Deep Learning and Computer Vision in High Energy Physics

Michael Kagan

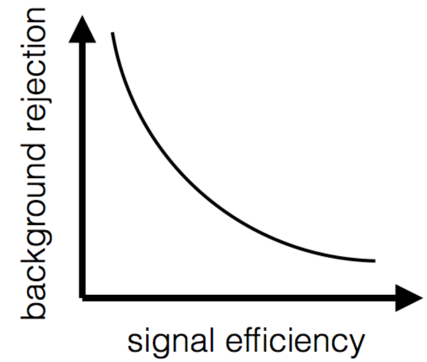
SLAC

LAL,

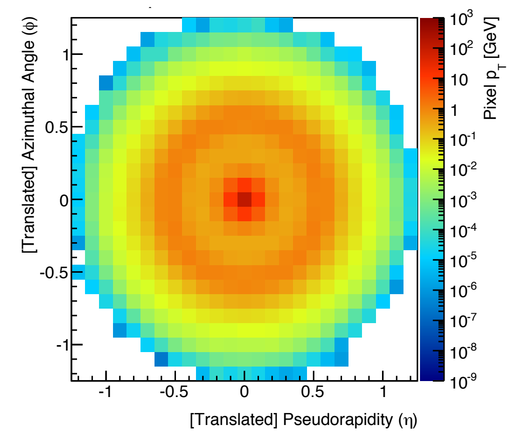
January 10, 2017

Aspects of Machine Learning (ML) in HEP

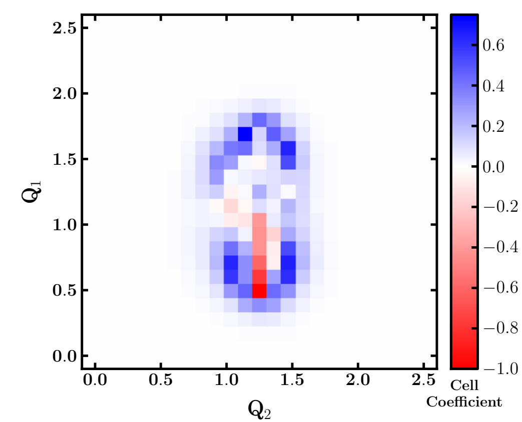
- **Optimization**
  - Bottom line is performance
  - But can we build new better (simple?) features?



- **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

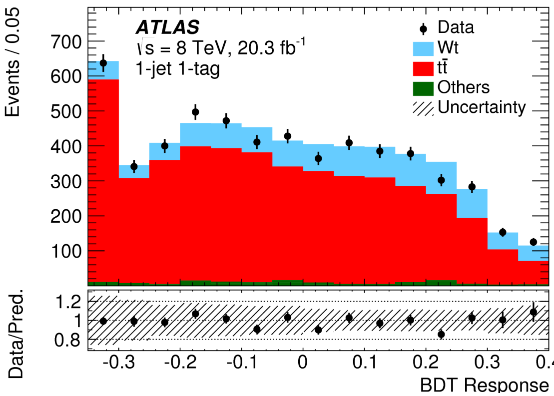


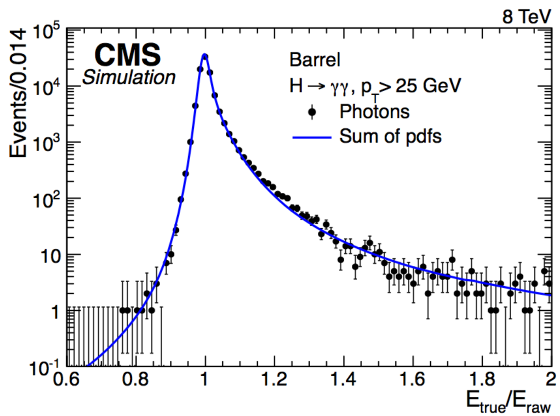
- **Learning from Learning** …(if we can)
  - Can we extract information about what the ML is learning?
  - Can we use this information to design new variables?
  - Often visualization is a key component

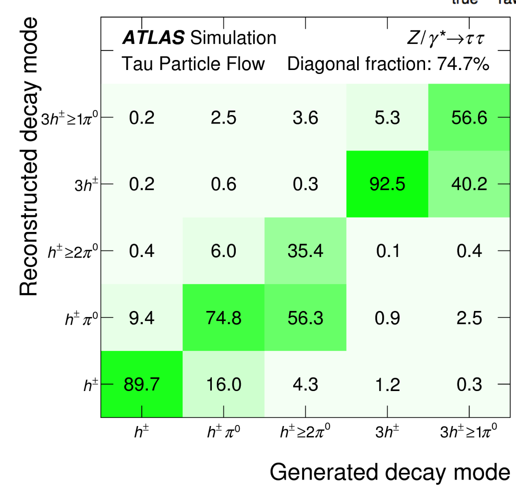


Machine Learning Applied Widely in HEP

- **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 location of dataset replicas

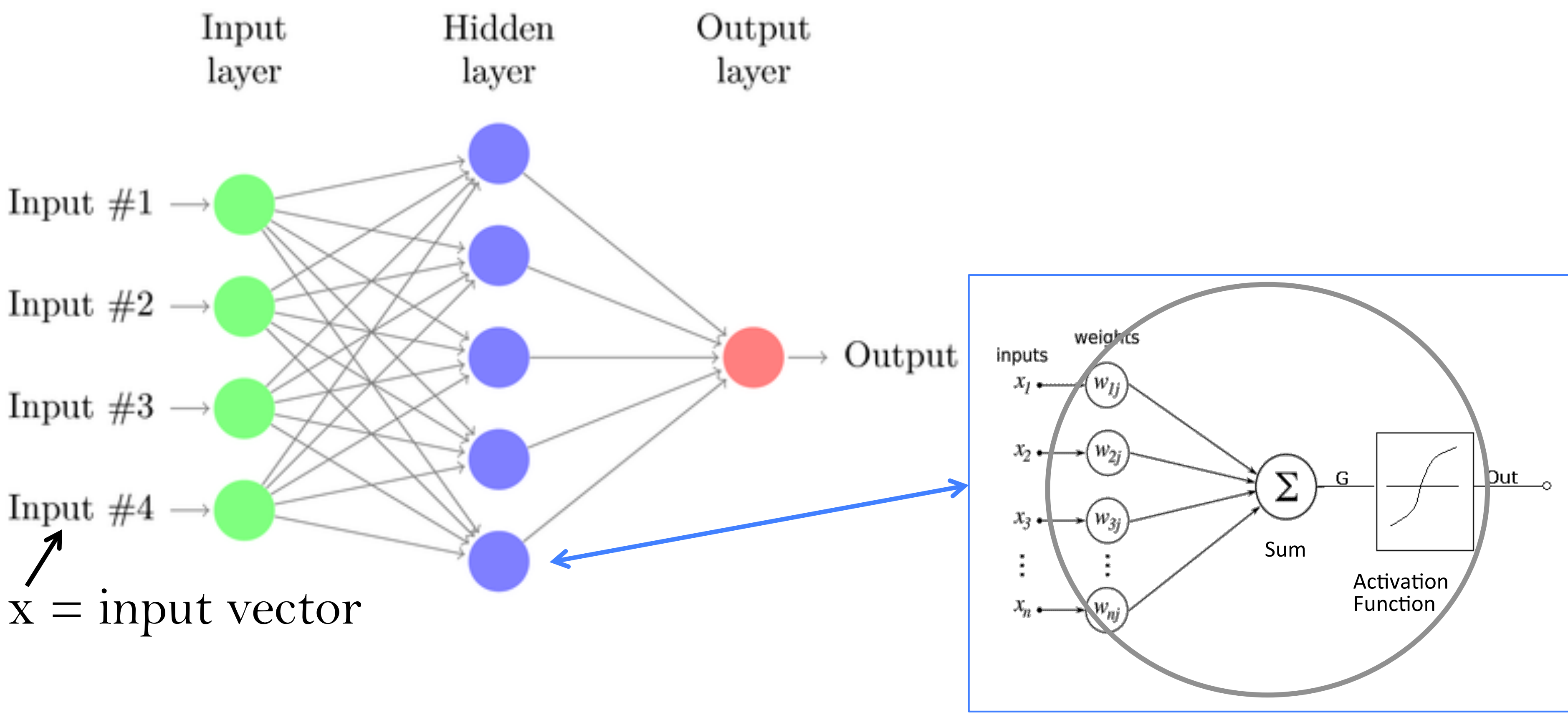






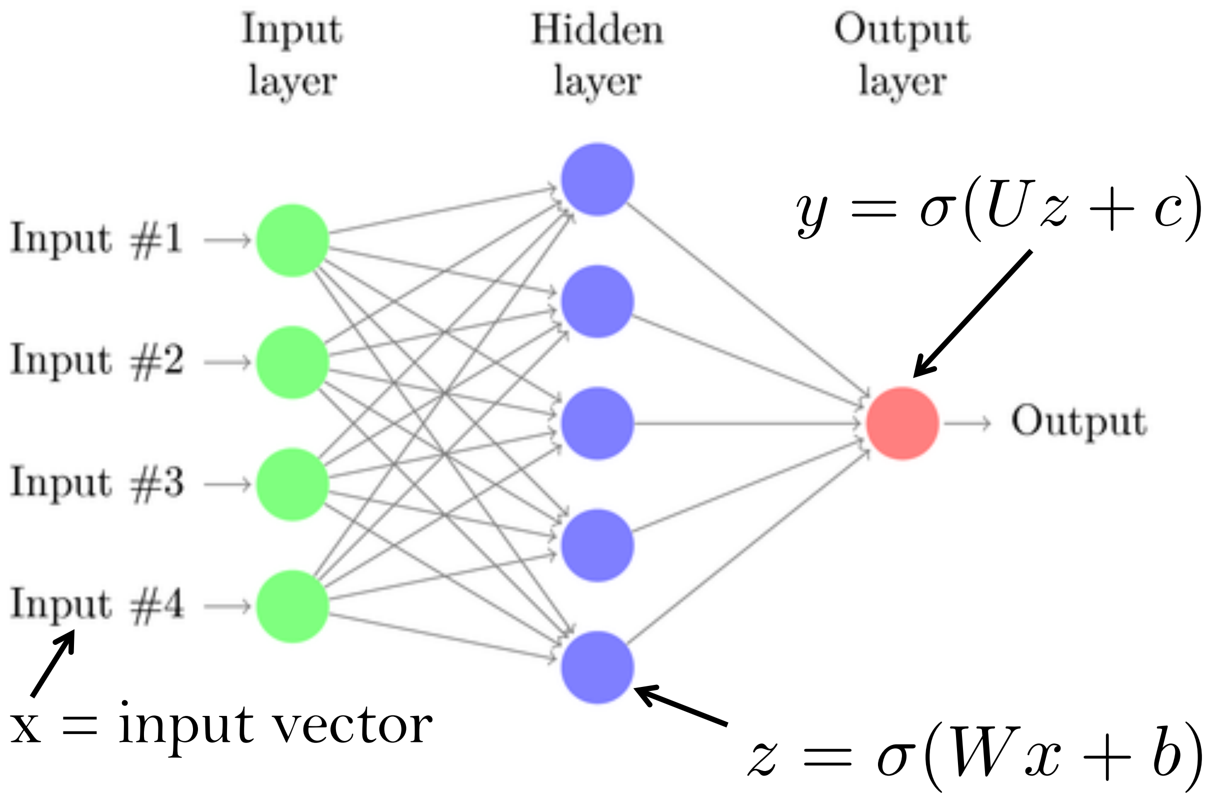
	$h^\pm$	$h^\pm\pi^0$	$h^\pm\geq2\pi^0$	$3h^\pm$	$3h^\pm\geq1\pi^0$
$3h^\pm\geq1\pi^0$	0.2	2.5	3.6	5.3	56.6
$3h^\pm$	0.2	0.6	0.3	92.5	40.2
$h^\pm\geq2\pi^0$	0.4	6.0	35.4	0.1	0.4
$h^\pm\pi^0$	9.4	74.8	56.3	0.9	2.5
$h^\pm$	89.7	16.0	4.3	1.2	0.3

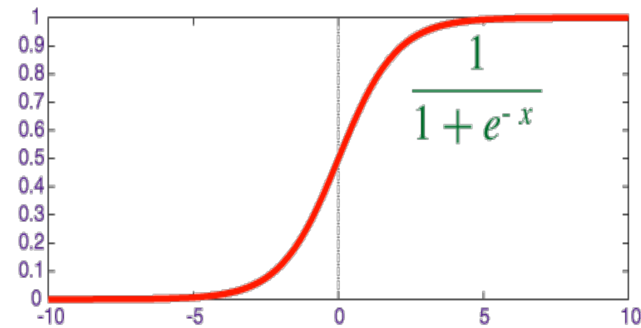
Neural Networks



- "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?

Neural Networks





σ(x) = sigmoid function
is the *Activation Function*

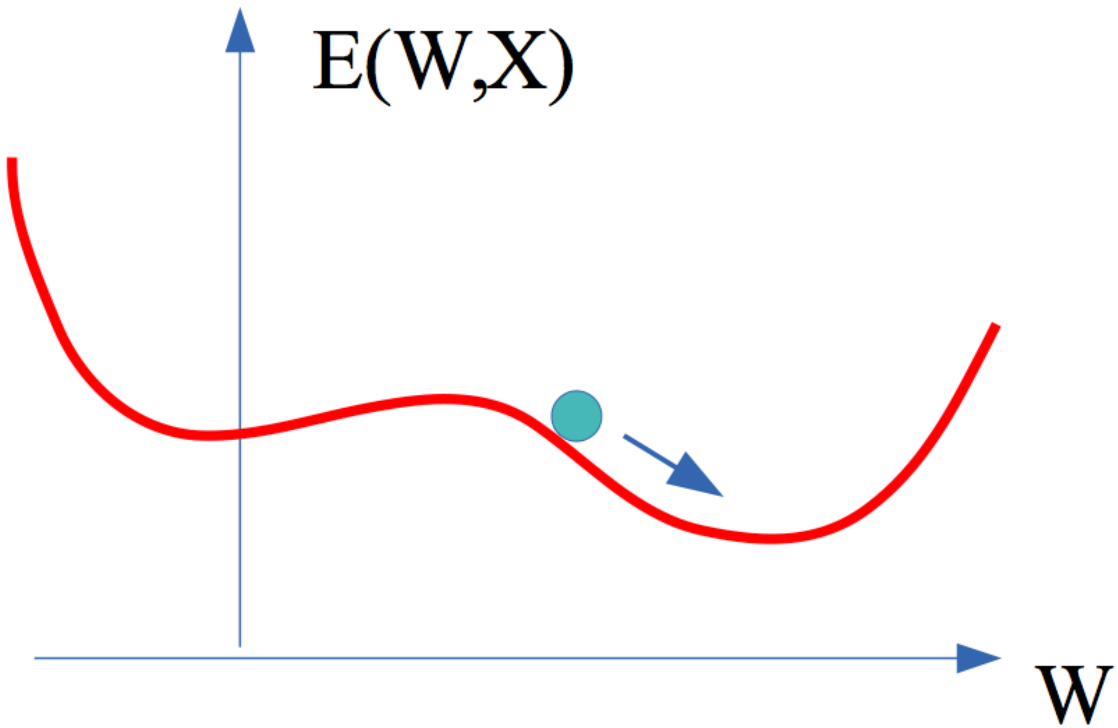
- "Typical" neural network circa 2005
- Typical questions of optimization
  - Which variables to choose as inputs? How correlated are they?
  - How many nodes in the hidden layer?

Training a Neural Network

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

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

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



Training a Neural Network

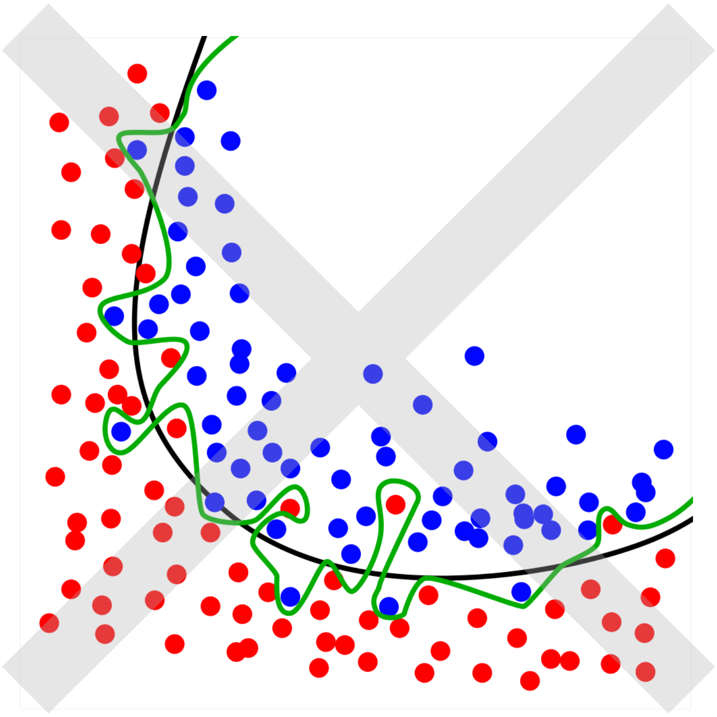
- Define a **loss function** that depends on predictions f(x;w) and targets y

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

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

- 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

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

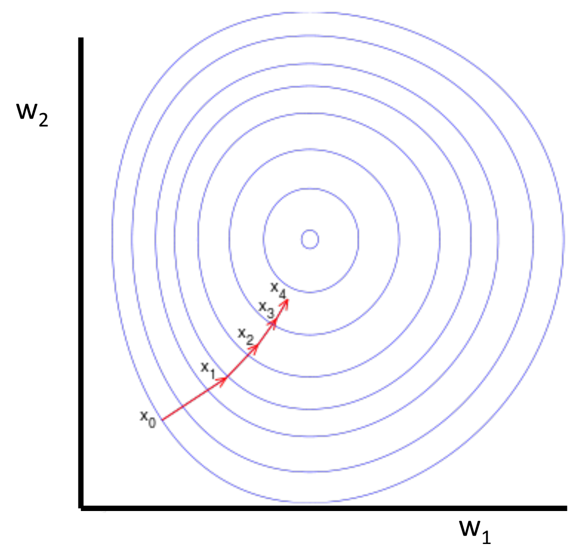
$$L_{BCE} = \frac{1}{N}\sum_{i=1}^{N_{examples}} -y_i \log f(x_i) - (1-y_i)\log(1-f(x_i))$$

- 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

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

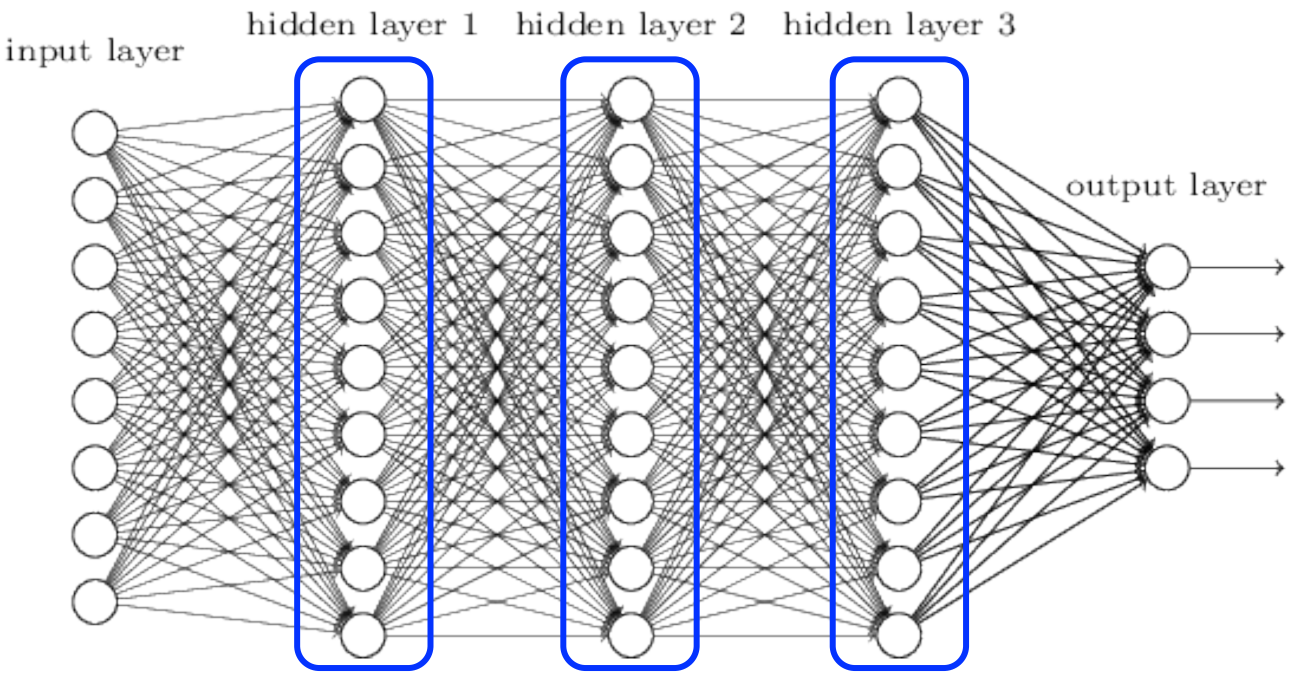


- Update weights with **gradient descent**

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

  - α is called the learning rate

Deep Neural Networks



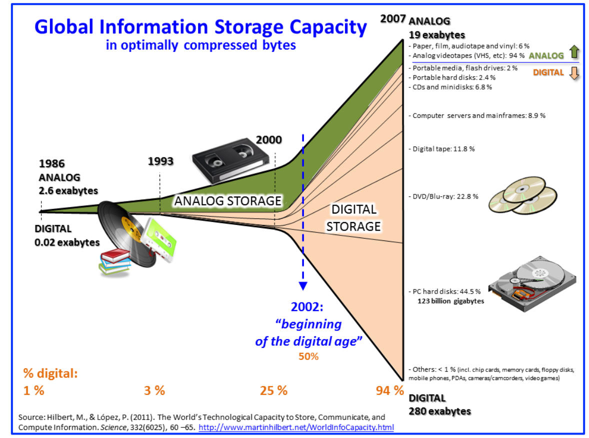
- As data complexity grows, need exponentially large number of neurons in a single-hidden-layer network to capture all the structure in the data

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

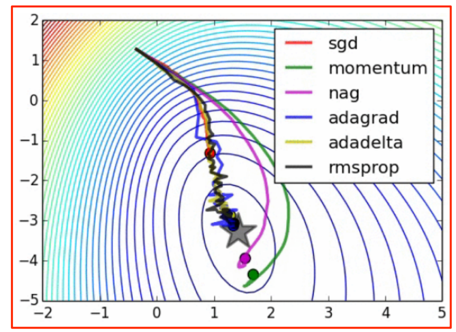
- Difficult to train, only recently has this become possible…

Why did it take so long to train DNN's?

