

University of South Dakota

**USD RED**

---

Dissertations and Theses

Theses, Dissertations, and Student Projects

---

2023

## Background Discrimination of a Neutrino Detector with Dense Neural Networks

Perry Siehien

Follow this and additional works at: <https://red.library.usd.edu/diss-thesis>



Part of the [Computer Sciences Commons](#), and the [Elementary Particles and Fields and String Theory Commons](#)

---

# BACKGROUND DISCRIMINATION OF A NEUTRINO DETECTOR WITH DENSE NEURAL NETWORKS

By  
Perry Siehien

B.S., Washington and Lee University, 2020

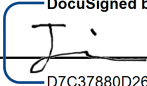
A Thesis Submitted in Partial Fulfillment of the  
Requirements for the Degree of Master of  
Science

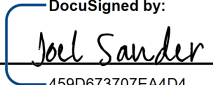
---

Department of Physics

Analytics for Large Data Sets Program  
In the Graduate School  
The University of South Dakota  
May 2023

The members of the Committee appointed to examine  
the Thesis of Perry Siehien  
find it satisfactory and recommend that it be accepted.

DocuSigned by:  
  
Chairperson  
D7C37880D26F476...


DocuSigned by:  
  
Joel Sander  
459D673707EA4D4...

DocuSigned by:  
  
K. Santosh  
BBEF270EA578492...

## ABSTRACT

Neutrinos are subatomic particles that weakly interact with matter due to their neutral charge and small cross section. Detectors that search for neutrinos require sensitive instrumentation, which makes them susceptible to various background sources such as gamma rays. Additionally, coherent elastic neutrino-nucleus scattering events, or CEvNS, are the weakest neutrino interactions at 1-25 keV, making them exceptionally difficult to observe. To understand the physics of CEvNS events within the detector material, the recoil signatures of relevant interactions must be determined. Traditional analysis methods are effective, but cannot be applied to energies below 50 keV, due to the overlap of discrimination criteria. In this thesis, we investigate the effectiveness of applied neural networks to distinguish between two recoil signatures present in COHERENT's CENNS-10 LAr detector. Results indicate that dense neural networks perform well on classifying low energy events that approach the CEvNS energy threshold. We also discuss modifications to the complexity and structure of the neural network, which may improve generality.

Thesis Advisor

  
Dr. Jing Liu

# Acknowledgements

I would like to extend my gratitude to all those who supported me throughout this project. I am certain without their efforts, this work would not have been possible. A huge thanks to Dr. Jing Liu, whose guidance on this thesis was invaluable. As my advisor, you were always able to show a clear path forward and spark motivation for our research. I attribute much of our success to his involvement.

Secondly, I would like to thank Dr. Joel Sander and Dr. KC Santosh for their input along the way. The conversations we had always inspired academic or professional growth, and I am deeply grateful for the opportunity to work with them both.

I would also like to give a big thanks to the COHERENT Collaboration. Without all their hard work in this field, this research would not be possible. The feedback you provided was greatly appreciated, and largely inspired our future work.

Lastly, I want to thank my friends and family who have supported me throughout my academic career. My parents, Michael and Lauren, have been a huge source of encouragement and inspiration, and I cannot thank them enough for everything they have done. I'd also like to thank my girlfriend, Nadia, for always motivating me to be my best. I couldn't of done it without you.

Thank you again to all those named, and unnamed, that helped me along. I will always remember this time in my life, and I am glad to have spent it with you all.

# Contents

Committee Signature Page	i
Abstract	ii
Acknowledgements	iii
Contents	iv
List of Figures	v
<b>1 Introduction</b>	<b>1</b>
1.1 Neutrinos and CEvNS . . . . .	1
1.2 CENNS-10 LAr Detector . . . . .	3
1.3 Recoil Discrimination . . . . .	7
<b>2 Literature Review</b>	<b>12</b>
2.1 Observing CEvNS . . . . .	12
2.2 CsI and LAr . . . . .	13
2.3 Reactor CEvNS . . . . .	15
2.4 Discrimination Methods . . . . .	16
<b>3 Method</b>	<b>19</b>
<b>4 Results</b>	<b>25</b>
<b>5 Conclusion</b>	<b>29</b>
<b>6 Current Efforts</b>	<b>30</b>
<b>References</b>	<b>35</b>
<b>7 Appendix</b>	<b>38</b>
A Codes . . . . .	38
B Package Details . . . . .	47

# List of Figures

1	Detector Suite at ORNL-SNS. . . . .	2
2	Construction diagram for the CENNS-10 LAr detector. . . . .	3
3	Electronic recoil mechanisms. . . . .	5
4	Interaction likelihood as a function of energy vs. $Z$ . . . . .	6
5	Nuclear recoil mechanism. . . . .	7
6	F90 plot of a AmBe calibration run. Distinct groupings highlight the decay behavior for both recoils. . . . .	8
7	Zoomed in region between 0-60 ADC of selected AmBe run, where the X-axis is converted to total PEs. Sampled regions are labeled by energy for future reference, with gamma and neutron bands highlighted. Region IV is an area of interest where F90 cannot be applied. . . . .	9
8	Nuclear and electronic recoil samples. Left samples are nuclear recoils descending in energy. Right samples are electronic recoils descending in energy. Due to normalization methods, the peak height may be larger for less energetic samples. . . . .	10
9	Nuclear recoil samples from separate AmBe calibration runs. Samples shown are taken from the same PE count in region I. . . . .	11
10	Electronic recoil samples from separate Co-57 calibration runs. Samples shown are taken from the same PE count in region I. . . . .	11
11	Flux-averaged CEvNS cross section as a function of neutron count[14]. The green curve represents Klein-Nystrand form factor, while the black curve assumes unity. . . . .	14
12	Generalized structure of our dense network. $X'$ and $X''$ are the number of nodes in Dense Layers I and II, respectively. . . . .	21
13	Averaged waveform of each class from our highest energy samples. . . . .	22
14	Fitted Gaussian peak to single PE. . . . .	23
15	Fitted Gaussian profile. . . . .	24
16	Fitted Gaussian profile, downsampled to real data sample rate. Axis change from downsampling and lead time. Final peak used is cut from bins 46-54. . . . .	24
17	Performance on training data from region I. Accuracy reaches 99% at epoch 12. . . . .	25
18	Performance on training data from region II. Accuracy reaches 99% at epoch 17. . . . .	26
19	Performance on training data from region III. Accuracy reaches 98% at epoch 20. Retraining with fresh weights finds 99% at epoch 25. . . . .	26
20	Simulated samples from energy region IV in Figure 7. Samples contain 20 PEs. . . . .	27
21	Performance on 24-40 PE simulated data, our highest energy range correlating to region IV. Asymptotic accuracy of 89.5% observed in epoch 20. Increase training time observed negligible accuracy gain, while loss plateaus around .35. . . . .	27
22	Time-shifted recoil vs calibration recoil of similar energy. . . . .	30
23	1-Dimensional Convolution. Filter performs elementwise multiplication then sums the shaded cells. The filter steps forward one input and repeats the operation. . . . .	32

24	Max Pooling Node. Only the largest value (shaded) from a given subset is kept for the next layer. Subsets may or may not overlap, depending on your parameter selection. . . . .	33
----	--	----



# 1 Introduction

## 1.1 Neutrinos and CEvNS

The direct observation of neutrinos is a long standing effort within the physics community since their proposal in 1930 by Wolfgang Pauli. Various detector technologies have been developed in the pursuit of determining the unknown properties of neutrinos, such as mass and antiparticle equivalents. Much like other subatomic particles, how neutrinos interact with matter depends on their energy and the incident nuclei. The Standard Model makes predictions on the properties of these interactions, which can be tested against the results of neutrino experiments. The validation of the Standard Model is a major focus within physics research, motivating the development of novel methods to detect neutrinos.

One type of neutrino interaction is called coherent elastic neutrino-nucleus scattering, or CEvNS. Much like classical elastic collisions, the neutrino collides with the nucleus of an atom resulting in a small exchange of energy. At this energy range, the neutrino sees the entire nucleus coherently, instead of individual nucleons. This coherence is predicted by the Standard Model to have a cross-section dependence on the number of nucleons squared ( $N^2$ ), which can be experimentally tested. This interaction was first proposed in 1974 by Freedman, which went undetected for nearly 50 years. These CEvNS events are the lowest energy interactions between neutrinos and nuclei with a range of 1 to 25 keV, making them difficult to detect. Researchers searching for CEvNS described this interaction as trying to “measure an elephant’s recoil after being bumped by a mosquito”[1]. Standard instruments lack the sensitivity to resolve this collision, so new detectors needed to be developed specifically for this search.

The COHERENT neutrino research group was founded in 2013 with the goal to make the first direct observation of a coherent elastic neutrino-nucleus scattering (CEvNS) event. The collaboration has members from 21 institutions across 4 countries, and successfully observed a CEvNS event with their CsI[Na] crystal (Cesium-Iodide) detector in 2017[2]. Along

with the CsI[Na] detector, COHERENT manages several detector systems housed at Oak Ridge National Laboratory (ORNL), which all look to further CEvNS event detection and research. These include a suite of state-of-the-art instruments including NUBES (neutrino cubes), high-purity germanium (HPGe), NaI[Ti], and CENNS-10 LAr (liquid argon) detectors. These detectors are housed in an underground facility at ORNL, called Neutrino Alley, near the Spallation Neutron Source (SNS). The SNS provides a consistent neutron/neutrino profile by colliding liquid mercury with accelerated protons, which serves as the primary source of neutrinos for COHERENT’s detectors.

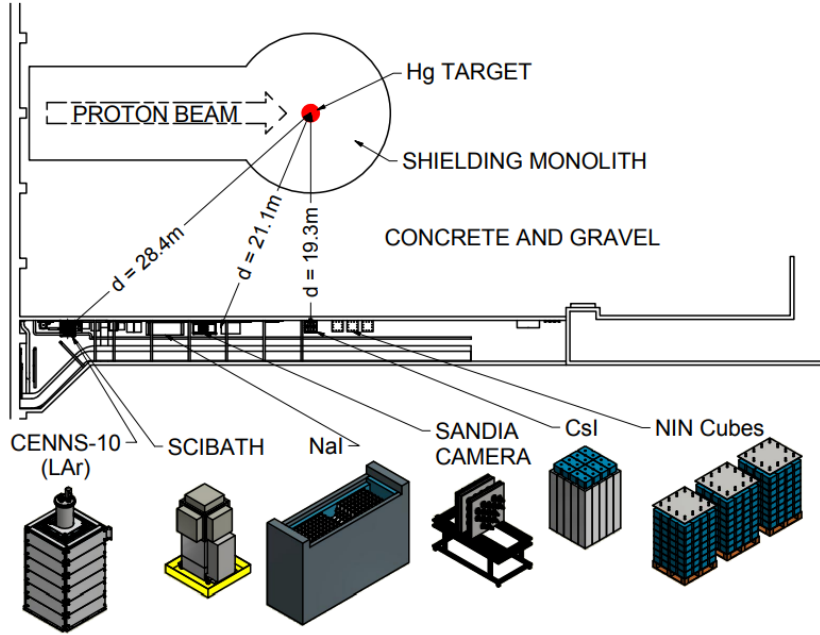


Figure 1: Detector Suite at ORNL-SNS.

The COHERENT detector suite has been operational for several years, with pauses in data acquisition (DAQ) for maintenance, upgrades, or system replacement. This consistent up-time generates a large amount of data to process, even for a single detector. To date, COHERENT manages and maintains over 1600 terabytes (1.6 petabytes) of collected data in a high-performance cluster. Event detection methods and DAQ can also vary, such

as the HPGe recording phonons, or CENNS-10 utilizing photo-multiplier tubes to record scintillation. In our research, we focus on data from the CENNS-10 LAr detector.

## 1.2 CENNS-10 LAr Detector

The CENNS-10 detector[3] in Neutrino Alley is a complex system, relying on several key components to remain operational. The main target is a Teflon chamber coated with tetraphenyl butadiene (TPB), housing 56.7 L of liquid argon. This chamber is capped at the ends with TPB coated Hamamatsu photomultiplier tubes to collect a scintillation event, which is captured by a CAEN 250 MHz digitizer. The system is then encapsulated by copper and lead shielding and filled with water to mitigate steady-state gamma backgrounds and incident beam-related neutrons.

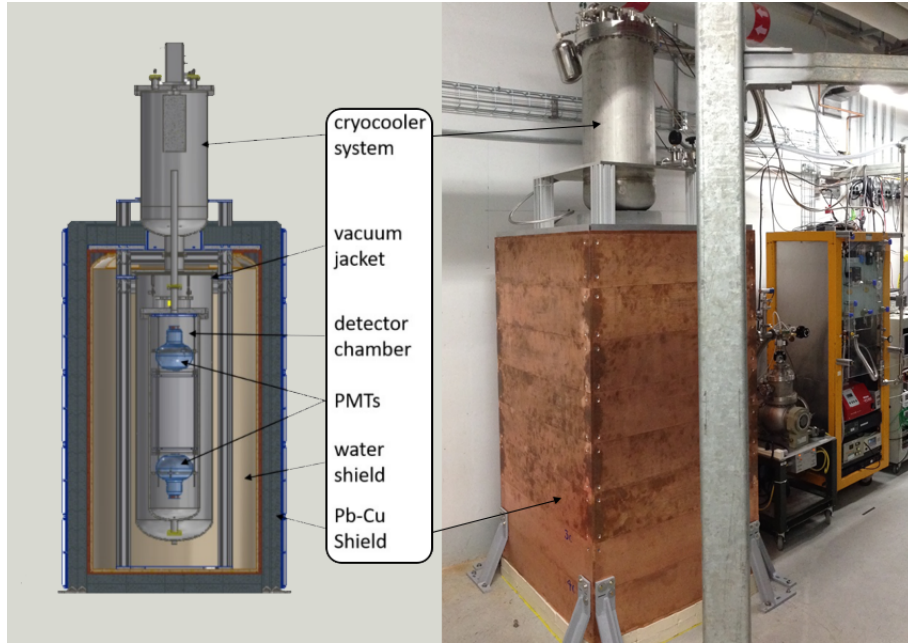


Figure 2: Construction diagram for the CENNS-10 LAr detector.

