FERMILAB-PUB-18-708-AE

arXiv:1811.11308v1 [physics.comp-ph] 27 Nov 2018

# Developing a Bubble Chamber Particle Discriminator Using Semi-Supervised Learning

B. Matusch*[1], C. Amole[2], M. Ardid[3], I. J. Arnquist[4], D. M. Asner†[4], D. Baxter[5, 6], E. Behnke[7], M. Bressler[8], B. Broerman[2], G. Cao[2], C. J. Chen[5], U. Chowdhury‡[2], K. Clark[2], J. I. Collar[6], P. S. Cooper[9], C. B. Coutu[12], C. Cowles[4], M. Crisler[4, 9], G. Crowder[2], N. A. Cruz-Venegas[12], C. E. Dahl[5, 9], M. Das[11], S. Fallows[12], J. Farine[13], I. Felis[3], R. Filgas[14], F. Girard[13, 15], G. Giroux[2], J. Hall[1], C. Hardy[2], O. Harris[16], T. Hillier[13], E. W. Hoppe[4], C. M. Jackson[4], M. Jin[5], L. Klopfenstein[7], C. B. Krauss[12], M. Laurin[15], I. Lawson[1, 13], A. Leblanc[13], I. Levine[7], C. Licciardi[13], W. H. Lippincott[9], B. Loer[4], F. Mamedov[14], P. Mitra[12], C. Moore[2], T. Nania[7], R. Neilson[8], A. J. Noble[2], P. Oedekerk[7], A. Ortega[6], M.-C. Piro[12], A. Plante[15], R. Podviyanuk[13], S. Priya[17], A. E. Robinson[15], S. Sahoo[11], O. Scallon[13], S. Seth[11], A. Sonnenschein[9], N. Starinski[15], I. Štekl[14], T. Sullivan[2], F. Tardif[15], E. Vázquez-Jáuregui[10, 13], N. Walkowski[7], E. Weima[13], U. Wichoski[13], K. Wierman[4], Y. Yan[17], V. Zacek[15], and J. Zhang§[5]

[1]SNOLAB, Lively, Ontario, P3Y 1N2, Canada
[2]Department of Physics, Queen's University, Kingston, K7L 3N6, Canada
[3]Departament de Física Aplicada, IGIC - Universitat Politècnica de València, Gandia 46730 Spain
[4]Pacific Northwest National Laboratory, Richland, Washington 99354, USA
[5]Department of Physics and Astronomy, Northwestern University, Evanston, Illinois 60208, USA
[6]Enrico Fermi Institute, KICP and Department of Physics, University of Chicago, Chicago, Illinois 60637, USA
[7]Department of Physics, Indiana University South Bend, South Bend, Indiana 46634, USA
[8]Department of Physics, Drexel University, Philadelphia, Pennsylvania 19104, USA
[9]Fermi National Accelerator Laboratory, Batavia, Illinois 60510, USA
[10]Instituto de Física, Universidad Nacional Autónoma de México, México D. F. 01000, México
[11]Astroparticle Physics and Cosmology Division, Saha Institute of Nuclear Physics, Kolkata, India
[12]Department of Physics, University of Alberta, Edmonton, T6G 2E1, Canada
[13]Department of Physics, Laurentian University, Sudbury, P3E 2C6, Canada
[14]Institute of Experimental and Applied Physics, Czech Technical University in Prague, Prague, Cz-12800, Czech Republic
[15]Département de Physique, Université de Montréal, Montréal, H3C 3J7, Canada
[16]Northeastern Illinois University, Chicago, Illinois 60625, USA
[17]Materials Research Institute, Penn State, University Park, Pennsylvania 16802, USA

November 2018

**Abstract**

The identification of non-signal events is a major hurdle to overcome for bubble chamber dark matter experiments such as PICO-60. The current practice of manually developing a discriminator function to eliminate background events is difficult when available calibration data is frequently impure and present only in small quantities. In this study, several different discriminator input/preprocessing formats and neural network architectures are applied to the task. First, they are optimized in a supervised learning context. Next, two novel semi-supervised learning algorithms are trained, and found to replicate the Acoustic Parameter (AP) discriminator previously used in PICO-60 with a mean of 97% accuracy.

*Corresponding: brendon.matusch@snolab.ca or analysis@picoexperiment.com
†now at Brookhaven National Laboratory
‡now at Canadian Nuclear Laboratories
§now at Argonne National Laboratory



This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.

## Overview

This paper is organized as follows:

- In Section 1, we outline generally the PICO-60 experiment and the context in which machine learning is applicable.
- In Section 2, we discuss the specific properties of the experiment and its data, as well as the existing signal versus background discriminator.
- In Section 3, we discuss various machine learning techniques that are applied to develop a new discriminator.
- In Section 4, we document experimentation with supervised learning, applied to various data formats.
- In Section 5, we document experimentation with semi-supervised learning with the goal of improving data efficiency.
- In Section 6, we conclude this study and discuss its implications.
- In Section 7, we overview the technologies and software used for this study.

# 1 Objective

In the PICO-60 [Amo+17] experiment, and in other experiments striving to directly detect WIMP dark matter [JKG96], one of the most important and challenging hurdles to overcome is that of background events. Unwanted particles from a variety of sources can produce event signatures that appear very similar to those which are expected to be created by dark matter candidates.

To resolve this, physicists must determine a discriminator function that can, based on experimental data, separate events generated by possible dark matter candidates from those due to background radiation. Two general problems stand in the way:

1. A detailed model of the physical environment is often used to produce an accurate discriminator. This takes a long time to develop and compute, and will have to be updated whenever the physical variables of the experimental apparatus change.
2. A major intent of many dark matter experiments is to reduce the number of background events to the lowest level possible. This means that there will be a very small number detected. This places a significant constraint on the amount of data that is available to optimize such a discriminator.

   Furthermore, what little data is available to optimize a discriminator very often contains impurities, because whatever background radiation is present during WIMP detection runs is also present during calibration runs. These are difficult to separate without already having access to a functional discriminator, creating a "chicken or the egg" problem (from which a common way to escape is to create a time- and labor-intensive simulation).

The objective of this study is to investigate the potential for machine learning techniques to address both of these issues. More specifically, in the context of the PICO-60 experiment, there are two objectives:

1. Determine the most effective machine learning architecture to use in development of a discriminator. Furthermore, develop several neural network architectures and apply them to different input/preprocessing formats (such as audio Fourier transforms, raw waveforms, and images), in order to find which are the most effective.
2. Determine whether these techniques can be extended, applying two original algorithms based on semi-supervised learning, to develop an effective discriminator function based on incomplete and inaccurate initial information.

For clarity, and with specific reference to the PICO-60 experiment, the intent is to accurately separate particle types using only calibration/background data collection run types (henceforth "run types") as training data. While run types are known to be mostly correlated with particle types, the relatively large percentage of impurities has the potential to hinder conventional learning techniques. Specifically, the run types available for training are:

1. Neutron calibration sets, which are known to contain approximately 90% neutrons, and
2. Previously blinded WIMP search runs and diagnostic background runs, which contain approximately 99% alpha particles.

The architectures described in this paper may have wider applications to several similar cases, such as those that require a discriminator which is difficult to construct using conventional techniques. The following are possible reasons for this limitation:

1. Limited ability to collect accurate calibration data, and/or imperfect correlation between calibration data and particle types.
2. Unknown and/or complex interactions in the experimental apparatus make a conventional discriminator difficult to define, where machine learning techniques may be able to uncover hidden correlations in the input data.

# 2 Previous PICO-60 Analysis

## 2.1 Background

### 2.1.1 Introduction

The PICO-60 [Amo+17] bubble chamber experiment, created for the detection of weakly interacting massive particles (WIMPs), was completed in 2017. It provides an excellent basis on which to develop and test the efficacy of new machine learning discriminators, relative to the techniques used during initial analysis of the experiment.

### 2.1.2 About the Detector

The detector was filled with 52 kg of liquid $C_3F_8$, which was held at a constant temperature by a large chilled water bath. The pressure of the target liquid, held inside a synthetic quartz inner vessel, was precisely controlled with propylene glycol as hydraulic fluid inside a stainless steel pressure vessel (Figure 1).

The target was superheated by reducing its pressure until it entered a metastable state in which the vapor phase became energetically accessible. In the presence of a sufficiently dense energy deposition, a bubble would be nucleated and would grow to visible size. Four cameras stereoscopically monitored the chamber for these bubbles, issuing a recompression trigger upon their appearance. At this time, acoustic traces were recorded that captured the sound of bubble formation, and a Dytran 2006V1 fast pressure transducer [Amo17] monitored the pressure rise. The rising pressure caused the vapor bubble to condense as the fluid reverted to a non-superheated state. Following this, the hydraulic system prepared to return the detector to the sensitive superheated state by once again expanding to a lower pressure.

Superheated fluid detectors have an intrinsic insensitivity to the relatively diffuse energy depositions from electron recoils [Amo+17], and neutron backgrounds are mitigated by shielding the detector volume in a 20 t water tank. Alpha decays along the U/Th chains present the dominant remaining background which must be distinguished from nuclear recoils acoustically [HPS53].