- Big Data: Large datasets vital for training (hundreds of) millions of parameters

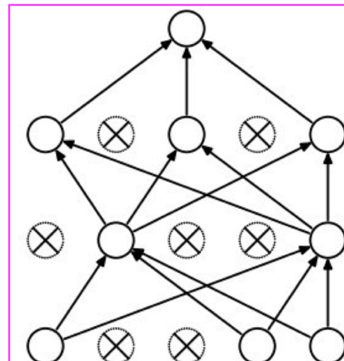


- GPU's: Dramatically increased speed of training



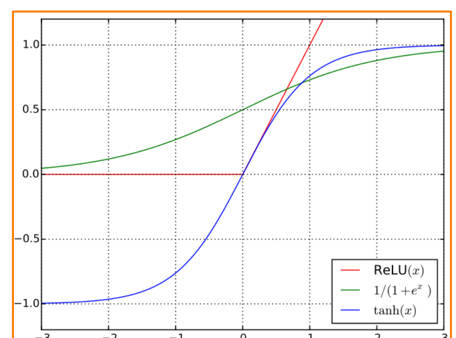
- Improved optimization algorithms

- New regularization techniques: dropout, batch normalization, etc.



(b) After applying dropout.

- New activation functions, like Rectified Linear Units



- …

Deep NNs in HEP analysis

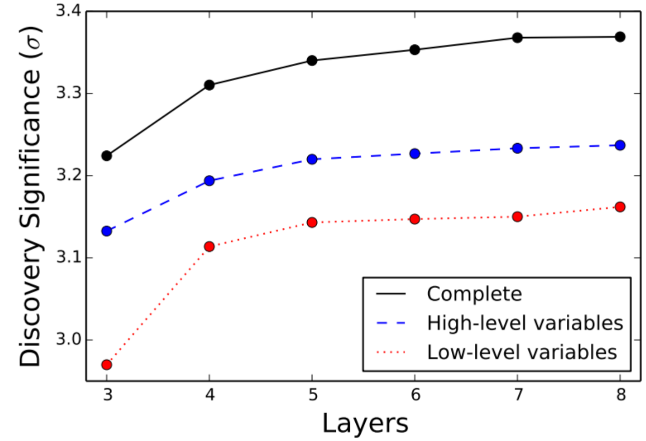
Nature Communications 5, 4308 (2014)
Phys. Rev. Lett. 114, 111801 (2015)

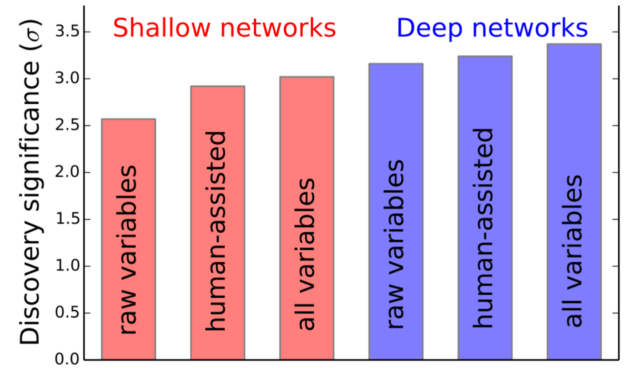
- Compare dense Deep NN against BDT's and shallow NN's

- Deep NN found to outperform shallow NN and BDT's
  - small but statistically significant gain over simpler ML algorithms

- Physicists are good at doing physics!
  - Typical physics variables are high performing (e.g. invariant mass, Razor, etc.)
  - But Deep NN's can learn well from only 4-vector inputs

H→ττ benchmark





BSM Higgs benchmark

	AUC		
Technique	Low-level	High-level	Complete
BDT	0.73 (0.01)	0.78 (0.01)	0.81 (0.01)
NN	0.733 (0.007)	0.777 (0.001)	0.816 (0.004)
DN	0.880 (0.001)	0.800 (< 0.001)	0.885 (0.002)

	Discovery significance		
Technique	Low-level	High-level	Complete
NN	2.5σ	3.1σ	3.7σ
DN	4.9σ	3.6σ	5.0σ

What is deep learning doing?

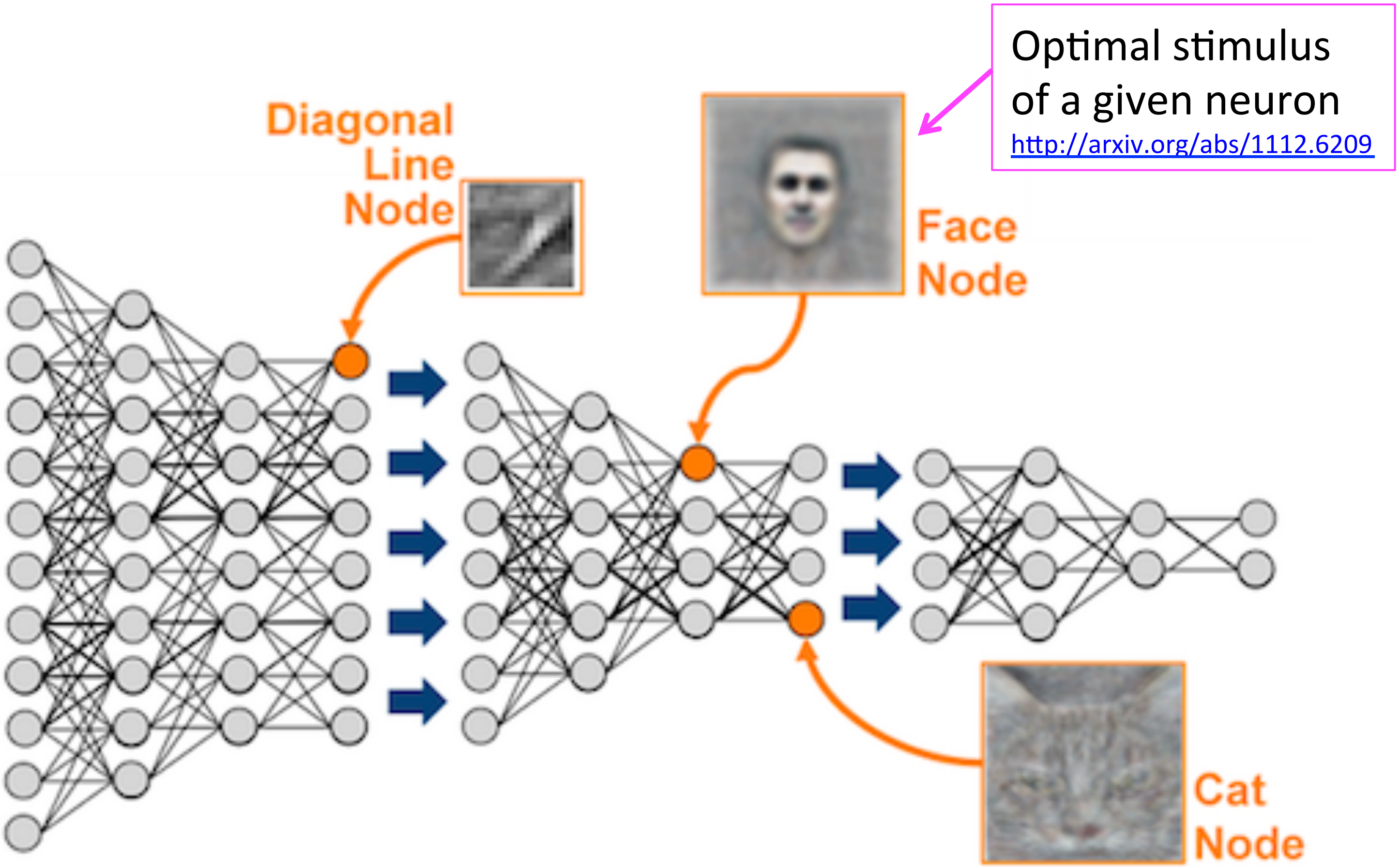
- **Hierarchical learning of representations**

- 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!

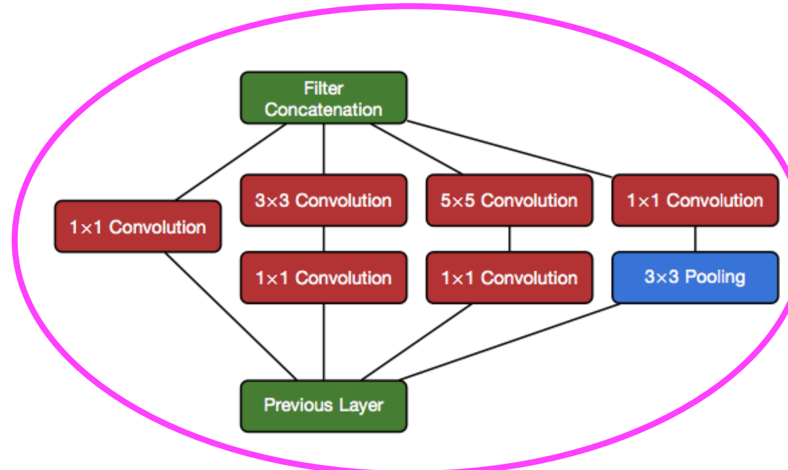
Higher Level Representations

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



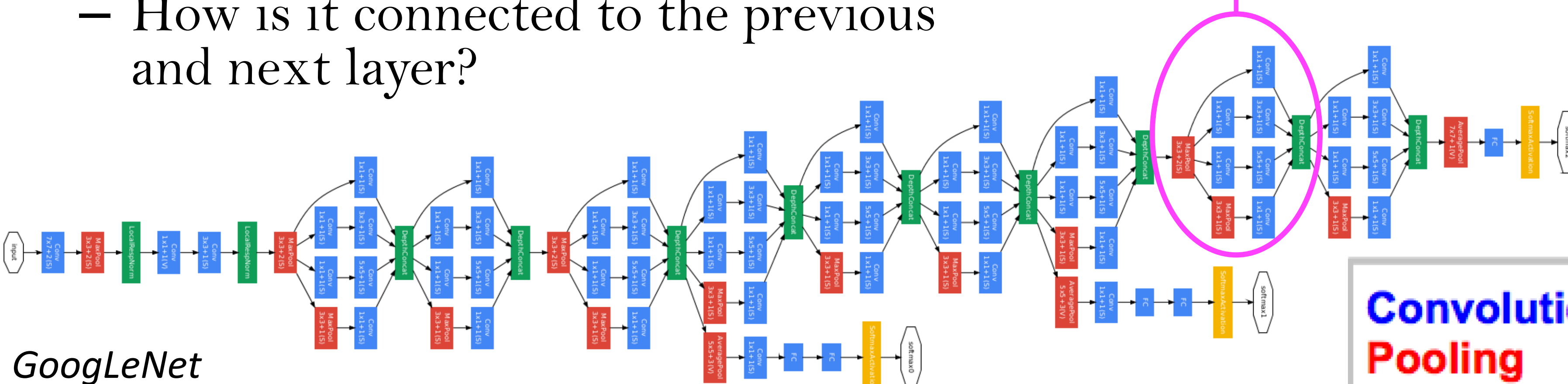
NOT Simple Feedforward Neural Networks

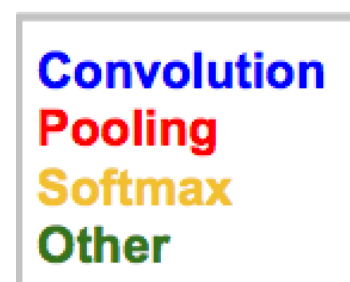
- NN's as a **complex graph**
  - Nodes of graph are the layers
  - Edges of graph are data flow
  - Layers added to achieve a specific task, e.g. regularization



Inception module
"Network-in-network"

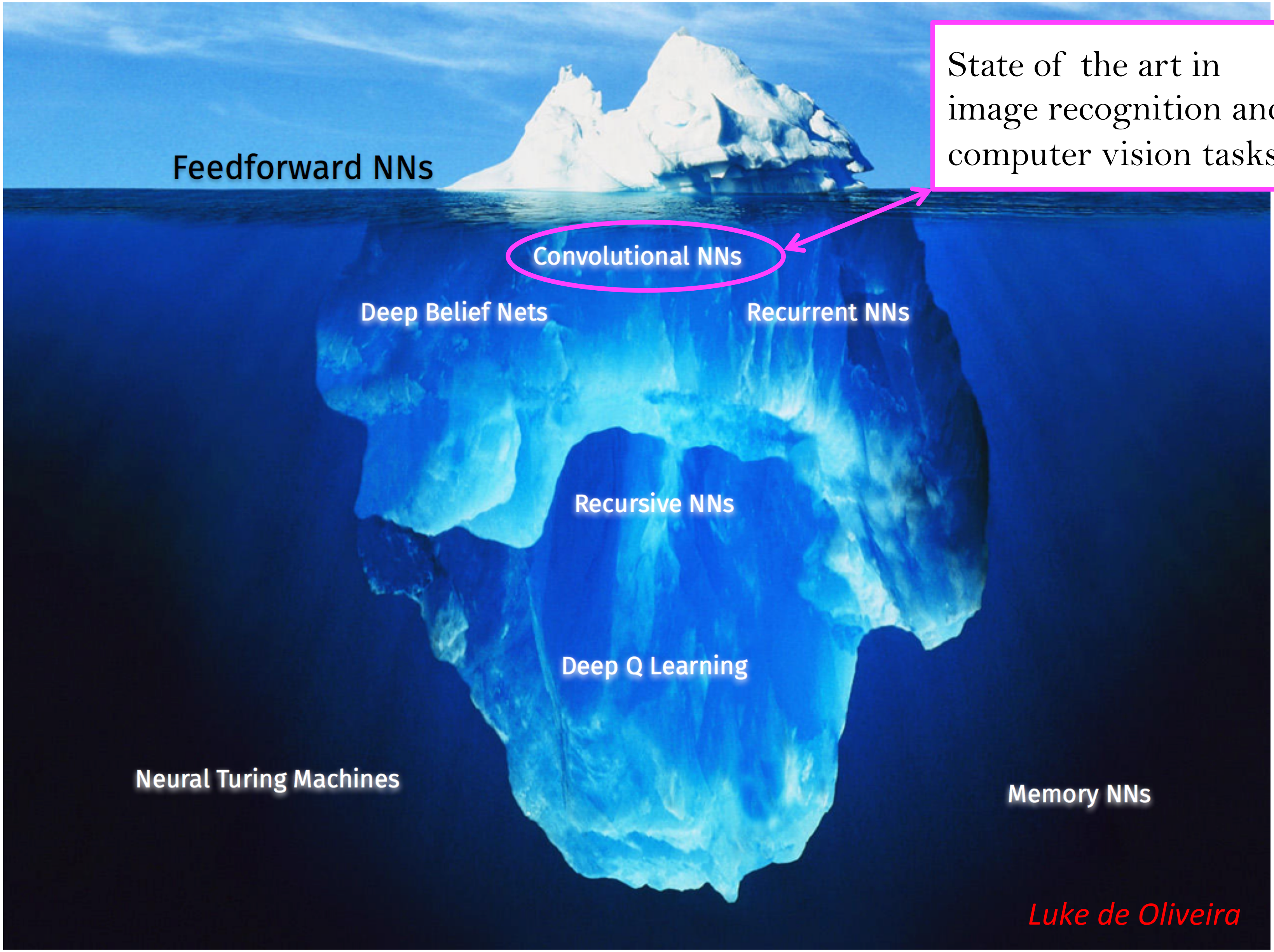
- Better to ask:
  - What does each layer / module do?
  - How is it connected to the previous and next layer?





