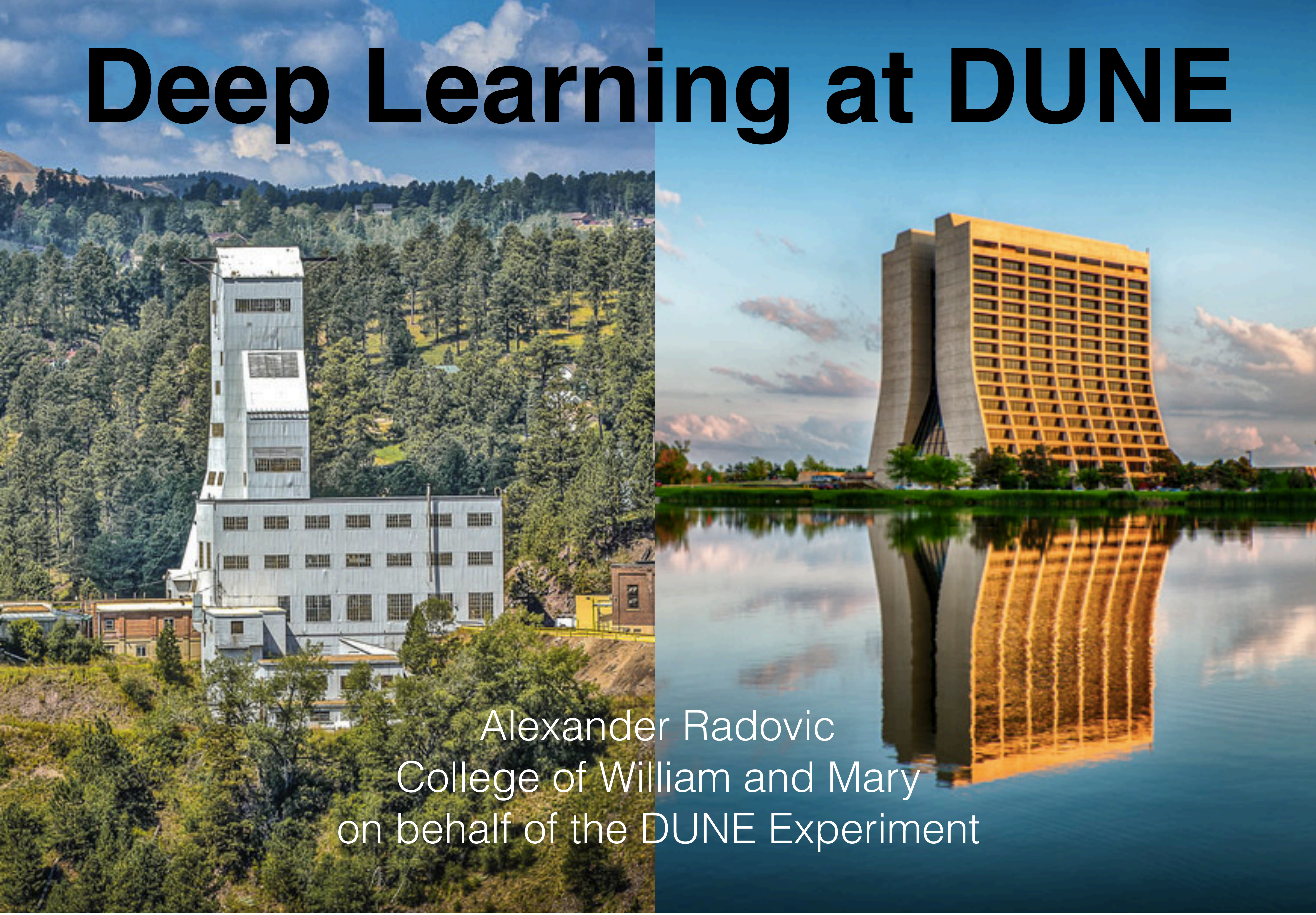
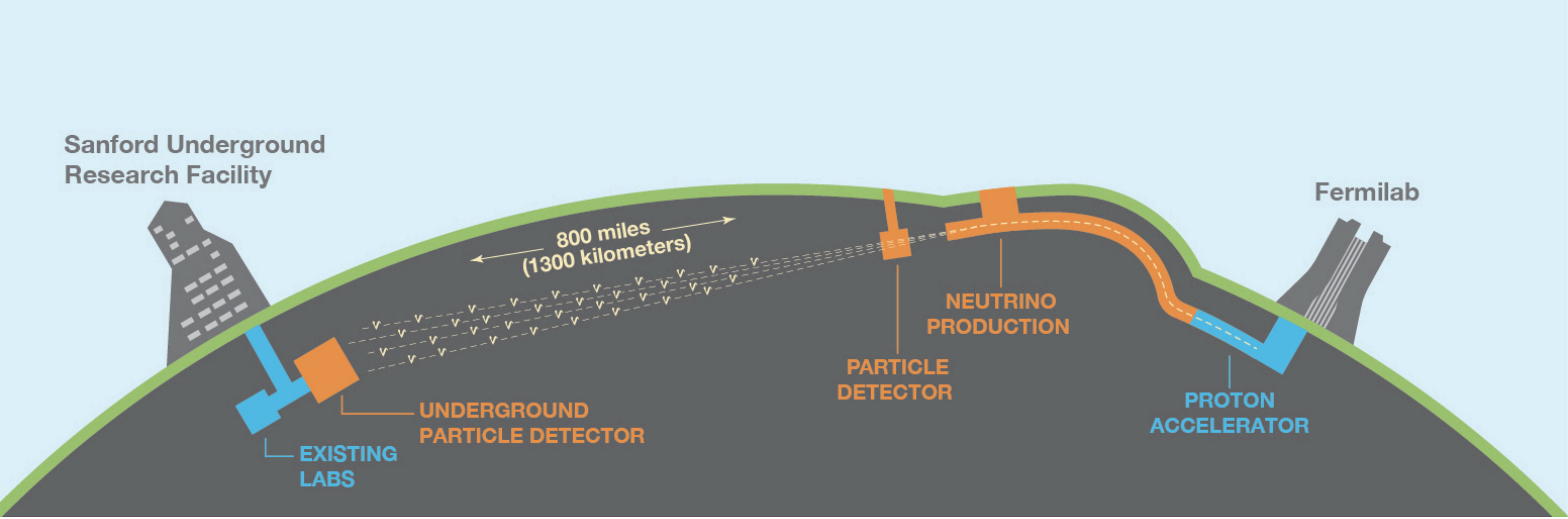
Deep Learning at DUNE



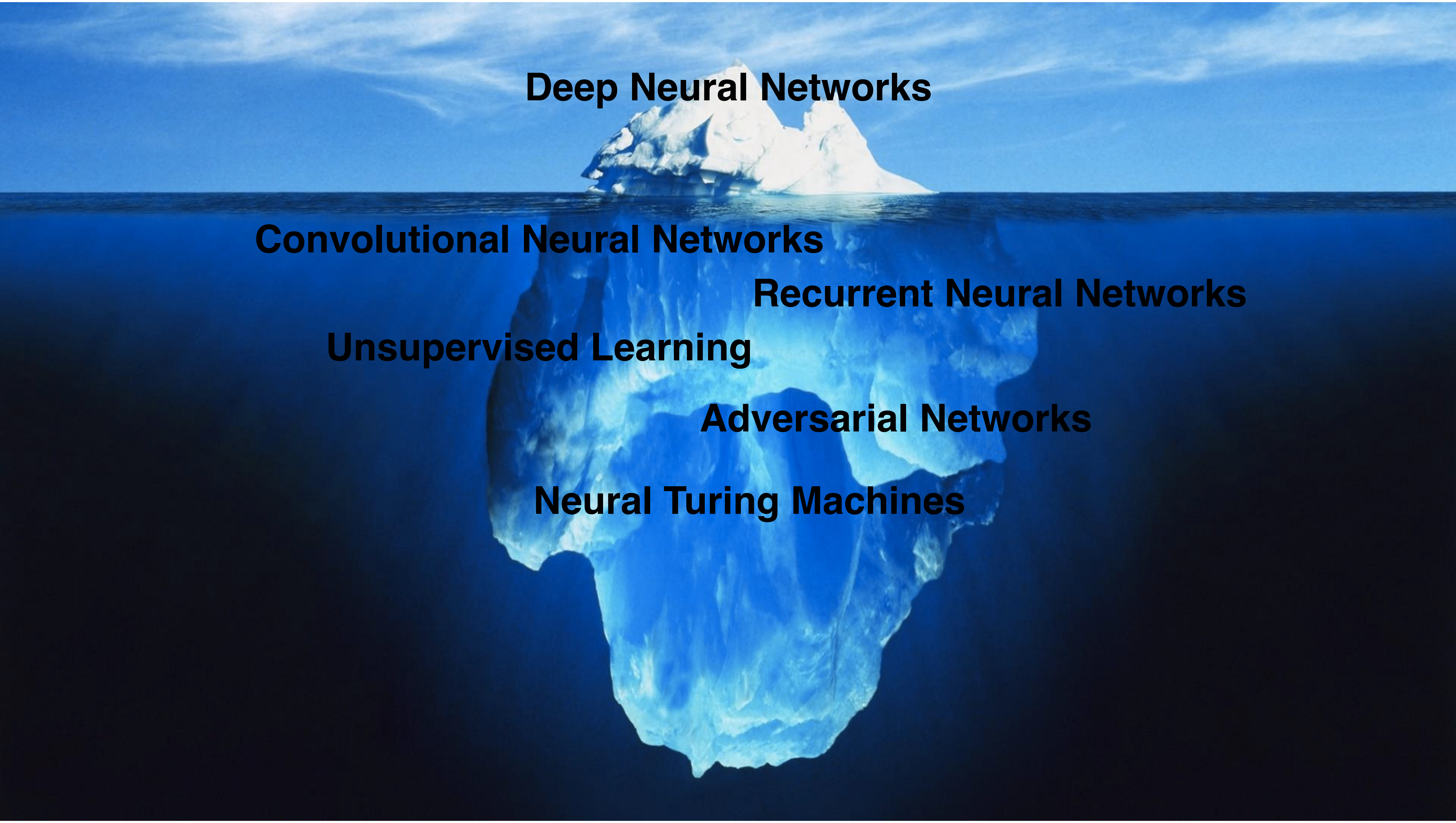
Alexander Radovic
College of William and Mary
on behalf of the DUNE Experiment

The DUNE Experiment

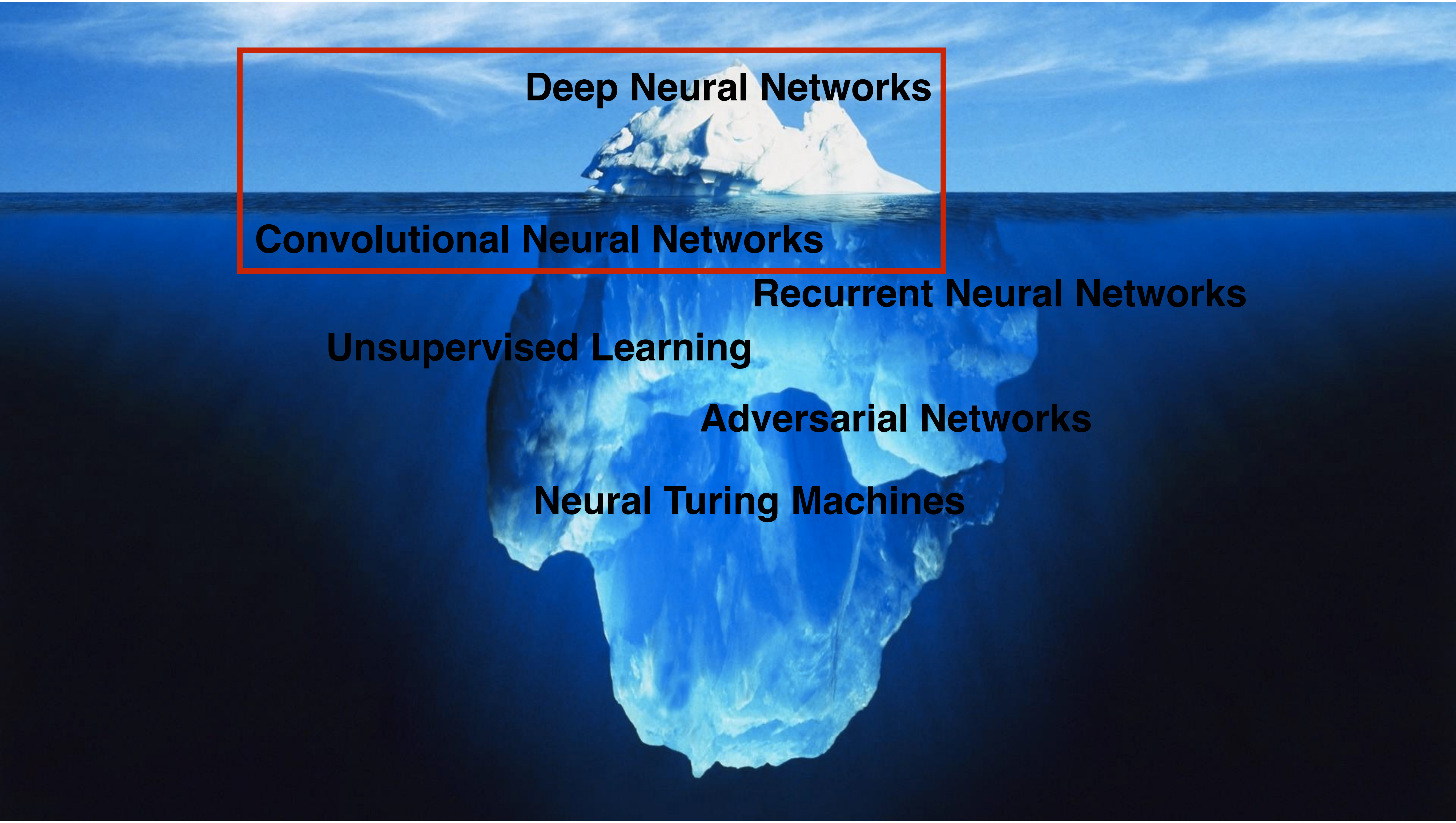
- Planned precision long baseline neutrino oscillation experiment.
- Designed to provide definitive answers on CP violation in the neutrino sector and the mass hierarchy.
- ND design still ongoing, but the FD will be a several KT underground liquid argon TPC.



Deep Learning



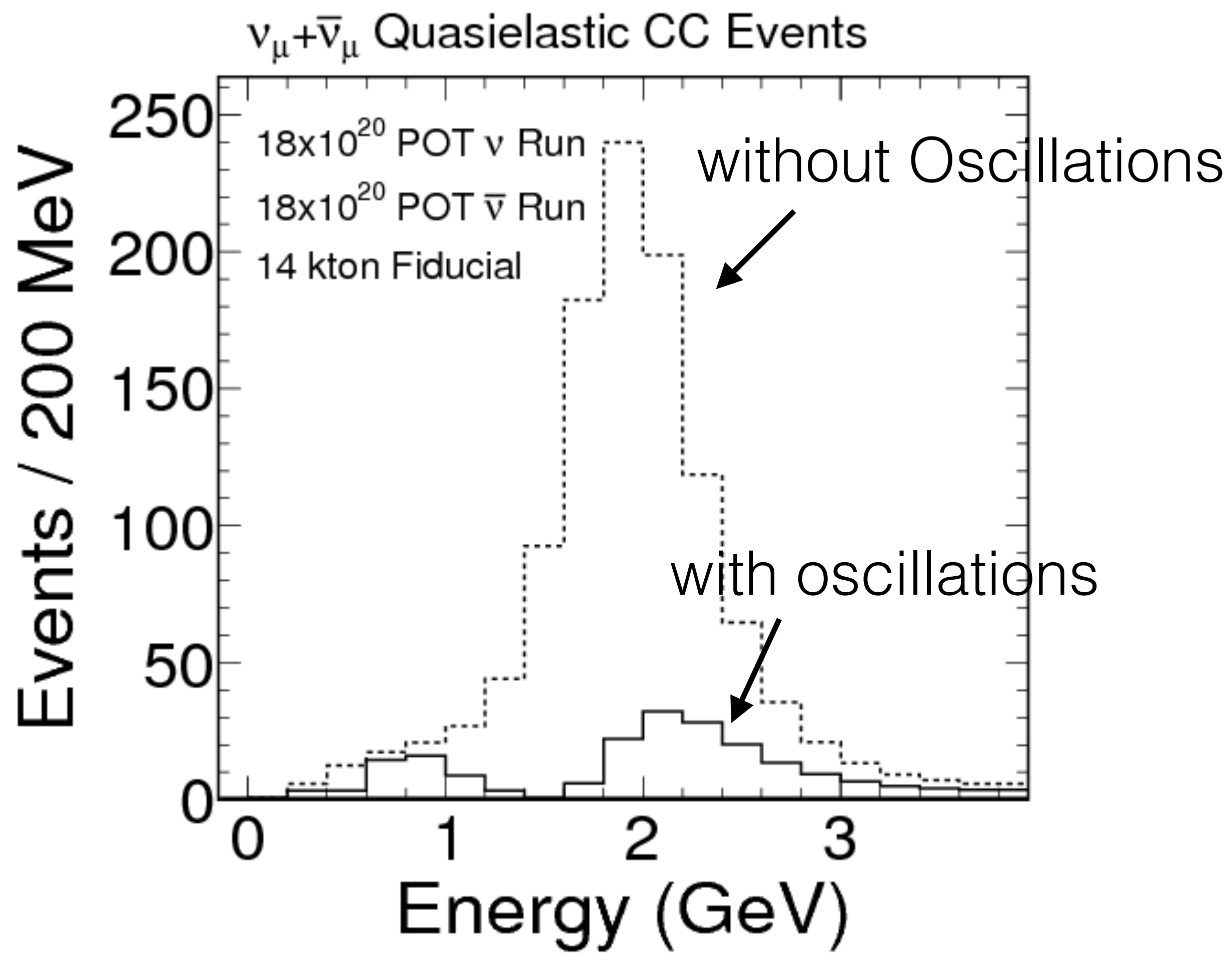
Deep Learning

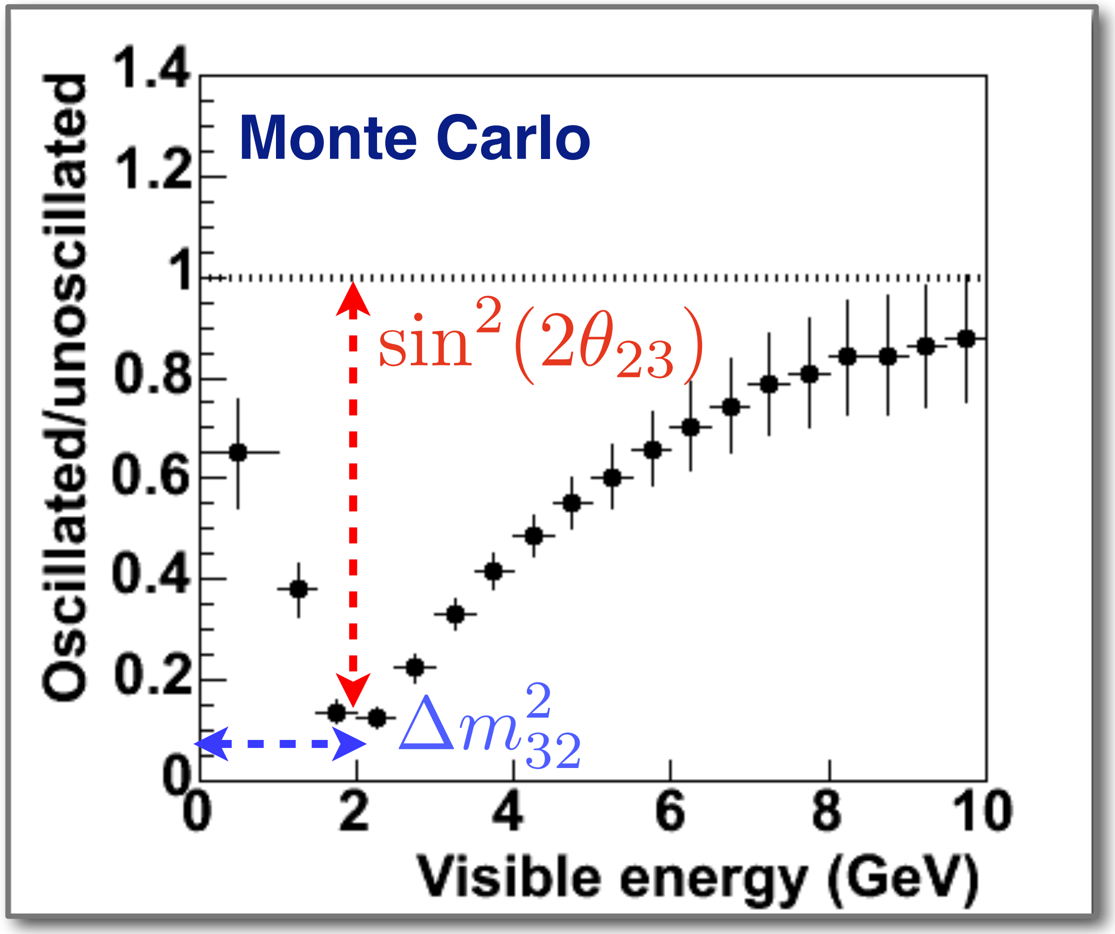


Why Deep Neural Networks?

