beier, E. W., ... & Smith, A. R. (2002). Measurement of day and night neutrino energy spectra at SNO and constraints on neutrino mixing parameters. Physical Review Letters, 89(1), 011302.

[12] Czakon, M., Gluza, J., & Zralek, M. (1999). Are neutrinos Dirac or Majorana particles?. arXiv preprint hep-ph/9910357.

[13] Frank, I. M. (1971). COHERENT RADIATION OF THE FAST ELECTRON IN MEDIUM (No. JINR-P4-5954). Joint Inst. for Nuclear Research, Dubna (USSR). Lab. of Neutron Physics.

[14] Zhou, Y. (2015, June). Analysis on the Design and Application of the Small and Large Current Photomultiplier Tube. In 2015 2nd International Conference on Electrical, Computer Engineering and Electronics (pp. 84-87). Atlantis Press.

[15] Bionta, R. M., Blewitt, G., Bratton, C. B., Casper, D., Ciocio, A., Claus, R., ... & Wuest, C. (1991). Observation of a neutrino burst in coincidence with supernova 1987A in the Large Magellanic Cloud. In Neutrinos And Other Matters: Selected Works of Frederick Reines (pp. 340-342).

[16] Hirata, K. S., Kajita, T., Koshiba, M., Nakahata, M., Oyama, Y., Sato, N., ... & Cortez, B. G. (1988). Observation in the Kamiokande-II detector of the neutrino burst from supernova SN1987A. Physical Review D, 38(2), 448.

[17] Hirata, K. S., Inoue, K., Ishida, T., Kajita, T., Kihara, K., Nakahata, M., ... & Zhang, W. (1991). Real-time, directional measurement of B 8 solar neutrinos in the Kamiokande II detector. Physical Review D, 44(8), 2241.

[18] Fukuda, Y., Hayakawa, T., Ichihara, E., Inoue, K., Ishihara, K., Ishino, H., ... & Young, K. (1998). Measurements of the solar neutrino flux from Super-Kamiokande's first 300 days. Physical Review Letters, 81(6), 1158.

[19] Fukuda, S., Fukuda, Y., Ishitsuka, M., Itow, Y., Kajita, T., Kameda, J., ... & Super-Kamiokande Collaboration. (2001). Constraints on neutrino oscillations using 1258 days of Super-Kamiokande solar neutrino data. Physical Review Letters, 86(25), 5656.

[20] Yokoyama, M. (2017). The hyper-Kamiokande experiment. arXiv preprint arXiv:1705.00306.

[21] Scott, M. (2016). An intermediate water Cherenkov detector at J-PARC. In Proceedings of the 10th International Workshop on Neutrino-Nucleus Interactions in Few-GeV Region (NuInt15) (p. 010039).

[22] Collaboration, T., Abe, K., Adam, J., Aihara, H., Akiri, T., Andreopoulos, C., ... & Koga, T. Measurements of neutrino oscillation in appearance and disappearance channels by the T2K experiment with 6.6 E20 protons on target.

[23] Barbi, M., Berardi, V. et al. (2019). A Water Cherenkov Test Beam Experiment for Hyper-Kamiokande and Future Large-scale Water-based Detectors. Scientific Committee Paper, CERN-SPSC-2019-042 [SPSC-I-254].

[24] Akutsu, R. (2021, September 9). The intermediate water Cherenkov detector for the Hyper-Kamiokande experiment [Conference session]. NuFact 2021: The 22nd International Workshop on Neutrinos from Accelerators, Zurich, Switzerland. `https://indico.cern.ch/event/855372/contributions/4441604/attachments/2306169/3923421/NuFact2021_RyosukeAkutsu_IWCD_v2.pdf`

[25] Fernandez, P., & Super-Kamiokande collaboration. (2016). Status of gadzooks!: Neutron tagging in super-kamiokande. Nuclear and particle physics proceedings, 273, 353-360.

[26] Vagins, M., Ishino, H., & Collaboration, S. K. (2012). GADZOOKS!. Phys. Rev. Lett, 108, 052505.

[27] Watanabe, H., Zhang, H., Abe, K., Hayato, Y., Iida, T., Ikeda, M., ... & Super-Kamiokande Collaboration. (2009). First study of neutron tagging with a water Cherenkov detector. Astroparticle Physics, 31(4), 320-328.

[28] Vagins, M., Ishino, H., & Collaboration, S. K. (2012). GADZOOKS!. Phys. Rev. Lett, 108, 052505.

[29] Hagiwara, K., Yano, T., Tanaka, T., Reen, M. S., Das, P. K., Lorenz, S., ... & Collazuol, G. (2019). Gamma-ray spectrum from thermal neutron capture on gadolinium-157. Progress of Theoretical and Experimental Physics, 2019(2), 023D01.

[30] Bourilkov, D. (2019). Machine and deep learning applications in particle physics. International Journal of Modern Physics A, 34(35), 1930019.

[31] Radovic, A., Williams, M., Rousseau, D., Kagan, M., Bonacorsi, D., Himmel, A., ... & Wongjirad, T. (2018). Machine learning at the energy and intensity frontiers of particle physics. Nature, 560(7716), 41-48.

[32] Guest, D., Cranmer, K., & Whiteson, D. (2018). Deep learning and its application to LHC physics. Annual Review of Nuclear and Particle Science, 68, 161-181.

[33] Carleo, G., Cirac, I., Cranmer, K., Daudet, L., Schuld, M., Tishby, N., ... & Zdeborová, L. (2019). Machine learning and the physical sciences. Reviews of Modern Physics, 91(4), 045002.

[34] Roe, B. P., Yang, H. J., Zhu, J., Liu, Y., Stancu, I., & McGregor, G. (2005). Boosted decision trees as an alternative to artificial neural networks for particle identification. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 543(2-3), 577-584.

[35] Gligorov, V. V., & Williams, M. (2013). Efficient, reliable and fast high-level triggering using a bonsai boosted decision tree. Journal of Instrumentation, 8(02), P02013.

[36] Cornell, A. S., Doorsamy, W., Fuks, B., Harmsen, G., & Mason, L. (2021). Boosted decision trees in the era of new physics: a smuon analysis case study. arXiv preprint arXiv:2109.11815.

[37] Schwartz, M. D. (2021). Modern Machine Learning and Particle Physics. arXiv preprint arXiv:2103.12226.

[38] Andrews, M., Paulini, M., Gleyzer, S., & Poczos, B. (2020). End-to-end physics event classification with CMS open data: Applying image-based deep learning to detector data for the direct classification of collision events at the LHC. Computing and Software for Big Science, 4(1), 1-14.

[39] Macaluso, S., & Shih, D. (2018). Pulling out all the tops with computer vision and deep learning. Journal of High Energy Physics, 2018(10), 1-27.

