FLERCNN Reconstruction Tech Note

Jessie Micallef, Shiqi Yu, Brian Clark

April 2022

Contents

1 Introduction

To detect atmospheric neutrinos above 10 GeV, detectors must be dense enough to reconstruct the neutrino interactions while also being large enough to compensate for the steeply falling atmospheric neutrino spectrum. This is a considerable experimental challenge that IceCube is uniquely suited to handle. Sensitivity to oscillation parameters, dependent on the distance traveled over the neutrino energy (L/E), is limited in IceCube by the resolution of the arrival

angle (which determines L) and energy (E). Event reconstruction improvements can therefore directly lead to advancements in oscillation results. Other important parameters are the flavor of the neutrino, classified in IceCube as track-like (ν_μ CC) or cascade like (ν_e CC, ν_τ CC, and all flavors NC). Other important neutrino interaction parameters to reconstruct include the interaction vertex of the neutrino (x, y, z) and being able to distinguish neutrino events from background muon events. Likelihood-based reconstruction methods are currently used to reconstruct the energy (E) and arrival angle (determines L), which are the necessary variables to constrain the oscillations parameters, along with variables such as the interaction vertex. These reconstructed variables are then fed into a BDT for both PID and muon/neutrino distinction. An alternate solution is to apply Convolutional Neural Networks (CNNs) to improve the resolution and runtime of these reconstructed variables. The Fast Low Energy Reconstruction using Convolutional Neural Networks (FLERCNN) is the attempt at this solution.

The IceCube DOMs record the Cherenkov radiation from neutrino interactions, which can happen anywhere in the detector volume, meaning the events are characterized by translational invariance. This property allows us to apply Convolutional Neural Networks (CNNs) to the event reconstruction. CNNs are often used in image recognition as they are able to identify an object independently of where the object is positioned within an image. [**?**, **?**]. In a CNN, a set of filters or 'kernels' is used to build maps of spatial features from the input data. By combining several convolutional layers, it is possible to reconstruct complex images. In IceCube, this architecture can be used to build a reconstruction using a network of convolutional layers.

The low energy CNN architecture is similar to the CNN applied for IceCube's high energy reconstruction [**?**]. In IceCube, low energies are particularly difficult to reconstruct due to the sparseness of the detector. Low-energy events also produce less scattered light, which reduces the information available for reconstruction and makes the event topologies less distinctive. Thus, the low-energy CNN includes key optimizations from the high-energy CNN reconstruction.

There are five uses for the FLERCNN architecture: regression for energy, regression for zenith, regression for vertex (x, y, z), classification for track vs. cascade-like (PID), and classification for muon vs. neutrino-like. Each network is trained with a specific training sample that is optimized for reconstruction or classification of the desired output. While the FLERCNN architecture remains largely the same, the activation function for the output is changed according to the variable.

2 Architecture

The low-energy CNN uses only the DeepCore strings and the center-most IceCube strings (Figure 1). All hits per DOM are summarized into 5 charge and timing variables, which are separately given to the CNN for the the DeepCore and IceCube strings because they have different z-spacing between DOMs, cre-

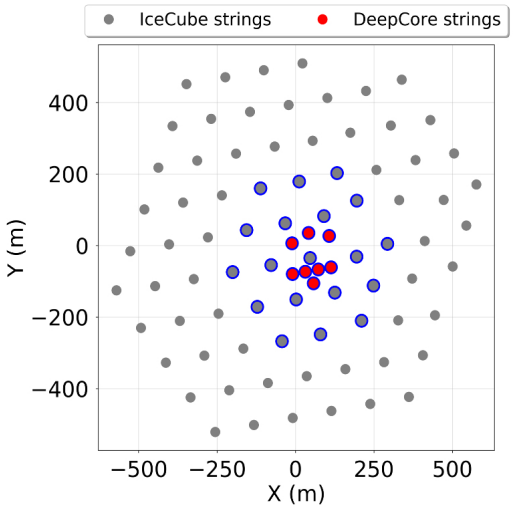

Figure 1: Top view of the IceCube detector, with DeepCore strings in red, IceCube strings in gray, and a blue highlight on all strings used for the low-energy CNN.

ating a two-branched network architecture 2).

All 8 DeepCore strings and the center-most 19 IceCube strings are used as input; this accounts for the first dimension in the input array, which corresponds to the string index. The CNN kernel spans the vertical, or z-depth, of the strings only (Figure **??**); no convolution is applied in the xy-plane since the DeepCore strings are deployed in an irregular array. The second dimension in the input array uses all 60 DOMs on both the DeepCore and IceCube strings. The last dimension corresponds to the 5 input variables that summarize the (potential) multiple hits that a DOM would record in an event. These are the sum of the charge, time of the first hit, time of the last hit, charge weighted mean of the hits, and charge weighted standard deviation of the hits.

There are two main differences in the architecture and training procedure of the networks: the chosen loss function, used to score how accurate the CNN's reconstruction is, and the activation of the final stage output. The main differences are between the regression networks (energy, zenith, and vertex) which predict a continous output value vs. the classification netwowrks (Muon and Track), which predict a single probability between two binary classes.

