Deep Learning in High Energy Physics

Michael Kagan

SLAC

11th Hadron Collider Physics Summer School, Fermilab
August 19, 2016
Outline

• Machine Learning (ML) and High Energy Physics (HEP)

• Basics of Neural Networks

• Deep learning

• Deep learning in HEP

• Future Directions
Aspects of Machine Learning (ML) in HEP

• **Optimization**
  – Bottom line is performance
  – But can we build new better (simple?) features?

• **Teaching the learning**
  – Guide and boost performance of ML algorithms using physics knowledge (i.e. domain specific knowledge)
  – We don’t want ML to relearn special relativity

• **Learning from Learning** ...(if we can)
  – Can we extract information about what the ML is learning?
  – Can we use this information to design new variables?
  – Often visualization is a key component
What is Machine Learning?

• Giving computers the ability to learn without explicitly programming them (Arthur Samuel, 1959)

• Statistics + Algorithms

• Computer Science + Probability + Optimization Techniques

• Fitting data with complex functions

• Pattern recognition: identifying patterns and regularities in data
What do we use ML for?

• **Supervised Learning**
  – Given data with variables / features \( \{x_i \in X\} \) and targets \( \{y_i \in Y\} \), learn the function mapping \( f(X)=Y \)

  – **Classification**: \( Y \) is a finite set of **labels**
  – **Regression**: \( Y \in \text{Real Numbers} \)

• **Unsupervised Learning**
  – Given some data \( D=\{x_i \in X\} \), but no labels, find structure in the data

  – **Clustering**: partition the data into groups
    \( D=\{D_1 \cup D_2 \cup D_3 \ldots \cup D_k\} \)
  – **Dimensionality reduction**: find a low dimensional (less complex) representation of the data with a mapping \( Z=h(X) \)

• **Reinforcement learning**
  – Learn to make the best sequence of decisions to achieve a given goal when feedback is delayed until you reach the goal
What do we use ML for?

• **Supervised Learning**
  – Given data with variables / features \( \{x_i \in X\} \) and targets \( \{y_i \in Y\} \), learn the function mapping \( f(X) = Y \)
  
  – **Classification**: \( Y \) is a finite set of **labels**
  – **Regression**: \( Y \in \text{Real Numbers} \)

• **Unsupervised Learning**
  – Given some data \( D = \{x_i \in X\} \), but no labels, find structure in the data
  
  – **Clustering**: partition the data into groups
  
  – **Dimensionality reduction**: find a low dimensional (less complex) representation of the data with a mapping \( Z = h(X) \)

  Won’t Discuss this today… But there are existing and future applications in HEP

• **Reinforcement learning**
  – Learn to make the best sequence of decisions to achieve a given goal when feedback is delayed until you reach the goal

  Won’t Discuss this at all today… Not yet clear how it will be used in HEP
Supervised Learning

- Design function with adjustable parameters
- Use a labeled training-set to compute error
- Adjust parameters to reduce error function
- Repeat until parameters stabilize
- Estimate final performance on test-set

True labels:
- Higgs = 1
- Bkg = 0

Y. Le Cun
Classification

• Learn a function to separate different classes of data

• Avoid over-fitting:
  – Learning too fine details about your training sample that will not generalize to unseen data
Machine Learning Applied Widely in HEP

• **In analysis:**
  – Classifying signal from background, especially in complex final states
  – Reconstructing heavy particles and improving the energy / mass resolution

• **In reconstruction:**
  – Improving detector level inputs to reconstruction
  – Particle identification tasks
  – Energy / direction calibration

• **In the trigger:**
  – Quickly identifying complex final states

• **In computing:**
  – Estimating dataset popularity, and determining how number and location of dataset replicas
Many recent applications of ML in HEP rely on Ensembles of decision trees, such as Boosted Decision Trees and Random Forests.

Powerful algorithms that are relatively simple, easy to train, and tend not to overfit (especially Random Forests).