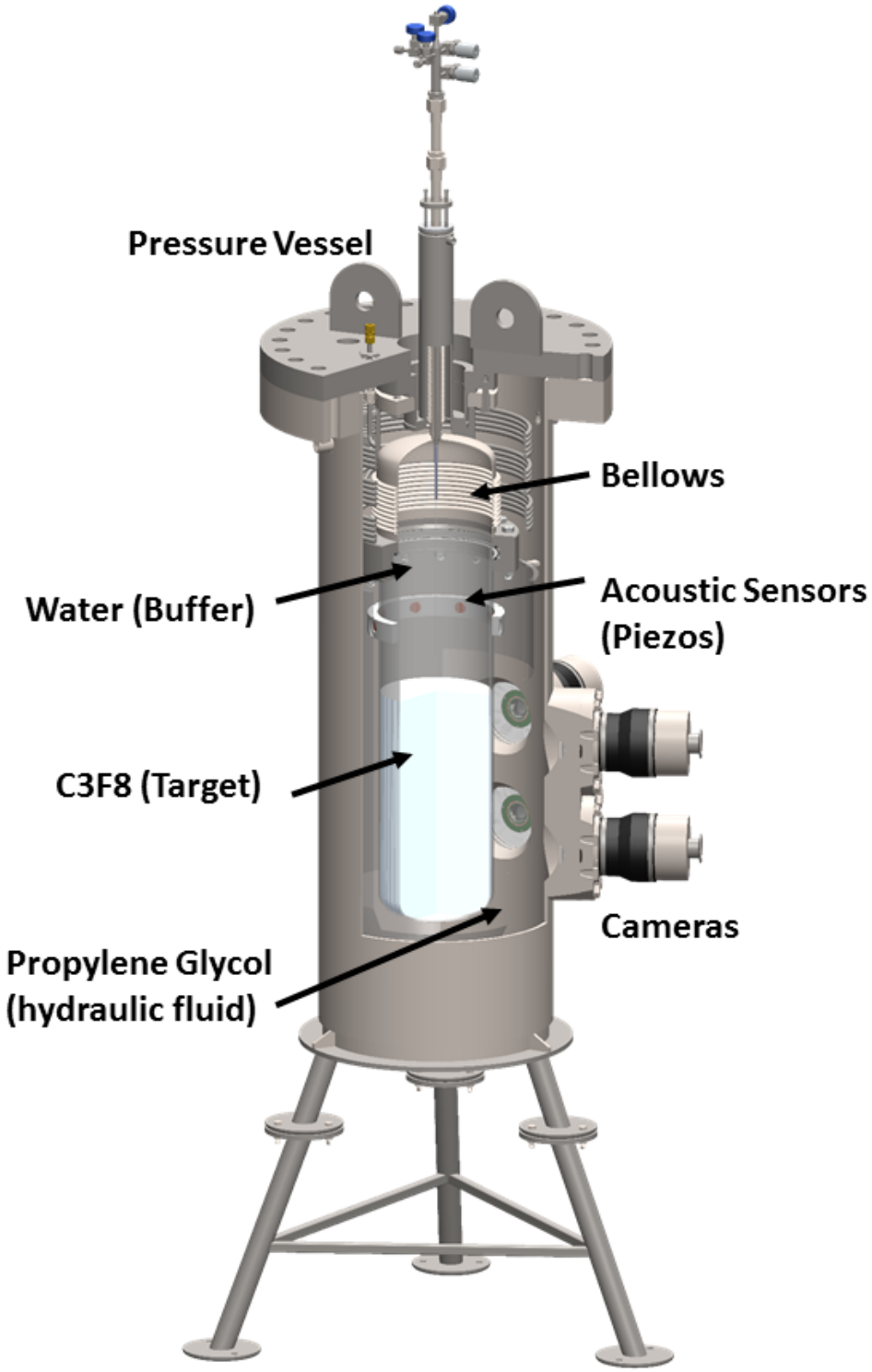


Figure 1: CAD rendering of the PICO-60 detector as configured for its operation with $C_3F_8$.

### 2.1.3 Properties of Events

The nuclear recoils produced by WIMP candidates (in the cross-section and mass ranges to which the PICO-60 detector is sensitive) are predicted to be indistinguishable from those induced by neutrons. The key difference is that neutrons frequently (but not always) scatter several times and produce multiple bubbles, where the extremely small predicted cross-sections of WIMP candidates mean that they should almost invariably create just one bubble. This means single-bubble neutron events can be used to optimize a discriminator to detect WIMP events.

The approximate expected ratio between single-bubble and multiple-bubble events generated by neutrons is known based on Monte Carlo simulation and extrapolations from experience with previous detectors. A measured quantity of single-bubble events generated by nuclear recoils, in excess of this prediction, would be indicative of WIMP interactions. Consequently, it is imperative to isolate only those events associated with nuclear recoils.

## 2.2 Discrimination Techniques

### 2.2.1 Overview

In this study, all newly developed machine learning-based discriminators were compared to the two discrimination techniques used previously for the PICO-60 experiment: Acoustic Parameter and machine learning.

### 2.2.2 Acoustic Parameter

An Acoustic Parameter method (furthermore AP) is the technique that was used for event discrimination in the original PICO-60 analysis. A function was defined that discriminates between alpha particles and nuclear recoils based on audio data. It was derived based on a physical model of the bubble chamber, and was tuned using neutron calibration sources to produce events in the PICO-60 apparatus.

Before AP can be calculated, the audio data must first be converted into the frequency domain (specifically, with the banded Fourier transform $\beta_8$). It must also be preprocessed with the position correction function (discussed in Section 2.3.2), which adjusts the overall amplitude of an audio recording according to its position in the vessel. This is done because the acoustic characteristics differ significantly depending on the position of the event in the vessel [Amo+17].

Finally, a set of data cuts must be applied to remove classes of events which are not accurately classified using AP alone. These classes include events originating close to the walls of the detector, and multiple-bubble events.

These conversion, preprocessing, and data cutting steps were reapplied during this study, and are described in Section 2.3.

### 2.2.3 Original Machine Learning Analysis

As part of the PICO-60 experiment, some testing was conducted to determine whether machine learning (in the form of a multi-layer perceptron) could be a viable technique to consider for a discrimination function. The results, published in the 2017 PICO-60 paper [Amo+17], indicated that the general technique warranted further investigation.

The neural network in this study used the same $\beta_8$ input format as was applied in this study (detailed in Section 2.3.1), with pre-trigger noise subtracted. Position corrections (also in Section 2.3.1) were not applied; rather, the position of the event in the vessel is additionally used as input. AP was used to restrict the data used as input to the neural network. In addition to neutrons and alpha particles, gamma events were used as training and testing data for this neural network.

Note that it is possible there are position-related biases present in this training data, because neutron events during calibration runs occur more frequently near the wall of the detector where the calibration source is placed. These biases are expected to be alleviated by application of position corrections which normalize the audio amplitude; these corrections are detailed in Section 2.3.1.

Software for this experiment was implemented using the MATLAB Neural Network Toolbox [Inc17].

## 2.3 Data Sources and Formats

Discriminators can use a variety of different data formats to make predictions. These are collected from the two essential sensors present in the PICO-60 apparatus: two piezoelectric microphones (piezos) and cameras.

### 2.3.1 Piezo-Derived

*Raw Waveform*

The lowest-level data derived from the two piezos in the PICO-60 apparatus is the raw waveform $\omega$. This consists of a series of 250,000 samples, collected at the rate of one per microsecond. Each sample is represented as a 16-bit integer. Only the section of the audio between sample 90,000 (inclusive) and sample 190,000 (exclusive) was used during most experiments, because there is only background noise prior to this window, and a clipped signal produced by hydraulics repressurizing the vessel after. These sections contain no information and can be undesirably fitted by neural networks. (For context, approximately half of the samples are recorded prior to the camera trigger.)

*Fourier Transform*

The raw waveform $\omega$ was converted into the frequency domain by means of a one-dimensional Discrete Fourier Transform (DFT) for real input. Since DFTs produce sequences of complex numbers, the magnitude of each element was computed. The direct output from such a DFT is the full-resolution Fourier transform $\beta_{50,001}$, which consists of a sequence of 50,001 numbers (half the length of the raw waveform $\omega$).

Beyond this, the banded Fourier transform $\beta_8$ (which is the input to the AP function and the original machine learning analysis) was computed by integrating the resonant energy over all frequencies within each of a set of eight frequency bands ranging from 1 kHz to 300 kHz [Amo+17]. This can be thought of as a method of signal downscaling, compressing the information into a smaller number of data points.

Both $\beta_{50,001}$ and $\beta_8$ were used as inputs to machine learning models applied in this study.

### 2.3.2 Camera-Derived

*Image Window Sequence*

Bubble events are detected using the four cameras within the PICO-60 apparatus, with an image-based trigger. When this happens, each camera records a sequence of 41 images before and after formation. These raw images contain a large amount of extraneous information; they encompass the entire vessel. To reduce the input information, the image window sequence $\iota$ includes 50×50 cropped windows around the position of the bubble. The 41 frames, many of which contain either no bubble or a bubble in later stages of formation, are reduced to 10 frames immediately around the formation of the bubble. Two to three frames from before the recording trigger are included in $\iota$, because the bubble does not cause a trigger until it is already of a significant size.

*3D Position*

The 3-dimensional position $\chi$ of the bubble within the vessel is calculated using triangulation, based on the known positions and angles of the cameras and the position of the bubble within the field of view of each camera.

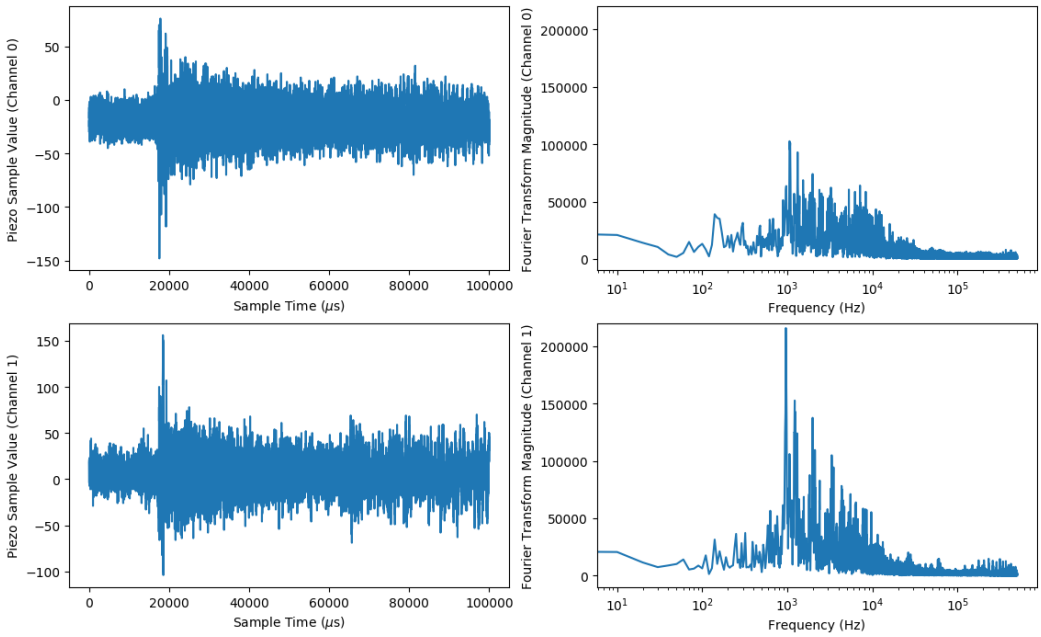


Figure 2: An example of the audio waveform $\omega$ with initial background and hydraulic noise cropped (left) and the resulting full resolution Fourier transform $\beta_{50,001}$ (right) for both audio channels. Recorded on 2016-12-03 at 04:22:45 UTC.

This is used in the banded frequency position correction function $PosCor(\beta_8, \chi)$, which corrects $\beta_8$ for variations in amplitude which depend on the position of the bubble. $PosCor(\beta_8, \chi)$ is used for training several machine learning models throughout this study. Details of its use are in Section 4.3.

## 2.4 Data Cuts

Techniques in this study are trained and validated on a number of different data sets. The selection of these sets focused on the fundamental trade-off between quality and quantity of data; by setting a higher standard for the validity of training data, one has less data to train on.

The initial data set $D$ to which these cuts are applied consists of all events recorded in the PICO-60 detector, at the 3.3keV threshold, between 2016-09-02 and 2017-06-20.