3 Preparing the Training Samples

Various Monte Carlo (MC) samples were specifically generated for training FLERCNN using GENIE [**?**], a neutrino MC generator with an emphasis on neutrino interaction physics at the few-GeV energy range. The ultimate goal is

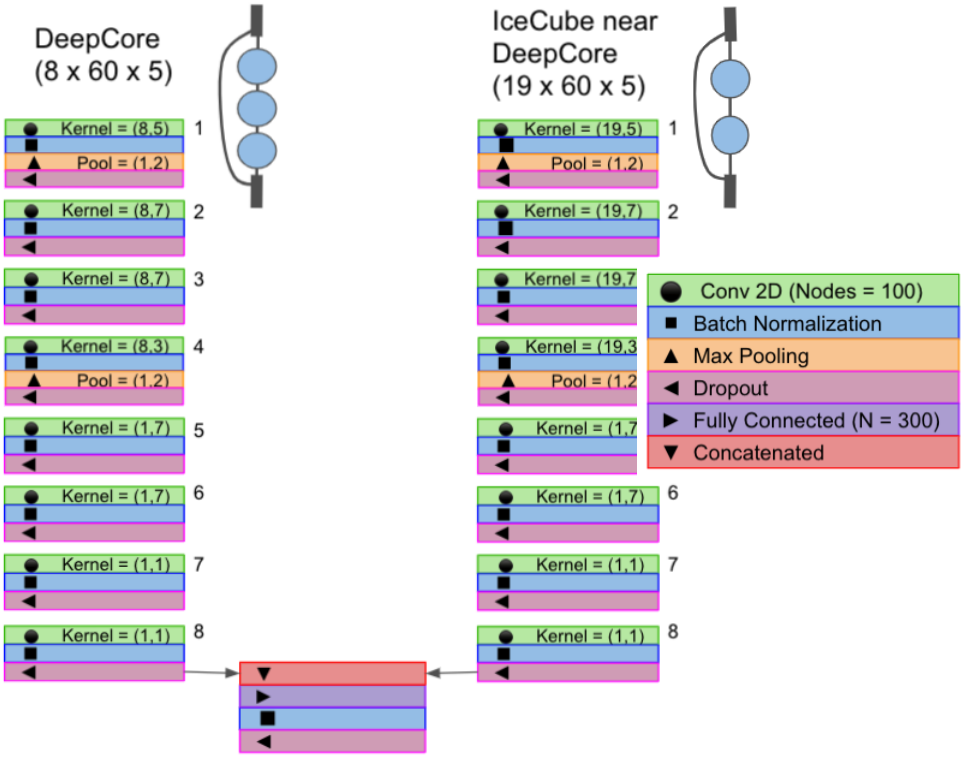

Figure 2: Detailed architecture for the Convolutional Neural Network, legend for color layers on the right.

to create samples that are unbiased for the specific training variable, one that ideally extends past the target reconstruction region.

3.1 Common Pre-Processing

Once the sample is generated, it is processed up to oscNext Level5 with no changes (oscNext Sample Tech Note). After this a "FLERCNN Level6" is run, which includes some fast algorithms (typically run at oscNext L6 and L7) without any Retro-Reconstruction dependence are run to determine variables that are later used in final analysis cuts to improve data/MC agreement. This includes the Coincident Cut variables, Corridor Cut variables, and SANTA Direct Pulses variables. FLERCNN Level6 also ensures that only frames that passed the Level 5 cuts are stored in the Level 6 files.

For the simulation generated with the intention of cutting for uniform energy, the raw files used to generate the training samples are stored at `/data/ana/LE/oscNextNN/level6`. For the simulation generated with the intention of cutting for uniform cosine zenith, the raw files used to generate the training samples

Table 1: Describes the loss function and activation used for each network.

CNN	Loss Function	Activation
Energy	Mean Absolute Percentage Error	Linear
Zenith	Mean Squared Error	Linear
Vertex	Mean Squared Error	Linear
Muon	Binary Crossentropy	Sigmoid
Track	Binary Crossentropy	Sigmoid

are stored at `/data/ana/LE/oscNext\_Zenith`.

All training sets use the `SRTTWOfflinePulsesDC` pulse series, which have a coincidence clustering algorithm applied to the series remove noise hits (see more at SLC hit cleaning.

A study performed by Spencer Axani showed that data/MC disagreement is evident in charges below 0.25 PE, as outlined in Spencer Axani's SPE Template Technote. A plot showing this from the technote is included in Figure 3. The data and MC used for FLERCNN uses the "default discriminator" settings, and thus pulses below the < 0.25 PE threshold are cut from both training and testing samples for FLERCNN.

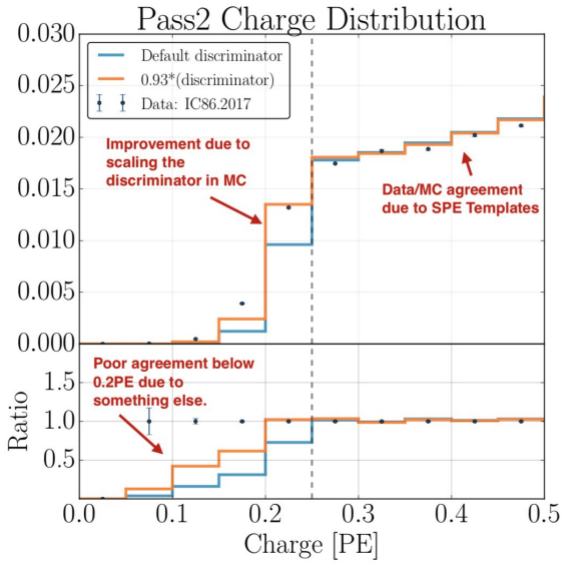

Figure 3: Shows the rate of the default discriminator in blue (from the SPE templates used to generate FLERCNN data and MC) compared to data as a function of charge. Agreement is seen above 0.25 PE.

