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 at the Large Hadron Collider (LHC) at CERN.

Below, we see a snapshot of a 13 TeV proton-proton collision. The ATLAS detector is one of the two general-purpose experiments at the LHC. The 100 million LHC events in new ways.

In our experiments, we build discriminants on top of Jet Images to distinguish between a true instance of Big-Bang physics and background noise. We perform this task in Computer Vision, to account for non-discriminative differences in pixel intensities.

To visualize what the network learns, we transform each image in a batch into its difference with the average image of that class. A true instance of Big-Bang physics should differ significantly from the average. A true instance of background noise should not.

Below, we have the learned convolutional filters (left) and the difference in between the average image and a typical event. The network learns a two-prong event, which is consistent with the two-photon production of a Higgs boson (W→WW). The network learns beyond theory-driven variables — enhancing the discovery potential of the LHC. More importantly, the improved performance is observable on new physics processes compared to state-of-the-art methods based on physics features, indicating that the deep network learns beyond theory-driven variables — enhancing the discovery potential of the LHC. More importantly, the improved performance is observable on new physics processes compared to state-of-the-art methods based on physics features.
What this is not
What this is not

...the Higgs boson?
What this is not

A replacement for a great online tutorial or a UC course

(STAT 24400-24500)
CMSC 25025/STAT 37601
CMSC 25400/STAT 27725
TTIC 31020 (see Toyota Institute)
TTIC 31230/CMSC 35300
STAT 24610
...

What is Machine Learning?
What is Machine Learning?

Answer: just about everything we do!

...algorithms for identifying and analyzing structure in data
What can we use machine learning for?

**Supervised learning**
- Classification
- Regression
- Generation

**Unsupervised learning**
- Clustering
- Anomaly detection

the machine is presented examples of multiple classes and learns to differentiate

the machine is presented data and asked to give you multiple classes
What can we use machine learning for?

Supervised learning
- Classification
  - Regression
  - Generation
Unsupervised learning
- Clustering
- Anomaly detection

Higgs boson or gluon? multiple classes
What can we use machine learning for?

**Supervised learning**
- Classification
- Regression
- Generation

**Unsupervised learning**
- Clustering
- Anomaly detection

---

What is the energy of this spray of particles (jet)?

What are the momenta of these charged particles?
What can we use machine learning for?

**Supervised learning**
- Classification
- Regression
- **Generation**

**Unsupervised learning**
- Clustering
- Anomaly detection

What would Higgs boson events look like with a different mass?
What can we use machine learning for?

**Supervised learning**
- Classification
- Regression
- Generation

**Unsupervised learning**
- Clustering
- Anomaly detection
What can we use machine learning for?

Supervised learning
- Classification
- Regression
- Generation

Unsupervised learning
- Clustering
- Anomaly detection
Classification

Goal: Given a *feature vector*, return an integer indexed by the set of possible *classes*.

In most cases, we care about *binary* classification in which there are only two classes (signal versus background).

There are some cases where we care about *multi-class classification*.

Feature vector can be many-dimensional

Harder = more overlap between for S and B
Classification

Goal: Given a feature vector, return an integer indexed by the set of possible classes.

In practice, we don’t just want one classifier, but an entire set of classifiers indexed by:

**True Positive Rate** = signal efficiency = 
Pr(label signal | signal) = sensitivity

**True Negative Rate** = 1 - background efficiency = rejection = Pr(label background | background) = specificity

*For a given TPR, we want the lowest possible TNR!*
Let’s consider an important special case: binary classification in 1D.
Let's consider an important special case: binary classification in 1D.

**Probability Distribution Function**

**Reconstructed Jet Mass [GeV]**

60  80  100  120  140

**Fraction / 2 GeV**

0  0.05  0.1  0.15  0.2  0.25  0.3

**ATLAS Simulation Preliminary**

\( \sqrt{s} = 8 \text{ TeV} \)  PYTHIA \( W' \rightarrow WZ \)

200 GeV < \( p_T^Z \) < 400 GeV

**could be e.g. the jet mass**
Let’s consider an important special case: binary classification in 1D

What is the optimal classifier?
Let’s consider an important special case: binary classification in 1D

You may be tempted to place a threshold on \( x \)
Let's consider an important special case: binary classification in 1D

You may be tempted to place a threshold on $x$

Threshold depends on natural relative abundance
In this simple case, the log LL is proportional to $x$: **no need for non-linearities!**

**Threshold cut is optimal**
The optimal procedure is a threshold on the LL.

"Receiver Operating Characteristic" (ROC) Curve

Pr(label signal | signal)
What if the distribution of $x$ is complicated?