One challenge of operating a LAr detector is keeping the argon in its liquid form. For CENNS-10, this is handled by a top-mounted Cryomech PT-90 cryo-refrigerator which continuously circulates and cools the argon. To maintain purity standards within the target(1

ppm), the cooling system is connected to a SAES Zr which boils off and removes residual gases. A higher presence of contaminants in the argon will alter the scintillation behavior, with nuclear collisions or internal radioactivity producing unwanted photons.

The use of TPB coatings on the PMTs and target chamber is to shift the original scintillated light from 128 nm to the visible spectrum, which is then collected by the PMTs. The use of TPB coatings increased light output by a factor of 5, with the shifted light now in the range for observation in PMTs. A single photoelectron (PE) calibration is performed on a weekly basis to ensure normal functions in the system, which can remain stable for some time[4]. Understanding the response from a single PE is critical for analysis methods and serves as the baseline energy of a single count in the PMTs.

CENNS-10 is sensitive to two primary recoils; electronic and nuclear. Due to the resolution required to observe CEvNS events, electronic related backgrounds cannot be conventionally mitigated. Gamma rays are a product of many natural decay processes, like potassium-40 or from cosmic weather. Due to their high penetration in matter, gamma rays are difficult to completely abate, even with lead or tungsten [5]. We consider gamma events as electronic recoils since they exchange energy directly with electron orbitals.

There are three primary gamma-atom interactions that can take place within the detector which depend on the atomic number and energy of the incident photon. These interactions affect the amount of light produced in the target material and how energy is deposited. At the lowest energy around 100 keV, the Photoelectric Effect is the primary mechanism of energy transfer. In this process, all the energy of the gamma ray is transferred to a bound electron. This electron is ejected from its bound state with an energy equal to the gamma ray minus the electron's binding energy. If the electron's binding energy is known, an accurate measurement of the gamma's energy can be made.

$$\Delta\lambda = \frac{h}{mc}(1 - \cos\theta) \tag{1}$$

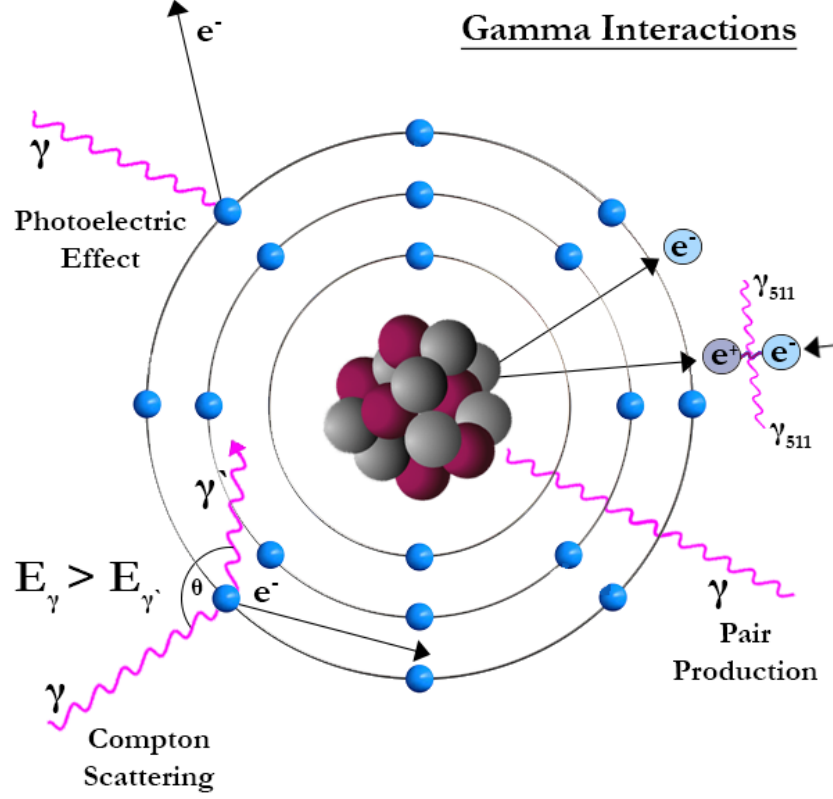


Figure 3: Electronic recoil mechanisms.

The second most energetic interaction takes place with gamma rays around 100-1,000 keV, which is called Compton Scattering. In this process, a photon deflects off weakly bound electrons, changing the gamma's direction and transferring some of its energy. The energy loss per interaction is given in Equation [1], and is converted to kinetic energy[6]. After each scattering event,  $\Delta\lambda$  is the change in wavelength of the incident gamma, in proportion to Planck's Constant over the electron rest mass and the speed of light. The amount of energy lost is highly dependent on the incident angle  $\theta$ , as a direct hit will theoretically transfer all its energy. If a direct hit is not made, the scattering can continue through material, where the gamma can eventually lose enough energy to be absorbed by a bound electron, as determined by the Photoelectric Effect.

High energy gamma rays above 1,022 keV that interact with an atom can produce electron-positron pairs, which annihilate to create two 511 keV gamma rays. This is an

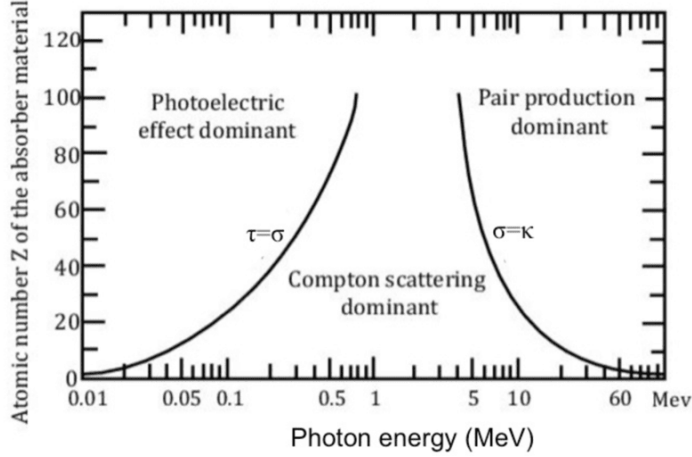


Figure 4: Interaction likelihood as a function of energy vs.  $Z$ .

interesting mechanism, as energy is converted directly into creating mass. For energies above 1,022 keV, the excess is transferred as momentum, resulting in a kinetic recoil. The probability of each of these interactions can be plotted as a function of energy to atomic number, as seen in Figure [4]. By calibrating CENNS-10 with a variety of gamma sources, such as krypton-83m or cobalt-57, we can observe the detector response across a range of energies. Since the primary background of CENNS-10 are gamma photons, it is necessary to understand these interactions.

A neutron background is also naturally present on Earth; mostly from cosmic muon induced spallation and natural fission within crustal deposits. Neutron interactions result in nuclear recoils, as the neutrally charged particle ignores electron orbitals and can strike the nucleus directly, observed in Figure [5]. Neutrons introduce another difficulty in observing a CEvNS event, as both neutron and neutrino interactions result in nuclear recoils. The resultant recoil waveforms share similar characteristics, as slow moving neutrons deposit energy similar to the energy deposit of high energy neutrinos[7]. One benefit of using the SNS as a neutrino source is that the neutron/neutrino profile is well studied, helping to differentiate these similar recoils. With two primary events observed in the detector, we need techniques to accurately discriminate between them if a CEvNS event is to be observed.

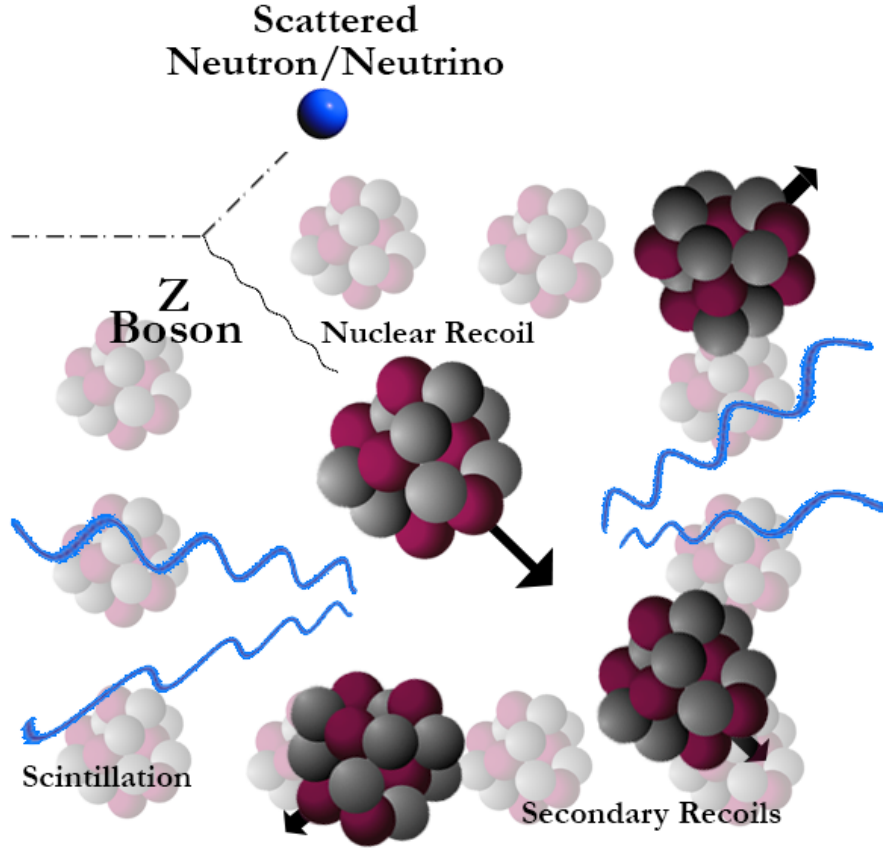


Figure 5: Nuclear recoil mechanism.

### 1.3 Recoil Discrimination

One method of classifying recoils is called the first 90 nanoseconds, or F90. This method involves computing the integral of the first 90 nanoseconds of a signal, divided by its total area, as seen in Equation [2]. We can plot the F90 of a detector run to see the general behavior of each recoil type, since electronic recoils have a longer scintillation process compared to nuclear recoils. Electronic recoils will have a lower F90 value due to this property, allowing graphical separation between classes, as seen in Figure[6] for a full AmBe calibration run.

$$f_{90} = \frac{\int_{t_0}^{90} X_i}{\int_{t_0}^{t_f} X_i} \quad (2)$$

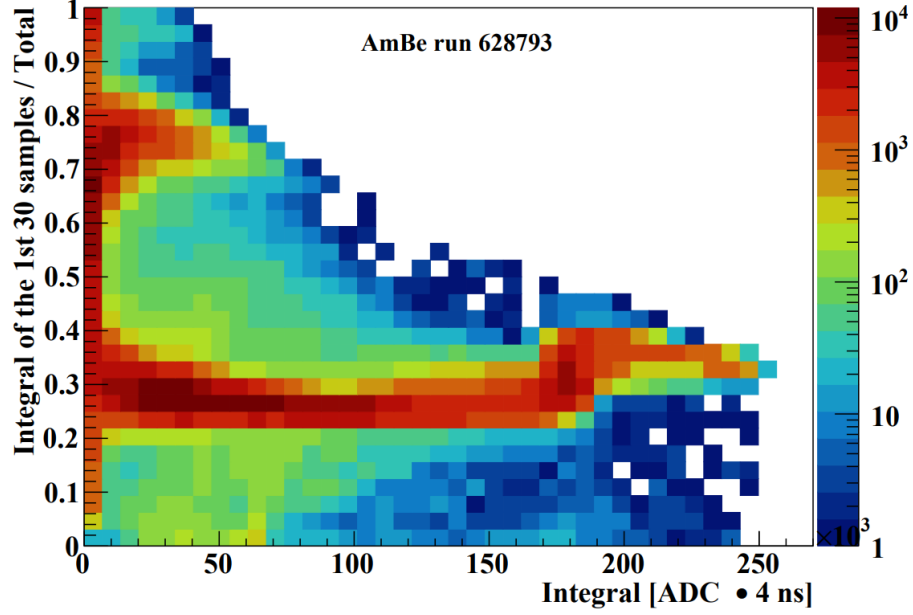


Figure 6: F90 plot of a AmBe calibration run. Distinct groupings highlight the decay behavior for both recoils.

This method is effective at moderate energy ranges (50 keV), but there is some ambiguity as we approach lower energies. If we observe the F90 plot, we can see a region where both classes start to overlap. By sampling events near this region, we see a non-linear relationship that separates the classes, preventing any linear hard cuts in classification, as seen in Figure [7] region IV. With potential issues in our target energy range, we must investigate other possible methods for classifying our recoils.