### 2.4.1 Basic Quality Cut

A number of cuts are necessarily applied to all data to ensure meaningful results. Otherwise, significant overfitting on biases in the data is likely. This basic quality cut $QualCut(D)$ consists of the following restrictions:

- The run was not collected during engineering or testing
- The recording process was triggered by the camera (as opposed to a manual trigger, et cetera)
- AP is not erroneously large and negative ($log_{10}(AP) > -100$)
- The event was recorded more than 25 seconds after reaching target pressure

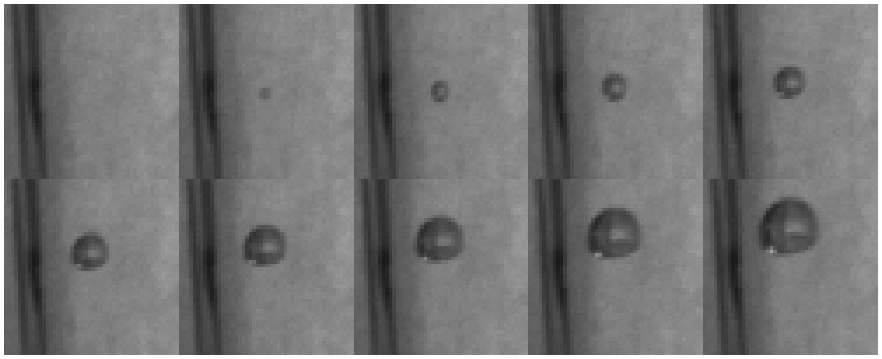

Figure 3: An example of the image sequence $\iota$. Recorded on 2016-09-23 at 22:15:44 UTC.

- The bubble position $\chi$ was successfully calculated ($[\chi_X, \chi_Y, \chi_Z] \neq [-100, -100, -100]$)

#### 2.4.2 Bubble Multiplicity Cut

PICO-60 events which include multiple bubbles are *always* neutron events; alpha particles are stopped electromagnetically, and WIMP candidates have a negligible probability of multiple scattering because of their extremely low predicted cross section. Thus, multi-bubble events can be safely removed. The bubble multiplicity cut $MultiCut(D)$ consists of the following restrictions:

- Either 0 or 1 bubbles are detected based on images from the camera
- The number of bubbles approximated using the fast pressure transducer is close to 1

#### 2.4.3 Wall Cut

Events that occur near the walls of the vessel have acoustic properties which are very different from events nearer the center of the vessel. It can be desirable for a discriminator to handle these events correctly; however, AP alone does not, and neither does the neural network used in the previous PICO-60 paper. Thus, removing wall events allowed for a more meaningful direct comparison between a new neural network and existing techniques.

The complete wall cut $WallCut(D)$ is a composition of the fiducial cut $FidWallCut(D)$ (which makes use of the 3D position $\chi$), the pressure cut $PresWallCut(D)$ (which uses data from the fast pressure transducer), and the acoustic cut $AcWallCut(D)$ (which uses the banded Fourier transform $\beta_8$). Those cuts restrict data as follows:

- The fiducial cut $FidWallCut(D)$ defines a spatial area along the walls of the vessel within which no events are accepted. It determines whether events fall within this area based on the visually calculated position $\chi$.
- The pressure cut $PresWallCut(D)$ restricts the pressure detected by the fast pressure transducer, without any position corrections, to be within a range of 0.3 of 1
- The acoustic cut $AcWallCut(D)$ is defined using the banded Fourier transform $\beta_8$, and takes advantage of differences in the frequency distribution (specifically the first and second frequency bands of the first and third piezos) of wall events and non-wall events.

In Figure 4, the relative and combined effectiveness of each of these cuts can be seen. It is clear that most of the time, both of these cuts agree, but there are certain cases in which only one or the other eliminates an event.

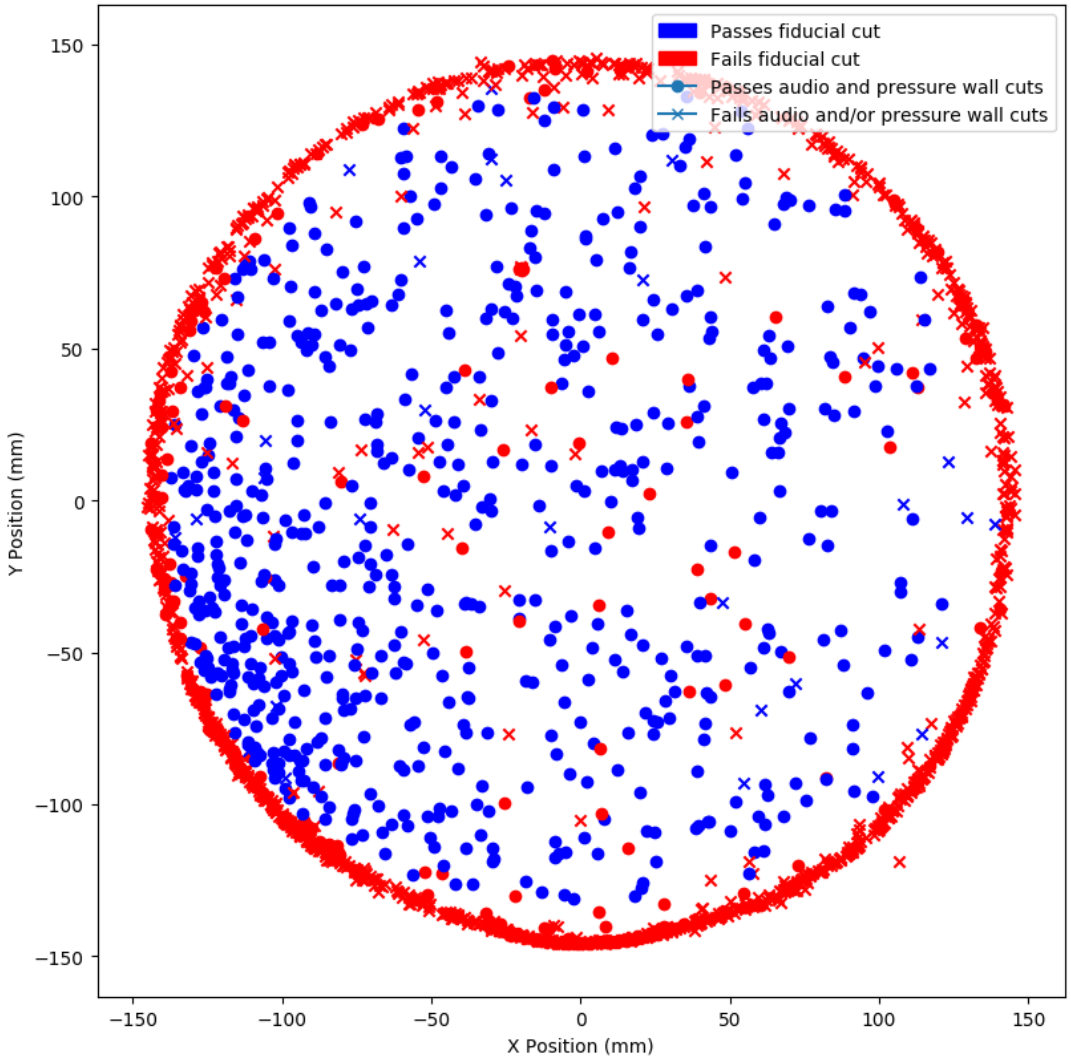


Figure 4: A visualization of the fiducial cuts and how they match up with the pressure and acoustic cuts.

Some events eliminated by fiducial cuts appear to be in the middle of the vessel in Figure 4. These events are at the top or bottom of the detector (along the Z axis, which is not displayed in the 2D plot).

The data set, after all of these cuts are applied, consists of 624 events. During each training run, 128 randomly selected events are set aside for validation, leaving a training set of 496 events.

# 3 Machine Learning Techniques

## 3.1 Background

While neural networks have existed for many years, recent developments in computing power, most importantly GPU acceleration [Cir+11], have opened up a wide variety of applications and fields in which they can now be used.

One of the key benefits of machine learning systems is that they have the capacity to approximate arbitrary functions (such as AP) without human intervention. This greatly alleviates the need for human programmers to define and optimize the exact algorithm used.

This characteristic has already shown wide-ranging implications in many fields, such as autonomous driving [Boj+16] and detection of heart arrhythmia [Raj+17]. For one, it has the potential to drastically increase the speed of iteration for the people working on a project, since retraining a machine learning system when a hardware or software variable changes is much faster than calibrating a human-designed model.

## 3.2 Techniques Considered

Machine learning techniques are usually divided into two main categories:

- Supervised learning, in which a system is trained on a set of fully-labeled data to classify unseen examples according to the patterns it observes in the training set, and
- Unsupervised learning, in which a system finds patterns or clusters in a set of unlabeled data. This can find order in almost any training set, but it is unpredictable and may not find the particular patterns desired.

A significant challenge in this application is that neither supervised nor unsupervised learning is ideal. The calibration and background runs available for training are not pure, so supervised learning is likely to overfit on biases and produce undesirable results. Conversely, unsupervised learning may be able to find clusters in the data, but it is impossible to guarantee that it will distinguish between nuclear recoils and alpha particles as opposed to some other binary separation.

For these reasons, a technique is needed that is not as sensitive to problematic training data as supervised learning, but more predictable and controllable than unsupervised learning. Semi-supervised learning is a middle ground in this regard. It makes use of a labeled set as well as an unlabeled set. It uses the labeled set for training, and uses the unlabeled set to further structure the patterns it finds in the training set (which may not be sufficiently large to apply to supervised learning).

Two original semi-supervised algorithms were developed and implemented for this study: gravitational differentiation (GD) and iterative cluster nucleation (ICN). Both are discussed in detail in Section 5.

## 3.3 Performance Analysis

Performance of the machine learning systems was evaluated using two metrics: classification accuracy and class-wise standard deviation.

Classification accuracy is the ability of the network to separate the events into the two desired classes. Numerically, this corresponds to the number of validation examples the network classifies correctly. The baseline for correctness is the run type during supervised learning experiments, and the AP prediction for semi-supervised experiments.

The correctness of the network's predictions is further detailed in the precision and recall statistics. Precision represents the proportion, of those that the network predicts to be nuclear recoils, that are actually nuclear recoils (or part of the corresponding calibration set). It is an indicator of how pure the network's predictions

are for the recoil class. Recall represents the proportion, of those that are actually nuclear recoils, that the network predicts to be nuclear recoils. It indicates how many examples the network misses (i.e., those that would erroneously be removed as background radiation).

Class-wise standard deviation, abbreviated CWSD, captures how decisive the network is: whether its prediction is confidently high or low, or is only slightly closer to one edge of the spectrum.