Real life is complicated!

Now what is the optimal classifier?
In this case, LL is highly non-linear (non-monotonic) function of x.

A threshold on x would be sub-optimal.
ROC is worse than the Gaussians, but that is expected since the overlap in their PDFs is higher.
Why don’t we always just compute the optimal classifier?

In the last slides, we had to estimate the likelihood ratio - this required binning the PDF. Binning works very well in 1D, but becomes quickly intractable as the feature vector dimension $\gg 1$ ("curse of dimensionality")

Machine learning for classification is simply the art of estimating the likelihood ratio with limited training examples.
Tools for Classification

= tools for likelihood ratio estimation

- “Histograming”
- Nearest Neighbors
- Support Vector Machines (SVM)
- (Boosted) Decision Trees
- (Deep) Neural Networks
- ...

Software: TMVA, scikit-learn, keras, ...

Data formats: .root, .npy, .hdf5

Not widely used; only useful if decision boundary is ‘simple’

Has most things and ROOT-compatible but the community base is much smaller than the other ones

does “everything” exempt DNNs

Python interface to DNN tools TensorFlow, Theano, CNTK
Histogramming

If you have a 1D problem, look no further!

If your problem can be decomposed into a product/sum of 1D problems…look no further!

If these do not apply… look elsewhere.

\[ p(M, Q, B|V) = \sum_{\mathcal{F}} \Pr(\mathcal{F}|V)p(M|\mathcal{F}, V)p(Q|\mathcal{F}, V)\Pr(B|\mathcal{F}, V), \]

**ATLAS** Simulation Preliminary

ATL, Simulation Preliminary

\( \sqrt{s} = 8 \, \text{TeV}, \) **PYTHIA** \( W' \rightarrow WZ \)

\( \epsilon_Z = 90\% \quad \epsilon_Z = 50\% \quad \epsilon_Z = 10\% \)
Nearest Neighbors

In 2D, a nice extension of histogramming is to estimate the likelihood ratio based on the number of S and B points nearby.
Boosted Decision Trees (BDTs)

We love BDTs.

If $3 < \dim(\text{feature vector}) < O(100)$

this is probably right for you!
We love BDTs because they are fast to train and do not have very many parameters. They are also rather robust to overtraining.
We love BDTs because they are fast to train and do not have very many parameters. They are also rather robust to overtraining.

Unless you have a lot of training data, it is better to use cross-validation instead of a single hold-out for evaluating out-of-sample performance.
Boosted Decision Trees (BDTs)

There is really not a good reason to use a DNN with << O(100) dimensions.

However, they are becoming increasingly easy to train …
Modern Deep NN’s for Classification

**Neural Network:** composition of functions $f(Ax+b)$ for inputs $x$ (features) matrix $A$ (weights), bias $b$, non-linearity $f$.

N.B. I’m not mentioning biology - there may be a vague resemblance to parts of the brain, but that is not what modern NN’s are about.

**Fact:** NN’s can approximate “any” function.
Choosing the non-linearity (activation function) $f$

**Logistic (aka Sigmoid):** one of the most widely-used functions in the past, no basically only used for the last layer.

Generalization to multi-dimensional input: softmax

$$f(\vec{x}) = \frac{e^{x_i}}{\sum_i e^{x_i}}$$

**tanh:** similar story to sigmoid.
Choosing the non-linearity (activation function) $f$

Rectified Linear Unit **ReLU**: one of the most widely-used functions now. 

do not suffer from the vanishing gradient problem

Leaky ReLU / Exponential LU (ELU): variations on the ReLU that are popular.
Functions that act on multiple nodes in one layer

**MaxOut:** Take the maximum of multiple inputs

*reduces the dimensionality of a hidden layer*

**DropOut:** Randomly remove (for one forward/backward pass) nodes from a layer.

*helps with over-training*
(D)NN Training

Training proceeds by minimizing a loss function.

