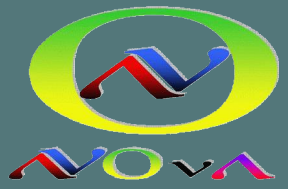


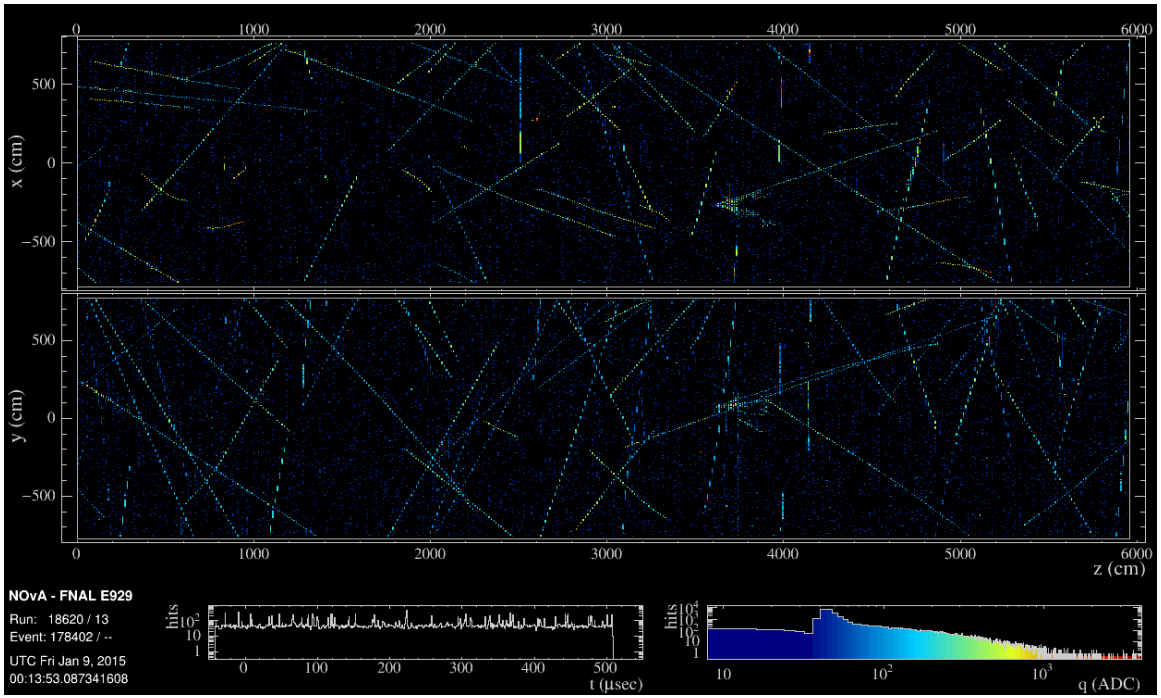
UNIVERSITY OF Cincinnati

Categorizing Neutrino Interactions using Convolutional Neural Networks

Adam Aurisano
University of Cincinnati

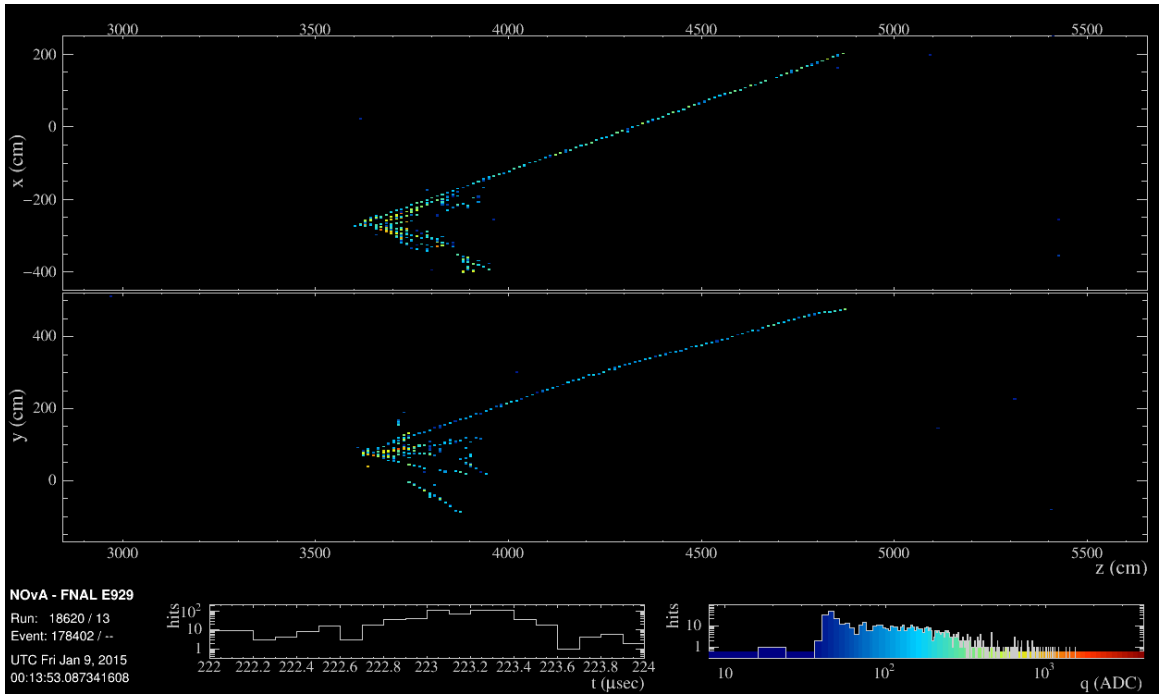
S2I2 HEP/CS Workshop
University of Illinois at Urbana-Champaign
8 December 2016

Can You Find the Neutrino?



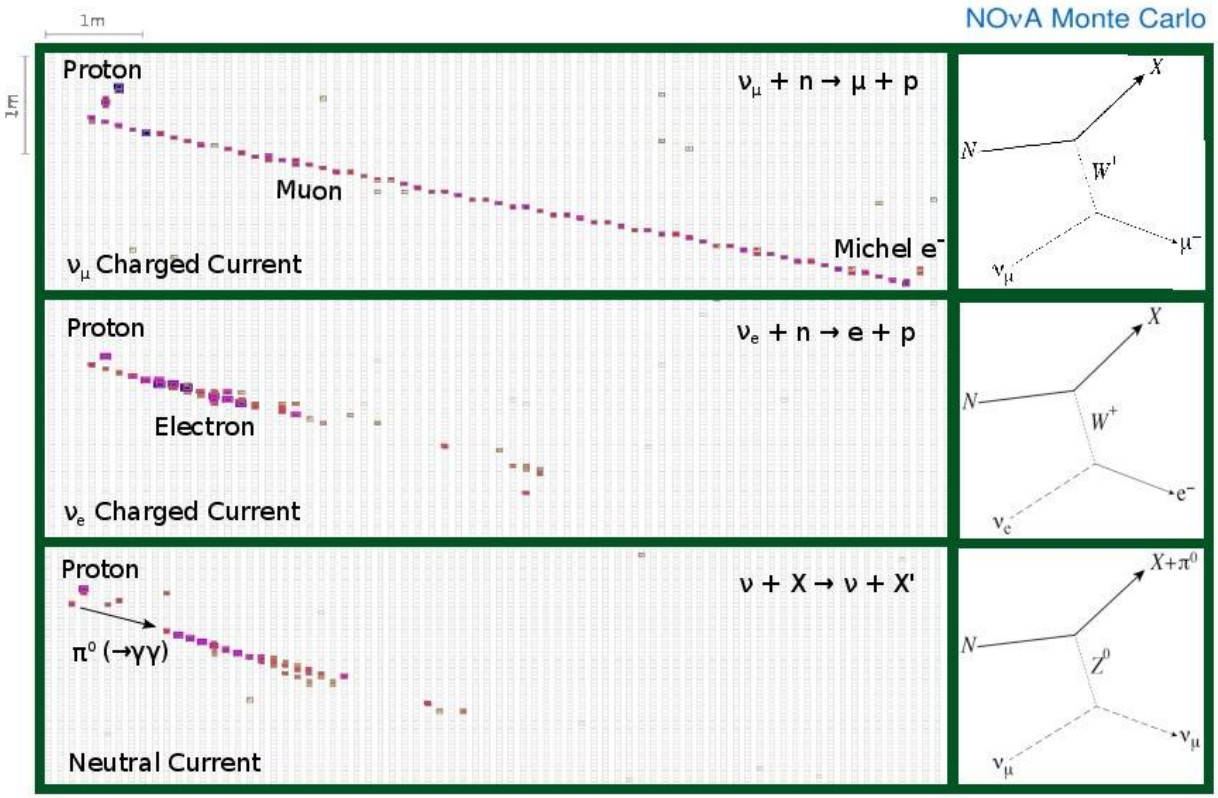
- At NOvA, data is taken in 550 μs intervals.
- Most of the activity is from cosmic rays
  - 100,000's cosmic rays/second
  - 100's ν_μ/year
  - 10's ν_e/year

Zooming in...



- Clustering in space and time with DBSCAN creates slices
  - Groups of hits likely to be causally related
  - Lets us separate neutrino events, cosmic rays, and noise
  - ...But it still doesn't tell use what each slice actually is – for that we need a classifier.

Event Topologies

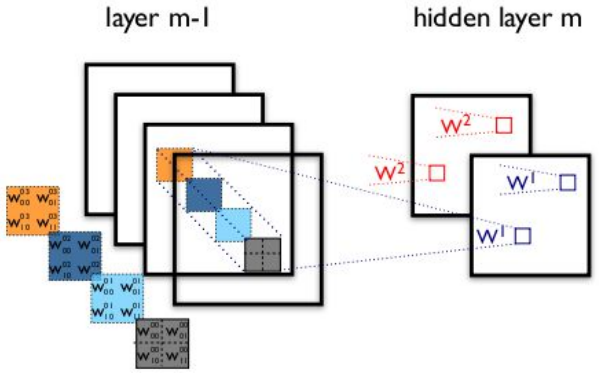


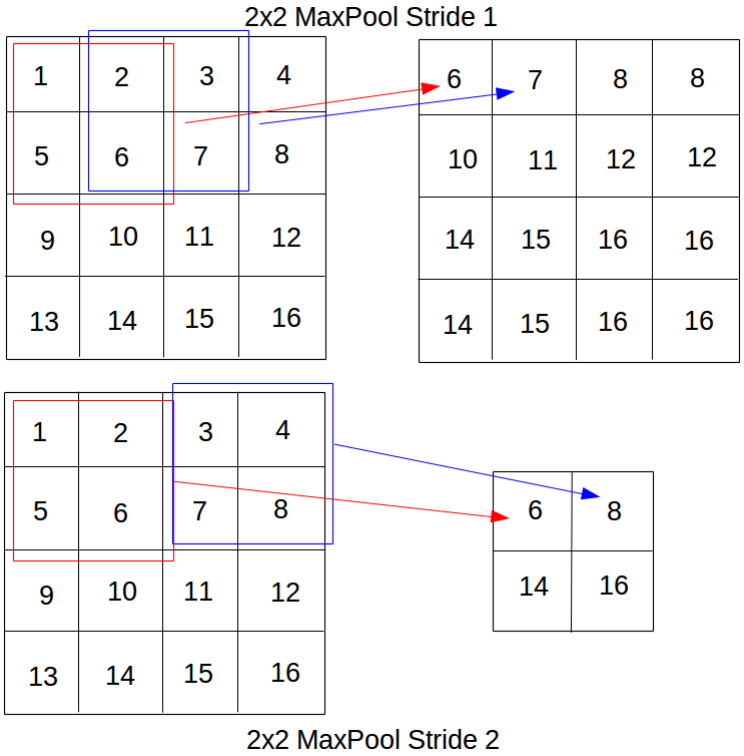
More complicated events can contain multiple charged pions make it more difficult to separate these event types.

- For the flagship analysis, the primary task is separating ν_e events from neutral current events.
- NOvA events already look like images
  - PVC cells = pixels
  - Charge deposited = color
- Try using computer vision techniques to classify events.