Lastly, some of the seasons of data are quantized into 0.05 PE steps. To make

all data and MC appear uniform, all pulse charges are quantized (in 0.05 PE steps) before being used to calculate the variables for the CNN's input features.

3.2 Pre-Processing Differences between Training Sets

As these networks were developed over the course of three years, along with developing unique samples optimized for specific training variables, there are some differences in the pre-processing for the FLERCNN networks. Table 2 summarizes which networks use the pre-processing methods described below.

To ensure that the events appear as "neutrino-like" as possible, a cut is applied on the oscNext BDT trained at Level4 to distinguish noise events. The training of this BDT is outlined in Section 3.6.2 of the oscNext technote. This cut is at ProbNu > 0.95. By Level6, this cut only affects a small number of neutrino events, as earlier cuts have already removed any suspicious-looking events. It should be noted that none of the CNN training samples include pure noise events, but all do include pulses in the neutrino (or muon) event that originate from noise, though it is limited by the cuts described in 3.1 and 3.3.

In accordance with the Retro Reconstruction method, a similar cut was applied to some CNN training samples to check that there are at least 8 hits in the pulse series. This cut is used on Retro due to the fact it is simultaneously reconstructing 8 features of the event. Requiring a minimum number of pulses seemed reasonable for the CNN too, and so this condition was initially required for most training samples. It was later recognized that the CNN cut could be more tuned, and thus a cut on number of DOMs in the CNN strings (8 DeepCore and center-most 19 IceCube strings) was used. During reconstruction, the number of hits and number of DOMs is not strictly enforced, but better energy resolution was found by cutting on number of CNN-seen DOMs and is therefore used at the analysis level to improve overall resolution. Some training samples apply one or both of these cuts, though it is suggested that future training samples for the CNN use just the number of DOMs in the CNN strings.

Lastly, most training samples relied on early processing (Level 2-5) to eliminate events with starting vertices outside of the DeepCore region. Training the zenith network, however, relies on showing the network examples of full track signatures. Thus, a cut was applied so that all events started and ended in the CNN strings, using the true vertex, direction, and track length to determine. This cut is called "containment" in Table 2

3.3 Preparing Input Features

All input features undergo a selection criteria, to ensure that the pulses given to the CNN have good data/MC agreement and that the sample has minimal extraneous noise hits.

The first step done in this process is that all pulse times are shifted to be relative to the SMT3 DeepCore trigger, so that the trigger time is at 0 ns. We then remove any events with no SMT3 trigger or greater than 1 SMT3 triggers, since it is unclear how to define 0 (trigger time) for the event in these cases. After

Table 2: Describes the additional training sample cuts applied during preprocessing.

CNN	**ProbNu > 0.95**	**# Pulses >= 8**	**# DOMs on CNN stings**	**Containment**
Energy	No	No	>= 7	No
Zenith	No	No	No	Yes
Vertex	Yes	Yes	No	No
Track	Yes	Yes	No	No
Muon	Yes	Yes	>= 4	No

Table 3: Describes the transformations that the input features undergo during preprocessing.

Input Variable	**Shift**	**Division Factor**	**Requirement**
Sum Charge	None	25	> 0.25 PE; rounded to nearest 0.05 step
Time First Pulse	By trigger time	4000	Uses times in [-500,4000] ns
Time Last Pulse	By trigger time	4000	Uses times in [-500,4000] ns
Charge Weighted Mean	Uses times shifted by trigger time	4000	Uses times in [-500,4000] ns; pulses > 0.25 in 0.05 PE steps
Charge Weighted Standard Deviation	Uses times shifted by trigger time	2000	Uses times in [-500,4000] ns; pulses > 0.25 in 0.05 PE steps

this shift is applied, only the pulses in [-500, 4000] ns window around trigger are used to train the CNN. This corresponds to about 8 scattering lengths, and thus it is assumed that most pulses outside this window are noise. If the pulse times do not fall into the time window, they are not used for any of the 5 summary variables, including the sum of the charge for that DOM.

There are some transformations applied to the input variables to help the CNN train quicker. CNNs prefer inputs in a defined range, particularly [-1,1]. Thus, the input variables are divided by static, near maximum numbers determined early on from histograms of the raw input. The following transformations per event per DOM variable are [sum of charge/25 PE, time of first pulse/ 4000 ns, time of the last pulse / 4000 ns, charge weighted mean of pulse times/ 4000 ns, charge weighted standard deviation of pulse times / 2000 ns].

All CNN samples undergo the pre-processing described in 3 before being specifically optimized for their reconstruction variable.

3.4 Training Each CNN

To confirm the network does not overtrain on the given training sample, 20% of the sample is set aside to validate the network's progress after each full pass through the data. After each full pass through the data (called an *epoch*), the network is evaluated on both the training and validation sample using a specified *loss function*, which is defined by the user and chosen per variable. Note that typically, FLERCNN was trained using multiple files containing training sample, so the loss function was evaluated at the end of each loss file, on a subset of the validation sample. The next section will outline the optimizations in the training sample, as well as the activation and loss functions, for each network adaptation.

The differences between the CNN training sample and network differences are summarized in Tables 4, 5, 6 with more depth in the following sections.