[40] Brickwedde, B., & Nachman, B. P. (2020). Convolutional neural networks with event images for pileup mitigation (No. ATL-PHYS-SLIDE-2020-030). ATL-COM-PHYS-2020-001.

[41] ATLAS Collaboration. Identification of Jets Containing b-Hadrons with Recurrent Neural Networks at the ATLAS Experiment. 2017.

[42] CMS Collaboration. (2020). Identification of heavy, energetic, hadronically decaying particles using machine-learning techniques.

[43] Note, A. P. (2020). Deep Sets based Neural Networks for Impact Parameter Flavour Tagging in ATLAS.

[44] Shlomi, J., Battaglia, P., & Vlimant, J. R. (2020). Graph neural networks in particle physics. Machine Learning: Science and Technology, 2(2), 021001.

[45] Qu, H., & Gouskos, L. (2020). Jet tagging via particle clouds. Physical Review D, 101(5), 056019.

[46] Henrion, I., Brehmer, J., Bruna, J., Cho, K., Cranmer, K., Louppe, G., & Rochette, G. (2017). Neural message passing for jet physics.

[47] Choma, N., Monti, F., Gerhardt, L., Palczewski, T., Ronaghi, Z., Prabhat, P., ... & Bruna, J. (2018, December). Graph neural networks for icecube signal classification. In 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 386-391). IEEE.

[48] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd International Conference on Knowledge Discovery and Data Mining (pp. 785-794).

[49] Parsa, A. B., Movahedi, A., Taghipour, H., Derrible, S., & Mohammadian, A. K. (2020). Toward safer highways, application of XGBoost and SHAP for real-time accident detection and feature analysis. Accident Analysis & Prevention, 136, 105405.

[50] Tahmassebi, A., Wengert, G. J., Helbich, T. H., Bago-Horvath, Z., Alaei, S., Bartsch, R., ... & Pinker, K. (2019). Impact of machine learning with multiparametric magnetic resonance imaging of the breast for early prediction of response to neoadjuvant chemotherapy and survival outcomes in breast cancer patients. Investigative Radiology, 54(2), 110.

[51] Bhattacharya, S., Maddikunta, P. K. R., Kaluri, R., Singh, S., Gadekallu, T. R., Alazab, M., & Tariq, U. (2020). A novel PCA-firefly based XGBoost

classification model for intrusion detection in networks using GPU. Electronics, 9(2), 219.

[52] Chen, T., & He, T. (2015, August). Higgs boson discovery with boosted trees. In NIPS 2014 workshop on high-energy physics and machine learning (pp. 69-80). PMLR.

[53] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. Annals of statistics, 1189-1232.

[54] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd International Conference on Knowledge Discovery and Data Mining (pp. 785-794).

[55] Mitchell, R., & Frank, E. (2017). Accelerating the XGBoost algorithm using GPU computing. PeerJ Computer Science, 3, e127.

[56] Lundberg, S. M., & Lee, S. I. (2017, December). A unified approach to interpreting model predictions. In Proceedings of the 31st international conference on neural information processing systems (pp. 4768-4777).

[57] Shapley, L. S. (1953). Stochastic games. Proceedings of the national academy of sciences, 39(10), 1095-1100.

[58] Winter, E. (2002). The shapley value. Handbook of game theory with economic applications, 3, 2025-2054.

[59] LeNail, (2019). NN-SVG: Publication-Ready Neural Network Architecture Schematics. Journal of Open Source Software, 4(33), 747, https://doi.org/10.21105/joss.00747

[60] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

[61] Aloysius, N., & Geetha, M. (2017, April). A review on deep convolutional neural networks. In 2017 International Conference on Communication and Signal Processing (ICCSP) (pp. 0588-0592). IEEE.

[62] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 1097-1105.

[63] Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010, September). Recurrent neural network based language model. In Interspeech (Vol. 2, No. 3, pp. 1045-1048).

[64] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., ... & Sun, M. (2020). Graph neural networks: A review of methods and applications. AI Open, 1, 57-81.

[65] Aurisano, A., Radovic, A., Rocco, D., Himmel, A., Messier, M. D., Niner, E., ... & Vahle, P. (2016). A convolutional neural network neutrino event classifier. Journal of Instrumentation, 11(09), P09001.

[66] Abhishek, A., Fedorko, W., de Perio, P., Prouse, N., & Ding, J. Z. (2019). Variational Autoencoders for Generative Modelling of Water Cherenkov Detectors. arXiv preprint arXiv:1911.02369.

[67] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The graph neural network model. IEEE transactions on neural networks, 20(1), 61-80.

[68] Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203.

[69] Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems, 29, 3844-3852.

[70] Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

[71] Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., & Bronstein, M. M. (2017). Geometric deep learning on graphs and manifolds using mixture model cnns. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5115-5124).

[72] Ma, Y., & Tang, J. (2021). Deep learning on graphs. Cambridge University Press.

[73] Hammond, David K., Pierre Vandergheynst, and Rémi Gribonval. "Wavelets on graphs via spectral graph theory." Applied and Computational Harmonic Analysis 30.2 (2011): 129-150.

[74] Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., & Solomon, J. M. (2019). Dynamic graph cnn for learning on point clouds. Acm Transactions On Graphics (tog), 38(5), 1-12.

[75] Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 652-660).

[76] GitHub repository for WCSim: https://github.com/WCSim/WCSim

[77] Agostinelli, S., Allison, J., Amako, K. A., Apostolakis, J., Araujo, H., Arce, P., ... & Geant4 Collaboration. (2003). GEANT4—a simulation toolkit. Nuclear instruments and methods in physics research section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 506(3), 250-303.

[78] Rene Brun & Fons Rademakers, ROOT - An Object Oriented Data Analysis Framework, Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86. See also http://root.cern.ch/.

[79] Baldwin, S. (2012, February). Compute Canada: advancing computational research. In Journal of Physics: Conference Series (Vol. 341, No. 1, p. 012001). IOP Publishing.

[80] GitHub repository for WCSim: https://github.com/nuPRISM

[81] Wright, D., & Incerti, S. (2012). A short guide to choosing physics lists. Geant4 Tutorial at Jefferson Lab, SLAC.

[82] Bernard, L. (2019). Spallation background in the Super-Kamiokande experiment. Super-Kamiokande Collaboration ICHEP conference, neutrino session.

[83] Li, S. W., & Beacom, J. F. (2014). First calculation of cosmic-ray muon spallation backgrounds for MeV astrophysical neutrino signals in Super-Kamiokande. Physical Review C, 89(4), 045801.

[84] Marin, F., Rohatgi, A., & Charlot, S. (2017). WebPlotDigitizer, a polyvalent and free software to extract spectra from old astronomical publications: application to ultraviolet spectropolarimetry. arXiv preprint arXiv:1708.02025.

[85] Abe, K. "Neutron Tagging Following Atmospheric Neutrino Events in a Water Cherenkov Detector." Prog. Theor. Exp. Phys., 2013.