They are very popular in general:
- Test 179 classifiers (no deep neural networks) on 121 datasets
- The classifiers most likely to be the bests are the random forest (RF) versions, the best of which (…) achieves 94.1% of the maximum accuracy overcoming 90% in the 84.3% of the data sets

But, Deep Neural Networks have outperformed such algorithms in certain domains, like Object Recognition in images.
Neural Networks

• “Typical” neural network circa 2005

• Typical questions of optimization
  – Which variables to choose as inputs?  How correlated are they?
  – How many nodes in the hidden layer?

$x = \text{input vector}$
Neural Networks

- “Typical” neural network circa 2005
- Typical questions of optimization
  - Which variables to choose as inputs? How correlated are they?
  - How many nodes in the hidden layer?

\[
y = \sigma(Uz + c) \\
z = \sigma(Wx + b)
\]

\[\sigma(x) = \frac{1}{1 + e^{-x}}\]

\(x = \text{input vector}\)

\(z = \sigma(Wx + b)\)

\(y = \sigma(Uz + c)\)
Training a Neural Network

- Define a **loss function** that depends on predictions $f(x;w)$ and targets $y$

\[
L_{MSE} = \frac{1}{N} \sum_{i=1}^{N_{\text{examples}}} (y_i - f(x_i))^2
\]

\[
L_{BCE} = \frac{1}{N} \sum_{i=1}^{N_{\text{examples}}} -y_i \log f(x_i) - (1 - y_i) \log(1 - f(x_i))
\]
Training a Neural Network

- Define a **loss function** that depends on predictions $f(x;w)$ and targets $y$

\[
L_{MSE} = \frac{1}{N} \sum_{i=1}^{N_{\text{examples}}} (y_i - f(x_i))^2
\]

\[
L_{BCE} = \frac{1}{N} \sum_{i=1}^{N_{\text{examples}}} -y_i \log f(x_i) - (1 - y_i) \log(1 - f(x_i))
\]

- Add **regularization** to control the model complexity and reduce overfitting

\[
L' = L + \frac{1}{2} \sum_j w_j^2
\]
Training a Neural Network

• Define a **loss function** that depends on predictions $f(x;w)$ and targets $y$

$$L_{MSE} = \frac{1}{N} \sum_{i=1}^{N_{\text{examples}}} (y_i - f(x_i))^2$$

$$L_{BCE} = \frac{1}{N} \sum_{i=1}^{N_{\text{examples}}} -y_i \log f(x_i) - (1 - y_i) \log(1 - f(x_i))$$

• Add **regularization** to control the model complexity and reduce overfitting

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

• Minimize the loss function using **backpropagation**
  - Fancy word for chain rule
  - Compute average gradient on training set

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

• Update weights with **gradient descent**

$$w_j \leftarrow w_j - \alpha \nabla_{w_j} L$$

  - $\alpha$ is called the learning rate
Deep Neural Networks

- As data complexity grows, need exponentially large number of neurons in a single-hidden-layer network to capture all the structure in the data

- Deep neural networks have many hidden layers
  - Factorize the learning of structure in the data across many layers

- Difficult to train, only recently has this become possible…
Why did it take so long to train DNN’s?

- **Big Data**
  - (Hundreds of) Millions of parameters → large dataset vital for training

- **GPU’s**
  - NN’s require a lot of matrix multiplications… perfect for GPU’s
  - Dramatically increased the speed of training

- But these aren’t the only reasons…
Training Improvements

• Gradient descent is computationally costly (since we compute gradient over full training set)

• Stochastic gradient descent
  – Compute gradient on one event at a time (in practice a small batch)
  – Noisy estimates average out
  – Stochastic behavior can allow “jumping” out of bad critical points

  – Scales well with dataset and model size
  – But can have some convergence difficulties

  – Improvements include: Momentum, RMSprop, AdaGrad, …
Better Activation Functions

- **Vanishing gradient problem**
  - Derivative of sigmoid:
    \[
    \frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x))
    \]
  - Nearly 0 when x is far from 0!
  - Gradient descent impossible!

