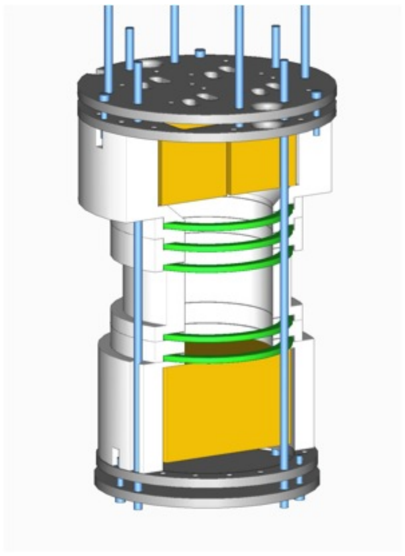
POSITION RECONSTRUCTION IN MINIATURE DETECTOR

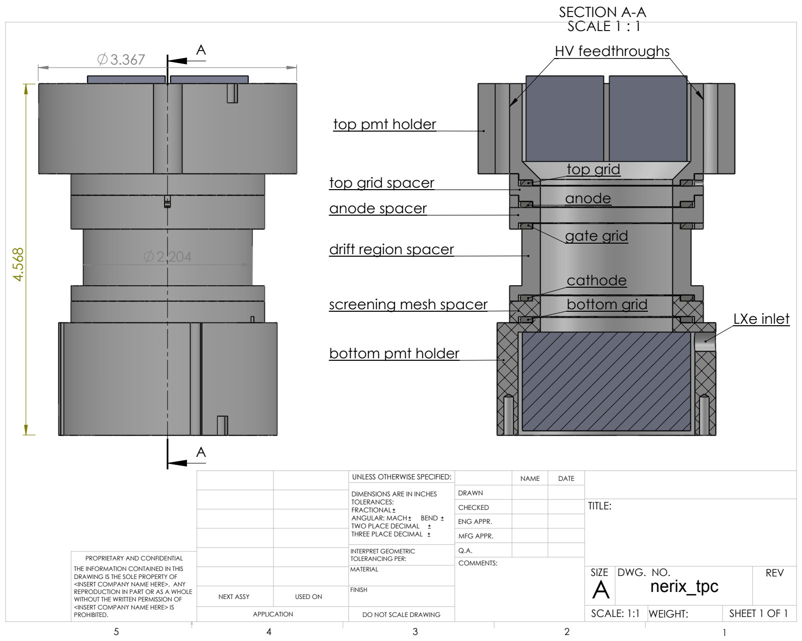
ADAM LEVINE



1. Introduction

3D position reconstruction of particle collisions in the liquid xenon Dark Matter detector enables further background reduction as well as better measurements on the mass and size of the fiduciary volume. This report discusses the development of a computer program that can translate a photo-multiplier tube (PMT) hit pattern into a point in the cross-sectional plane of the liquid xenon cylinder. Through Monte Carlo simulations of the S2 signal and the methods of Neural Networking, a program was trained to this task.

Date: August 29, 2012.