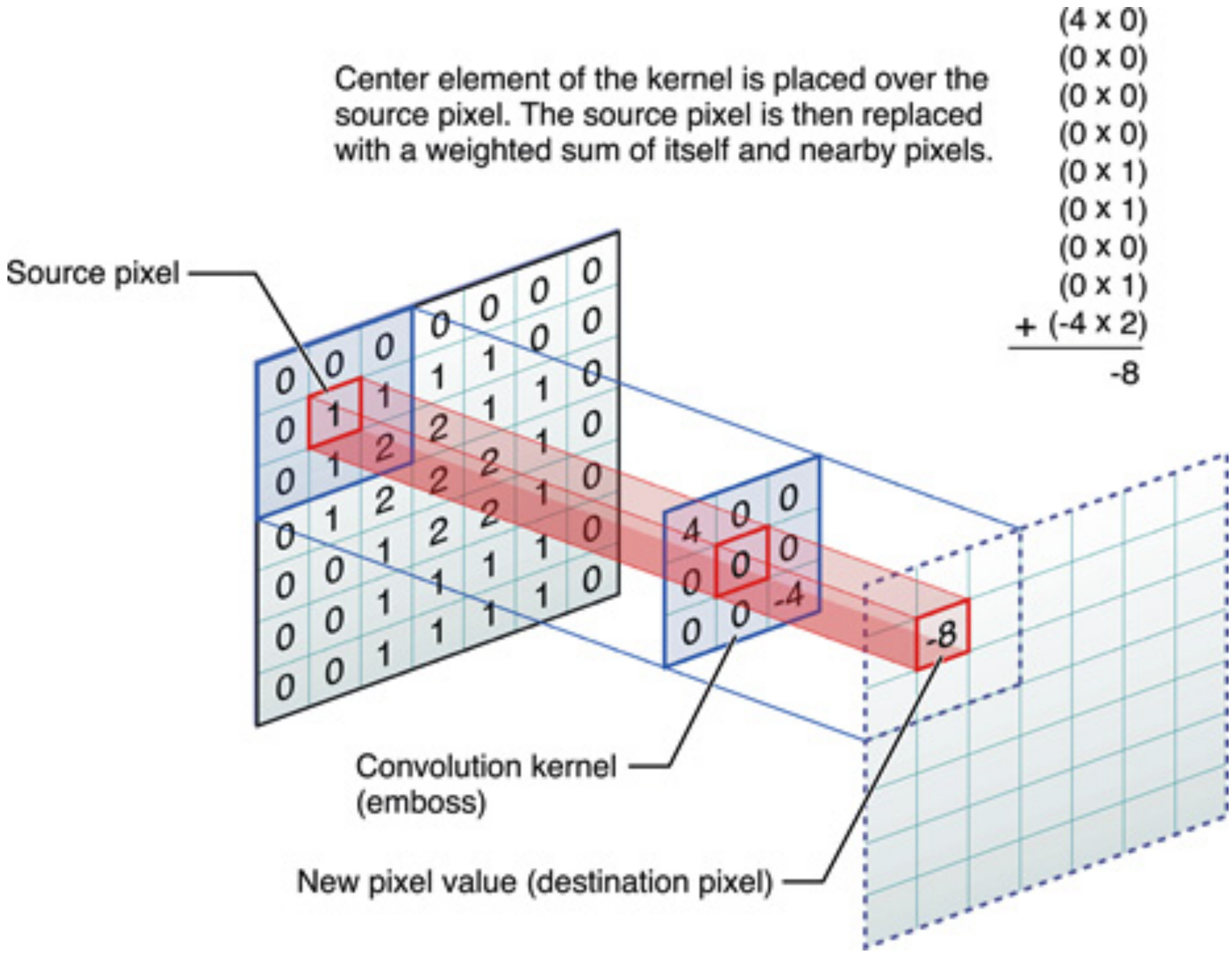
GoogLeNet
ILSVRC 2014 Winner
4M parameters

The Tip of the Iceberg

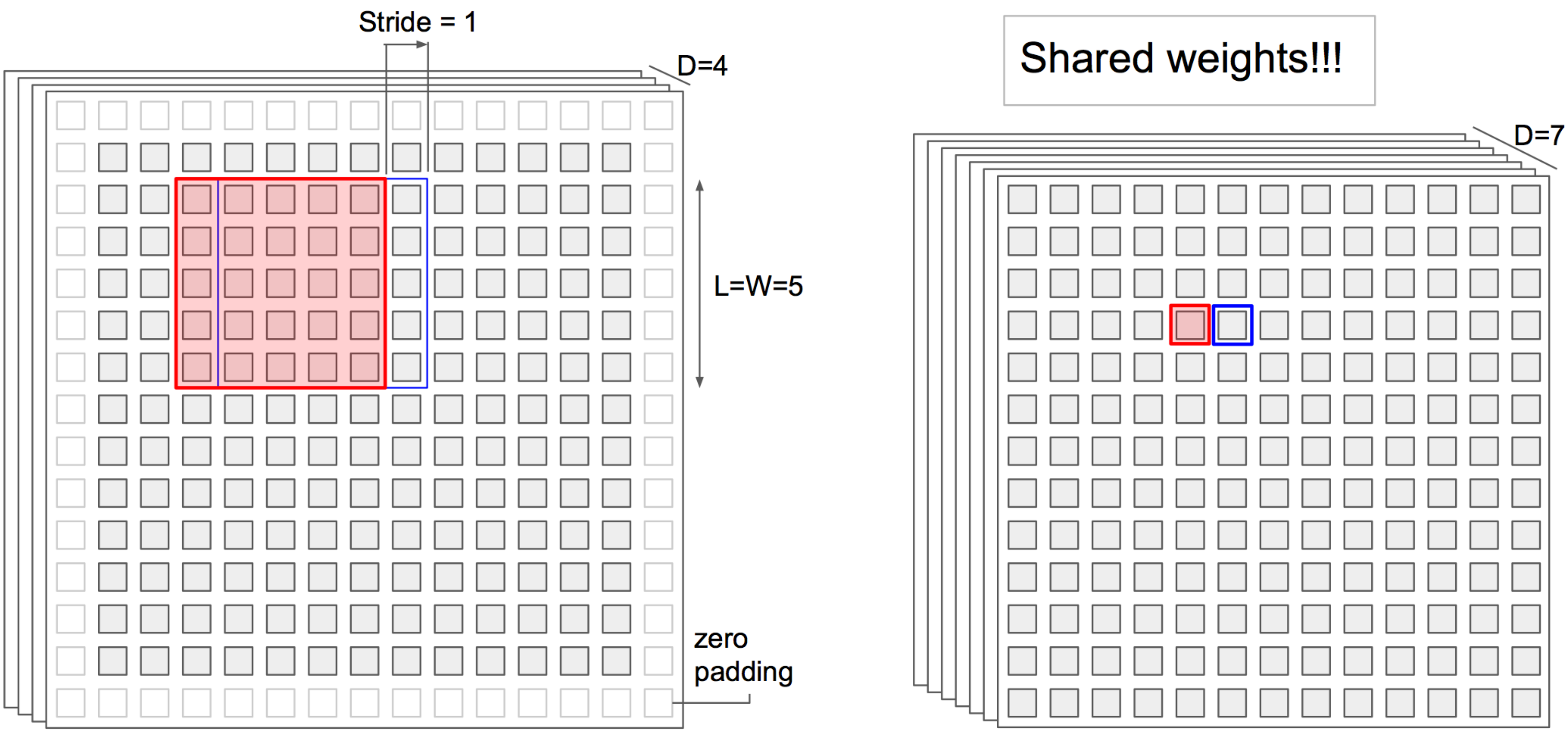


State of the art in image recognition and computer vision tasks

Convolution in 2D



Convolutions in 2D

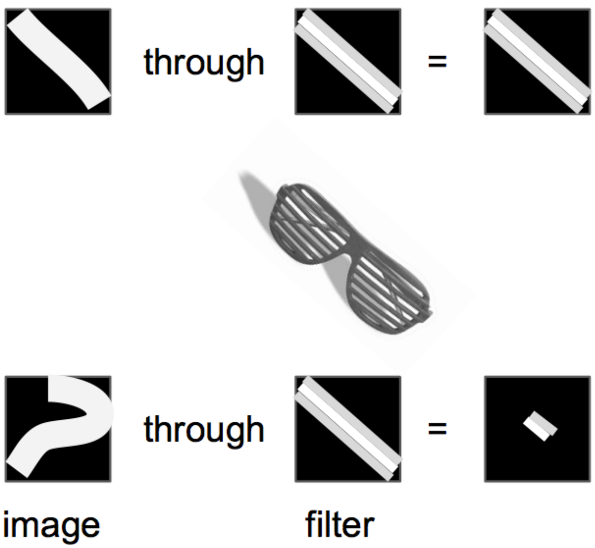


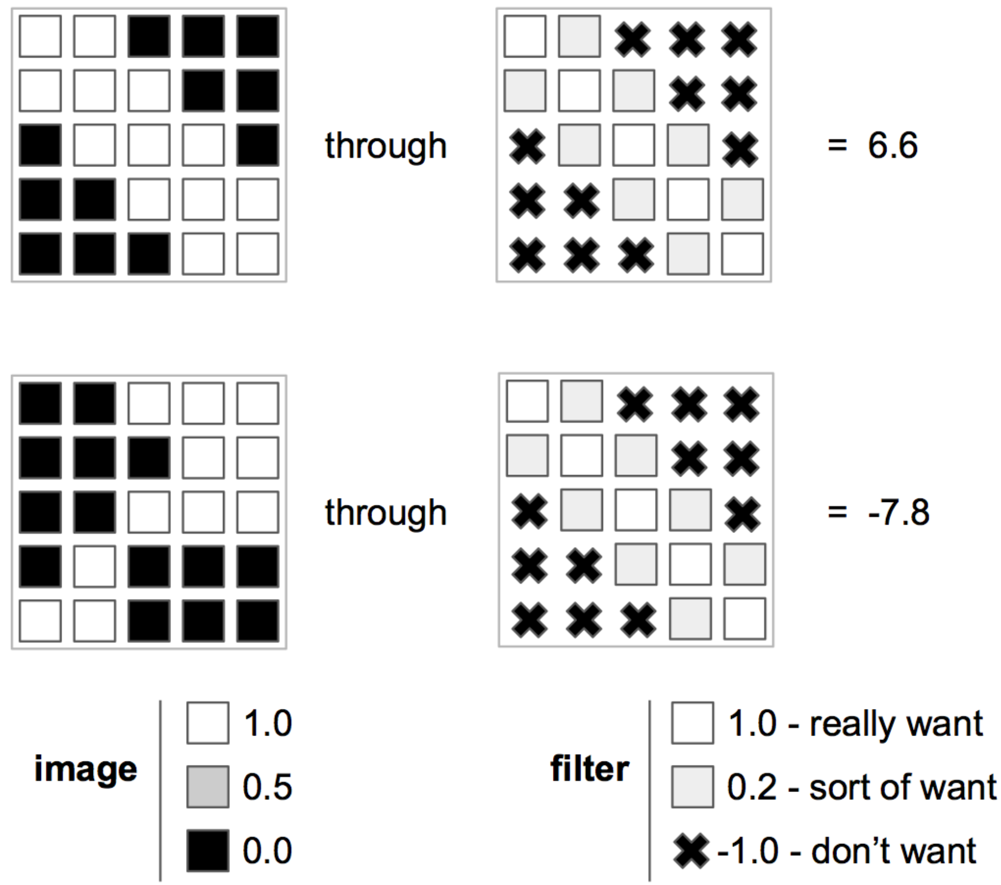
Input image

Convolved image

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

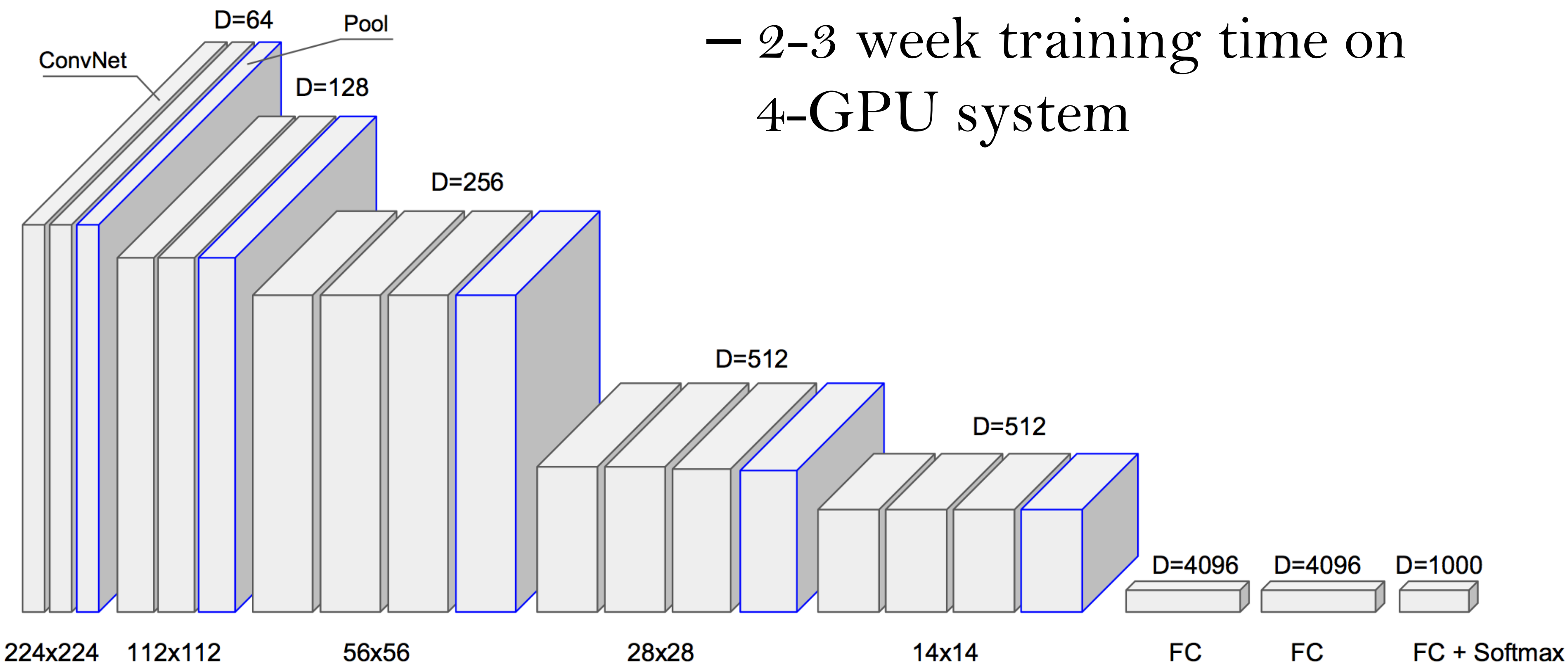
What do filters do?



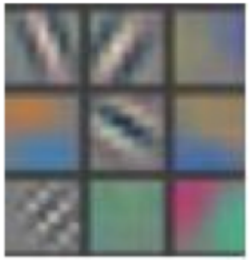


VGGNet (2014)

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



Representation Learning



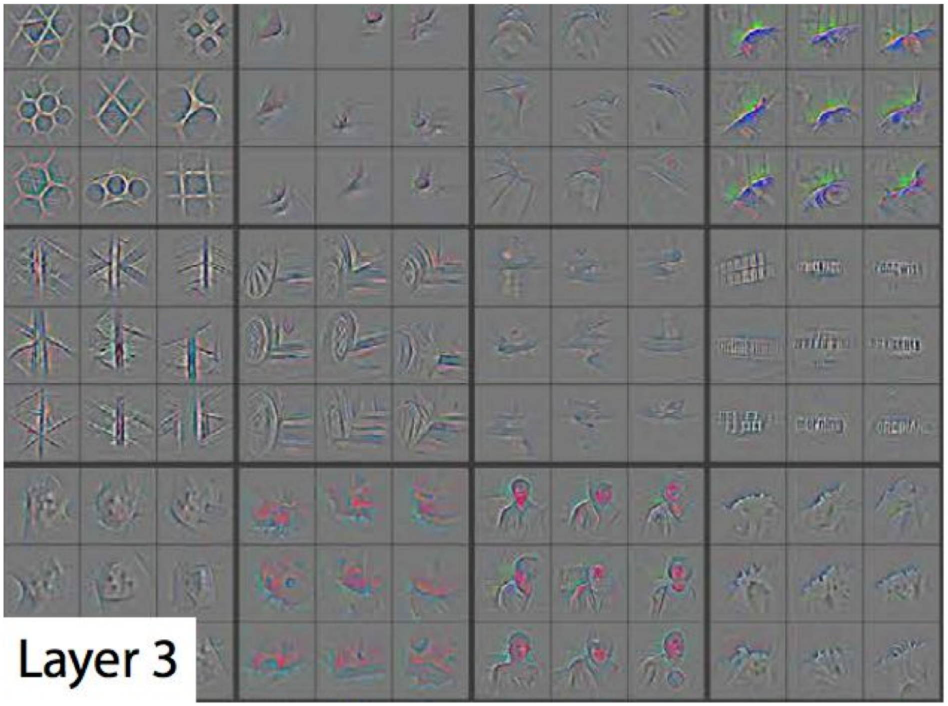
Filter

Matching images

Layer 1

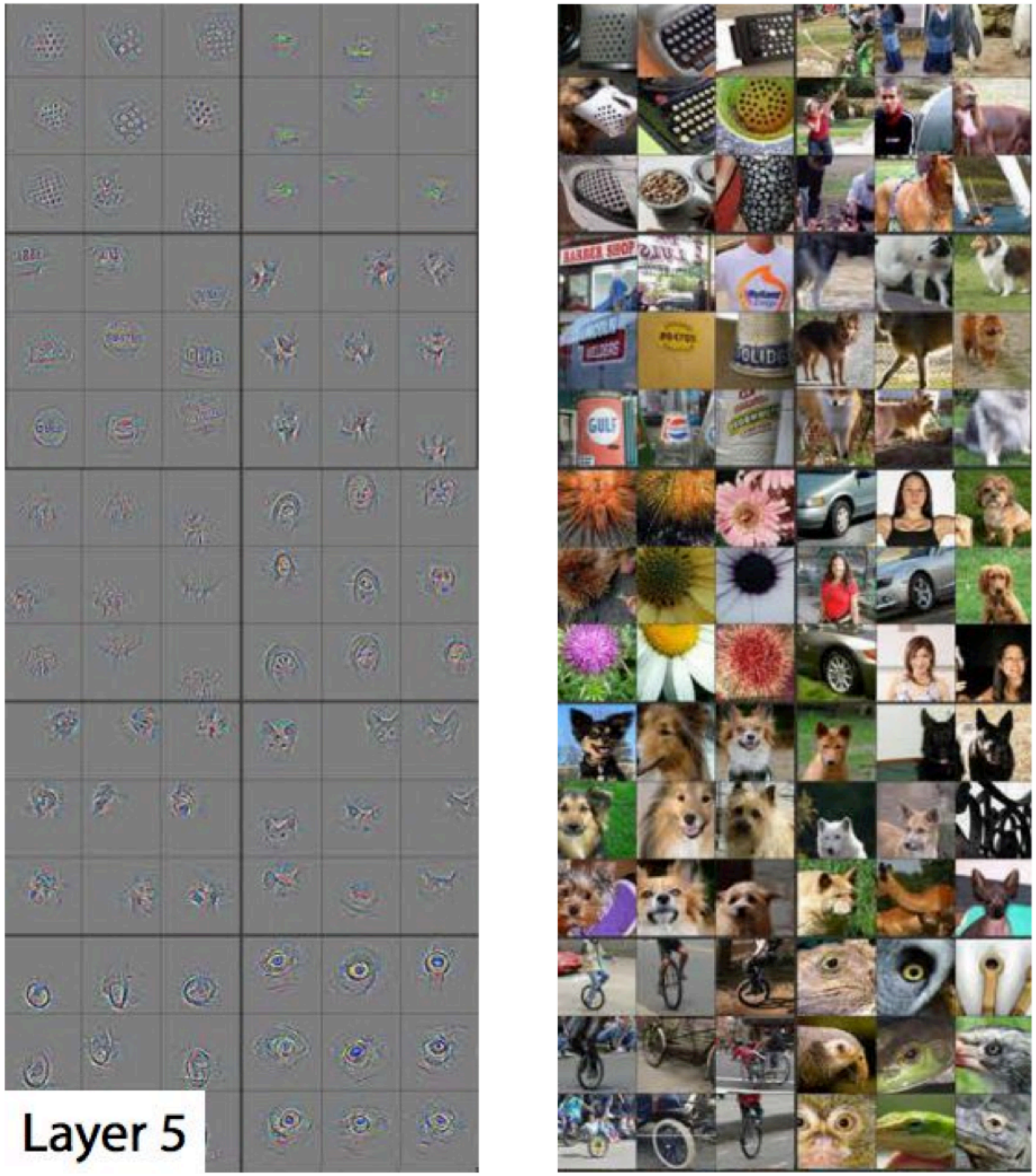
L. Monier, G. Renard, https://github.com/holbertonschool/deep-learning