Numerically, this is a variable defined below, where $N$ and $A$ are the sets of outputs of the supervised learning discriminator in question, corresponding to the sets of neutrons and alpha particles respectively (or calibration and background sets respectively, depending on which is used as ground truth data). It calculates the spread of the network's predictions for each ground truth class. This means that a decisive discriminator, which produces a wide separation between the two classes, is preferred over one that produces a nebulous cloud of outputs with a seemingly arbitrary decision boundary.

$$S = std(N \cup A) \tag{1}$$

$$CWSD = (std(N \div S) + std(A \div S)) \div 2 \tag{2}$$

The first step is to calculate the standard deviation $S$ of the union of $N$ and $A$. This gives an indication of the scale of the overall distribution. When $N$ and $A$ are divided by $S$, they are normalized so that the standard deviation of their union is equal to 1. While the neural network's outputs are bounded in the range of 0 to 1 with a sigmoid activation on the last layer, AP has a significantly wider range. Normalization of the union prevents this from creating a bias where AP would produce a higher standard deviation with a similarly proportioned error.

The second step is to calculate the mean of the standard deviations of the normalized sets of neutrons and alpha particles individually. This is an indication of how tightly clustered or widely dispersed the discriminator's predictions are for each class. Very consistent predictions of $x$ for neutrons and $y$ for alphas (for any given $x$ and $y$), with minimal variance off those specific values, will produce a low class-wise standard deviation.

### 3.4 Optimization Process

Each of the learning algorithms used in this study applies a neural network. For each of the general configurations (multi-layer perceptron, one-dimensional convolutional neural network, et cetera) there are many possible specific architectures with different hyperparameters, which have to be optimized to improve performance. Specific hyperparameters that were optimized include:

- Number of layers of each type (dense, convolutional)
- Number of neurons in dense layers
- Number of filters in convolutional layers (depth of output tensor)
- Kernel size in convolutional layers (spatial area that a single filter covers)
- Stride in convolutional layers (spacing between kernel positions during convolution)
- Dropout regularization parameter (proportion of neurons to randomly remove for any given training example)
- L2 regularization $\lambda$ (multiplier for squared weights before adding to loss function)

Several of these hyperparameters are optimized at a time, using a grid search. Given $n$ different hyperparameters to optimize and $m$ different possible values for each, $m^n$ different networks are trained (using every possible combination of hyperparameters) and tested on a validation set. For space reasons, only a small subset of the tested configurations are displayed in tables throughout this paper.

In general, the parameters for each grid search were chosen based on some initial empirical experimentation to determine the general range within the entire parameter space that work reasonably well. There was a limit to how many parameters could be tested, especially in the case of semi-supervised learning, as some of the grid searches took several days to run.

Where not otherwise specified, the following hyperparameters were used universally:

- Adam [KB14] stochastic optimizer
- Mean squared error loss function
- Batch size of 32
- *tanh* activation function

# 4 Supervised Learning

## 4.1 Overview

The primary objective, in experimentation with supervised learning, is to determine the most effective combination of input format and neural network architecture to use in replicating a discriminator function. Three major configurations were tested:

1. A convolutional neural network trained on the raw waveform $\omega$.
2. A dense neural network trained on banded Fourier transforms $\beta_N$.
3. A convolutional neural network trained on the image window data $\iota$.

All supervised learning configurations were trained and evaluated based on their ability to determine the origin of events: whether they are from neutron calibration source runs (predominantly neutrons) or background radiation runs (predominantly alpha particles). This metric was used for training and testing instead of AP for two reasons:

1. During early operations of future experiments, when no AP equivalent is yet available, impure data is the only information available for training. The network structure should be selected to accommodate this.
2. In general, supervised learning systems learn to replicate biases present in the training data set. Thus, to evaluate the network's effectiveness at processing the input data, the run type (which is used for training) must be used as a baseline instead of AP (which is never provided to the network during the training process).

## 4.2 Convolutional Neural Network for Raw Waveform Analysis

### 4.2.1 Structure

A convolutional neural network was used for analyzing the raw waveform $\omega$ directly, without any preprocessing whatsoever. It is furthermore referred to as $DeepConv(\omega)$. Direct processing of the waveform avoids any destruction of information; theoretically, a sufficiently complex neural network should be capable of deriving any characteristics needed.

For this task, a very deep (20 layers in its shallowest configuration) one-dimensional fully convolutional neural network was applied. The architecture was inspired by the M34-res network [Dai+16] for analysis of raw waveforms. L2 regularization was used to alleviate overfitting, and batch normalization was applied to the input to stabilize training.

The following hyperparameters were tested during a grid search:

- L2 Regularization $\lambda \in \{0.0003, 0.001, 0.003\}$
- Dense Layer Dropout $\in \{0, 0.25, 0.5\}$
- Convolutional Filters $\in \{24, 48\}$
- Convolutional Kernel Size $\in \{3, 5\}$
- Number of Convolutional Layers $\in \{17, 32\}$
- Number of Dense Layers $= 3$ (with neuron counts 64, 16, and 1)

#### 4.2.2 Results

The most accurate configuration during a waveform CNN grid search was 95% accurate and had a standard deviation of 0.52 (seen in Figure 5).

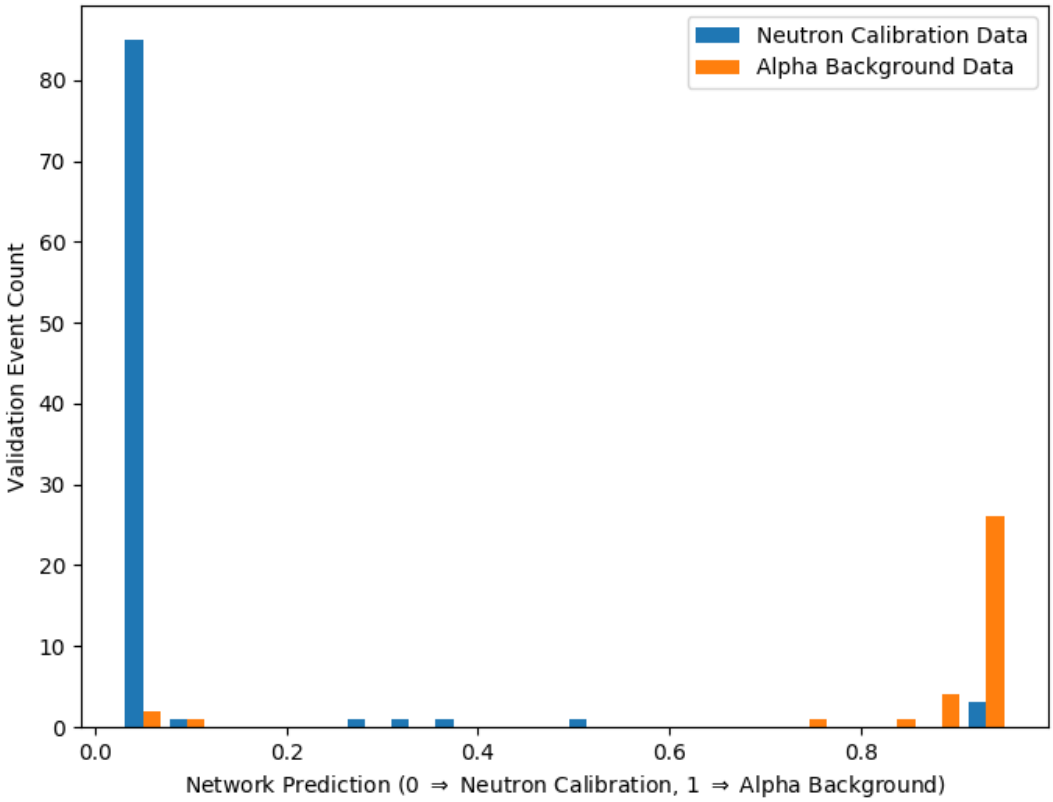


Figure 5: Prediction distribution of the best *DeepConv*($\omega$) discriminator. (Validation Event Count is the number of events in the validation set that fall within a certain network prediction band.)

Table 1 shows the results of several *DeepConv*($\omega$) discriminators produced throughout testing, with corresponding network hyperparameters.

Table 1: Performance of *DeepConv*($\omega$) configurations.

| L2 $\lambda$ | Dropout | Filters | Kernel | Conv Layers | Max Accuracy | Precision | Recall | CWSD |
|---|---|---|---|---|---|---|---|---|
| 0.003 | 0 | 24 | 3 | 17 | 95% | 97% | 97% | 0.52 |
| 0.003 | 0.25 | 48 | 3 | 17 | 94% | 100% | 91% | 0.44 |
| 0.001 | 0.25 | 24 | 5 | 17 | 95% | 99% | 94% | 0.46 |
| 0.001 | 0.25 | 24 | 5 | 32 | 91% | 94% | 94% | 0.66 |
| 0.0003 | 0 | 48 | 5 | 32 | 88% | 94% | 89% | 0.61 |
| 0.0003 | 0 | 24 | 3 | 17 | 93% | 96% | 95% | 0.59 |

Table 1 shows that simple network architectures seem to produce better accuracy, likely due to reduced overfitting. Larger kernel sizes, more filters, and more layers make performance generally worse, while higher L2 regularization does the opposite. Dropout seems to have a weak correlation, possibly reducing accuracy slightly.

## 4.3 Multi-Layer Perceptron for Fourier Transform Analysis

### 4.3.1 Structure

Preprocessing the audio by applying a Fourier transform may be advantageous. If the frequency distribution and overall amplitude are indeed the most important factors for discrimination, the network can gather them straight from the Fourier transform $\beta_N$ rather than having to analyze $\omega$ to extract this information.

In addition to the banded Fourier transform $\beta_8$ used in the original PICO-60 analysis, the full-resolution $\beta_{50,001}$ was input into a neural network directly, without any downscaling.

When $\beta_8$ was used as input, position corrections were optionally applied to the input data, as they were the input to Acoustic Parameter. The resulting configurations are furthermore referred to as $FourierMLP(\beta_N)$ and $FourierMLP(PosCor(\beta_N, \chi))$. While a variety of network architectures were tested, most had a small number of dense layers (on the order of three), applied dropout and L2 regularization, and used batch normalization once again.

The following hyperparameters were tested during a grid search:

- Dropout $\in \{0, 0.25, 0.5\}$
- L2 Regularization $\lambda \in \{0, 0.0003, 0.001, 0.003, 0.01\}$
- Number of Dense Layers $\in \{2, 3, 4\}$

### 4.3.2 Results

It became evident very quickly that a high resolution was not required to get a high accuracy relative to the ground truth data. *FourierMLP*($\beta_8$) managed an excellent 98% validation accuracy, and also produced a mean class-wise standard deviation of 0.29. Its very decisive prediction distribution can be observed in Figure 6.

Very interestingly, *FourierMLP*($\beta_8$) performed better than *FourierMLP*($PosCor(\beta_8), \chi)$), which produced an accuracy value of 96%. Both of these results were better than any of the network architectures trained on $\omega$.

Table 2 compares various multi-layer perceptron discriminators, with and without wall cuts.