Convolutional Neural Networks

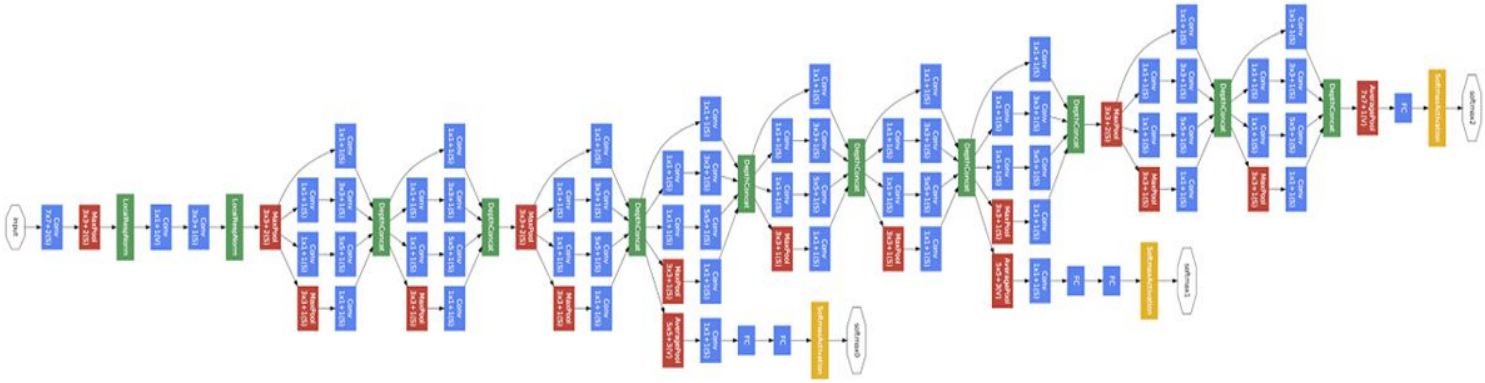
- Deep learning is a new paradigm that has caused a renaissance in the machine learning community.
  - Made possible by better activation functions, better weight initialization, and the advent of cheap GPUs.
- One variant – the convolutional neural network has been highly successful at image recognition tasks.
- Two basic type of layers:
  - Convolutional layers – apply discrete convolutions using learned kernels to extract features from the image.
  - Pooling layers – downsample the image and increase translational invariance in the final output.
- Stacked structure of convolutional and pooling layers extract increasingly abstract features from the input raw data encoding both local and global structure.
- Relatively new:
  - LeNet – one of the first (1998)
  - AlexNet – the one that started the revolution (2012)





GoogLeNet

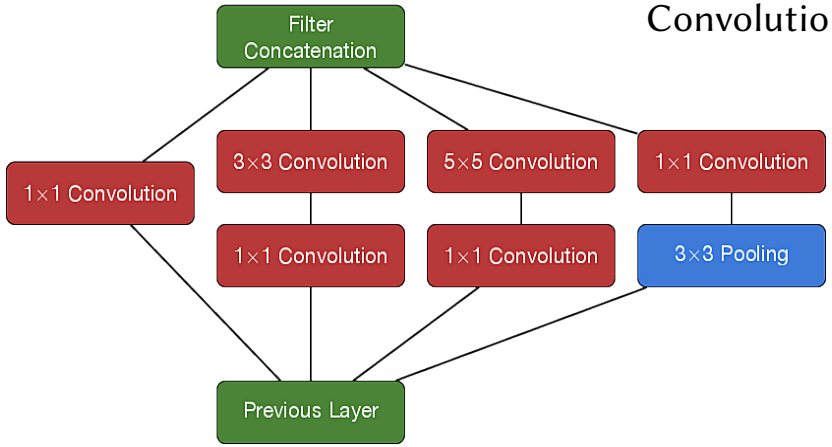
C. Szegedy, et. al., Going Deeper with Convolutions, arXiv:1409.4842 (2014)



- Due to the rise of relatively cheap GPUs, it has become possible to make increasingly complex network-in-network models.
- The GoogLeNet architecture is composed of a series of inception modules.
  - Outputs of the previous layer fans out to several convolutional layers with different kernel size.
  - Applies max pooling to downsample in feature map height and width and 1x1 convolutions to downsample the previous stack of feature maps into a smaller set of feature maps.
    - Designed to get maximum identification power out of as few operations as possible.
- In the ILSVRC 2014 image classification task, the correct classification was not one of the top 5 ranked categories out of 1,000 only 6.67% of the time.

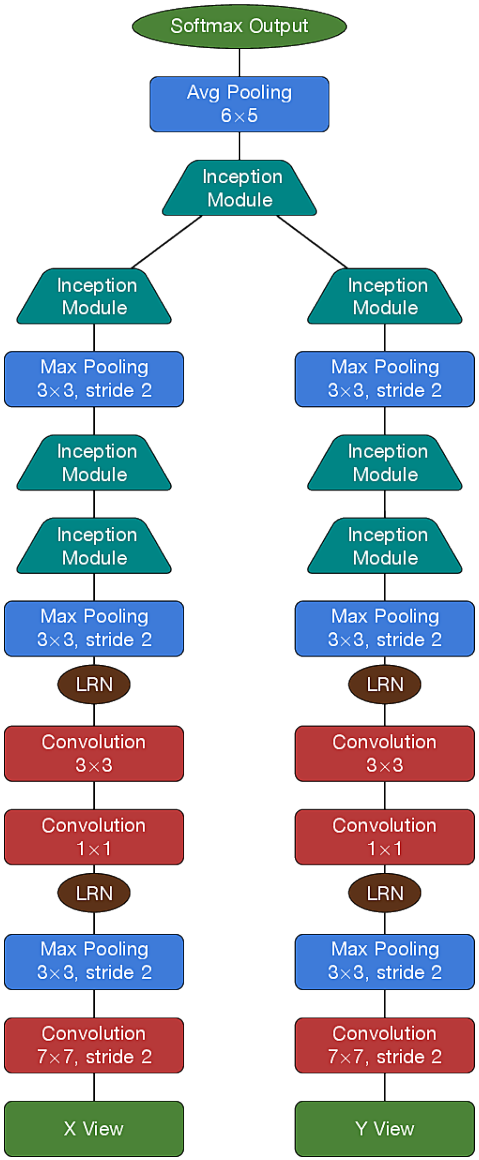
GoogLeNet

C. Szegedy, et. al., Going Deeper with Convolutions, arXiv:1409.4842 (2014)



- Due to the rise of relatively cheap GPUs, it has become possible to make increasingly complex network-in-network models.
- The GoogLeNet architecture is composed of a series of inception modules.
  - Outputs of the previous layer fans out to several convolutional layers with different kernel size.
  - Applies max pooling to downsample in feature map height and width and 1x1 convolutions to downsample the previous stack of feature maps into a smaller set of feature maps.
    - Designed to get maximum identification power out of as few operations as possible.
- In the ILSVRC 2014 image classification task, the correct classification was not one of the top 5 ranked categories out of 1,000 only 6.67% of the time.

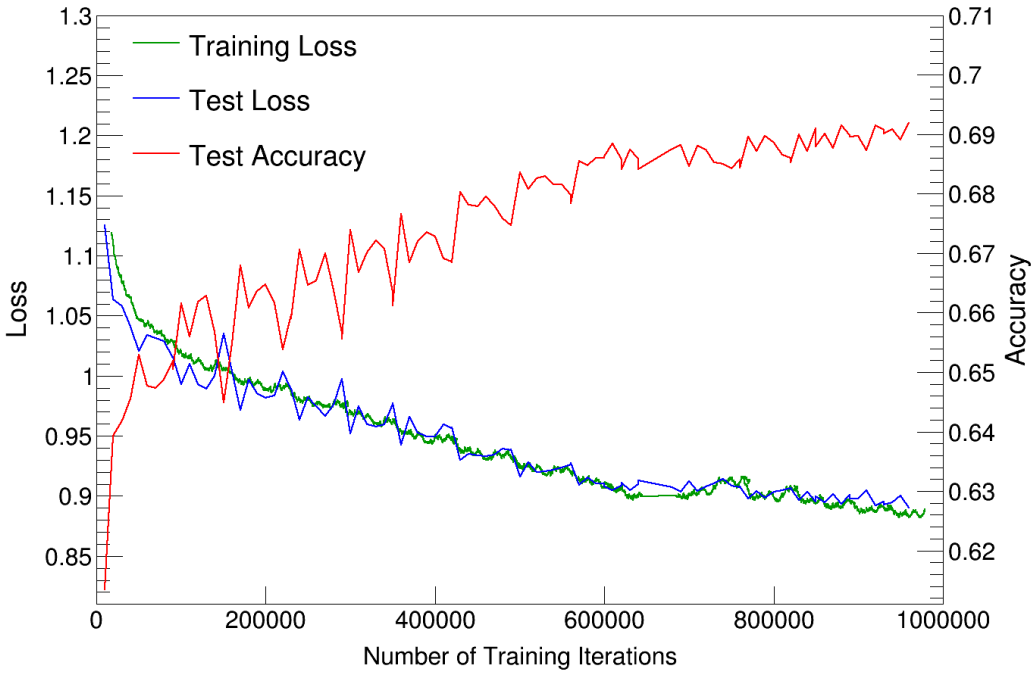
Our Architecture



- GoogLeNet showed the most promise in our early testing.
- Detector is composed of alternating horizontal and vertical planes.
  - Two views of the event- one from top and one from side
- Instead, create a "siamese" GoogLeNet variant.
  - Split the views early and double the architecture. Each parallel GoogLeNet learns separate features. These are merged together at the end before going through fully connected layers.
  - 1024 features are used in the final layer for classification.
- The architecture is a multi-classifier ⟶ attempts to categorize events as $\{\nu_\mu, \nu_e, \nu_\tau\}$ x {QE,RES,DIS} or cosmic rays.
  - QE (quasi-elastic): Incoming neutrino interacts with a single nucleon. Results in an out-going lepton + a recoil nucleon
  - RES (resonance): Interaction of the neutrino excites a Δ resonance, often leading to a pion in the final state (in addition to the lepton and nucleon).
  - DIS (deep inelastic scattering): The interaction causes fragmentation of the nucleus.
  - These are approximately in order of increasing complexity, though final state interactions make this more complicated.
- In principle, this architecture a universal classifier (rather than ν_e only).

Caffe

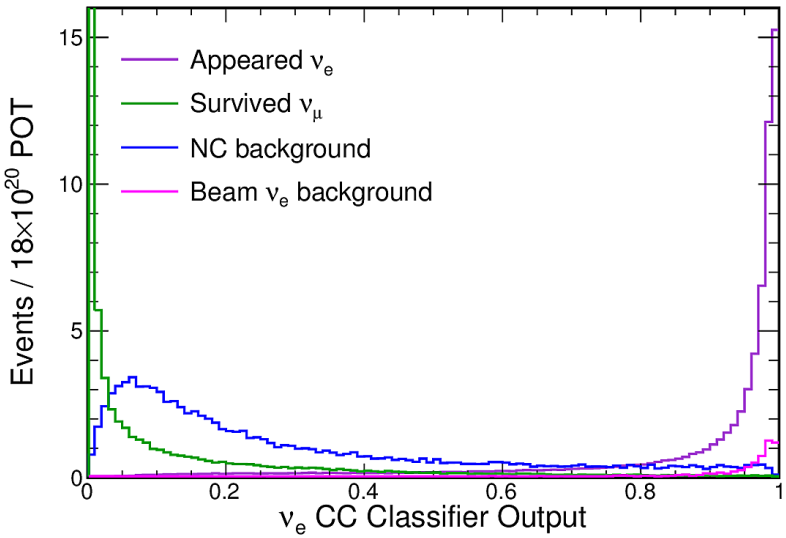
- Caffe (caffe.berkeleyvision.org) is an open source deep learning framework developed by the Berkeley Vision and Learning Center.
- Comes pre-packaged with a large variety of layer types.
- Comes with a variety of models produced by different computer vision groups for image classification contests.
- Able to run on GPUs without any extra effort (only change one line in the configuration file)
- C++ API made it easy to integrate with NOvA's *art* framework.
  - Training on GPUs.
  - Production-time evaluation on the grid on CPUs.

