Abstract

In the XENON1T experiment, both scintillation and ionization signals are measured directly and indirectly by PMTs through the light they produce. Classification of peaks in aggregated PMT signals as S1s or S2s is done through low-dimensional cuts; however, small S2s, on the order of a single electron, can easily fall under size cuts and be counted as S1s. To create a better classifier in this region of confusion, Monte Carlo data were created and used to train a machine learning based classifier. Additionally, there is currently no way to determine the depth of an S2 without picking an S1 and calculating drift time. This can present an issue when an S2 is detected without an S1 to pair it with. Since electron dispersion occurs due to drifting through the detector, it is expected that S2 depth can be reconstructed from the waveform shape, for use in the rejection of events outside of the fiduciary region of the TPC. To accomplish this, a convolutional recurrent neural network has been trained on simulated data.

Contents

1 Introduction .................................................. 2
   1.1 Dark Matter .............................................. 2
   1.2 XENON .................................................. 2

2 S1-S2 Classification ....................................... 4
   2.1 Scikit-Learn and Random Forest Classifiers ............ 4
   2.2 Adaptive Boosting ...................................... 4
   2.3 Data Simulation ........................................ 5
   2.4 Feature Selection ...................................... 5
   2.5 Training ............................................... 6
   2.6 Results .............................................. 6
   2.7 Summary ............................................ 7

3 S2 Position Classification ................................. 7
   3.1 Keras ................................................ 7
   3.2 Convolutional Neural Networks ........................ 11
   3.3 Recurrent Neural Networks ............................ 11
   3.4 Constructing a Neural Network ....................... 11
   3.5 Training ........................................... 11
   3.6 Results ............................................ 13
   3.7 Summary ............................................ 13

4 Acknowledgments ........................................... 15
1 Introduction

1.1 Dark Matter

Dark matter is a postulated type of matter that does not interact via the electromagnetic or the strong nuclear forces, instead interacting only through the gravitational and weak forces. Despite these strange properties, many astronomical observations support the existence of some kind of matter beyond the “regular matter” that we can observe with a telescope. One popular theory is that dark matter should take the form of weakly interacting massive particles, or WIMPs. These WIMPs would have a mass of roughly 100 GeV/c$^2$ and may be predicted stable supersymmetric partners of some of the traditional elementary particles of the standard model. The existence of dark matter was first inferred by Fritz Zwicky in 1933, after a study on the Como cluster revealed results inconsistent with its observed mass. Using his observed galactic velocities, Zwicky calculated a mass hundreds of times larger than could be explained by the visible matter contained within it. Surmising that matter was there, yet somehow unobservable with a telescope, he called this extra mass “dark matter.”

More evidence for the existence of dark matter accumulated during the 1970’s, following the work of Vera Rubin and Kent Ford. Rubin and Ford measured the rate of rotation about galactic centers at varying radii. What they found was that the rotation speed of objects within a galaxy does not obey the $\frac{1}{r}$ relation predicted from the distribution of its luminous mass, which forms a dense nucleus at the center. Instead, the rotation rate stays almost constant moving outwards from the galactic center. This would require about half of the mass of a galaxy to be contained in its dark outer regions.

Of course, many scientists remained skeptical about the idea of dark matter. With the work of Rubin, Kent, and others, several alternative explanations arose for their results. One of the most common areas for these explanations shifts the blame for the deviation of observations from theory from our understanding of matter to our understanding of gravity, postulating that at large distances, or for very small forces, gravity might act differently than either Newton or Einstein predicted.

One of the best pieces of evidence for dark matter—and one of the hardest for proponents of modified gravity to explain—is the bullet cluster. More properly the site of two colliding galactic clusters, the bullet cluster exhibits behavior that is very difficult to attribute to effects other than that of dark matter. When galactic clusters collide, the great majority of the regular matter in each cluster will collide only rarely with regular matter from the other, but the effect is still important to the dynamics of the collision; however, dark matter particles have no way to collide at all, and will essentially pass straight through one another, only affected by the changing gravitational field. In figure 1, one can see the spread in the distributions of regular and dark matter. It is easy to see that much of the mass of the bullet cluster is completely separate from the regular matter, and in line with a collision slowing down regular matter while dark matter proceeds unimpeded.