Table 4: Describes the composition of the training samples for each network

CNN	**Particle(s)**	**ν Interactions**	**# Events**	**Energy Range (GeV)**
Energy	ν_μ	CC	9 million	1-500
Zenith	ν_μ	CC	5 million	1-300
Vertex	ν_μ	CC	5 million	1-500
Track	ν_μ, ν_e	CC, NC	5 million	5-200
Muon	ν_μ, ν_e, μ	CC, NC	7 million	ν 5-200, μ 150-5000

3.4.1 Energy

A sample of more than 9 million unweighted, charge-current (CC) ν_μ was generated to train FLERCNN for energy. The goal was to have a flat, unbiased energy distribution for training, between 1-500 GeV, since the target reconstruction region is 5-100 GeV for FLERCNN to use for the muon neutrino disappearance

Table 5: Describes the training samples sizes and final model used

CNN	**# Files**	**1 File Size**	**Raw Training Pass**	**Epoch**
Energy	18	16G	594	33
Zenith	10	19G	700	70
Vertex	8	21G	232	29
Track	12	15G	192	16
Muon	12	18G	240	24

Table 6: Describes the training samples learning rate and changes

CNN	**LR Start**	**LR Drop**	**LR Drop Factor**	**LR at chosen model**
Energy	0.001	200	0.5	2.5e-4
Zenith	0.001	80	0.8	1.7e-4
Vertex	0.001	50	0.1	1e-7
Track	0.001	50	0.1	1e-6
Muon	0.001	-	-	0.001

Table 7: Describes loss functions used for each of the networks, equations show in the loss function plots

CNN	**Loss Function**
Energy	Mean Absolute Percentage Error
Zenith	Mean Squared Error
Vertex	Sum(Mean Squared Error) for X, Y, & Z
Track	Binary Crossentropy
Muon	Binary Crossentropy

analysis. Most energy bins (1 GeV) have 20,000 events per bin, creating the "flat" sample.

It was found that the events with low number of DOMs hit were difficult for the network to resolve (shown at this presentation, and thus, these events were removed both when training and evaluated the network. The cut requires that the CNN has at least 7 DOMs with input information, requiring that there are 7 DOMs hit between the 8 DeepCore strings and 19 inner-most IceCube strings for the event to be reconstructed with FLERCNN. The same was split into 18 files, each of which had about X number of events for training and X for validation.

The total sample size is 9,026,280 events with 80% of the sample was used for training and 20% for validation. To cope with the large file size that comes with storing the necessary pulse series information for training, the sample was split into 18 files that were 16G each, stored on MUS's HPCC in the directory `/mnt/research/IceCube/jmicallef/DNN_files/NuMu_genie_149999\_final_level6_cleanedpulses_transformed_IC19_nocut8hits_E1to500_CC_all_start_all_end_1minDOMflat_499bins_20000evtperbin_file??.hdf5`.

To train, the learning rate started at 0.001 and was changed by a factor of 0.5 after 200 passes through training. Since each training pass was only through 1/18 of the data, the drop occurs at the equivalent of 11.11 epochs. The model used is after 594 training passes, or 33 epochs.

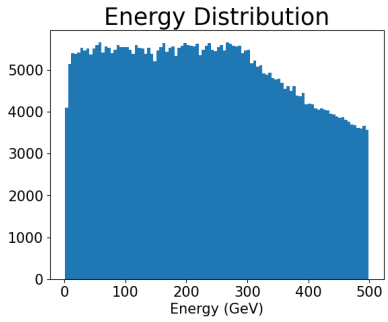

Figure 4: Part of the energy training sample, unbiased in energy due to the nearly flat distribution per GeV

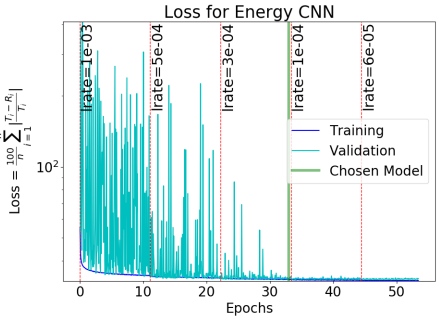

Figure 5: Loss vs. epoch progression as the energy network trained

The training sample proved particularly tricky for the zenith sample as the sample cannot be generated with extended bounds in the same way that the energy network is. The network was trained for the zenith angle in radians. Even though the training bounds could not be extended, a distribution with a flat zenith sample was created exclusively to train for zenith while minimizing bias. This flat zenith training distribution was generated and tested by Dr. Shiqi Yu. The containment cuts applied to the starting and ending position of the muon from the ν_μ CC interaction requires $-505m < z < 500m$ and

$r_{string36} < 260m$.

After these cuts, there is a total of 5,024,876 events with 80% of the sample was used for training and 20% for validation.

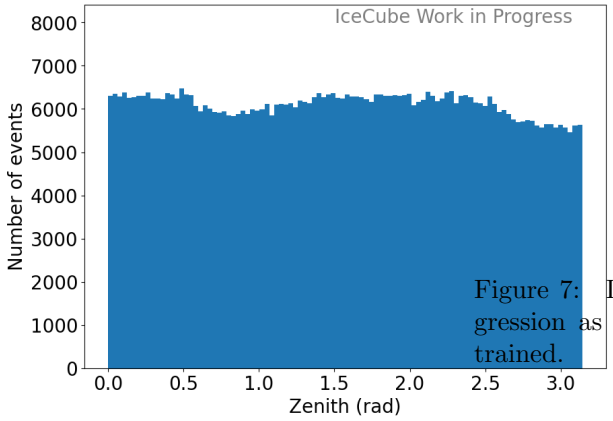

Figure 6: Zenith training sample, unbiased in zenith angle due to the nearly flat distribution per radian.

figures/flercnn/loss_vs_epochs_zenith.png

Figure 7: Loss vs. epoch progression as the zenith network trained.