- Measuring neutrino oscillations is all about measuring how neutrinos change between different lepton flavor states as a function of distance traveled and neutrino energy.

$$P(\nu_\mu \rightarrow \nu_\mu) \approx 1 - \sin^2(2\theta_{23}) \sin^2\left(\frac{1.27\Delta m^2_{atm} L}{E}\right)$$



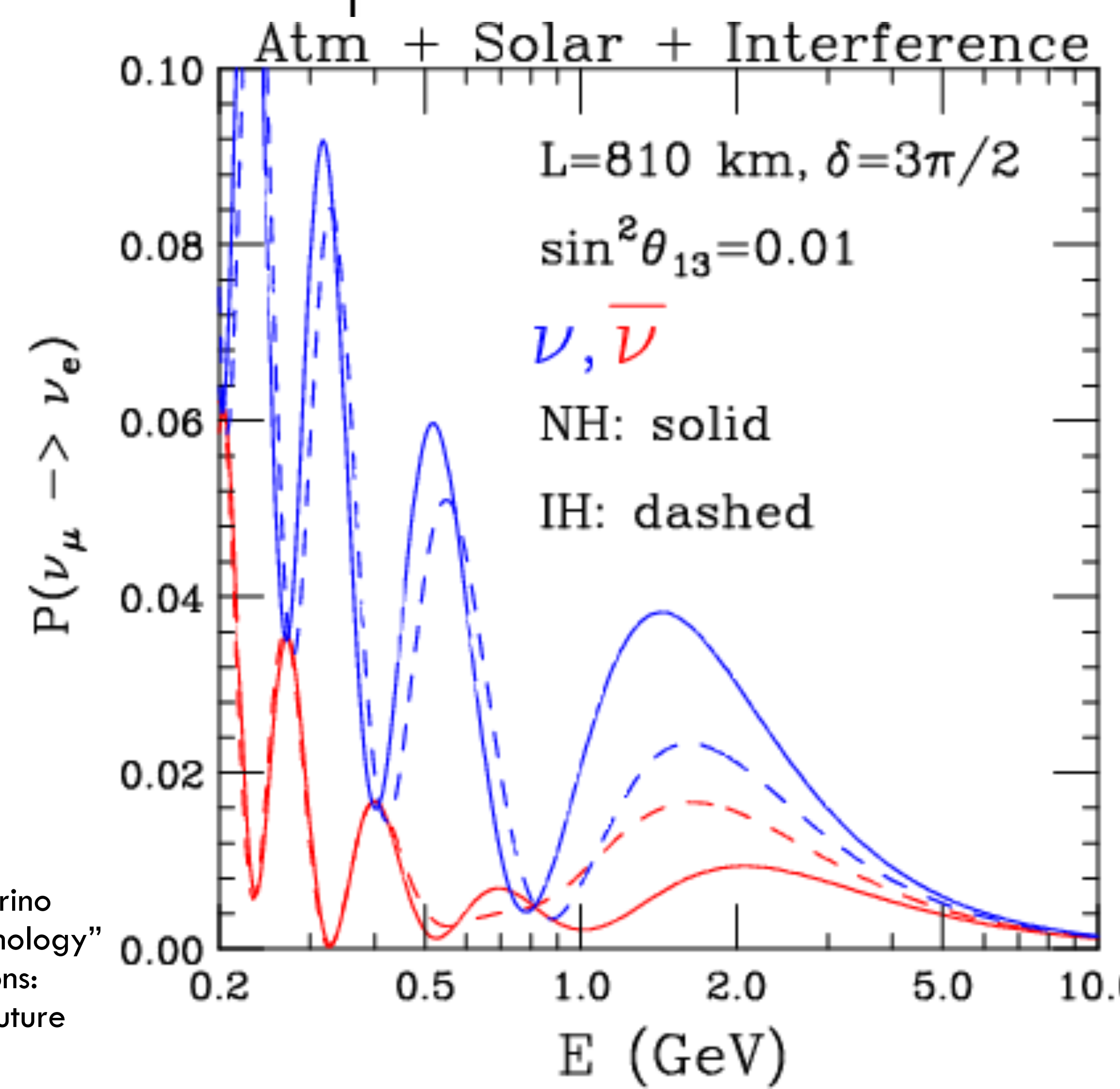


Why Deep Neural Networks?

- Measuring neutrino oscillations is all about measuring how neutrinos change between different lepton flavor states as a function of distance traveled and neutrino energy.

$$P(\nu_\mu \rightarrow \nu_e) \approx \left| \sqrt{P_{atm}} e^{-i\left(\frac{\Delta m^2_{32} L}{4E} + \delta_{cp}\right)} + \sqrt{P_{sol}} \right|^2$$

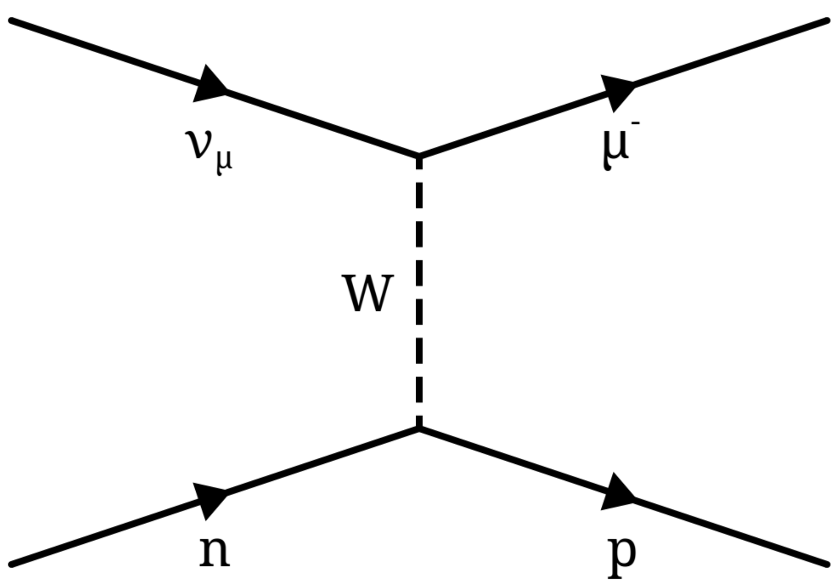
$$P_{atm} = \sin^2\theta_{23} \sin^2 2\theta_{13} \sin^2 \frac{\Delta m^2_{31} L}{4E}$$



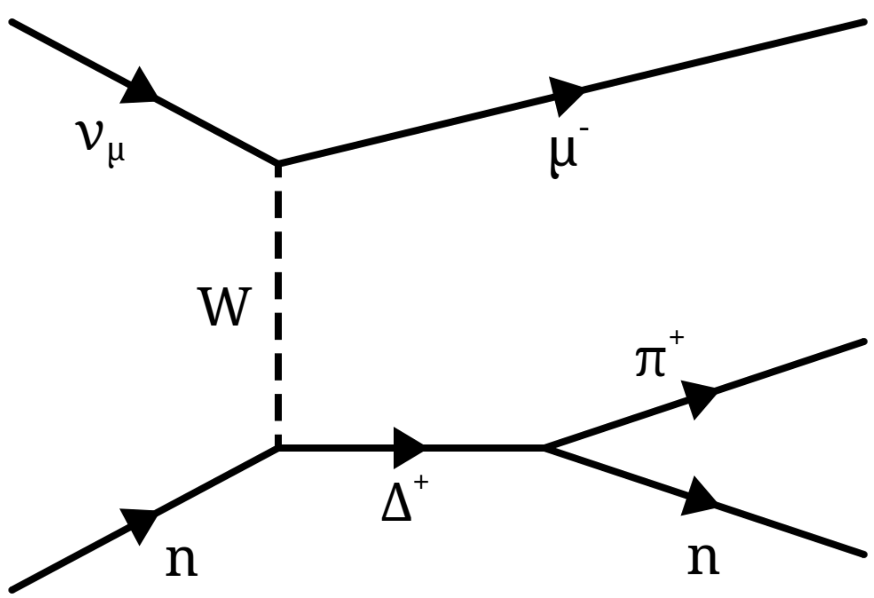
From S. Parke, "Neutrino Oscillation Phenomenology" in Neutrino Oscillations: Present Status and Future Plans

Why Deep Neural Networks?

- Any oscillation analysis can benefit from precise identification of the interaction in two ways:
  - Estimating the lepton flavor of the incoming neutrino.
  - Correctly identifying the type of neutrino interaction, to better estimate the neutrino energy, aka is it a quasi elastic event or a resonance event?



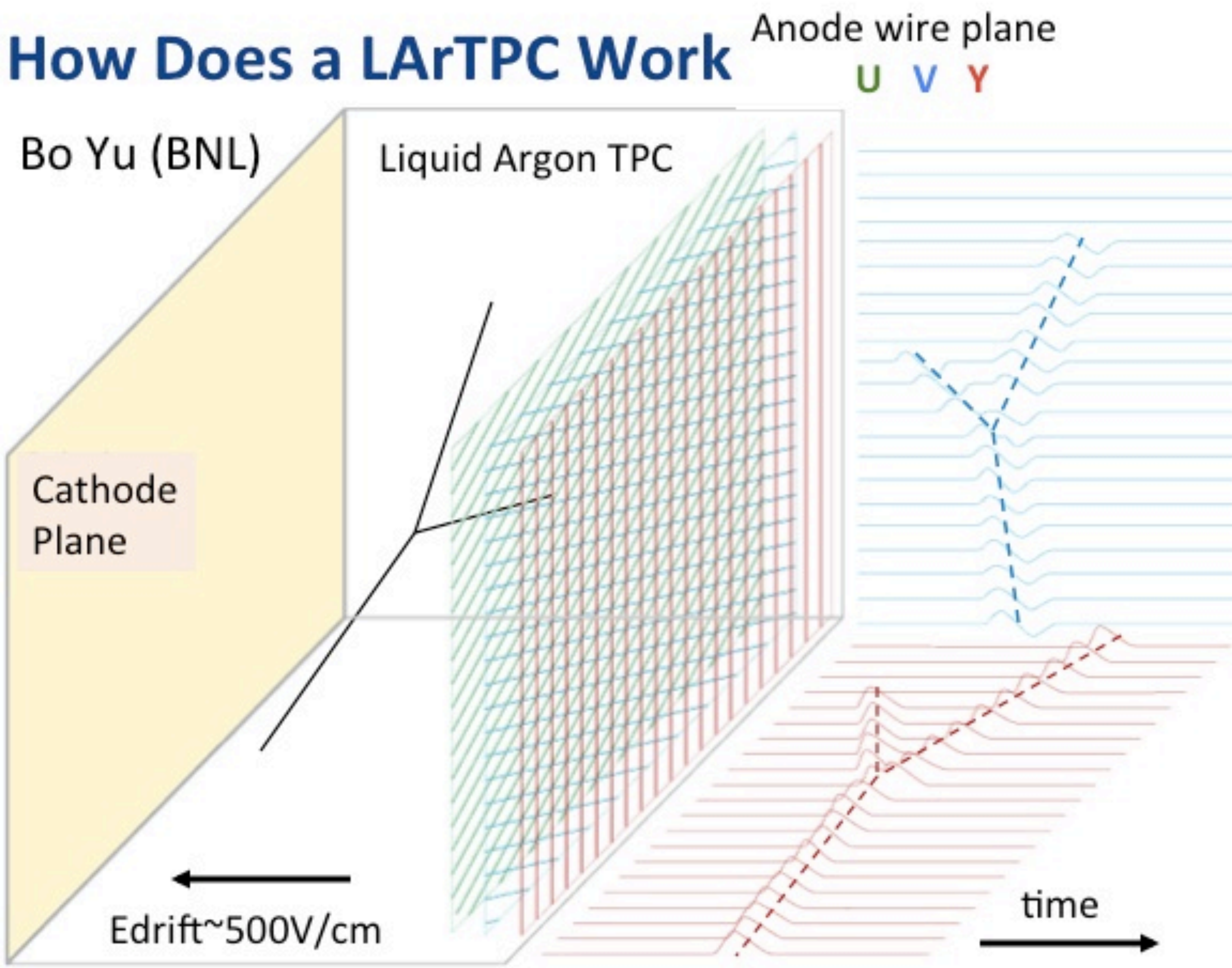
Quasi-Elastic



Resonance

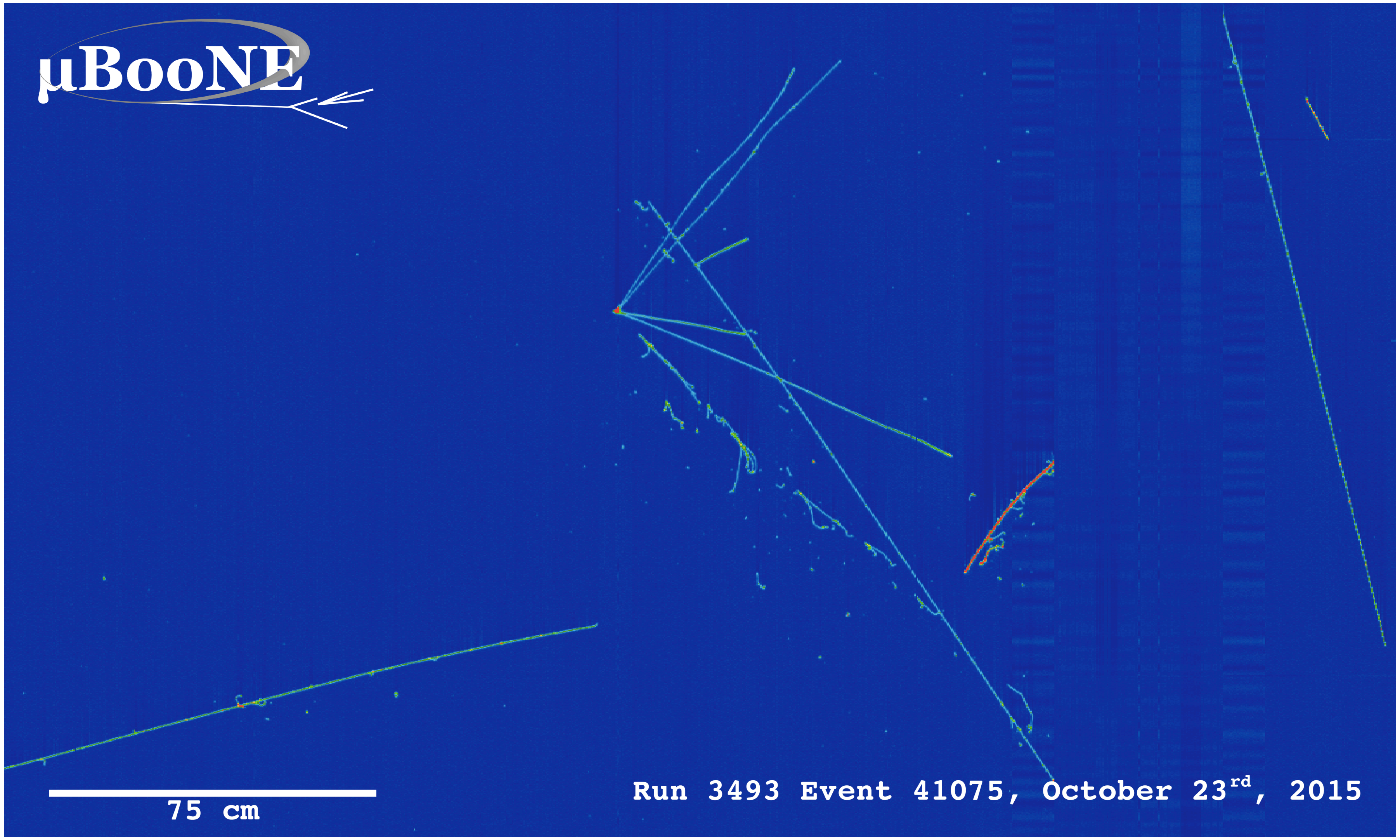
Why Deep Neural Networks?

- Liquid argon detectors are also the perfect domain:
  - Large ~uniform volumes where spatially invariant response is a benefit.
  - One, main, detector system.



Why Deep Neural Networks?

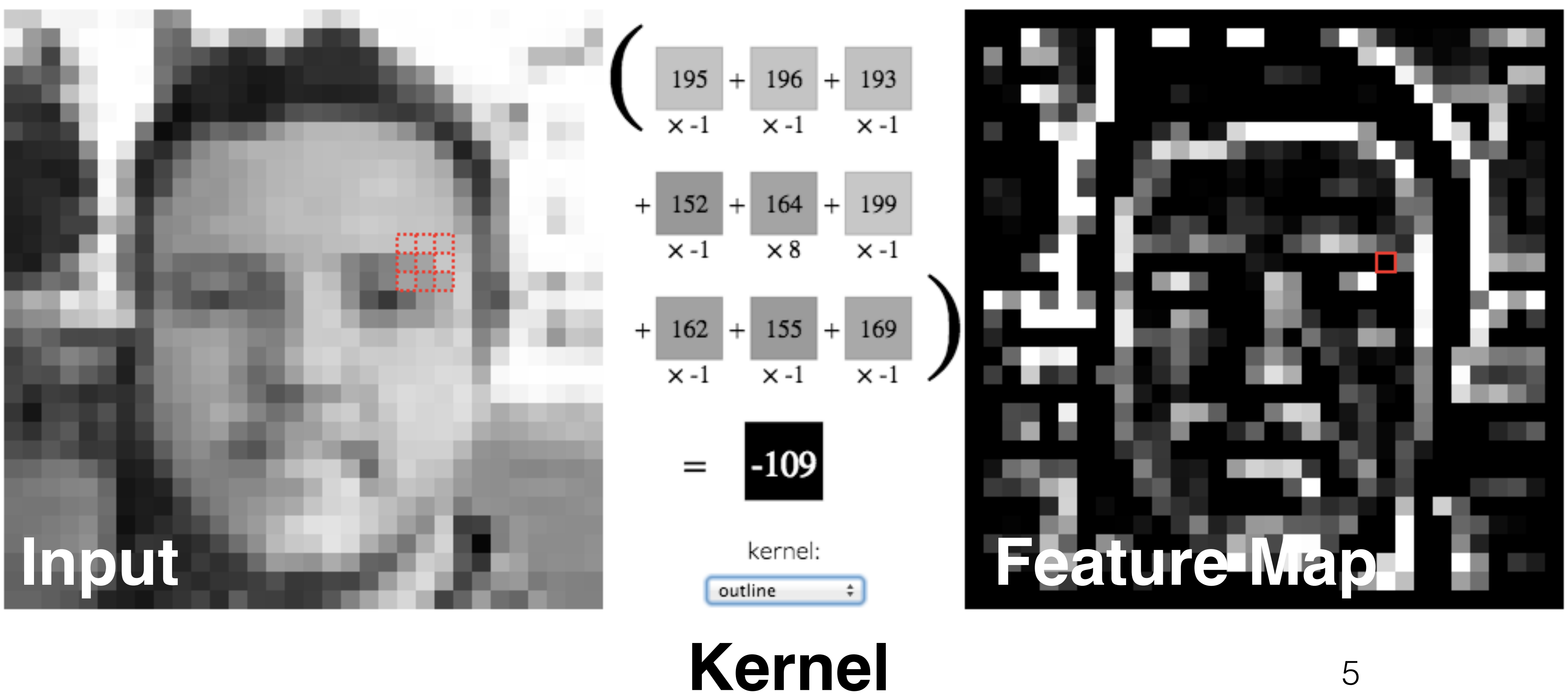
- Liquid argon detectors are also the perfect domain:
  - Large ~uniform volumes where spatially invariant response is a benefit.
  - One, main, detector system.



Convolutional Neural Networks

Instead of training a weight for every input pixel, try learning weights that describe kernel operations, convolving that kernel across the entire image to exaggerate useful features.

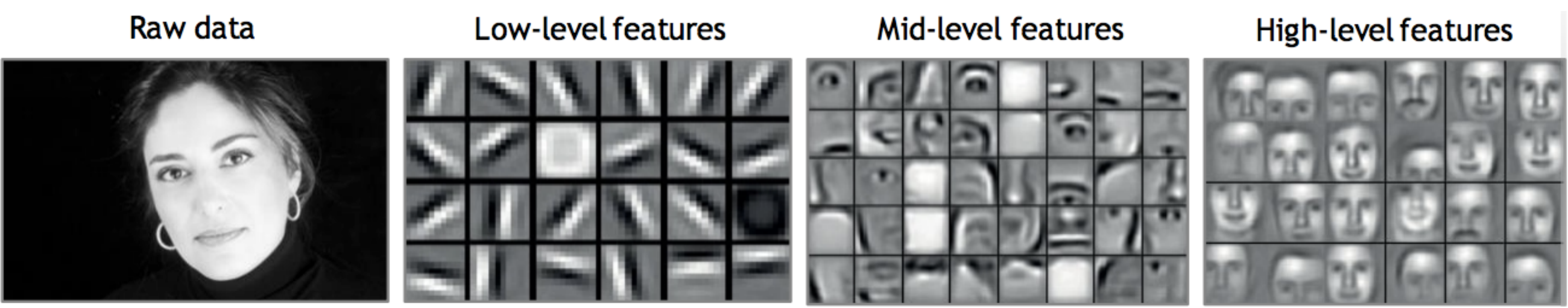
Inspired by research showing that cells in the visual cortex are only responsive to small portions of the visual field.



