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 Neural Networks
Convolutional Neural Networks
Recurrent Neural Networks
Unsupervised Learning
Adversarial Networks
Neural Turing Machines
Deep Learning

Deep Neural Networks

Convolutional Neural Networks

Recurrent Neural Networks

Unsupervised Learning

Adversarial Networks

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

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

Monte Carlo with oscillations

Monte Carlo without Oscillations
Measuring neutrino oscillations is all about measuring how neutrinos change between different lepton flavor states as a function of distance traveled and neutrino energy.

\[ P(\nu_\mu \rightarrow \nu_e) \approx \left| \sqrt{P_{\text{atm}}} e^{-i\left(\frac{\Delta m_{32}^2 L}{4E} + \delta_{\text{CP}}\right)} + \sqrt{P_{\text{sol}}} \right|^2 \]

\[ P_{\text{atm}} = \sin^2 \theta_{23} \sin^2 2\theta_{13} \sin^2 \frac{\Delta m_{31}^2 L}{4E} \]

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

**Why Deep Neural Networks?**

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.
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.
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.

Work in progress
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 \{v_μ, v_e, v_τ \} \times \{QE,RES,DIS\}, NC.

Built in the excellent CAFFE framework.
No sign of overtraining - exceptional training test set performance agreement!
Here the earliest convolutional layer in the network starts by pulling out primitive shapes and lines.

Already “showers” and “tracks” are starting to form.
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.
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.
Cut at 0.5, guarantees no double counting due to softmax output of CVN
NuMu Selected Events, Reconstructed Energy Spectra

Neutrino Beam

<table>
<thead>
<tr>
<th></th>
<th>Survived NuMu</th>
<th>Beam NuE</th>
<th>NC</th>
<th>Appeared NuE</th>
<th>Appeared NuTau</th>
</tr>
</thead>
<tbody>
<tr>
<td>Efficiency</td>
<td>80.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rejection</td>
<td>99.0</td>
<td>98.7</td>
<td><strong>97.6</strong></td>
<td>81.5</td>
<td></td>
</tr>
</tbody>
</table>

Anti-Neutrino Beam

<table>
<thead>
<tr>
<th></th>
<th>Survived NuMu</th>
<th>Beam NuE</th>
<th>NC</th>
<th>Appeared NuE</th>
<th>Appeared NuTau</th>
</tr>
</thead>
<tbody>
<tr>
<td>Efficiency</td>
<td>87.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rejection</td>
<td>99.6</td>
<td>99.3</td>
<td><strong>98.3</strong></td>
<td>81.4</td>
<td></td>
</tr>
</tbody>
</table>

Work in progress

Alexander Radovic  Deep Learning at DUNE  13
NuE PID

Neutrino Beam

Anti-Neutrino Beam

Cut at 0.8, optimized for $S/\sqrt{S+B}$

Work in progress
NuE Selected Events, Reconstructed Energy Spectra

**Neutrino Beam**

- Efficiency: 67.5
- Rejection: 99.8, 52.1, 98.6, 85.8

**Anti-Neutrino Beam**

- Efficiency: 79.3
- Rejection: 99.9, 48.2, 98.8, 87.6

Work in progress
Excellent efficiency already achieved, rapidly making progress towards the TDR goals.
Deep Learning for Event Reconstruction
Reconstruction?

Where we’re going, we don’t need reconstruction.
Where we’re really going

Deep Learning

Conventional Reconstruction
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.
Many thanks to the DUNE collaboration, Fermilab National Accelerator laboratory, and to the National Science Foundation.
Neural Networks

![Diagram of a neural network with input, hidden, and output layers. The diagram includes nodes connected by weighted edges, with an output node labeled with \( y \).]
Neural Networks

$x = \text{input vector}$

$y = \sigma (W x + b)$

$\sigma = \frac{1}{1 + e^{-x}}$
Start with a “Loss” function which characterizes the performance of the network. For supervised learning:

\[
L(W, X) = \frac{1}{N} \sum_{i=1}^{N_{examples}} -y_i \log(f(x_i)) - (1 - y_i) \log(1 - f(x_i))
\]
Start with a “Loss” function which characterizes the performance of the network. For supervised learning:

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

Add in a regularization term to avoid overfitting:

$$L' = L + \frac{1}{2} \sum_j w_j^2$$
Start with a “Loss” function which characterizes the performance of the network. For supervised learning:

\[
L(W, X) = \frac{1}{N} \sum_{1}^{N_{\text{examples}}} -y_i \log(f(x_i)) - (1 - y_i) \log(1 - f(x_i))
\]

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:

\[
w'_j = w_j - \alpha \nabla_{w_j} L
\]
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:

3 hidden neurons

6 hidden neurons

20 hidden neurons


Possible to train now with new activation functions, GPUs etc.
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.
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 \( N \times M \times H \times W \) (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/
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.
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 “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.
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%.