Representation Learning



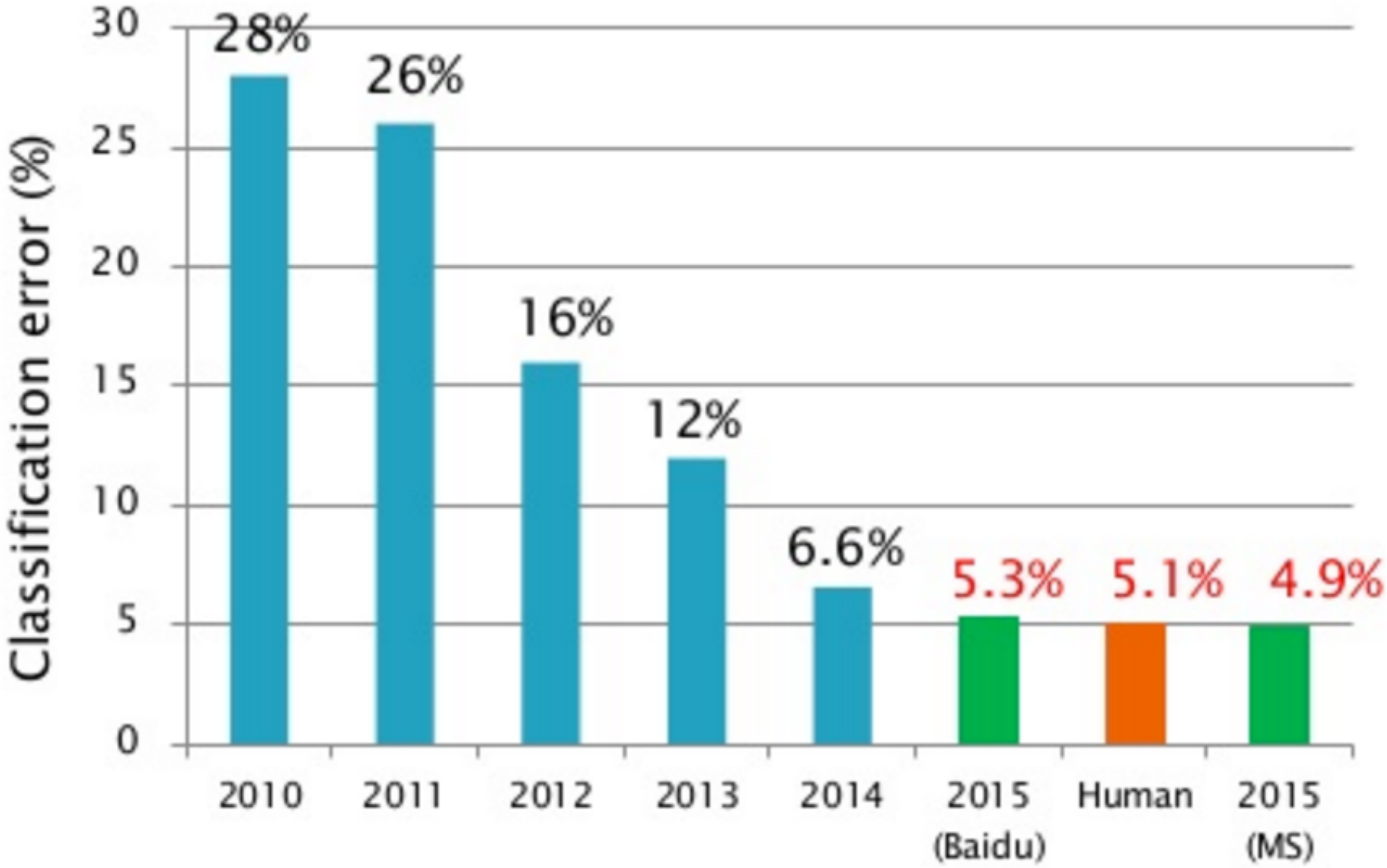
L. Monier, G. Renard, https://github.com/holbertonschool/deep-learning

Representation Learning



L. Monier, G. Renard, https://github.com/holbertonschool/deep-learning

Deep Learning for Image Recognition



- Deep Convolutional Networks now have *super-human* performance in image recognition (ILSVRC Challenge)

Deep learning and High Energy Physics

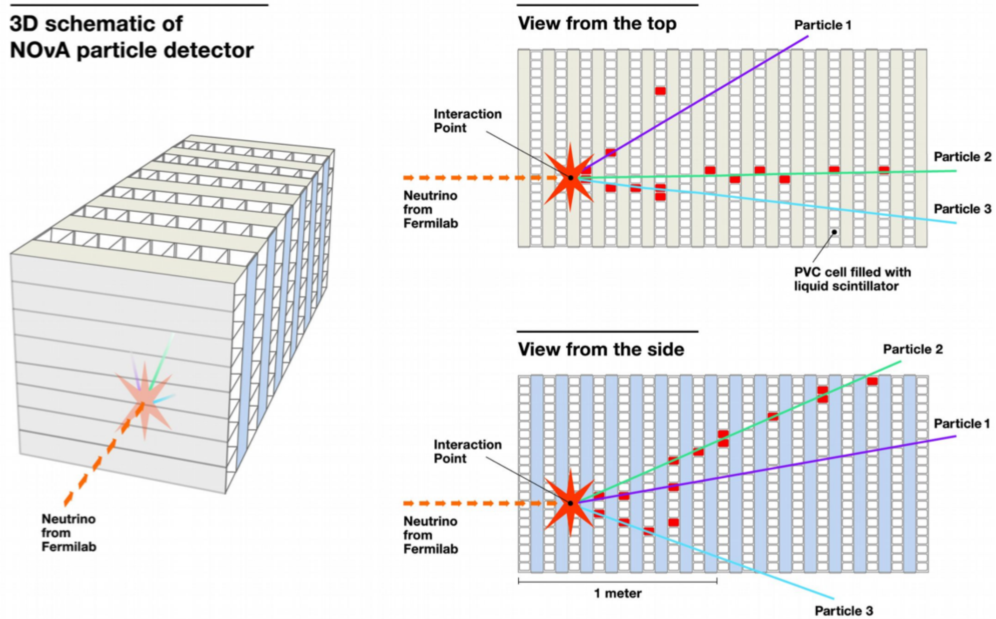
Deep learning and High Energy Physics

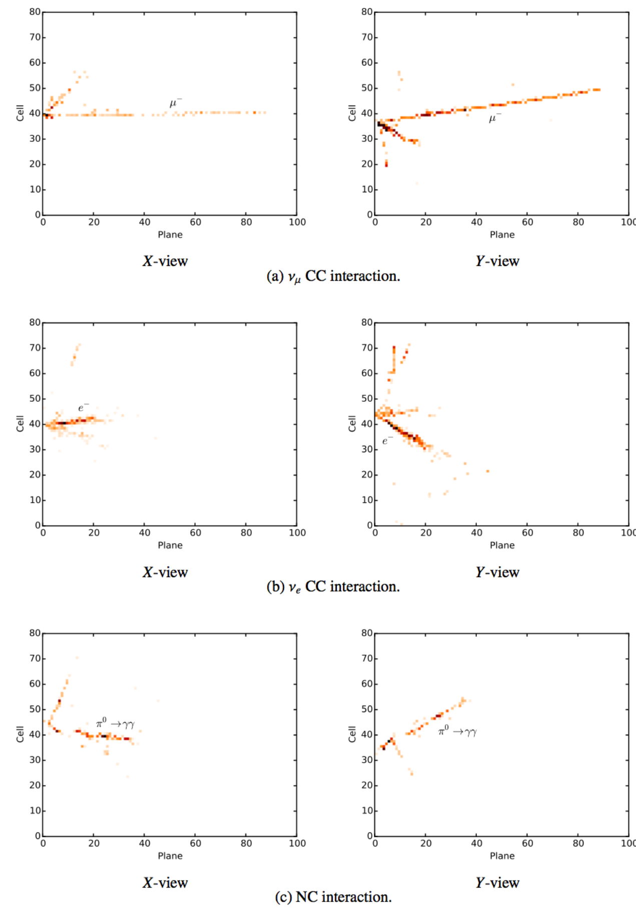
- How can we make use of high-performance deep learning algorithms in HEP?

- 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 news ways that are amenable to deep learning
  - Can we frame HEP questions as if they were image recognition tasks?

Neutrino Identification at NOνA

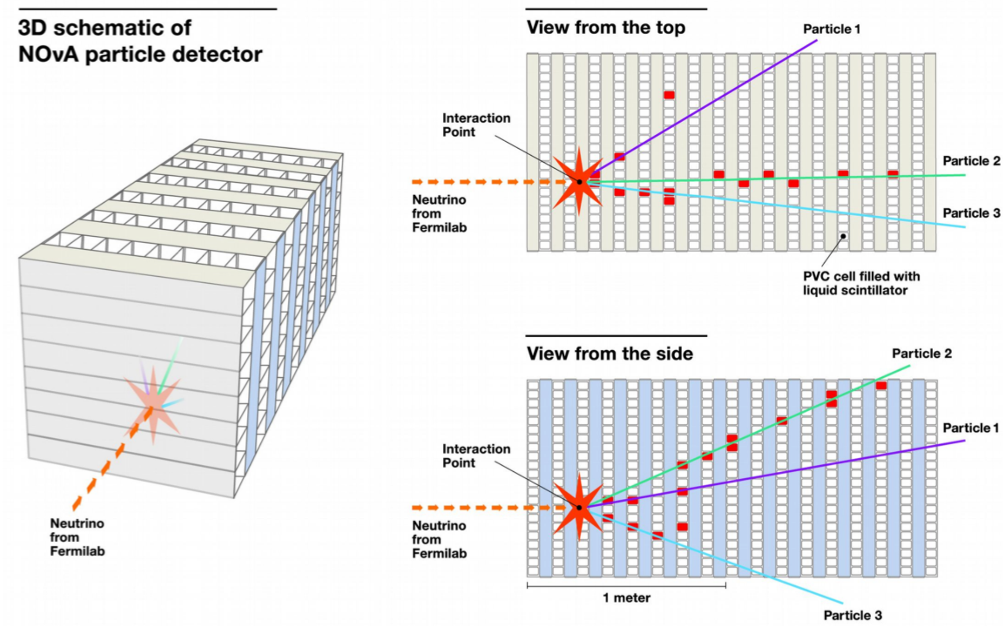


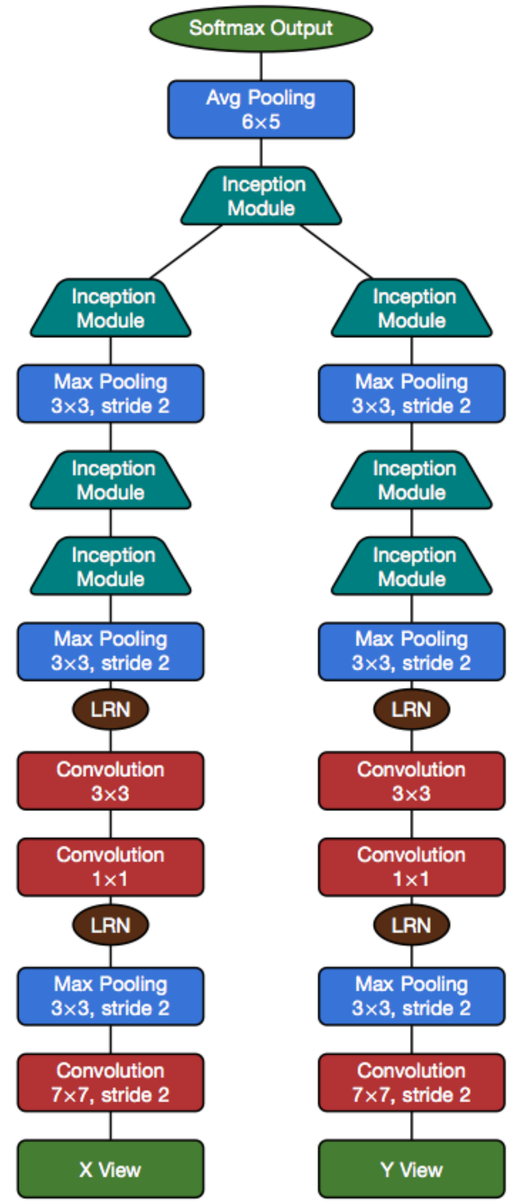


- Two 2D projections of the interactions
- Goal: discriminate between different neutrino interactions / backgrounds

Neutrino Identification at NOνA

arXiv:1604.01444

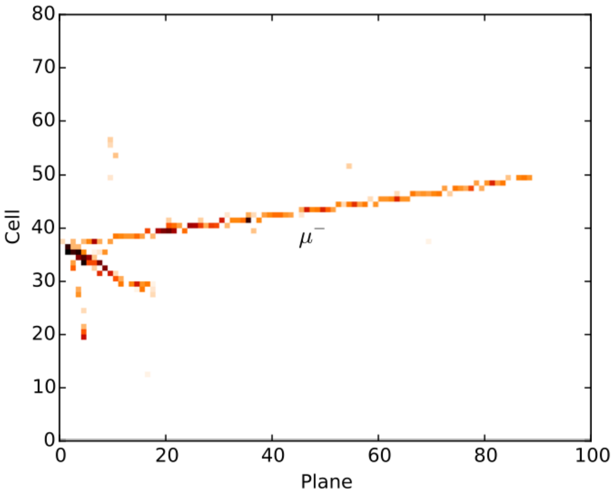


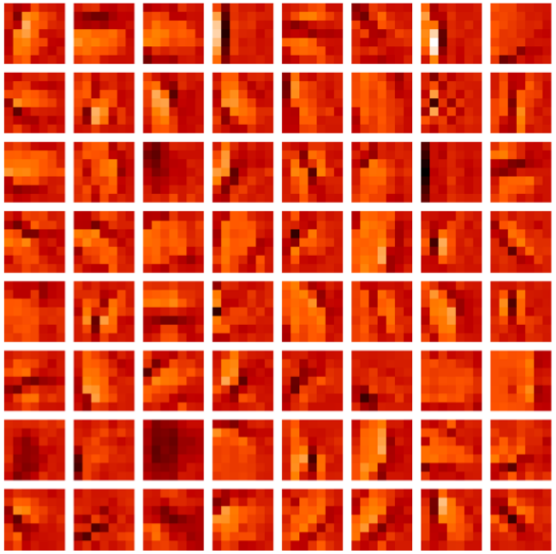


- Treat 2D projections as images
  - Convolutional Neural network for imaging tasks
- Make use of GoogLeNet
  - Use first layers with useful representations for structures in NOνA detector (e.g. edges, …)
  - Train with two image inputs, one for each view

Neutrino Identification at NOνA

arXiv:1604.01444





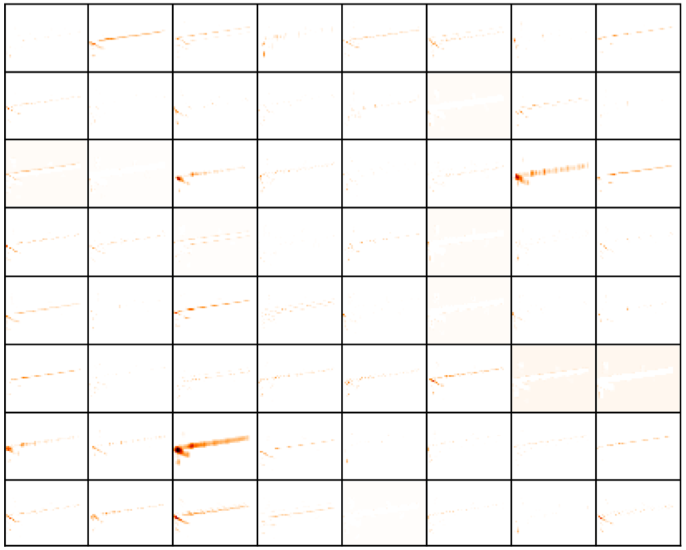
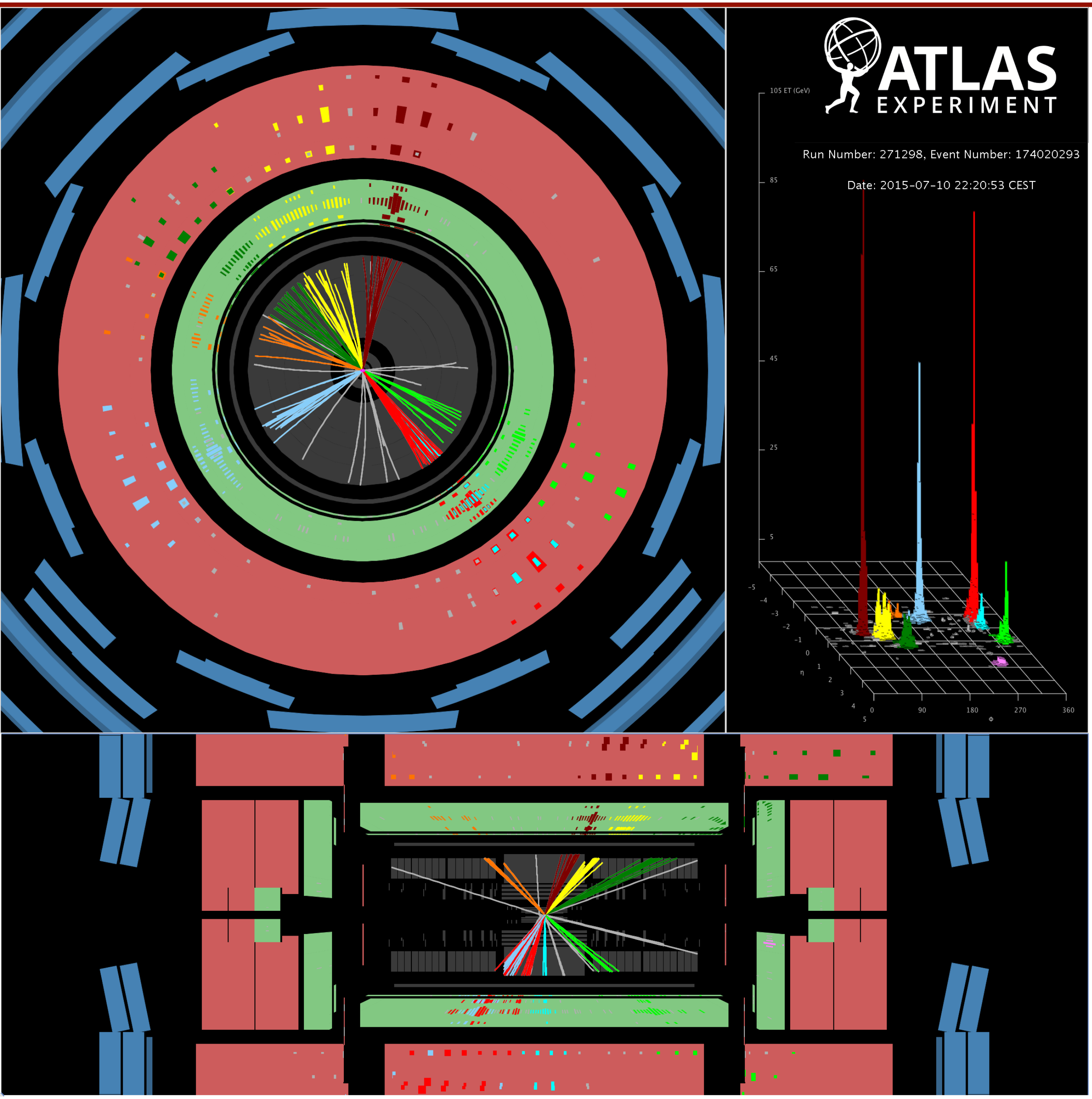


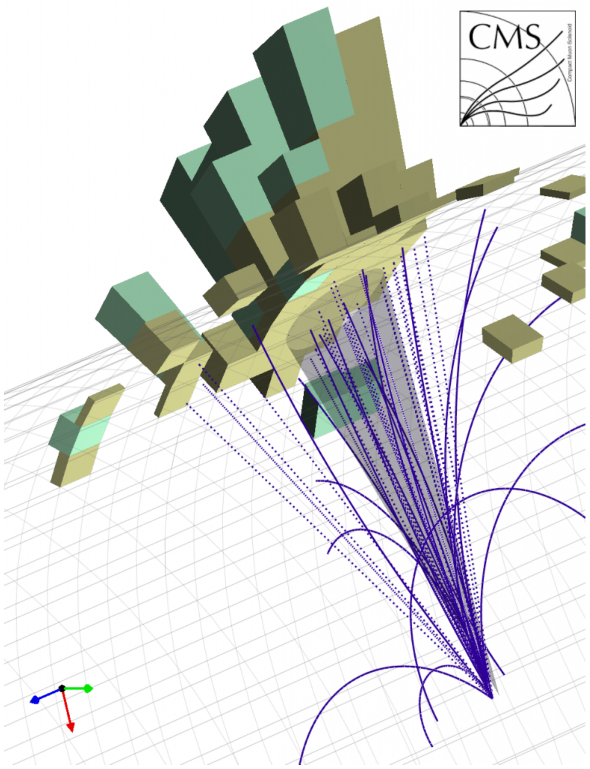
Image Y-view → First CONV layer filters → Output of convolution

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

- Major gains over current algorithms in ν_e-CC discrimination: **35% → 49% signal efficiency for the same background rejection**