One proposed method of classification is the use of machine learning, which has shown success across many domains[8]. Although a variety of machine learning models exist, we are interested in the viability of neural networks in classifying our time-series data. The architecture of each neural network is largely dependent on how it functions; convolutional neural networks (CNN) convolve input data to smaller features, while recurrent neural networks (RNN) can send node outputs back to previous layers.

In many ways, neural networks behave similarly to biological neurons. The network receives an input and applies a function, where the output of the function is pushed to another layer in the model. A weight, or importance, is given at each step, which updates



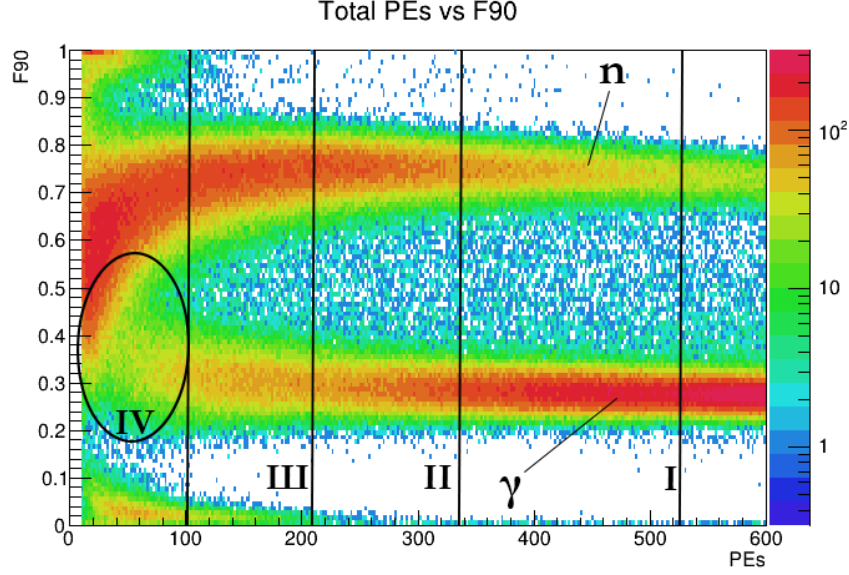


Figure 7: Zoomed in region between 0-60 ADC of selected AmBe run, where the X-axis is converted to total PEs. Sampled regions are labeled by energy for future reference, with gamma and neutron bands highlighted. Region IV is an area of interest where F90 cannot be applied.

during the training phase. This process is repeated in varying complexity based on the network and input data. This structure determines the overall behavior of the model, which influences performance for various datatypes.

One network of interest is the dense neural network (DNN), where each node is connected to every node in the next layer. This feeds forward all information learned in the previous layer to every upcoming input, providing a compact view of all features with variable weight. This behavior reduces gradient disappearance while improving robustness[9], allowing room for generalization if the model performs well on calibration data. The desired model would be able to classify recoils at any amplitude from any detector run, which is key for the detection of a CEvNS event.

In order for the DNN, or any neural network, to be effective, there are a number of conditions that must be met[10]. The primary consideration is whether you have enough data to properly train the model. Although highly dependent on the datatype and model used, it is generally thought that 1,000 samples per class are a minimum for classification problems [11] in the time-series domain. Another necessary condition is the use of high quality data during

### Selected Samples For Each Recoil

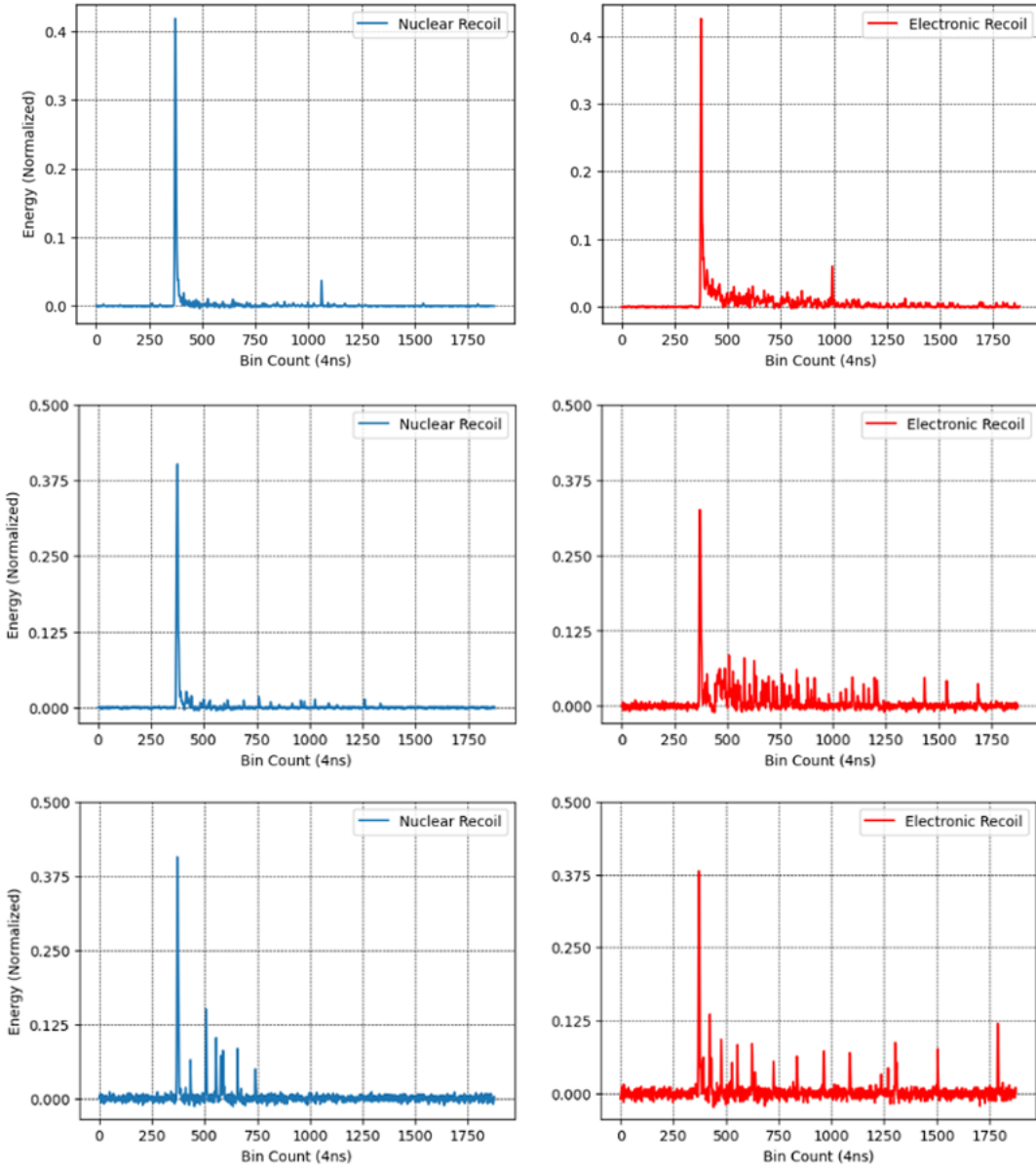


Figure 8: Nuclear and electronic recoil samples. Left samples are nuclear recoils descending in energy. Right samples are electronic recoils descending in energy. Due to normalization methods, the peak height may be larger for less energetic samples.

the training phase. If the input data doesn't reliably capture the signal's true behavior, the model cannot learn what the true behavior is. Samples selected as representations of each class are shown in Figure [8].

These conditions are favorably met with data received from CENNS-10. There is a large quantity of data available for analysis, with an average of 500,000+ events per run. Waveforms are compared across calibration runs to ensure a consistent detector response from each recoil, as seen in Figures [9] and [10]. This was performed by verifying the calibration PE response, and aligning the initial scintillation peak with recoils of the same energy. Since the criteria is satisfied, we can prepare the data for analysis.

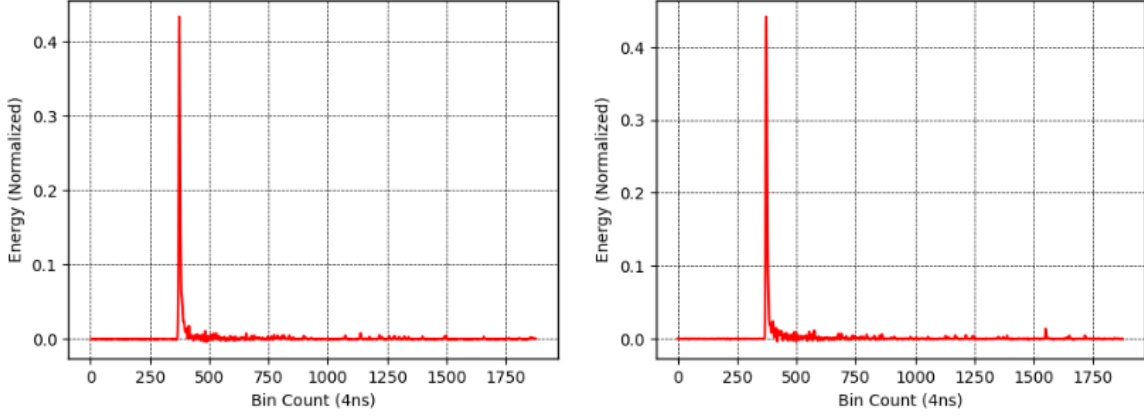


Figure 9: Nuclear recoil samples from separate AmBe calibration runs. Samples shown are taken from the same PE count in region I.

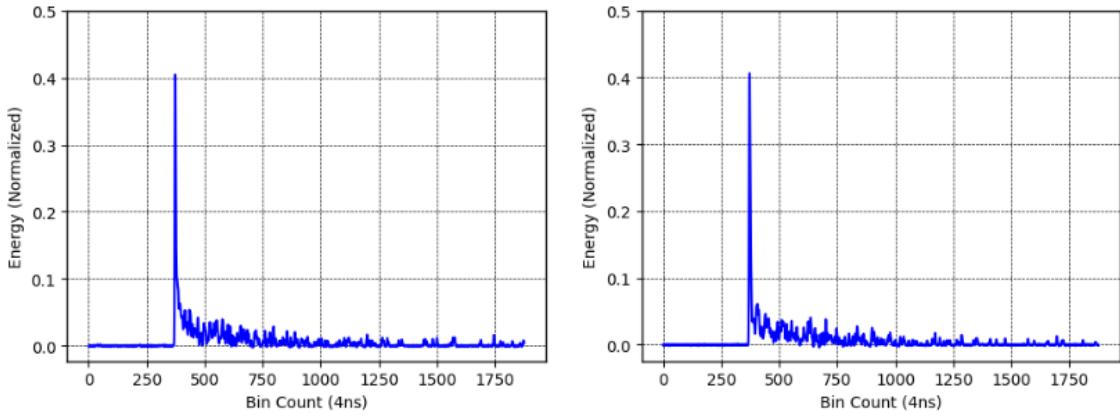


Figure 10: Electronic recoil samples from separate Co-57 calibration runs. Samples shown are taken from the same PE count in region I.

## 2 Literature Review

### 2.1 Observing CEvNS

Since the age of modern physics, researchers have continued to develop technology to observe phenomena on increasingly smaller scales. Many contemporary pursuits aim to detect interactions on a subatomic level, such as neutrino collisions and dark matter. Understanding the physics behind these exotic particles is crucial to the verification of the Standard Model and many important theories. The research behind these projects inspires new theory and technological development, advancing both physics and scientific knowledge in general.

Neutrino interactions imprint only small amounts of observable energy in matter, making them especially difficult to observe. One proposed interaction of neutrinos are coherent elastic neutrino nucleus scattering events, or CEvNS. CEvNS events are the lowest energy interactions between neutrinos and matter in the Standard Model, with a range of 1-25 keV[3]. In order to observe these interactions, new methods of detection and analysis were needed. In this review, we discuss current detector systems and their related analysis frameworks in the search for CEvNS.

There are many potential difficulties in observing CEvNS events due to their low energy. A primary consideration when designing a CEvNS detector is the target material. The coherence implies a neutrino is able to see the nucleus as a whole, instead of individual nucleons at this low energy[12]. Due to the almost point like nature of neutrinos, the probability of a collision is largely related to the target’s atomic cross-section. This introduces another potential issue, as we want to maximize this probability, but heavier atoms react less drastically to the collision. Too light of a nucleus, and the probability to observe a collision rapidly decreases. This implies a “sweet spot” of target nuclei, optimizing the relation between cross-section and collision dynamics.

Another potential difficulty lies in the analysis of detector data searching for CEvNS events. Neutrinos are neutrally charged particles, so they are able to interact directly with

the target nucleus producing a nuclear recoil signature. Other neutrally charged particles may interact with the detector, such as neutron collisions which produce similar nuclear recoil signatures. Near the CEvNS energy threshold, low energy neutrons can effectively mimic a CEvNS event further complicating analysis[12]. Additional recoil signatures can also be detected, such as a gamma ray interaction that produces electronic recoil signatures. Since photons interact with the electrons surrounding the nucleus, the signature has different characteristics compared to a nuclear recoil. Although these recoils are trivial to distinguish at energies above 50 keV, low energy electronic recoils can share some parameters in existing analysis methods, such as pulse shape discrimination.

Since there is some restraint in the selection of target materials, it is usually the primary consideration when designing a CEvNS detector. As previously discussed, the cross-section is a critical parameter, as the Standard Model directly predicts the likelihood of CEvNS events. If a target nucleus' parameters are well known and the rate of CEvNS detection can be found, a direct test of the Standard Model can be performed[13]. By observing CEvNS events with a variety of detector materials, a confident bound can be shown for the dependence on cross-section.