1.2 XENON

The XENON1T experiment is a search for dark matter operated by the XENON collaboration and housed under at the Gran Sasso National Laboratory in Italy. Designed as a successor to the earlier efforts of XENON10 and XENON100, XENON1T has so far provided the lowest limit on the mass and cross section of dark matter particles to date, and attempts to detect potential dark matter particles directly, rather than discover it by other means. The XENON1T experiment is a two phase Time Projection Chamber (TPC) (figure 2).
consisting of 3.2 t of cryogenic liquid xenon, (LXe), of which 2.0 t are in the TPC at a time, with a thin region of gaseous xenon at the top. The top and bottom of the TPC are covered in photomultiplier tubes (PMTs) which are used to observe signals within the detector. The inside wall of the TPC is made of highly reflective PTFE panels. When a particle enters the detector, it will have a chance of interacting with the xenon atoms in the liquid, either recoiling off of the electron shell if it has charge, or recoiling off of the nucleus if it is neutral. As xenon is a scintillator, either interaction will cause both a scintillation and an ionization of the xenon atom. The scintillation light from this initial interaction is called the S1, for first scintillation. As the PMTs are observing these S1 photons, the ionization electrons begin to drift up to the top of the detector due to the electric field maintained throughout. Once they reach the top, they enter a region of much larger electric field, which plucks them from the liquid and rapidly accelerates them through the gaseous xenon.

This creates many scintillating collisions, and thus a shower of photons called the S2, for second scintillation. Since the electric field is very nearly perfectly vertical and the PMTs are distributed about the top and bottom of the detector, the (x,y) position of the event can be calculated from the S2, and the z position of the event can be obtained via the time delay between the S1 and the S2. This position reconstruction is important for fiducialization in the detector; while XENON1T contains 2.0 t of LXe, the 1.0 t at the center gets much lower background than the outer material, so the ability to reject events from outside of this region lowers background events significantly. The XENON collaboration developed a custom software package called PAX to analyze data from the detector, and part of what it does is to take the waveform coming from the PMTs, identify the peaks, classify them as either S1s, S2s, noise, or unknown, and calculate scalar properties of these peaks like width, area, height, rise time, and others.
2 S1-S2 Classification

An important part of the methodology of the XENON1T experiment is the ability to differentiate S1s from S2s. Any analysis of data taken from the detector depends on correct peak identification, as events should always consist of an S1 and one or more S2s. Currently the way that PAX categorizes peaks is through several simple cuts in the space of area, rise time, and width. This works very well in general, but in the case of a small S2, on the order of 1 ionization electron, these cuts can misclassify peaks. To work in this region, it makes sense to use more sophisticated methods to get better classification accuracy, like machine learning.

2.1 Scikit-Learn and Random Forest Classifiers

Machine learning is currently an extremely broad subject with applications to a wide variety of fields, and there are too many different tools and algorithms that fall under the term to list here. Scikit-Learn is a very popular resource that offers many advanced machine learning tools in a single Python library\[6\].

One of these tools is the random forest classifier, often cited as one of the best performing machine learning methods over all. A random forest classifier is an example of a supervised machine learning algorithm: it is created and fitted with a training data set, consisting of many rows of observations and several columns of features, paired row-wise with a target outcome for each observation, and attempts are made iteratively to minimize the “error” in its predictions, with the aim of making class predictions where the outcome is unknown. Specifically, a random forest consists of a set of decision trees. A decision tree starts at the “root” with the whole training data set unsorted, and will at each step find a split in the data set that maximizes a quantity called Information Gain (IG). In the most common case of a binary decision tree, IG takes the form

$$IG(D_p, f) = I(D_{tot}) - \frac{N_\rightarrow}{N_{tot}} I(D_\rightarrow) - \frac{N_\leftarrow}{N_{tot}} I(D_\leftarrow)$$