- **Rectified Linear Unit (ReLU)**
  - ReLU(x) = max {0, x}
  - Derivative is constant!
    \[
    \frac{\partial \text{ReLU}(x)}{\partial x} = \begin{cases} 
      1 & \text{when } x > 0 \\
      0 & \text{otherwise}
    \end{cases}
    \]
  - ReLU gradient doesn’t vanish
Better Regularization Inside the Network

- **Dropout**
  - Randomly remove nodes during training
  - Avoid co-adaptation of nodes
  - Essentially a large model averaging procedure
Deep NNs in HEP analysis

• Compare dense Deep NN against BDT’s and shallow NN’s
  – small but statistically significant gain over simpler ML algorithms

• Deep NN found to outperform shallow NN and BDT’s
  – Typical physics variables are high performing (e.g. invariant mass, Razor, etc.)
  – But Deep NN’s can learn well from only 4-vector inputs

• Physicists are good at doing physics!
  – Typical physics variables are high performing (e.g. invariant mass, Razor, etc.)

H$\rightarrow$ττ benchmark

BSM Higgs benchmark

<table>
<thead>
<tr>
<th>Technique</th>
<th>Low-level AUC</th>
<th>High-level AUC</th>
<th>Complete AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDT</td>
<td>0.73 (0.01)</td>
<td>0.78 (0.01)</td>
<td>0.81 (0.01)</td>
</tr>
<tr>
<td>NN</td>
<td>0.733 (0.007)</td>
<td>0.777 (0.001)</td>
<td>0.816 (0.004)</td>
</tr>
<tr>
<td>DN</td>
<td>0.880 (0.001)</td>
<td>0.800 (&lt;0.001)</td>
<td>0.885 (0.002)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Discovery significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technique</td>
</tr>
<tr>
<td>NN</td>
</tr>
<tr>
<td>DN</td>
</tr>
</tbody>
</table>
What is deep learning doing?

• Hierarchical learning of representations

• Use low level inputs in smart ways
  – e.g. Feed in image pixels, rather than pre-computed features
  – Learn the structure in the data, rather than engineer it
  – No explicit need for feature engineering… unless you want to

• What deep learning is NOT:
  – A silver bullet
  – Replacement for thinking + domain knowledge
  – Always better than BDT, SVM, …
  – Just feedforward neural networks!
Higher Level Representations

- Successive layers build upon information learned in lower layers to construct progressively higher level representations of data

Optimal stimulus of a given neuron
http://arxiv.org/abs/1112.6209
NOT Simple Feedforward Neural Networks

- **NN’s as a complex graph**
  - Nodes of graph are the layers
  - Edges of graph are data flow
  - Layers added to achieve a specific task, e.g. regularization

- Better to ask:
  - What does each layer / module do?
  - How is it connected to the previous and next layer?

*Inception module*

"Network-in-network"

**GoogLeNet**

ILSVRC 2014 Winner

4M parameters
The Tip of the Iceberg

- Feedforward NNs
- Convolutional NNs
- Deep Belief Nets
- Recurrent NNs
- Recursive NNs
- Deep Q Learning
- Neural Turing Machines
- Memory NNs

Luke de Oliveira
The Tip of the Iceberg

State of the art in image recognition and computer vision tasks

Feedforward NNs
Convolutional NNs
Deep Belief Nets
Recurrent NNs
Recursive NNs
Deep Q Learning
Neural Turing Machines
Memory NNs

Luke de Oliveira
Typical Neural Network Hidden Layer

output

\[ z_1 \quad z_2 \quad z_3 \quad z_4 \]

Hidden layer
Different Colors represent different weights \( W^*x \)

input

\[ x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \]
Local connectivity: each neuron has a small “field of view” of a few inputs.
Shared Weights → Convolutions

Shared weights: each neuron uses the same weights...