## 2.2 CsI and LAr

First proposed in 2015, Cesium-Iodide crystals are a strong target candidate for the detection of CEvNS events. The combined mass of both atoms is within the desired range for observation, with a 14 kg detector theoretically capable of detecting 550 CEvNS events per year[2]. Since the response of this crystal to nuclear recoils is well studied, a low energy observational threshold around 7 keV can be demonstrated which is well within the CEvNS energy threshold. The COHERENT collaboration's CsI detector made the first observation of a CEvNS interaction in 2017 at Oak Ridge National Laboratory's Spallation Neutron Source (SNS)[13]. An advantage of utilizing a well known spallation source can be seen in the analysis of data from CsI, as researchers were able to distinguish the delayed response

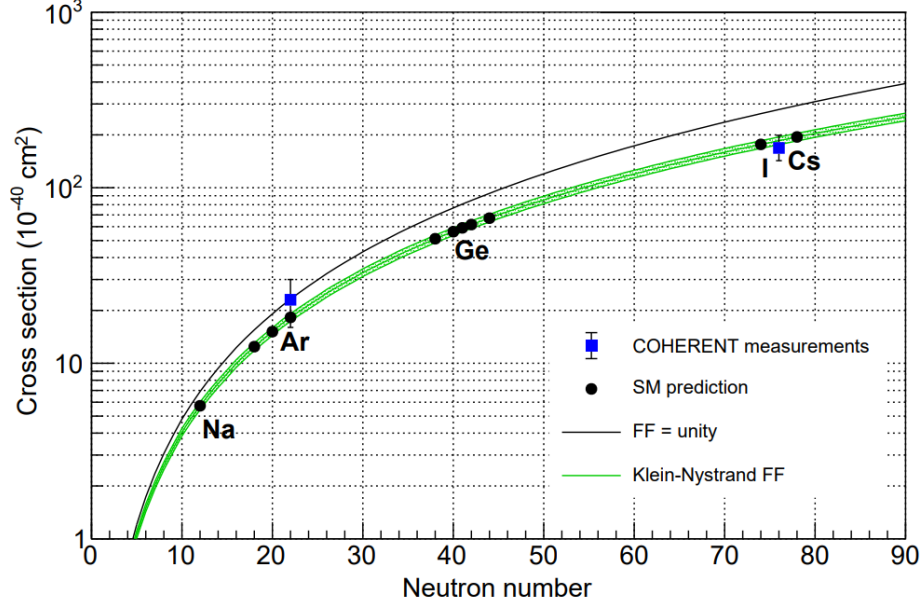


Figure 11: Flux-averaged CEvNS cross section as a function of neutron count[14]. The green curve represents Klein-Nystrand form factor, while the black curve assumes unity.

of incident neutrons to neutrinos produced from decay. Since the recoil signatures of these two particles at low energy are very similar, behaviors such as this must be incorporated into analysis to distinguish the events with high confidence. One potential issue with a similar analysis would be the observation of CEvNS events without a spallation source, as no timing information would be available to separate neutron from neutrino recoil distributions. Additionally, researchers have noted considerations of smearing and bin choice can have a large impact on the constraining of final results of current analysis[15].

Argon has also been discussed as another material with a desirable cross-section and mass for CEvNS event detection[16]. Commissioned in 2016, COHERENT's CENNS-10 detector is a liquid argon based scintillator with a low energy threshold. The detector was upgraded after initial engineering runs to see up to 150 CEvNS events per year, which has led to another subsequent observation of CEvNS[17]. Analysis frameworks for data from CENNS-10 includes pulse shape discrimination, which performs well on typical energies. Since CENNS-10 is operated in Neutrino Alley at the SNS, analysis considers timing information similar to the CsI detector, although CENNS-10 is slightly further from the spallation source. Re-

sults indicate that although this analysis can detect CEvNS events, it cannot be used for the whole energetic range of events within the detector. There is a significant portion of recorded events where test values from pulse shape discrimination of nuclear and electronic recoils overlap. Additionally, CEvNS events that occur within the main arrival window of beam-related neutrons will be lost if the delay time is fully considered.

## 2.3 Reactor CEvNS

Several CEvNS experiments search specifically for reactor produced neutrino events, such as CONNIE[18], CONUS[19] and  $\nu$ GEN[20]. CONNIE is a charge coupled device (CCD) based detection system, housed at the Angra dos Reis nuclear plant outside Rio de Janeiro, Brazil. The CCDs have a pixel dimension of  $15\mu\text{m} \times 15\mu\text{m}$ , with nearly 8 million pixels in their 3cm model. Currently, the collaboration is upgrading their systems to observe below 10 keV events and improve shielding around the reactor. Related analysis efforts are being updated to include the new sensitivity, but similarly to detectors at the SNS, timing information plays a crucial role for a CEvNS observation.

The CONUS experiment is another reactor based CEvNS detector located in Brokdorf, Germany. CONUS deploys a germanium target positioned 17m from the reactor core, which provides an intense profile of lower energy radiation[19]. In an analysis report published in 2021, CONUS presents a new upper bound on CEvNS event rates for their germanium target and the calculated quenching factor. The expected rate of observed events per year is 11.6, which may not be high enough to classify the event within the data collected. Upgrades to the system are currently planned, but current analysis methods may not detect a CEvNS event. Further considerations to mitigate background and improve signal to noise ratios are considered.  $\nu$ GEN is another germanium based detector located in a reactor facility outside of Moscow, Russia. Data is currently being taken, but initial analysis shows an observable range within the CEvNS threshold. Extensive efforts to compare backgrounds for ON/OFF reactor operations has been completed, with additional 1.0 kg and 1.4 kg germanium targets

being prepared for installation[20]. An internal sodium iodide (NaI) veto system is being tested to identify backgrounds and reject collection by the DAQ.

Liquid time projection chambers have shown high sensitivity to observe low energy events and their trajectory through the detector material. Experiments such as XENONnT indirectly search for CEvNS[21], as neutrinos will serve as a primary background for dark matter searches. Since weakly-interacting massive particles (WIMPs) are expected to produce nuclear recoils similar to CEvNS, identifying their recoil signatures is crucial for confident detection at these low energies. The XENONnT detector is a 1 ton dual-phase xenon system located at the Laboratori Nazionali del Gran Sasso in Italy, with an effective submersion under 3600m of water. Scintillation is captured by dual PMTs, with a design similar, but much larger, than COHERENT’s CENNS-10 detector. Work is underway on background calibrations with a variety of sources to observe the effect on sensitivity, with expected results being published in the near future.

## 2.4 Discrimination Methods

Several pulse shape discrimination methods have been used by researchers to distinguish electronic from nuclear recoil responses. COHERENT’s framework focuses on the first 90 nanoseconds, or F90. The F90 value is a scalar that relates the area of the first 90 nanoseconds of a waveform to its total area. This method works well as it relies on the physical difference in light output between the two recoils; electronic recoils have a longer decay profile than nuclear recoils, so more area is distributed across the event sample. This results in a lower F90 value for electronic events, which can be easily distinguished from a nuclear recoil’s higher F90 when plotted. As the observed energy is decreased, the full behavior of each recoil cannot be guaranteed to match its higher energy counterpart, leading to similar F90 values for both events. Additional methods such as figure of merit, or FOM, have been discussed to distinguish recoils[22]. In contrast to F90, FOM primarily considers the full width at half max (FWHM) as its main parameter. This method may work for data from CENNS-10, as



similar energy nuclear and electronic recoils will have a different max amplitude, even after normalization. Several other pulse shape discrimination techniques are available to distinguish nuclear from electronic recoils, such as zero crossing or discrete wavelet transforms. Since F90 was shown to be effective for data from CENNS-10, comparisons to other pulse shape discrimination methods have not been published.

Machine learning has been utilized by many domains for classification problems such as high energy and plasma physics. Although many forms of machine learning exist, several model types are discussed for the classification of time series data. Recurrent neural networks (RNN), convolutional neural networks (CNN), and long short term memory (LSTM) are among the many networks utilized for classification. In [23], RNNs were shown to be effective at problems requiring minimal backpropagation. This effect can be magnified the longer the network is run, leading to a disappearing or exploding gradient. Complex data with many relevant features may require longer training times, which lead to the development of LSTMs to combat the issue. LSTMs can introduce their own training problems and are generally not recommended for time series classification. Multi-Layer-Perceptrons (MLP) have shown success in classification problems and work well handling a variety of data[24], with densely connected MLPs sacrificing performance for improved accuracy. The dense feed-forward behavior allows the whole signal to be considered at the first layer, which could preserve important features at the cost of increased computational demand. This model performs quite well on data where the distribution vs time is consistent, where temporal information predominantly informs the behavior. Shifts in the timing distribution of waveforms are considered later in this paper, with the related effects on network response discussed in future work.

Many analytical tools are available to help researchers distinguish nuclear from electronic recoils, but few publications exist on the comparison of pulse shape discrimination techniques, or to these techniques and machine learning. Evidence shows that methods such as F90 have a high degree of accuracy over a large energy range of detector samples, but may not have

confidence towards the CEvNS energy threshold. Research suggests the machine learning models, such as dense MLPs, may be able to classify events at a lower energy than F90.

With few publications, additional research is required on the comparison of these two methods. To establish any relevant metric, several aspects must be fully investigated. The lowest energy where each method is confident in classification, the generality of both techniques, and their relation to the larger body of physics are all questions that must be explored. The detection of CEvNS events have implications beyond neutrino characterization, pushing researchers closer to the observational limit of dark matter interactions. The improvement and comparison of analytical methods at higher energies could improve or inspire detection methods for lower energy physics, which is a primary goal of the community at large.

### 3 Method

In the following section, a complete data pipeline from detector DAQ to analysis is presented. Several programming languages are discussed such as ROOT and C++, but most of this work relies on Python and several key packages. These languages are referenced as they appear in the pipeline, with all supplementary code provided in Appendix A. A brief description of non-trivial packages is given in Appendix B. Parameters for our machine learning model are briefly discussed, with more detail given in the Results section of this paper.

To begin our analysis, we start at the detector. When an event is recorded by CENNS-10, the data is output in the form of a ROOT file and saved to ORNL managed servers. ROOT is a proprietary language developed by researchers at CERN to better handle the large and complex nature of data in particle physics. For our purposes, a preliminary cut is made to reduce the size of the file to several gigabytes before transferring from the high-performance cluster which retains all of COHERENT's data. This is accomplished by a C++ script which collects events based on our parameter selection, such as total area of each waveform. The preliminary cut is made to remove events above our target maximum energy, and with another to isolate recoils within a set F90 bound. This allows us to obtain pure recoil samples from, for example, a calibration run which uses Americium-Beryllium (AmBe) as a neutron source. We can similarly obtain electronic samples by selecting a run that is subject to a direct gamma source, such as cobalt-57. By building datasets that contain only pure electronic or nuclear responses, we can begin to process the data and prepare it for analysis.

For the rest of this work, we utilize Python as our main language which is deployed as a modified container in Docker. This container provides a reproducible virtual environment with all the necessary packages and custom mounting, allowing consistent interpretation over all devices. In our deployment, the standard TensorFlow image is modified with additional packages such as Seaborn and UpRoot, and mounted to a directory we can access with Git commands. This step provides code consistency across personal hardware, Docker, and

GitHub repositories, and ensures the correct package versions are maintained. We can now initialize a Jupyter server with our modifications and pull our data into Python.

There are several packages available that can process ROOT files, such as UpRoot and PyRoot, which are used to import the data and convert them into arrays. When the data has been loaded, we can use Pandas dataframes to make various cuts and bin our target energy ranges. Another benefit to processing in Python is the ability to quickly output smaller csv files, so that we can build a collection of individual datasets that contain only electronic or nuclear recoils at each specific energy range. This helps to simplify data management and greatly helps with the reproduction of analysis results. After the initial Python processing, we will have 6 separate files: 3 files for electronic recoils at a low, medium, and high energy, and 3 files for nuclear recoils at low, medium, and high energy. Our low, medium and high energy ranges contain waveforms with 100-210 PEs, 211-340 PEs and 341-520 PEs respectively. From the initial 500,000+ events in both runs, each csv file now contains only 2,000 samples, resulting in a large reduction of handled data. 2,000 events were chosen as an arbitrary minimum for training/testing, but research suggests downsampling to 1,000 events may still provide decent training.

Now that each class and energy range of our recoils has been isolated, we can process one energy range for both classes to create our training and testing datasets. The csv files are concatenated and stored in a dataframe, where we now have 4,000 rows of events. It is important to normalize the data before assigning class labels, which would otherwise be included in the calculation. In this case, we compute the integral of each event and set our normalization to 1. Each of the events will now have unit area, at which point we add our class labels. To remove any bias from event selection, we shuffle the dataset multiple times and separate our labels into a new dataframe. It is critical to not reshuffle after this stage, as the label dataframe is ordered with each event. Additional shuffling after class separation would mismatch the event label during training, invalidating any results.

The data is now ready to be split into training and testing datasets. Following standard

convention, a train to test ratio of 3:1 was chosen for our model[25]. There are now 3,000 samples to train and 1,000 to test, which have been stored in separate dataframes. We can now initialize our model and prepare for the training process. To start, we generate a dense neural network (DNN) with 4 layers using the Python module TensorFlow. The first layer is simply the input and is the length of our signal. The second and third layers are densely connected, with a node size of 24 and 128 respectively. The final layer is a binary classifier with a sigmoid activation since we only have two types of recoil events. The general structure of the network is shown in Figure [12].