[86] Wilson, J. R. (2015). An Experimental Review of Solar Neutrinos. arXiv preprint 1504.04281.

[87] Dunmore, Jessica A. (2004). The Separation of CC and NC Events in the Sudbury Neutrino Observatory

[88] Abe, K., Haga, Y., Hayato, Y., Ikeda, et al. (2013). Neutron Tagging following Atmospheric Neutrino Events in a Water Cherenkov Detector. Prog. Theor. Exp. Phys. PTEP (and similar studies)

[89] Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. arXiv preprint arXiv:1903.02428.

[90] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[91] Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (pp. 448-456). PMLR.

[92] Thekumparampil, K. K., Wang, C., Oh, S., & Li, L. J. (2018). Attention-based graph neural network for semi-supervised learning. arXiv preprint arXiv:1803.03735.

[93] Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., & Weinberger, K. (2019, May). Simplifying graph convolutional networks. In International conference on machine learning (pp. 6861-6871). PMLR.

[94] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.

Appendix

Sections 5 and 6 present the results of feature engineering and SHAP analysis for the neutron capture and spallation electron background dataset. The spallation dataset was included in the main thesis body as was designed to be the most realistic of the three datasets, and hopefully the most representative of real neutron capture and background physics events in a water Cherenkov detector. For the sake of space, the corresponding figures for the other two datasets, lowE and highE, and presented here in the appendix.

Subsection 8.1 presents the beta parameter histograms, overall engineered feature histograms, XGB confusion matrix, SHAP beeswarm plot and feature importance plot for the highE dataset in Figs. 8.1, 8.2, 8.3, 8.4 and 8.5. The corresponding plots for the lowE dataset are then shown in Subsection 8.2 in Figs. 8.6, 8.7, 8.8, 8.9 and 8.10. Subsection 8.3 contains the final supplementary material, including a plot of the feature importance distributions for the lowE XGBoost model arranged according to the different metrics of weight, gain and cover in Fig. 8.11.

8.1 highE Dataset Supplementary

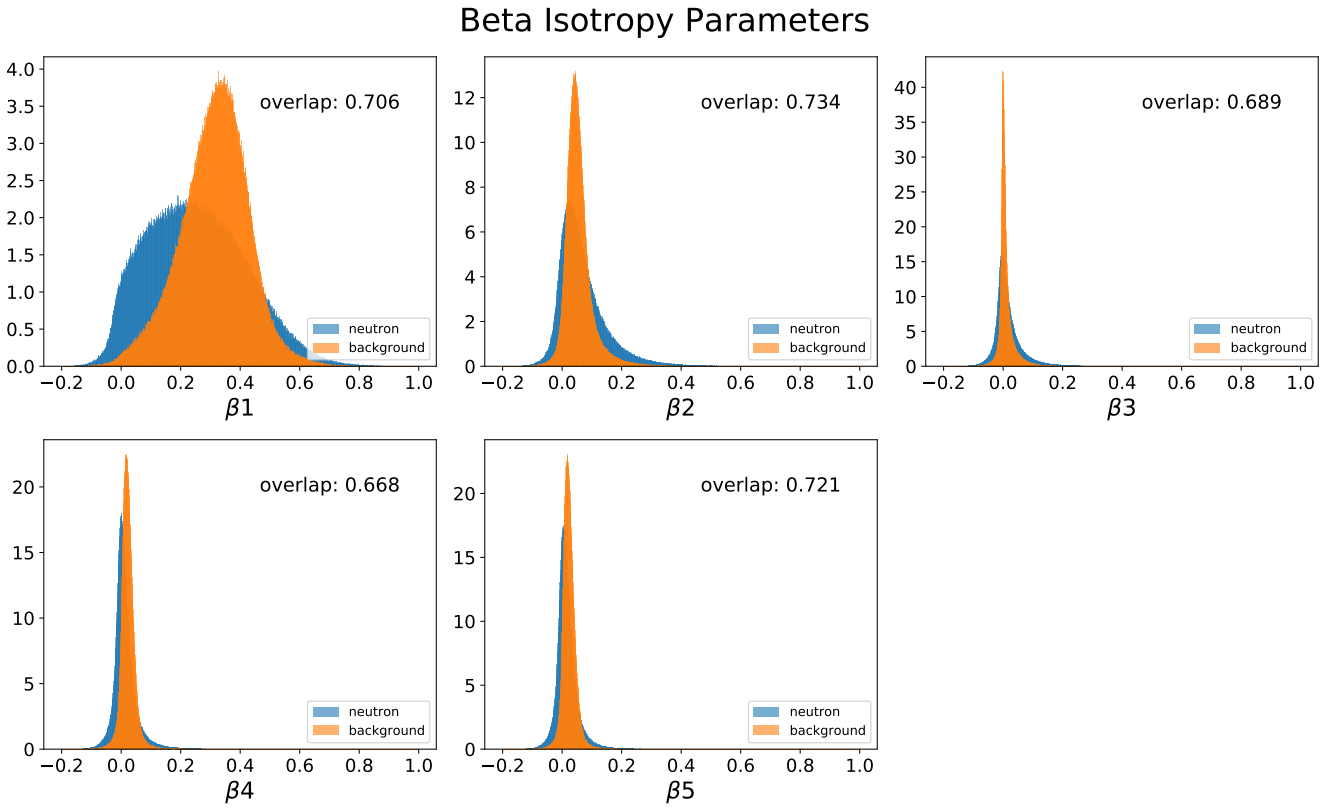


Figure 8.1: Distributions of the beta parameters $\beta 1$ to $\beta 5$ are shown for the dataset of neutron captures and high energy electron background (0-20MeV), simulated using WCSim for the IWCD short tank geometry. The amount of overlap between the histograms is shown by the number in the top-right corner of every subplot. $\beta 4$ and $\beta 3$ have the least overlaps of 66.8% and 68.9% respectively.