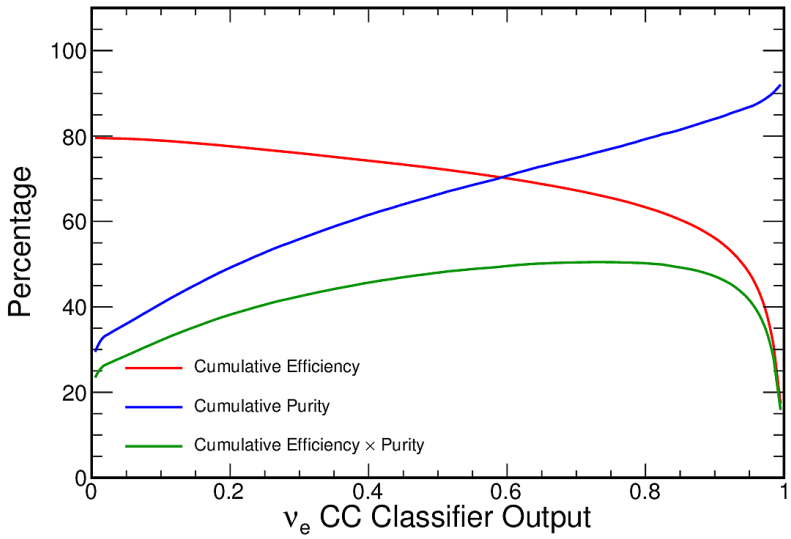


Trained on 2 k40 GPUs on Fermilab's Wilson cluster for ~1 week.

Y. Jia et. al., Caffe: Convolutional Architecture for Fast Feature Embedding, arXiv:1408.5093 (2014)

Performance as a ν_e Selector

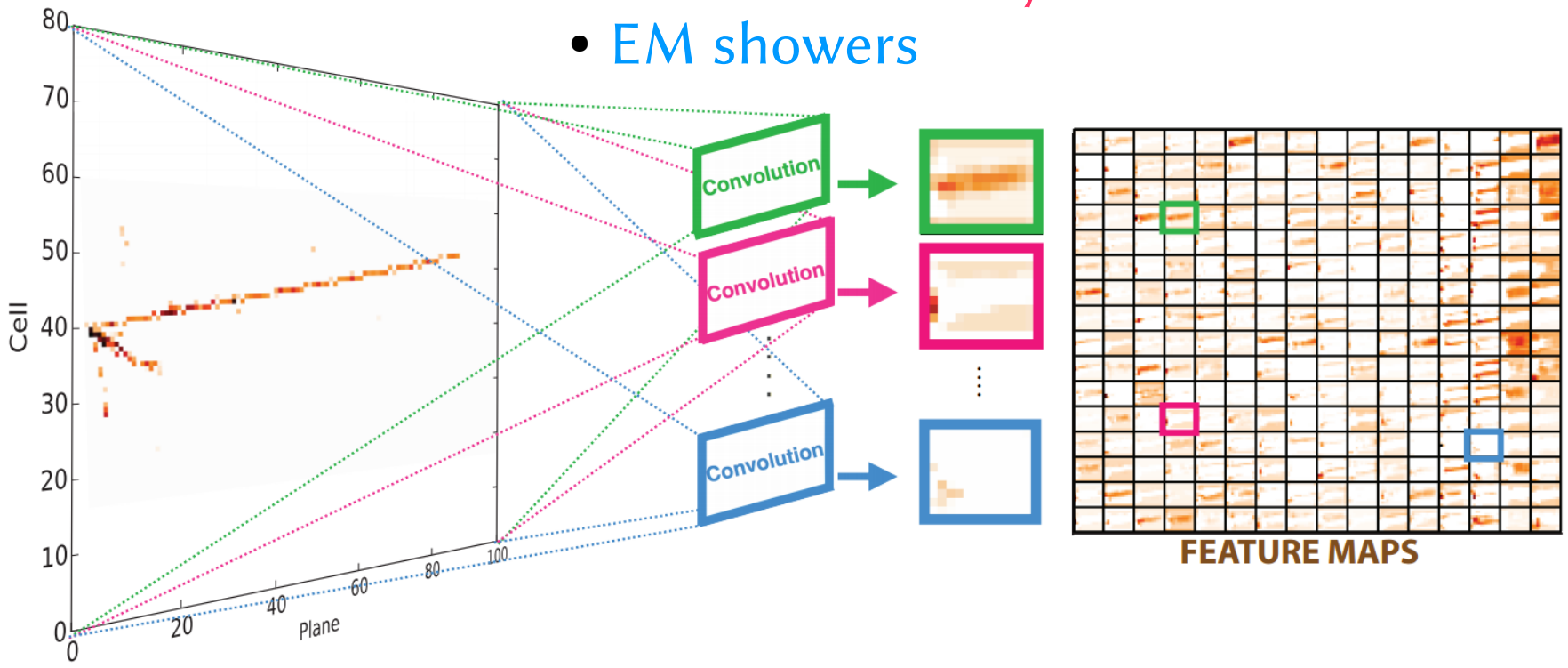




- Sum over all ν_e interaction types to create a ν_e selector.
- Evaluate on a statistically independent realistic MC sample.
- Weight by the simulated NOvA flux and neutrino oscillation probabilities using global best fits of oscillation parameters.
- Optimize selection for discovery using FOM S/sqrt(B)
- Using this optimization, CVN achieves an FOM increase relative to previous selectors equivalent to collecting 30% more data.
- Primarily due to improved efficiency in selecting resonance and deep inelastic scattering interactions.

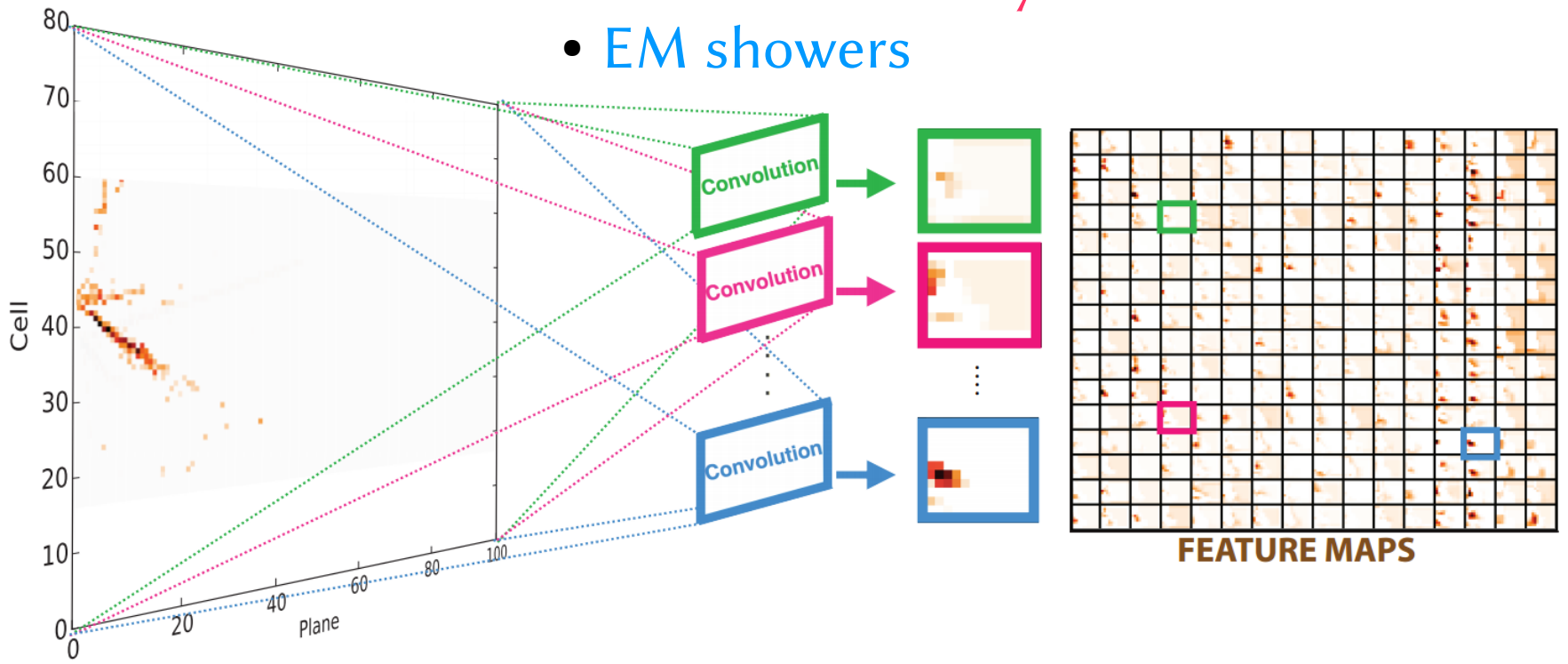
Understanding the Network: Intermediate Features

- Example simulated ν_μ
- Feature maps after the first inception module
- Highlighted maps seem to pick out:
  - Muon-like tracks
  - Hadronic activity
  - EM showers



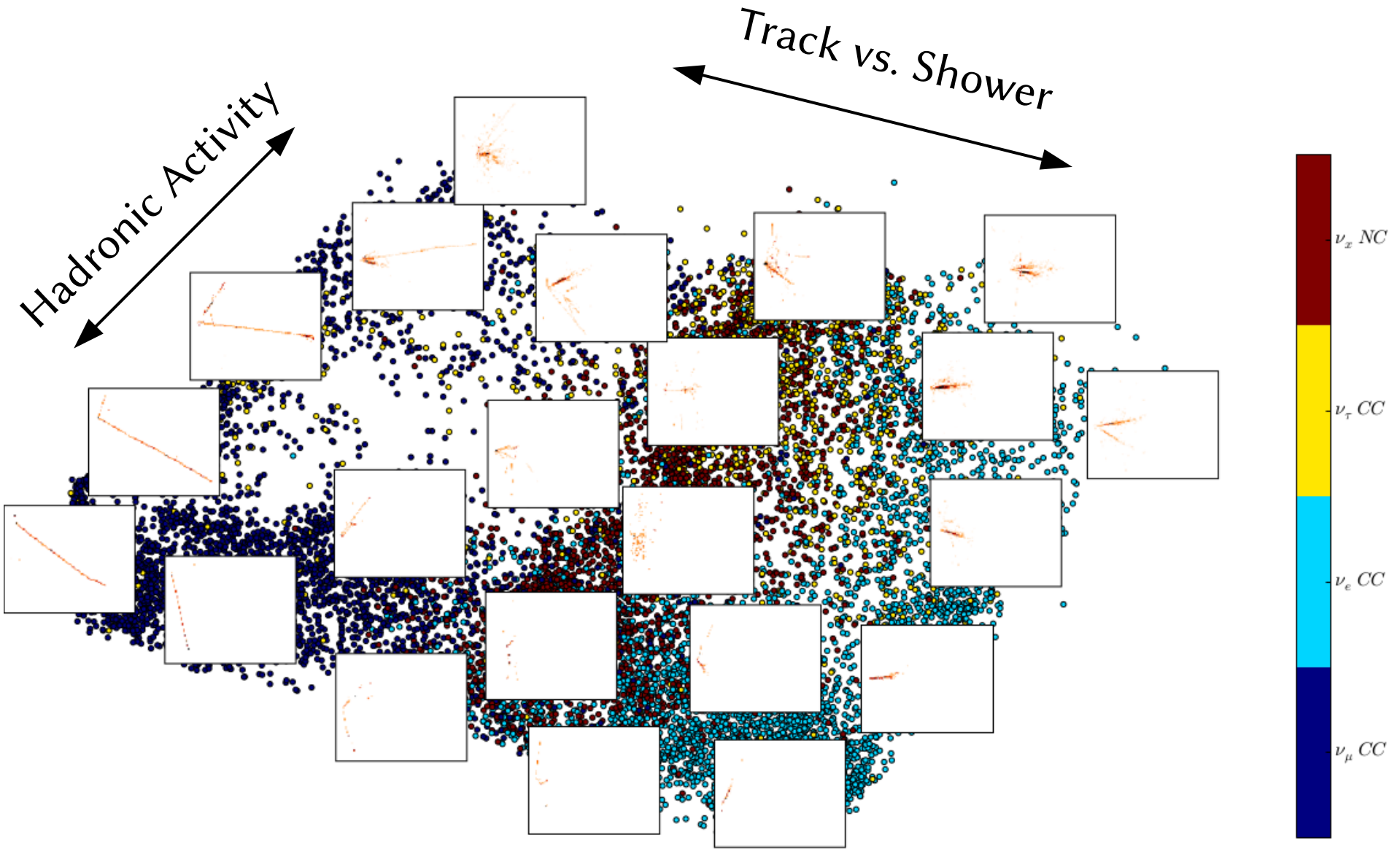
Understanding the Network: Intermediate Features