3.4.2 Vertex

The vertex was trained using an older version of a training sample initially created to train the energy CNN. As the energy sample, it includes only ν_μ CC events. Instead of the number of DOMs in CNN strings cut, it uses the number of pulses cut (required at least 8 hits). This training sample was not specifically optimized for vertex reconstruction, but it was found that it had a passable resolution and thus no further optimizations were applied. There were a few attempts at optimization, testing a few different learning rates and learning rate drops, and also transforming the input data. However the original learning rate of 0.001 and the untransformed data performed just as well as these, so we kept the original parameters.

The total sample size is 4,982,335 events with 80% of the sample was used for training and 20% for validation. To cope with the large file size that comes with storing the necessary pulse series information for training, the sample was split into 8 files that were 21G each, stored on MUS's HPCC at `/mnt/research/IceCube/jmicallef/DNN_files/old_files/extended_jan/NuMu_genie_149999_level6_cleanedpulses_transformed_IC19_E1to500_CC_all_start_all_end_flat_499bins_20000evtperbin_file00.hdf5`.

To train, the learning rate started at 0.001 and was changed by a factor of 0.1 after 50 passes through training. Since each training pass was only through

1/8 of the data, the drop occurs at the equivalent of 6.25 epochs. The model used is after 232 training passes, or 29 epochs.

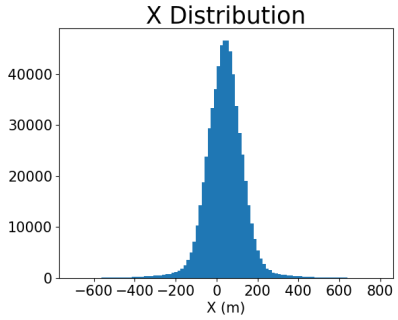

Figure 8: X distribution for part of the vertex CNN training sample, optimized to be unbiased in energy but repurposed to train the vertex classifier

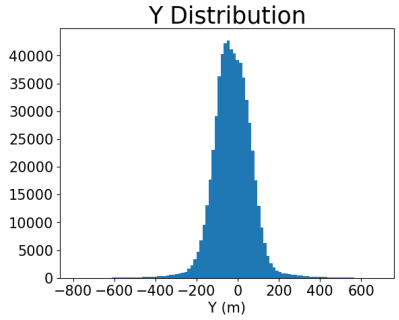

Figure 9: Y distribution for part of the vertex CNN training sample, optimized to be unbiased in energy but repurposed to train the vertex classifier

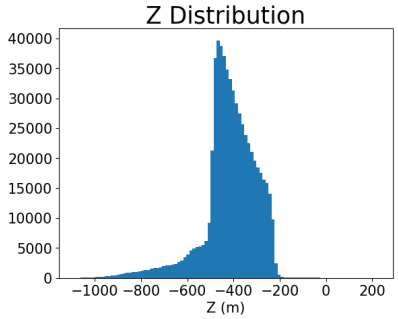

Figure 10: Z distribution for part of the vertex CNN training sample, optimized to be unbiased in energy but repurposed to train the vertex classifier

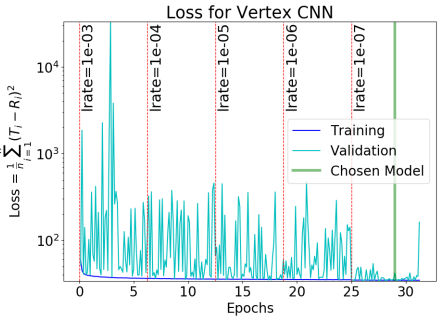

Figure 11: Loss vs. epoch progression as the vertex network trained, where the loss is the sum of the mean squared error calculation for X, Y, and Z

3.4.3 Track Classifier

The PID sample was enforced to have a 50:50 distribution of track:cascade-like events by using ν_μ CC MC for track-like and ν_μ NC and ν_e CC & NC for cascade-like events. This was enforced per energy bin, so that across the energy range (5-200 GeV), there are 15,000 track events and 15,000 cascade events per 1 GeV. In total, it contained 5,263,514 events, with 80% being used for training

and 20% for validation. To cope with the large file size that comes with storing the necessary pulse series information for training, the sample was split into 12 files that were 15G each, stored on MUS's HPCC at `/mnt/research/IceCube/jmicallef/DNN_files/PID_TracksCascades_genie_level6_cleanedpulses_transformed_IC19__E5to200_all_all_start_all_end_flat_195bins_30000evtperbin_file??.hdf5`.

To train, the learning rate started at 0.001 and was changed by a factor of 0.1 after 50 passes through training. Since each training pass was only through 1/12 of the data, the drop occurs at the equivalent of 4.17 epochs. The model used is after 192 training passes, or 16 epochs.

