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
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 needed number and best location of dataset replicas

- Much of this work has been done using Boosted Decision Trees
  - Well suited when using heavily engineered high level features

---

JHEP 01 (2016) 064
JINST 10 P08010 2015
arXiv:1512.05955
Generated decay mode

---
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?

\[ y = \sigma(Uz + c) \]

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

\[ \sigma(x) = \text{sigmoid function} \]

is the \textit{Activation Function}
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 NNs have many hidden layers
  - Factorize the learning of structure in the data across many layers

- Difficult to train, only recently possible with large datasets, fast computing (GPU) and new training procedures / network structures

http://www.asimovinstitute.org/neural-network-zoo/
Machine Learning and Jet Physics
• 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?
Jet Images and Computer Vision

- Recast our conception of jets: **Jet-Images**
  - Treat energy depositions like pixels in an image

- Use modern computer vision approach to classification: deep convolutional networks
  - Scan “filters” over the 2D image, producing the convolved images
  - Filters perform local pattern matching

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

- Input image
- Convolved image

Convolutions

- Stride = 1
- \( D = 4 \)
- \( L \times W = 5 \)
- Zero padding

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.

(4 x 0)
(0 x 0)
(0 x 0)
(0 x 0)
(0 x 1)
(0 x 1)
(0 x 0)
(0 x 1)
+ (4 x 0)
-6

Shared weights!!!
Jet-Images Performance

State of the art single feature in physics

Correlation between input pixels and NN output

Correlation of Deep Network output with pixel activations.

Pythia 8, $\sqrt{s} = 13$ TeV

$250 < p_T$/GeV $< 300$ GeV, $65 <$ mass/GeV $< 95$

Deep Neural Networks

1/(Background Efficiency)

Signal Efficiency
• Can also generate Jet-Images using Generative Adversarial Networks!

• Similar setup allows fast simulation of 3D particle energy depositions in calorimeter
Beyond Images: Deep Learning on Jet Constituents

- Feed constituents (not images) directly into deep fully connected network
  - Feed only top 120 $p_T$ constituents

- Use recursive tree structure (unique to each jet) to process one jet constituent at a time
  - Tree structure can match that of a jet algorithm!

-- Diagram of tree structure

--- Graph showing DNN performance for top quark jets vs. QCD

- Boosted top quark jets vs. QCD
  - arXiv:1704.02124

--- Graph showing DNN performance for W jets vs. QCD

- Boosted W jets vs. QCD
  - arXiv:1702.00748
• Identify jets containing $b$-hadrons by finding displaced vertices or large impact parameter tracks
Impact Parameter Tagging

- Impact parameter based tagging difficult due to high dimensional space of all tracks in a jet
- Typically make assumption that properties of tracks are uncorrelated between different tracks
  - But correlations do exist!
Sequence learning and Recurrent NN

- Instead of considering the tracks as individual objects, treat them as sequence
  - Make use of sequence classification techniques in ML!
  - Naturally unordered sequence, but can impose a physics-inspired ordering:
    Order here by *largest impact parameter significance*

- Neural network approach to analyzing sequences is Recurrent Neural Networks
  - Used in sentence classification, Natural Language Processing, time-series analysis, etc.
RNN b-tagging

- RNN captures correlations not seen by IP3D
  - even with only impact parameter significance and track category inputs
Dealing with Systematic Uncertainties
Dealing with Systematic Uncertainties

- Systematic uncertainties encapsulate our incomplete knowledge of physical processes and detectors
  - Systematic uncertainty encoded as nuisance parameters, $Z$

- Can we teach a classifier to be robust to these kinds of uncertainties?

**Boosted W vs. QCD with Jet Images**

- Background rejection
  - Pile-up $\langle \mu \rangle = 50$
  - DNN(image)
  - BDT(expert)
  - $D^{\theta=2}$ + mass
  - $D^{\theta=1}$ + mass
  - Jet mass

- Deep Neural Network Performance

- Jet Mass and N-subjettiness Ratio $\tau_2$

arXiv:1603.09349  
arXiv:1609.00607
Dealing with Systematic Uncertainties

- Systematic uncertainties encapsulate our incomplete knowledge of physical processes and detectors
  - Systematic uncertainty encoded as nuisance parameters, $Z$

- Can we teach a classifier to be robust to these kinds of uncertainties?

\[
\hat{\theta}_f, \hat{\theta}_r = \arg \min_{\theta_f} \max_{\theta_r} E(\theta_f, \theta_r).
\]
\[
E_\lambda(\theta_f, \theta_r) = \mathcal{L}_f(\theta_f) - \lambda \mathcal{L}_r(\theta_f, \theta_r),
\]
Learning to Pivot

• Tune the classification vs robustness in training to maximize significance, even beyond standard approaches

• Example:
  – W-tagging vs QCD
  – Physics inspired variables as inputs
  – Systematic: noise from additional “pileup” interactions in collision

  – Count events passing minimum network output threshold → compute significance including uncertainty (AMS)

Optimal tradeoff of performance vs. robustness

Non-Adversarial training

arXiv:1611.01046
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
  – …

• Sequence learning techniques may have broad applicability in tasks with variable length data
  – Tracking, jets with variable numbers of constituents, variable number of jets in an event, …