- Example simulated ν_e
- Feature maps after the first inception module
- Highlighted maps seem to pick out:
  - Muon-like tracks
  - Hadronic activity
  - EM showers



Understanding the Network: Feature Embedding with t-SNE

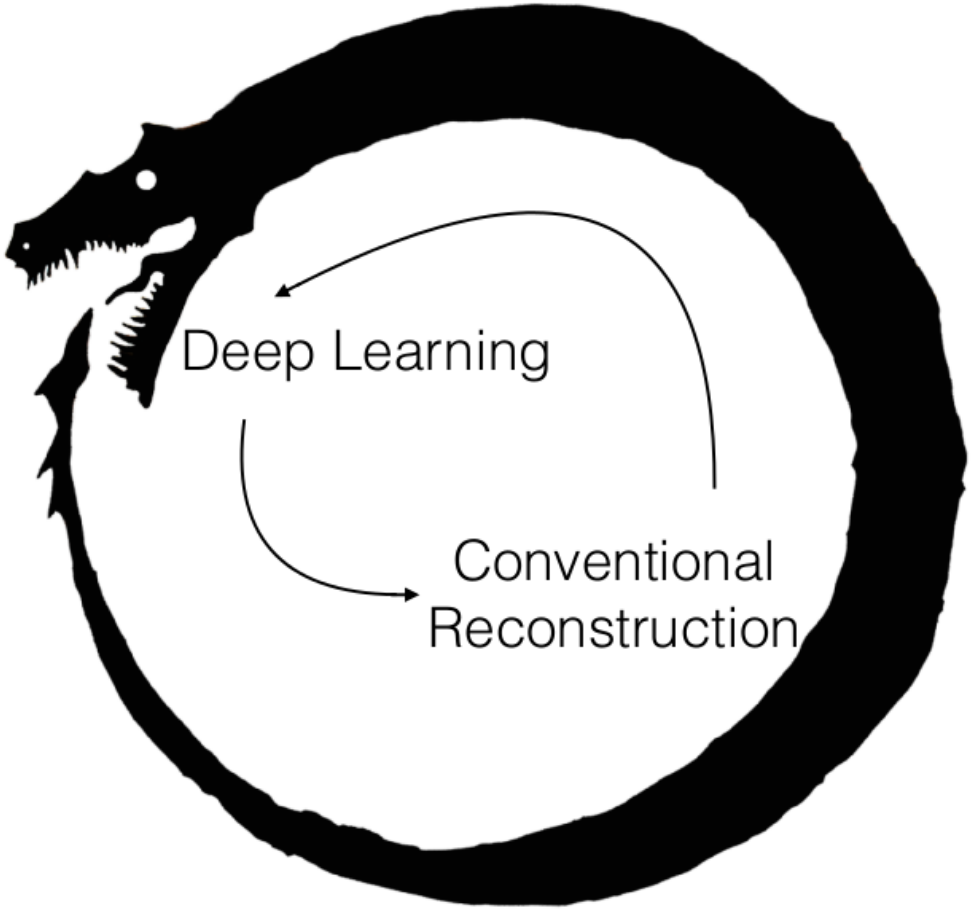
Understanding the Network: Feature Embedding with t-SNE



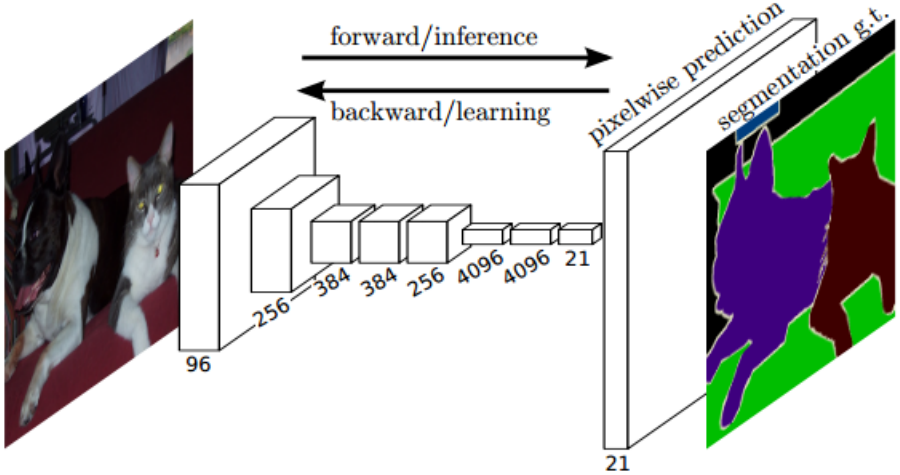
Generalizing the Features?

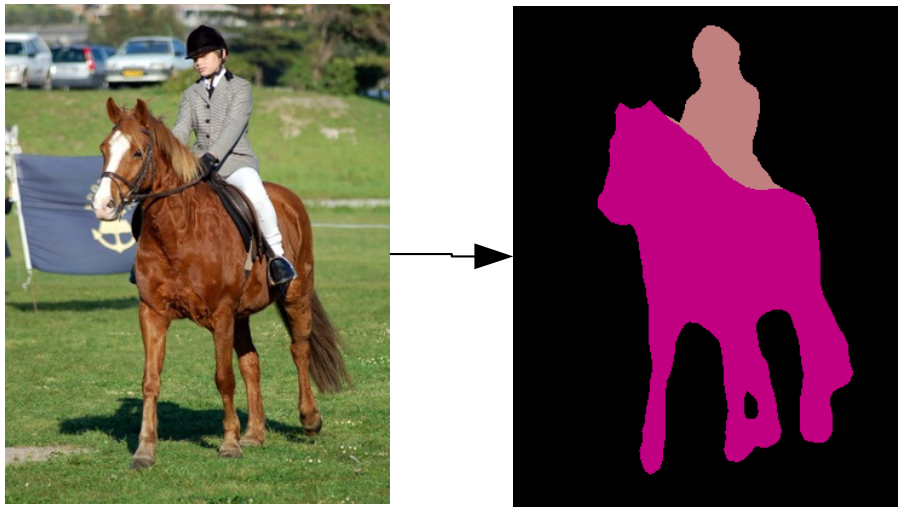
- From the t-SNE embedding, it's clear that the final 1024 features encode fairly general information about the neutrino interaction.
- Can these features be reused for other tasks?
  - The full network is somewhat slow – evaluating once upstream and reusing the features in shallow methods could provide time savings.
  - Essentially transfer learning.
  - Currently in testing in NOvA for cross-section analyses that want to select final states not in the original list of categories.

The Future?



Semantic Segmentation





arXiv:1411:4038

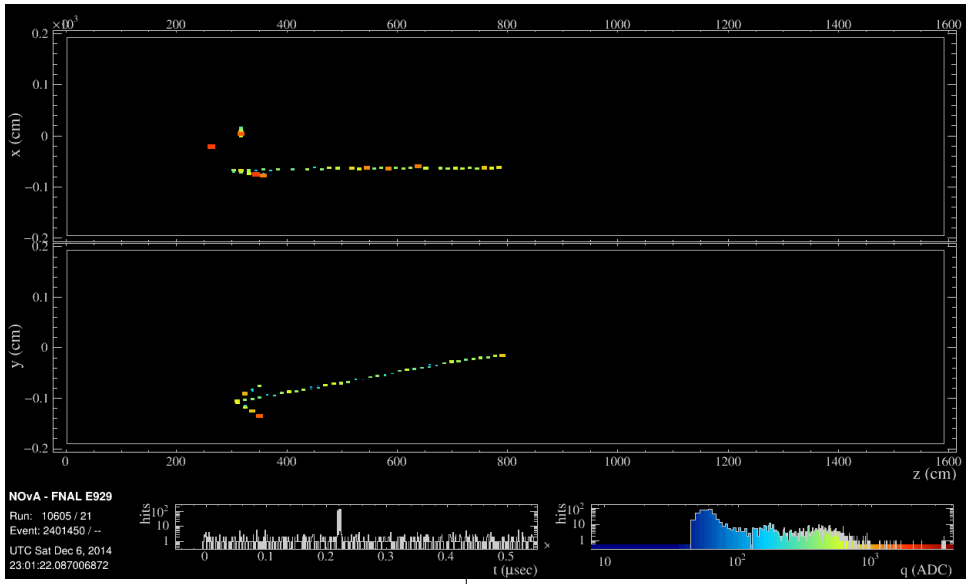
- Semantic segmentation refers to the process of labeling pixels according to what object they are a part of.
- Cutting edge research has demonstrated that this is possible using convolutional neural networks.
  - Extract information from all pixels across all layers corresponding to the pixel of interest (a hyper column of pixels)
- This could potentially turn reconstruction on its head
  - Instead of reconstructing objects and then identifying them, label all pixels according to what they likely came from first, and then use that information to assist reconstruction.
  - Instead of clustering in time and space, cluster in time, space, and PID.
  - Could potentially make it easier to reconstruct neutron interactions that are not well connected to the event vertex.