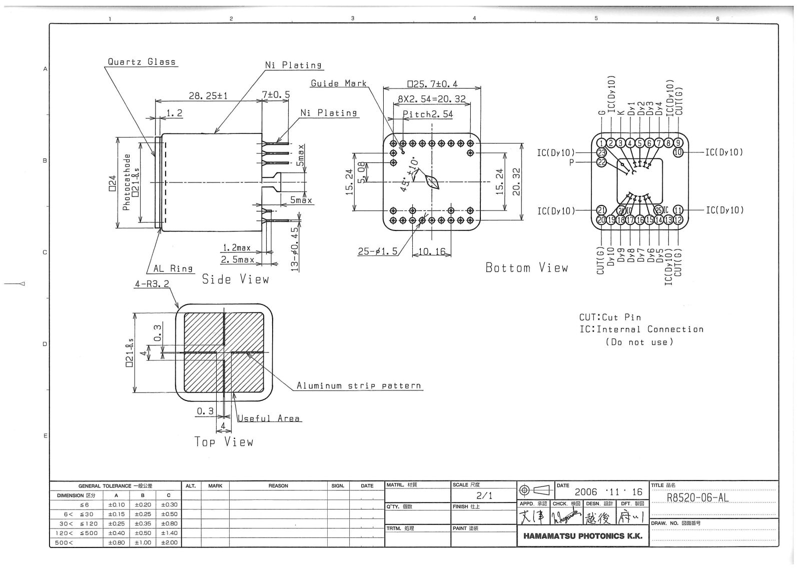
2. Monte-Carlo Simulations and Geant4

2.1. **Geometry.** The detector geometry was implemented using the C++ libraries of Geant4, a package that specializes in high-energy physics simulations. The core of this detector consists of five PMTs – four on top for S2 light collection and one on the bottom for S1 - - and the cylindrical time projection chamber (TPC) in between. Interspersed throughout the TPC are five grid meshes as can be seen above. Most of the light collected to reconstruct the horizontal position comes from the S2 signal, which occurs as ionization electrons are pulled from the liquid-gas interface in the TPC by a strong electric field. Thus, the TPC was simulated with dual phases, each a logical volume within Geant4, one for the liquid phase and one for the gaseous phase. The interface between these two phases sits in between the gate grid mesh and the anode mesh.

The TPC was constructed in pieces, adding the ability to take out a section of the TPC, bringing the PMTs closer to the GXe/LXe boundary. In this case each PMT will subtend a larger angle for the S2 photons and the array should be more sensitive to pattern changes and thus provide better position resolution.

2.1.1. *PMTs.* This detector contains 5 PMTs, only the top four of which play a role in lateral position reconstruction. These PMTs were simulated by placing four identical logical volumes filled with aluminum on top of the TPC in a grid pattern. Each individual PMT was then filled with a vacuum logical volume, giving the aluminum a thickness of 1 mm. The bottom of each PMT is shielded by a quartz window but the other sides are in direct contact with the gaseous xenon. Although there are only four full PMTs on top, each individual has four physically distinct anodes on which the amplified signal lands. Thus, effectively, there are 16 independent regions of sensitivity. These added divisions should lead to better position resolution.

All the simulated PMTs were assigned to a G4SensitiveDetector that stored data about interactions occurring within or on the boundary of the PMT physical volumes. This sensitive detector provided the logical structure that retained the PMT signal as well as other event information.



2.1.2. *Meshes.* One simulation difficulty came in correctly implementing the meshes. This detector has five meshes spread out within the TPC. Two - the top grid and the anode mesh - sit above the liquid level; the rest sit below it. These meshes were simulated by creating a very thin cylinder with diameter equal to that of the TPC. The thickness used was .0125 cm - the diameter of the mesh wire. The mesh logical volumes were then filled with their own characteristic metal with an adjustable absorption length. This length is in turn determined by the transparency of the meshes, which can be approximated by dividing the total area of a grid square by the hole area within that square. In the case of these meshes, the transparency ≈ .8. We know that the fraction of incident photons left unabsorbed by a medium is related to the depth into the medium by the Beer-Lambert law: $P(x) = e^{\frac{-x}{\lambda}}$ where λ is the characteristic attenuation length, P is the fraction of remaining photons and x is the distance into the medium. Then, setting $P(x) = .8$ and $x = .0125\,\text{cm}$, we get $\lambda \approx .112\,\text{cm}$.

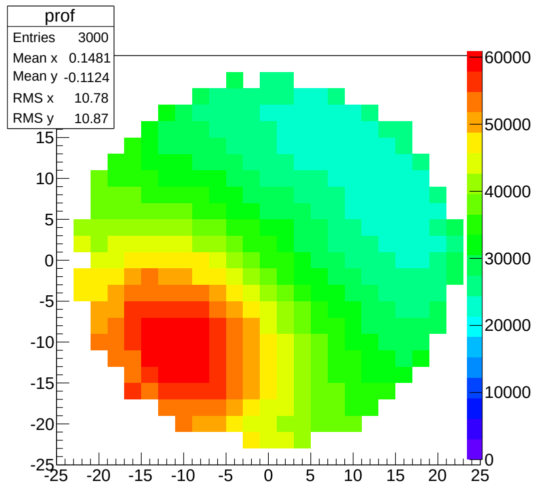
The mesh metal's absorption length is then set to this value in order to obtain the wanted effective transparency.

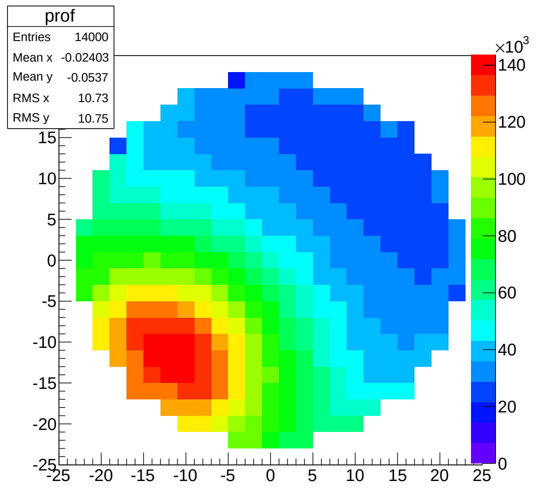
2.2. **Simulation.** One full simulation consists of running multiple events. One event consists of randomly generating S2 photons and storing the hit pattern in a vector, whose i^{th}

component corresponded to the number of photons that hit the i^{th} PMT. 3,000 events were simulated on a Columbia campus computer, and another 14,000 were simulated without the removable top mesh. The photon source is created at a randomly chosen point in the horizontal plane for each event.