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

Convolutional Neural Networks

Instead of training a weight for every input pixel, try learning weights that describe kernel operations, convolving that kernel across the entire image to exaggerate useful features. Inspired by research showing that cells in the visual cortex are only responsive to small portions of the visual field.

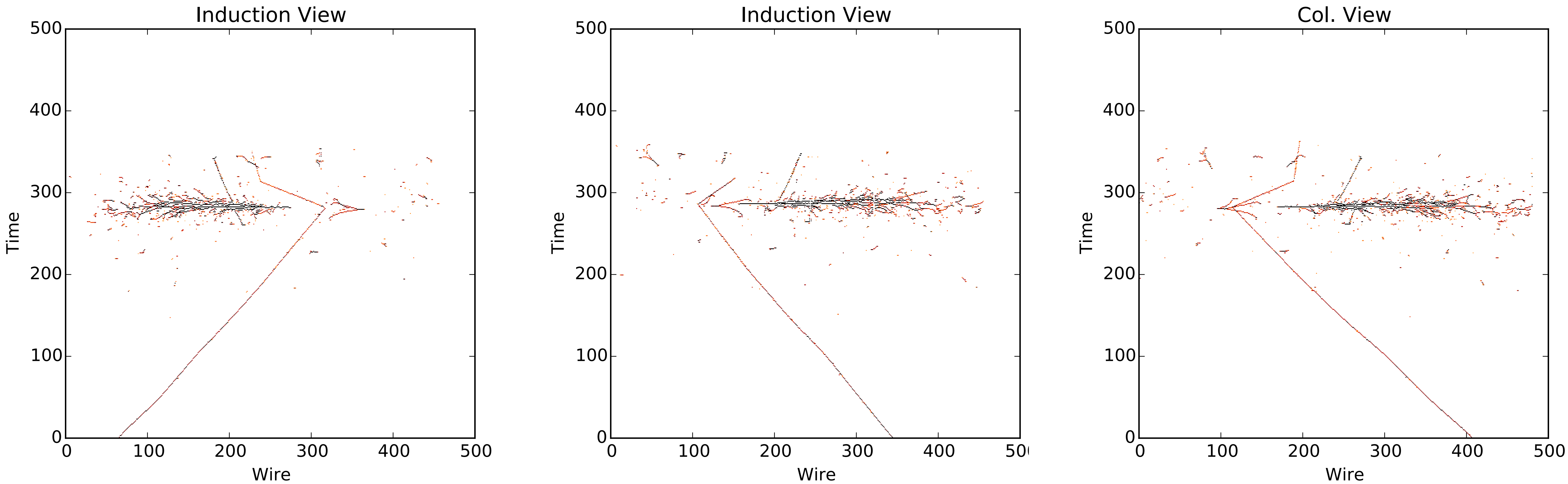


https://developer.nvidia.com/deep-learning-courses

Deep Learning for Event Identification

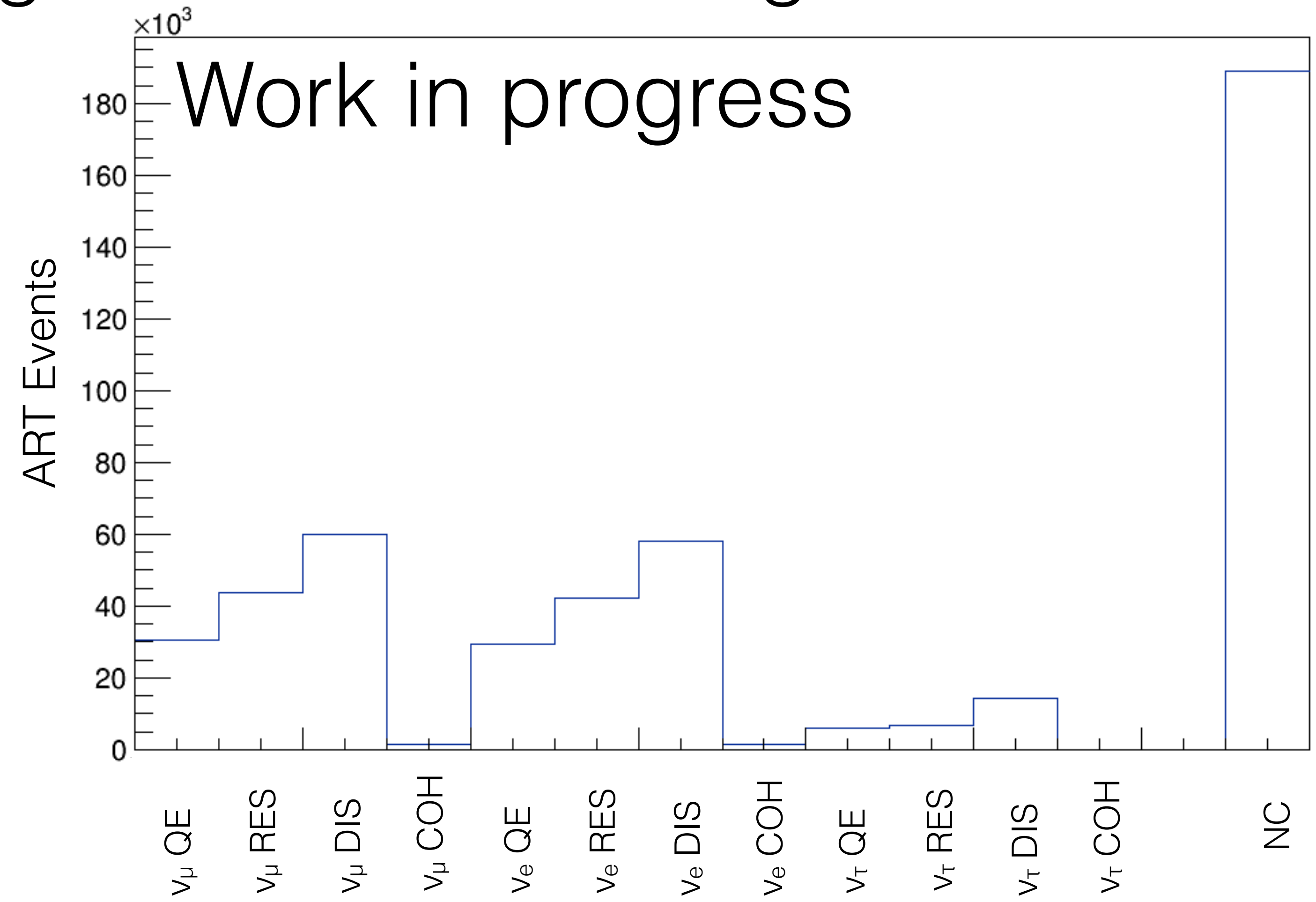
Our Input

Each "pixel" is the integrated ADC response in that time/ space slice. These maps are chosen to be 500 wires long and 1.2ms wide (split into 500 time chunks).



The Training Sample

- 1.2M events, only preselection requiring 100 hits split across any number of planes.
- Labels are from GENIE truth, neutrino vs. antineutrino is ignored.
- No oscillation information, just the raw input distributions.
- 80% for training and 20% for testing.

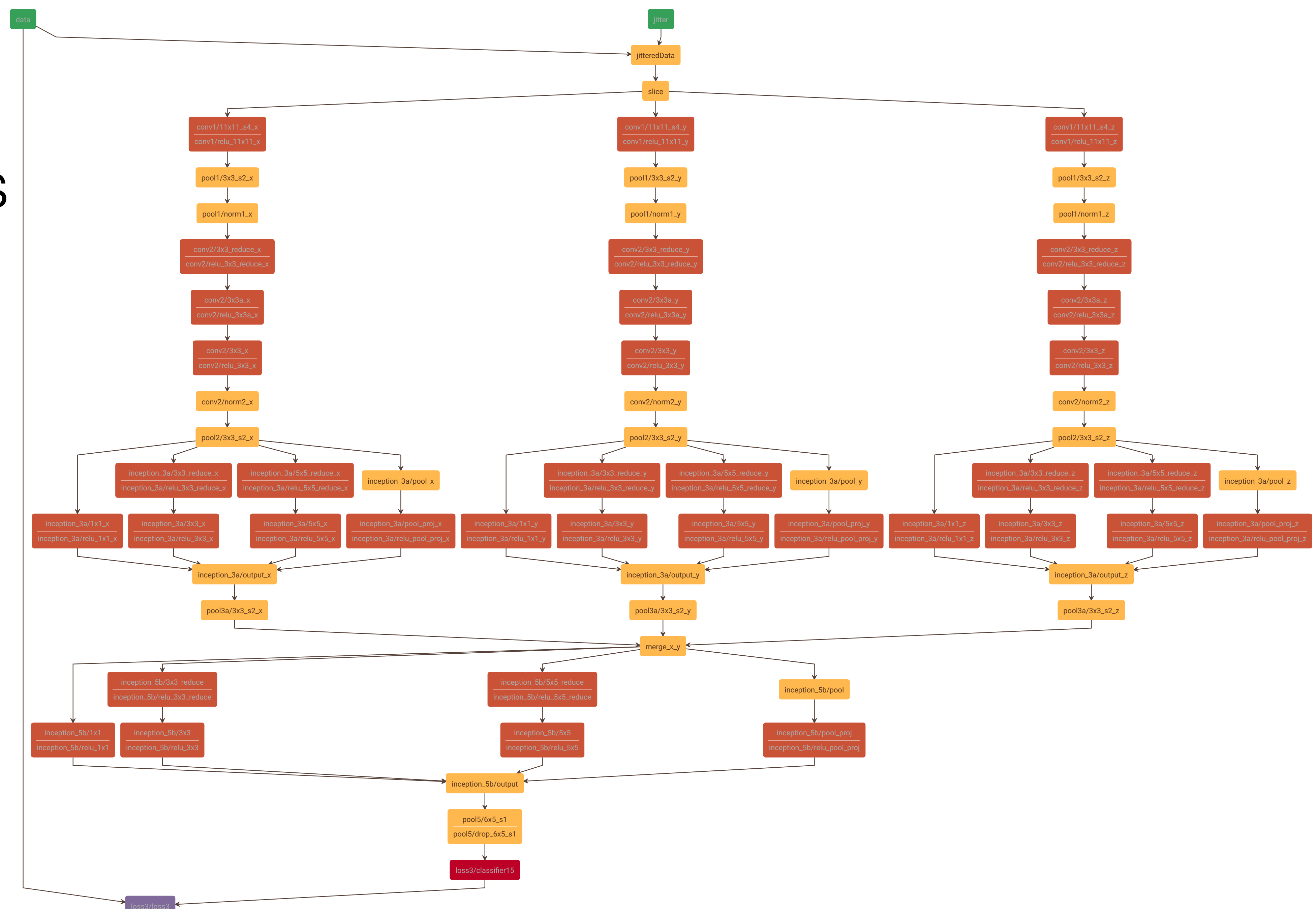


Our Architecture

Based on the NOvA CNN, named **CVN**. Small edits to better suit a larger input image and three distinct views.

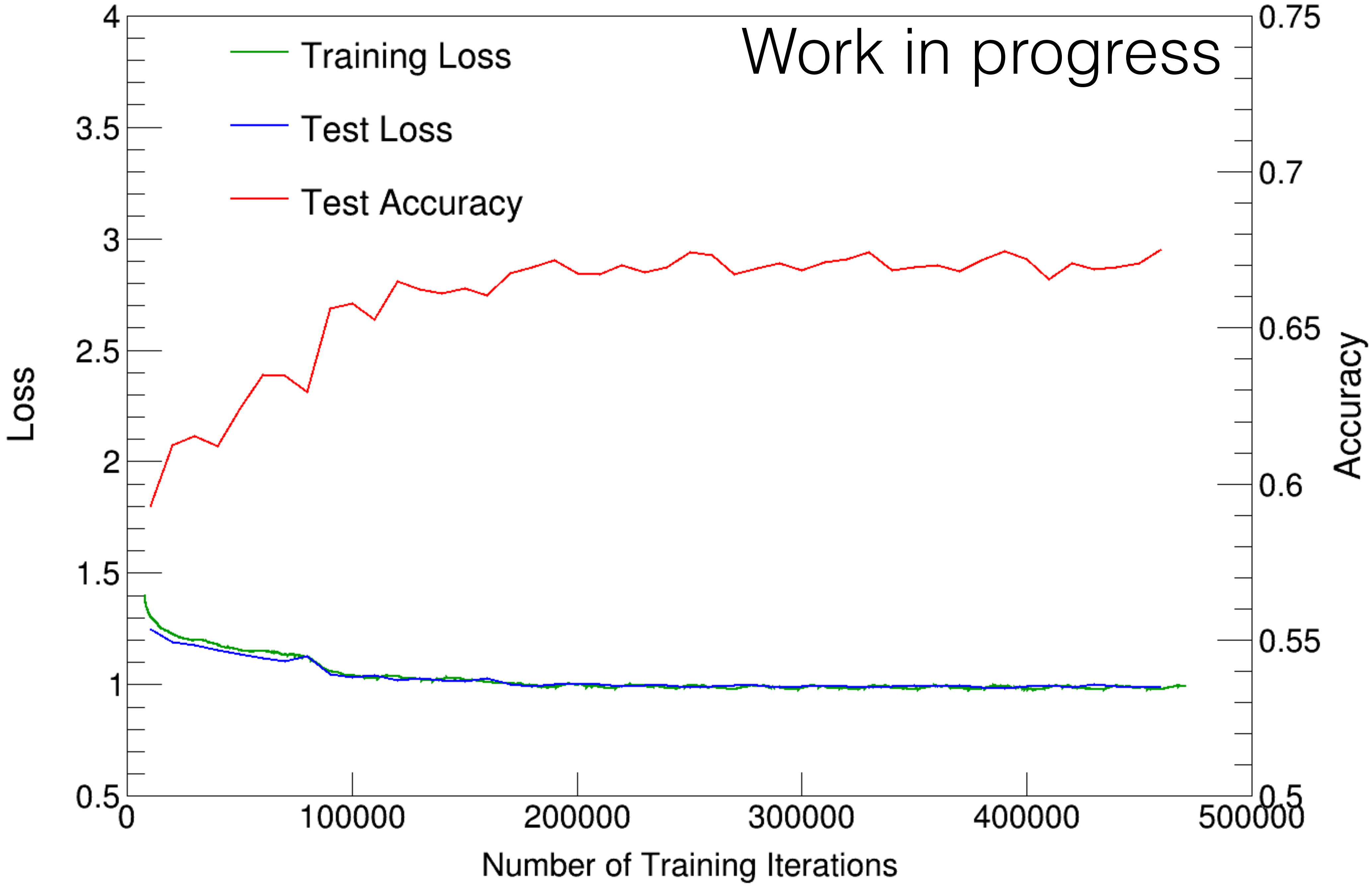
The architecture attempts to categorize events as $\{\nu_\mu, \nu_e, \nu_\tau\} \times$ {QE,RES,DIS}, NC.

Built in the excellent CAFFE framework.

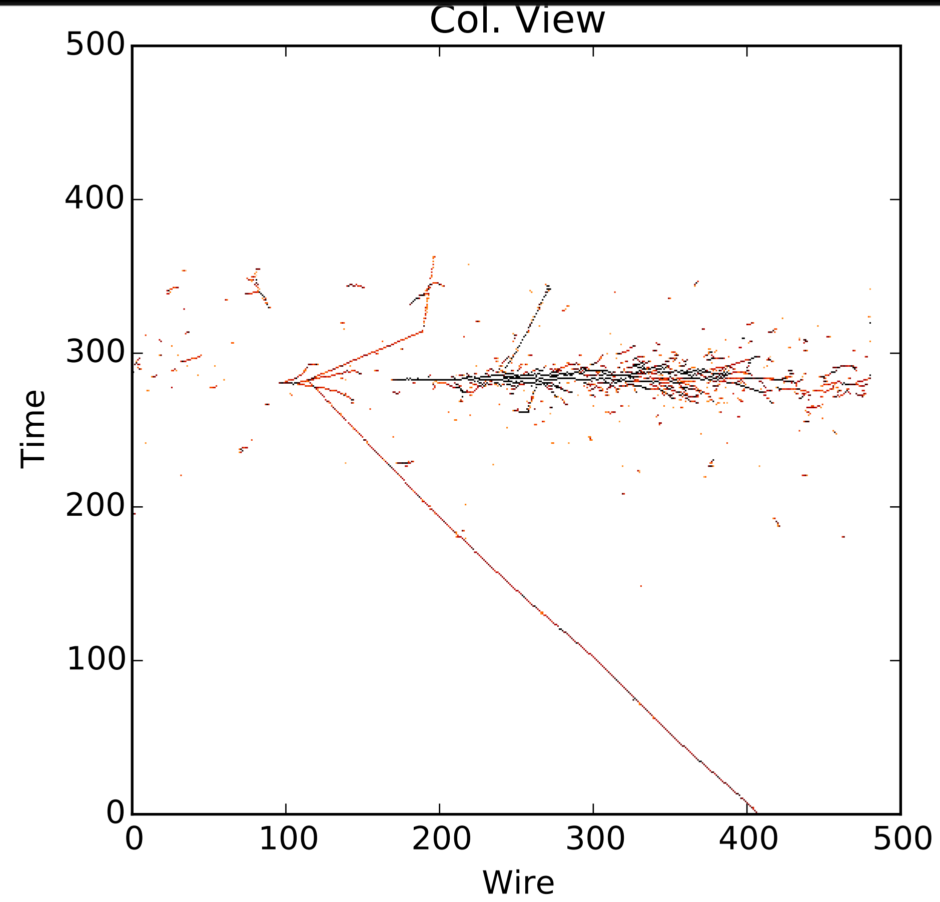


Training Performance

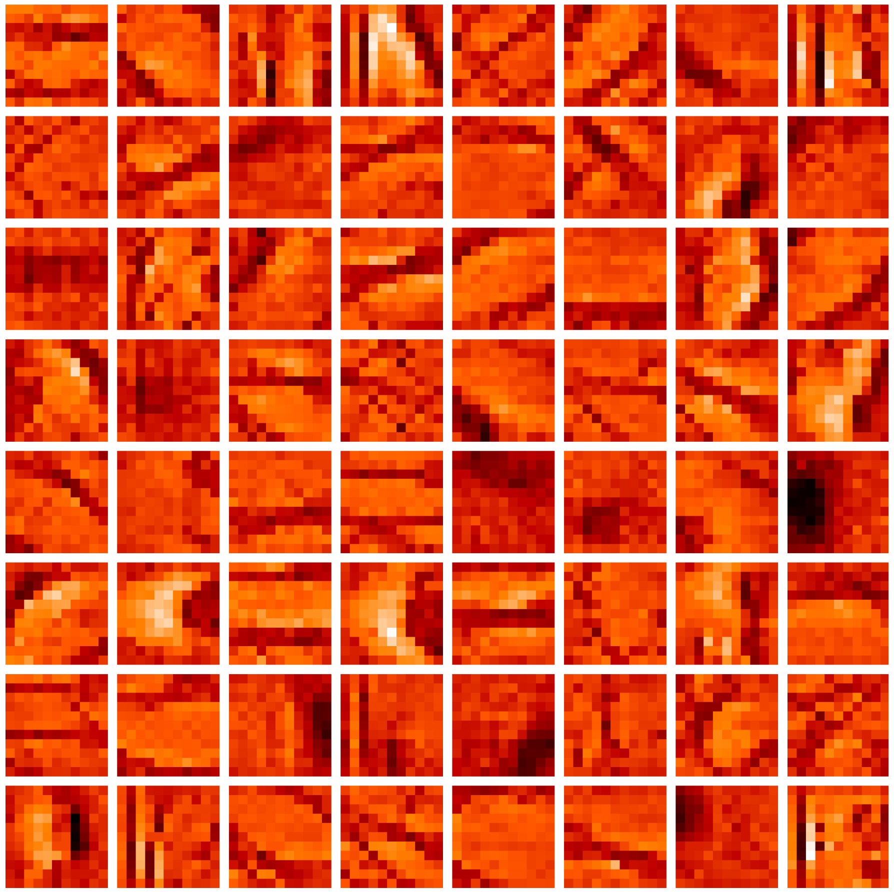
No sign of overtraining- exceptional training test set performance agreement!