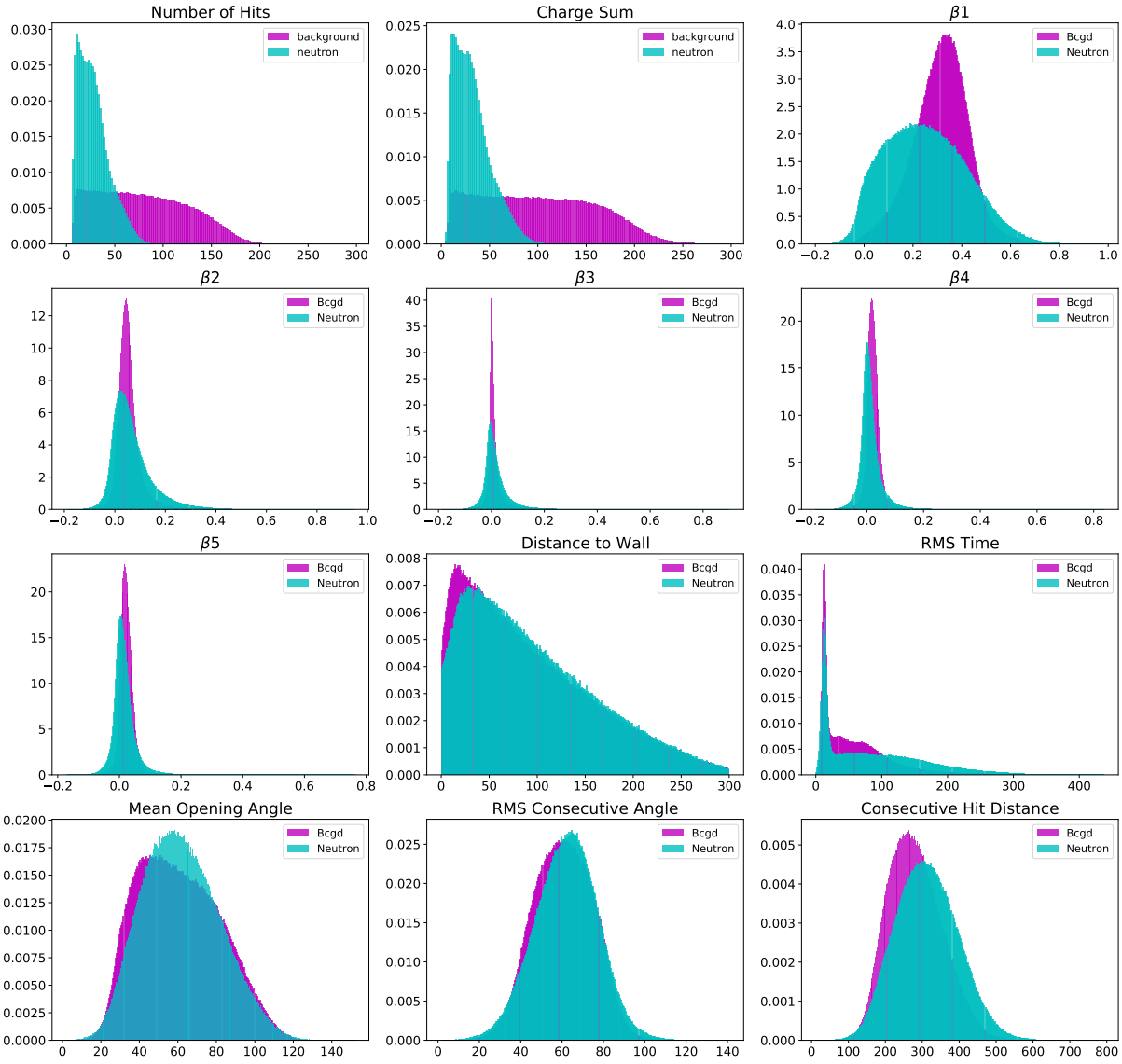


Figure 8.2: Comparison of twelve engineered features separated by neutron capture and high energy electron background events. The data consists of nearly 1.6 million events, generated by WCSim for the IWCD detector geometry. Relative to the low energy and spallation background datasets, there is the most discrimination between signal and background for this dataset.

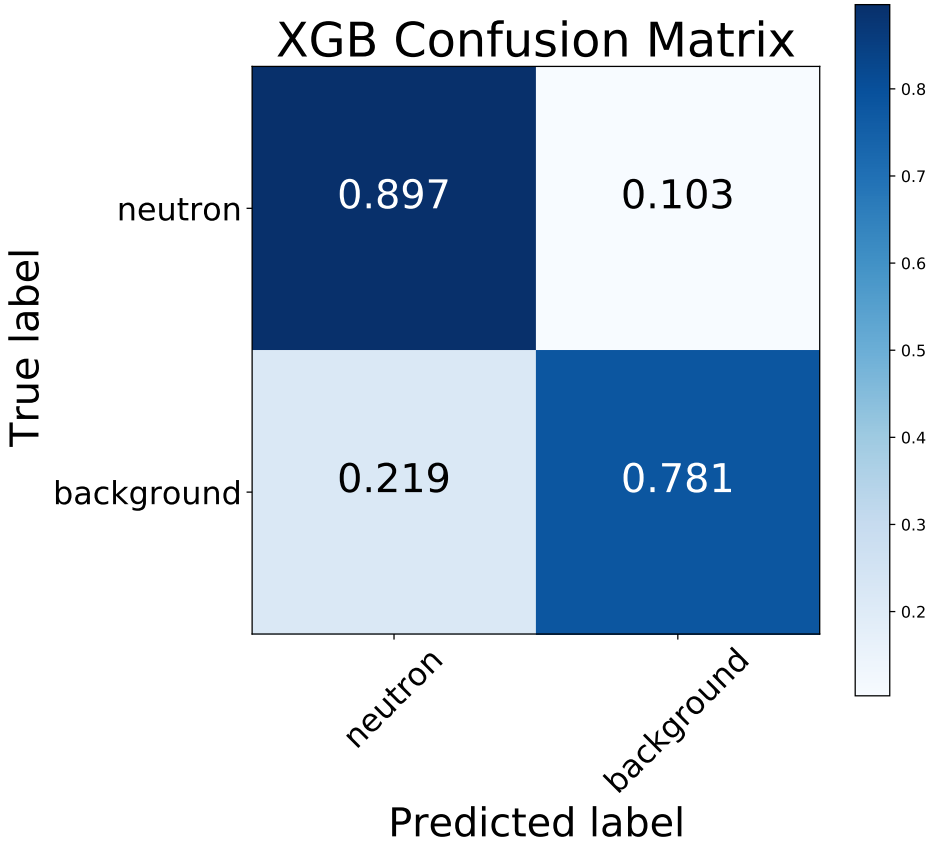


Figure 8.3: Confusion matrix for the XGBoost model trained on the dataset of neutron capture and high energy background electron events. The neutron recall rate exceeds the electron recall rate, indicating the neutron capture events are simpler to identify for this dataset.