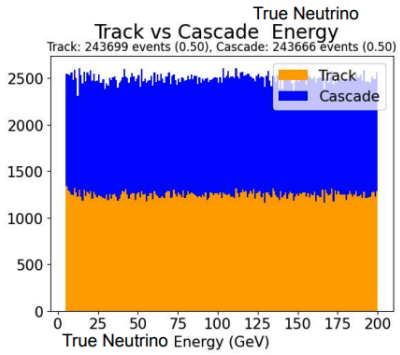

Figure 12: Energy distribution for the PID training sample, illustrating that the sample was balance both in track/cascade composition along with energy distribution

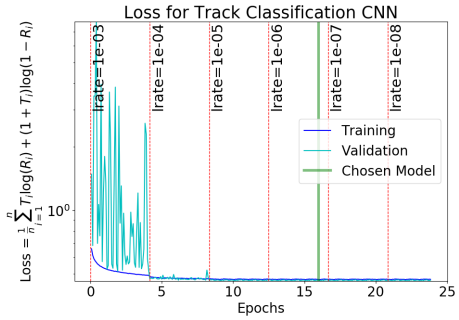

Figure 13: Loss vs. epoch progression as the track classification network trained, using the binary crossentropy loss function

3.4.4 Muon Background Classifier

Part of the Track Classifier Training sample was re-purposed to train the muon classifier, with the addition of the existing oscNext L4 Muongun sample. The Level4 Muongun is used due to lack of statistics at FLERCNN Level6, as described in this presentation There is a cut applied enforcing that number of DOMs in the CNN strings >= 4. The neutrino samples are derived from the PID sample, with the additional number of DOMs cut. It was also cut down in size to match the muon sample, which was 10% of the generated Level 4 Muongun.

The sample is split 40:40:20 between μ:ν_μ:ν_e, including both CC and NC events in the neutrino samples. The original training sample comprised of 50:25:25 distribution of μ:ν_μ:ν_e found a bias with more ν_μ events mis-classified as μ. Therefore, more ν_μ sample was added to help the network distinguish the two event types.

The total sample size is 6,923,001 events, with about 2.8 million μ and ν_μ events, and 1.8 million ν_e events. Again, 80% of the sample was used for training and 20% for validation. To cope with the large file size that comes with storing the necessary pulse series information for training, the sample was split into 12 files that were 18G each, stored on MUS's HPCC at `/mnt/research/IceCube/jmicallef/DNN_files/oscNext_level6_flercnn_pass2.129999_1.4mill_149999_2.8mill_130000_2.8mill.cleanedpulses_transformed_IC19.no_cuts_file??.hdf5`.

To train, the learning rate started at 0.001 and was not changed through the duration of training. The model used is after 240 training passes, or 24 epochs since there are 12 files total in the training sample.

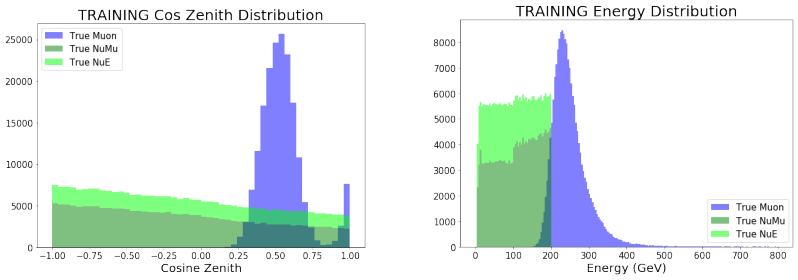

Figure 14: True cosine zenith and energy distribution used to train the muon classifier. Note that this is different from the deposited/visible energy, which is much lower for the muon sample due to the majority of the remaining muon being muons sneaking through the corridors of IceCube. The ν_μ and ν_e histograms are stacked, in comparison with the muon histogram (not stacked).

4 Performance

Testing samples use the oscNext generated simulation. More information on the generation, along with processing levels 2-5 are in the oscNext sample tech note. After Level 5, there is a FLERCNN specified pipeline that uses the FLERCNN reconstruction to apply final level analysis cuts optimized for a ν_μ disappearance analysis. More detailed information on these cuts and their motivation can be found in the FLEERCNN analysis tech note.

The testing MC distributions are weighted with atmospheric flux and oscillation weights, making them look as close as possible to the expected data distribution. This is very unlike the training sample, generated to be unbiased in the training sample. The difference between the training and testing sample is helpful to evaluate the adaptability of the CNN and make sure it is not overtrained to only perform well on distributions that look like the training sample.

In most the following plots, the sample is split into resolutions for two samples: true tracks and and true cascades. True tracks are ν_μ CC events only,

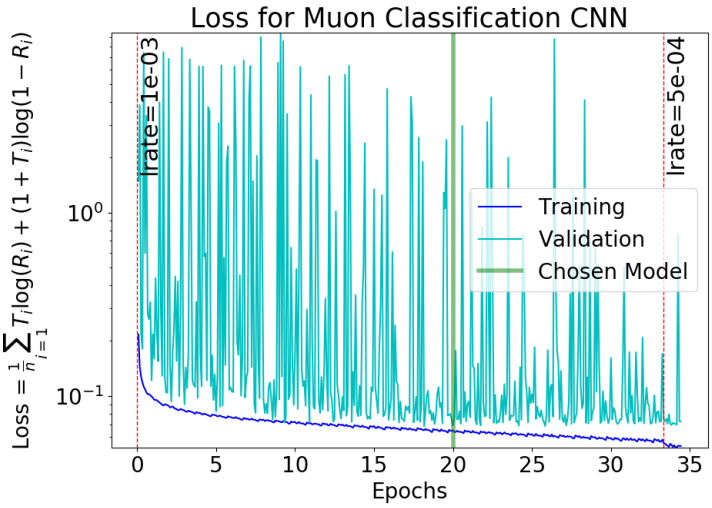

Figure 15: Loss vs. epoch progression as the muon classification network trained, using the binary crossentropy loss function.