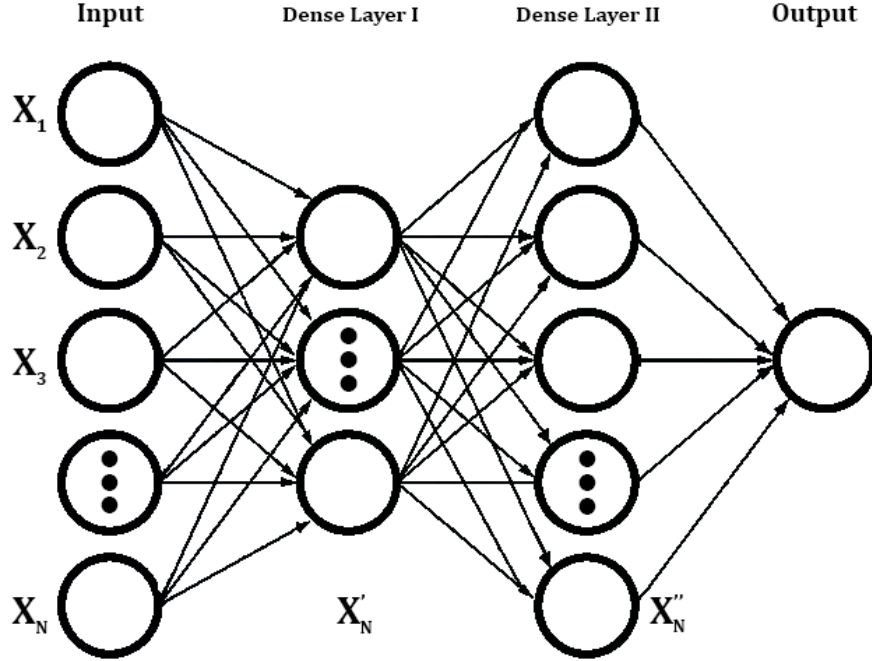


Figure 12: Generalized structure of our dense network.  $X'$  and  $X''$  are the number of nodes in Dense Layers I and II, respectively.

It is time to set the optimizer, learning rate, and epochs for training. The Adam optimizer was chosen with a .001 learning rate, over 20 epochs. The learning rate determines how much the model can adjust its weights each update. Epochs reflect the total number of training cycles, which restart after every samples has been fed through the model. The amount of training time is arbitrary at this stage since it will be adjusted, depending on model performance, but 20 is a reasonable start. Generally, if increasing the total epochs doesn't

provide increased accuracy, the model itself needs adjustment. All the parameters and inputs for the model are now ready, so training can begin. To evaluate the model's performance, we can review the accuracy and cross-entropy loss of the model over time. These values are plotted to gain a sense of how well the model learns, and should follow the general trend of increasing accuracy and decreasing loss. Another method of determining your model's performance is to generate additional datasets independent from data used during training and testing. A preferred method is to process new calibration runs to obtain similar, but unique events, so we can test using data that the model has never seen. This new dataset can be manually fed through the testing function of the model, which will predict each event's class based on prior training. This new dataset's accuracy is another indicator of the model's performance, and is a standard method for validating overall model accuracy. From here, any of the parameters in the model can be adjusted, such as node size, learning rate, or training time. Modifications to the network and parameter selection are discussed later in this paper.

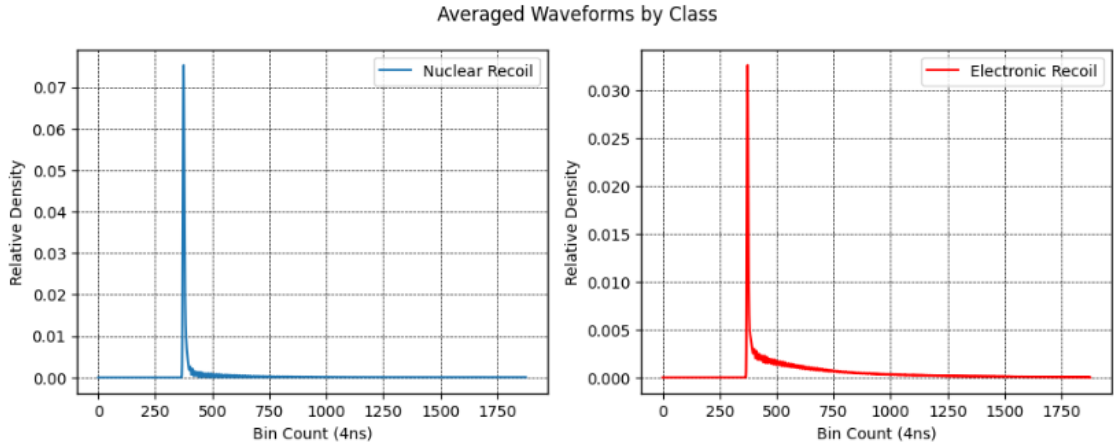


Figure 13: Averaged waveform of each class from our highest energy samples.

One further consideration is to generate simulated events from real data, so that other energy ranges can be investigated. In the case of data from CENNS-10, signals producing less than 40 photoelectrons can complicate traditional analysis methods such as F90, since testing parameters begin to overlap. To investigate this region, we sum all the events from

one class and normalize, to produce an average density distribution for that class, as seen in Figure [13]. This can be used to simulate signals based on the probability of individual photoelectrons hitting the detector at each time interval, which is obtained from our average waveform. By applying a random sampling algorithm based on this distribution, we can generate locations of random photoelectrons. By saving those locations in a list, we can iterate through and place Gaussian peaks at each photoelectron location to build up our signal. The peak shape was determined using a chi-squared test of best fit, and normalized to the calibration response of 1 photoelectron. When the peaks have been placed, we can add in the electronic noise from instruments in CENNS-10. These values were calculated from several runs, and used to create a Gaussian noise filter to add on top the simulated waveform. The construction of nuclear recoil signals can be seen in Figures [14]-[16] in several steps, with electronic recoils receiving the same treatment. Results on testing with simulated data are presented later in this paper.

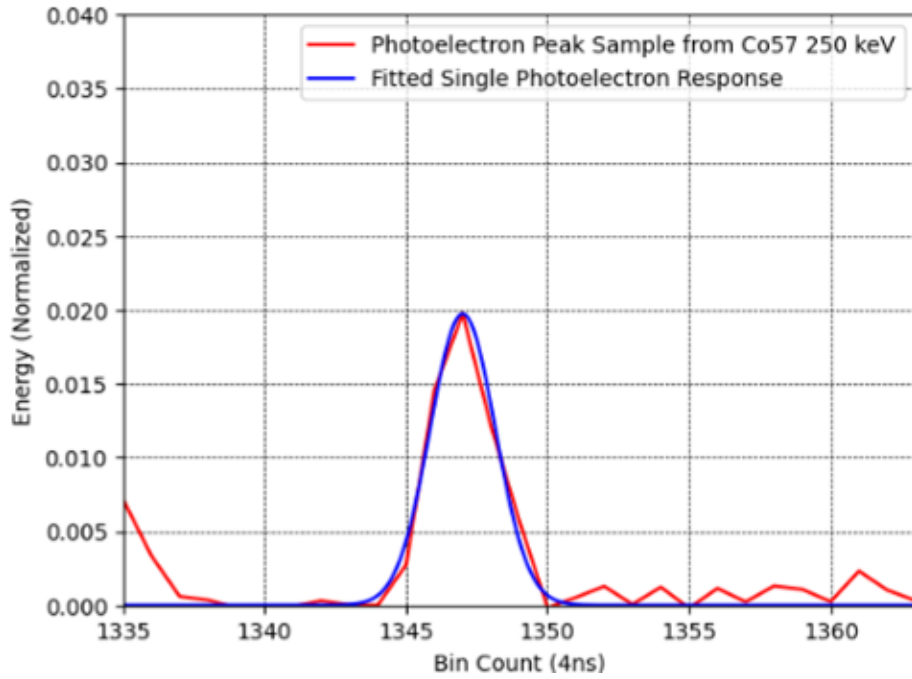


Figure 14: Fitted Gaussian peak to single PE.

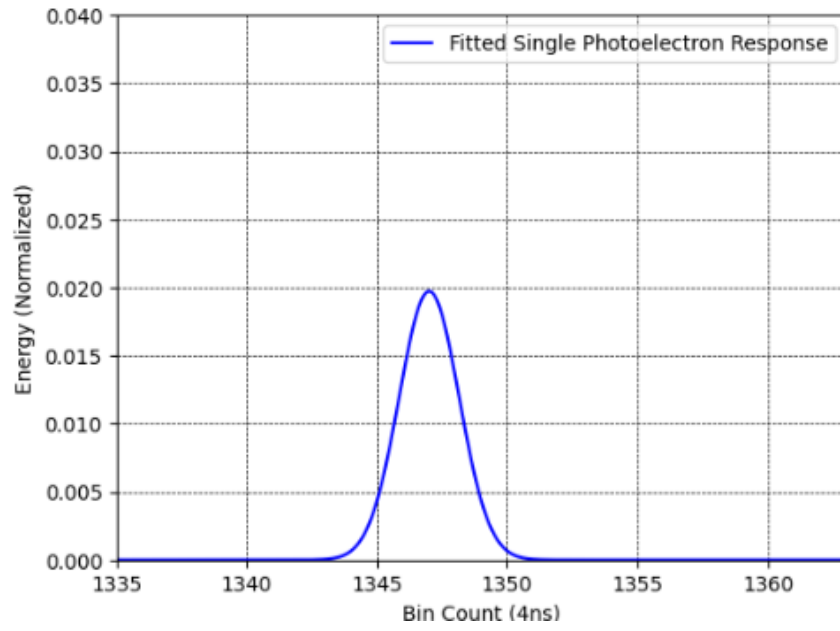


Figure 15: Fitted Gaussian profile.

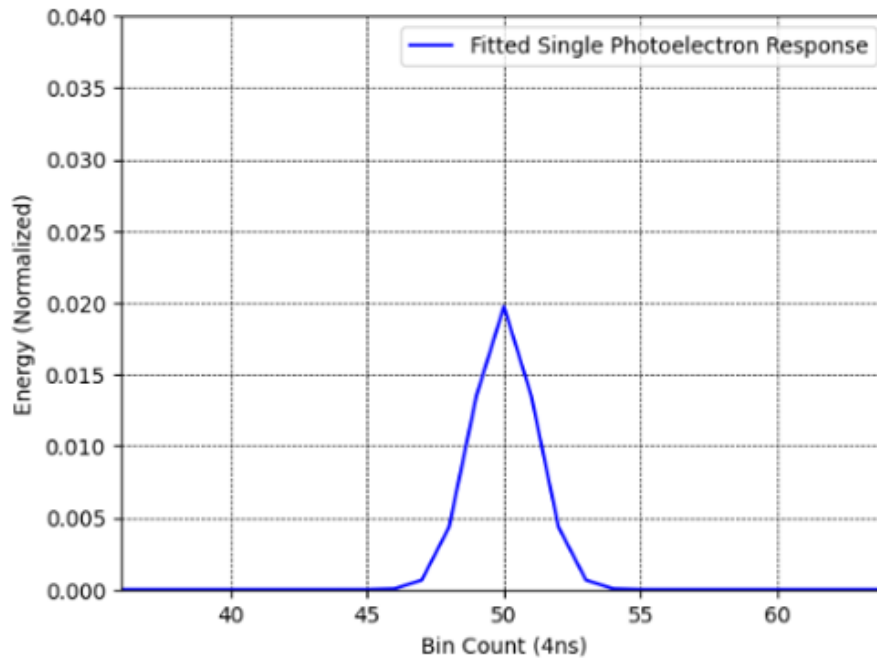


Figure 16: Fitted Gaussian profile, downsampled to real data sample rate. Axis change from downsampling and lead time. Final peak used is cut from bins 46-54.



## 4 Results

To determine the relative performance of our model, we must compare the results to existing methods of background discrimination. A common method used to identify recoils is the previously discussed F90, or first 90 nanoseconds. This method computes the area of the first 90 nanoseconds of each signal, divided by the total area. Each event's F90 value can be plotted to generate a heatmap, showing the general behavior separating each class. Since the scintillation decay is much faster for nuclear recoils, we expect those events to have a higher F90 value. We can verify this grouping method by randomly sampling events from each cluster, and checking whether it is the expected recoil type. This method is robust at typical energies, but becomes less confident as total photoelectrons decrease. In our sampled energy regions, F90 is approximately 99.99% accurate at typical constraints (nuclear recoils  $F90 \approx .7$ , electronic recoils  $F90 \approx .3$ ). Misclassifications can occur with anomalous signals, or as a result of photon statistics[26]. At very low energies, both classes' F90 scores can overlap, as we can see in low PE simulated events.