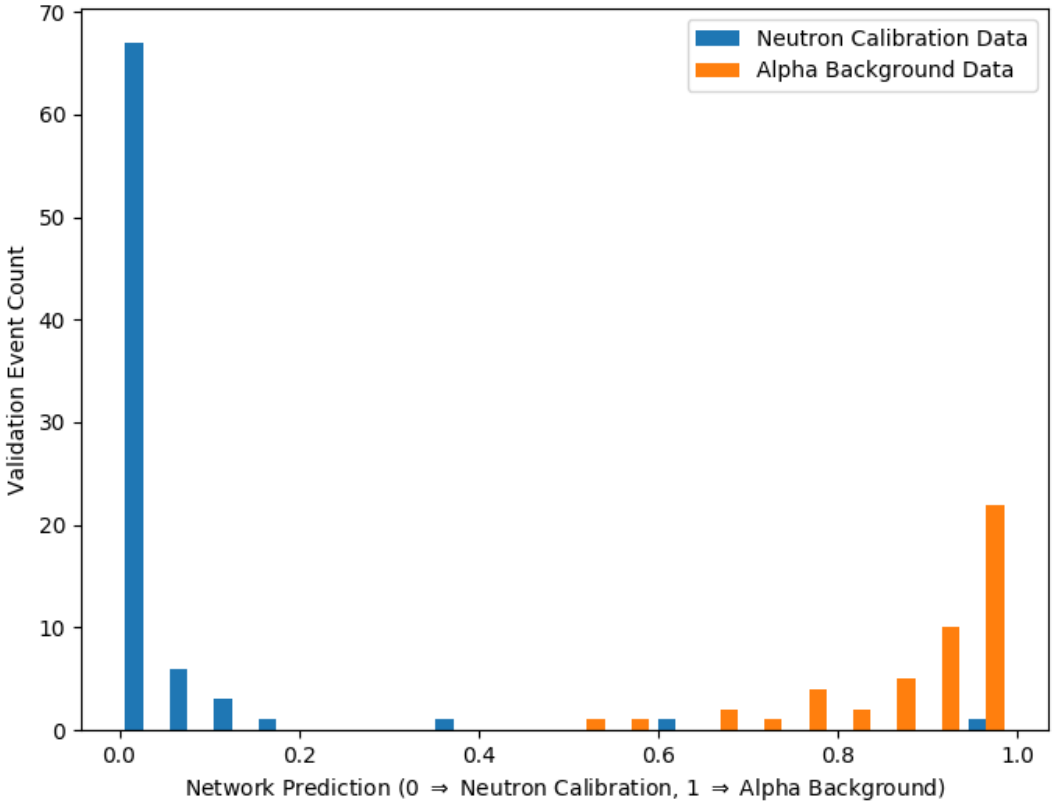


Figure 6: Prediction distribution of the best $FourierMLP(\beta_8)$ discriminator.

Table 2: Performance of $FourierMLP(\beta_N)$ configurations.

| Configuration | L2 $\lambda$ | Dropout | Layers | Max Acc | Prec | Recall | CWSD |
|---|---|---|---|---|---|---|---|
| $FourierMLP(PosCor(\beta_8, \chi))$ | 0 | 0.5 | 2 | 96% | 98% | 97% | 0.42 |
| $FourierMLP(\beta_8)$ | 0 | 0.5 | 2 | 98% | 100% | 98% | 0.29 |
| $FourierMLP(\beta_{50,001})$ | 0 | 0 | 2 | 95% | 94% | 98% | 0.47 |
| $FourierMLP(\beta_{50,001})$ | 0.001 | 0 | 4 | 95% | 95% | 98% | 0.46 |
| $FourierMLP(\beta_{50,001})$ | 0.001 | 0.5 | 4 | 94% | 93% | 98% | 0.54 |
| $FourierMLP(\beta_{50,001})$ | 0.003 | 0.25 | 3 | 93% | 92% | 98% | 0.57 |

It is very clear that higher resolution data does not improve accuracy in a relatively shallow network. Regularization, both dropout and L2, appears to have relatively little effect.

## 4.4 Convolutional Neural Network for Image Window Analysis

### 4.4.1 Structure

It is an open question whether or not there is any information in the image data $\iota$ that could be used to distinguish between particle classes. Relative to the extremely short period of time in which the bubble forms (on the scale of nanoseconds), the framerate of the camera (340Hz) is extremely slow. While the very early stages of bubble formation (when the sound is produced) are known to differ depending on whether

the bubble was created by a nuclear recoil or an alpha particle, it is not experimentally known whether any visually apparent differences persist when the bubble is visible.

In an effort to resolve this, a two-dimensional convolutional neural network was applied to the task of discriminating based on $\iota$, referred to as $ImageConv(\iota)$. The network architecture consists of a moderate number of convolutional layers (on the scale of nine) and three dense layers at the end. L2 regularization was used on all layers, and dropout was additionally used on the dense layers.

The following hyperparameters were tested during a grid search:

- Dropout $\in \{0, 0.25, 0.5\}$
- L2 Regularization $\lambda \in \{0, 0.0003, 0.001, 0.003, 0.006, 0.01\}$
- Number of Convolutional Layers $\in \{6, 9, 12\}$
- Number of Dense Layers = 3 (with neuron counts 64, 16, and 1)

#### 4.4.2 Results

Throughout a variety of different network architectures, the best validation accuracy obtained was 69%. The corresponding training accuracy of 100% indicates that severe overfitting is taking place (not unexpected, given the relatively small image data set). Also, the class-wise standard deviation was a very high 0.99, which implies its decisions are nearly random (as seen in Figure 7). It is likely fitting poorly on some form of noise within $\iota$. The fact that throughout many trials during a grid search, no good performance on validation data was observed, provides significant evidence that images provide insufficient information for effective discrimination.

Table 3: Performance of $ImageConv(\iota)$ configurations.

| L2 $\lambda$ | Dropout | Convolutional Layers | Dense Layers | Max Accuracy | Precision | Recall | CWSD |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 6 | 3 | 69% | 69% | 100% | 0.99 |
| 0.01 | 0.25 | 6 | 3 | 68% | 68% | 100% | 0.98 |
| 0.003 | 0.25 | 9 | 3 | 68% | 68% | 100% | 0.98 |
| 0.01 | 0.25 | 12 | 3 | 68% | 68% | 100% | 0.97 |
| 0.01 | 0.5 | 12 | 3 | 68% | 68% | 100% | 0.99 |

In Table 3, high standard deviations across the board are further evidence for the hypothesis that there is insufficient information in the image data. The 100% recall statistics are not due to high classification accuracy, but rather due to the network making predictions less than 0.5 for all examples of both classes.

### 4.5 Performance Summary

Table 4 compares the accuracy and mean class-wise standard deviation values of certain successful supervised learning configurations with multiple input formats.

Table 4: Summary of supervised learning configurations.

| Configuration | Max Accuracy | Precision | Recall | CWSD |
|---|---|---|---|---|
| $DeepConv(\omega)$ | 95% | 97% | 97% | 0.52 |
| $FourierMLP(PosCor(\beta_8))$ | 96% | 98% | 97% | 0.42 |
| $FourierMLP(\beta_8)$ | 98% | 100% | 98% | 0.29 |
| $FourierMLP(\beta_{50,001})$ | 95% | 95% | 98% | 0.46 |
| $ImageConv(\iota)$ | 68% | 68% | 100% | 0.97 |

The key conclusion to be drawn from the above table is that the low-resolution Fourier transform $\beta_8$ is by far the most effective input format. Higher resolutions seem to make performance worse, likely due to overfitting.

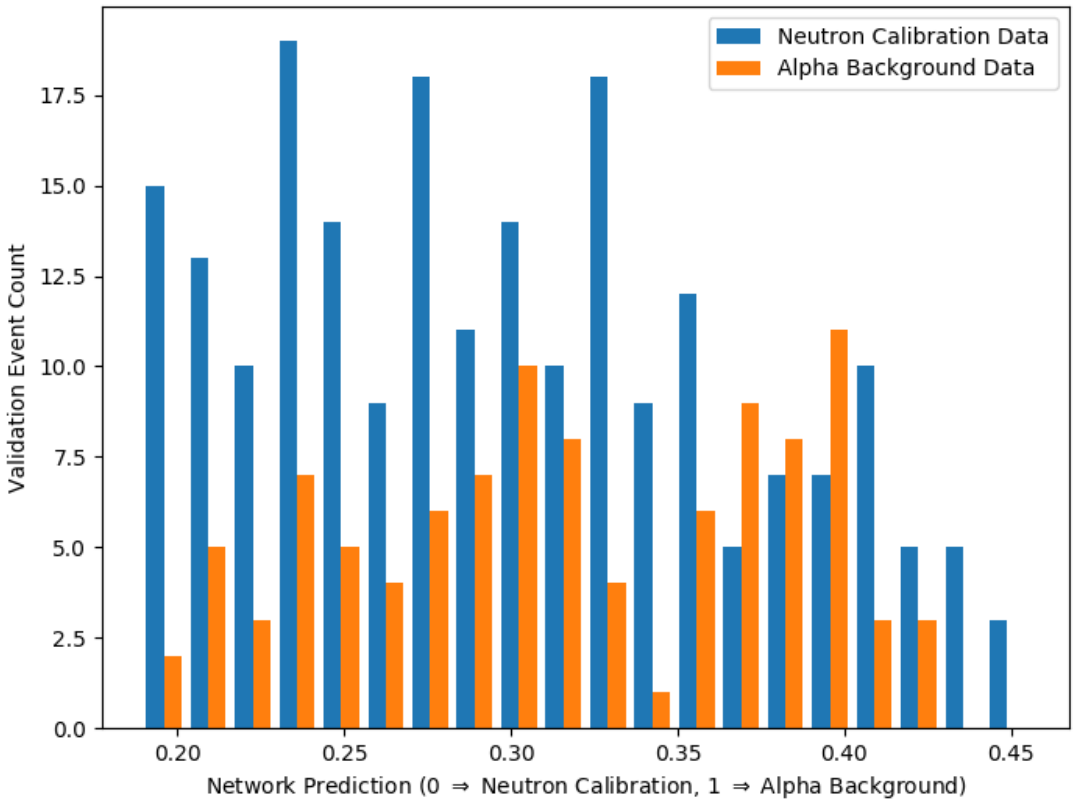


Figure 7: Prediction distribution of one of the best $ImageConv(\iota)$ discriminators. (Note the truncated X axis.)

Based on this result, experimentation with semi-supervised learning proceeded using $FourierMLP(PosCor(\beta_8, \chi))$, applying a three-layer perceptron architecture. There were three main reasons for this:

1. This was a highly effective configuration, producing 96% accuracy and a class-wise standard deviation dramatically better than the original PICO-60 neural network on that data set.
2. The use of the exact same input format as AP makes the algorithms more directly comparable.
3. If position correction is not applied, there is some risk of introducing a bias to the network's training data because there are different position distributions of the two classes (as described in Section 2.2.3).