Table 8: Describes cuts on the analysis sample and their individual impact when added to the final sample

Cut Name	**Value**	**Percent Removed**
# CNN DOMs	≥ 7	1.05%
Direct Pulses	> 2.5	10.50%
NTop15	< 0.5	0.03%
NOuter	< 7.5	0.001%
Radius	< 200 m	0.09%
Z	-495 m $<$ z $<$ -225 m	5.48%
Vertex	R and Z cut	6.12%
Energy	5 GeV $<$ E $<$ 100 GeV	20.70%
Cosine Zenith	< 0.3	19.66%

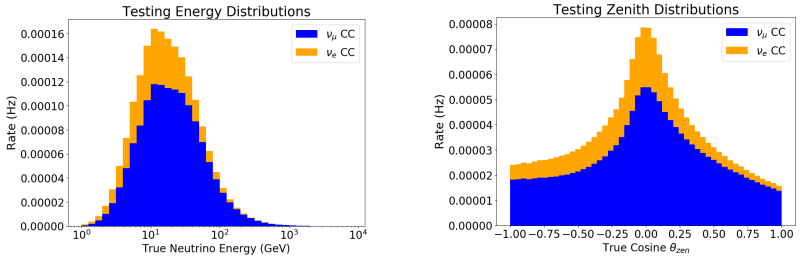

Figure 16: Testing distributions for the ν_μ CC and ν_e CC events. The energy and zenith cuts are not applied in this case, since they will be applied based on the individual reconstructions' outputs during testing.

where true cascades are ν_e CC, $\nu_t au$ CC, and all ν NC. Note that 17% of ν_τ CC events are tracks, but are misclassified here in the resolution plots.

The following results compare FLERCNN (called CNN) with Retro Reco (called Likelihood). Retro Reco is a table-based likelihood reconstruction method. More information on this method is outlined in the Low Energy Event Reconstruction in IceCube DeepCore paper.

4.1 Energy

When plotting the energy resolution, the energy cut is not applied. This is because it would cause an artificial bias on the edges of the true energy resolution range.

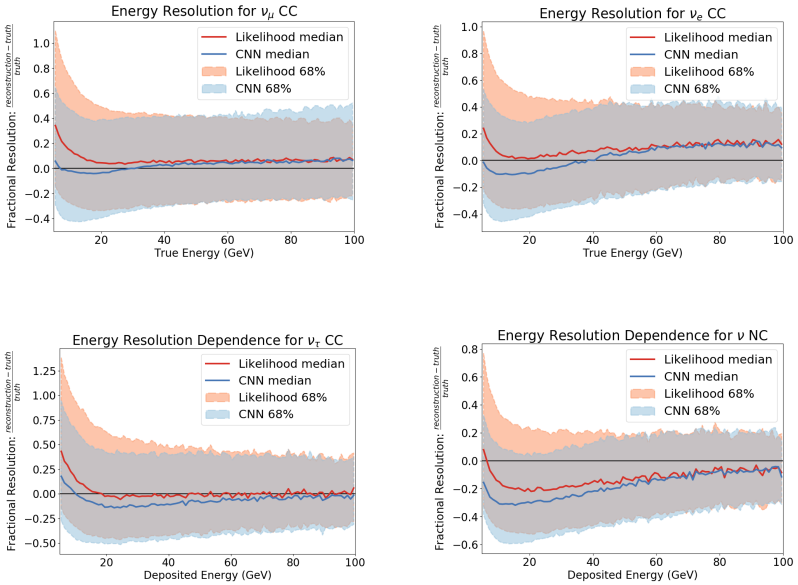

Figure 17: Resolution shown for true energy for the ν_μ CC and ν_e CC events. With the invisible energy carried off by the outgoing neutrinos from NC interactions and some ν_τ CC interactions, the resolution is shown vs. deposited energy for these selections.

4.2 Zenith

High energy tracks are generally easier for both reconstructions (CNN and Retro) due to the long lever arm that the reconstructions have to work with. The slight difference in the 68% region for teh CNN at high energies is likely due to the track leaving the CNN string region (19 IceCube strings + 8 DeepCore strings). Future imporovements to resolution could consider adding additional strings (potentially in some summary manner, since the network trains slower

and would require much more memory with even one additional ring of IceCube strings added). Another solution would to be to train an ending position vertex network that could provide a containment cut to ensure events remain in the CNN string region.

Cascades prove to be more challenging for both reconstructions, though they have similar resolutions across all energies. Even though the CNN is trained on ν_μ CC events only, it manages to reconstruct reasonable zenith positions on average for the sample, with a 68% spread comparable to Retro Reco.

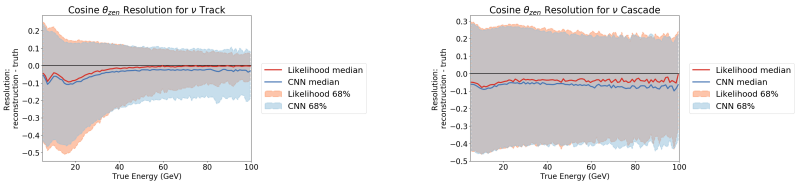

Figure 18: Resolution shown for reconstructed cosine zenith for the ν_μ CC tracks and remaining cascade events.