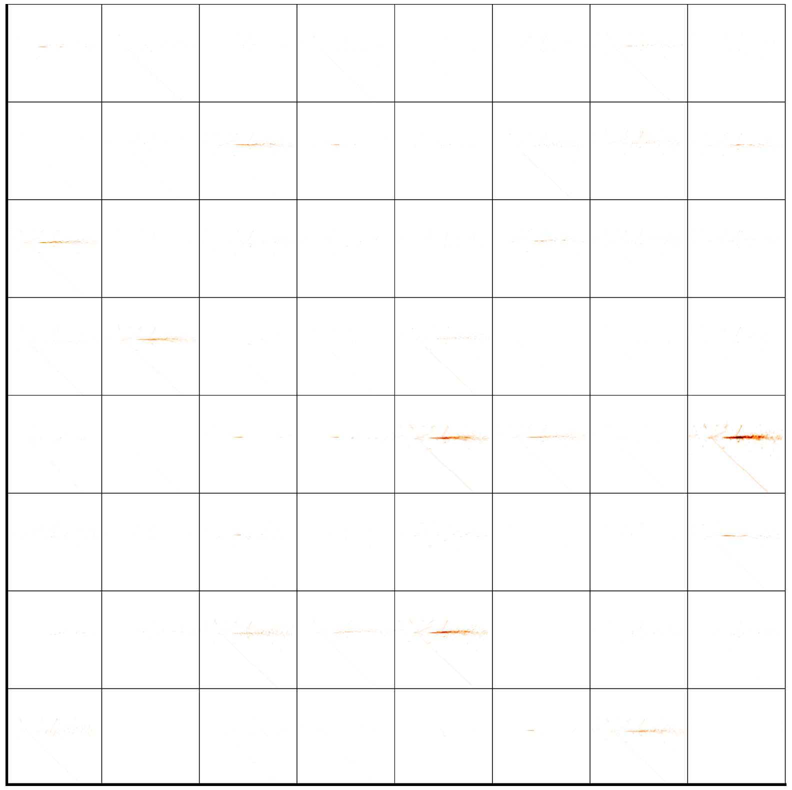
Example CVN Kernels In Action: First Convolution



X



=



Here the earliest convolutional layer in the network starts by pulling out primitive shapes and lines.

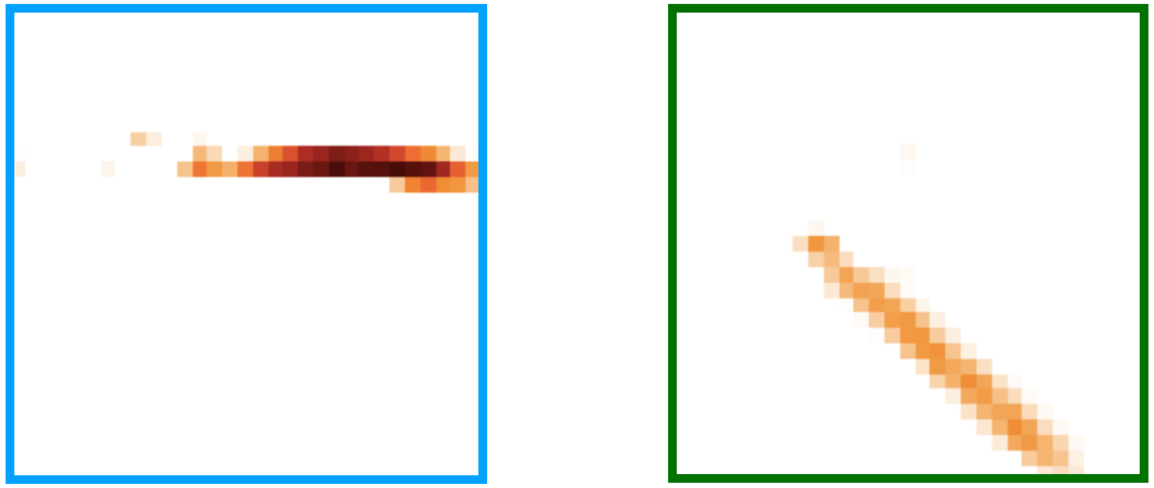
Already “showers” and “tracks” are starting to form.

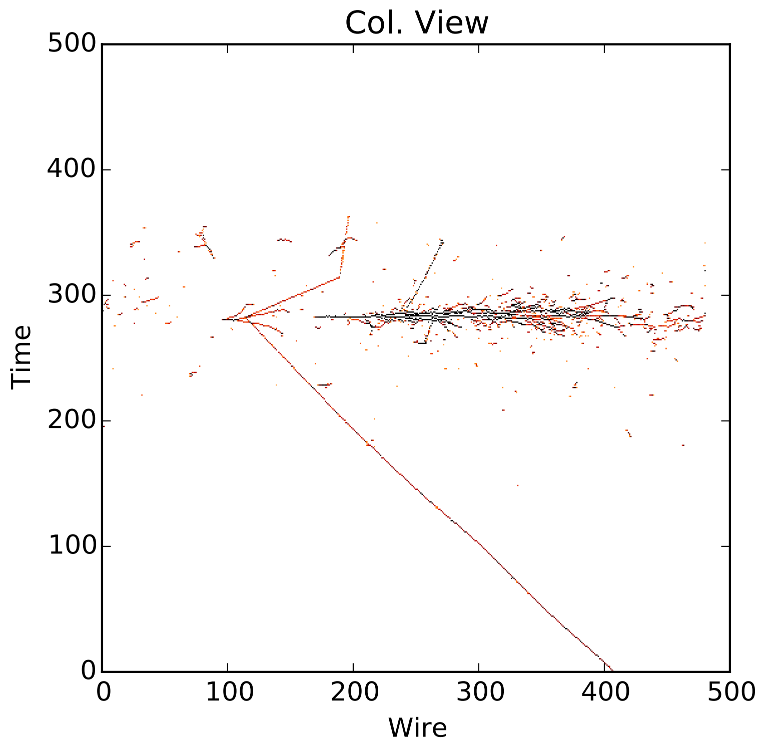
Example CVN Kernels In Action: First Inception Module Output

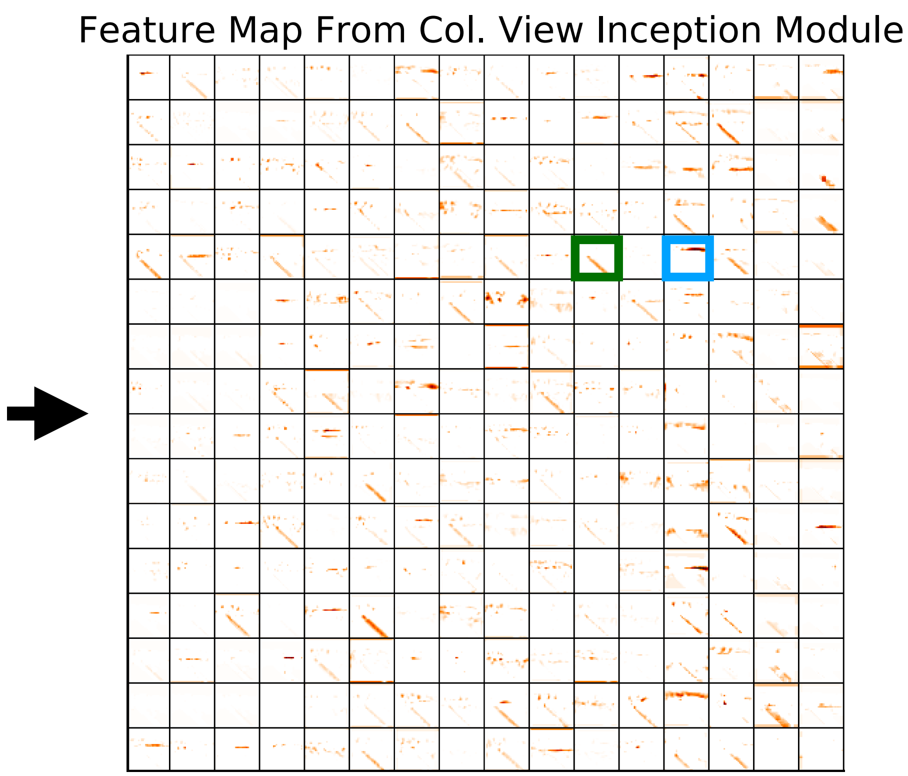
Deeper in the network, now after the first inception module we can see more complex features have started to be extracted.

Some seem particularly sensitive to muon tracks, EM showers.

True NuMu DIS Event





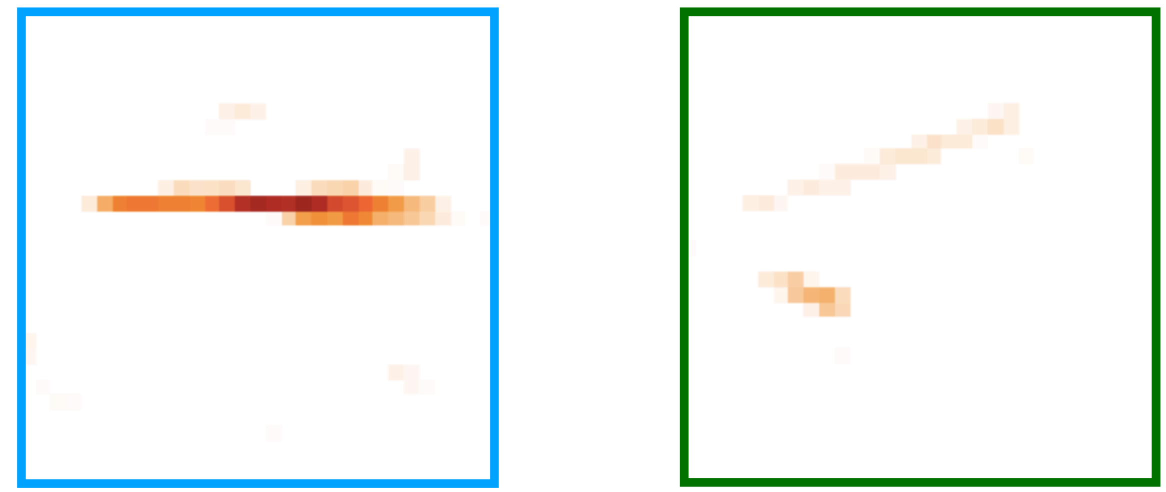


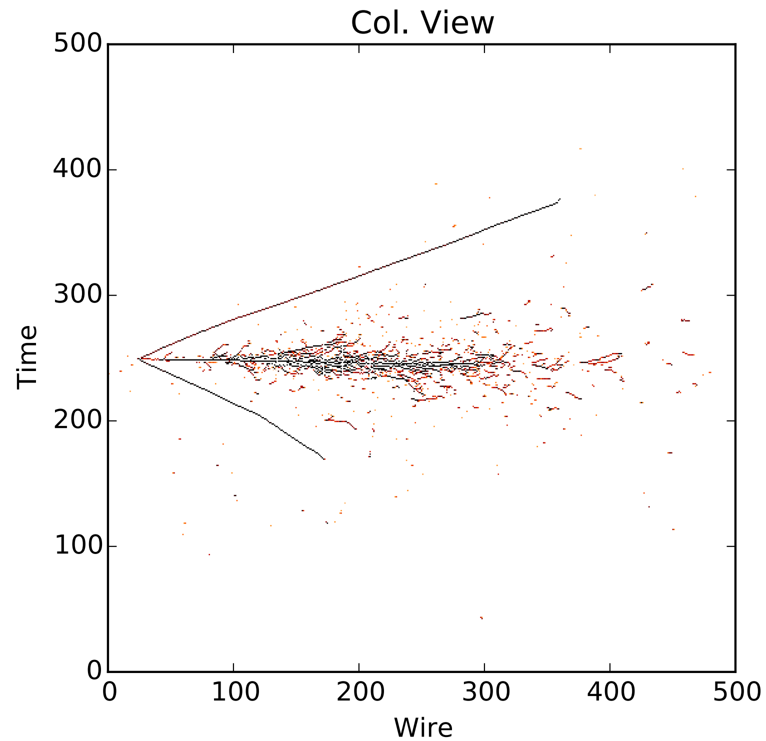
Example CVN Kernels In Action: First Inception Module Output

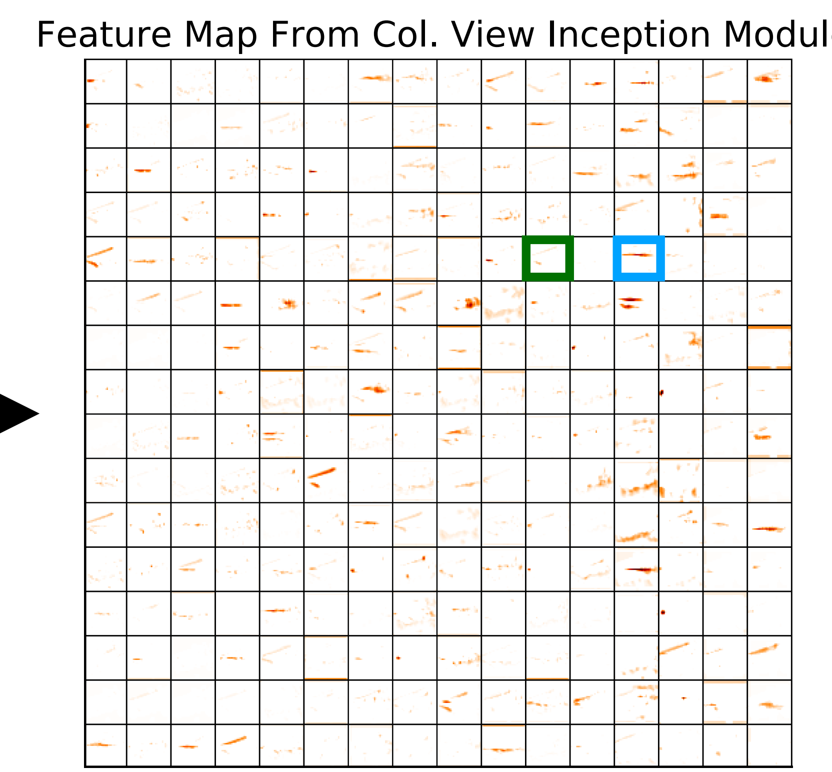
Deeper in the network, now after the first inception module we can see more complex features have started to be extracted.

Some seem particularly sensitive to muon tracks, EM showers.