2.2.1. *Photon Generation.* Besides minor corrections due to lateral drift as the electrons move from the interaction point along the TPC axis, the S2 photons are generated roughly where the original interaction occurred in the horizontal plane. These photons are simulated by producing scintillation light with a $1\,keV\ e^-$ in a plane just above the LXe surface.

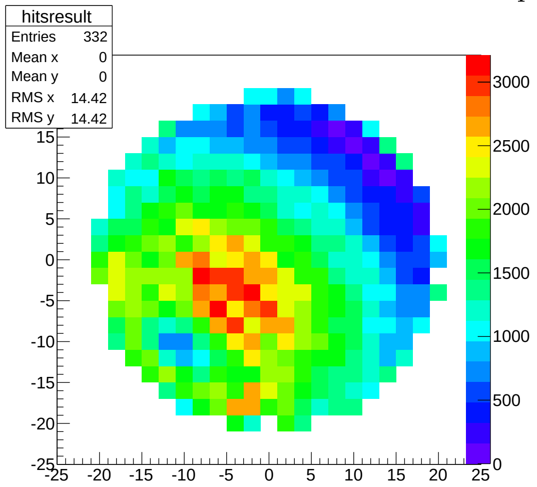
The e^- interaction length is extremely short so the scintillation point is essentially the generation point of the electron. The number of photons then depends directly upon the scintillation yield. In order to get better position reconstruction accuracy, a higher number of photons is needed. To get close to 1% uncertainty, each PMT needs N photons where $\frac{1}{\sqrt{N}} \approx .01$. So each PMT needs to average at least 10,000 photon hits. Thus, the simulations had a scintillation yield of $\approx 375{,}000\,photons/keV$. This means in each event the e^- interaction released 375,000 scintillation photons. This corresponds to an average photon hit number of $\approx 30,000$ per PMT.

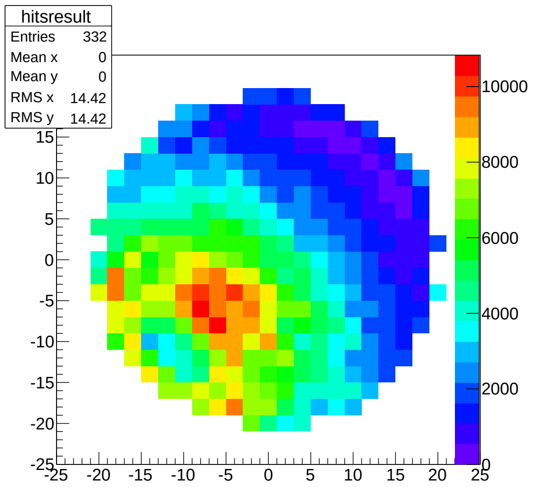




Left: Number of hits on PMT1 as function of source location in the horizontal plane. *Right*: Same but with raised liquid level. (PMT1 is in the lower left hand corner of the each graph.)

2.2.2. *Simulation with no top mesh.* Instead of creating a separate geometry to model the detector without a top mesh, the liquid xenon level was raised to between the anode mesh and the top mesh instead. This is valid because the extra mesh below the liquid level should not have much effect on the S2 photons hitting the PMTs. With the PMTs now closer to the point source, the maximum number of photons hits on each PMT increases by approximately twofold. Not only this but the pattern's sensitivity to changes in source position increases. This can be modeled by taking averaging each pixels difference with the surrounding pixels in the above graphs. The change in hit pattern is clearly much more pronounced when the PMTs are closer to the point source.