Effect → the neuron is scanned over different fields of view → Convolution
Add more neurons which scans the field of view

Each neuron is a Filter being convolved with the input

Convolutional Layer with 4 filters production 4x4 output vector size
Convolution in 2D

Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

Source pixel

Convolution kernel (emboss)

New pixel value (destination pixel)

\[ (4 \times 0) \\
(0 \times 0) \\
(0 \times 0) \\
(0 \times 0) \\
(0 \times 1) \\
(0 \times 1) \\
(0 \times 0) \\
(0 \times 1) \\
+ (-4 \times 2) \\
-8 \]
What do filters do?

Through image =

<table>
<thead>
<tr>
<th>image</th>
<th>filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>0.5</td>
</tr>
<tr>
<td>0.0</td>
<td>-1.0</td>
</tr>
</tbody>
</table>

Through =

- 6.6
- 7.8
VGGNet (2014)

- Runner up, 2014 ILSVRC image recognition challenge
  - 140M parameters
  - 2-3 week training time on 4-GPU system
Representation Learning

Layer 1

Filter

Matching images

Representation Learning

Layer 3

L. Monier, G. Renard, [https://github.com/holbertonschool/deep-learning](https://github.com/holbertonschool/deep-learning)
Representation Learning

Layer 5

• Deep Convolutional Networks now have *super-human* performance in image recognition (ILSVRC Challenge)
Deep learning and High Energy Physics

• How can we make use of high-performance deep learning algorithms in HEP?

• Can deep learning find interesting and useful high-level representations of physics data?
  – Can they teach us something new?

• Think about our low-level data in new ways that are amenable to deep learning
  – Can we frame HEP questions as if they were image recognition tasks?
Neutrino Identification at NOvA

- Two 2D projections of the interactions
- Goal: discriminate between different neutrino interactions / backgrounds
Neutrino Identification at NOvA

- Treat 2D projections as images
  - Convolutional Neural network for imaging tasks

- Make use of GoogLeNet
  - Use first layers with useful representations for structures in NOvA detector (e.g. edges, ...)
  - Train with two image inputs, one for each view
Neutrino Identification at NOvA

• Convolution filters and outputs show interesting features about how the NN is providing discrimination

• Major gains over current algorithms in $\nu_e$-CC discrimination: $35\% \rightarrow 49\%$ signal efficiency for the same background rejection
Jets at the LHC
• Can we use in internal structure of a jet (i.e. the individual energy depositions) to classify different kinds of jets?

• Subfield of jet-substructure tries to answer this question using physics motivated features

• Can we learn the important information for discrimination directly from the data? And understand what we learned?
The Jet-Image

- Treat the detector as a camera: **The Jet-Image**
  - Calorimeter towers as pixels
  - Energy depositions as intensity

- Use all available information for jet classification

![Jet-Image Diagram]

**Diagram**: A scatter plot illustrating the distribution of energy depositions in calorimeter towers, with relative $\phi$ and relative $\eta$ axes.
Image pre-processing

**Pythia 8, W' → WZ, \( \sqrt{s} = 13 \text{ TeV} \)**

- **Pixelate** → **Translate** → **Rotate** → **Re-grid** → **Flip**

Use subjets to align images. Make use of symmetries: center, rotate, translate

- **Pythia 8, W' → WZ, \( \sqrt{s} = 13 \text{ TeV} \)**
  - 240 < \( p_T / \text{GeV} \) < 260 GeV, 65 < m_{mass}/GeV < 95
Discriminating Signal and Background

- In the past, explored linear classification techniques applied to Jet-Images
  - Similar / improved performance over physics-inspired variables
  - Image paradigm allows excellent insight into the “physics” governing discrimination through visualization

- Linear methods can be limited
  - All the physics inside of a jet is not linear

\[ \text{Discriminant } f(x) = w^*x \]
Vision, we study the potential of deep learning for interpreting data. Scientists use machine learning for rare-event detection, and hope to catch glimpses of new physics events.