True NuE COH Event

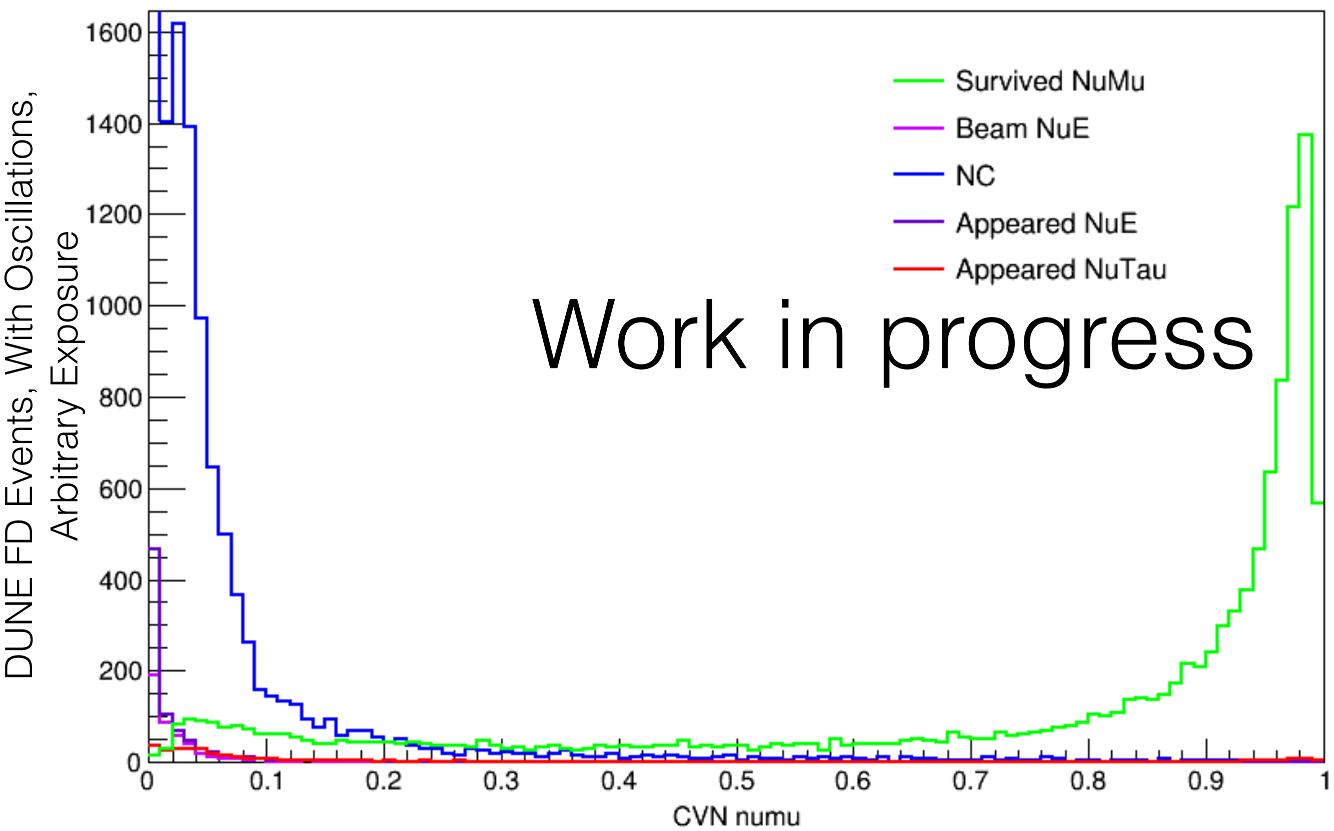




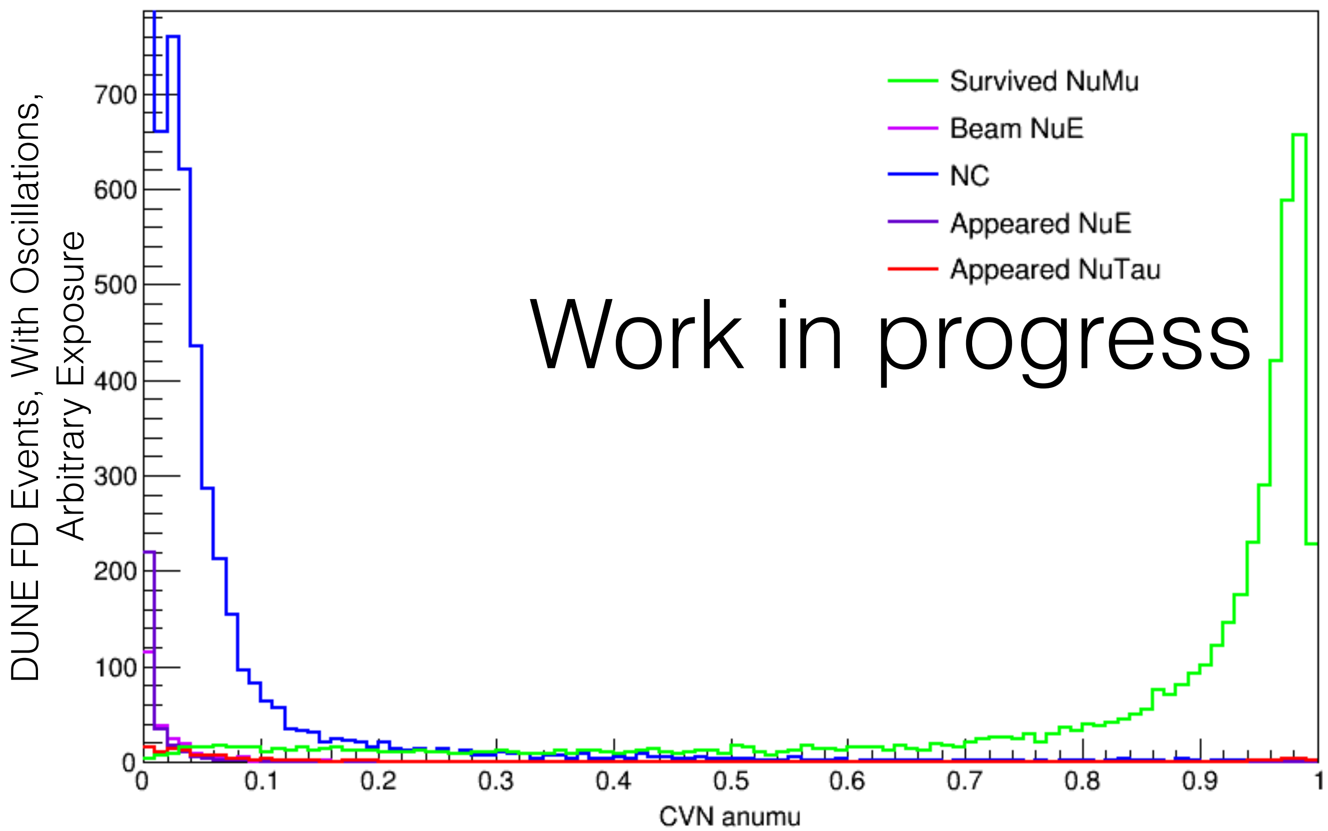


NuMu PID

Neutrino Beam



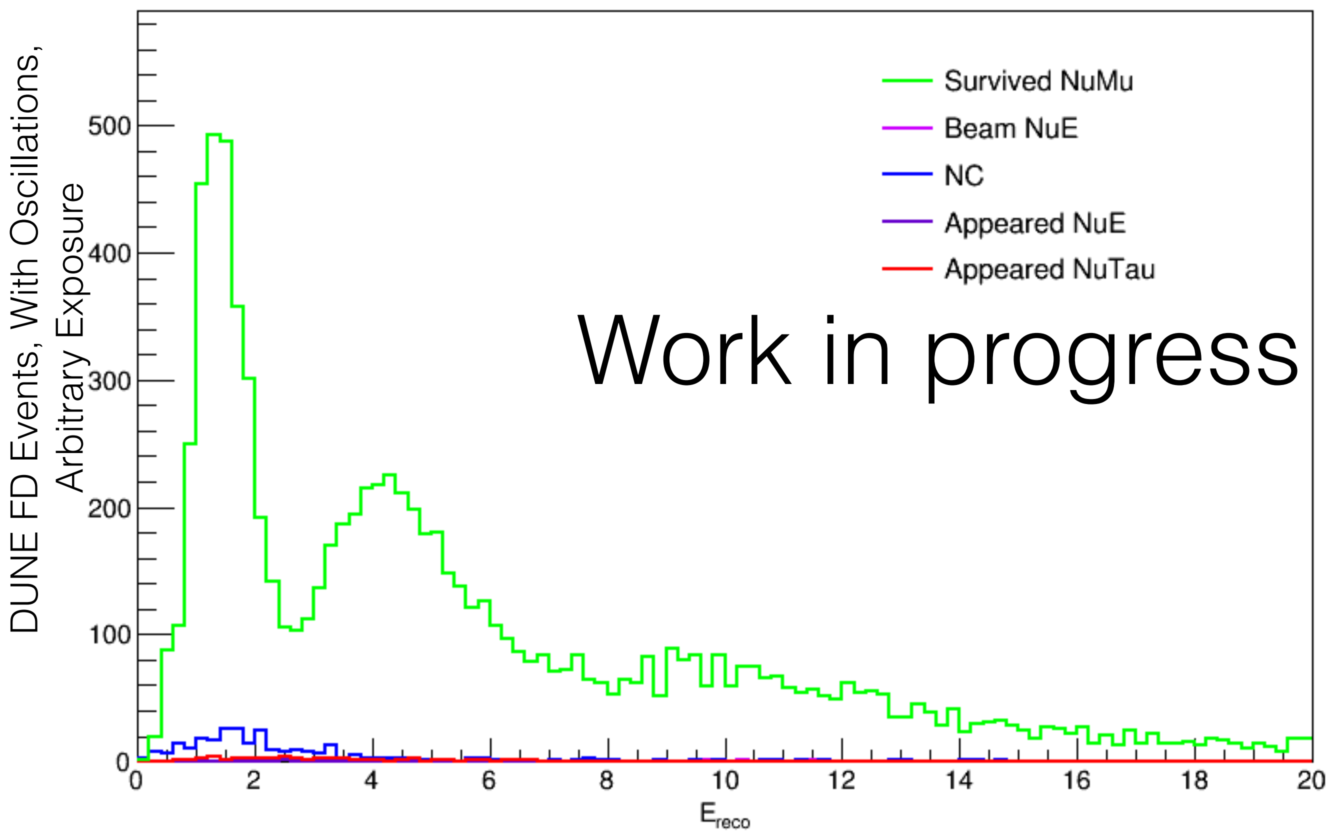
Anti-Neutrino Beam



Cut at 0.5, guarantees no double counting due to sofmax output of CVN

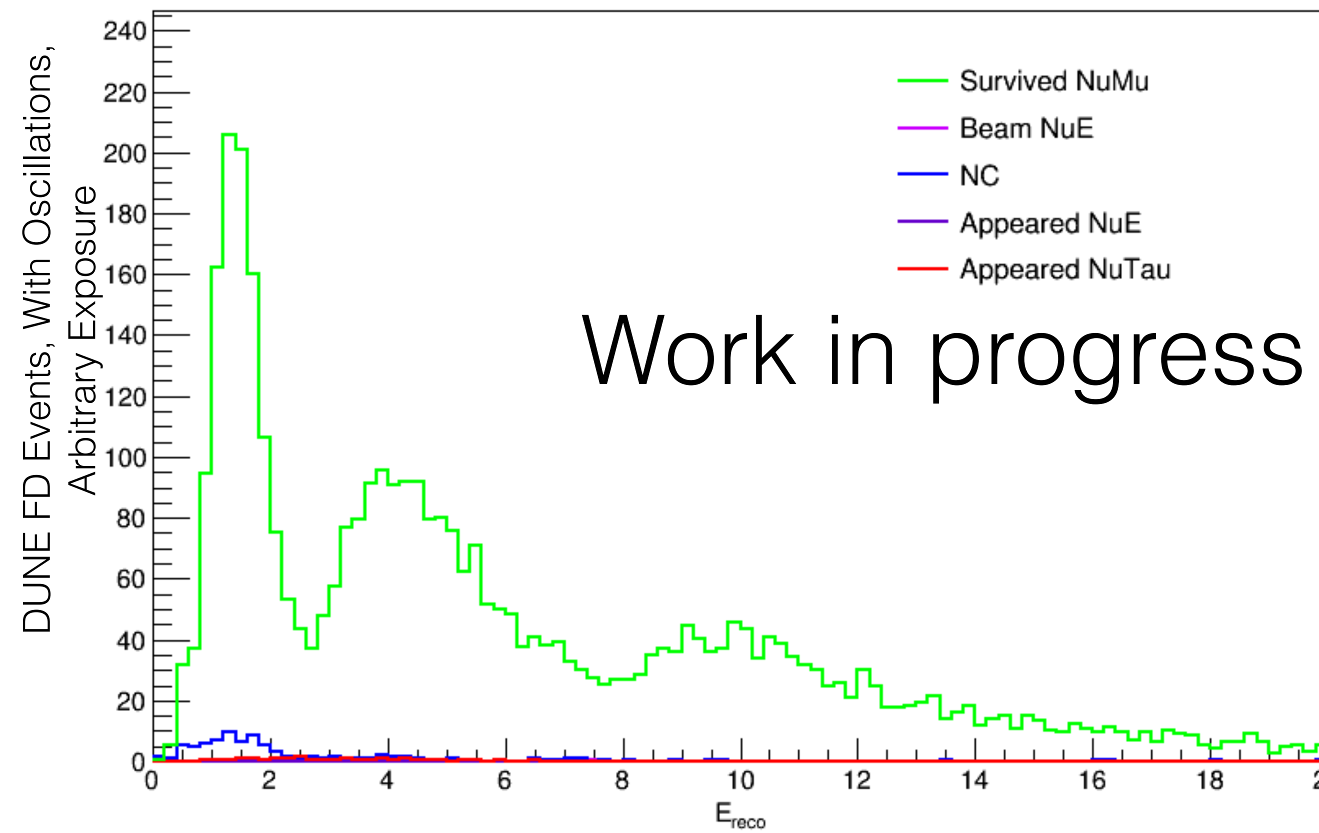
NuMu Selected Events, Reconstructed Energy Spectra

Neutrino Beam



	NuMu	Appeared NuE	Beam NuE	NC	NuTau
Efficiency	**80.6**				
Rejection		99.0	98.7	**97.6**	81.5

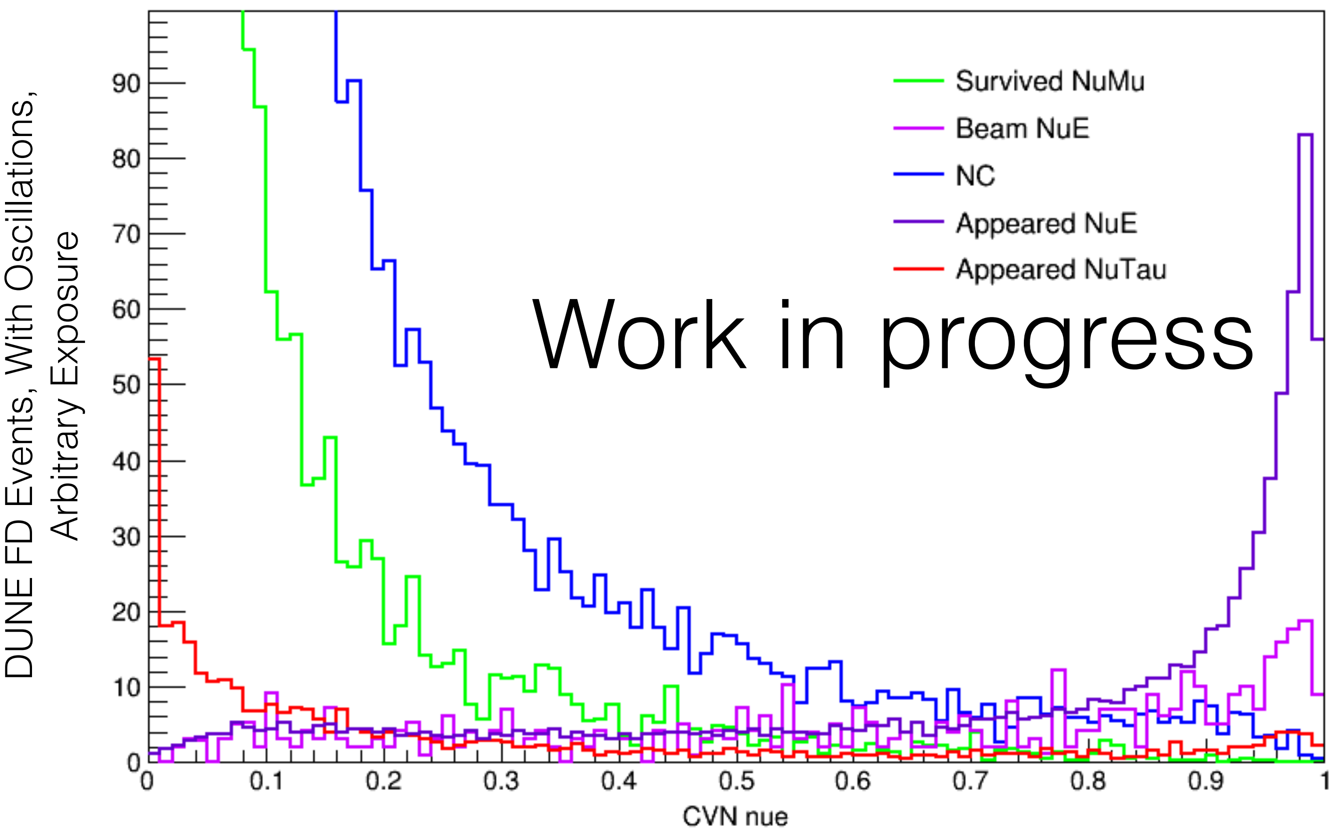
Anti-Neutrino Beam



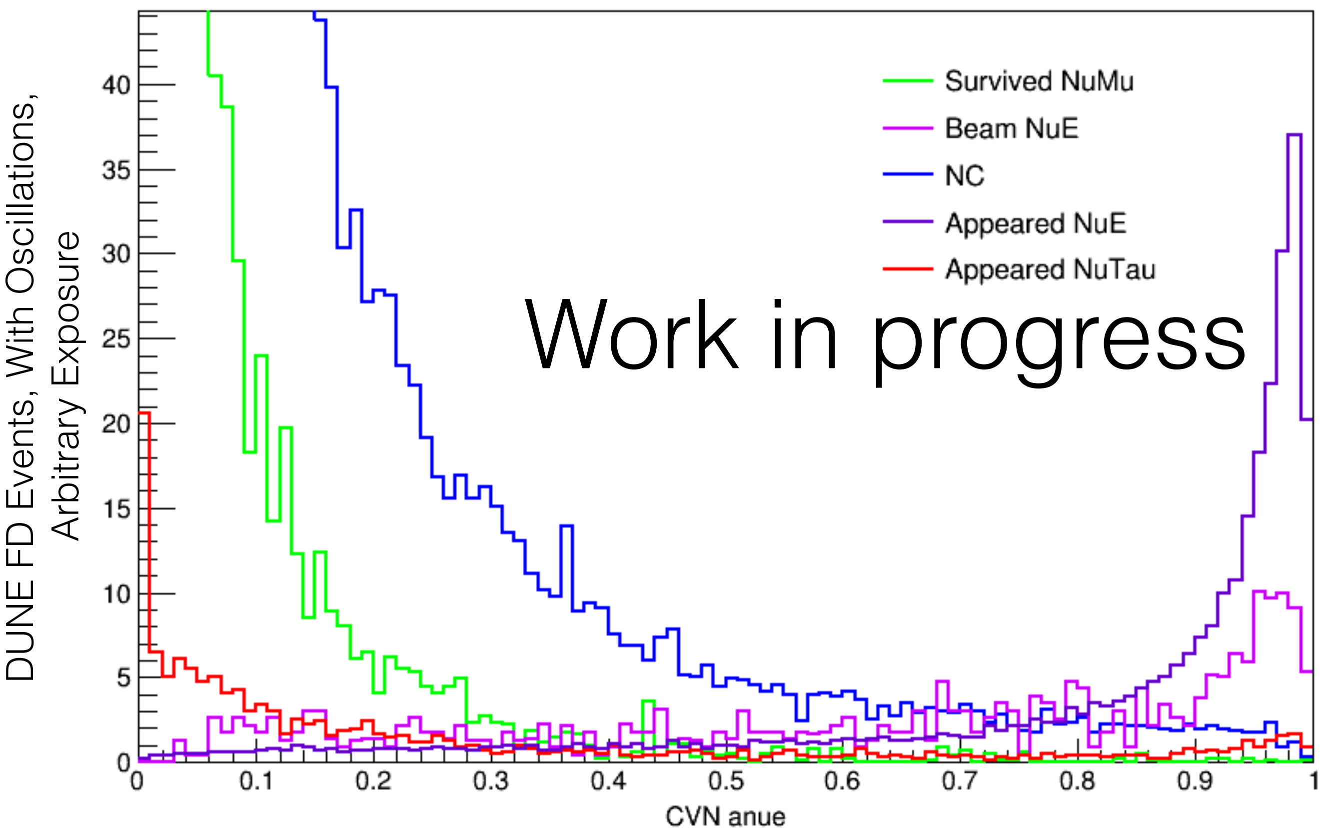
	NuMu	Appeared NuE	Beam NuE	NC	NuTau
Efficiency	**87.7**				
Rejection		99.6	99.3	**98.3**	81.4

NuE PID

Neutrino Beam



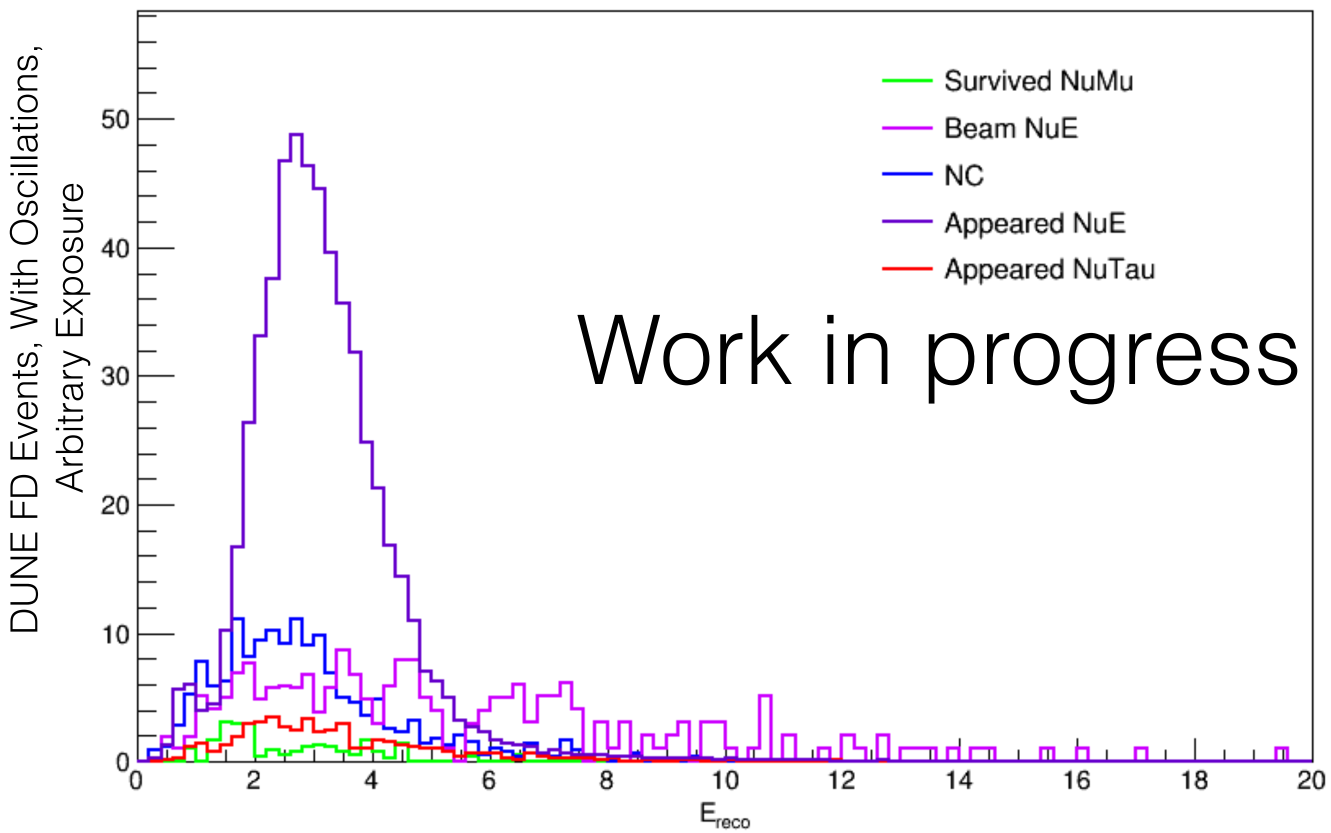
Anti-Neutrino Beam



Cut at 0.8, optimized for S/Sqrt(S+B)

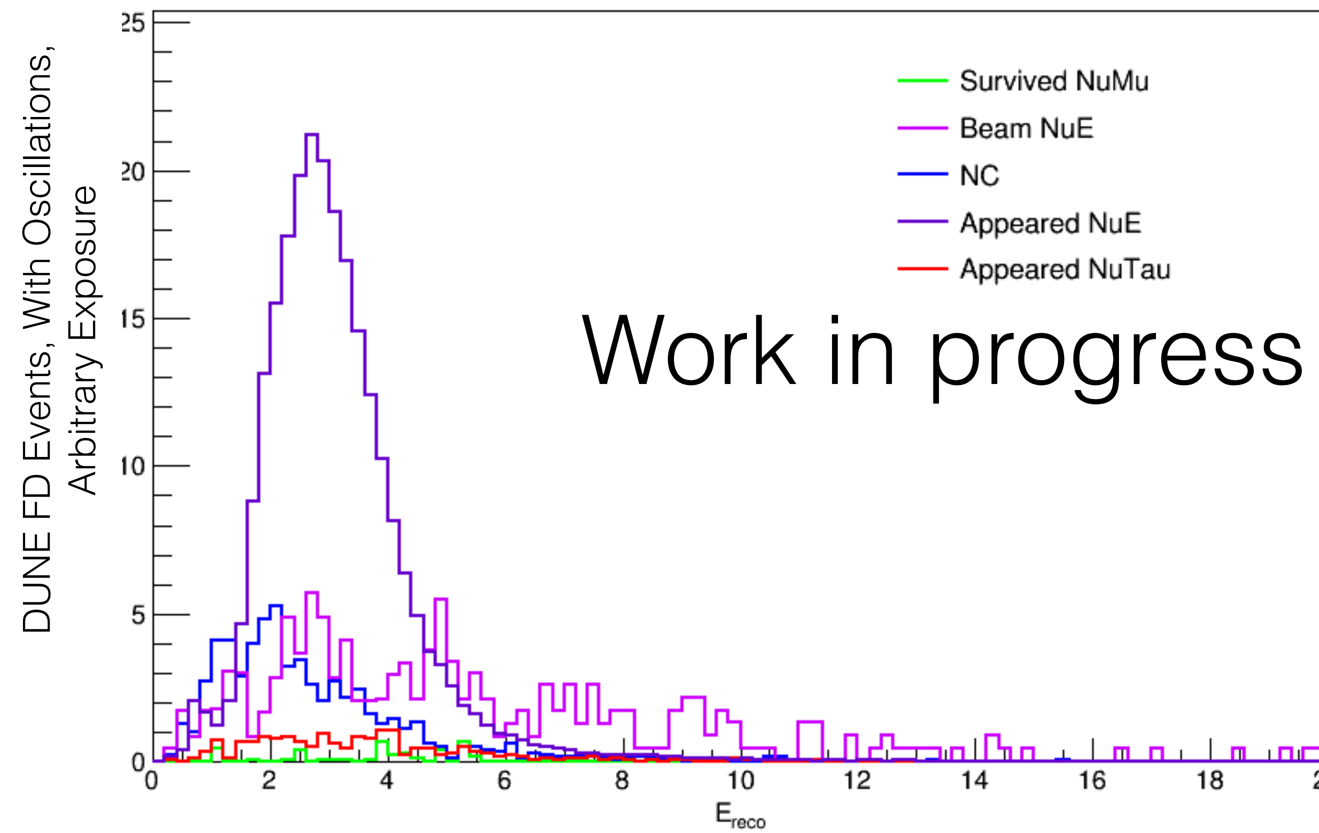
NuE Selected Events, Reconstructed Energy Spectra

Neutrino Beam



	Appeared NuE	NuMu	Beam NuE	NC	NuTau
Efficiency	**67.5**				
Rejection		99.8	52.1	**98.6**	85.8