4.3 Vertex

For the vertex reconstruction, there is a known artifact in the CNN reconstruction where the string positions are favored in the x and y reconstruction (contributing to ρ_{36}). Since this reconstruction is used for the containment cut, the precise reconstruction position is not necessary to make a reasonable containment cut. The favoring of string positions is evident in Figure 19.

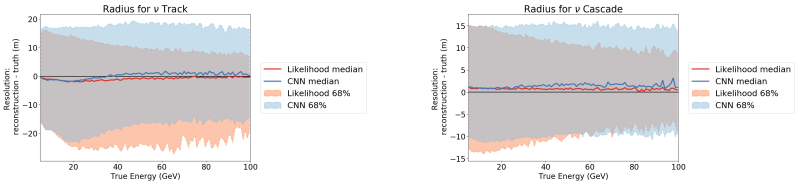

Figure 19: Resolution shown for reconstructed radius from the center string (string 36) of the starting vertex for the ν_μ CC tracks and remaining cascade events.

The performance of using this reconstruction as a classifier of events inside the volume vs. outside the contained volume is in Figure 21. Note that it keeps most of the events inside of the desired detector region (95.65%) and successfully eliminates 52.03% of the events outside of the region.

Future studies with the vertex classifier could be to generate a uniform in vertex sample, or at least a sample that has more events outside of the desired detector region. CNNs are known for their ability to interpolate, not extrapolate, so it is expected that the CNN performs better on classifying the event

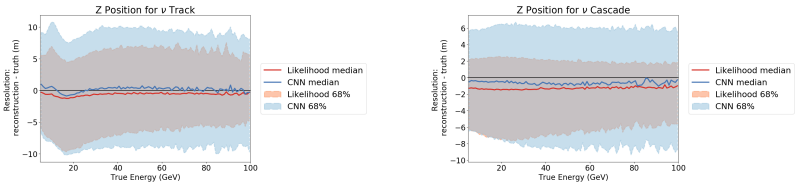

Figure 20: Resolution shown for reconstructed z position of the starting vertex for the ν_μ CC tracks and remaining cascade events.

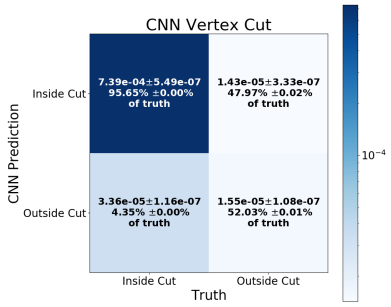

Figure 21

types it has seen (inside of the desired cut volume) rather than events that were more rare in the training sample (outside of the desired cut volume). As is, this cut still has the necessary effect to improve the sample resolution and contribute to eliminating remaining background events, such at atmospheric muons.

4.4 Track Classifier

At the GeV-scale for IceCube analyses, it is known that many version of track classifiers (typically BDT-based in the past) have difficulty with accurate classification. These events often have only a handful of hit DOMs, making it difficult to determine the track-like structure from the cascade due to short track lengths. Thus, we do not expect an Area Under the Curve close to 1.0, but can see that the

4.5 Muon Classifier

5 Robustness to Systematics

5.1 Conclusion

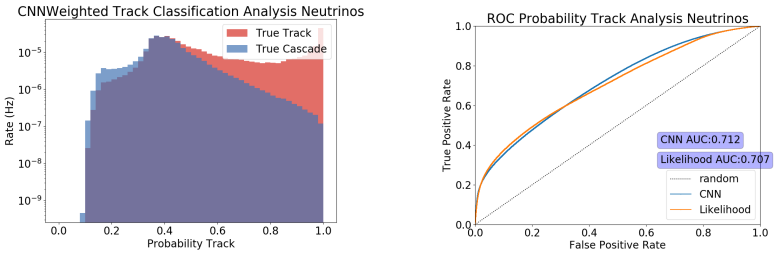

Figure 22: Performance of Track classifier

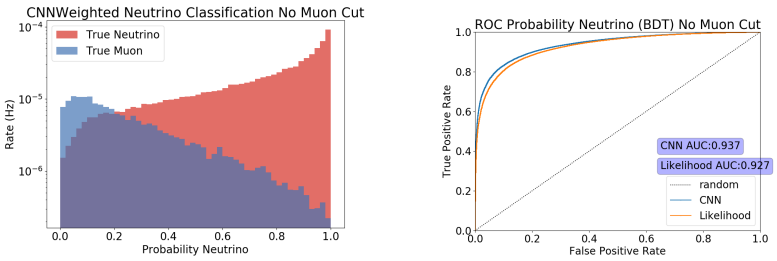

Figure 23: Performance of Muon classifier

Figure 24: Muon efficiency as a function of energy

Reconstruction Method	**Avg Time per Event**	**Events per day (single core)**	**Time for 10^8 events (1000 CPU cores or GPU nodes)**
5 CNNs on GPU	0.0011 s	80,000,000	2 min
5 CNNs on CPU	0.29 s	300,000	8 hours
SANTA/LEERA	0.16 s [**?**]	500,000	4 hours
Retro	40 s [**?**]	2,000	46 days

Table 9: Runtime and implications of applying the different reconstructions to the MC sample.