Where \( f \) is a feature of the data, \( D_{tot} \) is the whole dataset at a node, \( D_\rightarrow \) and \( D_\leftarrow \) are the two portions of \( D_{tot} \), \( N_{tot} \) is the number of observations of \( D_{tot} \), and \( N_\rightarrow \) and \( N_\leftarrow \) are defined analogously. The function \( I(D) \) is a measure of the “impurity” at a node: a common choice for \( I(D) \) is entropy, where

$$I(D) = -\sum_{i=1}^{c} p(i|t) \ln p(i|t)$$

where \( c \) is the number of classes, \( t \) is a node, and \( p(i|t) \) is the ratio of the number of members of class \( i \) at node \( t \) to the total number of data at node \( t \). Hence, entropy would be at a minimum if all of the nodes only contained members of a single class, and it would be at a maximum if all the nodes were completely mixed. This gives an algorithm that can produce a tree of arbitrary depth, decreasing entropy with each step by picking the feature that creates the split with the highest IG; yet, allowing a decision tree to get too deep causes a problem of overfitting. Overfitting occurs when a model is allowed to conform too closely to the data used to train it, to the detriment of its performance on new data.

Using a random forest instead of a single decision tree can greatly reduce the effects of overfitting, creating a much more robust model. To create a random forest, simply create an arbitrary number of decision trees, choosing a random subset of the whole training data with replacement to grow each tree, and choosing a random subset of the set of features without replacement to create the splits. After the forest is grown, predictions are made by a majority vote, that is, a single observation will get sorted by each tree, and the most popular prediction becomes the final output.

2.2 Adaptive Boosting

Adaboost, or adaptive boosting, is another method of using several copies of a model to create a more accurate version. The algorithm works by creating an arbitrary number of identical classifiers one by one,
starting with each observation weighted equally, and increasing the weight of any misclassified observations at each step.

Adaboost was originally designed to be used with relatively weak classifiers like decision trees, but it has been shown that an Adaboost classifier built from stronger classifiers like random forests will have notably better performance\[5\].

2.3 Data Simulation

Supervised machine learning is a powerful tool, but it requires a set of data with trustworthy labels to start, and it is easy to see the problem this can cause when trying to create a classifier. An answer to this issue is to simulate data with given labels. PAX contains a utility designed to simulate data based on instructions presented in a csv file specifying positions, numbers of scintillation photons and ionization electrons, time, and recoil type for each event.

I simulated data for each type separately. First, I generated small S2s, each occurring at the same time of 600,000 ns within their own simulation, randomly distributed throughout the active volume of the detector, and given a uniform distribution of ionization electrons between 1 and 5. Second, I simulated an equal number of S1 events, individually placed at 600,000 ns, randomly distributed throughout the active volume of the detector, and given a uniform distribution between 20 and 670 scintillation photons, to create an area spectrum with as much overlap with that of the S2s as possible, see figure 5. There is still a visible difference between the two spectra, as the S1 spectrum should be more uniform while the S2 spectrum logically has notable peaks near integer multiples of the average area value of a single electron, 21.3 photoelectrons\[1\].

2.4 Feature Selection

A known problem in machine learning is what’s called the “curse of dimensionality”\[6\]. In low dimensional data, which has few features, adding one more feature might improve performance, but it can be shown that as the dimensionality of the parameter space (number of features) increases, the performance of a classifier given the same number of training samples decreases exponentially after some maximum value due to increased overfitting and the strange geometry of high dimensional spaces.
I thus needed to prudently select features to train a classifier, since the amount of data I could work with is finite. PAX provides more features for an event than can be easily represented here, but based on prior work, I chose 5 features as predictors for the adaboost classifier: 90% width, rise time, area fraction top, 50% width, and height. Here, 90% width refers to the width at 10% of the peak height, with a similar definition for 50% width, often referred to as width, rise time refers to the time required to go from 10% height to 90% height, area fraction top is the ratio of hits measured by the top PMTs to the total number of hits, and height is measured in photoelectrons per bin.