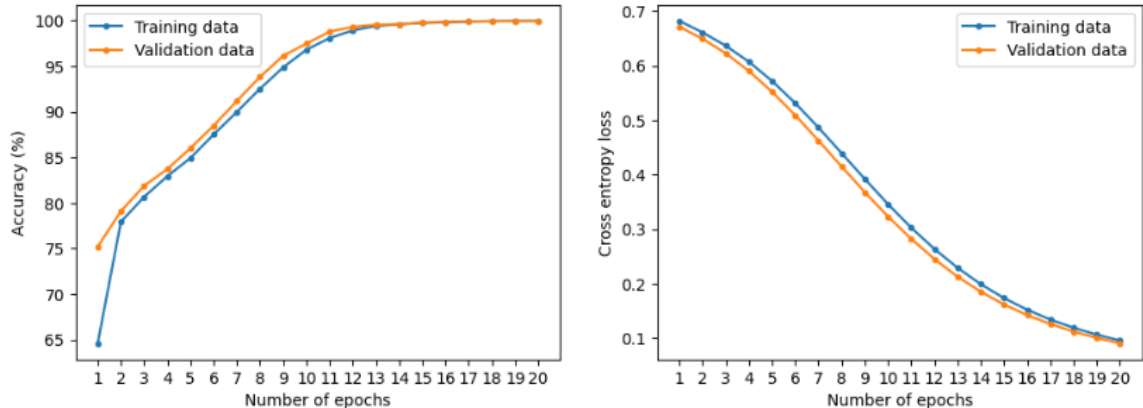


Figure 17: Performance on training data from region I. Accuracy reaches 99% at epoch 12.

At our three energy ranges with real data, the neural network classifies recoils with similar accuracy to the F90 method, approximately 99%. When observing the training of each energy range, we can note the epochs required to reach maximum accuracy increases

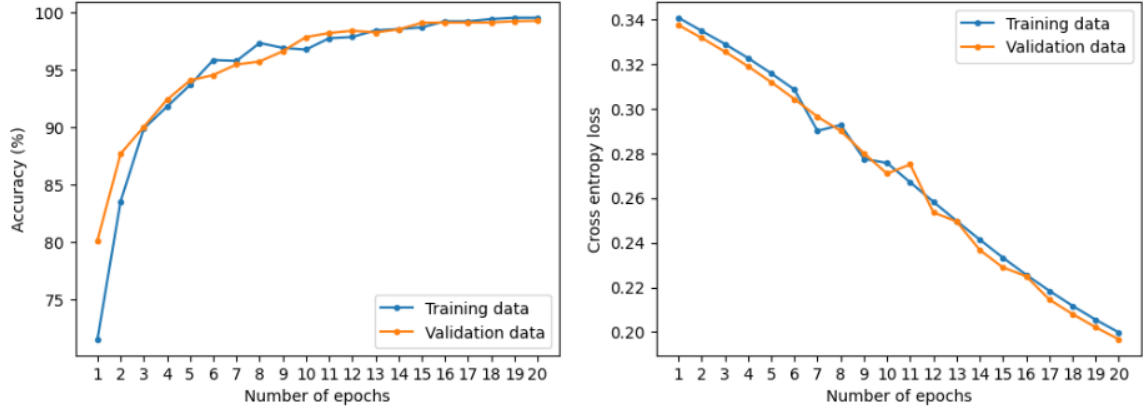


Figure 18: Performance on training data from region II. Accuracy reaches 99% at epoch 17.

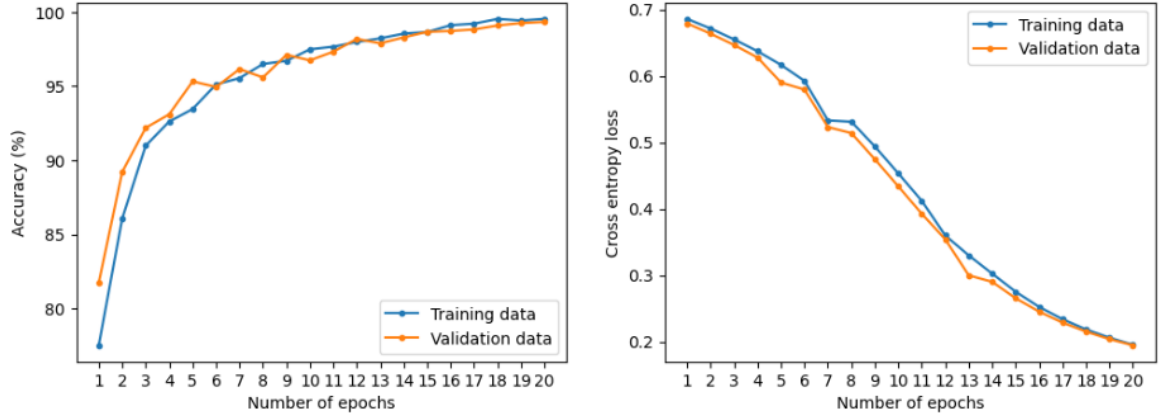


Figure 19: Performance on training data from region III. Accuracy reaches 98% at epoch 20. Retraining with fresh weights finds 99% at epoch 25.

as energy decreases, as shown in Figures [17]-[19]. Training time was increased for networks until asymptotic accuracy was observed, and only the first 20 epochs are shown for relative comparison. This behavior is expected, but suggests there is still room to investigate lower energy regions.

Since the model was shown to accurately predict classes from calibration runs, we can apply the model to our simulated data. We generate various energy ranges by controlling the number of photoelectrons the algorithm will produce, and bin them accordingly. The first simulation created samples from the region around where F90 can no longer be applied, between 24-40 PEs[26]. The second and third simulations contain events with 12-24 PEs, and 7-12 PEs, respectively, with selected samples of 20 PEs shown in Figure [20]. The final

generated dataset contains events with 4-7 PEs ( $\approx 15$  keV) and is the lowest threshold we test against our model,

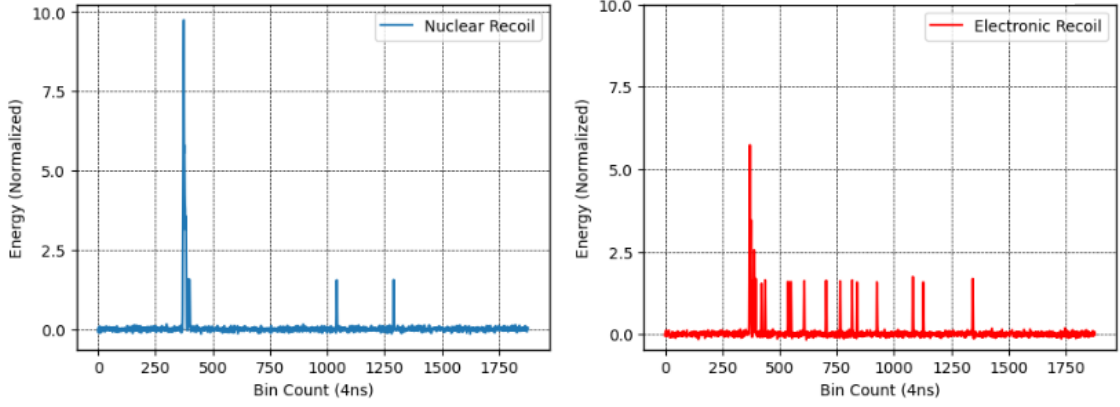


Figure 20: Simulated samples from energy region IV in Figure 7. Samples contain 20 PEs.

Moderate performance was observed in the first two energy ranges, with a testing performance around 90%. It is interesting to note at these energies, the simulated and real events after normalization appear visually similar, but their appearance becomes less pronounced in the 7-12 photoelectron range, along with a decrease in model performance. The lowest energy events had the least accuracy of all tested data, achieving a maximum of 80%.

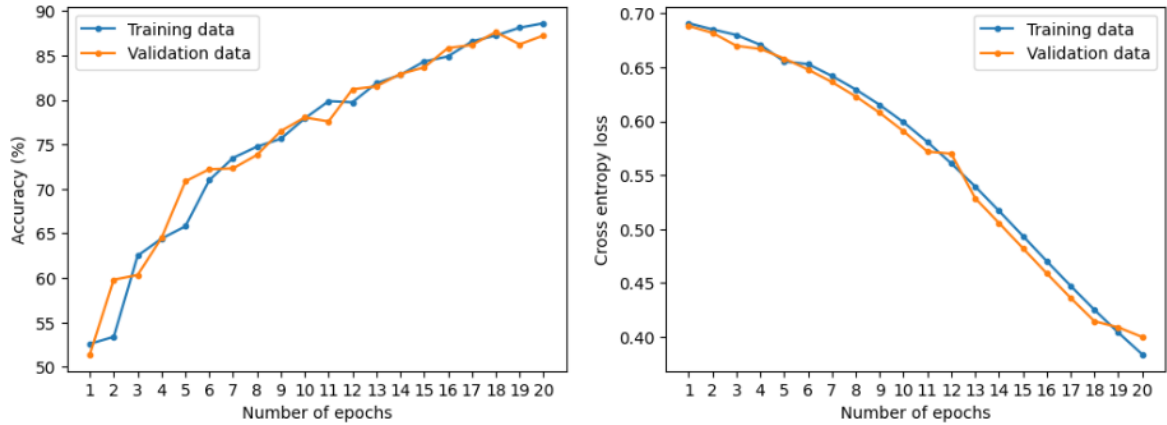


Figure 21: Performance on 24-40 PE simulated data, our highest energy range correlating to region IV. Asymptotic accuracy of 89.5% observed in epoch 20. Increase training time observed negligible accuracy gain, while loss plateaus around .35.

We are encouraged by the initial results of our neural network on classifying recoils. Fur-

ther development of the model could lead to lower classification energies and total generality. Since experimental data may have a small time shift from the DAQ trigger, we sought to establish a baseline of performance on calibration data. Modifications to the model structure, parameter selection, and other techniques to introduce generalization are discussed later in this paper.

## 5 Conclusion

The application of dense neural networks has been shown to be effective for classifying calibration recoils from the CENNS-10 detector. A major motivation of this research was to compare the effectiveness of existing analysis frameworks with our machine learning method with events below 100 keV . Results indicate that our dense network may classify recoils at energies one order of magnitude lower than F90, which approaches the CEvNS energy domain around 10 keV. Although the performance exceeded our initial expectations, we are eager to further generalize our model for experimental runs and simulated data. A network able to classify recoils at the CEvNS energy threshold, on any run, would be a huge success for this research, and is our primary motivation moving forward. There are several proposals to address this topic, which are discussed in the following section. We are excited to continue this research and look forward to future results.

## 6 Current Efforts

The model’s performance on calibration and simulated data was adequate for our initial research, but we seek to further generalize the model to classify experimental runs. As previously discussed, calibration data acquisition is triggered by the arrival of 2 photoelectrons at each PMT, so that each sample’s main scintillation peak occurs approximately in the same bin. Since the accuracy of our model is likely tied to the consistent temporal distribution of calibration data, we must consider data when the trigger mechanism is not identical across samples.

A simple case to highlight this potential issue would be a waveform that has a shorter lead time, so the main peak is slightly to the left as seen in Figure[22]. Since our trained model has given null weights to data before bin 372, a peak falling in this region would not be considered an important feature, likely leading to misclassification. Since experimental data can have a range of possible lead times, we must adjust the model or augment the data to compensate for this shift. We propose several modifications to the network’s topology, along with possible augmentation methods for the experimental data.

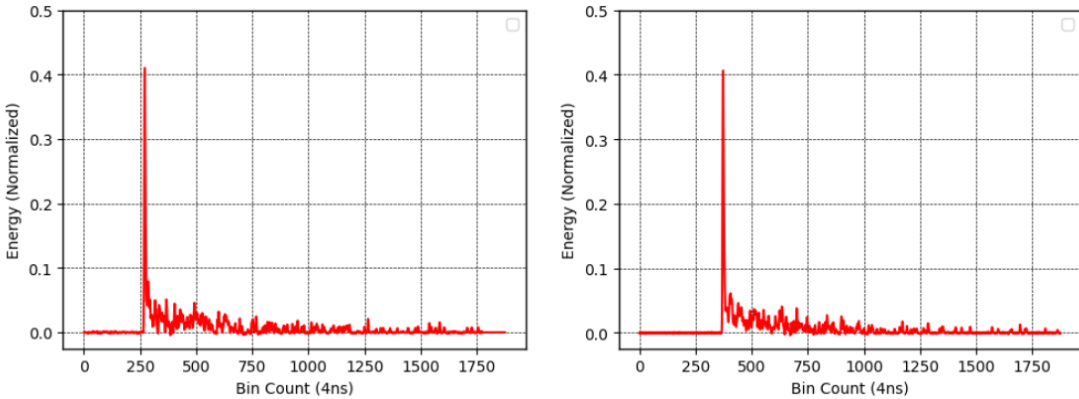


Figure 22: Time-shifted recoil vs calibration recoil of similar energy.

Before discussing our proposals, it is important to note the simple issue presented highlights the robustness of the F90 framework. Since the F90 value is calculated from the trigger point, it can shift with the data freely without any parameter adjustment. This means the

F90 value for nuclear recoils from calibration and experimental data should be identical, resulting in no significant change in how the analysis is performed. This robustness does not extend to the target region below 25 keV, so there is still motivation for improving our network.

Since the main motivation for generality comes from a potential temporal shift in the waveform, we must consider the addition of new layer architectures beyond our dense connections. A popular choice in time series data with temporal feature variance is the use of single dimension convolutional layers. Convolutions apply a vector filter, typically much smaller than your input length. This filter is applied sequentially through your data, with filter size and stride length being the main parameters. This technique aims to reduce the input data to its important features, mitigating reliance on exact timing information. There would be some subtlety in the implementation of a convolutional layer for our data, and could be an interesting optimization problem in itself. In order to compensate for the lead time variance, the convolutional filter would need to be able to handle the entire range of temporal shifts, and collapse the main peak information into the same bin, regardless the shift. It is possible a proper convolution cannot be performed to result in the necessary criteria, so additional layer architectures are also considered.