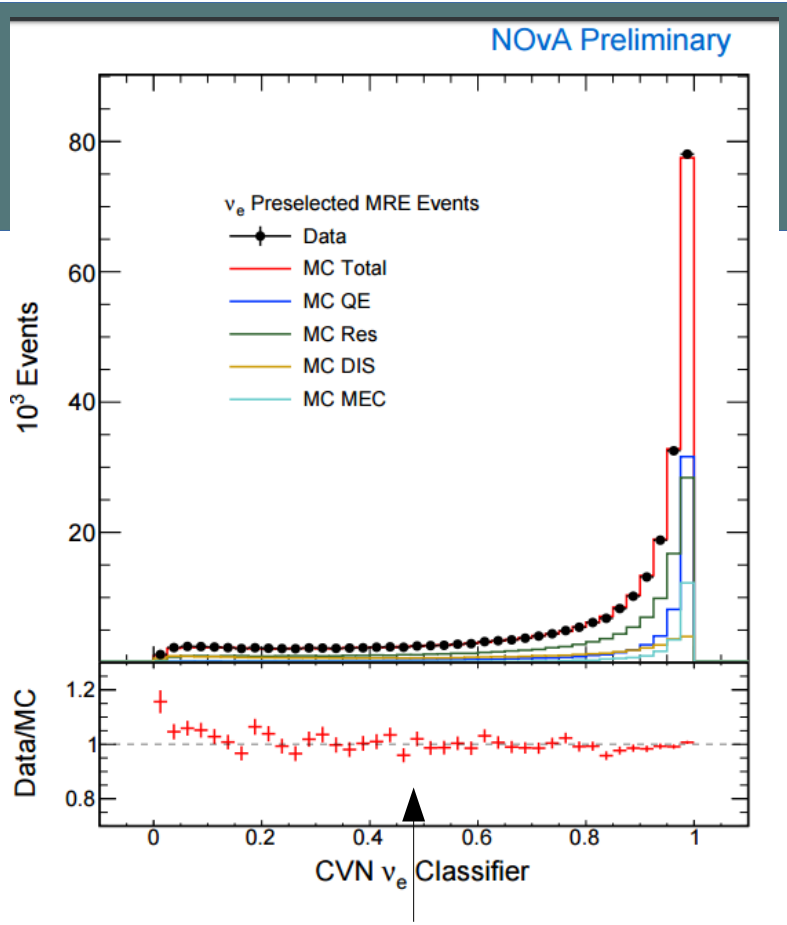
Conclusions

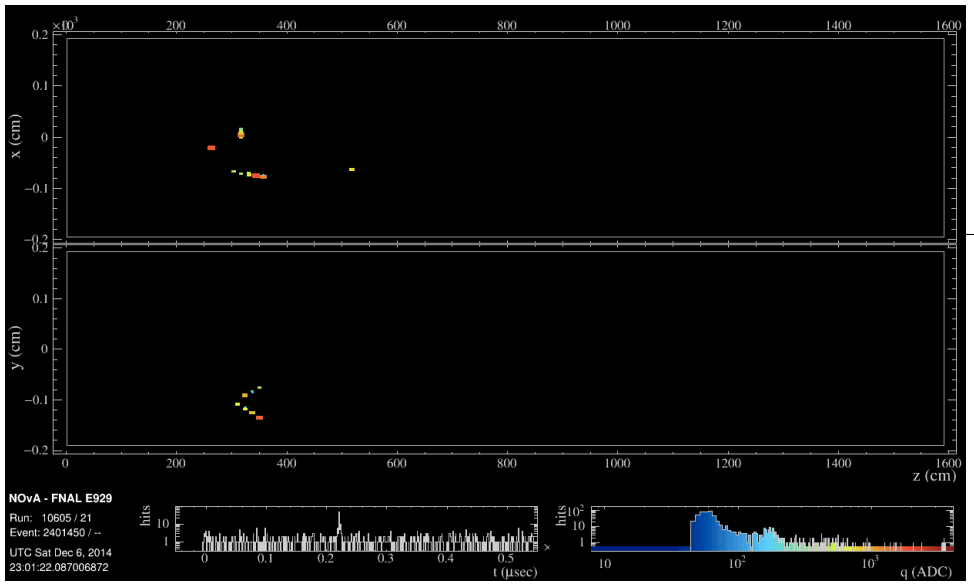
- Modern deep learning techniques are very powerful.
- Using a modified GoogLeNet architecture, it is possible to build and train a neutrino event classifier that can achieve excellent signal and background separation for the ν_e appearance analysis.
  - Uses minimal reconstruction
  - Equivalent to a 30% increase in statistics.
- GPU acceleration makes it possible to train complex networks in ~1 week.
- This technique is directly applicable to a number of analyses using other flavors or interaction types.
- We've only scratched the surface of what might be possible with deep learning.