The Large Hadron Collider (LHC) at CERN is the largest and most powerful particle accelerator in the world, collecting 3.2 million terabytes of proton-proton collision data every year.

Below, we see a snapshot of a 13 TeV proton-proton collision. We focus our attention on the Calorimeter, which we treat as a digital camera in cylindrical space. A channel detector captures snapshots of particle collisions occurring 40 million times per second. The ATLAS detector is one of the two general-purpose experiments at the LHC. The 100 million frames per second of data we take are used to define an image grid based on pseudorapidity ($\eta$) and azimuth ($\phi$) coordinates.

Hypothetical new physics events, done in Computer Vision, to account for non-discriminative difference in pixel intensities. We hypothesize these may have to do with difference-visualization techniques.

We transform each image in (left) enabling the connection between LHC physics event reconstruction and computer vision.

These images — called Jet Images — were first introduced by our group [JHEP 02 (2015) 118]. According to the grid arrangement, during a collision, energy from particles are deposited in pixels in (left), rotate around the jet-axis, and normalize each image, as is often done in Convolutional Neural Networks.

→

A true instance of Big Data. We show that modern Deep Convolutional Architectures can significantly enhance the discovery potential of the LHC. More importantly, the improved performance enhancing the discovery potential of the LHC.

S}
Performance with Deep Neural Networks

Pythia 8, $\sqrt{s} = 13$ TeV
$250 < p_T/\text{GeV} < 300 \text{ GeV}, 65 < \text{mass}/\text{GeV} < 95$

State of the art
In physics

Deep NN’s
Combining Deep NN’s with Substructure Variables

Pythia 8, $\sqrt{s} = 13$ TeV
$250 < p_T/\text{GeV} < 300 \text{ GeV}, 65 < \text{mass}/\text{GeV} < 95$

Large gains from combining mass with Deep NN

Deep NN’s not fully learning jet mass?

Combine Deep NN’s with jet mass
Combine Deep NN’s with substructure features
Looking “into” the network to better see what it is learning
Convolved representations

First layer 11x11 convolutional filters

Convolved jet image differences

\[ X_{\text{sig}} * w - X_{\text{bkg}} * w \]
Average Most Activating Jet Images
Pearson Correlation Coefficient of the pixels intensity with the network output: how discriminating information is contained within the network
Physics in deep representations

Additional radiation in QCD jets

Correlation of Deep Network output with pixel activations.

$\rho_{W}^{p} \in [250, 300]$ matched to QCD, $m_{W} \in [65, 95]$ GeV

Pearson Correlation Coefficient of the pixels intensity with the network output: how discriminating information is contained within the network
Learning something beyond mass and $\tau_{21}$ …

Spatial information indicative of radiation pattern for W and QCD:
New information learned by the network potentially related to colorflow
Where is DL in HEP going next?
Where is DL in HEP going next?

• Computer vision and imaging techniques may have broad applicability…
  – Calorimeter shower classification?
  – Energy calibration regression?
  – Pileup reduction?
  – Tracking?

• Sequential learning techniques (not discussed in this talk) may be useful in tasks with variable length data
  – Typical neural networks and BDT’s require a fixed input size
  – But not all discrimination tasks in HEP have a fixed size data representation, e.g. jets with variable numbers of constituents, variable number of jets in an events, …

• New network training paradigms may help fast simulations, or reduce systematic uncertainties…
New way to train networks… Potential for HEP?

- Train two networks “against” each other
  - One to generates an image
  - Second one to distinguish real / fake images
  - Potential applications for fast simulation?

- Domain adaptation: train with one dataset (MC) and apply on a slightly different one (data)
  - Minimize use of information not in both domains
  - Potential to reduce data/MC differences and systematic uncertainties during training?