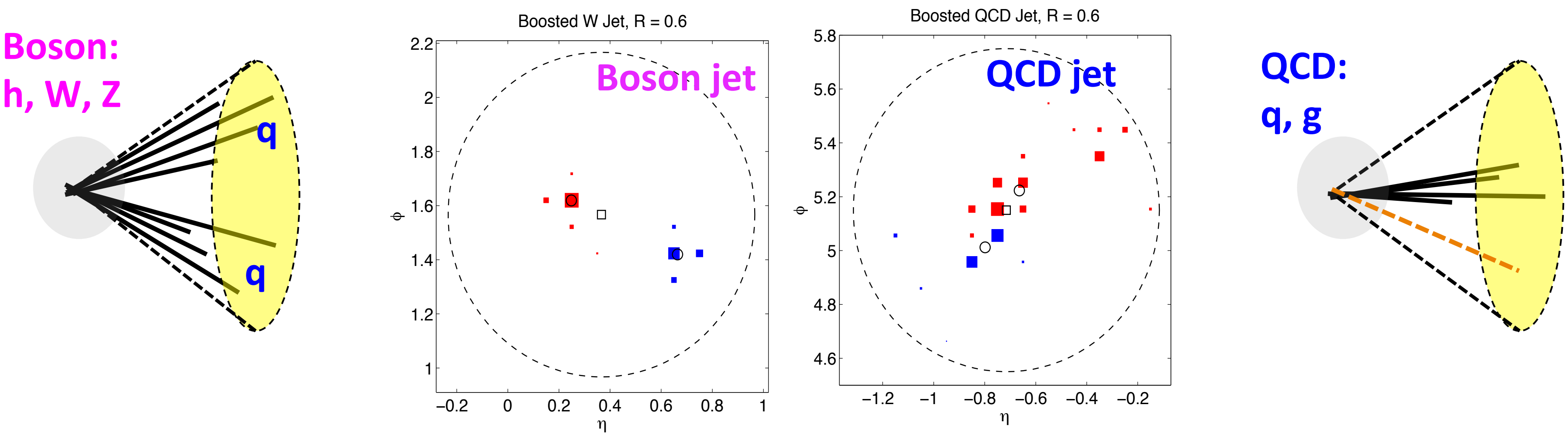
Jets at the LHC





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

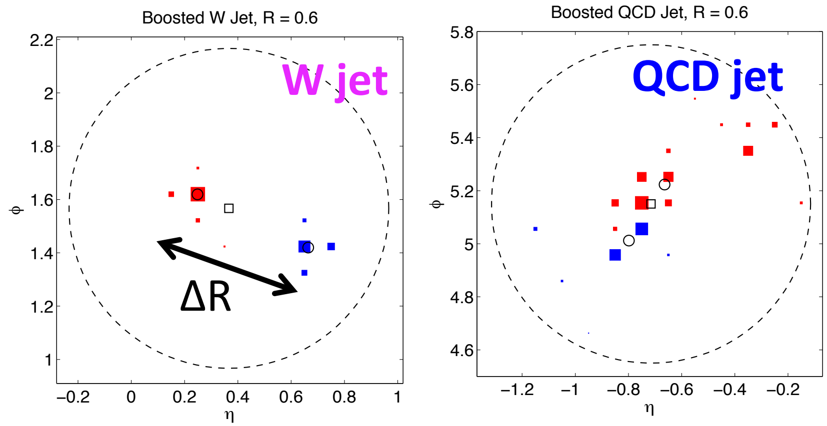
- Can we learn the important information for discrimination directly from the data? And understand what we learned?

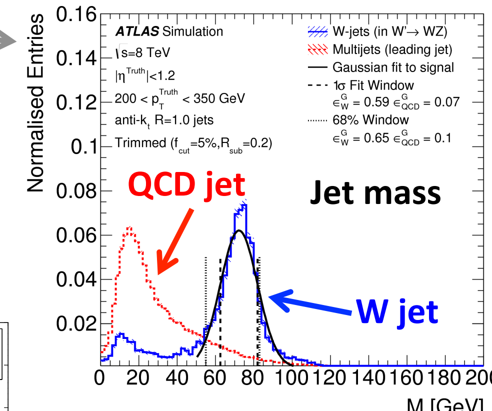
Jet tagging using jet substructure

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

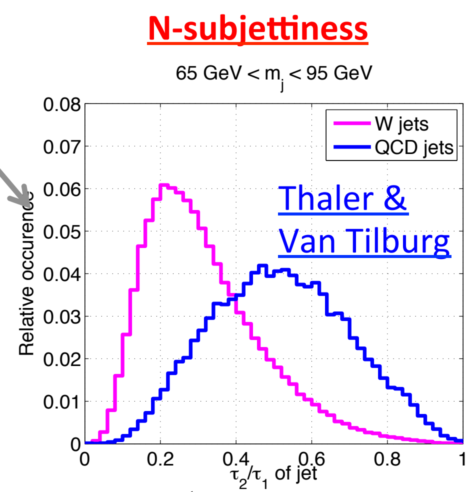
- Typical physics inspired variables exploit differences in:
  - **Jet mass**
  - **N-prong structure**:
    - 1-prong (QCD)
    - 2-prong (W,Z,H)
    - 3-prong (top)
  - **Radiation pattern:**
    - Soft gluon emission
    - Color flow

- Motivated data compressions, inspired by understanding of what should be discriminating…
  - We are likely losing information!





arXiv:1510.05821



$$\tau_N = \frac{1}{d_0} \sum p_{T,k} \min\{\Delta R_{k,axis-1}, \ldots, \Delta R_{k,axis-n}\}$$

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

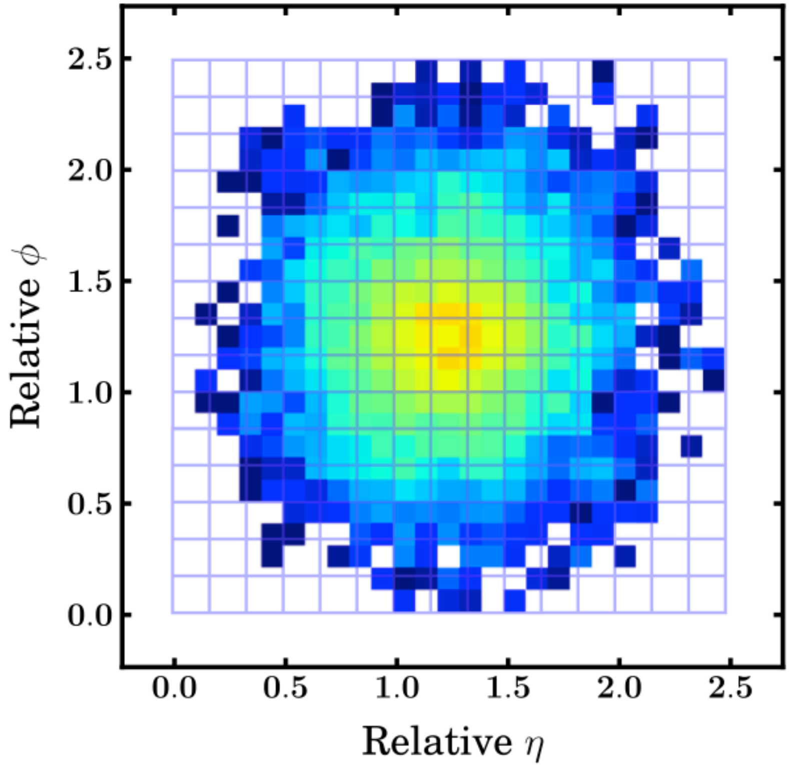
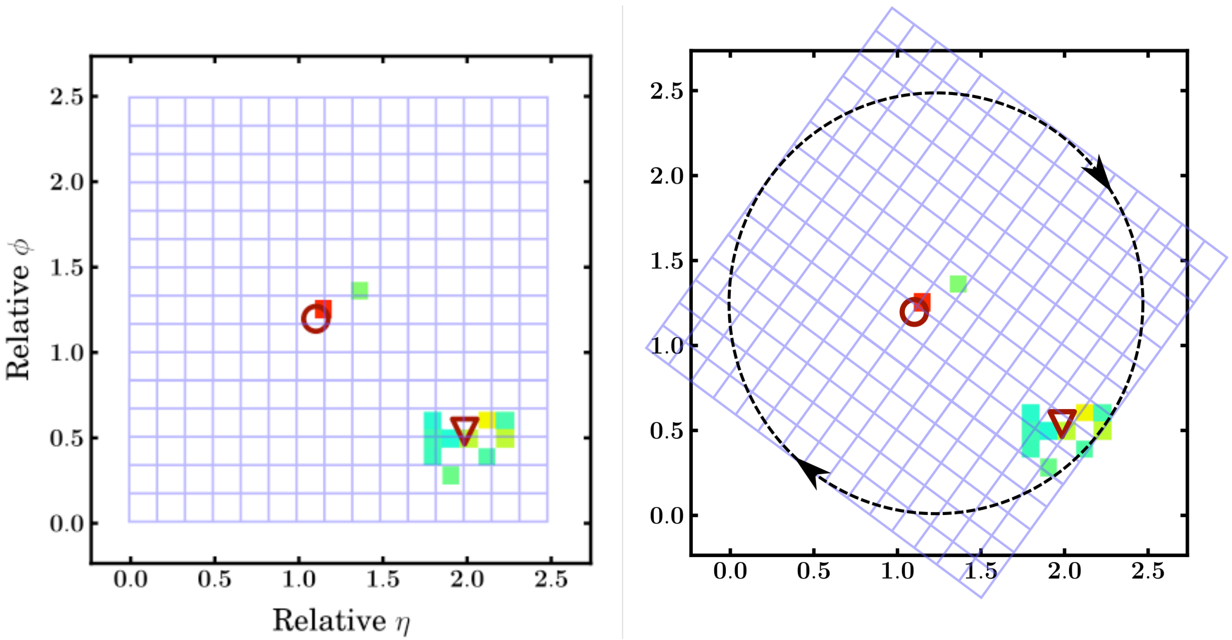
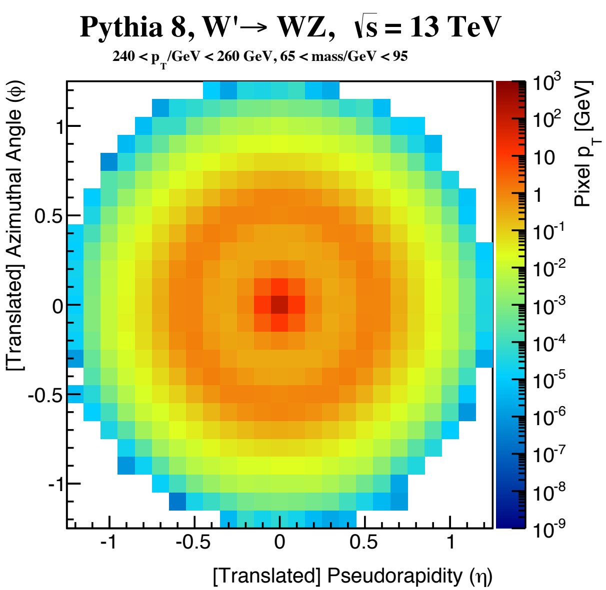


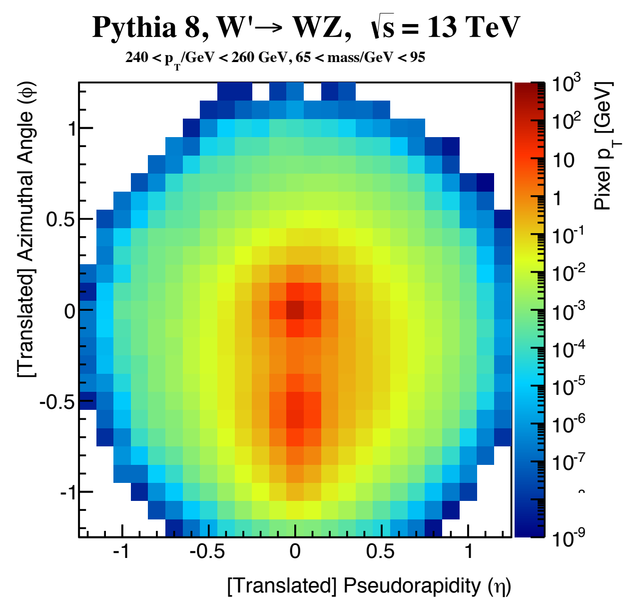
Image pre-processing





Pixelate → Translate → Rotate → Re-grid → Flip

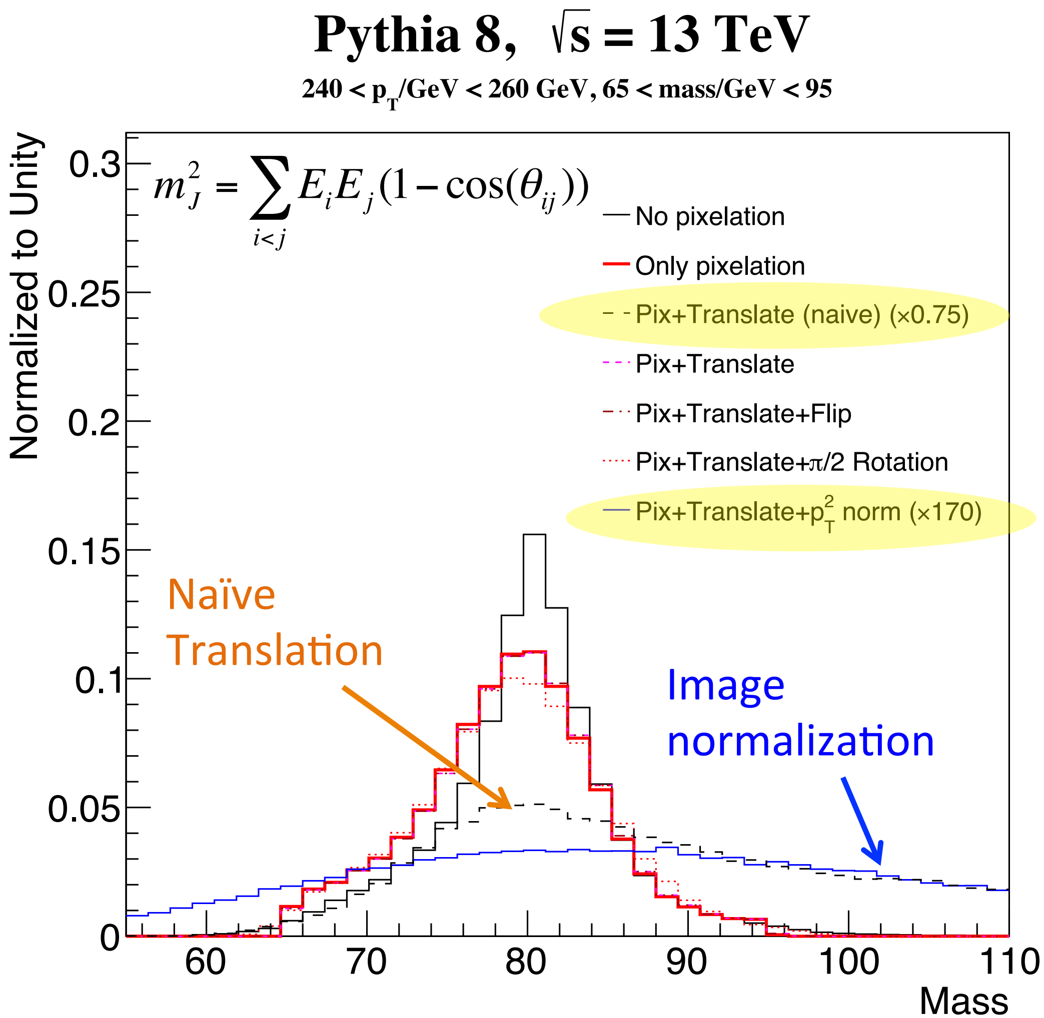
Use subjets to align images. Make use of symmetries: center, rotate, translate



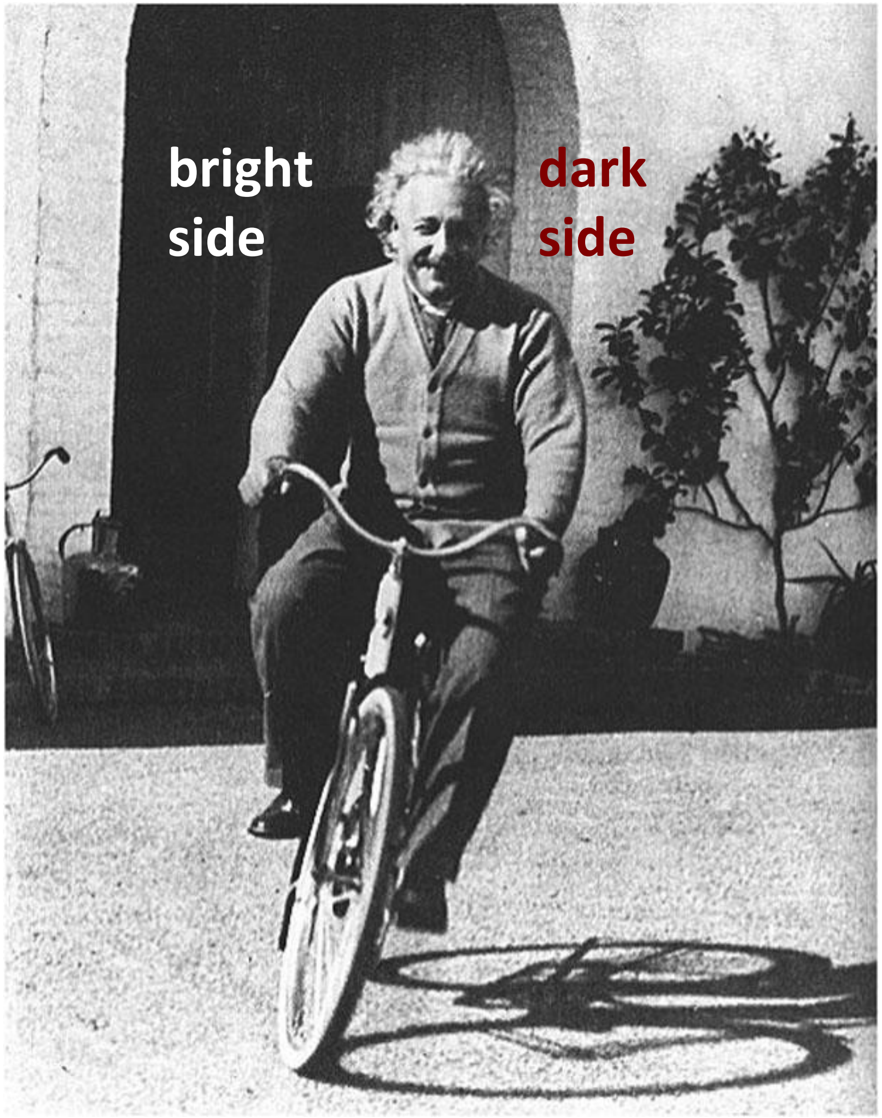
Pre-processing and space-time symmetries

Pre-processing steps may not be Lorentz Invariant

- Translations in η are Lorentz boosts along z-axis
  - Do not preserve the pixel energies
  - Use **transverse energy** rather than energy as pixel intensity
- Jet mass is not invariant under Image normalization