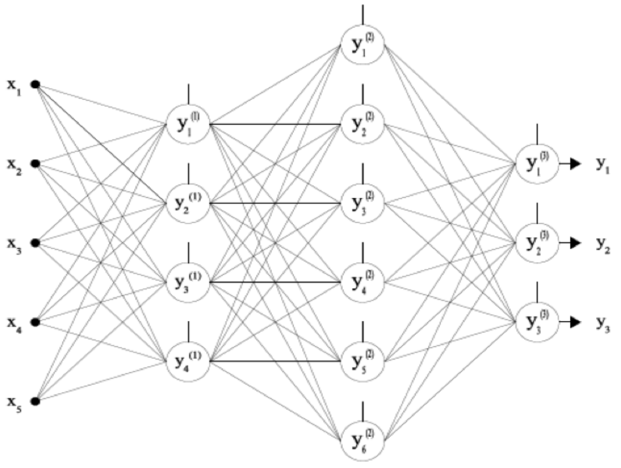




Left: Change in hit pattern on PMT1 with a small change in position for low liquid level. *Right*: Same but with raised liquid level.

3. Developing An Algorithm

The goal of these simulations was to eventually create a function $f : \mathbb{R}^N \to \mathbb{R}^2$ that could take as input a PMT signal and output a point corresponding to the position of the source in the horizontal plane. N here is the number of S2 PMTs. In order to do this, a *Neural Network* was created using the FLOOD3 C++ libraries that could then be trained by the data produced in the Geant4 simulations. Although there are different types of neural networks, the following only focuses on Multilayer Perceptrons (MLPs).



A simple multilayer perceptron with two hidden layers.

3.1. **The Theory of Neural Networks.** A simple MLP neural network consists of neurons connected by synaptic weights. There is an input layer (on the left in the diagram above) and an output layer (on the right). Every layer in between is called a *hidden layer*. Each neuron stores its own value. The synaptic weights are visualized above by the lines mapping each neuron to one in the next layer. A line is associated with a number - directly analogous to a coefficient in a matrix. The value of a non-input neuron is determined by the sum of the previous neurons times their weights. Using the notation in the diagram above then:

$$y_i^{(1)} = \sum_{j=1}^{n} W_{ij} x_j + b_j$$

where n is the number of input neurons (5 in the figure above) and b_j is called the *bias*. After each hidden layer neuron has been calculated from the previous layer, it gets sent through an *Activation Function*, which usually scales its input to somewhere in between -1 and 1 or 0 and 1. The function used here is the hyperbolic tangent function.

For the reconstruction algorithm only one hidden layer was used. Thus the output components can be mathematically represented as:

$$o_i = \sum_{j=1}^{n_{hidden}} W_{ij}^{(2)} tanh((\sum_{k=1}^{n_{PMT}} W_{jk}^{(1)} x_k) + b_j^{(1)}) + b_i^{(2)} \qquad i = 1, 2 \tag{1}$$

3.2. **Training and the Objective Function.** The output vector then is a function of not only the input vector, but also the bias and synaptic weight values. In order to get the perceptron to output “good” values - i.e. an output vector close to the real vector - a training structure is introduced. This combines an *objective function* - in this case a function that measures the average error in the output - with a *training algorithm.* The data from the simulation is fed into the MLP and the output for each event is stored in an array. This array is fed into the objective function and then the training algorithm adjusts the parameters as needed to reach a certain goal - namely, the minimization of the objective function.

3.2.1. *Objective Function.* In this case, the function that needs to be minimized is the average error associated with the MLP output data. This is pretty clearly just the average radial distance from the true source point to the reconstructed one in the horizontal plane. Thus the objective function $O : \mathbb{R}^2 \to \mathbb{R}$ should take the form:

$$O(\mathbf{o^{MLP}}; \mathbf{W^{(1)}}, \mathbf{W^{(2)}}, \mathbf{b^{(1)}}, \mathbf{b^{(2)}}) = \frac{1}{N_{events}} \sum_{i=1}^{N_{events}} \sqrt{(o_{i1}^{ideal} - o_{i1}^{MLP})^2 + (o_{i2}^{ideal} - o_{i2}^{MLP})^2}$$

Since $\mathbf{o^{ideal}}$ is fed in through the simulation data, these do not change throughout the training process and can just be considered constants. Also, since $\mathbf{o^{MLP}}$ is implicitly defined by (1), O is really a function $O : \mathbb{R}^{N_{parameters}} \to \mathbb{R}$ where $N_{parameters} = n_{PMT} \cdot n_{hidden} + 3 \cdot n_{hidden} + 2$. Thus, training the MLP has been reduced to minimizing this function using the training algorithm.