Generative Adversarial Nets

\[
\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]
\]

https://arxiv.org/abs/1406.2661

Gradient Reversal Layers and Domain Adaptation

http://arxiv.org/abs/1409.7495
Conclusion

• Machine learning already used widely in HEP

• Deep learning is a new and powerful paradigm for machine learning in certain contexts

• Framing HEP data in the new ways can allow us to benefit from deep learning

• Already seen performance improvements and new insights when using deep learning in HEP

• Large potential for new image recognition and deep learning applications in HEP
Useful Python ML software

- Anaconda / Conda → easy to setup python ML / scientific computing environments
  - https://www.continuum.io/downloads
  - http://conda.pydata.org/docs/get-started.html

- Integrating ROOT / PyROOT into conda
  - https://nlesc.gitbooks.io/cern-root-conda-recipes/content/index.html
  - https://conda.anaconda.org/NLeSC

- Converting ROOT trees to python numpy arrays / panda dataframes
  - https://pypi.python.org/pypi/root_numpy/
  - https://github.com/ibab/root_pandas

- Scikit-learn → general ML library

- Deep learning frameworks / auto-differentiation packages
  - https://www.tensorflow.org/
  - http://deeplearning.net/software/theano/

- High level deep learning package build on top of Theano / Tensorflow
  - https://keras.io/
Decision Trees
Optimizing a Decision Tree

• Building an optimal decision tree is an NP-complete problem
  – Hard to find a global optimization for all splittings at the same time

• Greedy optimization → optimize one split at a time
  – Start with one leaf
  – Split leaf in two
  – Repeat as needed
Optimizing a Decision Tree

• When to split? Minimize impurity = $\Sigma_{\text{leaf}} \text{Impurity(leaf)} \times \text{size(leaf)}$
  
  – Typical leaf impurity functions:
  – Gini = $p \times (1-p)$
  – Entropy = $-p \times \log(p) - (1-p) \times \log(1-p)$
  
  • Where $p$ is the fraction of signal events in leaf, and size is the number of events falling into that leaf

  – Mean Square Error (regression): $(1/n_i) \Sigma_{i \in \text{leaf}} (y_i - m)^2$
  
  • Where $y_i$ is the true value, and $m$ is the average $y$ of events in the leaf

• When to stop splitting? Many criteria
  – Fixed tree depth
  – Information gain is not enough
  – Fix minimum samples needed in leaf
  – Fix minimum number of samples needed to split leaf
• Single decision trees can quickly overfit
• Especially when increasing the depth of the tree
Ensemble Methods

- Combine many decision trees, use the ensemble for prediction

- Averaging: \[ D(x) = \frac{1}{N_{\text{tree}}} \sum_{i=1}^{N_{\text{tree}}} d_i(x) \]

  - **Random Forest**, averaging combined with:
    - **Bagging**: Only use a subset of events for each tree training
    - **Feature subsets**: Only use a subset of features for each tree

- Boosting (weighted voting): \[ D(x) = \sum_{i=1}^{N_{\text{tree}}} \alpha_i d_i(x) \]

  - Weight computed such that events in current tree have higher weight misclassified in previous trees

  - Several boosting algorithms
    - AdaBoost
    - Gradient Boosting
    - XGBoost
Ensembles of Trees

- Ensembles of trees tend to work very well
  - Relatively simple
  - Relatively easy to train
  - Tend not to overfit (especially random forests)
  - Work with different feature types: continuous, categorical, etc.
CMS $h \rightarrow \gamma\gamma$ (8 TeV)
Non-Linear Activations

- The activation function in the NN must be a non-linear function
  - If all the activations were linear, the network would be linear:
    \[ f(X) = W_n(W_{n-1}(\ldots W_1X)) = UX, \quad \text{where} \quad U = \Pi_i W_i \]

- Linear functions can only correctly classify linearly separable data!

- For complex datasets, need nonlinearities to properly learn data structure
• Large NN’s difficult to train...trapping in local minimum?