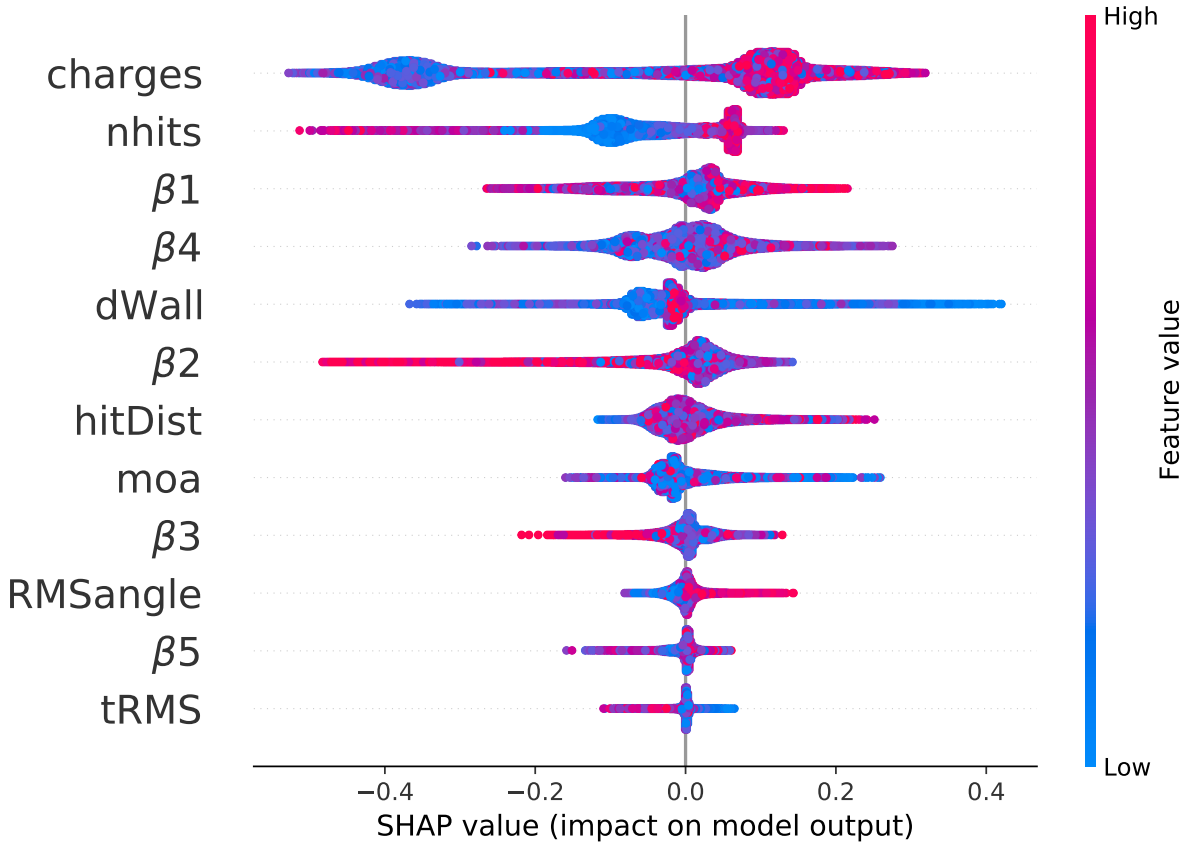


Figure 8.4: Beeswarm plot of SHAP values for the neutron capture and high energy background dataset simulated using WCSim for the IWCD tank geometry. The SHAP value for each feature in every event is plotted as a dot in the plot, where the x axis position corresponds to the SHAP value and the colorbar shows the feature value (blue is low, red is high). High SHAP values influence the model output towards 1 (electron-like event) and low SHAP values (negative) influence the model outputs toward 0 (neutron-like event).

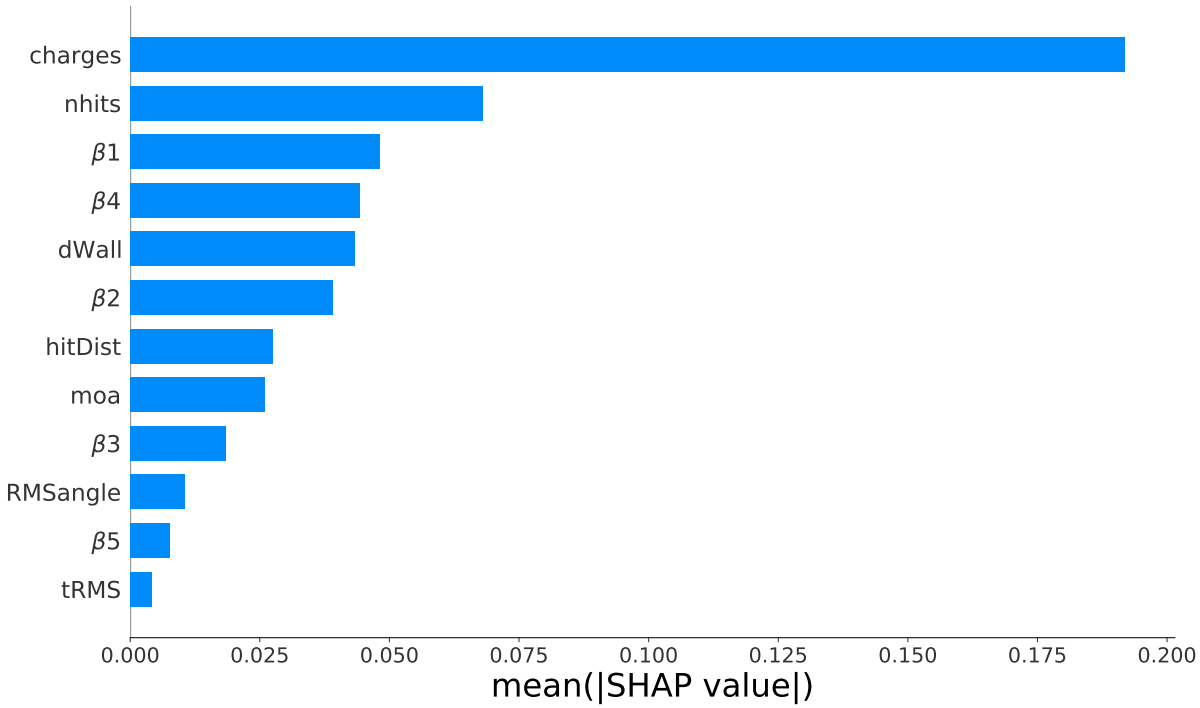


Figure 8.5: The relative feature importances are shown for the neutron capture and high energy electron background dataset, ranked by average absolute SHAP value. The charge sum is clearly the biggest predictor of the model outcome result, followed by the number of hits in the event. The relative SHAP importances demonstrate the XGBoost is learning mostly from a couple features and constructing a relatively simple model.