2.5 Training

Once I simulated the data and selected the features, I could actually create the classifier. Scikit-Learn provides tools for creating and training many types of classifier, including random forests and adaboost classifiers, as well as mechanisms for testing model performance in a statistically significant way, by selecting random portions of the data as training and test sets, and using a technique known as cross-validation on the training data.

Additionally, Scikit-learn provides tools for hyper-parameter selections: grid search and random search, grid search will create as many models as there are permutations of the provided parameter options, such as learning rate for logistic regression, while random search will search the parameter space in a random fashion, and is typically much less resource intensive. While both adaboost and the random forests used in it are notable for their lack of many different important parameters, they both require a specific number of sub-classifiers to work with. But, since the parameter space for our classifier is two dimensional, consisting of just possible numbers of sub-classifiers for both the adaboost and the random forests, a I chose a grid search to ensure optimal performance. The optimal topology as determined by the grid search was an adaboost classifier using 23 random forests, each with 5 trees.

2.6 Results

Training this classifier on $\sim$27,000 simulated events gave a cross validation accuracy of 99.843% ± 0.059% with a test accuracy of 99.90% on $\sim$3000 simulated events. A confusion matrix of the model on the data is shown in figure 10 and figure 7 shows the simulated data plotted in the width/rise time space. In figure 6 we see the reported importances of each feature used by the classifier. In agreement with earlier work, rise time is one of the most important features; however, PAX currently has an issue in S1 identification [1]. Any peak with a rise time over 70 ns is considered an S2, including 4.47% of simulated S1s, despite the fact that the simulated S1 events had 0 ionization electrons. Figure 8 shows this cut applied to the simulated S1s, showing the effect it has on the performance of a classifier. Figure 11 shows a confusion matrix of PAX on
the simulated data used to train the adaboost classifier, including the S1s misidentified as S2s. Additionally, figure \[9\] the incorrect predictions of the classifier, which are spread evenly through the region. This points to good generalization of the classifier, as its high performance does not seem to have any areas of large variance.

### 2.7 Summary

The current PAX framework performs well on differentiating S1s from large S2s, but it has a problem with small S2s and with slow rising S1s. The high overlap in small S2s and S1s, in particular slow rising and large S1s, means that cuts in the space of rise time, width, and even area fraction top, which provides very good separation on its own, cannot perform well on all events. Since these cuts aren’t enough, machine learning was expected to outperform them. My work was to simulate small S2s and S1s, and design a machine learning based classifier to sort them. After a survey of different algorithms, I arrived at an adaboost classifier with random forests as base classifiers. This adaboost classifier was able to correctly identify more of the simulated S1s than PAX, which indicates that a study should be done retraining this classifier with real data as well, as if its performance remains the same on real data, it will improve PAX based data analysis overall.

### 3 S2 Position Classification

In the XENON1T experiment, there is currently no way to determine the depth of an S2 without comparing it to an S1 and multiplying the time delay by the drift rate of $1.335 \mu \text{m} \text{ns}^{-1}$. As the smallest S1s are not able to be detected by XENON1T, being able to determine the depth of an S2 on its own would lower the energy threshold of the detector. The scalar data reported by PAX for each observation do not have strong correlation with S2 depth; however, due to the effects of drifting through the LXe on the dispersion of ionization electrons, the one dimensional temporal sum waveform data is thought to contain depth information in some sense. Modern machine learning techniques exist for handling temporal data with complicated structure, such as convolutional neural networks (CNNs) and Long Short Term Memory neural networks (LSTMs). Both are used in conjunction here to perform different functions in analyzing data.