# 5 Semi-Supervised Learning

## 5.1 Overview

Semi-supervised learning is an uncommon machine learning technique, relative to widely-used supervised and unsupervised learning. The concept is to train a machine learning model on a set of labeled (classified) data in addition to a set of unlabeled (unclassified) data.

In the context of PICO-60, the labeled sets consist of the neutron calibration runs, and the background radiation runs (which consist predominantly of alpha particles). In general, these labeled sets should have a strong but not perfect correlation with particle types. The unlabeled data can consist of any mixture of particle types.

The primary advantage of semi-supervised learning is clear: it requires fewer labels to be collected than conventional supervised learning. A less obvious but powerful secondary advantage is that, since the learning process allows the network to reinforce its own decisions, the negative effect of impure training data is minimized, potentially allowing the resulting network to perform better than a network trained with supervised learning.

Two techniques for semi-supervised learning have been developed for this study: gravitational differentiation and iterative cluster nucleation. The essence of both algorithms is a positive feedback loop. First, the network is trained on a smaller amount of imperfect data. Throughout the training process, the network runs predictions on unlabeled data. By using its most confident predictions in the training process, the desired function (separating particle types) is continuously reinforced, while less confident predictions are de-emphasized. This effectively draws new, unlabeled data into the training set.

Because all data in the PICO-60 experiment is labeled, unlabeled training data sets are randomly sampled from the 496 training examples remaining (after the validation set is selected).

Since the intent of these techniques is to separate particle types (the same as AP), performance evaluations (accuracy and class-wise standard deviation) use AP as a baseline. All graphs in this section use AP's predictions to define the sets of neutrons and alpha particles.

To ensure that accuracy statistics are as consistent and reliable as possible, every configuration was trained three times, with a different randomly selected validation set each time. (Using more than three training runs would have taken prohibitive amounts of time.) Accuracy, precision, recall, and class-wise standard deviation statistics are the mean over those three tests.

## 5.2 Gravitational Differentiation

### 5.2.1 Concept

Gravitational differentiation (GD) is a novel technique, developed during this study, for training a neural network in a semi-supervised fashion on a relatively small set of imperfectly labeled data, while simultaneously incorporating the network's changing predictions on a set of unlabeled data to encourage decisive classifications.

In general terms, it amplifies the training effect associated with high-confidence predictions (those that most clearly show characteristics of either an alpha particle or a neutron) on unlabeled examples. Meanwhile, it minimizes the training impact of low-confidence predictions, such that the network will not be trained to make incorrect predictions.

Analogous to a gravitational effect, events with high-confidence predictions are attracted closer to their corresponding classifications. They are then used as training data, allowing the patterns they represent to be used more generally to make predictions on other events.

The system accomplishes this using a new method for calculating gradients for the last layer of the neural network. Based on the current predictions of a trained network on a certain unlabeled training example, a gradient is calculated for the last layer with the following purposes:

- For examples with confident predictions close to 0 or 1: Create a large gradient that pulls the prediction closer to 0 or 1 through gradient descent (like gravity).
- For examples with low-confidence predictions close to 0.5: Create a gradient near 0 that does not affect training significantly.

The gradients produced by this technique are used for training in the same way as gradients calculated with a loss function. Backpropagation is used to calculate gradients for previous layers, and a stochastic gradient descent optimizer [KW52] is used for weight updates.

### 5.2.2 Algorithm

Gradients are calculated using the gravitational differentiation function $\text{GravDiff}(p, \psi, g)$, which is parameterized by the network's existing prediction $p$ on the training example in question, the degree $\psi$ of the piecewise exponential function used to distort the response of the gradient, and the gravitational multiplier $g$. The function is defined as follows:

$$\text{GravDiff}(p, \psi, g) = g \cdot \text{sgn}(p) \cdot \text{abs}(\tanh(2(p - 0.5)))^{\psi} \tag{3}$$

It is a distortion of the hyperbolic tangent that flattens the central range and comparatively exaggerates the asymptote on either side. The equation above can be described in the following steps:

1. Transform the network's prediction $p$, so its range is 1 to -1 rather than 0 to 1.
2. Apply the sigmoidal hyperbolic tangent, producing a value of 0 in the center and asymptotic slopes to -1 and 1 at the edges.
3. Apply a relatively large exponent which is $\psi$ (in the range of 3 to 11) to squash values close to 0 closer to 0. This ensures a shallow slope in the center (so low-confidence network outputs near 0.5 will produce very small output gradients).

(Note that step 3 only works if $\psi$ is an odd integer (because odd integer exponents preserve the sign), limiting the ability to fine-tune this function by changing $\psi$. To resolve this, a piecewise exponential function is used instead, where the absolute value is taken prior to applying the power and the sign is multiplied back in afterwards. This permits use of the full range of curves produced by even and non-integer values of $\psi$.)

### 5.2.3 Application

This function is applied in a training algorithm that is parameterized by the set of imperfectly labeled data $\varsigma$, the set of unlabeled training data $\upsilon$, a binary classification neural network $NN(x)$ (where $x$ is the input data format), and the gravitational multiplier increment $\delta_g$. The distortion power $\psi$ is assumed to be constant throughout a training run, and the gravitational multiplier $g$ is initialized at 0, gradually increasing during training. It iterates as follows:

1. Train $NN(x)$ for a single epoch on the combined set $\varsigma \cup \upsilon$, using the predefined ground truths for $\varsigma$ (with a mean squared error loss function) and the most recently calculated gravitational gradients for each of the examples in $\upsilon$.
2. Calculate predictions $p$ with $NN(x)$ on the entirety of $\upsilon$. Calculate $\text{GravDiff}(p, \psi, g)$ and record the resulting gradients for the next iteration.
3. Increment $g$ by $\delta_g$. At the beginning, when $g = 0$, training will be entirely based on the labeled set $\varsigma$, and progressively, over the course of a run, the gravitational effect increases with $g$.

This system was trained using a multi-layer perceptron with an 8-band Fourier transform input for $NN(x)$, because this was one of the most successful configurations in supervised learning. It produced the highest accuracy as well as the lowest class-wise standard deviation.

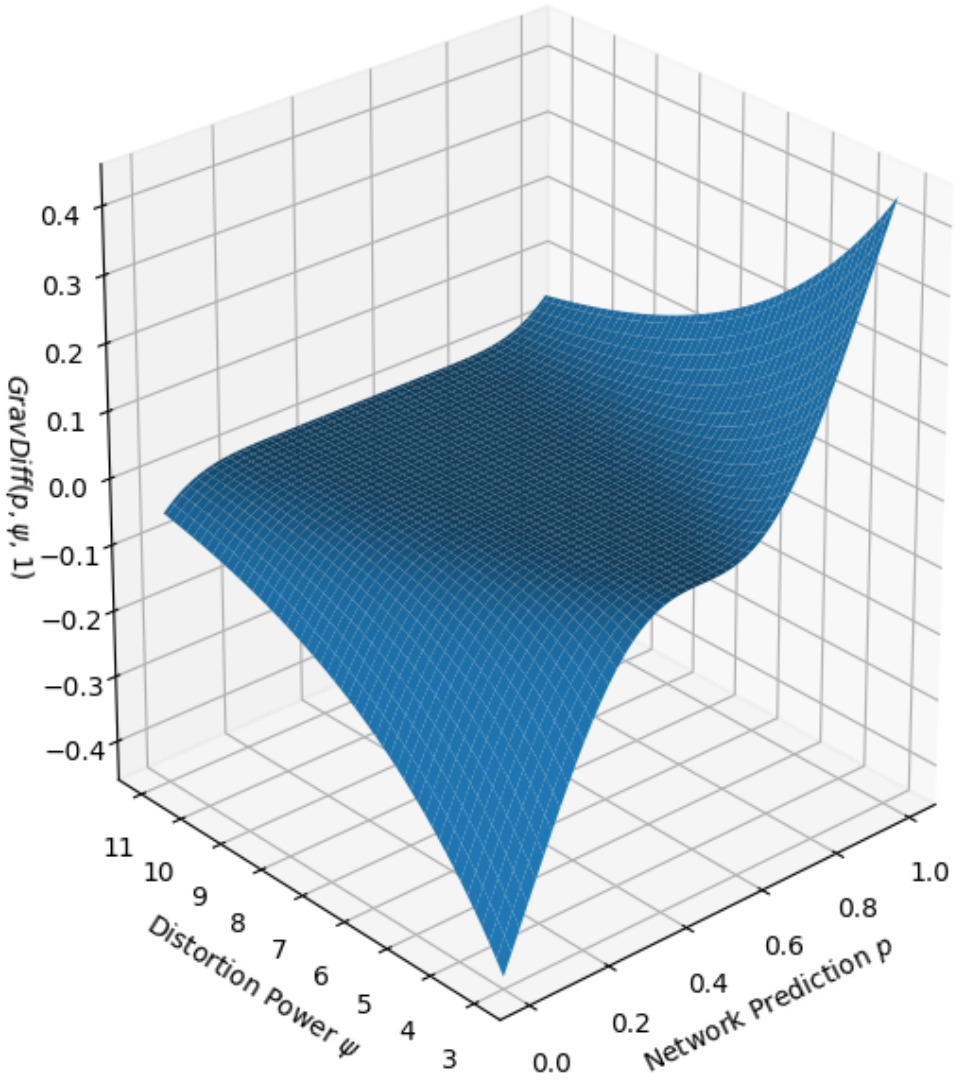


Figure 8: A visualization of $\text{GravDiff}(p, \psi, g)$ with respect to $p$ and $\psi$.

#### 5.2.4 Results

During a grid search of parameters of the gravitational differentiation algorithm, in addition to the stochastic gradient descent learning rate (a multiplier that defines how quickly and precisely optimization occurs), the highest mean accuracy over three runs of an individual configuration was 99%. The highest accuracy was obtained with 256 labeled examples in the set $\varsigma$ (out of 496 total training examples), which equates to 52% of the data being labeled. High accuracy statistics in this range were also obtained using 128 examples (99% accuracy, once again) and 64 examples (97% accuracy). 32 examples produced a much lower maximum accuracy, at only 83%. From this, it is evident that it is possible to achieve very high accuracy using this technique. It is also clear that, in general, larger initial sets produce higher performance.

However, the four hyperparameters optimized during the grid search were observed to have a weak polynomial correlation with the network's accuracy. This implies that the training process is generally insensitive to variations in those particular network parameters.

This lack of correlation also means that the mean accuracy over all configurations with a given $\varsigma$ size is a

meaningful statistic, since the validation accuracy for an individual configuration is difficult to confidently predict. These statistics, shown in Table 5, show that for all four set sizes, there is a fairly wide spread between maximum and mean accuracy.