8.2 lowE Dataset Supplementary

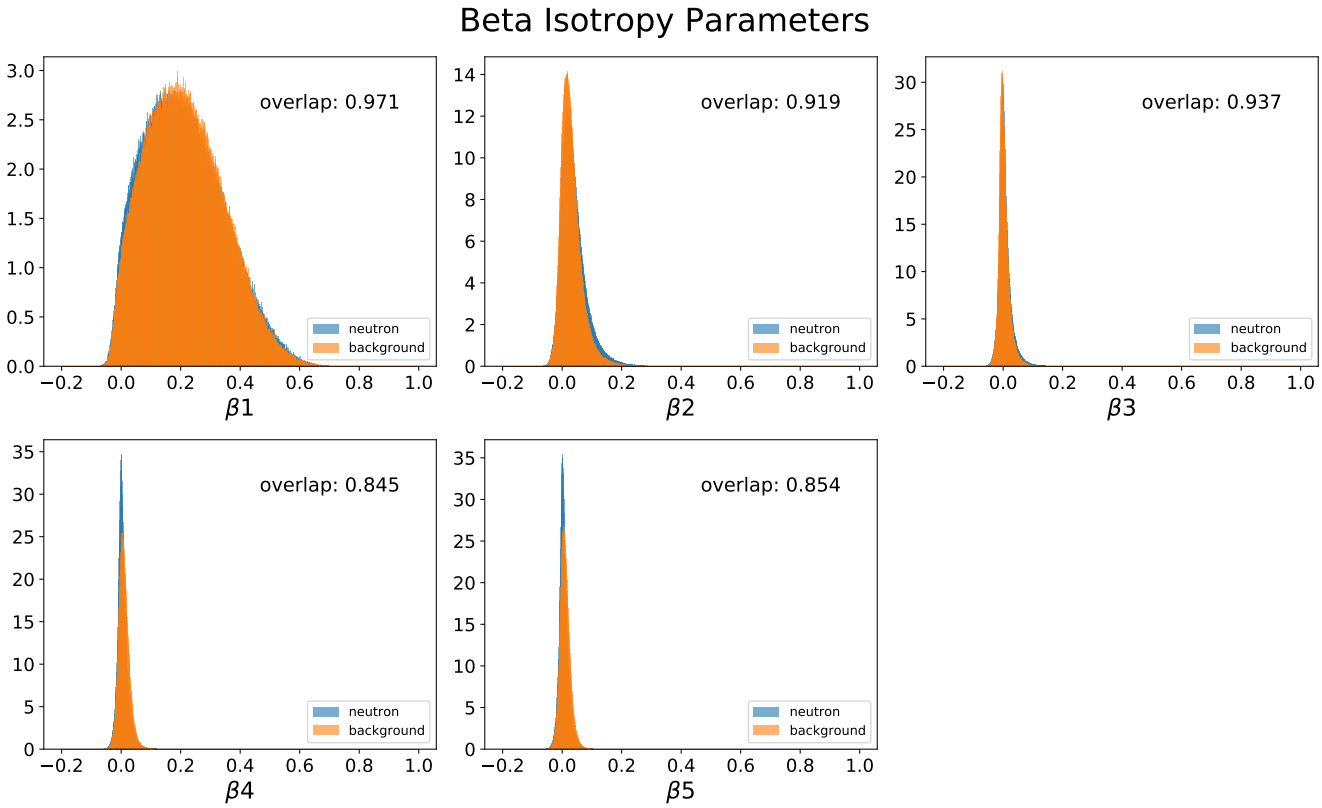


Figure 8.6: Distributions of the beta parameters $\beta 1$ to $\beta 5$ are shown for the dataset of neutron capture and low energy background electron events (0-8MeV), generated by WCSim for the IWCD short tank geometry. The amount of overlap between the histograms is shown by the number in the top-left corner for every subplot. $\beta 1$ through $\beta 3$ have overlaps above 90% while $\beta 4$ and $\beta 5$ have the least overlap of 84.5% and 85.4% respectively.

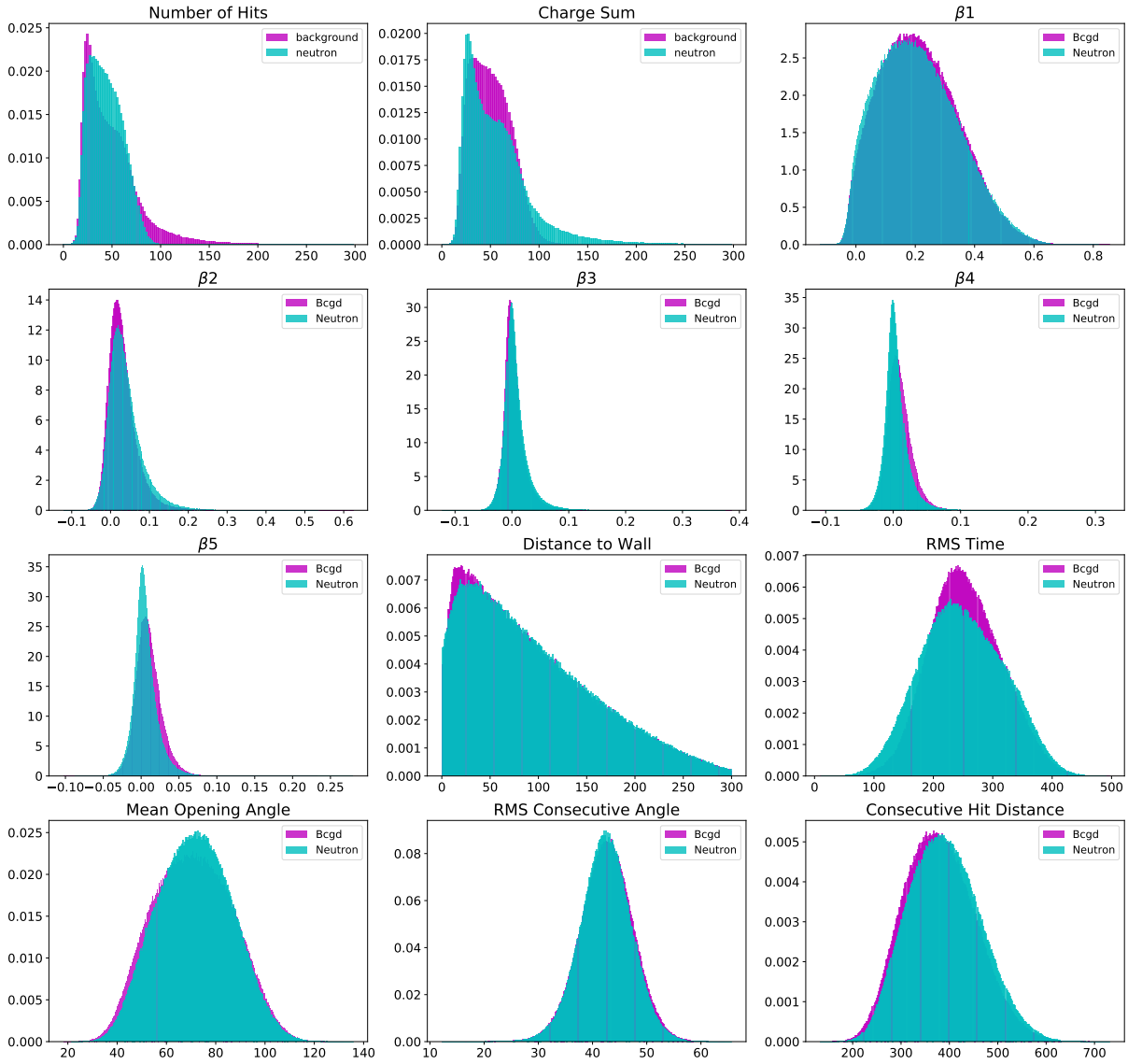


Figure 8.7: Comparison of twelve engineered features separated by neutron capture and low energy electron background events. The data consists of nearly 1.6 million events, generated by WCSim for the IWCD detector geometry. Relative to the high energy and spallation background datasets, there is the least discrimination between signal and background for this dataset.