3.2.2. *Training Algorithm.* The training algorithm's job then is just to find the point $\mathbf{p}^*$ in parameter space such that $\nabla O(\mathbf{p}^*) = 0$ and $O(\mathbf{p}^*)$ is a global minimum. The basic idea is to loop until specific goals are reached, each time resetting the parameters. One loop is called an *epoch.*

There are many different algorithms to do this, some faster than others. The most basic is called "Gradient Descent." In this method, the algorithm takes steps in parameter space along the negative of the gradient vector. The algorithm is defined recursively as:

$$\mathbf{p}_{i+1} = -\nabla O(\mathbf{p}_i) \cdot \gamma_i + \mathbf{p}_i$$

where t_i is called the *training rate.* The training rate is adjusted so that the algorithm does not overstep a minimum and get into an oscillation around it, but large enough so that training takes a practical amount of time.

The method used here is an improved version of the Gradient Descent method called the Conjugate Gradient, which steps in parameter space according to $\mathbf{p}_{i+1} = \mathbf{p}_i + \mathbf{t}_i \cdot \gamma_i$. Here, however, instead of $-\nabla O(\mathbf{p}_i)$ we have $\mathbf{t}_i$. This stepping direction is then defined recursively itself by the equations

$$\mathbf{t}_{i+1} = \nabla O(\mathbf{p}_{i+1}) + \mathbf{t}_i \cdot \gamma_i$$

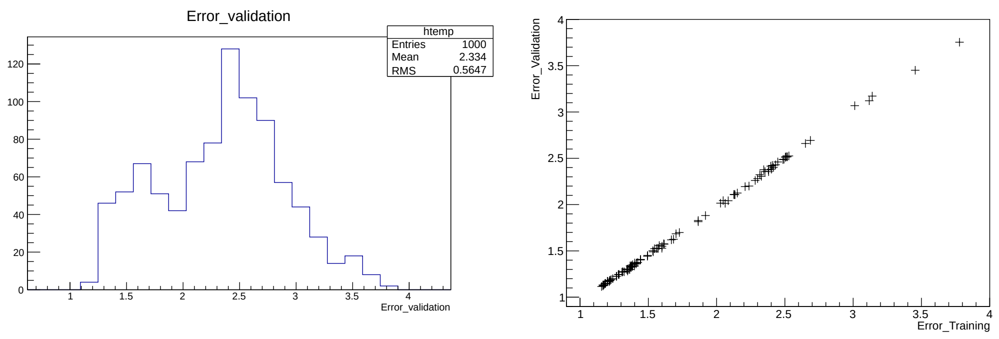
where $\mathbf{t}_0 = -\nabla O(\mathbf{p}_0)$. γ_i is the conjugate parameter that is the ratio between squared magnitudes of the $(i+1)$th and the ith gradient vectors.

3.2.3. *Initializing Parameters.* Within the data set of information gathered from the GEANT4 simulations, a division is made between data used to train the algorithm and data used to test it. FLOOD3 provides an object class called the "InputTargetDataSet" (ITDS). This matrix stores the PMT hit numbers and the primary position in each row. The number of columns then is $n_{PMT} + 2$. The InputTargetDataSet class has a member function named "split random indices" which takes as its argument the percentages of rows used for training and validation. For what follows, 10% of the rows were used for training and 90% were used for validation.

Every algorithm begins by initializing the parameters to some value. Unfortunately, since the objective function is often fairly complicated with many constants, there could be many local minima and the algorithm could get stuck on these points since they satisfy the criteria - namely that $\nabla O(\mathbf{p}^*) = 0$. So, if the MLP starts out with parameters near a local minimum they can easily fall in. Finding the global minimum then depends upon finding the right initial parameters.

Although other techniques have been presented, the technique used for this task was to train the perceptron 100 times. At the start of each run, the initial parameters are randomly initialized within a certain range. Out of these 100 runs, the one that ends at the lowest value for $O(\mathbf{p}^*)$ is singled out and the parameters are saved in the final perceptron. Obviously, as the number of runs increases, the chance that the algorithm finds the global minimum increases, but so does the run time.

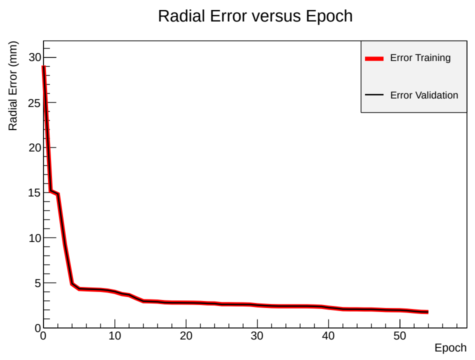
After running the training algorithm 1000 times, a histogram is made of the mean validation error.