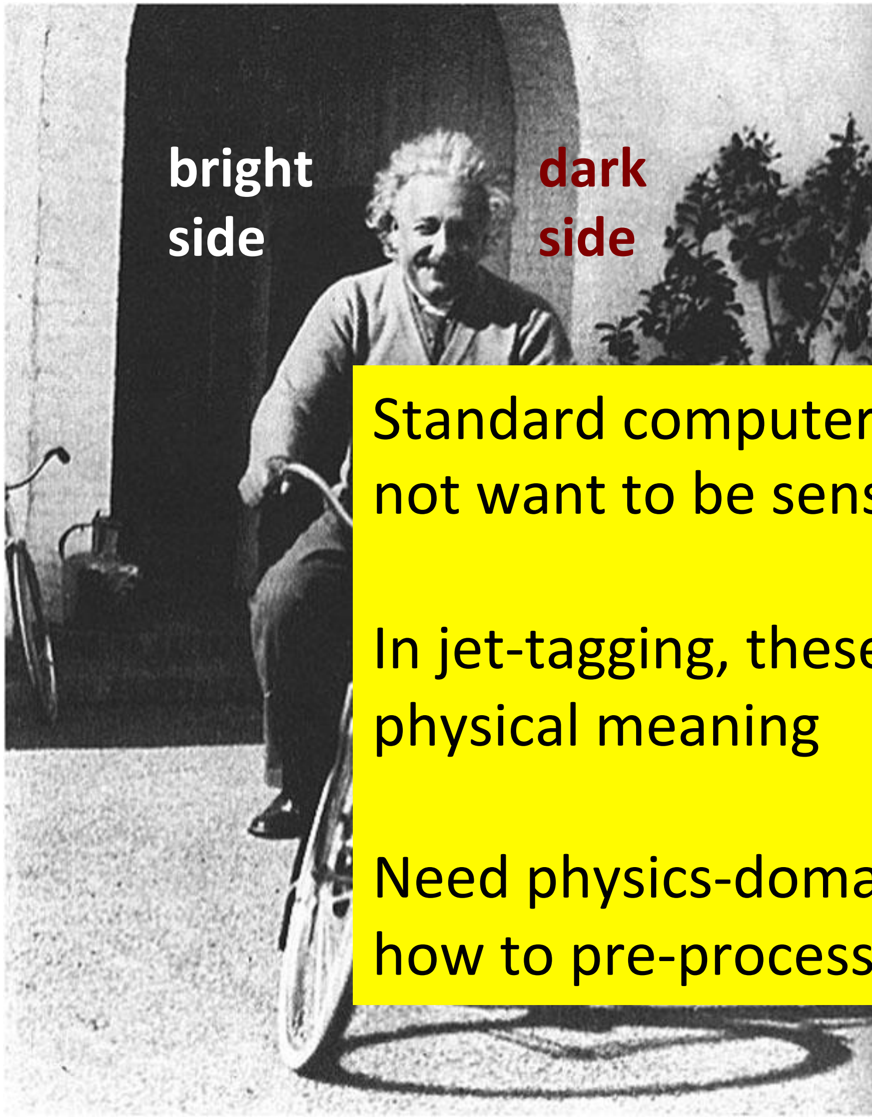
Pre-processing and space-time symmetries

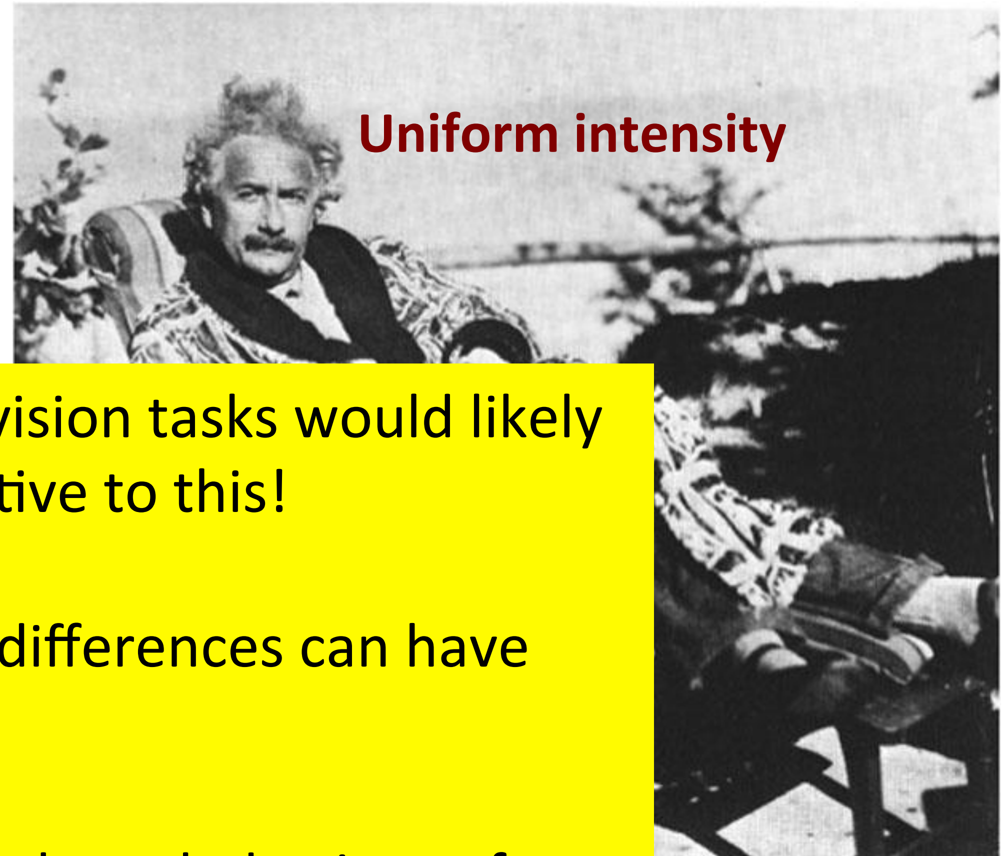


In both pictures the total intensity of Einstein's face is about the same. However, **the image mass is different**!

http://mentalfloss.com/article/49222/11-unserious-photos-albert-einstein

Pre-processing and space-time symmetries





Standard computer vision tasks would likely not want to be sensitive to this!

In jet-tagging, these differences can have physical meaning

Need physics-domain knowledge input for how to pre-process

…sity of

…ne.

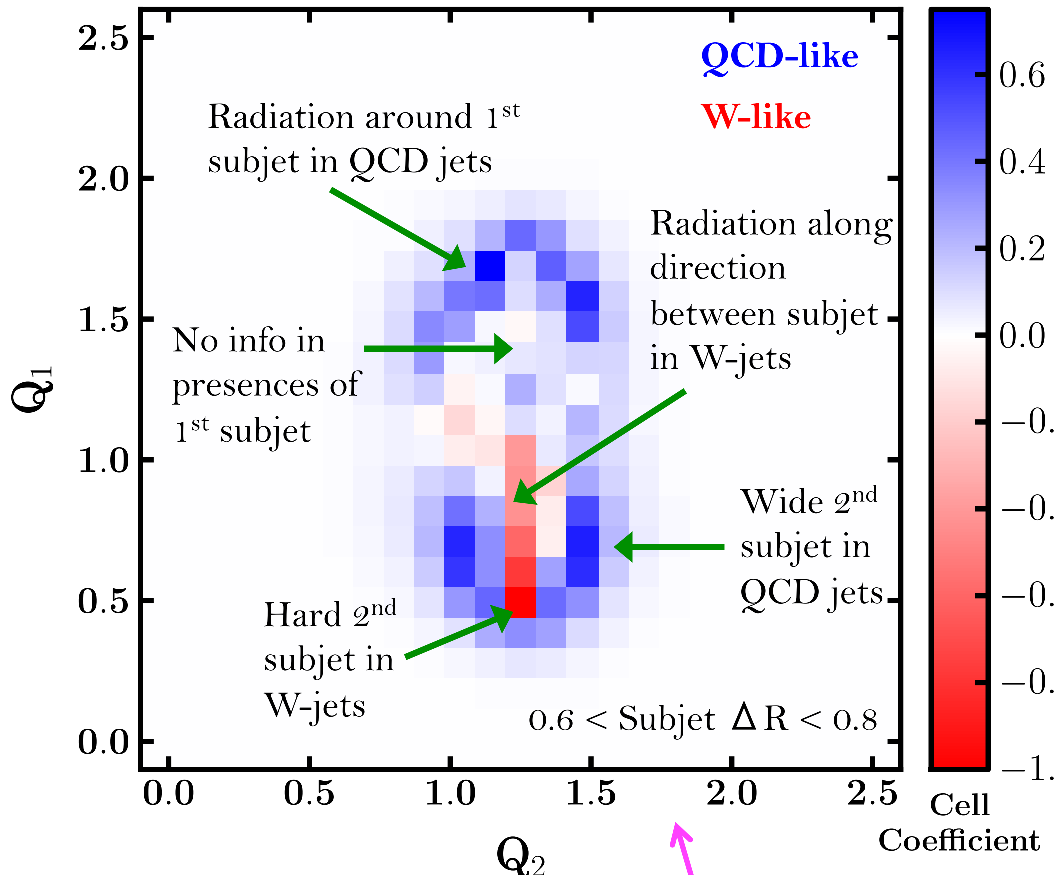
However, the image mass is different!

http://mentalfloss.com/article/49222/11-unserious-photos-albert-einstein

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

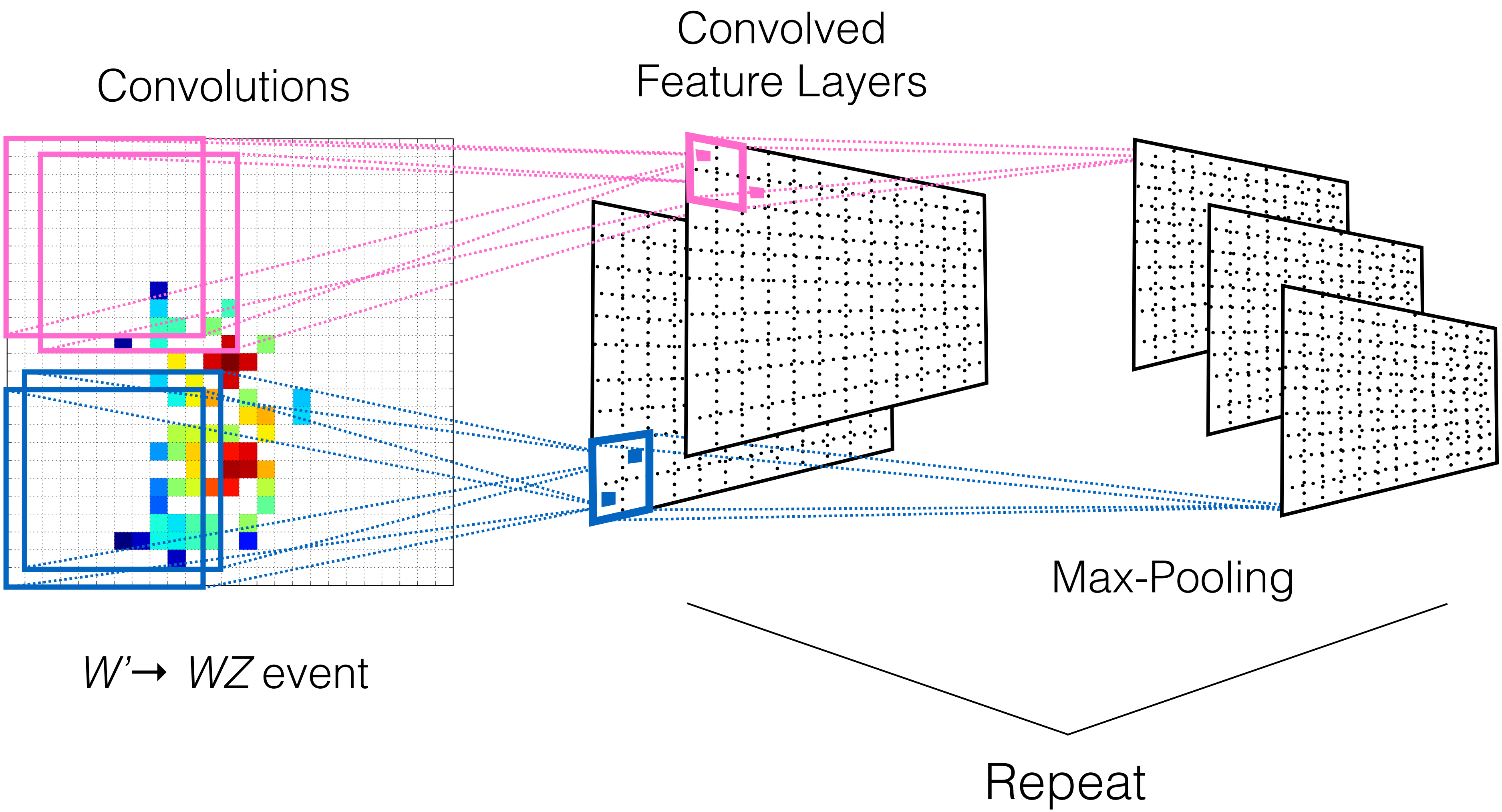
http://arxiv.org/abs/1407.5675



Discriminant $f(x) = w^*x$

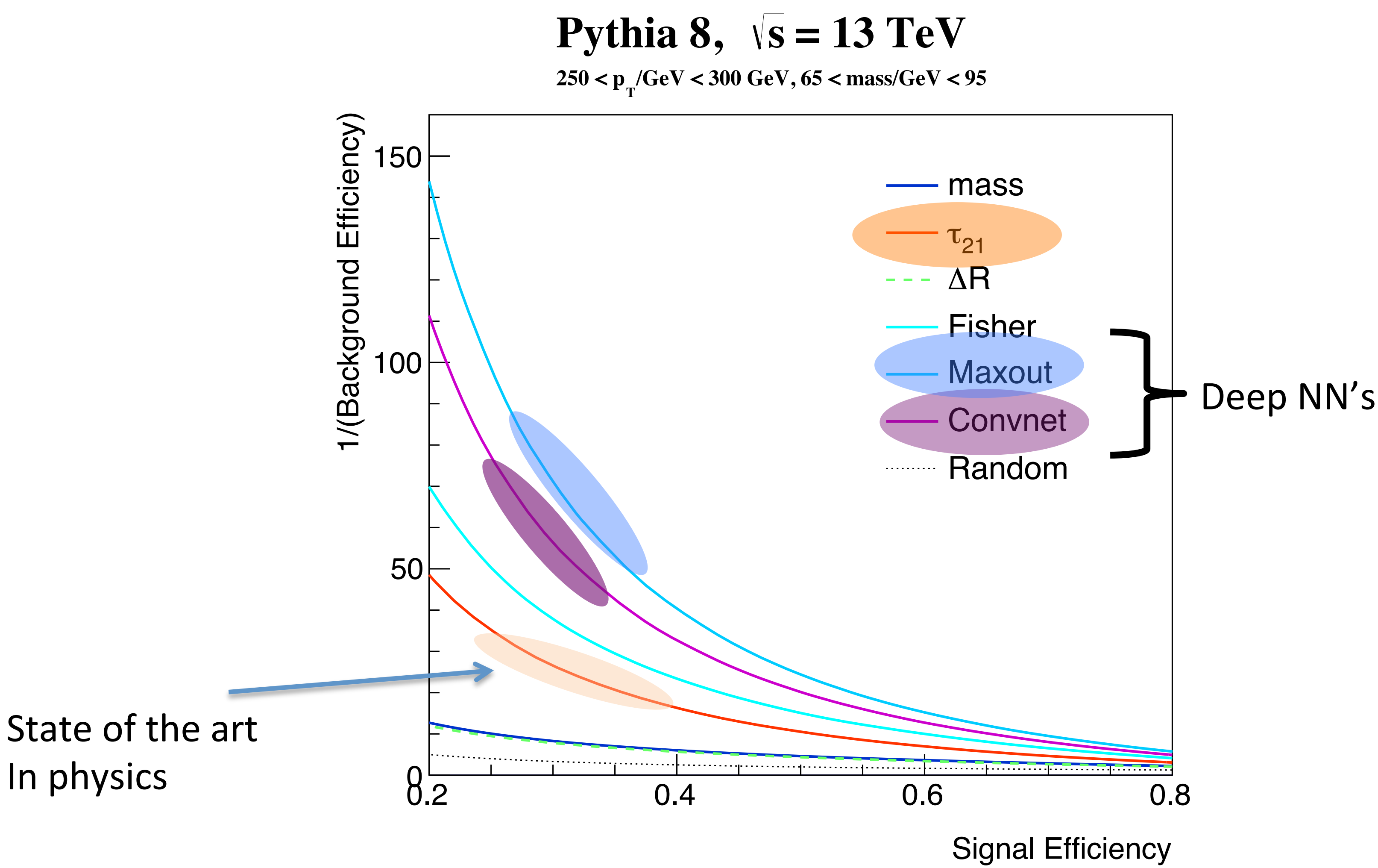
- Linear methods can be limited
  - All the physics inside of a jet is not linear

Deep Jets – Convolutional Neural Networks

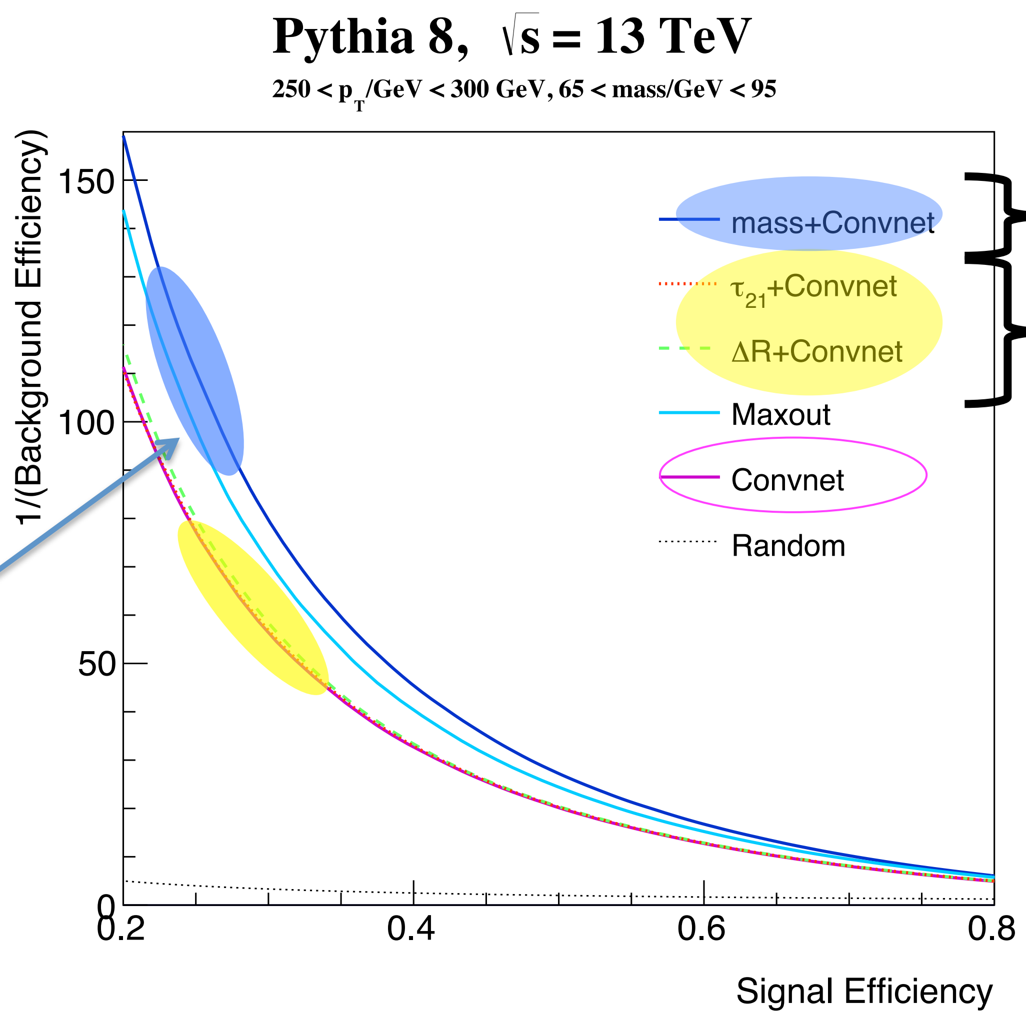


W' → *WZ* event

Performance with Deep Neural Networks



Combining Deep NN's with Substructure Variables



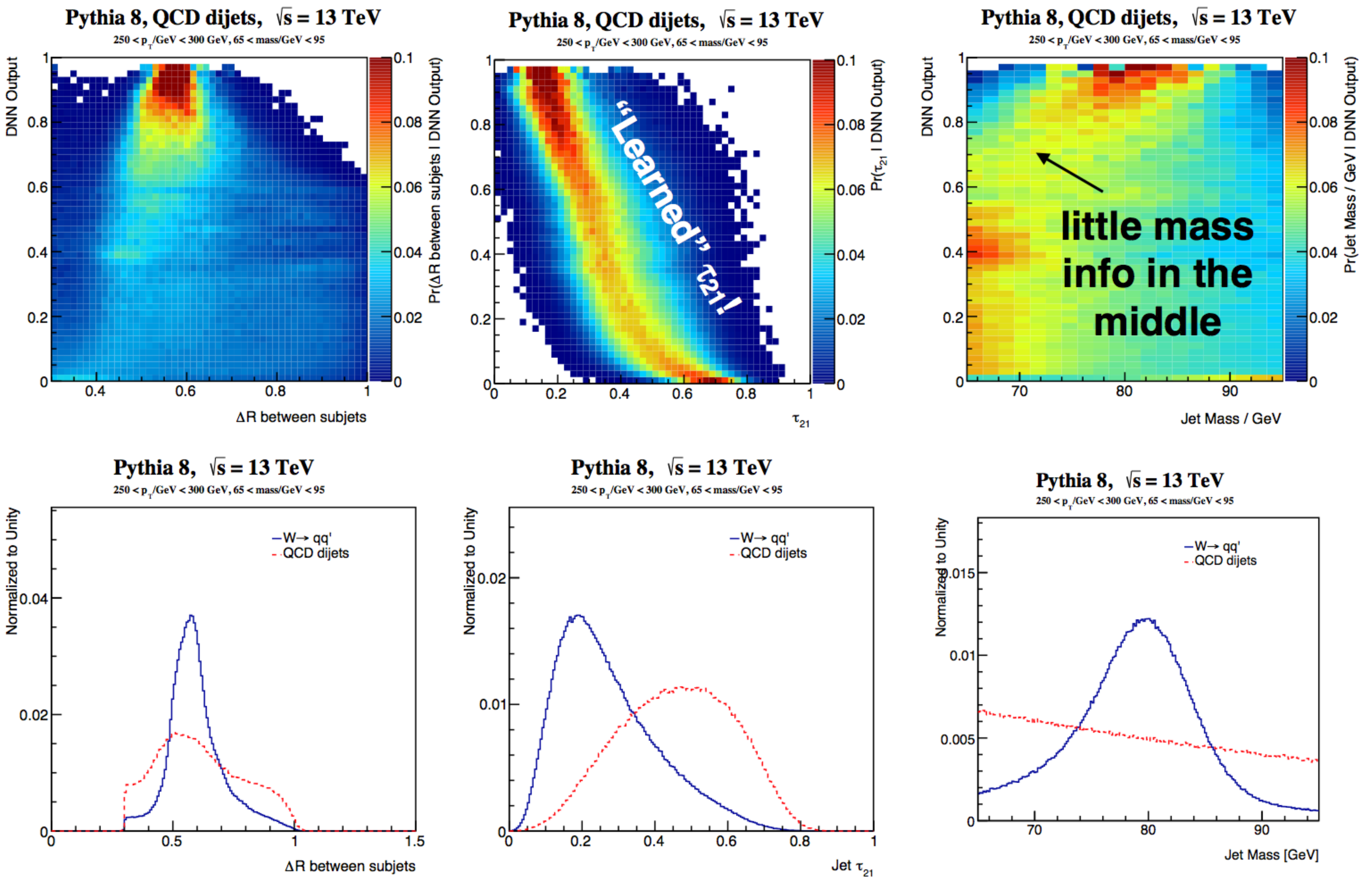
Combine Deep NN's with jet mass

Combine Deep NN's with substructure features

Large gains from combining mass with Deep NN

Deep NN's not fully learning jet mass?