  – Most local minima equivalent, and reasonable
  – Global minima may represent overtraining
  – Most bad (high error) critical points are saddle points (different than small NN’s)
Weight Initializations and Training Procedures

• Used to set weights to some small initial value
  – Creates an almost linear classifier

• Now initialize such that node outputs are normally distributed

• Pre-training with auto-encoder
  – Network reproduces the inputs
  – Hidden layer is a non-linear dimensionality reduction
  – Learn important features of the input
  – Not as common anymore, except in certain circumstances…

• Adversarial training, invented 2014
  – Will potential HEP applications later
MaxOut

\[
\text{Max}\{z_1, z_2, z_3, z_4\}
\]

Hidden layer
Different Colors represent different weights \( W^*x \)

\begin{align*}
\text{output} & \\
\text{input} & \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6
\end{align*}
ReLU Networks

- Sparse propagation of activations and gradients in a network of rectifier units. The input selects a subset of active neurons and computation is linear in this subset.

- Model is “linear-by-parts”, and can thus be seen as an exponential number of linear models that share parameters

- Non-linearity in model comes from path selection

http://www.jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf
Convolutions in 2D

- Scan the filters over the 2D image, producing the convolved images
Max Pooling

- Down-sample the input by taking MAX or average over a region of inputs
  - Keep only the most useful information
Daya Bay Neutrino Experiment

• Aim to reconstruct inverse $\beta$-decay interactions from scintillation light recorded in 8x24 PMT’s

• Study discrimination power using CNN’s
  – Supervised learning $\rightarrow$ observed excellent performance (97% accuracy)
  – Unsupervised learning: ML learns itself what is interesting!
Jet tagging using jet substructure

- **Typical approach:**
  Use physics inspired variables to provide signal / background discrimination

- **Typical physics inspired variables exploit differences in:**
  - **Jet mass**
  - **N-prong structure:**
    - 1-prong (QCD)
    - 2-prong (W,Z,H)
    - 3-prong (top)
  - **Radiation pattern:**
    - Soft gluon emission
    - Color flow
Jet tagging using jet substructure

- **Typical approach:** Use physics inspired variables to provide signal / background discrimination

- **Typical physics inspired variables exploit differences in:**
  - **Jet mass**
  - **N-prong structure:**
    - 1-prong (QCD)
    - 2-prong (W,Z,H)
    - 3-prong (top)
  - **Radiation pattern:**
    - Soft gluon emission
    - Color flow

\[
\tau_N = \frac{1}{d_0} \sum p_{T,k} \min\{\Delta R_{k,\text{axis}-1}, \ldots, \Delta R_{k,\text{axis}-n}\}
\]
Pre-processing steps may not be Lorentz Invariant

- Translations in $\eta$ are Lorentz boosts along $z$-axis
  - Do not preserve the pixel energies
  - Use $p_T$ rather than $E$ as pixel intensity
- Jet mass is not invariant under Image normalization

\[
m^2_J = \sum_{i<j} E_i E_j (1 - \cos(\theta_{ij}))
\]
Restricted phase space

2-prong

W jets

QCD jets

τ_{21}

1-prong

Pythia 8, W\rightarrow WZ, \tilde{G} = 13 \, \text{TeV}

240 < p_T < 300 \, \text{GeV}, 0.19 < \eta < 0.21

Pythia 8, QCD dijets, \tilde{G} = 13 \, \text{TeV}

240 < p_T < 300 \, \text{GeV}, 0.19 < \eta < 0.21

Pythia 8, QCD dijets, \tilde{G} = 13 \, \text{TeV}

240 < p_T < 300 \, \text{GeV}, 0.19 < \eta < 0.21

2-prong

τ_{21}

1-prong

79 < m < 81 \, \text{GeV}

0.19 < \tau_{21} < 0.21

Restrict the phase space in very small mass and τ_{21} bins:
Improvement in discrimination from new, unique, information learned by the network
Spatial information indicative of radiation pattern for W and QCD: where in the image the network is looking for discriminating features