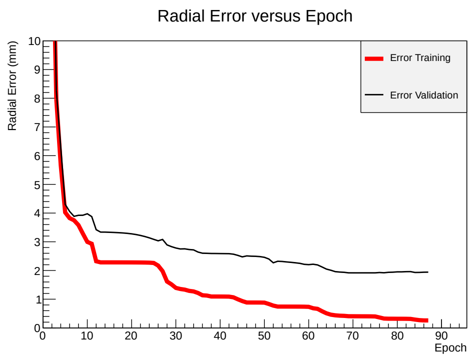


Left: Validation set's radial error after 1000 training runs. *Right*: The validation error versus the training error.

Clearly there is a spread of error, corresponding to different local minima in parameter space. We can also probe for issues by seeing if there is an odd correlation between the radial error for the validation set and the same quantity for the training set. Hopefully, they fit a line angled at 45 degrees. It would be bad, for example, if, as the training error goes down, the validation error goes up. Fortunately, we do not see this, and there is

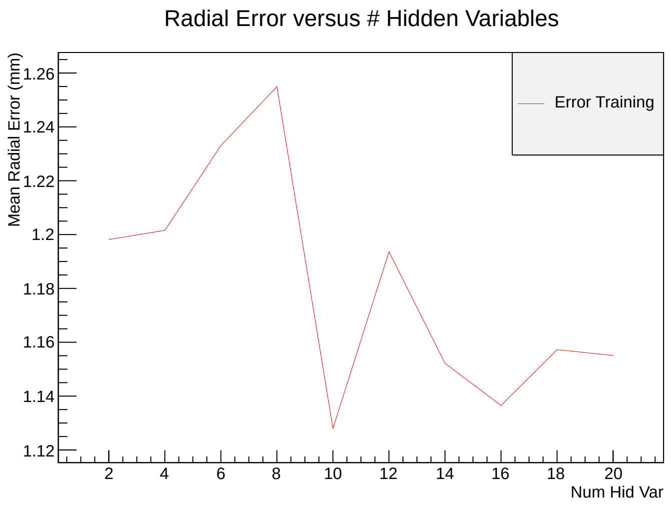
3.2.4. *Overtraining.* As the percent allocated for training decreases, the position accuracy in the validation instances increases as a result of overtraining. The regression fits the output function too tightly to the training points and as a result increases the average error between the function points and the validation points. One can see this in a graph of the training set's error and the validation set's error versus the training epoch. As the epoch increases, if the validation error starts to rise up from the training error then this is a hint that overtraining may be happening. This does not occur, however, for the ratios stated above.





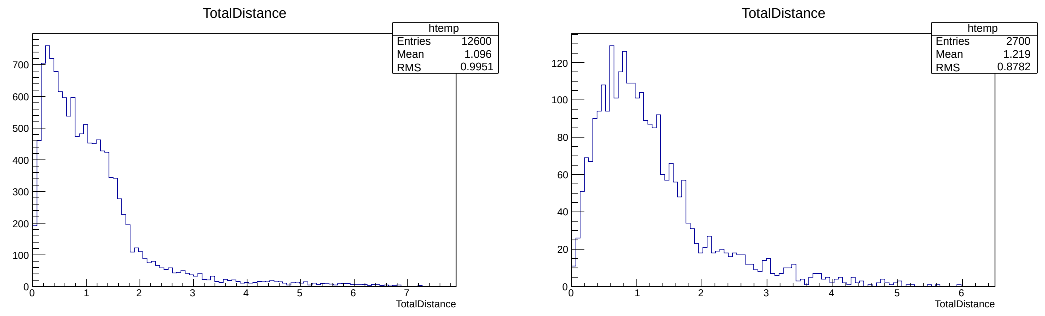
Left: Radial error versus epoch for 10% training. *Right*: Overtraining with .1% of the ITDS instances used for training.

3.2.5. *Hidden Neurons.* The reconstruction accuracy should depend upon the number of hidden layer neurons present. More neurons allows the perceptron to more easily span the space of potential functions. Too many neurons, however, could lead to overtraining. In the case of the present perceptron, the performance of the algorithm seems to be fairly independent of the number of hidden neurons as seen below. Noting the scale of the y-axis, it is clear that whatever benefit may come from additional neurons is on the order of 50 nm, which is negligible compared to the radial error RMS.