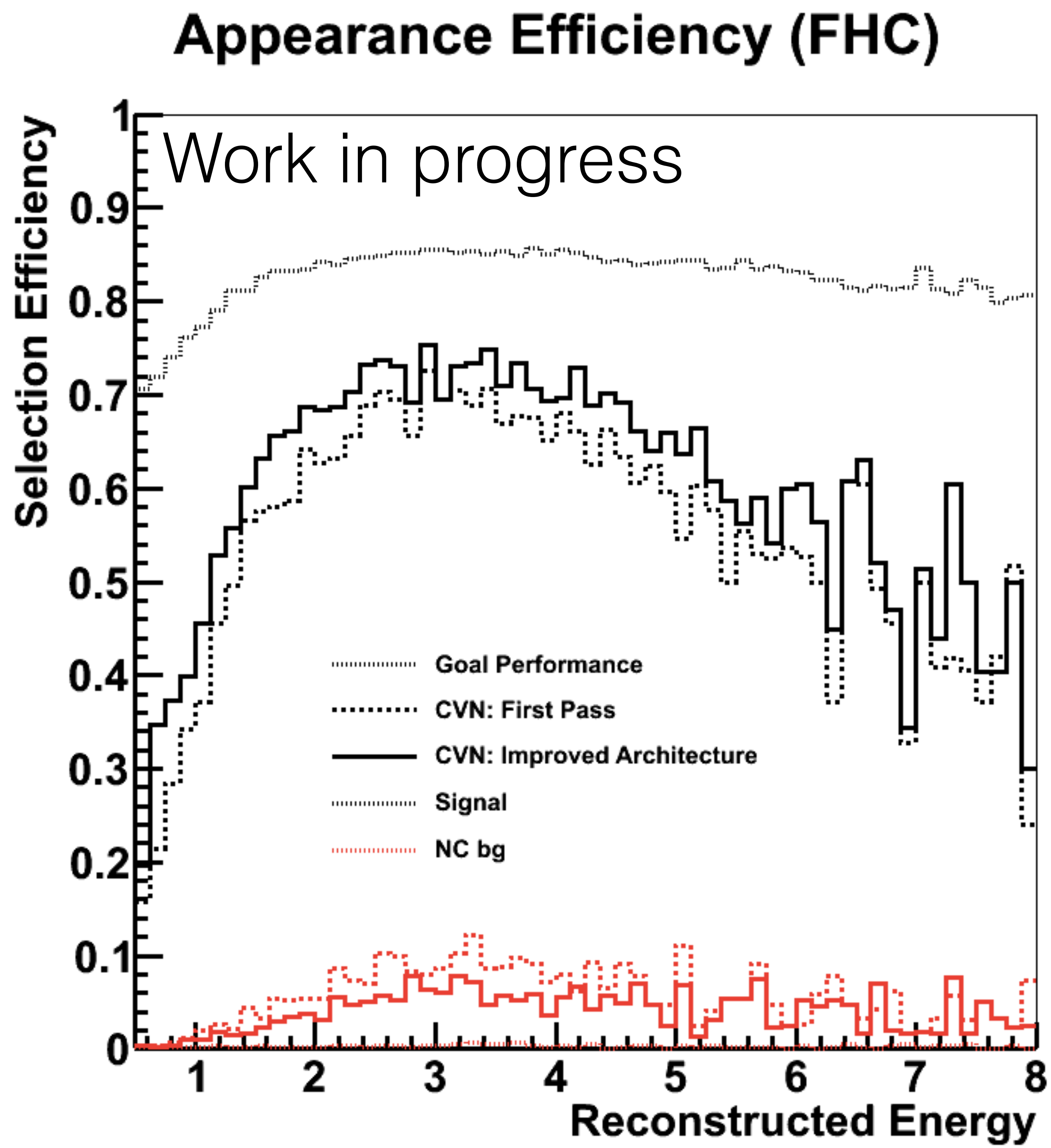
Anti-Neutrino Beam



	Appeared NuE	NuMu	Beam NuE	NC	NuTau
Efficiency	**79.3**				
Rejection		99.9	48.2	**98.8**	87.6

The Bottom Line

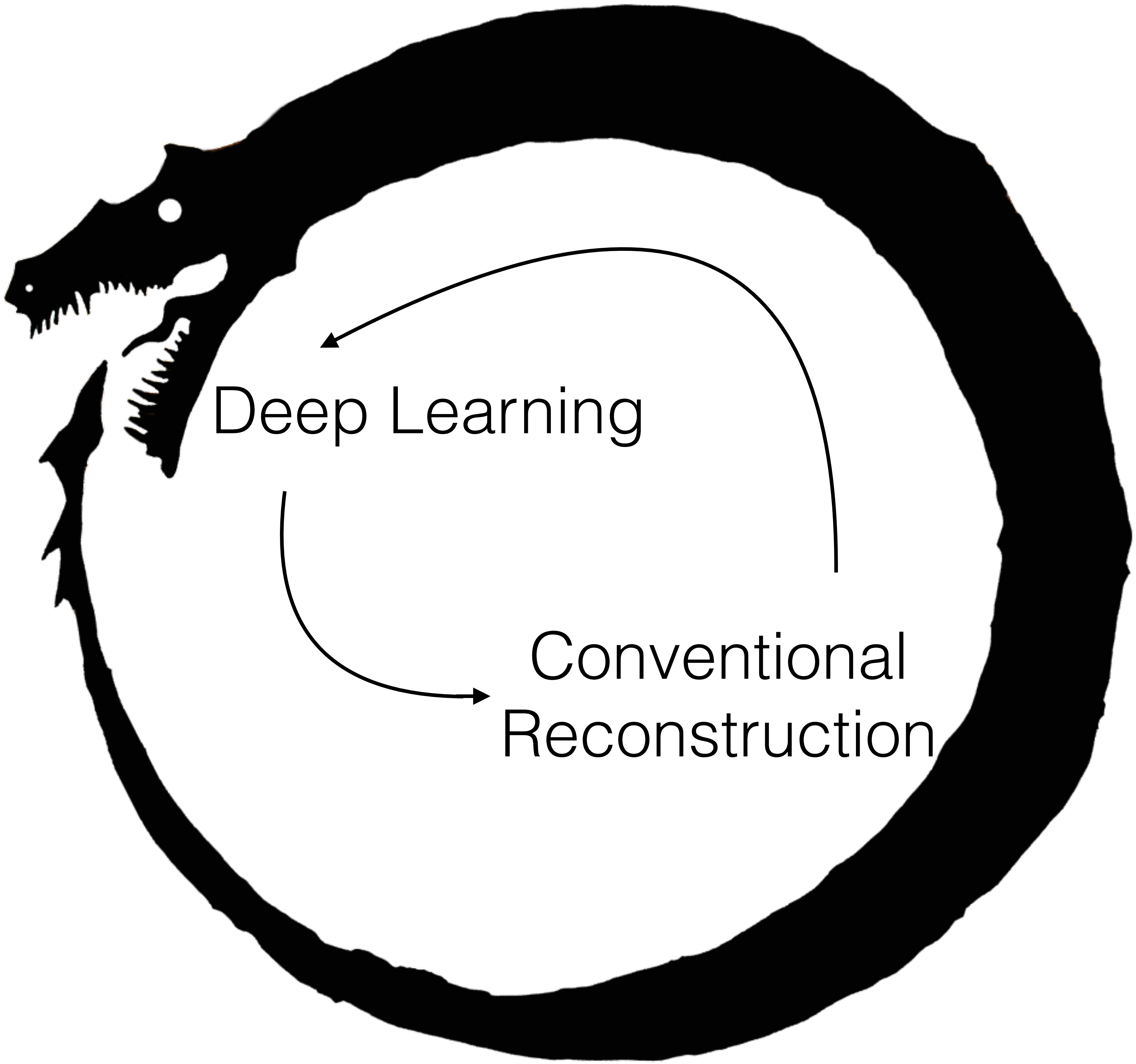
Excellent efficiency already achieved, rapidly making progress towards the TDR goals.



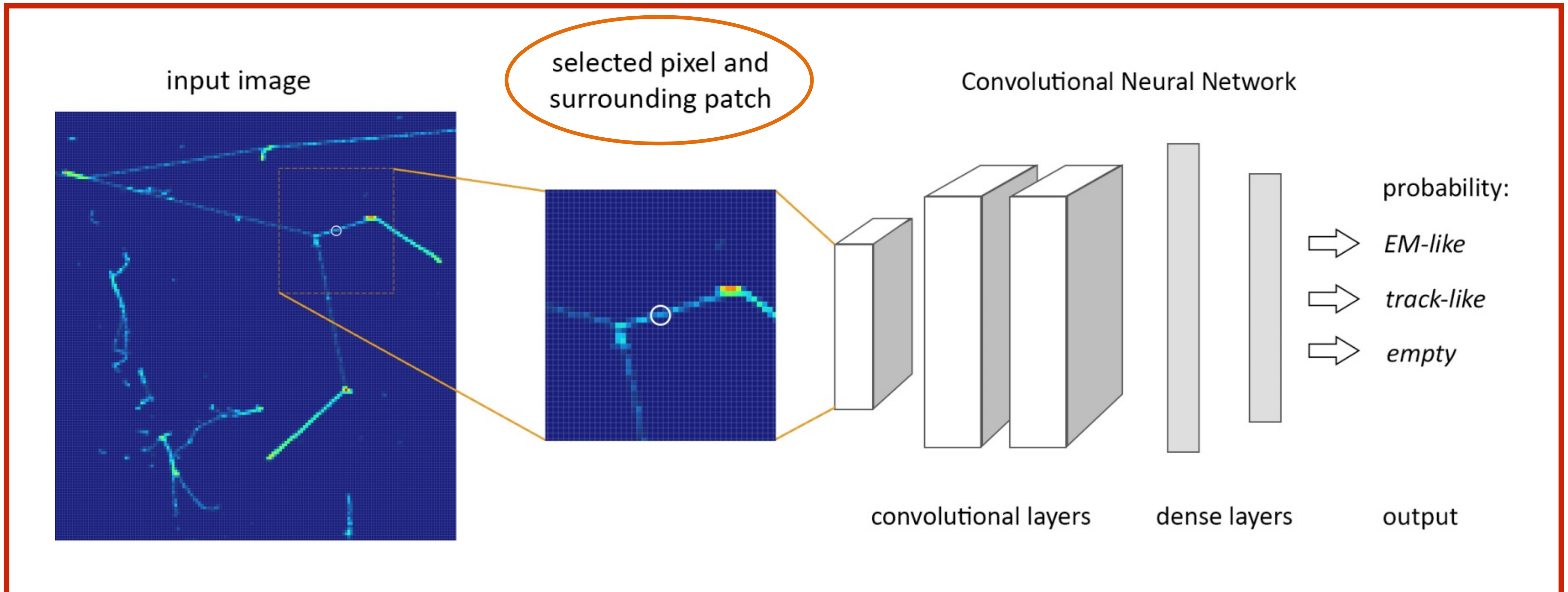
Deep Learning for Event Reconstruction

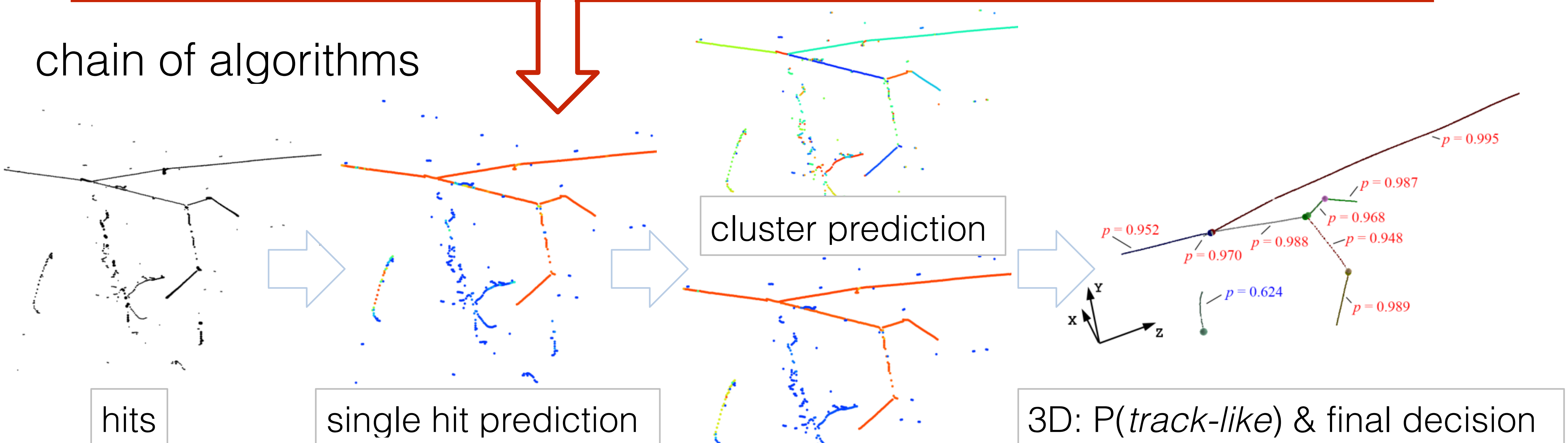
The original dream

Where we're really going



CNNs For Hit Level ID

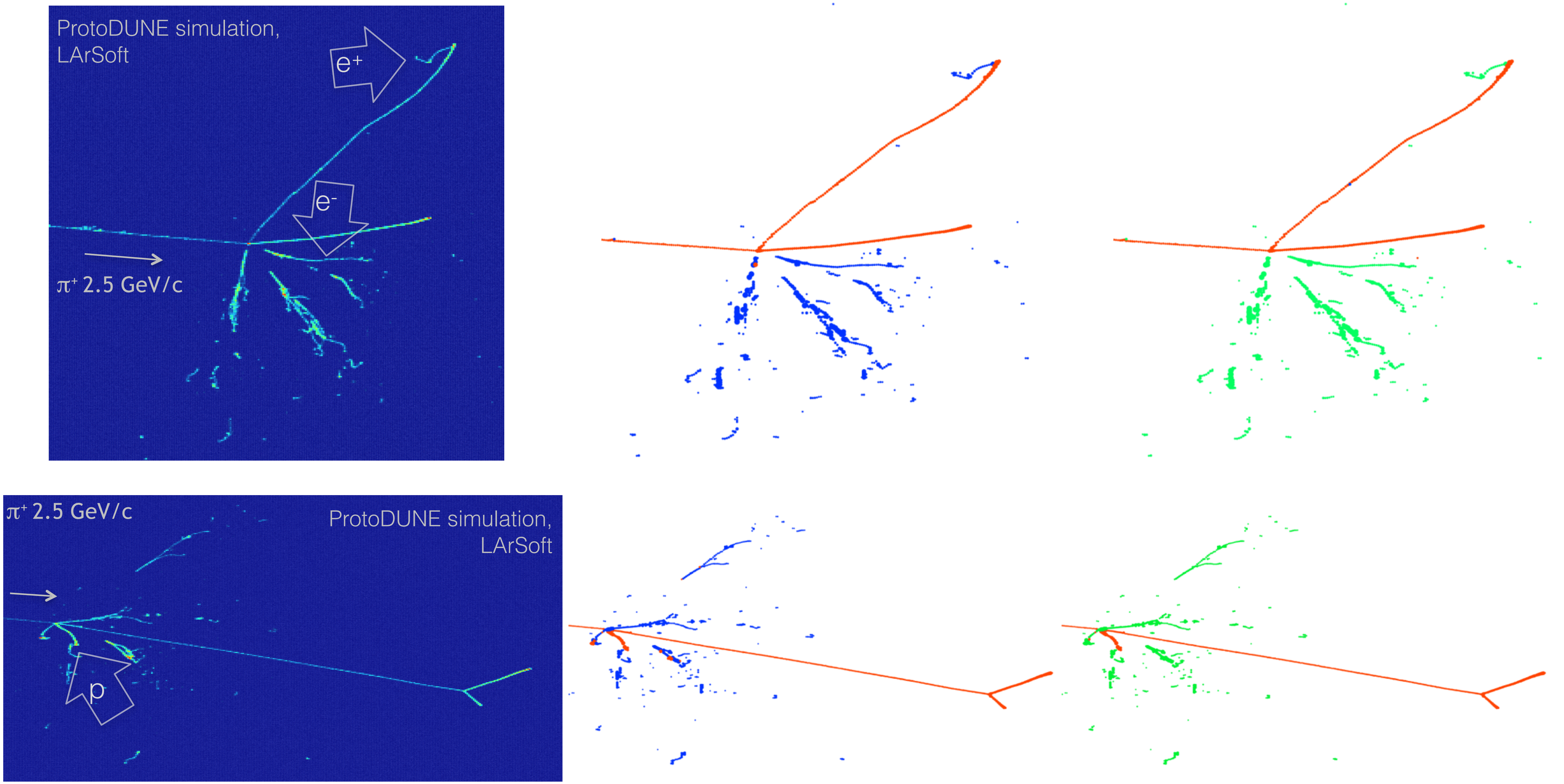




ProtoDUNE simulation, LArSoft. *Gauss hit finder* for hits, *linecluster* for 2D clusters, and *PMA* for 3D tracking/vertexing is used.

CNNs For Hit Level ID

EM / track separation: examples of ProtoDUNE events



input: 2D ADC

CNN output:
EM-like (blue) / track-like (red)

MC truth:
EM-like (green) / track-like (red)

Event displays: R.Sulej, Connecting The Dots / Intelligent Trackers, May 2017, LAL-Orsay, France

Conclusions

Active part of the rapid development of deep learning tools for liquid argon TPCs (see previous, excellent, talk).

Early attempts at taking the event classification work pioneered at NOvA to DUNE already show excellent performance, rapidly closing in on the TDR targets.

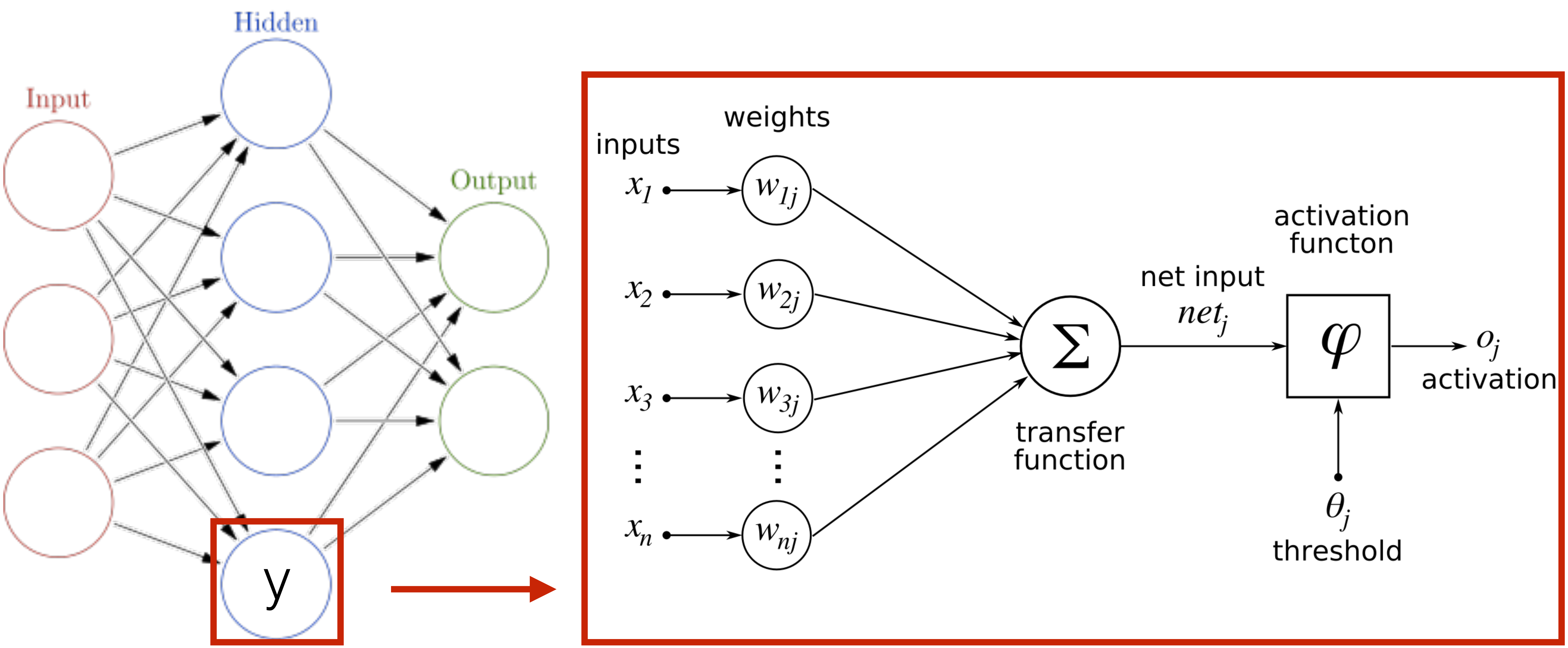
Exciting working beyond event classification, building tools which might help solve the difficult problem of liquid argon reconstruction.

Just the tip of the iceberg! Huge amounts of room to optimize our classification network, and to explore other possibilities.