Table 5: Gravitational differentiation accuracy statistics by size of $\varsigma$.

| Initial $\varsigma$ Size | Proportion of Data Labeled | Max Accuracy | Mean Accuracy |
|---|---|---|---|
| 32 | 6% | 83% | 69% |
| 64 | 13% | 97% | 73% |
| 128 | 26% | 99% | 84% |
| 256 | 52% | 99% | 94% |

Table 6 represents the performance of several high-accuracy gravitational differentiation configurations tested during the grid search. The accuracy statistic is the mean over three tests of the same configuration.

Table 6: Performance of gravitational differentiation configurations.

| $\varsigma$ Size | $\delta g$ | Learning Rate | $\psi$ | Mean AP Accuracy | Precision | Recall | CWSD |
|---|---|---|---|---|---|---|---|
| 32 | 0.0005 | 0.001 | 5 | 73% | 72% | 90% | 0.78 |
| 32 | 0.0005 | 0.003 | 9 | 83% | 80% | 99% | 0.63 |
| 64 | 0.0005 | 0.003 | 9 | 77% | 74% | 97% | 0.71 |
| 64 | 0.003 | 0.03 | 7 | 97% | 96% | 100% | 0.18 |
| 128 | 0.008 | 0.001 | 9 | 74% | 71% | 95% | 0.70 |
| 128 | 0.0005 | 0.03 | 11 | 99% | 99% | 99% | 0.17 |
| 256 | 0.005 | 0.01 | 11 | 97% | 96% | 99% | 0.25 |
| 256 | 0.001 | 0.03 | 3 | 99% | 99% | 100% | 0.10 |

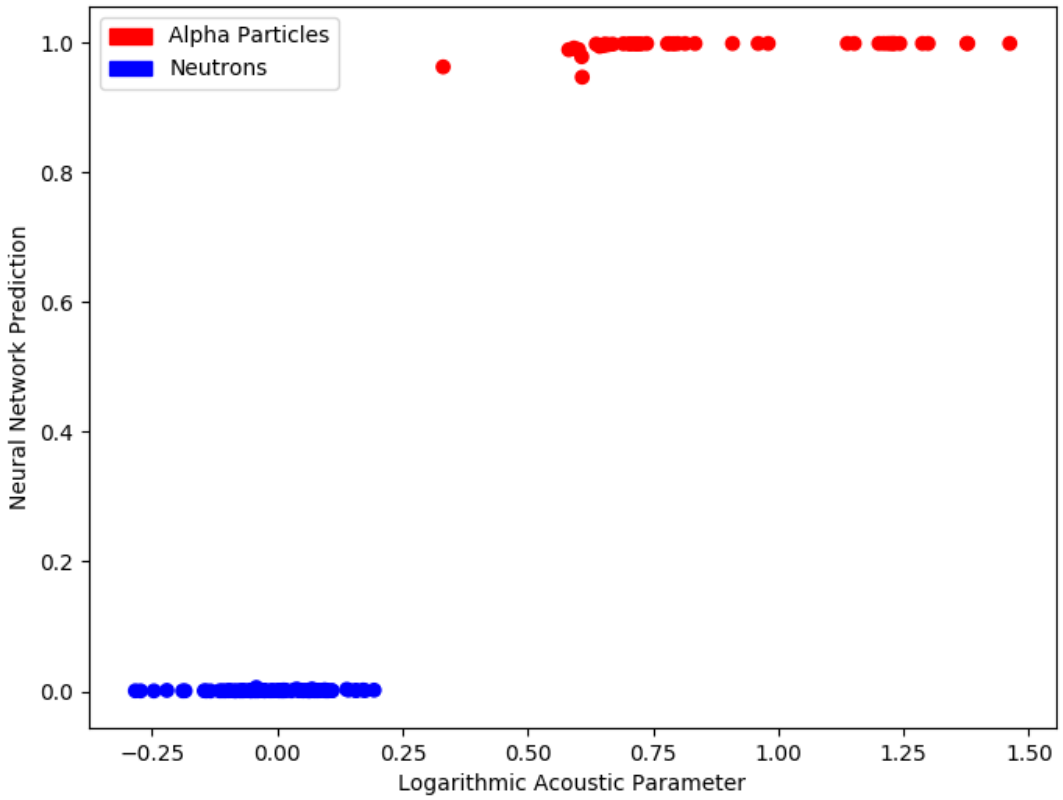


Figure 9: Predictions of best gravitational differentiation model compared to AP (on randomly selected validation set).

Note that in Figure 9, AP is normalized using a similar method to the original PICO-60 study [Amo+17]. In this normalized spectrum, the decision boundary between neutrons and alpha particles is at 0.25.

## 5.3 Iterative Cluster Nucleation

### 5.3.1 Concept

Iterative cluster nucleation (ICN) is a second semi-supervised learning algorithm developed as part of this study. It takes advantage of some amount of imperfectly labeled data, using it to classify the rest of an unlabeled training set while optimizing to produce an effective discriminator.

The basic concept is similar to that of gravitational differentiation. Starting with a relatively small amount of labeled data from two classes (nuclear recoils and alpha particles, in this case), predictions are run on a set of unlabeled data, and those predictions are used to produce further training data. However, there are three key differences:

1. In iterative cluster nucleation, an unlabeled example is not included in the training process at all, until a very confident prediction is made on it. Once that occurs, it is added to the training set, expanding one of the two "clusters" of training data.

2. Once an example is added to the active training set, it is never removed. This ensures that the highly confident predictions made at the very beginning continue to influence the training process.
3. The unlabeled examples, once added to the active training set, are weighted just as strongly as examples from the original labeled data set. This ensures imperfect labels in the labeled data do not outweigh the confident and more likely correct predictions from later in the training process.

#### 5.3.2 Algorithm

The algorithm is initialized as follows:

1. Take a subset of the training data, referred to as the seed set $\varsigma$. Use this as the beginning of the training set.
2. Remove classifications from any other available training examples. These create the unlabeled set $\upsilon$.
3. Compile and randomly initialize the weights of a binary classification neural network $NN(x)$.

Parameterized by the initial seed threshold $j$ (on the order of 0.01) and the seed threshold multiplier $k$ (on the order of 1.05), the algorithm follows these iterative steps:

1. Train $NN(x)$ for 30 epochs on $\varsigma$, using the imperfect ground truth values available.
2. Using the partially trained weights of $NN(x)$, run inference on the entirety of $\upsilon$, producing a set of predictions $p$.
3. Find predictions within $p$ that are within a distance of $j$ of either 0 or 1; as this is a binary classifier, such a prediction represents high confidence. Remove any such examples from $\upsilon$ and add them to $\varsigma$. The principle is that, when $NN(x)$ is trained on a relatively small set for short period of time, the few predictions within $p$ that are very confident are highly likely to be correct.
4. If no examples have been removed from $\upsilon$ and added to $\varsigma$, multiply $j$ by $k$ in place. This is done to increase the acceptance rate later in the training process, when most easily classifiable examples have been added to $\varsigma$. Otherwise, gridlock would occur, where certain examples could not be confidently classified given the training set.

#### 5.3.3 Results

During a grid search of these parameters (in addition to regularization parameters of the network), the best mean accuracy over three runs of an individual configuration was 99%, using 256 initial labeled examples in $\varsigma$. (The network's predictions on the best performing epoch can be observed in Figure 10.) With 128 examples, the maximum was 98%; with 64 it was 97%; and with 32, 95%. As with gravitational differentiation, larger set sizes improve accuracy significantly.

Once again, the parameters inherent to the ICN algorithm, as well as dropout and L2 regularization, do not have a meaningful polynomial correlation with accuracy. Note that there are several hyperparameters not specifically optimized in the context of iterative cluster nucleation, including the number of layers and numbers of neurons within those layers. The correlation between those hyperparameters and accuracy is subject to future research.

The mean accuracy for a given initial $\varsigma$ is again important. In Table 7, it is clear that with small initial $\varsigma$ sets, the training process is very unreliable, much like gravitational differentiation. However, the mean accuracy is much closer to the maximum for initial $\varsigma$ sets of 128 and 256, indicating that the training process of ICN is significantly more consistent than gravitational differentiation.

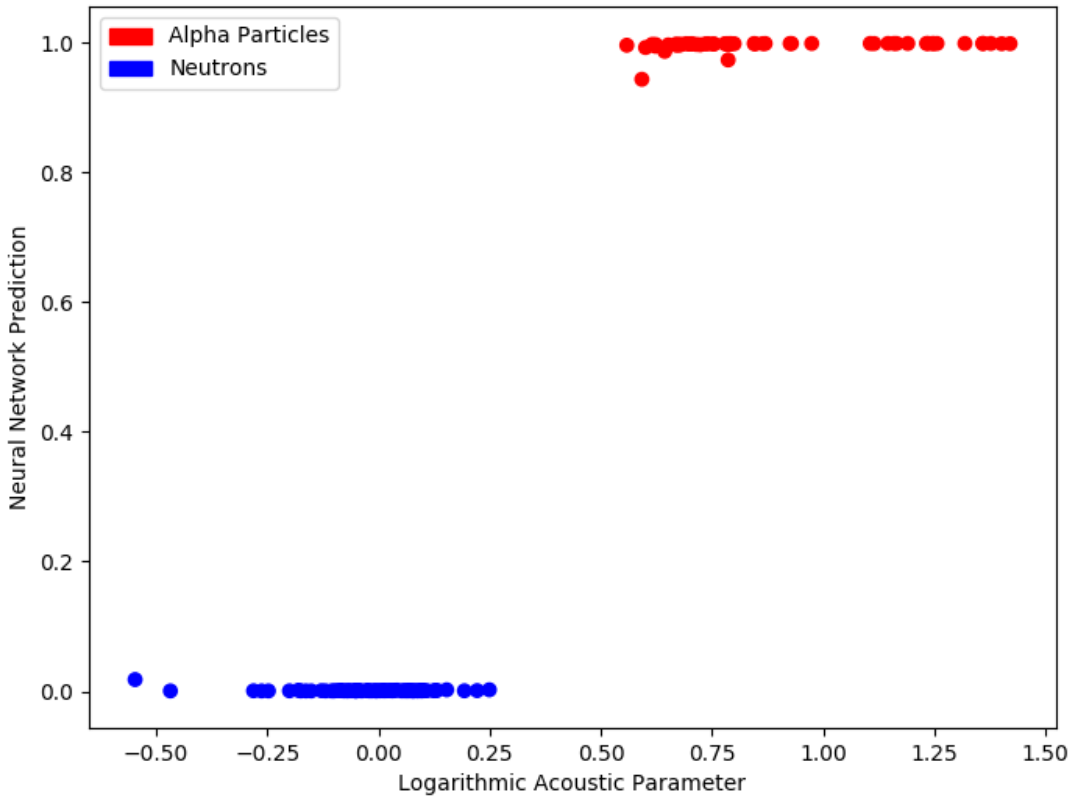


Figure 10: Predictions of best iterative cluster nucleation model compared to AP (on randomly selected validation set).