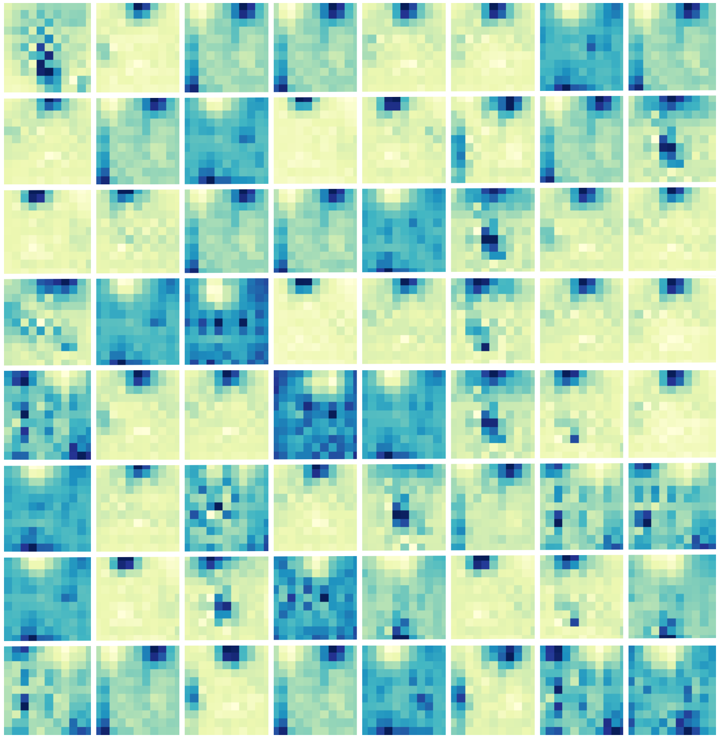
Conditional Correlations with Network Output



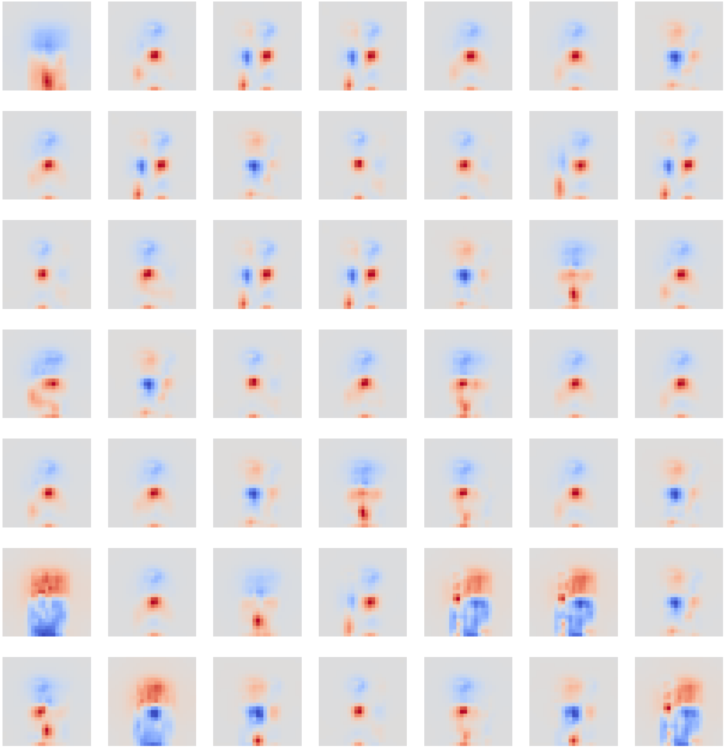
Extracting the Physics

Looking "into" the network to better see what it is learning

Convolved representations



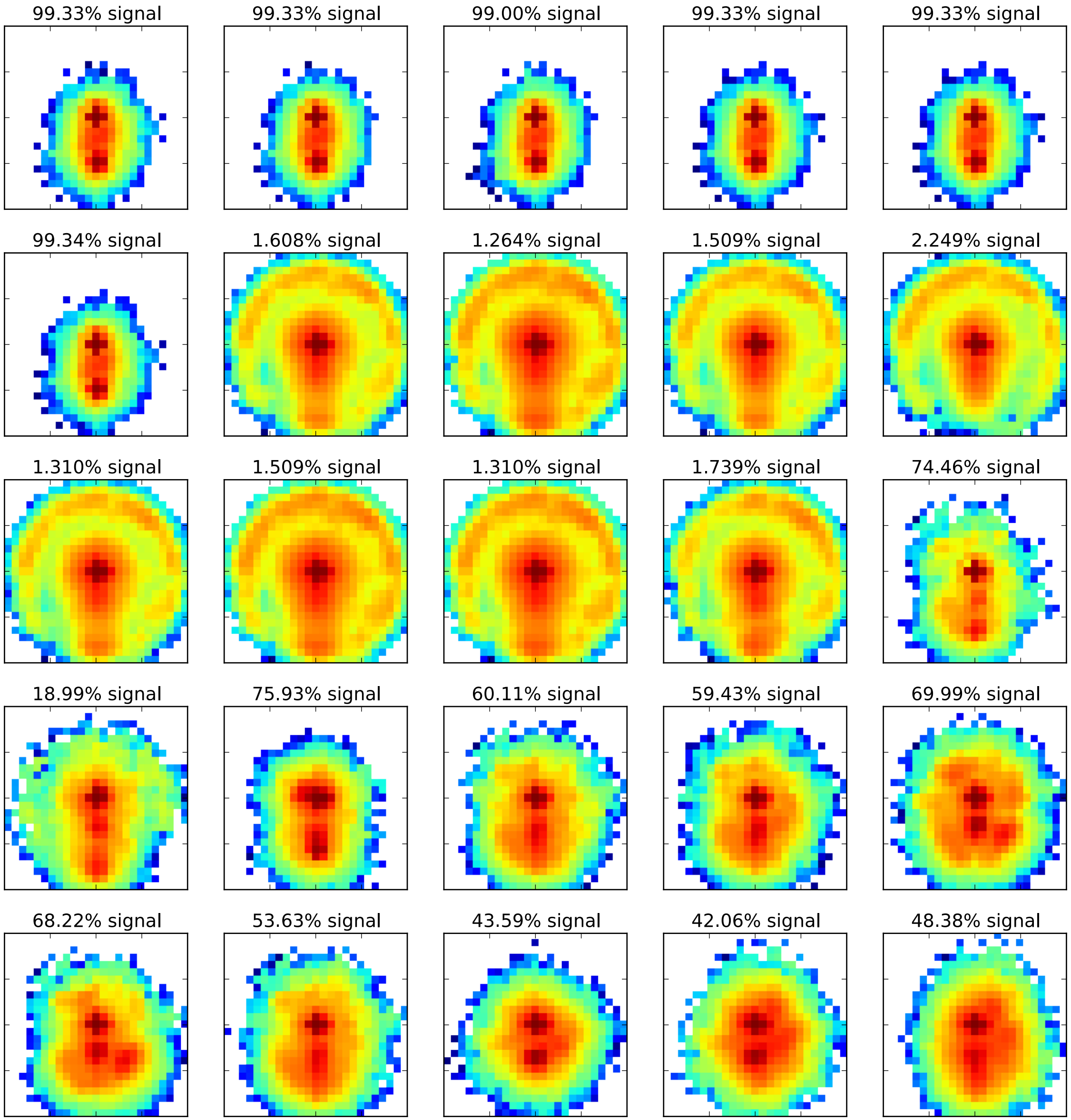
First layer 11x11 convolutional filters



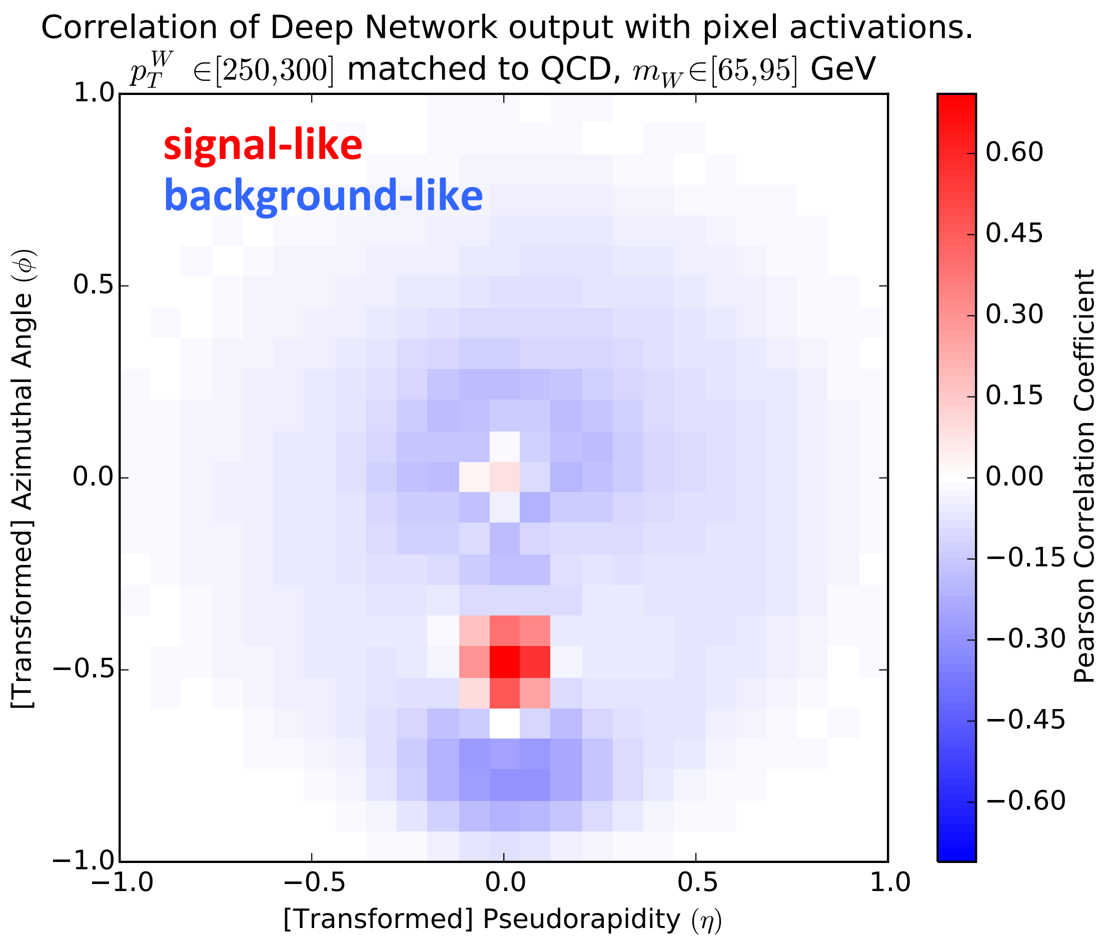
Convolved jet image differences

$$X_{sig} * w - X_{bkg} * w$$

Average Most Activating Jet Images



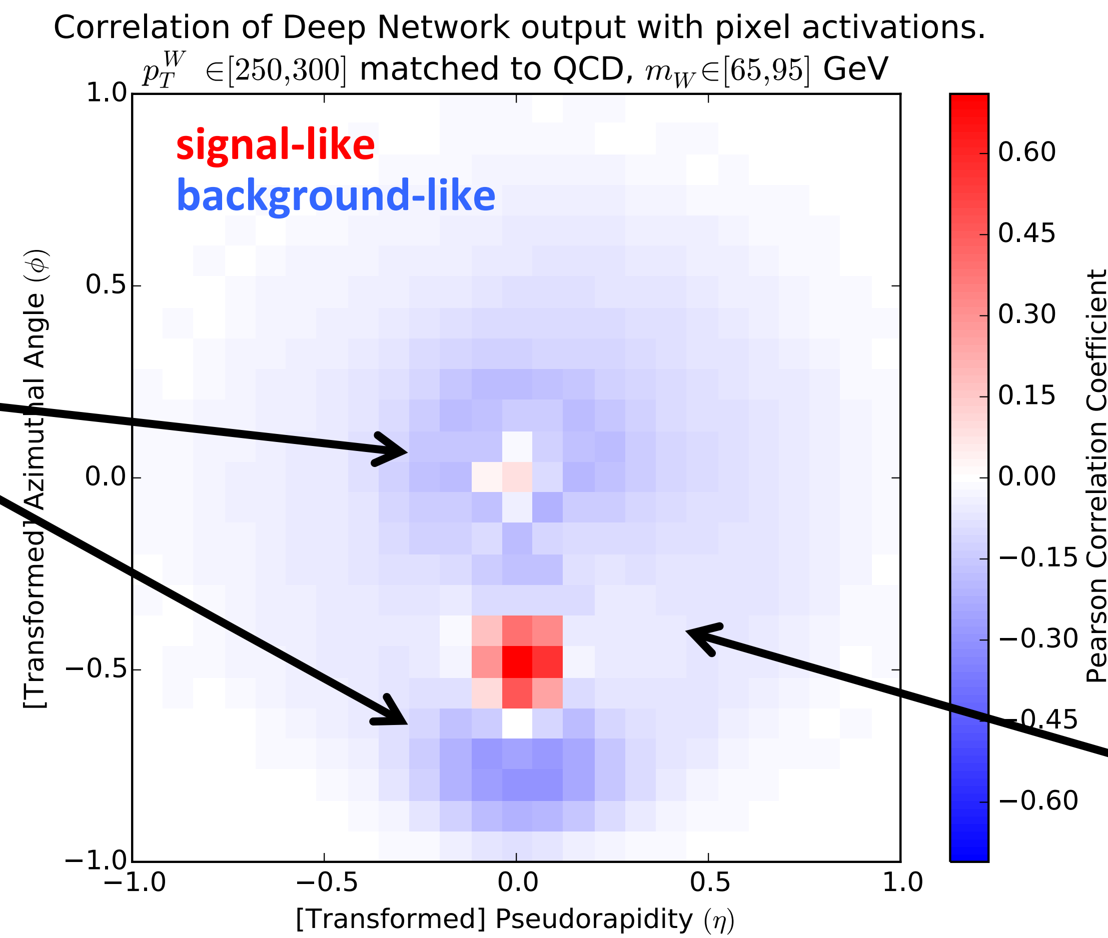
Physics in deep representations



Pearson Correlation Coefficient of the pixels intensity with the network output: <u>how discriminating information is contained within the network</u>

Physics in deep representations

Additional radiation in QCD jets

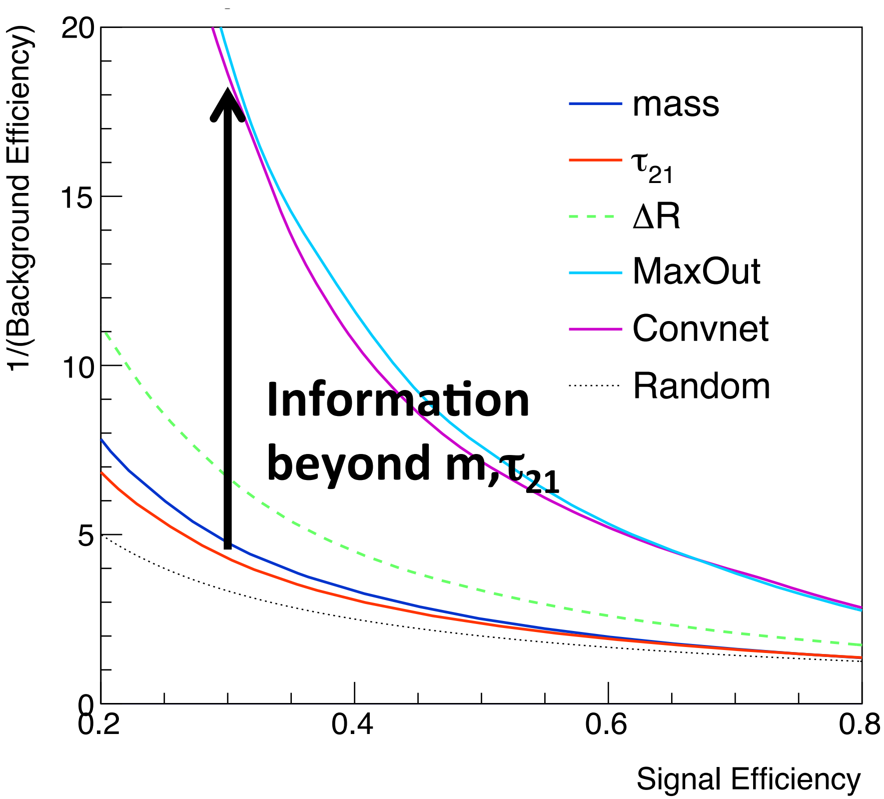


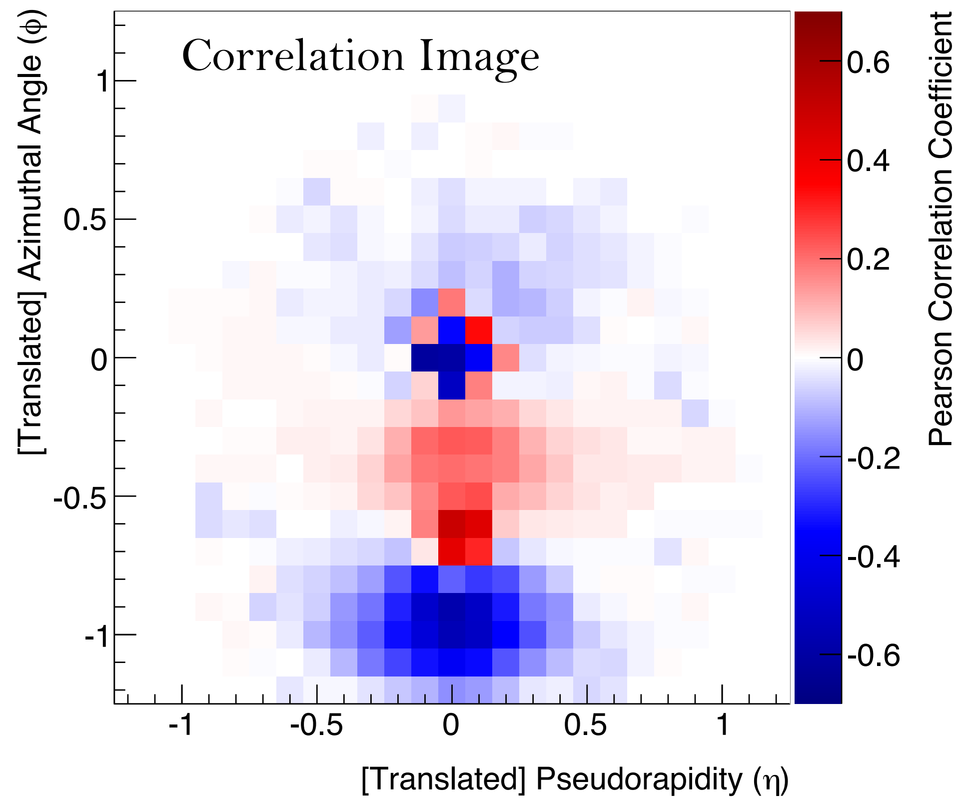
Soft QCD gluon emission

Pearson Correlation Coefficient of the pixels intensity with the network output: <u>how discriminating information is contained within the network</u>

Learning About Learning

Restricted Phase Space: 79 < m < 81 GeV and 0.19 < τ_{21} <0.21





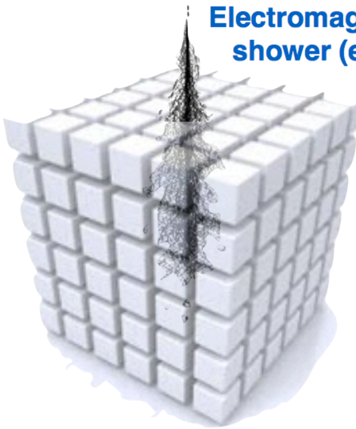
Learning something beyond mass and τ_{21} …

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?