Similar to convolutional layers, pooling layers seek to extract the most relevant features from a subset of data. In contrast to convolutions, pooling does not overlap when striding nor does it apply a mathematical function, it only takes the maximum value within a set bin range. These maximum values are collected into a new vector, with a reduced length proportional to the pool size. One proposed use of pooling in our model would be to apply pooling to the first 400 bins, and passing this parameter forward independent from the input. Further questions must be addressed if a pooling layer is added to the network; do we keep the original input or remove the 400 bins when feeding forward? It is believed the parameters would compete if the full waveform was included, but that has yet to be fully determined.

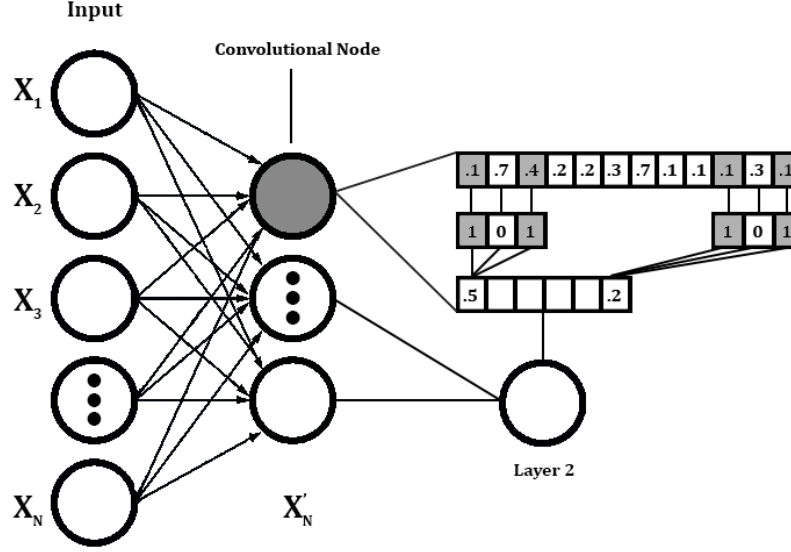


Figure 23: 1-Dimensional Convolution. Filter performs elementwise multiplication then sums the shaded cells. The filter steps forward one input and repeats the operation.

Before making any modifications to the network, we seek to evaluate the current model on augmented data. The proper augmentation of data may preserve important features while minimizing the effects of any temporal shift in our sample. We consider two primary changes to our data; the shifting of the initial scintillation peak from the calibration location around bin 370 and dimensionality reduction. The first method can be accomplished with our simulated dataset, as we control all of the input parameters. When simulating a sample, we can randomly generate a bin shift value to pad or delete our leading entries, then remove or add, respectively, trailing zeroes to maintain the proper sample length. The process is more complicated if we wish to augment calibration data, as removing entries or adding padding will introduce zeroes to a non-zero vector, leaving a flatline that abruptly changes to noise. We can add noise to the zero components, but even a slightly different noise distribution could be detected as a feature by the model, adding significant bias. Current efforts are primarily shifting the simulated data, but we plan to revisit modifying calibration data at a later date.

A final consideration before simulating time-shifted samples is performing F90 analysis on



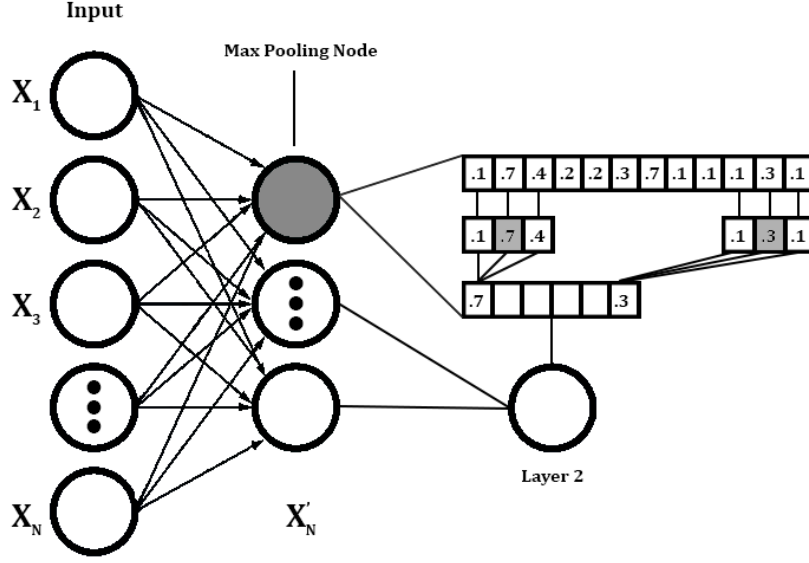


Figure 24: Max Pooling Node. Only the largest value (shaded) from a given subset is kept for the next layer. Subsets may or may not overlap, depending on your parameter selection.

current simulation waveforms. Although the samples appear visually similar to the expected class, it is important they capture the underlying behavior. Even though the simulation may generate distinct recoils, for the region of interest, it may be too distinct. The region of interest is where the F90 distributions overlap, not near the distinct bands of both recoils. Initial findings show that selected nuclear samples are close to the expected F90 value of .3-.5, but a full simulation of 2-150 PE waveforms is required. It is believed that a sample of 1000 waveforms for each class, at each PE, may be sufficient.

Dimensionality reduction methods are also discussed as a way to mitigate both the temporal shift and sparse entries within the data. Sparsity refers to the amount of featureless data within the sample, such as noise, that is included in the training process. We have neglected to consider sparsity up to this point as training times were somewhat reasonable, but if we continue to expand the topology of our network, computational costs should be considered. Although there are many reduction methods, we are primarily interested in principal component analysis, or PCA. This technique aims to transform the original input to a reduced dimension, with the most important features, or principal components, as leading

entries. If component analysis finds the initial peak to be the most important feature, it is likely to reduce this information to the same index for all transformed samples, reducing or removing any effects from temporal shifting. It is expected that noise, or sparsity, will be an unimportant feature and either removed, or placed at the trailing end of the transformation. The main parameter we can adjust is the amount of principal components kept in the final transformation, which introduces some complexity across our target energy ranges. Research on determining how many components are optimal for our classification is ongoing, with initial results suggesting between 100-200 to be sufficient.

Initial results from testing are promising, but research into these efforts is ongoing. We expect conclusive results in the near future as the proposed changes do not require a total reconfiguration of the network, and the changes to our simulation algorithm have been implemented. The comparison of experimental data to the new simulation is underway.

## References

- [1] COHERENT Collaboration. *What is CEvNS?* 2019. URL: <https://coherent.ornl.gov/what-is-cevns> (visited on 03/28/2023).
- [2] COHERENT Collaboration. “Measurement of the Coherent Elastic Neutrino-Nucleus Scattering Cross Section on CsI by COHERENT”. In: *Phys. Rev. Lett.* 129 (8 Aug. 2022), p. 081801. DOI: 10.1103/PhysRevLett.129.081801. URL: <https://link.aps.org/doi/10.1103/PhysRevLett.129.081801>.
- [3] R. Tayloe. “The CENNS-10 liquid argon detector to measure CEvNS at the Spallation Neutron Source”. In: *Journal of Instrumentation* 13.04 (Apr. 2018), pp. C04005–C04005. DOI: 10.1088/1748-0221/13/04/c04005. URL: <https://doi.org/10.1088>.
- [4] B. Becker V. Belov I. Bernardi M.A. Blackston D. Akimov P. An C. Awe P.S. Barbea and L. Blokland. “Development of a 83m-Kr source for the calibration of the CENNS-10 liquid argon detector”. In: *Journal of Instrumentation* 16 (Apr. 2021). DOI: 10.1088/1748-0221/16/04/P04002. URL: <https://iopscience.iop.org/article/10.1088/1748-0221/16/04/P04002/meta>.
- [5] Wanderson P Dionísio C Braga M Galbiatti H. “Evaluation of gamma-gamma well logging data applied to iron ore exploration – vale geophysical well logging test facility”. In: *Revista Brasileira de Geofísica* 34 (June 2016). DOI: 10.22564/rbgf.v34i2.796.
- [6] P Christillin. “Nuclear Compton scattering”. In: *Journal of Physics G: Nuclear Physics* 12.9 (Sept. 1986), p. 837. DOI: 10.1088/0305-4616/12/9/008. URL: <https://dx.doi.org/10.1088/0305-4616/12/9/008>.
- [7] S.B. Shea. *Hunting for Neutrinos: When the Unexpected Happens*. 2018. URL: <https://www.energy.gov/science/articles/hunting-neutrinos-when-ordinary-unexpected> (visited on 04/04/2023).
- [8] Ravita Chahar and Deepinder Kaur. “A systematic review of the machine learning algorithms for the computational analysis in different domains”. In: *International Journal of Advanced Technology and Engineering Exploration* 7 (Oct. 2020), pp. 147–164. DOI: 10.19101/IJATEE.2020.762057.
- [9] Zhou T Ye X Lu H Zheng X Qiu S Liu Y. “Dense Convolutional Network and Its Application in Medical Image Analysis”. In: *Biomed Res Int* (Apr. 2022), p. 3. DOI: 10.1155/2022/2384830. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9060995/>.
- [10] Jain A Patel H Nagalapatti L Gupta N Mehta S Guttula S Mujumdar S Afzal S Sharma M Ruhi M Vitobha M. “Overview and Importance of Data Quality for Machine Learning Tasks”. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Association for Computing Machinery, 2020, pp. 3561–3562. ISBN: 9781450379984. DOI: 10.1145/3394486.3406477. URL: <https://doi.org/10.1145/3394486.3406477>.

- [11] S. van Cranenburgh A. Alwosheel and C. Chorus. “Is your dataset big enough? Sample size requirements when using artificial neural networks for discrete choice analysis”. In: *Journal of Choice Modelling* 28 (2018), pp. 167–182. ISSN: 1755-5345. DOI: <https://doi.org/10.1016/j.jocm.2018.07.002>. URL: <https://www.sciencedirect.com/science/article/pii/S1755534518300058>.
- [12] Jack D. Shergold. “Updated detection prospects for relic neutrinos using coherent scattering”. In: *Journal of Cosmology and Astroparticle Physics* 2021.11 (Nov. 2021), p. 052. URL: <http://dro.dur.ac.uk/35078/>.
- [13] J.I. Collar et al. “Coherent neutrino-nucleus scattering detection with a CsI[Na] scintillator at the SNS spallation source”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 773 (2015), pp. 56–65. ISSN: 0168-9002. DOI: <https://doi.org/10.1016/j.nima.2014.11.037>. URL: <https://www.sciencedirect.com/science/article/pii/S0168900214013254>.
- [14] D. Akimov et al. *The COHERENT Experimental Program*. 2022. arXiv: 2204.04575 [hep-ex].
- [15] P. Denton. “A Statistical Analysis of the COHERENT Data and Applications to New Physics”. In: *Journal of High Energy Physics* 166 (Apr. 2021), pp. 17–20. DOI: 10.1007/JHEP04(2021)266. URL: <https://www.osti.gov/pages/servlets/purl/1782539>.
- [16] COHERENT Collaboration. *Coherent elastic neutrino-nucleus scattering: Terrestrial and astrophysical applications*. 2022. arXiv: 2203.07361 [hep-ph].
- [17] J Zettlemoyer. “First Detection Of Coherent Elastic Neutrino Nucleus Scattering On An Argon Target”. PhD thesis. 2020.
- [18] CONNIE Collaboration. *The CONNIE experiment*. 2016. arXiv: 1608.01565.
- [19] H. Bonet et al. “Constraints on Elastic Neutrino Nucleus Scattering in the Fully Coherent Regime from the CONUS Experiment”. In: *Physical Review Letters* 126.4 (Jan. 2021). DOI: 10.1103/physrevlett.126.041804. URL: <https://doi.org/10.1103/PhysRevLett.126.041804>.
- [20] nuGEN Collaboration. “First results of the  $\nu$ GeN experiment on coherent elastic neutrino-nucleus scattering”. In: *Phys. Rev. D* 106 (5 Sept. 2022), p. L051101. DOI: 10.1103/PhysRevD.106.L051101. URL: <https://link.aps.org/doi/10.1103/PhysRevD.106.L051101>.
- [21] XENON Collaboration. “Projected WIMP sensitivity of the XENONnT dark matter experiment”. In: *Journal of Cosmology and Astroparticle Physics* 2020.11 (Nov. 2020), pp. 031–031. DOI: 10.1088/1475-7516/2020/11/031. URL: <https://doi.org/10.1088/1475-7516/2020/11/031>.
- [22] A Linteruer J Ely J Stave B McDonald. “Neutron and Gamma Ray Pulse Shape Discrimination with Polyvinyltoluene”. In: *Pacific Northwest Laboratory Technical Reports* (Mar. 2012), pp. 2–21. DOI: DE-AC05-76RL01830.

- [23] Alex Sherstinsky. “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network”. In: *Physica D: Nonlinear Phenomena* 404 (Mar. 2020). DOI: 10.1016/j.physd.2019.132306. URL: <https://doi.org/10.1016%5C%2Fj.physd.2019.132306>.
- [24] Y Chen Y Zhu Z Yan Y Huang Z Ren J Shen L Chen. *Effective Audio Classification Network Based on Paired Inverse Pyramid Structure and Dense MLP Block*. 2023. arXiv: 2211.02940 [cs.SD].
- [25] V. Roshan Joseph. “Optimal ratio for data splitting”. In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 15.4 (Apr. 2022), pp. 531–538. DOI: 10.1002/sam.11583. URL: <https://doi.org/10.1002%5C%2Fsam.11583>.
- [26] L. Hime A. Brice S. Cooper R. DeJongh F. Garrison et al. “A New Method for Measuring Coherent Elastic Neutrino Nucleus Scattering at an Off-Axis High-Energy Neutrino Beam Target”. In: (Nov. 2013), pp. 7–9.