Table 7: Iterative cluster nucleation accuracy statistics by initial size of $\varsigma$.

| Initial $\varsigma$ Size | Proportion of Data Labeled | Max Accuracy | Mean Accuracy |
|---|---|---|---|
| 32 | 6% | 95% | 75% |
| 64 | 13% | 97% | 84% |
| 128 | 26% | 98% | 95% |
| 256 | 52% | 99% | 97% |

Table 8 represents the performance of a variety of high-accuracy iterative cluster nucleation configurations tested during the grid search. As with gravitational differentiation, the accuracy statistic is the mean over three tests of the same configuration.

Table 8: Performance of iterative cluster nucleation configurations.

| Initial $\varsigma$ Size | L2 $\lambda$ | Dropout | Initial $j$ | $k$ | Mean AP Accuracy | Precision | Recall | CWSD |
|---|---|---|---|---|---|---|---|---|
| 32 | 0 | 0.25 | 0.01 | 1.025 | 80% | 79% | 99% | 0.60 |
| 32 | 0 | 0 | 0.01 | 1.025 | 95% | 94% | 99% | 0.38 |
| 64 | 0.001 | 0 | 0.01 | 1.05 | 97% | 96% | 99% | 0.28 |
| 64 | 0 | 0.25 | 0.02 | 1.025 | 97% | 95% | 100% | 0.27 |
| 128 | 0 | 0.5 | 0.01 | 1.05 | 92% | 90% | 100% | 0.35 |
| 128 | 0 | 0.5 | 0.02 | 1.05 | 98% | 98% | 100% | 0.21 |
| 256 | 0 | 0.5 | 0.01 | 1.05 | 99% | 99% | 100% | 0.15 |
| 256 | 0.003 | 0.5 | 0.01 | 1.05 | 98% | 98% | 98% | 0.26 |

## 5.4 Performance Summary

Table 9 directly compares performance of the most successful semi-supervised learning configurations, with AP provided as a baseline for comparison. Remember that "Mean AP Accuracy" is calculated relative to AP's predictions, rather than replication of the ground truths.

Table 9: Summary of the most successful semi-supervised learning models.

| Initial $\varsigma$ Size | Technique | Mean AP Accuracy | Mean Precision | Mean Recall | Mean CWSD |
|---|---|---|---|---|---|
| 32 | GD | 83% | 80% | 99% | 0.63 |
| 32 | ICN | 95% | 94% | 99% | 0.38 |
| 64 | GD | 97% | 96% | 100% | 0.18 |
| 64 | ICN | 97% | 96% | 99% | 0.28 |
| 128 | GD | 99% | 99% | 99% | 0.17 |
| 128 | ICN | 98% | 98% | 100% | 0.21 |
| 256 | GD | 99% | 99% | 100% | 0.10 |
| 256 | ICN | 99% | 99% | 100% | 0.15 |
| 278 | PICO-60 ML | 80% | 76% | 100% | 0.64 |
| N/A | AP | 100% | 100% | 100% | 0.41 |

As observed in this table, semi-supervised learning algorithms are significantly more accurate at replicating AP than the original PICO-60 machine learning analysis. Compared to AP, in addition to achieving similar accuracy, configurations from both semi-supervised learning algorithms produce significantly more decisive predictions as measured by the class-wise standard deviation.

Table 10 compares the overall average performance of each of the semi-supervised learning configurations for different labeled set sizes.

Table 10: Summary of the average accuracy values for each semi-supervised learning technique.

| Initial $\varsigma$ Size | Technique | Mean Accuracy |
|---|---|---|
| 32 | GD | 69% |
| 64 | GD | 73% |
| 128 | GD | 84% |
| 256 | GD | 94% |
| 32 | ICN | 75% |
| 64 | ICN | 84% |
| 128 | ICN | 95% |
| 256 | ICN | 97% |

# 6 Conclusions

This study has demonstrated that it is possible, using semi-supervised learning, to automatically optimize a discriminator with performance comparable to one developed and tuned by humans like the Acoustic Parameter technique. This conclusion has been achieved through two rounds of optimization, the first to explore and evaluate various network structures and input formats using supervised learning, and the second to incorporate the most successful configuration into semi-supervised learning systems with the goal of producing a high-accuracy discriminator to complement AP.

In the realm of supervised learning, it was made clear that the banded Fourier transform is the most effective input format. It produced significantly higher accuracy and tighter prediction distributions than neural networks trained on either the raw audio waveform or image window data. There is strong evidence that image data in particular does not contain sufficient information to be used for discrimination.

Two semi-supervised learning algorithms were developed, applying the best configuration found for supervised learning. While both techniques produced some very effective discriminators, iterative cluster nucleation was found to be the more effective of the two. As seen in Table 10, it replicated AP with a mean of 97% accuracy, using 256 labeled training examples. This indicates that the technique shows significant promise.

As discussed in Section 1, discriminators such as AP are difficult to develop until a large quantity of calibration data has been collected and the physical principles of the experiment are understood well. The innovations developed in this study suggest that semi-supervised learning can efficiently replicate AP, thus facilitating very quick development of discriminators immediately after the first calibration runs are conducted on a new experimental apparatus. This has the potential to allow the teams working on these experiments to iterate more quickly, and to better understand intermediate results.

# 7 Technologies

All programming for this study was done in Python 3. Keras [Cho+15], running on a TensorFlow [Aba+15] backend, was used for all machine learning tasks. NumPy [Oli06] and SciPy [JOP+01] were used for linear algebra and signal processing. ROOT [BR96], scikit-image [Wal+14], and scikit-learn [Ped+11] were used for data loading and storage. Matplotlib [Hun07] was used for data visualization.

Implementation code for all algorithms discussed in this paper can be found at `https://github.com/brendon-ai/dark-matter`.

# 8 Acknowledgements

The PICO collaboration wishes to thank SNOLAB and its staff for support through underground space, logistical and technical services. SNOLAB operations are supported by the Canada Foundation for Innovation and the Province of Ontario Ministry of Research and Innovation, with underground access provided by Vale at the Creighton mine site. We wish to acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Canada Foundation for Innovation (CFI) for funding. We acknowledge the support from National Science Foundation (NSF) (Grant 0919526, 1506337, 1242637 and 1205987). We acknowledge that this work is supported by the U.S. Department of Energy (DOE) Office of Science, Office of High Energy Physics (under award DE-SC-0012161), by the DOE Office of Science Graduate Student Research (SCGSR) award, by DGAPA-UNAM (PAPIIT No. IA100316 and IA100118) and Consejo Nacional de Ciencia y Tecnología (CONACyT, México, Grant No. 252167), by the Department of Atomic Energy (DAE), the Government of India, under the Center of AstroParticle Physics II project (,CAPP-II) at SAHA Institute of Nuclear Physics (SINP), European Regional Development Fund-Project "Engineering applications of microworld physics" (No. CZ.02.1.01/0.0/0.0/16_019/0000766), and the Spanish

Ministerio de Economía y Competitividad, Consolider MultiDark (Grant CSD2009-00064). This work is partially supported by the Kavli Institute for Cosmological Physics at the University of Chicago through NSF grant 1125897, and an endowment from the Kavli Foundation and its founder Fred Kavli. We also wish to acknowledge the support from Fermi National Accelerator Laboratory under Contract No. DE-AC02-07CH11359, and Pacific Northwest National Laboratory, which is operated by Battelle for the U.S. Department of Energy under Contract No. DE-AC05-76RL01830. We also thank Compute Canada (`www.computecanada.ca`) and the Centre for Advanced Computing, ACENET, Calcul Québec, Compute Ontario and WestGrid for the computational support.

# References

[KW52] J. Kiefer and J. Wolfowitz. "Stochastic Estimation of the Maximum of a Regression Function". In: *Ann. Math. Statist.* 23.3 (Sept. 1952), pp. 462–466. DOI: 10.1214/aoms/1177729392. URL: https://doi.org/10.1214/aoms/1177729392.

[HPS53] J. M. Hollander, I. Perlman, and G. T. Seaborg. "Table of Isotopes". In: *Rev. Mod. Phys.* 25 (2 1953), pp. 469–651. DOI: 10.1103/RevModPhys.25.469. URL: https://link.aps.org/doi/10.1103/RevModPhys.25.469.

[BR96] Rene Brun and Fons Rademakers. "ROOT - An Object Oriented Data Analysis Framework". In: *AIHENP'96 Workshop, Lausanne*. Vol. 389. 1996, pp. 81–86.

[JKG96] Gerard Jungman, Marc Kamionkowski, and Kim Griest. "Supersymmetric dark matter". In: *Phys. Rept.* 267 (1996), pp. 195–373. DOI: 10.1016/0370-1573(95)00058-5. arXiv: hep-ph/9506380 [hep-ph].

[JOP+01] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. 2001–. URL: http://www.scipy.org/.

[Oli06] Travis Oliphant. *A guide to NumPy*. 2006.

[Hun07] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing In Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[Cir+11] Dan C. Ciresan et al. "High-Performance Neural Networks for Visual Object Classification". In: *CoRR* abs/1102.0183 (2011). arXiv: 1102.0183. URL: http://arxiv.org/abs/1102.0183.

[Ped+11] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[KB14] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014). arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980.

[Wal+14] Stéfan van der Walt et al. "scikit-image: image processing in Python". In: *PeerJ* 2 (June 2014), e453. ISSN: 2167-8359. DOI: 10.7717/peerj.453. URL: http://dx.doi.org/10.7717/peerj.453.

[Aba+15] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.

[Cho+15] François Chollet et al. *Keras*. https://keras.io. 2015.

[Boj+16] Mariusz Bojarski et al. "End to End Learning for Self-Driving Cars". In: *CoRR* abs/1604.07316 (2016). arXiv: 1604.07316. URL: http://arxiv.org/abs/1604.07316.

[Dai+16] Wei Dai et al. "Very Deep Convolutional Neural Networks for Raw Waveforms". In: *CoRR* abs/1610.00087 (2016). arXiv: 1610.00087. URL: http://arxiv.org/abs/1610.00087.

[Amo17] C. Amole. PhD thesis. Queen's University, 2017.

[Amo+17] C. Amole et al. "Dark Matter Search Results from the PICO−60 $C_3F_8$ Bubble Chamber". In: *Phys. Rev. Lett.* 118 (25 2017), p. 251301. DOI: 10.1103/PhysRevLett.118.251301. URL: https://link.aps.org/doi/10.1103/PhysRevLett.118.251301.

[Inc17] The MathWorks Inc. *MATLAB Neural Network Toolbox*. Natick, Massachusetts, 2017.

[Raj+17] Pranav Rajpurkar et al. "Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks". In: *CoRR* abs/1707.01836 (2017). arXiv: 1707.01836. URL: http://arxiv.org/abs/1707.01836.