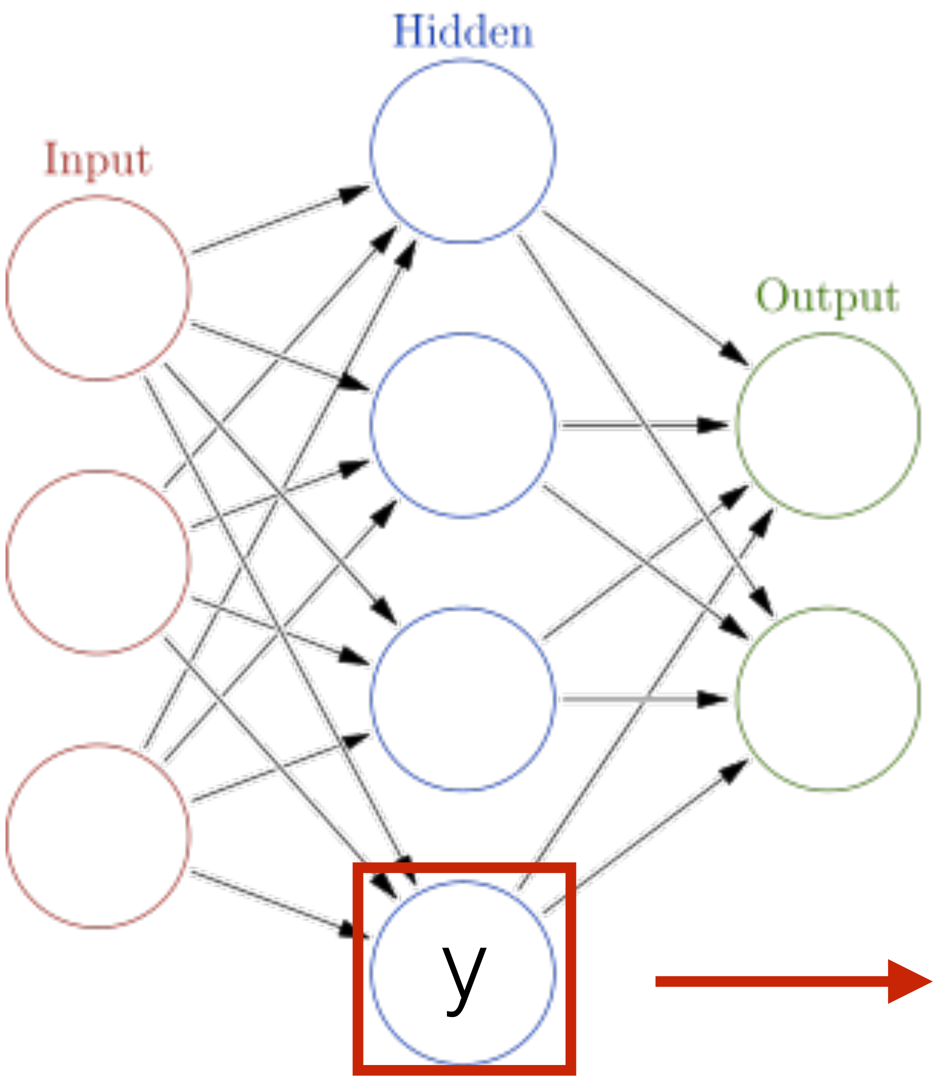
Q&A

Many thanks to the DUNE collaboration, Fermilab National Accelerator laboratory, and to the National Science Foundation.

Neural Networks



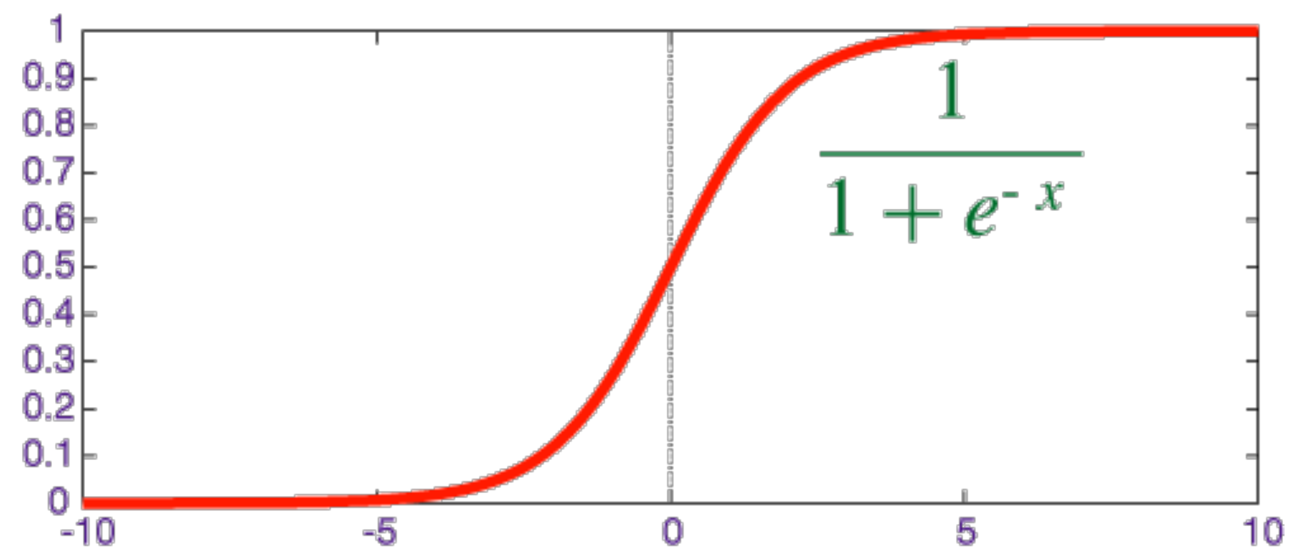
Neural Networks



$$x = input\ vector$$

$$y = \sigma\left(Wx + b\right)$$

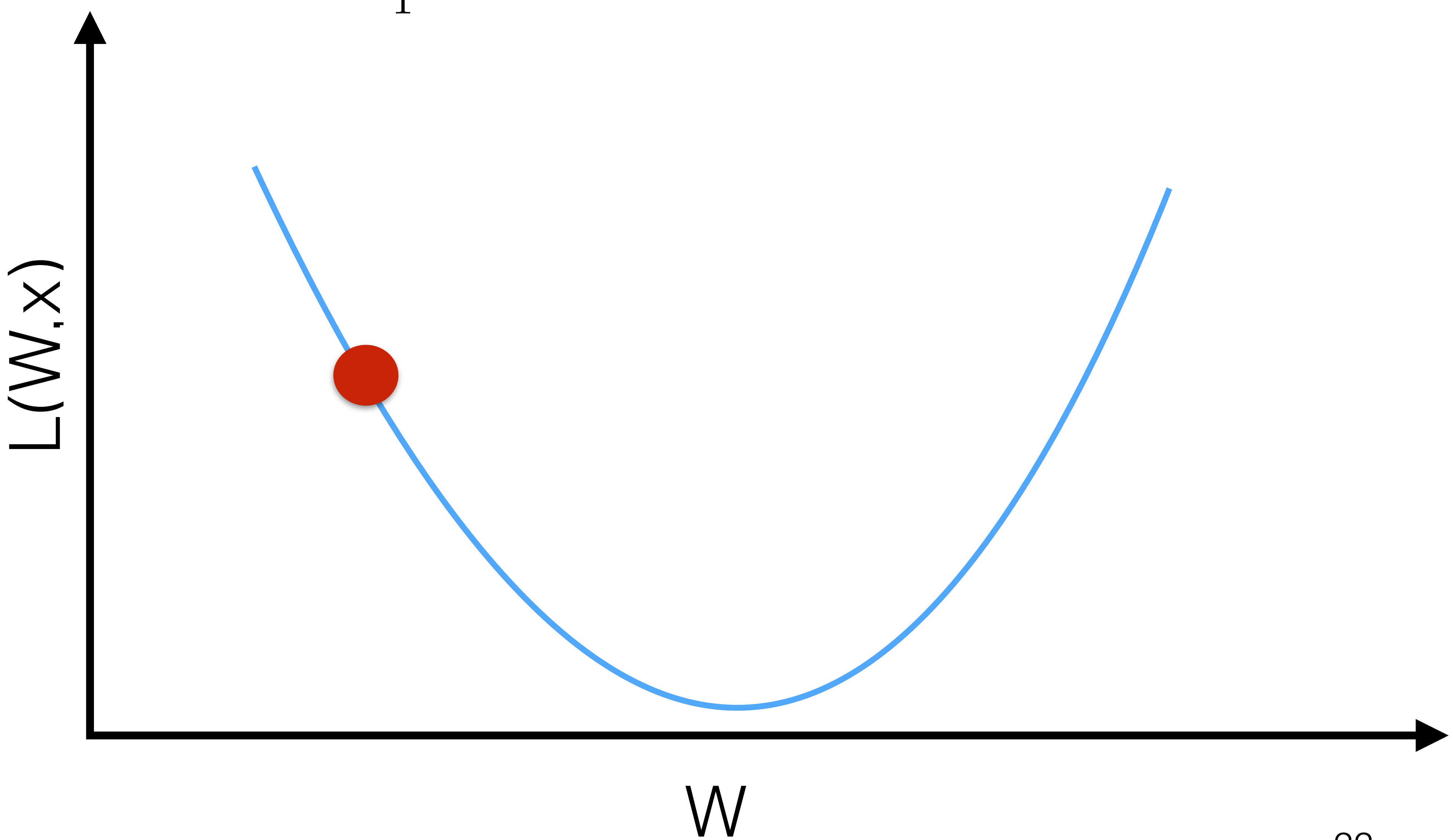
$$\sigma =$$



Training A Neural Network

Start with a "Loss" function which characterizes the performance of the network. For supervised learning:

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



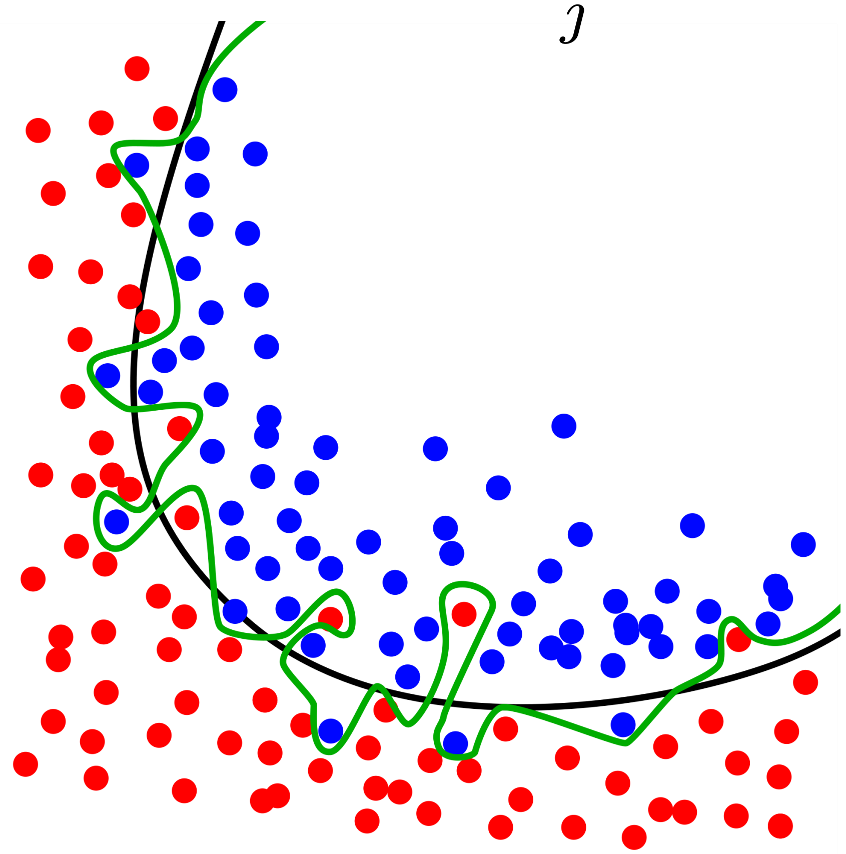
Training A Neural Network

Start with a "Loss" function which characterizes the performance of the network. For supervised learning:

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

Add in a regularization term to avoid overfitting:

$$L' = L + \frac{1}{2}\sum_j w_j^2$$



Training A Neural Network

Start with a "Loss" function which characterizes the performance of the network. For supervised learning:

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

Add in a regularization term to avoid overfitting:

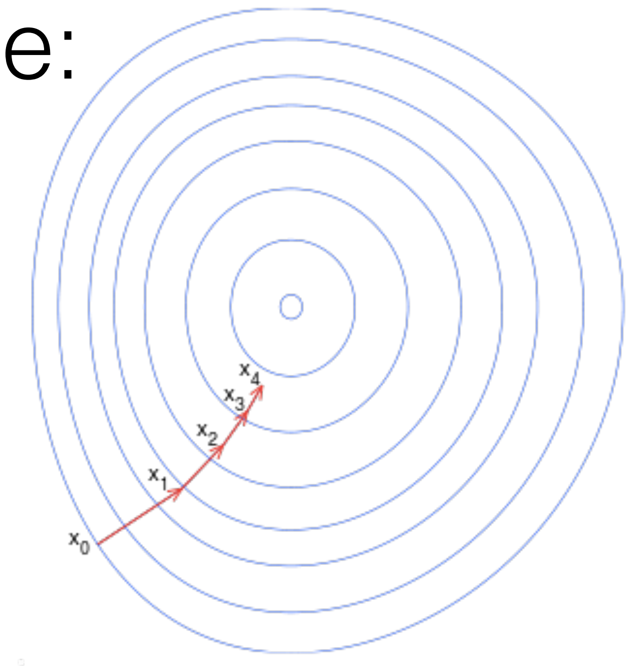
$$L' = L + \frac{1}{2} \sum_{j} w_j^2$$

Propagate the gradient of the network back to specific nodes using back propagation. AKA apply the chain rule:

$$\nabla_{w_j} L = \frac{\delta L}{\delta f} \frac{\delta f}{\delta g_n} \frac{\delta g_n}{\delta g_{n-1}} \ldots \frac{\delta g_{k+1}}{\delta g_k} \frac{\delta g_k}{\delta w_j}$$

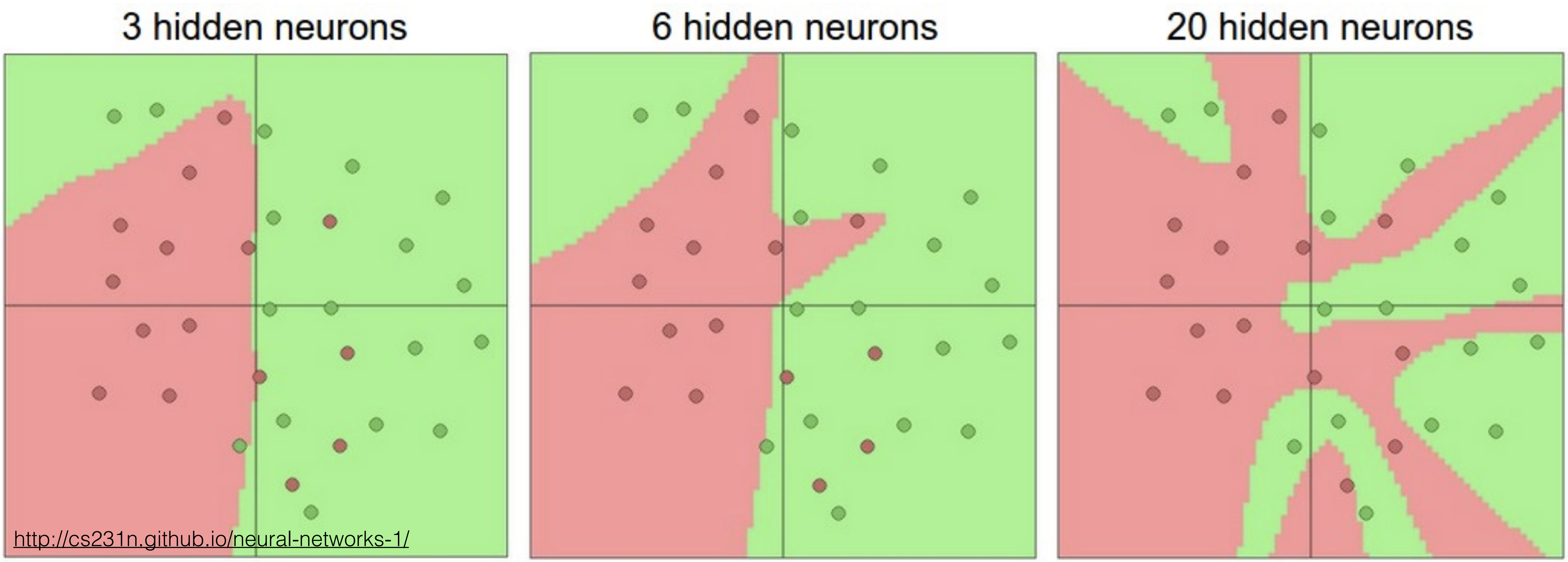
Update weights using gradient descent:

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



Deep Neural Networks

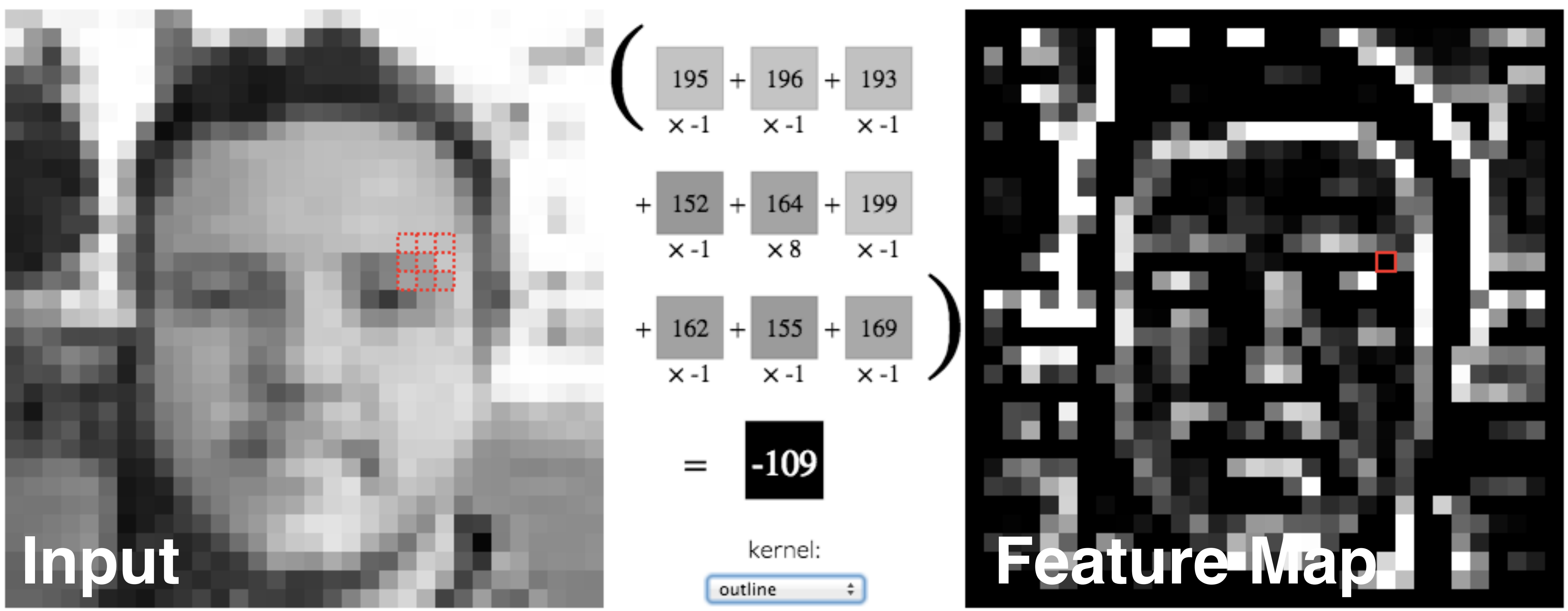
What if we try to keep all the input data? Why not rely on a wide, extremely Deep Neural Network (DNN) to learn the features it needs? Sufficiently deep networks make excellent function approximators:



Possible to train now with new activation functions, GPUs etc.

Convolutional Neural Networks

Instead of training a weight for every input pixel, try learning weights that describe kernel operations, convolving that kernel across the entire image to exaggerate useful features. Inspired by research showing that cells in the visual cortex are only responsive to small portions of the visual field.