Typical loss functions

Squared error: \( (y_i - \hat{y}_i)^2 \)

Cross-entropy: \[-y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i)\]
Objective function is minimized using stochastic gradient decent (almost exclusively with the Adam algorithm)

Stochastic gradient decent: Using single (or multiple “mini-batches”) examples, weights are updated:

$$A_{ij} \rightarrow A_{ij} - \eta \nabla_{ij} \mathcal{L}$$

learning rate back-propagation: weights updated backwards and gradients are recycled.

N.B. a NN can do better than random **before any training**!
For instance, if you initialize all the weights to 1 and the signal has generally higher values then the NN will beat random.
(D)NN Training

Training proceeds multiple times (epochs), reshuffling the data.

Early stopping: stop at the epoch where the validation error starts to increase.
In the tutorials today, you will get a chance to apply these concepts in practice.

Before closing, I’ll leave you with one last concept: semi-supervised learning.

unsupervised          semi-supervised          supervised

nothing               e.g. class proportions   everything

how much you know about per-example labels
For supervised learning, we depend on labels labels usually come from simulation.

For supervised learning, we depend on labels labels usually come from simulation.

For supervised learning, we depend on labels labels usually come from simulation.

What if data and simulation are very different?...your classifier will be sub-optimal.
Boosted $W$ boson jets

DNN classifiers can **exploit** subtle features

subtle features are **hard to model**

we need to be careful about which models we use - **only data is correct**

N.B. not all of these have been tuned to the same data

We will take about image feature vectors later today
What if data and simulation are very different?

...your classifier will be sub-optimal.

For supervised learning, we depend on labels

jets in quark vs gluon

been recent advances [equivalent to the problem of learning with asymmetric random label noise, where there have...full-supervised learning. We explore the practical performance of CWoLa in Secs. guarantees that the optimal classifier from CWoLa is the same as the optimal classifier from classifier. Therefore,

which is a monotonically increasing rescaling of the likelihood

we can relate these two likelihood ratios algebraically:

Proof.

Discriminant for MC-Based Tagger

The problem of learning from unknown mixed samples can be shown to be mathematically

An important feature of CWoLa is that, unlike the LLP-style weak supervision in Sec.

about the signal/background labels or class proportions in the mixed samples is used during training.

coming either from the first or second mixed sample, labeled as 0 and 1 respectively. No information

signal (S/B) and (M/M)

Figure 1

S/B

The optimal classifier to distinguish examples drawn from

M

| < 0.8

L

| < 0.8

L = q/(q+g)

Pythia MC11,

160 GeV

1708.02949

related ideas: L. Dery, BPN, F. Rubbo, A. Schwartzman, JHEP 05 (2017) 145
Training on data:

learning when you know (basically) nothing

For supervised learning, we depend on labels

labels usually come from simulation.

Where

...your classifier will be sub-optimal
Training on data: learning when you know (basically) nothing

For supervised learning, we depend on labels labeled as either 0 or 1. If training on data:

$$L_{S/B} = \frac{L}{1 + L}$$

$$L_p = 1 - \frac{1}{m_H}$$

$$L_{S/B} \approx \frac{S}{B}$$

$$\frac{S}{B} \approx \frac{L}{1 + L}$$

If

$$\int f \, d\beta = 0$$

then one obtains the reversed

$$f, \text{optimal classifier trained to}$$

$$f_1, f_2 = 0.8, 0.2$$

$$pp \rightarrow H \rightarrow q\bar{q}/gg$$

$$p_T^D$$

$$\sqrt{s} = 13 \text{ TeV}$$

$$m_H = 500 \text{ GeV}$$

What if data and simulation are very different?

...your classifier will be sub-optimal

E. Metodiev, BPN, J. Thaler, 1708.02949

related ideas: L. Dery, BPN, F. Rubbo, A. Schwartzman, JHEP 05 (2017) 145
The future

(D)NN’s are powerful tools that will help us fully exploit the physics potential of our experiments.

We must be cautious to apply the right tool for the right job. The more you know, the less black the boxes will be...