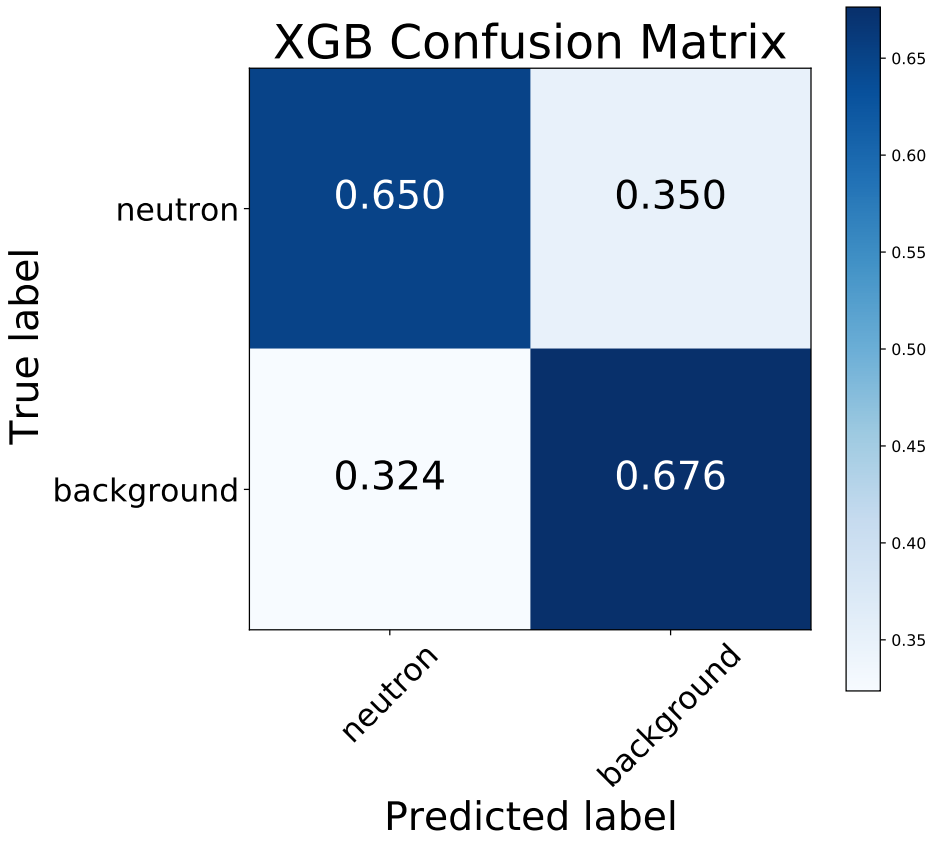


Figure 8.8: Confusion matrix for the XGBoost model trained on the dataset of neutron capture and low energy background electron events. The electron recall rate exceeds the neutron recall rate, indicating the neutron capture events are more difficult to identify correctly for this dataset.

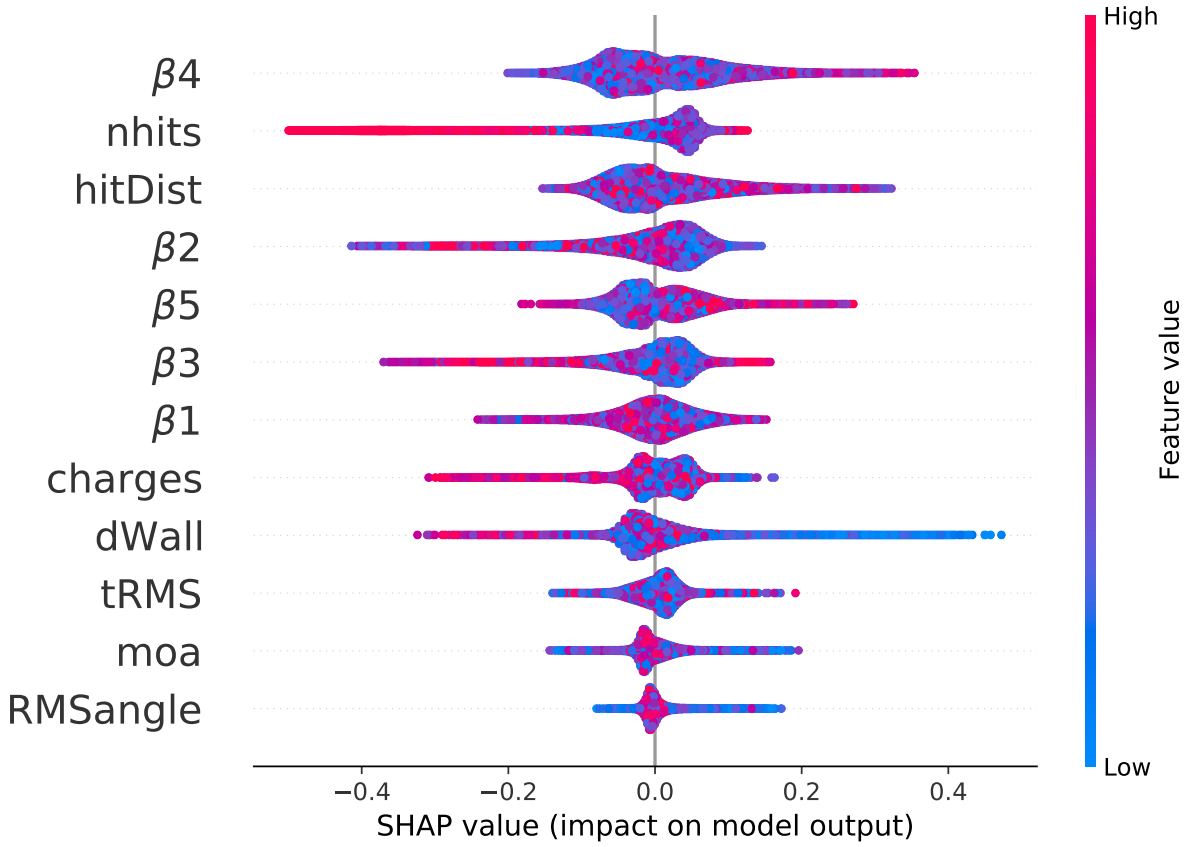


Figure 8.9: Beeswarm plot of SHAP values for the neutron capture and low energy background dataset simulated using WCSim for the IWCD tank geometry. The SHAP value for each feature in every event is plotted as a dot in the plot, where the x axis position corresponds to the SHAP value and the colorbar shows the feature value (blue is low, red is high). High SHAP values influence the model output towards 1 (electron-like event) and low SHAP values (negative) influence the model outputs toward 0 (neutron-like event).