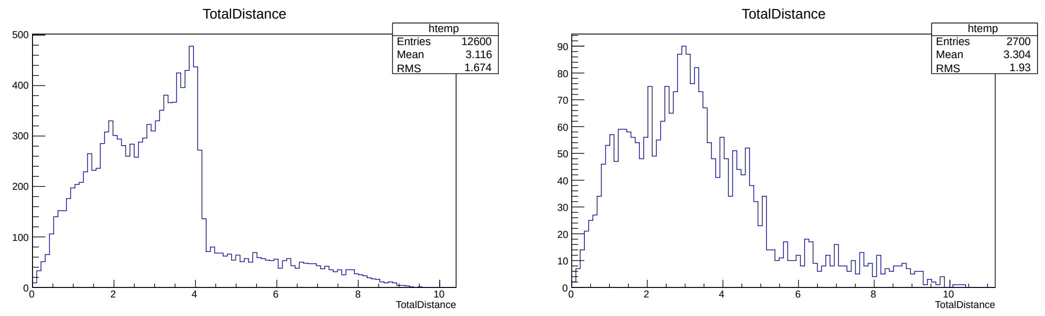
4. Training Results

There are two noteworthy points of flexibility in this detector: the ability to effectively raise the liquid xenon level by taking out the top mesh of the TPC and the ability to add sensitive regions to the PMTs. One can probe the effect of the former by graphing a histogram of the radial error with and without the top mesh.



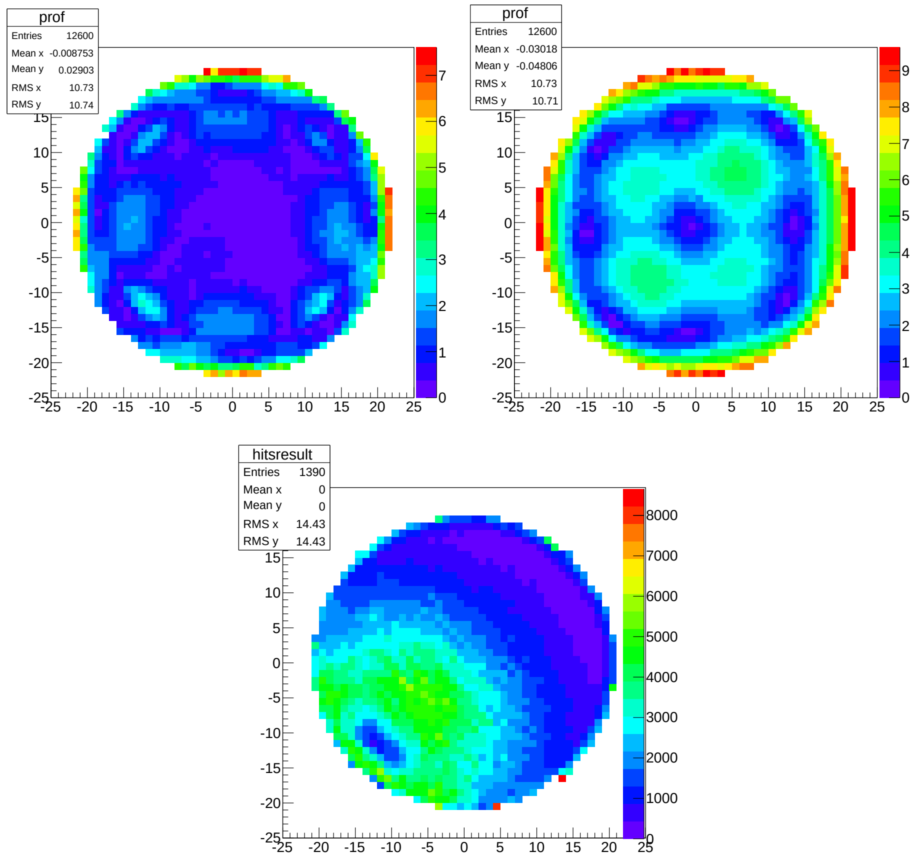
Top: Radial error with raised liquid level and 16 effective PMTs. *Bottom*: Radial error with lowered liquid level and 16 effective PMTs

Clearly bringing the PMTs closer to the primary point makes a small, but noticeable difference in the position resolution. This is to be expected as was noted earlier. Now for just 4 effective PMTs.



Top: Radial error with raised liquid level and 4 effective PMTs. *Bottom*: Radial error with lowered liquid level and 4 effective PMTs