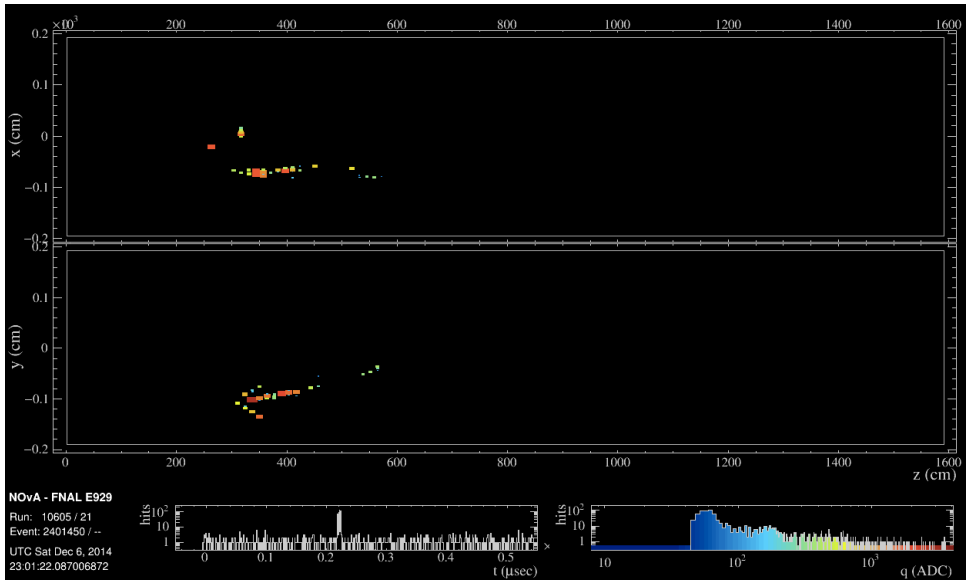
Backup

Muon removed, electron added

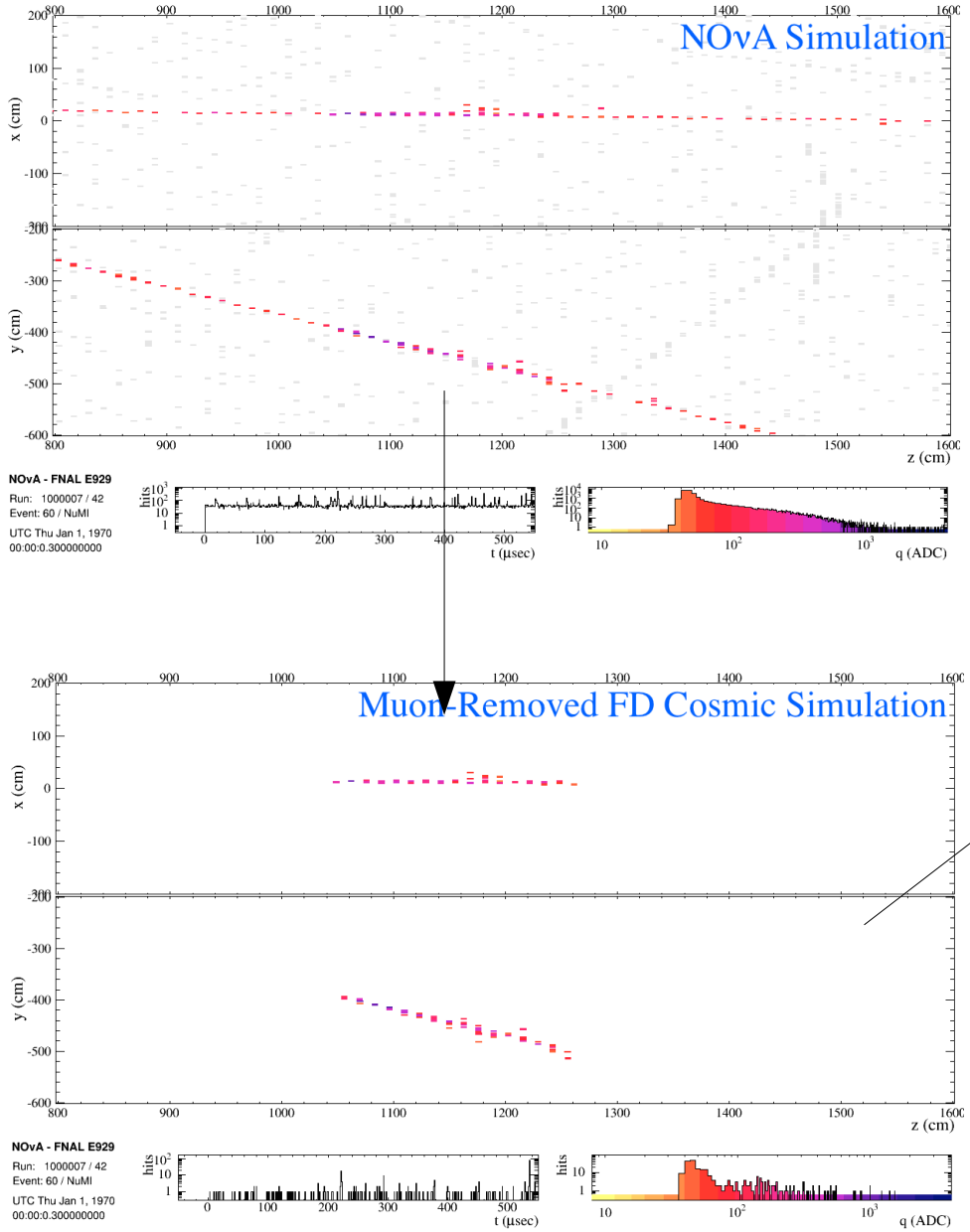


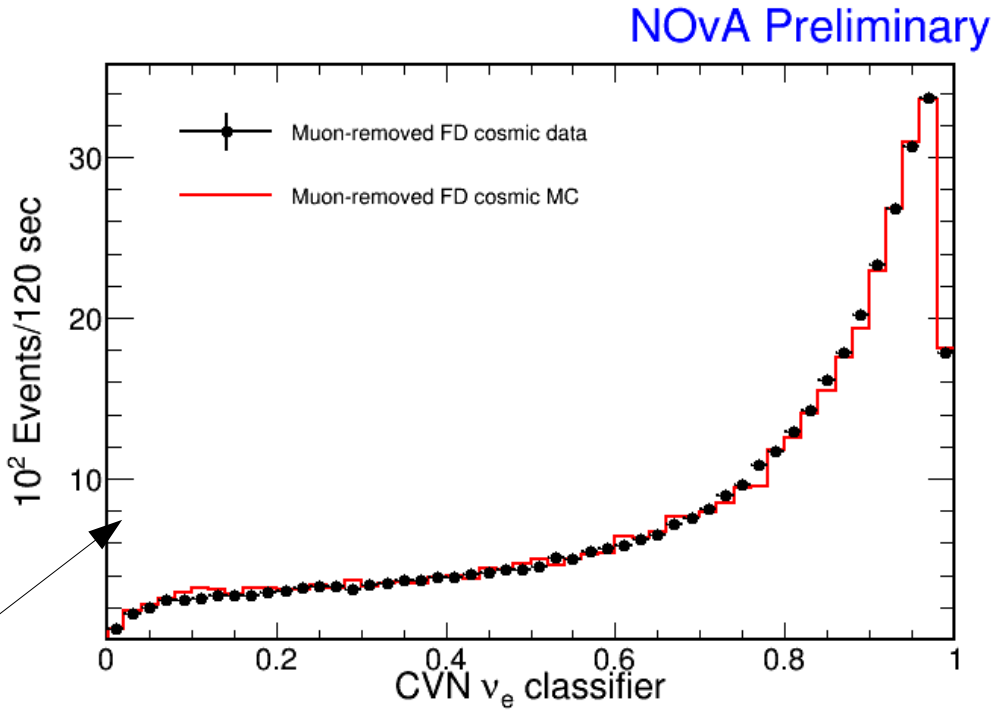




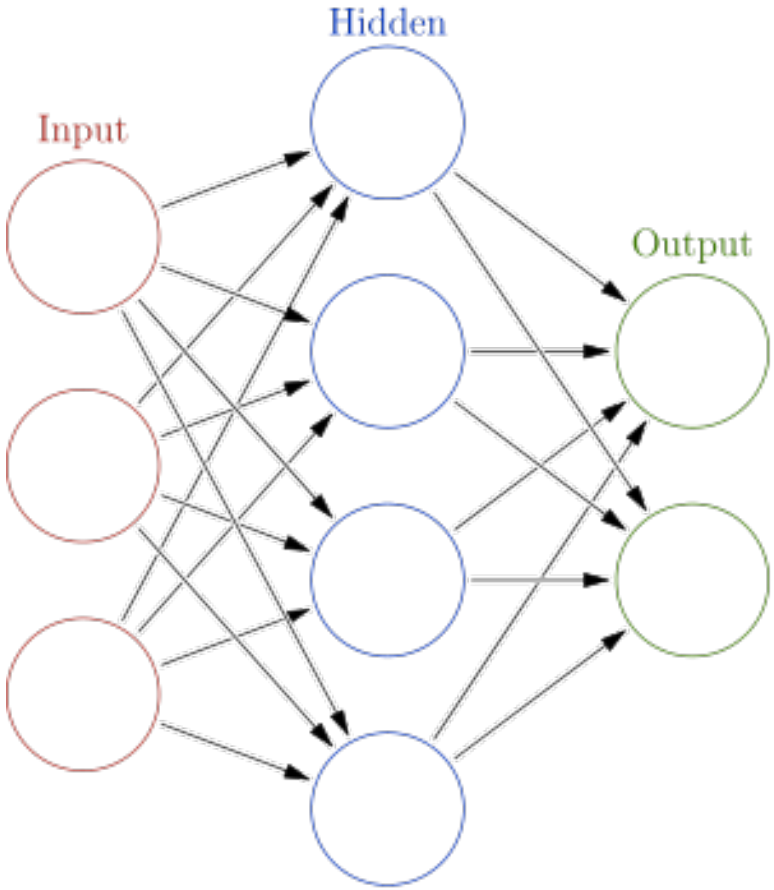


Muon removed cosmic brem: EM shower response



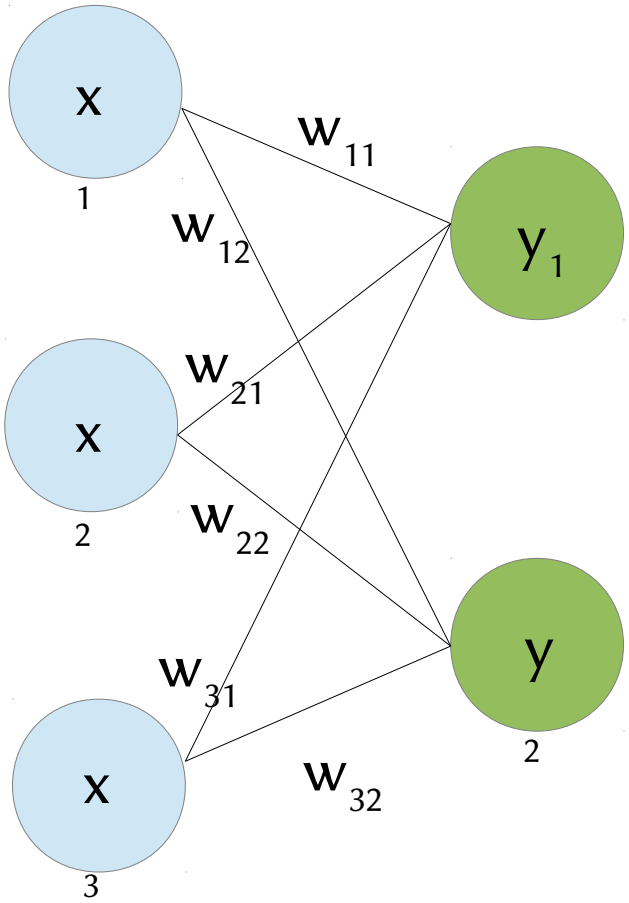


Traditional Neural Networks



- Traditional neural nets are powerful machine learning tools in widespread use through HEP.
- Nodes organized in layer.
- Each node performs a weighted sum on the output of all nodes in the previous layer, and the result is pushed through a non-linear function.
- Optimize the weights using an iterative learning procedure.

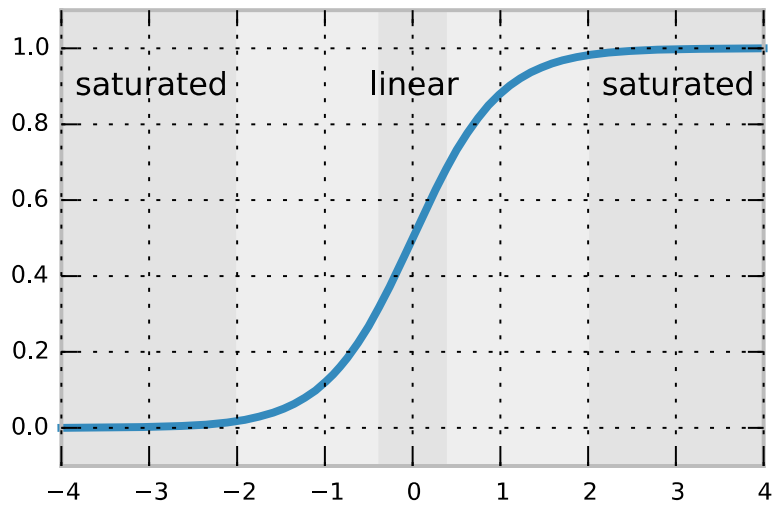
Simplified Example



$$y_1 = \sigma(w_{11}{}^*x_1 + w_{21}{}^*x_2 + w_{31}{}^*x_3 + b_1)$$
$$y_2 = \sigma(w_{12}{}^*x_1 + w_{22}{}^*x_2 + w_{32}{}^*x_3 + b_2)$$

where w_{ij} are tunable weights and b_i are tunable biases.

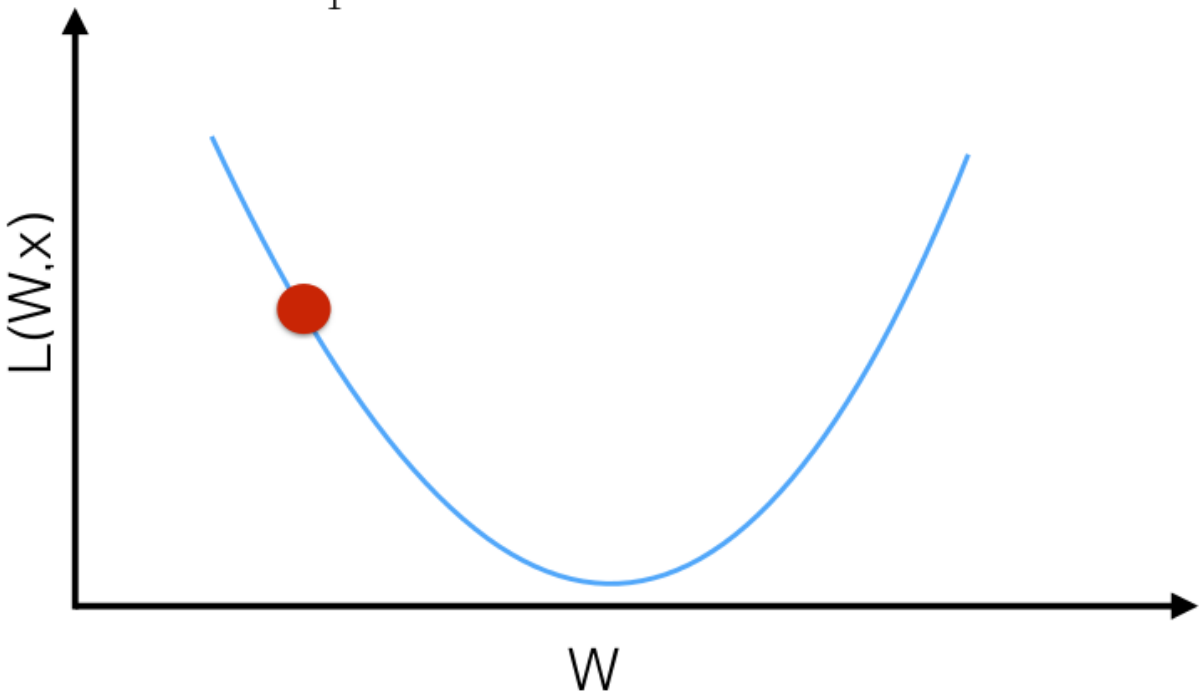
$\sigma =$ Sigmoid



Loss Function

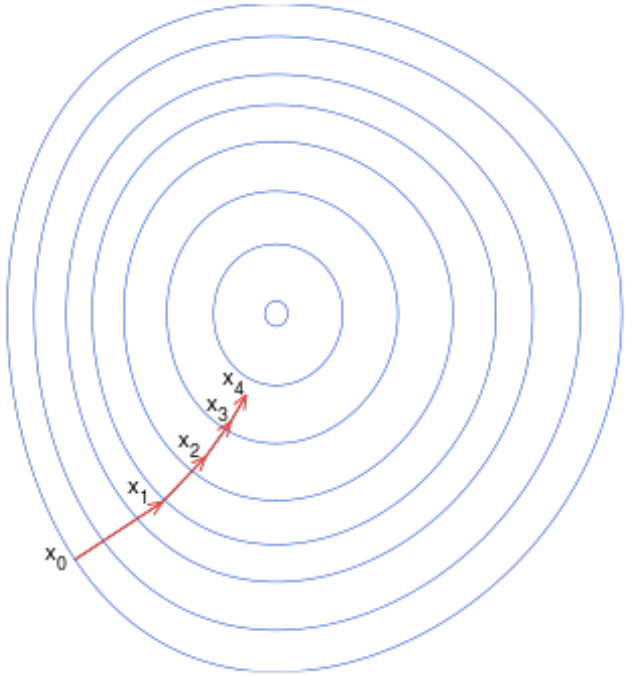
- To optimize the weights, we first need an objective function.
  - Usually called the loss.
- Many options for the definition of the loss, depending on the problem.
  - Cross-entropy loss, shown here, is useful for classification problems.

$$L(W,X) = \frac{1}{N} \sum_{1}^{N_{examples}} -y_i \log\left(f(x_i)\right) - (1-y_i)\log\left(1-f(x_i)\right)$$



Training

- Minimize the loss function using gradient descent.
  - Calculate the loss for a batch of labeled examples.
  - Use back propagation to calculate the gradient.
    - Effectively just the chain rule.
  - Update the weights according to:



$$w'_j = w_j - \alpha \nabla_{w_j} L$$

learning rate

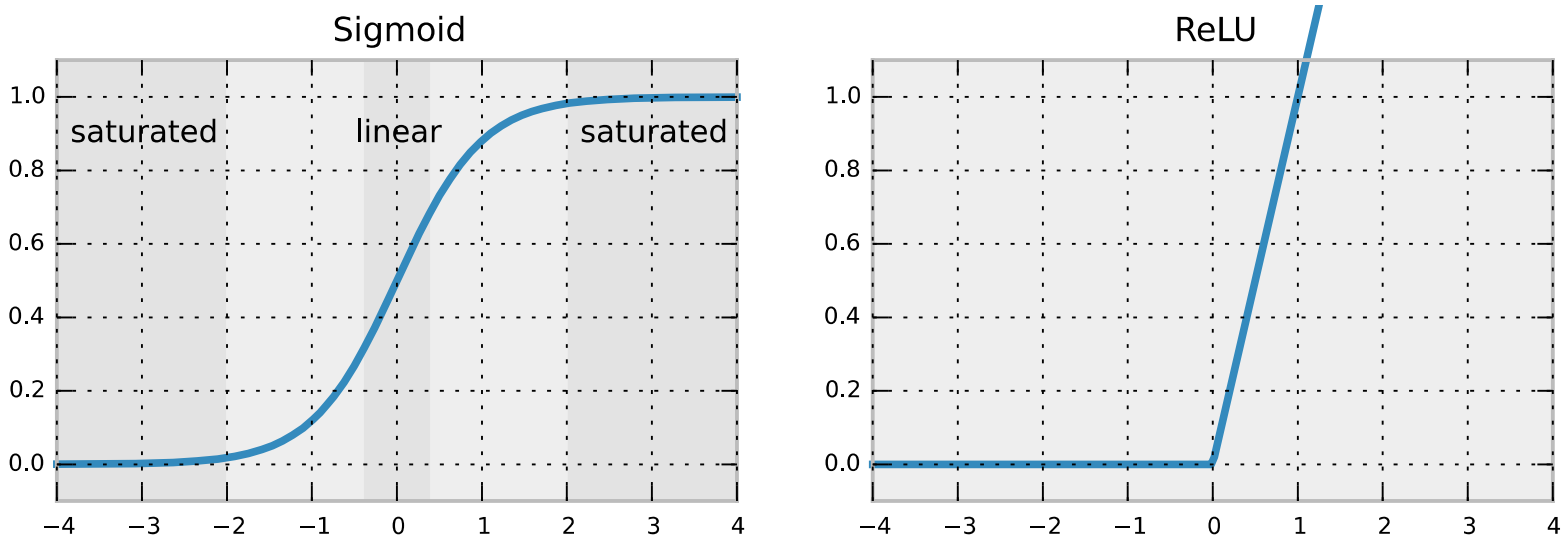
gradient

Why Isn't this Good Enough?

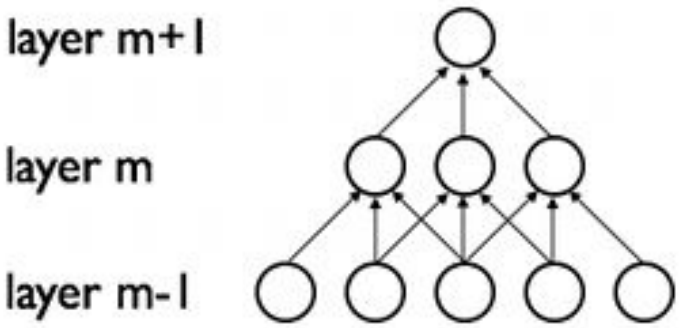
- In theory, a single layer network with a sufficient number of node can approximate most functions to arbitrary precision.
- Multi-layer networks can often approximate a function with fewer nodes than a single layer network.
  - These networks become very difficult to train.
  - Difficulty training deep networks led to neural nets falling out of favor.
- Due to the fully connected nature of traditional neural nets, the number of free parameters increases sharply with additional nodes.