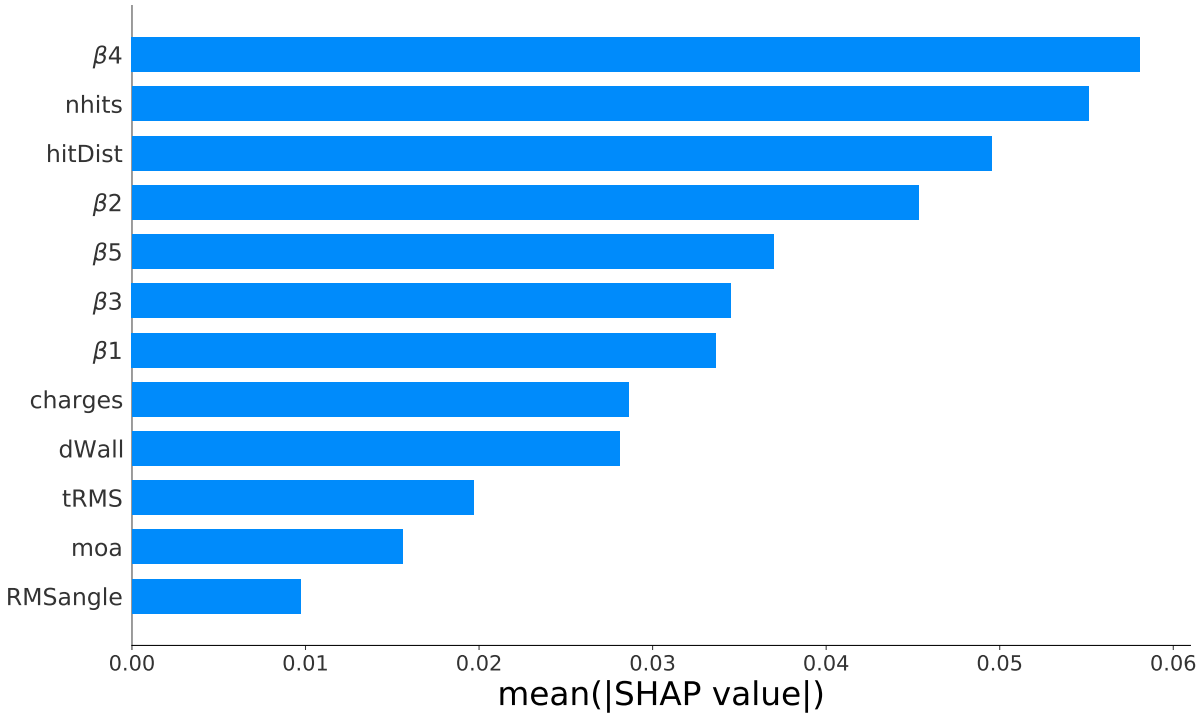
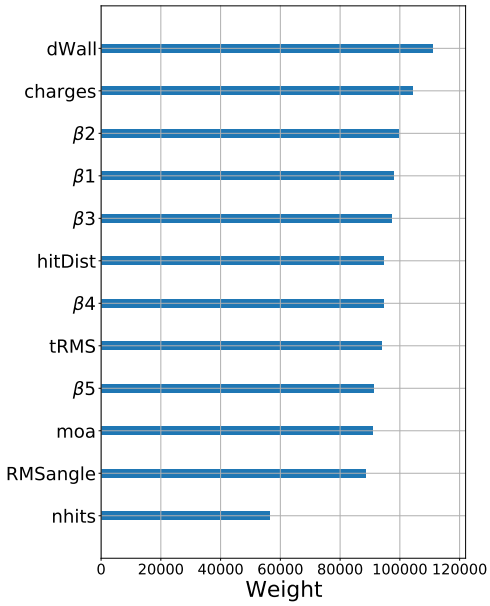
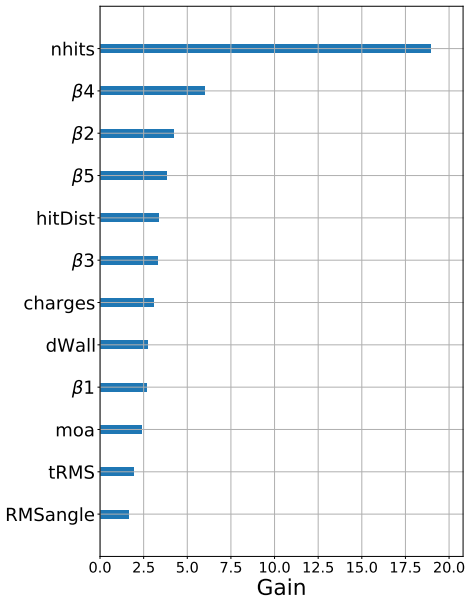


Figure 8.10: The relative feature importances are shown for the neutron capture and low energy electron background dataset, ranked by average absolute SHAP value. These results show that the beta parameters (mostly $\beta 4$), the event number of hits, and the average distance between consecutive PMT hits in a given event are the most importance features for the XGBoost model to predict event outcomes.

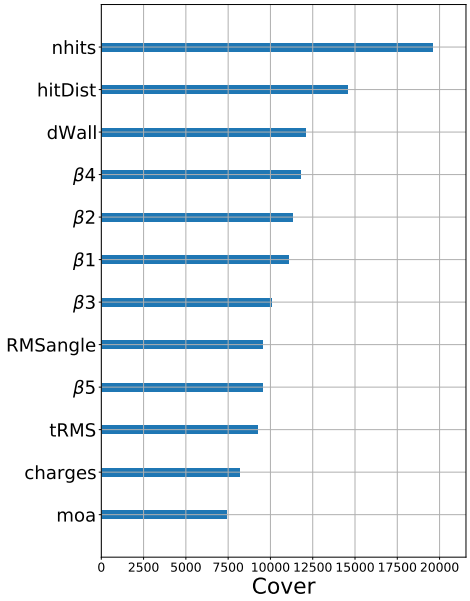
8.3 Other



(a) Importance by Weight



(b) Importance by Gain



(c) Importance by Cover

Figure 8.11: Feature importances are shown for the XGBoost model trained on the IWCD simulated neutron capture and low energy electron background dataset. The feature are shown sorted by weight, gain and cover. The resulting importances are inconsistent with each other, and do not satisfy the properties of the SHAP values (accuracy, missingness and consistency), making it difficult to draw conclusions from the above plots.