## 7 Appendix

### A Codes

The following network was used to generate all performance metrics. Arrows indicate margin overflow, not indentations. This was deployed on a modified Tensorflow container with Docker, as a Jupyter Notebook. Line breaks are delimiters between cells. The Docker image and compose file, along with various labelled datasets, are available at [https://github.com/siehienp20/USD\\_Ms\\_Thesis](https://github.com/siehienp20/USD_Ms_Thesis).

This code was designed to be interpreted in a Jupyter Notebook, but can be ran as a stand alone script. In order to run the .py, you will need the imported samples in the same directory. Imported samples need to be in dimensions of (X,1876).

The second code is a preprocessing script to bin up our energy ranges and export to csv. The context of imported data is important, as ROOT files have distinct data structures depending on DAQ. If processing other ROOT files, you will need to replace the appropriate keys. For example, 's' is the raw waveform for our data. The processing only needs the raw waveform and total area, 'a', in this case.

Lastly, we present the code used to simulate region IV samples. The amount of PEs generated can be adjusted by the random variable function inside `peak_gen()`, here 20 is used as an example. The final loop can be adjusted for desired sample size, but 20,000 is recommended as an upper bound if working in Jupyter. These parameters generated the waveforms seen in Figure [20].

## Dense Neural Network in Python

```
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
import matplotlib as mpl
from sklearn.utils import shuffle
from sklearn.preprocessing import normalize
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sys import getsizeof
from sklearn.metrics import accuracy_score

ER_data_pack = pd.read_csv('electronic_recoil_samples.csv', header=None)
ER_data_pack = normalize(ER_data_pack)
NR_data_pack = pd.read_csv('nuclear_recoil_samples.csv', header=None)
NR_data_pack = normalize(NR_data_pack)

plt.plot(NR_data_pack[360], 'g-')
plt.plot(ER_data_pack[340], 'r:')
plt.show()

NR_data_pack[1876]=1
ER_data_pack[1876]=0
NR_data_pack = pd.DataFrame(NR_data_pack)
ER_data_pack = pd.DataFrame(ER_data_pack)

combo_data_pack=pd.concat([NR_data_pack, ER_data_pack.reindex(columns =
    -> NR_data_pack.columns)],ignore_index =True)

combo_shuffle = shuffle(combo_data_pack)
combo_shuffle = shuffle(combo_shuffle)
combo_shuffle = shuffle(combo_shuffle)
combo_shuffle.reset_index(inplace=True,drop=True)

train_data = combo_shuffle.drop([1876], axis=1)
train_data = np.square(train_data)
train_data = np.sqrt(train_data)
train_data = normalize(train_data)

y = combo_shuffle[1876]
y = np.array(y)
```

```

X_train, X_test, y_train, y_test = train_test_split(train_data, y, test_size
-> = 0.25, random_state = 100)
X_val, X_val_test, y2_train, y2_test = train_test_split(train_data, y,
-> test_size = .01, random_state = 101)
len(X_test)

for var in optimizer.variables():
    var.assign(tf.zeros_like(var))

tf.keras.backend.clear_session()
model_1 = tf.keras.models.Sequential()
model_1.add(tf.keras.layers.Input(shape=(1876,)))
model_1.add(tf.keras.layers.Dense(24, activation='relu'))
model_1.add(tf.keras.layers.Dense(128, activation='relu'))
model_1.add(tf.keras.layers.Dense(1, activation='sigmoid'))
model_1.summary(line_length=55)
loss = tf.keras.metrics.binary_crossentropy
optimizer = tf.keras.optimizers.SGD(learning_rate=1e-3)
metric_list = [tf.keras.metrics.BinaryAccuracy()]
model_1.compile(loss=loss, optimizer=optimizer, metrics=metric_list)

y_pred_vals_model_1 = model_1.predict(X_train, batch_size=512, verbose=1)
y_pred_dense_model_1 = (y_pred_vals_model_1 < 0.5).astype(int)
print(len(y_pred_vals_model_1))
acc_dense_model_1 = accuracy_score(y_train, y_pred_dense_model_1) * 100
print(f"Overall accuracy: {acc_dense_model_1:.2f}%")

%%time
history_1 = model_1.fit(X_train, y_train, epochs=20, verbose=1,
-> validation_data=(X_val, y2_train))

acc_1_train = np.array(history_1.history["binary_accuracy"]) * 100
loss_1_train = np.array(history_1.history["loss"])

acc_1_val = np.array(history_1.history["val_binary_accuracy"]) * 100
loss_1_val = np.array(history_1.history["val_loss"])

epochs_1 = np.arange(1, len(acc_1_train) + 1, 1)
plt.rcParams['axes.linewidth'] = 1

fig_metrics_simple_1, axes_metrics_simple_1 = plt.subplots(ncols=2, figsize
-> =(12, 4))

axes_metrics_simple_1[0].plot(epochs_1, acc_1_train, '-.', label="Training_
-> data")

```



```

axes_metrics_simple_1[0].plot(epochs_1, acc_1_val, '-.', label="Validation_
-> data")
axes_metrics_simple_1[0].set_xticks(epochs_1)
axes_metrics_simple_1[0].set_xlabel("Number_of_epochs")
axes_metrics_simple_1[0].set_ylabel("Accuracy(%)")
axes_metrics_simple_1[0].legend()

axes_metrics_simple_1[1].plot(epochs_1, loss_1_train, '-.', label="Training_
-> data")
axes_metrics_simple_1[1].plot(epochs_1, loss_1_val, '-.', label="Validation_
-> data")
axes_metrics_simple_1[1].set_xticks(epochs_1)
axes_metrics_simple_1[1].set_xlabel("Number_of_epochs")
axes_metrics_simple_1[1].set_ylabel("Cross_entropy_loss")
axes_metrics_simple_1[1].legend()
plt.show()

y_pred_vals_model_1 = model_1.predict(X_test, batch_size=512, verbose=1)
y_pred_dense_model_1 = (y_pred_vals_model_1 > 0.5).astype(int)
print(len(y_pred_vals_model_1))
acc_dense_model_1 = accuracy_score(y_test, y_pred_dense_model_1) * 100
print(f"Overall_accuracy:{acc_dense_model_1:.2f}%")

```

## Processing ROOT Files in Python

```
import uproot as up
import awkward as aw
import numpy as np
import pandas as pd
import sys
import os
import matplotlib.pyplot as plt

file=up.open("ambeRun628793.root")
file.keys()
file_2=up.open('co57Run629846.root')
file2.keys()

data_nr=file["t"]
file.classnames()
data_er=file2["t"]
file2.classnames()

branches_nr=data_nr.arrays()
branches_er=data_er.arrays()

weakn=branches_nr[(branches_nr.a>=5000)]
midn=branches_nr[(branches_nr.a>=15000)]
fulln=branches_nr[(branches_nr.a>=30000)]

weakn_nr=weakn[weakn.a<=15000]
midn_nr=midn[midn.a<=30000]
full_nr=fulln[fulln.a<=50000]

branches_er=data_er.arrays()

weake=branches_er[(branches_er.a>=5000)]
mide=branches_er[(branches_er.a>=15000)]
fulle=branches_er[(branches_er.a>=30000)]

weake_er=weake[weake.a<=15000]
mide_er=mide[mide.a<=30000]
fulle_er=fulle[fulle.a<=50000]

full_nr2=full_nr['s']
fulle_er2=fulle_er['s']
fulle_er2=pd.DataFrame(fulle_er2)
```

```

full_nr2=pd.DataFrame(full_nr2)

midb_nr2=midn_nr['s']
mide_er2=mide_er['s']
mide_er2=pd.DataFrame(mide_er2)
midb_nr2=pd.DataFrame(midb_nr2)

weakb_nr2=weakn_nr['s']
weake_er2=weake_er['s']
weake_er2=pd.DataFrame(weake_er2)
weakb_nr2=pd.DataFrame(weakb_nr2)

fulle_er3=fulle_er2.sample(n=20000)
full_nr3=full_nr2.sample(n=20000)

mide_er3=mide_er2.sample(n=20000)
midb_nr3=midb_nr2.sample(n=20000)

weake_er3=weake_er2.sample(n=10000)
weakb_nr3=weakb_nr2.sample(n=10000)

savetxt('NR_high_energy.csv',full_nr3,delimiter=',')
savetxt('ER_high_energy.csv',fulle_er3,delimiter=',')

savetxt('NR_mid_energy.csv',midb_nr3,delimiter=',')
savetxt('ER_mid_energy.csv',mide_er3,delimiter=',')

savetxt('NR_low_energy.csv',weakb_nr3,delimiter=',')
savetxt('ER_low_energy.csv',weake_er3,delimiter=',')

```

## Waveform Simulation

```
import numpy as np
import pandas as pd
from scipy.stats import norm
from scipy.stats import rv_discrete
from numpy import trapz

ER_data_pack = pd.read_csv('ER_high_energy.csv', header=None)
NR_data_pack = pd.read_csv('NR_high_energy.csv', header=None)

ER_data_pack = ER_data_pack.to_numpy()
NR_data_pack = NR_data_pack.to_numpy()

ER_sum = np.sum(ER_data_pack, axis=0)
NR_sum = np.sum(NR_data_pack, axis=0)

ER_area = trapz(ER_sum, dx=1)
NR_area = trapz(NR_sum, dx=1)

ER_prob = ER_sum/ER_area
NR_prob = NR_sum/NR_area

x = np.arange(0, 100, 1)
y = norm.pdf(x,50,3)*4

ER_numbers = np.arange(len(ER_prob))
NR_numbers = np.arange(len(NR_prob))

ER_rv = rv_discrete(values=(ER_numbers,ER_prob))
NR_rv = rv_discrete(values=(NR_numbers,NR_prob))

peak = y

def peak_gen():
    baseline = np.zeros(1876)
    for i in range(0,19):
        tes5 = ER_rv.rvs(size=20)
        noise = np.random.normal(0, .01, 1876)
        if tes5[i] > 1876 - 50:
            continue
        a = np.zeros(tes5[i]-50)
```

```

        c = np.append(a,peak)
        b = np.zeros(1776-tes5[i]+50)
        d = np.append(c,b)
        baseline += d
        baseline += noise
    return baseline

def peak_gen_nr():
    baseline = np.zeros(1876)
    for i in range(0,19):
        tes6 = NR_rv.rvs(size=20)
        noise = np.random.normal(0, .01, 1876)
        if tes6[i] > 1876 - 50:
            continue
        a = np.zeros(tes6[i]-50)
        c = np.append(a,peak)
        b = np.zeros(1776-tes6[i]+50)
        d = np.append(c,b)
        baseline += d
        baseline += noise
    return baseline

ER_peak_array = []
for _ in range(20000):
    peak_locs=peak_gen()
    ER_peak_array.append(peak_locs)

NR_peak_array = []
for _ in range(20000):
    peak_locs_nr=peak_gen_nr()
    NR_peak_array.append(peak_locs_nr)

NR_samples = np.stack(NR_peak_array)
NR_samples.shape

ER_samples = np.stack(ER_peak_array)
ER_samples.shape

plt.plot(ER_samples[0])
plt.show()

DF_ER = pd.DataFrame(ER_samples)
DF.to_csv("ER_20PE_sim.csv")

DF_NR = pd.DataFrame(NR_samples)

```

```
DF2.to_csv("NR_20PE_sim.csv")
```

## B Package Details

**UpRoot** – Main utility for opening and converting ROOT files into arrays. Unlike PyRoot, UpRoot has no C++ dependencies. UpRoot uses Numpy to cast ROOT structures to Numpy arrays, and can handle large files relatively quickly. Recommended for those wishing to avoid full treatment of data in ROOT.

**Awkward** – Utility for handling jagged arrays and heterogeneous data structures from ROOT files. Highly recommended to pair with UpRoot operations. Can be converted to Numpy or Pandas arrays on the fly.

**Numpy** – Although ubiquitous in Python coding, the use of Numpy in our case relies on its neat data structures and pre-built matrix operations. For example, on page 38, ‘np.square’ trivializes an element by element, row by row multiplication, which would otherwise involve looping or linear operations.

**Pandas** – Another widely used data structure module. Many pre-built functions ease the process of data handling and analysis in Python, such as importing csv files with index and heading control. DataFrames are similar to Numpy arrays, but with a column-first syntax.

**Sklearn** – Workhorse of machine learning processing and performance metrics. Many useful tools built around ML operations, such as training/testing data splits, normalization methods, and accuracy interpretation.

**TensorFlow** – All inclusive machine learning ops. Paired with the Keras layer API, sequentially building models is made somewhat trivial. Simple architectures will have output dimensions automatically aligned with forward layers. All parameters are adjustable, and the backend is transparent for any call. Performance metrics are easily obtained if paired with Sklearn functions.

**Scipy** - A scientific computing platform with many useful tools. Here, we make use of the statistics package to control our random variable and generate our Gaussian peak.

**sys** - Specifically getsizeof, as large files can bottleneck memory limitations unexpectedly. Calling this function around will help track any potential memory issues. It’s nice to know the exact allocation for a given variable too.