What Made Deep Networks Possible?

- Biggest problem with deep networks is the difficulty in training them.
- Several improvements made this possible:
  - Better initialization of weights
  - Better non-linearities
    - Sigmoids lead to the vanishing gradient problem making training slow
    - ReLU allow the gradient to propagate back without vanishing
  - Advent of cheap GPUs



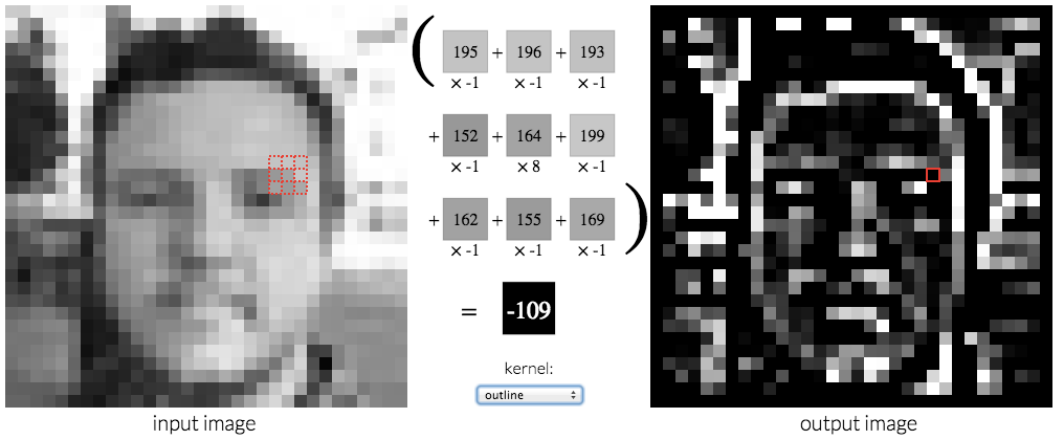
Convolutional Neural Networks



http://deeplearning.net/tutorial/lenet.html

- Convolutional neural nets are a very successful deep learning method.
- Inspired by research showing that the cells in the visual cortex are only responsive to small portions of the visual field - “receptive field”.
- Some cells collect information from small patches – sensitive to edge-like features.
- Other cells collect information from large patches.
- Effectively, these cells are applying convolutional kernels across the visual field.

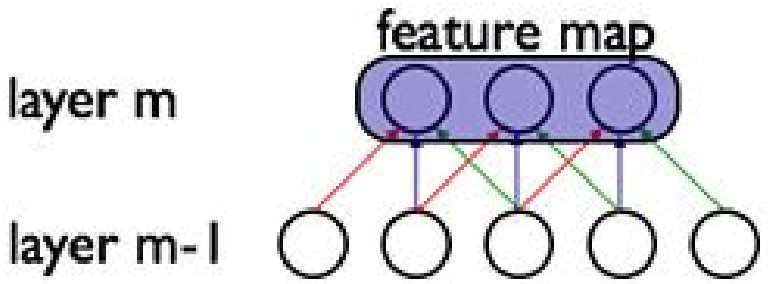
Convolutional Kernels

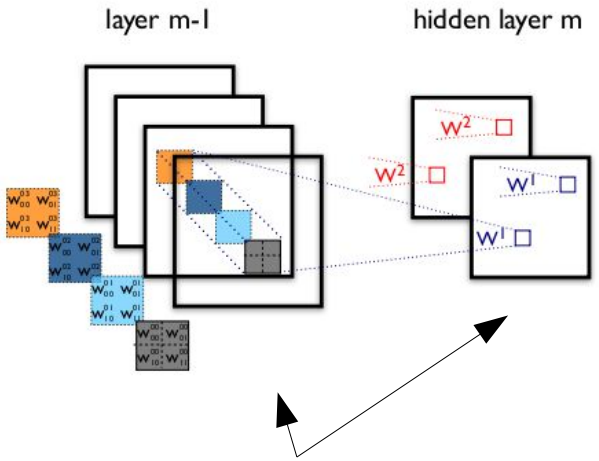


- Convolutional kernels are well known in computer graphics.
- Kernels transform images.
  - The one above outlines objects in the image.
- Many common kernels exist, but it we want to learn optimal kernels directly from the data.

http://setosa.io/ev/image-kernels/

Convolutional Layers



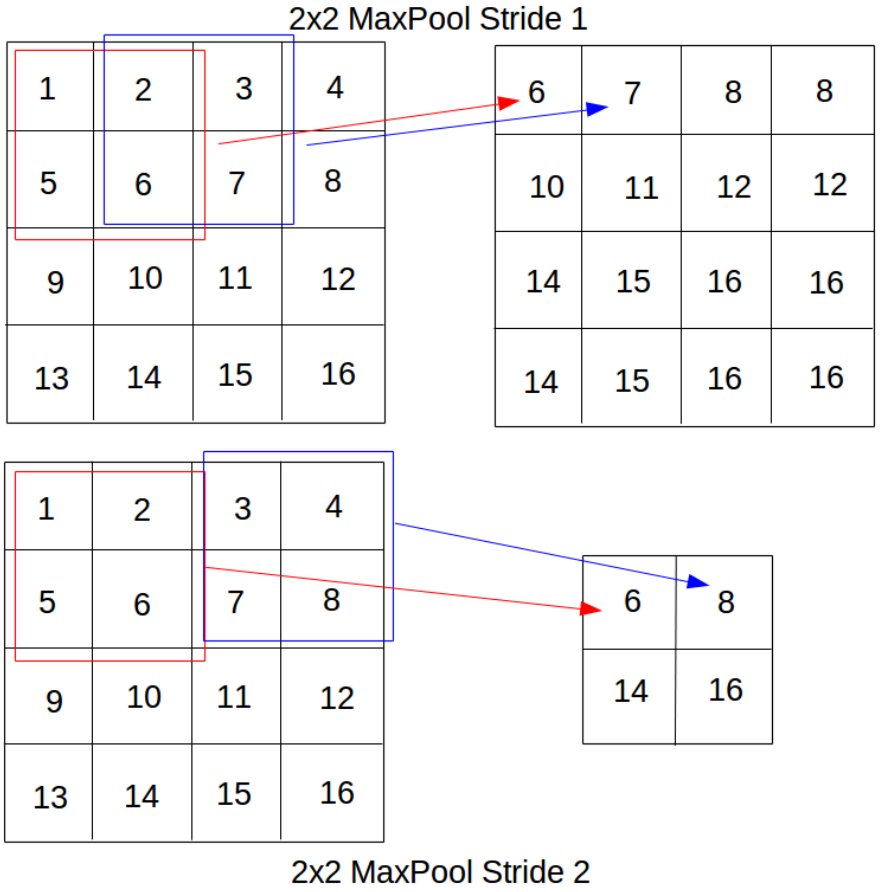


Each pixel is the result of a tensor dot product of the weights with a tower of patches in the incoming feature maps

- Each kernel we create stays the same as we apply it across the image.
  - Weight sharing reduces the number of free parameters, lowering the risk of overtraining.
- Each convolutional layer trains an array of kernels which produce corresponding feature maps.
- Weights going from layer to the next are a 4D tensor of NxMxHxW
  - N is number of incoming feature maps
  - M is the number of outgoing feature maps
  - H and W are the height and width of the outgoing convolutional kernels.
- The next layer applies kernels to combine the information in a receptive field across feature maps in the previous layer to create new feature maps.

http://deeplearning.net/tutorial/lenet.html

Pooling Layers

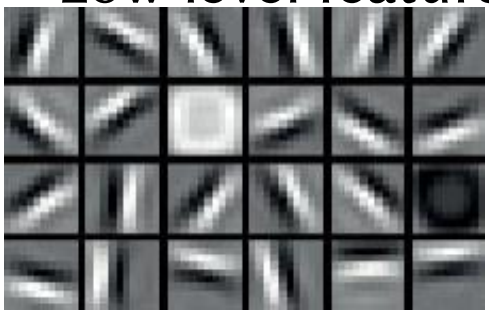


- Pooling is a technique to down sample information.
  - Output pixel is either maximum value of a patch of input pixels (max pooling) or the average (average pooling).
- Can be thought of as a type of smoothing to remove less significant information.
- Can either be strided or unstrided
  - Controls how much information is lost
- Number of output feature maps is the same as the number of input maps.

Feature Extraction

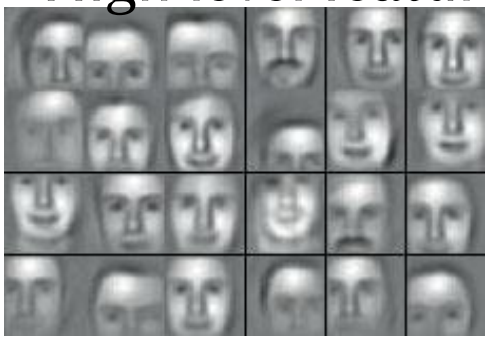
Raw input

Low level features



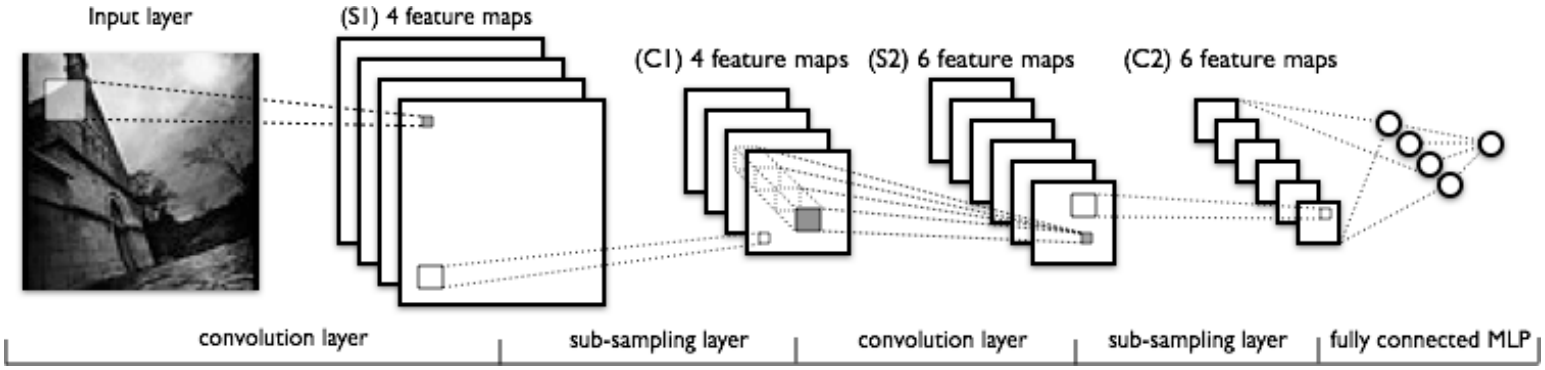
Mid level features

High level features



- Convolutional layers learn about local structure in the image.
- Layers deeper in the network combine features extracted from many small patches.
- Each layer extracts increasingly complex features from the input image.

LeNet-5 Model



- CNNs have existed for a while now, but they've only recently become easy to use.
- One early example was the LeNet architecture.
- Composed of alternating convolutional and max pooling layers that ends in a fully connected MLP.
- Max pooling partitions the feature map into non-overlapping rectangles and downsamples by only keeping the maximum value contained in each.
- Max pooling + convolutional layers add a degree of translational invariance to the net.