### 3.1 Keras

Keras is a powerful API for creating neural networks, written in Python and used on top of a low level backend like TensorFlow or Theano\[3\]. TensorFlow, created by Google as a utility to make high performance
Figure 8: Width/Rise time space with PAX cut

Figure 9: Difference between predicted and true classification
Figure 10: Confusion matrix of the adaboost classifier
Figure 11: Confusion matrix of the PAX peak classifier
computing easier to implement, is used here for its CUDA interface, which allows the computations of training a neural network to be completed much more quickly by distributing the work across the several thousand cores of an Nvidia GPU (Graphics Processing Unit). The work presented here was performed using an Nvidia GeForce GTX 1080 GPU with 8113 MB of DDR5 VRAM.

3.2 Convolutional Neural Networks

A CNN is a kind of neural network designed to learn features of data that depend on spatial relationships between the data points in 1 or more dimensions. For instance, the current state of the art in computer image recognition are neural networks composed of several convolutional layers. Much of the power of these CNNs lies in the repetition of this process, as each successive layer learns to recognize increasingly abstract structures of the training data. Since the waveform data of an S2 is 1 dimensional and complex, it is logical to apply the automatic feature learning power of CNNs to it with the hope that several convolutional layers can recognize important patterns that relate to the depth the ionization electrons drifted through. While CNNs are quite useful, there is still an issue that must be addressed: the features that it learns do not have a global order within the time series. This isn’t a problem that an image classifier would face, as there is no intrinsic order to the pixels in a picture, but there is certainly an intrinsic order to the intervals of an S2 waveform.

3.3 Recurrent Neural Networks

An RNN is a kind of neural network designed to learn temporal, or order based, features of one dimensional data. In addition to time series data, RNNs are used frequently with text data as in problems of machine translation. An important type of RNN is the LSTM, which is by design able to learn temporal features of arbitrary separation. A neuron in an LSTM layer contains a memory cell that can hold information from any time in a waveform for use at any later time. This ability is central to an effective study of these waveforms, which have 600 time steps and potentially have features that span the whole width.

3.4 Constructing a Neural Network

Keras makes it easy to generate many different kinds of neural network by combining different kinds of layers one after the other. Convolutional layers and LSTMs have been discussed above, but other simpler layers can be added to a network to improve generalization, like Dropout, which will randomly disconnect neurons from the network to avoid “accidental” activations, that is, it selects against associations that occur rarely, gaussian noise will insert randomness between two layers to reduce overfitting, and batch normalization between convolutional layers helps improve performance by normalizing the activation of the layer between batches. The table below shows the topology of the neural network used.

3.5 Training

Due to time constraints, the first problem addressed was one of classification. Classification is simpler to implement in a neural net, and can be used to ascertain the viability of creating an effective regressor in future work.