• New network training paradigms may help fast simulations, or reduce systematic uncertainties…
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, sequence learning, and deep learning applications in HEP
Convolutions in 2D

- State of the art in computer vision
- Scan the filters over the 2D image, producing the convolved images
- Filters perform local pattern matching
Neural Network Architectures

- Structure of the networks, and the node connectivity can be adapted for problem at hand

- **Convolutions**: shared weights of neurons, but each neuron only takes subset of inputs

[Bishop]

http://www.asimovinstitute.org/neural-network-zoo/
GANs for Simulation

\[ e^+ \quad \gamma \quad \pi^+ \]

<table>
<thead>
<tr>
<th>Generation Method</th>
<th>Hardware</th>
<th>Batch Size</th>
<th>milliseconds/shower</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEANT4</td>
<td>CPU</td>
<td>N/A</td>
<td>1772</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>1</td>
<td>13.1</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>10</td>
<td>5.11</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>128</td>
<td>2.19</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>1024</td>
<td>2.03</td>
</tr>
<tr>
<td>CALOGAN</td>
<td>CPU</td>
<td>1</td>
<td>14.5</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>4</td>
<td>3.68</td>
</tr>
<tr>
<td></td>
<td>GPU</td>
<td>128</td>
<td>0.021</td>
</tr>
<tr>
<td></td>
<td>GPU</td>
<td>512</td>
<td>0.014</td>
</tr>
<tr>
<td></td>
<td>GPU</td>
<td>1024</td>
<td>0.012</td>
</tr>
</tbody>
</table>

arXiv:1705.02355
RNN b-tagging

**ATLAS Simulation Preliminary**

$t_s=13$ TeV, $t\bar{t}$

$p_T>20$ GeV, $|\eta|<2.5$

- $1/\epsilon_l$, light-jet rejection
- $1/\epsilon_c$, c-jet rejection

Differing lines represent different algorithms:

- Red: MV2c10
- Dotted Red: RNNIP
- Blue: IP3D
- Dotted Blue: SV1
RNN b-tagging

**ATLAS Simulation Preliminary**

- **$\sqrt{s}=13$ TeV, $t\bar{t}$**
- $p_T>20$ GeV, $|\eta|<2.5$

**Correlation, $p(D_{\text{RNN}, S_{20}})$**

- b-jets
- c-jets
- light-jets

**Correlation, $p(D_{\text{RNN}, \Delta R})$**

- b-jets
- c-jets
- light-jets

**Correlation, $p(D_{\text{RNN}, S_{30}})$**

- b-jets
- c-jets
- light-jets

**Correlation, $p(D_{\text{RNN}, S_{40}})$**

- b-jets
- c-jets
- light-jets
Deep Neural Network Based b-tagging

- Exploring several deep NN architectures to combine engineered features with per track level information

### DeepCSV
- Input (displ. sort.): up to 6x, up to 1x
- Features: charged part. 8, sec. vert. 8, global, 12
- Dense: 5x100
- Output: b, bb, c, l

### DeepFlavour
- Input (displ. sort.): up to 25x, up to 25x, up to 4x
- Features: charged part. 16, neutral part. 8, sec. vert. 17, global, 6
- 1x1 conv.: 64/32/32/8, 32/16/4, 64/32/32/8
- Dense: 350, 6x100
- Output: b, bb, c, l

---

**CMS Simulation**
- Preliminary
- $s=13$ TeV, Phase 1
- tf events
- AK4jets ($p_T > 30$ GeV)
- Curves: DeepCSV, noConv, DeepFlavour, udsg, c
- DP-2017-013
Adversarial Networks

• Adversarial training: a mini-max game
  – Train one neural network ($f$) to perform the classification task
  – Train a second network ($r$) to predict the nuisance parameter $Z$ from $f$

• The loss encodes the performance of both classifiers, but is penalized when $r$ does well

\[
\hat{\theta}_f, \hat{\theta}_r = \arg \min_{\theta_f} \max_{\theta_r} E(\theta_f, \theta_r).
\]

\[
E_\lambda(\theta_f, \theta_r) = \mathcal{L}_f(\theta_f) - \lambda \mathcal{L}_r(\theta_f, \theta_r),
\]

arXiv:1611.01046
Learning to Pivot: Toy Example

- 2D example
  \[ x \sim \mathcal{N}\left((0, 0), \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}\right) \] when \( Y = 0, \)
  \[ x \sim \mathcal{N}\left((1, 1 + Z), \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) \] when \( Y = 1. \)

- Without adversary (top) large variations in network output with nuisance parameter

- With adversary (bottom) performance is independent!
Decorrelating Variables

- Can use same adversarial setup and procedure to decorrelate a classifier from a chosen variable (rather than nuisance parameter)

arXiv:1703.03507
Weakly Supervised Training on Data

- If we can train directly on data, many analysis uncertainties can be avoided.
- If we have multiple samples with known class proportions, we can train on proportions instead of labels.
  - When the proportions are non-unity, it is possible to modify the loss and learn.