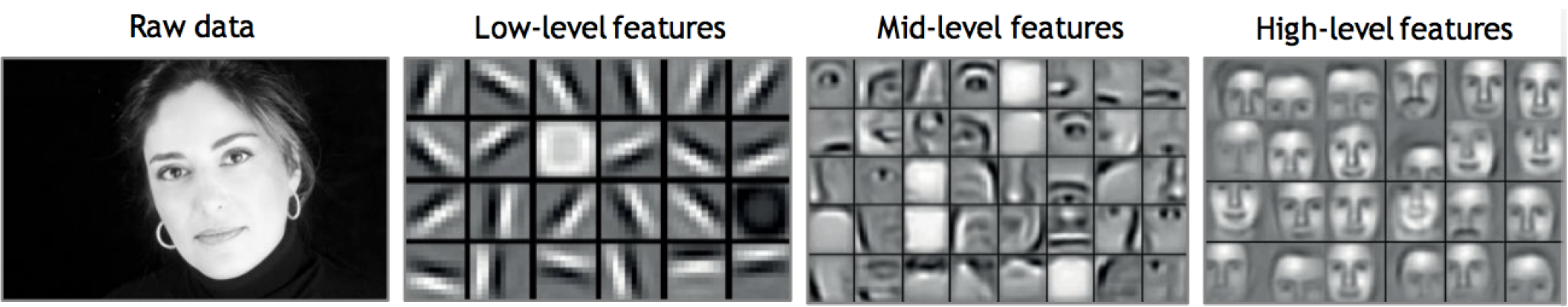


Kernel

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

Convolutional Neural Networks

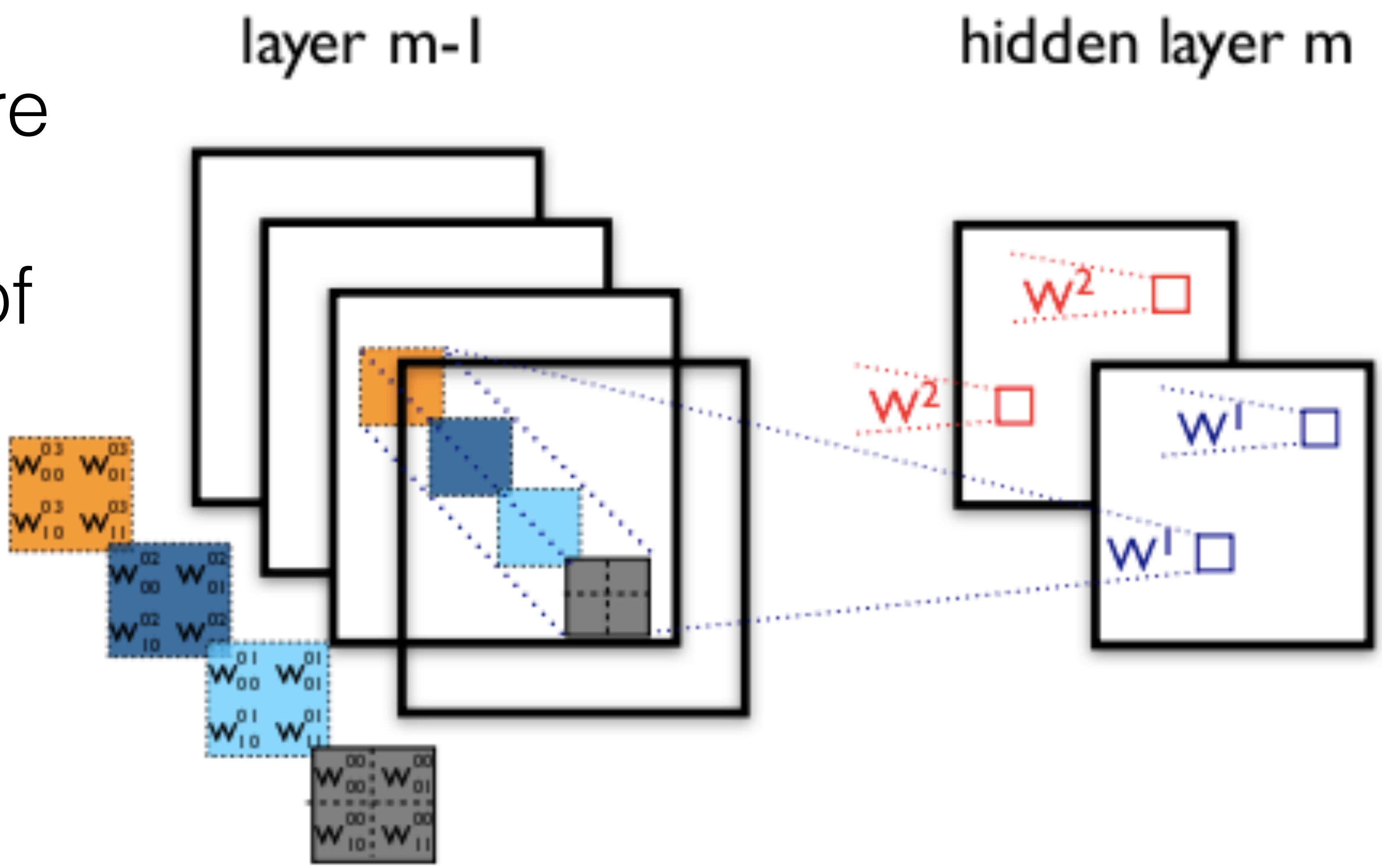
Instead of training a weight for every input pixel, try learning weights that describe kernel operations, convolving that kernel across the entire image to exaggerate useful features. Inspired by research showing that cells in the visual cortex are only responsive to small portions of the visual field.



https://developer.nvidia.com/deep-learning-courses

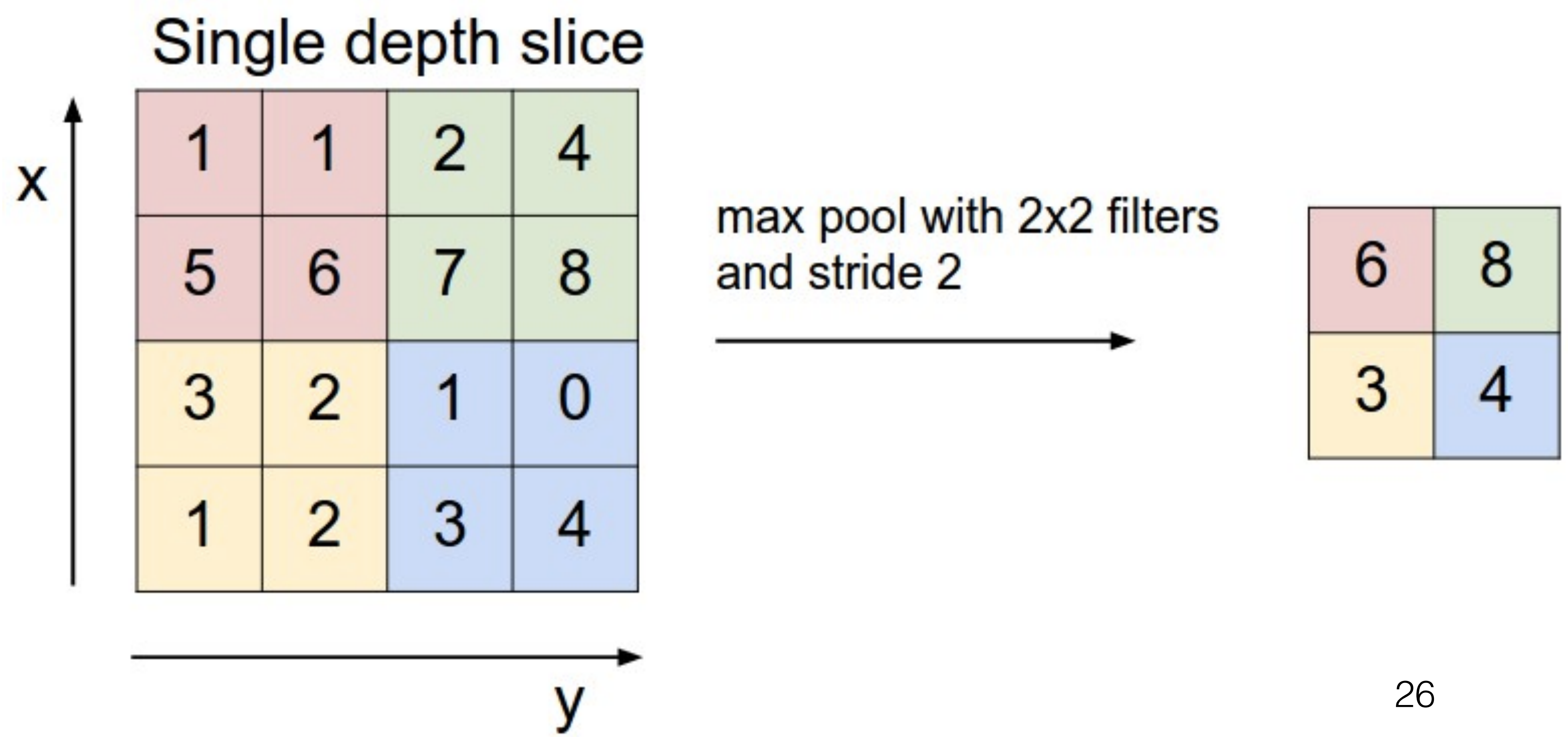
Convolutional Layers

- Every trained kernel operation is the same across an entire input image or feature map.
- Each convolutional layer trains an array of kernels to produce output feature maps.
- Weights for a given convolutional layer are a 4D tensor of NxMxHxW (number of incoming features, number of outgoing features, height, and width)



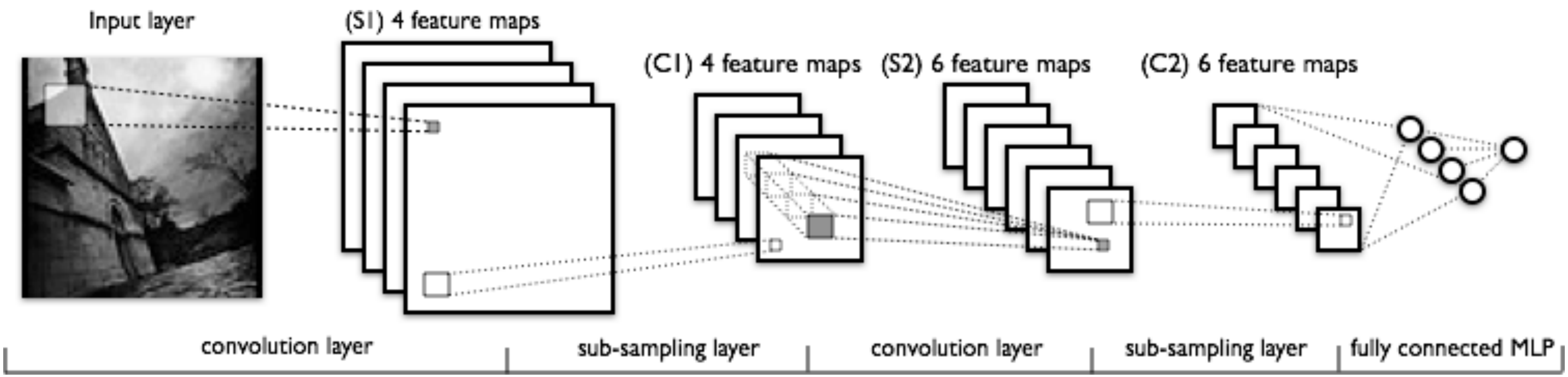
Pooling Layers

- Intelligent downscaling of input feature maps.
- Stride across images taking either the maximum or average value in a patch.
- Same number of feature maps, with each individual feature map shrunk by an amount dependent on the stride of the pooling layers.



The LeNet

In its simplest form a convolutional neural network is a series of convolutional, max pooling, and MLP layers:

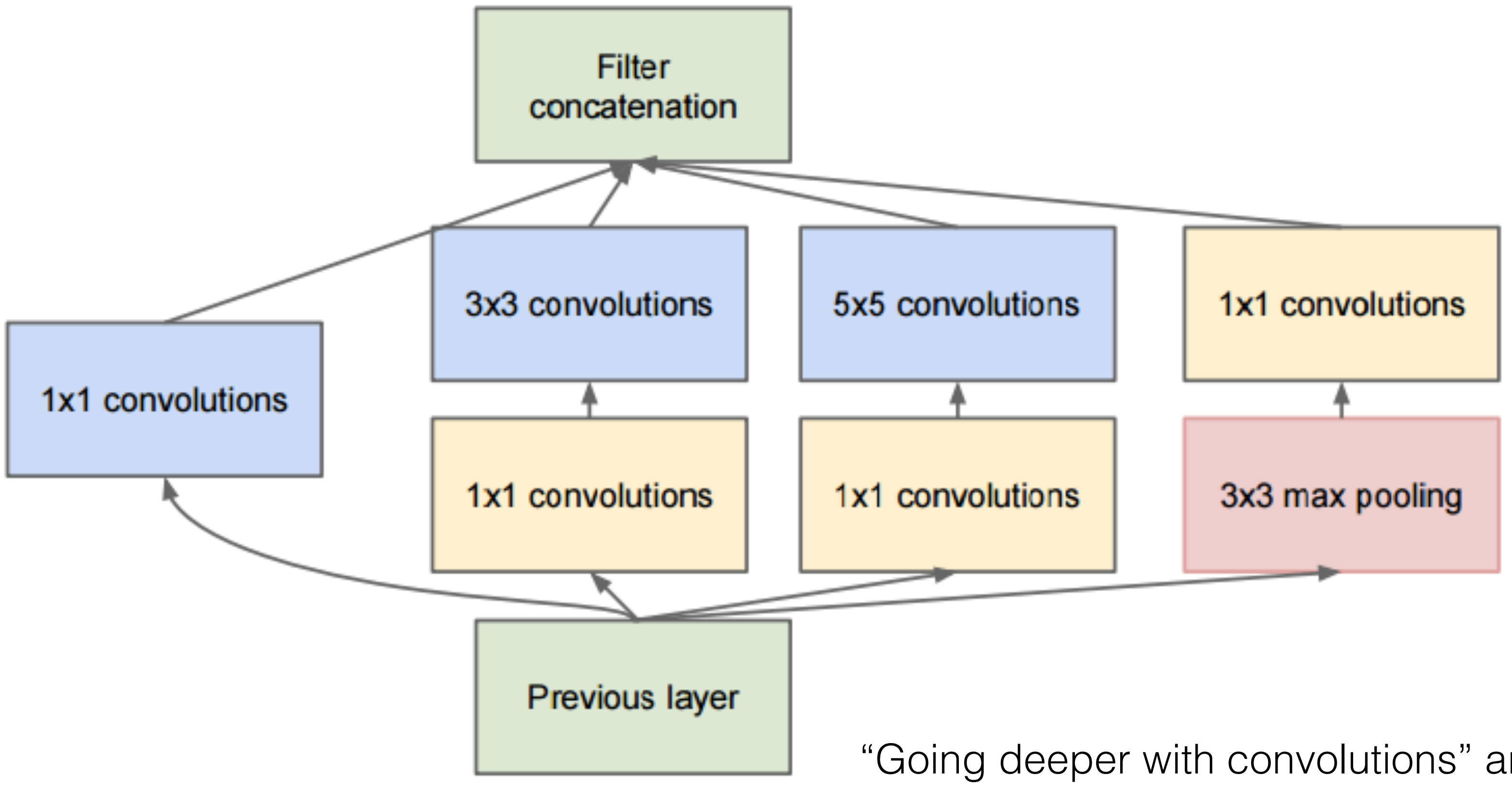


The "LeNet" circa 1989

http://deeplearning.net/tutorial/lenet.html http://yann.lecun.com/exdb/lenet/

Modern CNNs

Renaissance in CNN use over the last few years, with increasingly complex network-in-network models that allow for deeper learning of more complex features.



"Going deeper with convolutions" arXiv:1409.4842

The brilliance of this inception module is that it uses kernels of several sizes but keeps the number of feature maps under control by use of a 1x1 convolution.

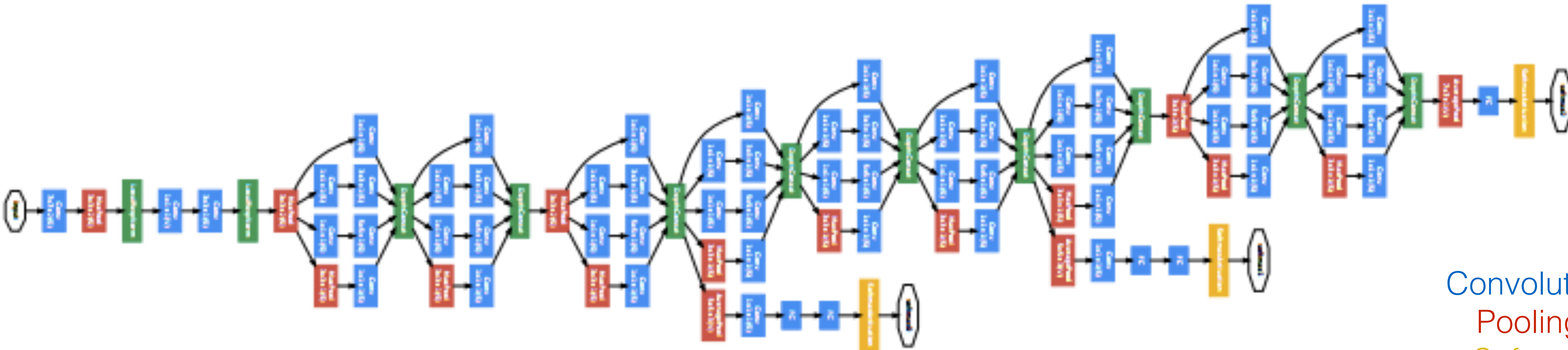
Modern CNNs

Renaissance in CNN use over the last few years, with increasingly complex network-in-network models that allow for deeper learning of more complex features.

The brilliance of this inception module is that it uses kernels of several sizes but keeps the number of feature maps under control by use of a 1x1 convolution.

Modern CNNs

Renaissance in CNN use over the last few years, with increasingly complex network-in-network models that allow for deeper learning of more complex features.

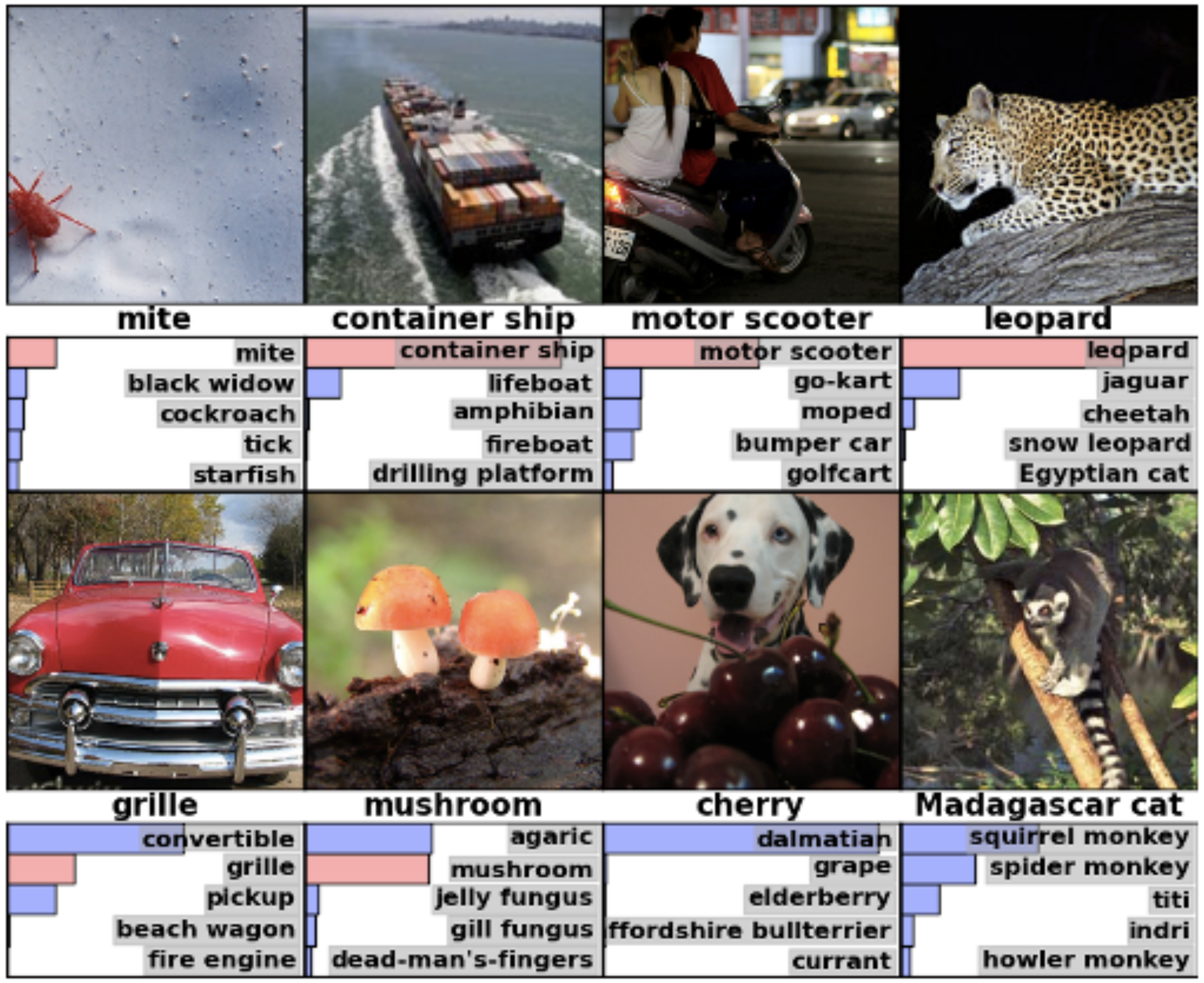


Convolution
Pooling
Softmax
Other

The "GoogleNet" circa 2014

The brilliance of this inception module is that it uses kernels of several sizes but keeps the number of feature maps under control by use of a 1x1 convolution.

Superhuman Performance



Some examples from one of the early breakout CNNs. Googles latest “Inception-v4” net achieves 3.46% top 5 error rate on the image net dataset. Human performance is at ~5%.