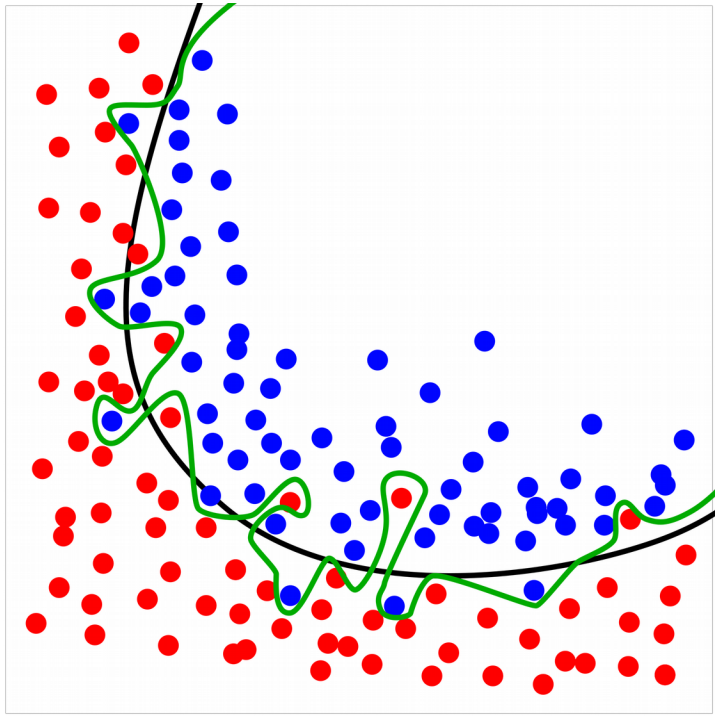
Y. LeCun, L. Bottou, P. Haffner, Proceedings on the IEEE, 86(11), 2278-2324, (1998d)

What is Overtraining?

- Sometimes called a failure to generalize
  - Networks contain large numbers of parameters – sometimes they learn how to classify the training data exactly at the expense of generalizing well to new data.
  - Can be seen if the evaluation of testing data begins to diverge from that of training data.
- Convolutional neural networks tend to already be more robust due to having fewer trainable parameters compared to fully connected networks.
- In our case, our training data is entire synthetic
  - Must also make sure we generalize from simulation to data.
- Techniques use to prevent overtraining
  - Early stopping
    - Hard to make rigorous – will not use with CVN.
  - Regularization
  - Dropout
  - Data Augmentation

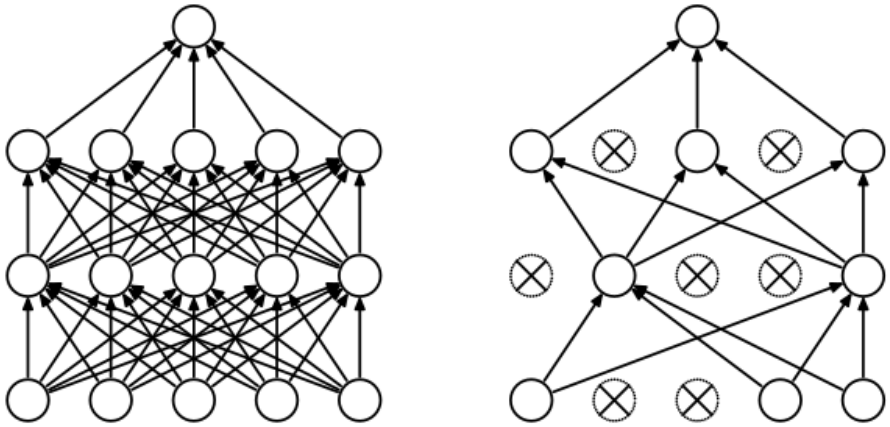
Regularization

- Add a term to the loss of the form: $\frac{1}{2}\lambda\sum w_i^2$
- Decrease the number of effective free parameters.
- Prevents any weight from being too large unless there is strong evidence that it needs to be.
  - Makes it difficult for the network to finely tune weights to perfectly categorize training examples.



Moody, J., et al. "A simple weight decay can improve generalization." Advances in neural information processing systems 4 (1995): 950-957.

Dropout

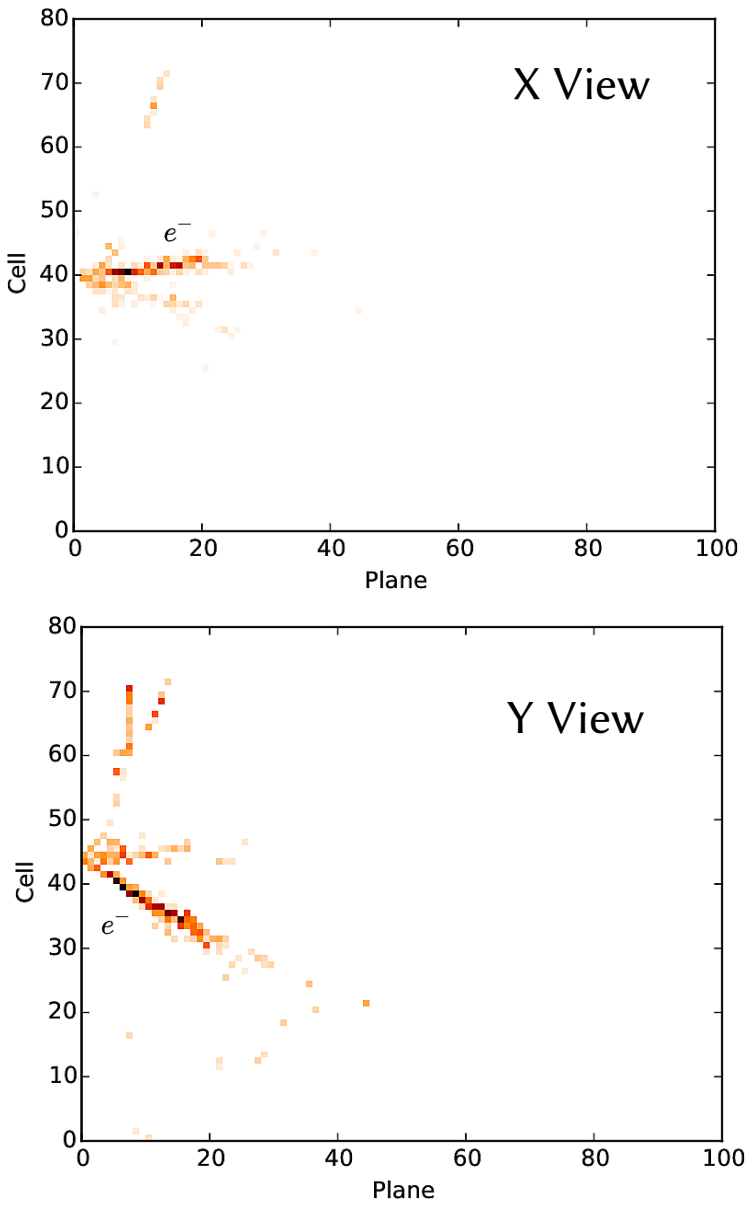


(a) Standard Neural Net (b) After applying dropout.

- In the fully connected layer in CVN, we apply the dropout technique.
  - At each iteration, randomly set 40% of weights to zero and scale the rest up by 1/(1 – 0.4).
  - Since no weight is reliably in use with any other weight, weights can not be strongly correlated.
  - Preventing weight co-adaptation strongly promotes generalization.
  - Can be thought of as an ensemble of smaller networks.

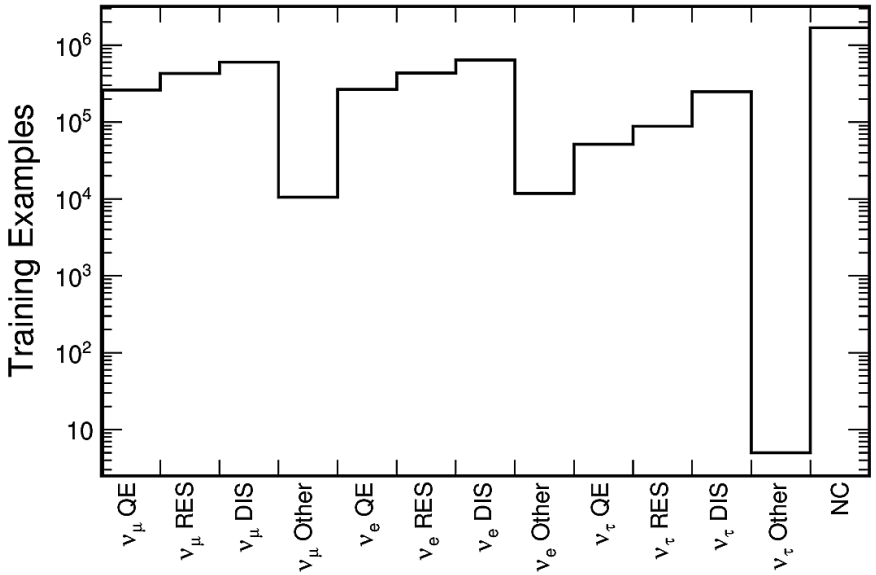
Srivasta et al, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", JMLR 2014

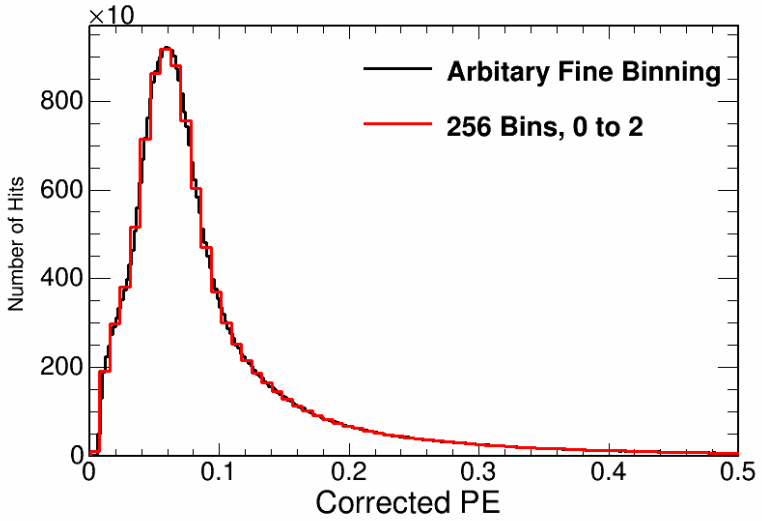
Constructing Input Images



- For our input, we construct "pixel maps" from hits in a slice.
  - No reconstruction other than initial hit clustering.
  - All slices with at least 15 distinct hits were used in training.
- The X view is composed of all planes with vertical cells
  - A projection on the x,z plane
- The Y view is composed of all planes with horizontal cells.
  - A projection on the y,z plane
- We take all hits in a 100 plane by 80 cell box for each view
  - ~14.52 m deep and ~4.18 m wide

Building Training and Testing Datasets





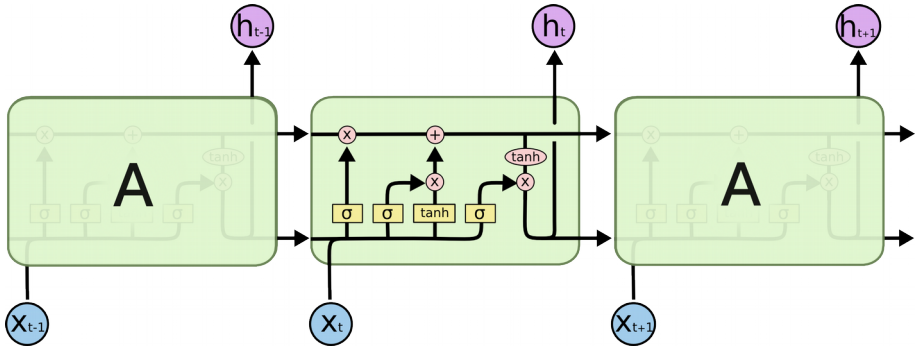
Take existing simulations containing ν_μ, ν_e, and ν_τ events.

Construct a pixel array pixels[c][h][w] where c indexes the view, h indexes cells, and w indexes planes from pixel maps.

Push into LevelDB databases: 80% for training and 20% for testing.

- Rescale the corrected hit energy to go from 0 – 255 and recast to 8 bits to save space.
- Almost all hits have exactly zero energy (suppressed in these plots).
- Distribution very different from natural images.

Recurrent Neural Networks



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

- Neural networks with loops in them are called recurrent.
- Can be proven to be Turing complete
  - Can simulate arbitrary programs.
- These architectures have a concept of time since information propagates around these loops once per time step.
- Ideal for problems with indeterminate input or output length.
- Has been used to scan over complicated images to decrease the computational complexity of each given subsample.
- Could be used to incorporate temporal information to distinguish up and downward going muon tracks (to look for atmospheric neutrinos) or to incorporate information from late interactions like neutrons and Michel electrons.
- Possibly also useful for online triggering.