To prepare the data, the waveforms were scaled so that each had a maximum height of 1 and a minimum height of 0 to improve the ability of the neural network to recognize common features. Additionally, the z positions of the S2s were classified into two groups: those inside the fiduciary region of the detector and those outside of it. I chose to split up the data like this because rejecting background events in this way made a better test case than any other simple classification. I trained the neural net was trained on 297,345 events, validated on 33,038 events, and finally tested on 36,709 events. Additionally, due to the statistics of producing randomly distributed events, those inside the fiduciary region occur \( \sim 5 \) times more often than those outside; to account for this, outside waveforms were weighted 5 times more heavily when training the network.
<table>
<thead>
<tr>
<th>Layer type</th>
<th>Output Shape</th>
<th>Number of Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>1D Convolution</td>
<td>(None, 601, 100)</td>
<td>600</td>
</tr>
<tr>
<td>Dropout</td>
<td>(None, 601, 100)</td>
<td>0</td>
</tr>
<tr>
<td>Gaussian Noise</td>
<td>(None, 601, 100)</td>
<td>0</td>
</tr>
<tr>
<td>Batch Normalization</td>
<td>(None, 601, 100)</td>
<td>400</td>
</tr>
<tr>
<td>1D Convolution</td>
<td>(None, 601, 100)</td>
<td>50,100</td>
</tr>
<tr>
<td>Dropout</td>
<td>(None, 601, 100)</td>
<td>0</td>
</tr>
<tr>
<td>Gaussian Noise</td>
<td>(None, 601, 100)</td>
<td>0</td>
</tr>
<tr>
<td>Batch Normalization</td>
<td>(None, 601, 100)</td>
<td>400</td>
</tr>
<tr>
<td>1D Convolution</td>
<td>(None, 601, 100)</td>
<td>1,000,100</td>
</tr>
<tr>
<td>Dropout</td>
<td>(None, 601, 100)</td>
<td>0</td>
</tr>
<tr>
<td>Gaussian Noise</td>
<td>(None, 601, 100)</td>
<td>0</td>
</tr>
<tr>
<td>LSTM</td>
<td>(None, 601, 20)</td>
<td>9760</td>
</tr>
<tr>
<td>Dropout</td>
<td>(None, 601, 20)</td>
<td>0</td>
</tr>
<tr>
<td>Gaussian Noise</td>
<td>(None, 601, 20)</td>
<td>0</td>
</tr>
<tr>
<td>LSTM</td>
<td>(None, 601, 20)</td>
<td>3360</td>
</tr>
<tr>
<td>Dropout</td>
<td>(None, 601, 20)</td>
<td>0</td>
</tr>
<tr>
<td>Gaussian Noise</td>
<td>(None, 601, 20)</td>
<td>0</td>
</tr>
<tr>
<td>LSTM</td>
<td>(None, 20)</td>
<td>3360</td>
</tr>
<tr>
<td>Dropout</td>
<td>(None, 20)</td>
<td>0</td>
</tr>
<tr>
<td>Gaussian Noise</td>
<td>(None, 20)</td>
<td>0</td>
</tr>
<tr>
<td>Dense</td>
<td>(None, 2)</td>
<td>42</td>
</tr>
</tbody>
</table>

Table 1: Representation of the linear organization of the layers of the neural net used to classify S2s. Layer type is self explanatory, output shape is used by TensorFlow and is related to the number of features extracted from the arbitrary number of 601 time step long waveforms, and the number of parameters is related to the complexity of the network.
3.6 Results

In Figure 12, we see a confusion matrix of the predictions of the neural network on all of the simulated data. The neural network achieved an accuracy of 78.12% in correctly identifying the region of origin of each simulated event, but it should be noted that it greatly improved the purity of the events predicted as inside when compared to no classification. Originally, events outside the fiduciary region made up 16.30% of all events, but of the events predicted to be inside the fiduciary region, only 9.04% were from outside. Altogether, this indicates that further work should be able to produce a much more effective classifier, or even a true regressor, able to precisely predict the z position from only the S2 waveform.

3.7 Summary

At this time, PAX has no way of determining the depth of an S2 without comparing it to an S1, which presents issues as it raises the lowest possible interaction energy threshold of the detector, directly impacting the dark matter search. My work here was to use the sum waveform information of simulated S2s to create a convolutional neural network able to make predictions about the depth of an S2 without an associated S1. This neural network was unable to achieve the accuracy I desired, but it does succeed in reducing the ratio of background events to events within the fiduciary region of the detector. Further, I believe that future work in this area could create a neural network of much higher accuracy, or even a model able to regress the
depth to a high degree of certainty from the waveform alone.
4 Acknowledgments

I would like to thank the NSF and the REU program at Nevis Labs for providing an incredibly enriching experience. Thank you to the XENON collaboration and in particular Dr. Elena Aprile for the opportunity to work with the XENON group at Columbia. Special thanks to Joey Howlett, Fei Gao, and Tianyu Zhu for lending their expertise.

References


[7] NASA/CXC/M Weiss. English: Composite image showing the galaxy cluster 1E 0657-56, better known as bullet cluster. The image in background showing the visible spectrum of light stems from Magellan and Hubble Space Telescope images. The pink overlay shows the x-ray emission (recorded by Chandra Telescope) of the colliding clusters, the blue one represents the mass distribution of the clusters calculated from gravitational lens effects. Aug. 21, 2006. URL: https://commons.wikimedia.org/wiki/File:1e0657_scale.jpg#metadata (visited on 07/27/2018).