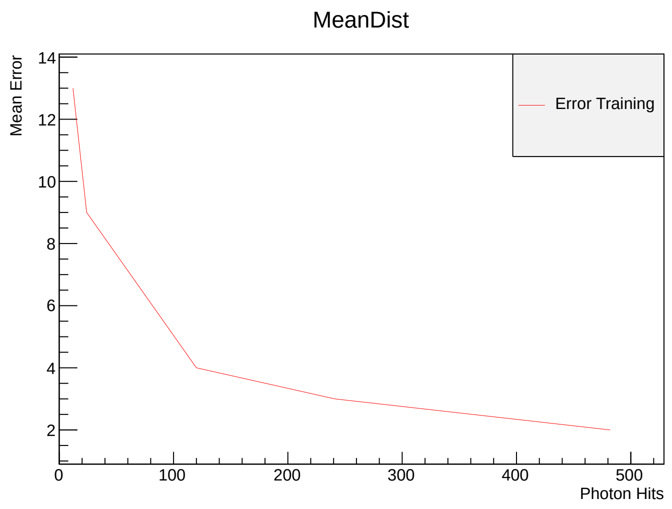
For all of these runs, the $n_{hiddenneurons} = n_{PMT}$. Thus, adding sensitivity to the PMT configuration decreases the mean, radial error by about a factor of 3. One can also examine the uniformity of the MLP's accuracy in the horizontal plane.



Left: Radial Error (mm) versus the primary position in the horizontal plane with 16 PMTs (no top mesh). *Right*: Radial Error (mm) versus primary position with only 4 PMTs (no top mesh). *Bottom*: Change in PMT1 photon count due to a small change in position versus the primary position (no top mesh).

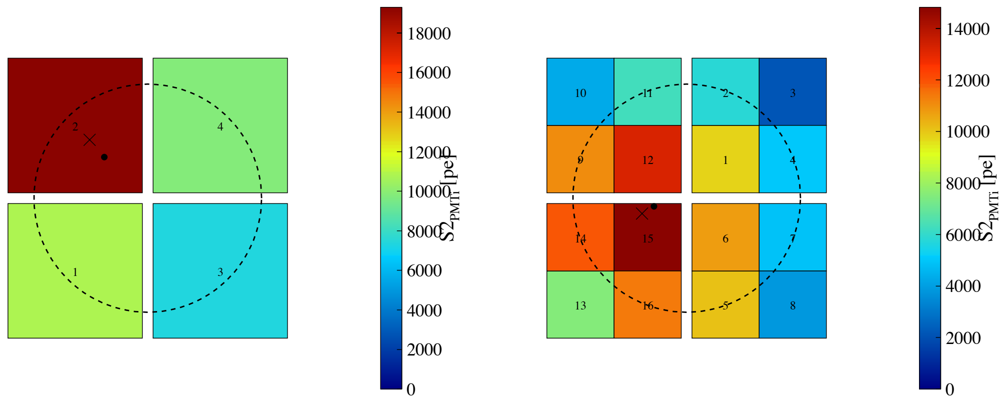
According to the top left graph, approximately 91% of the cross-sectional area has an error below 3mm, 86% has error below 2mm and 52% an error below 1mm. For the top right graph, approximately 52% of the area is below 3mm, 24% below 2mm and only 5% below 1mm. Obviously, the split PMT anodes make a significant difference in position resolution. Some non-uniformity is to be expected for both configurations. Near the edges, reflections from the side of the TPC can take a toll on resolution. Also, when the primary is directly over a PMT, that PMT gets saturated with photons. In other words, a small change in position around that point leads to an indiscernible change in the hit pattern, as can be seen for PMT1 in the bottom graph. PMT1 is in the lower left hand side of the cylinder.

These simulations were designed to get the best position resolution possible. With more realistic scintillation yields, this resolution capabilities get worse. This can easily be seen by graphing the average radial error versus the number of detected photons, which is directly proportional to the number of photo-electrons by the Quantum Efficiency (QE) of the PMTs.



Radial error versus number of photons that hit the top four PMTs.

Finally, a more down to earth look at what a test run looks like. Both of these MLPs were trained with four top PMTs in the simulation. The PMTs are numbered as they were in the simulation. The point marked with an X is the reconstructed position. The point marked with the dot is the actual, simulated position.



Left: Number of detected photons versus PMT number along with reconstructed and actual position. *Right*: Same but with added sensitivity.

5. Summary

Using the Geant4 simulations to create a training data set, a multilayer perceptron was trained to the task of reconstructing the horizontal event position. With 4 photo-sensitive regions, a resolution was obtained on the order of $\approx 3\,\text{mm}$. With 16 photo-sensitive regions, the accuracy improved with a mean error on the order of $\approx 1\,\text{mm}$.

References

[1] Roberto Lopez, *Flood: A User Guide*. International Center for Numerical Methods in Engineering. http://www.cimne.com/flood/download.asp.

[2] Guillaume Plante, *The XENON100 Dark Matter Experiment*. Ph.D. Dissertation, Columbia University, 2012.

[3] Yuan Mei, *Direct Dark Matter Search with the XENON100 Experiment*. Ph.D. Dissertation, Rice University, 2012.