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



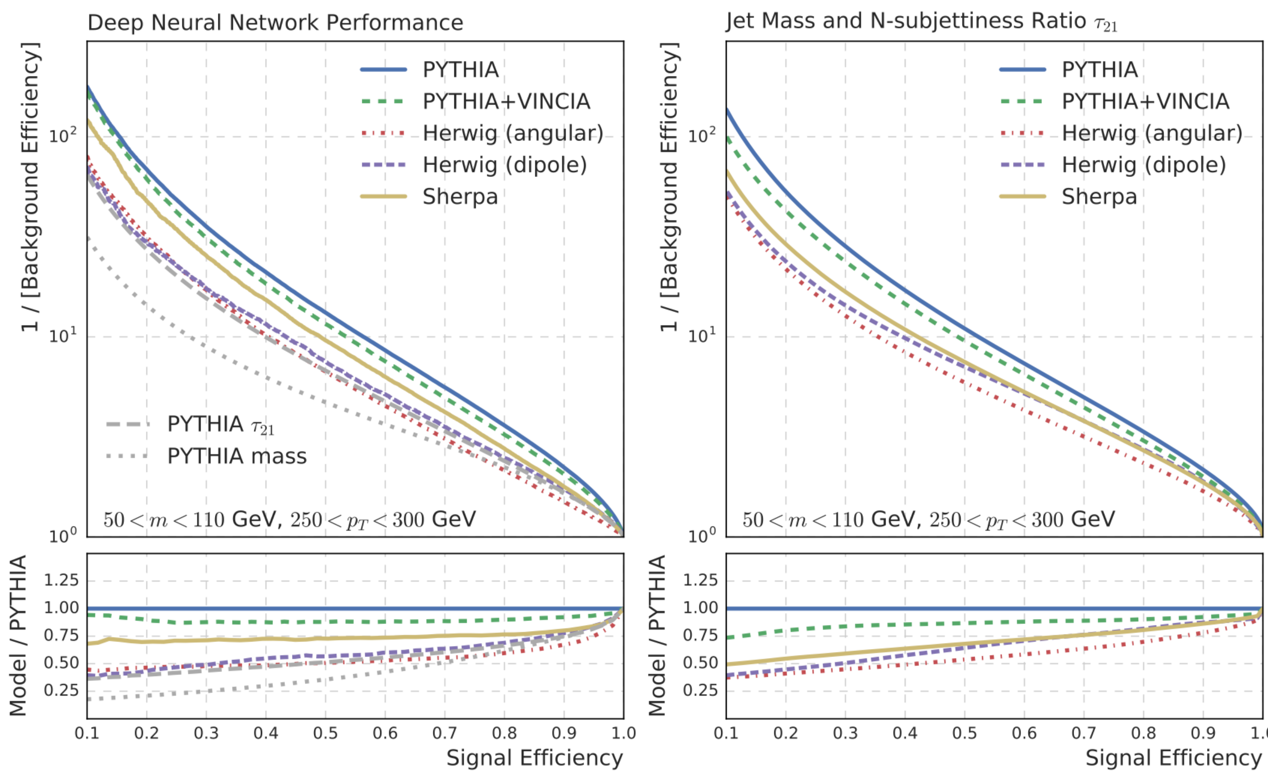
- Sequential learning techniques (not discussed in this talk) may be useful in tasks with variable length data
  - Typical neural networks and BDT's require a fixed input size
  - But not all discrimination tasks in HEP have a fixed size data representation, e.g. jets with variable numbers of constituents,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

- New network training paradigms may help with statistical inference, fast simulations, or reduce 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?



arXiv:1609.00607

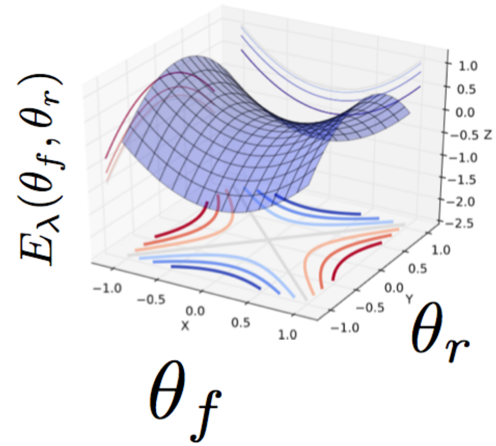
Adversarial Networks

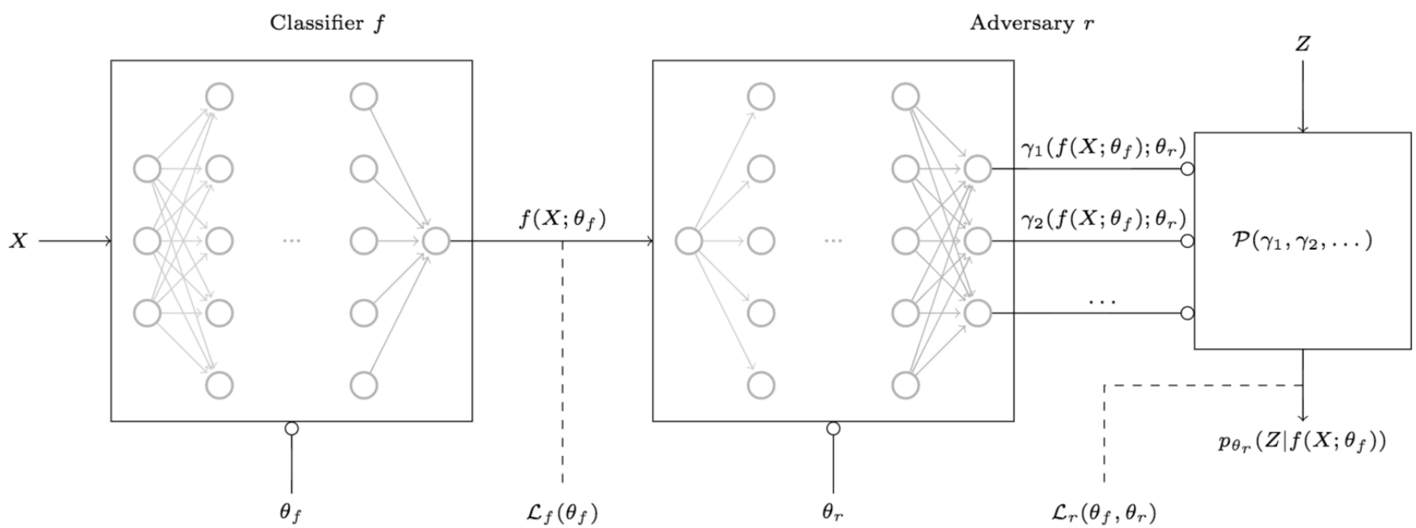
G. Louppe, M. K., K. Cranmer, arXiv:1611.01046

- 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

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

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





Learning to Pivot: Toy Example

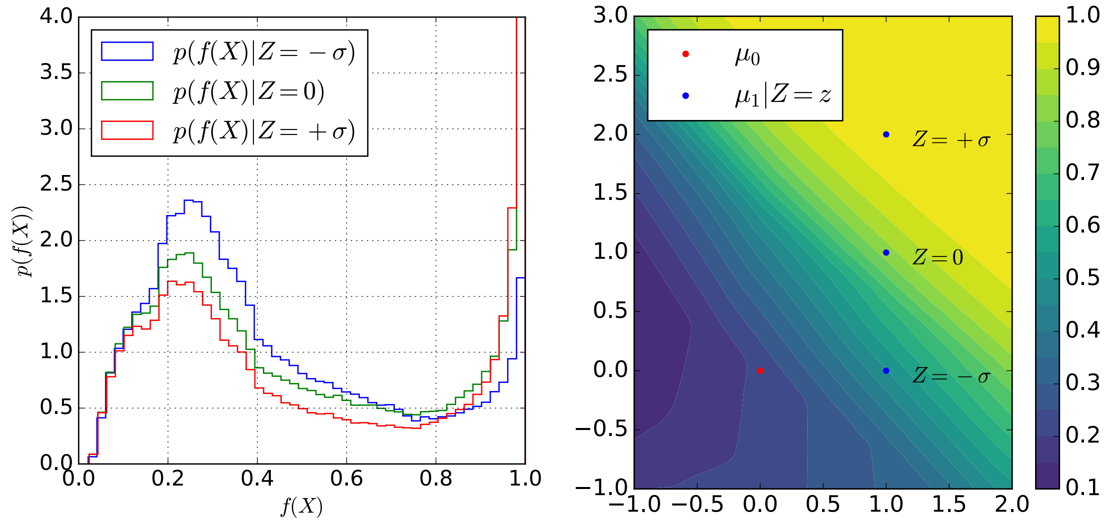
G. Louppe, M. K., K. Cranmer, arXiv:1611.01046

- 2D example

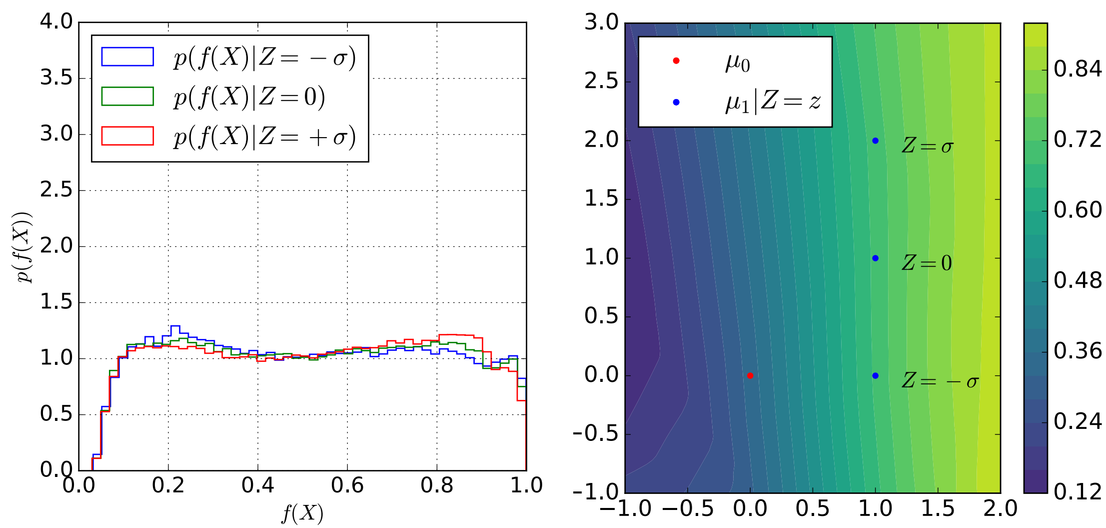
$$x \sim \mathcal{N}\left((0,0), \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}\right) \qquad \text{when } Y = 0,$$

$$x \sim \mathcal{N}\left((1, 1+Z), \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) \qquad \text{when } Y = 1.$$

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



- With adversary (bottom) performance is independent!



Learning to Pivot: Physics Example

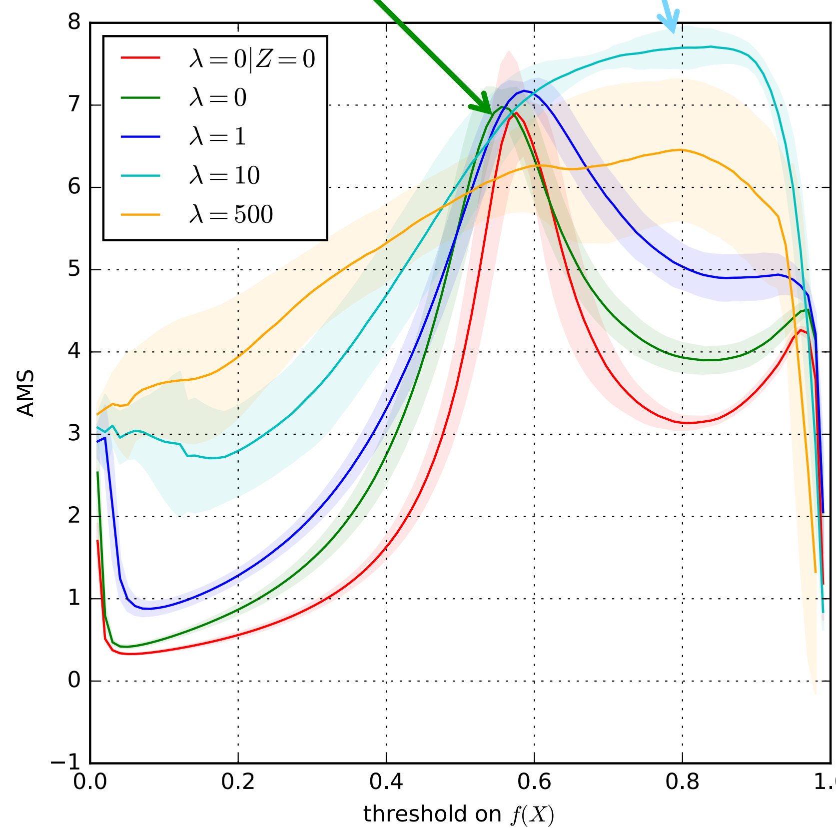
G. Louppe, M. K., K. Cranmer,
arXiv:1611.01046

- 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



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

Backup

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
  - http://scikit-learn.org/stable/
- 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/

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**: $Y \in$ 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**: $Y \in$ Real Numbers

Main focus today on supervised learning in HEP

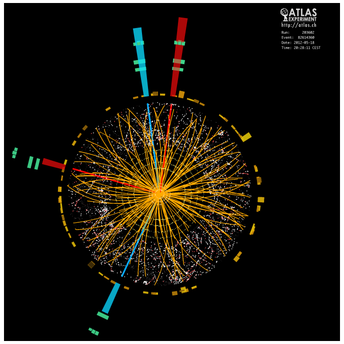
- ~~Unsupervised Learning~~
  - ~~Given some data $D=\{x_i \in X\}$, but no labels, find structure in the data~~
  - ~~Clustering: partition the data into groups~~
  - ~~Dimensionality reduction: find a low dimensional (less complex) representation of the data with a mapping $Z=h(X)$~~

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



Function with adjustable parameters

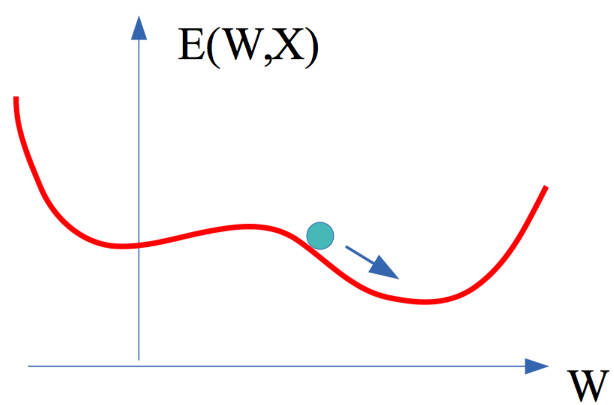
Error/Loss Function

Error

True labels:
Higgs = 1
Bkg = 0

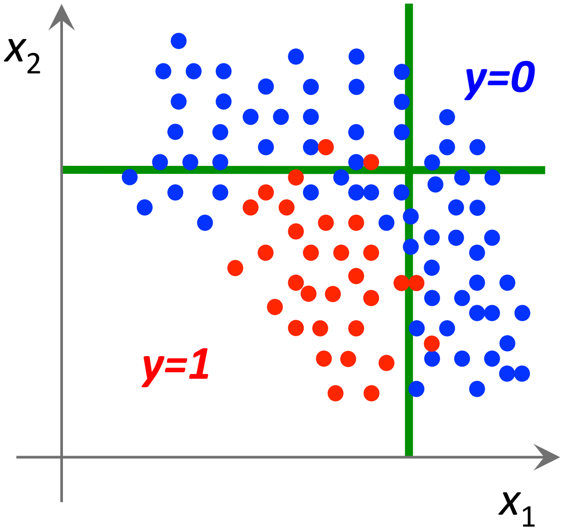
Y. Le Cun

- 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*

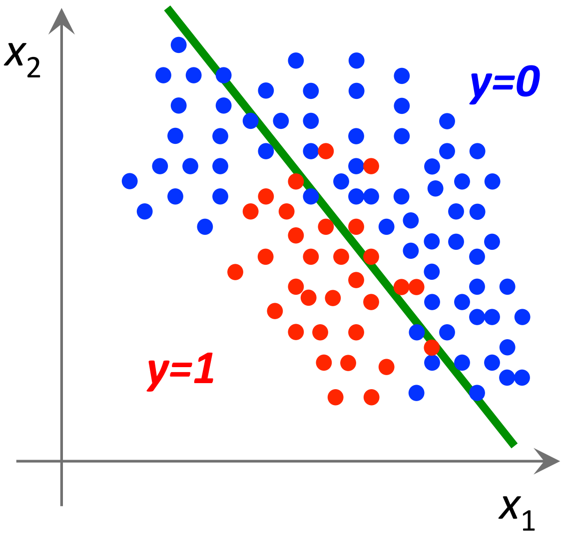


Classification

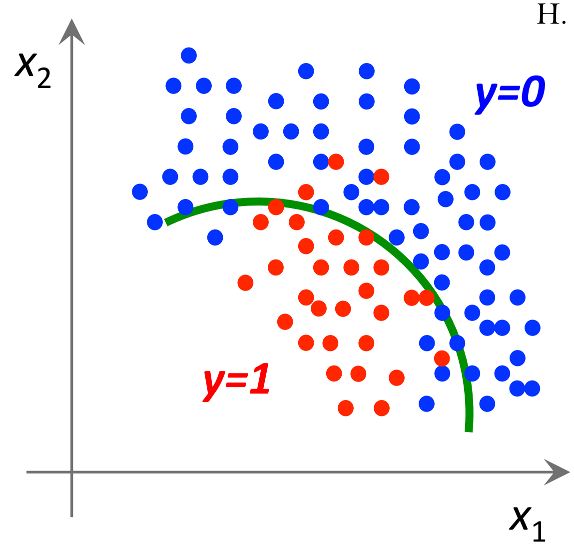
H. Voss



Rectangular cuts



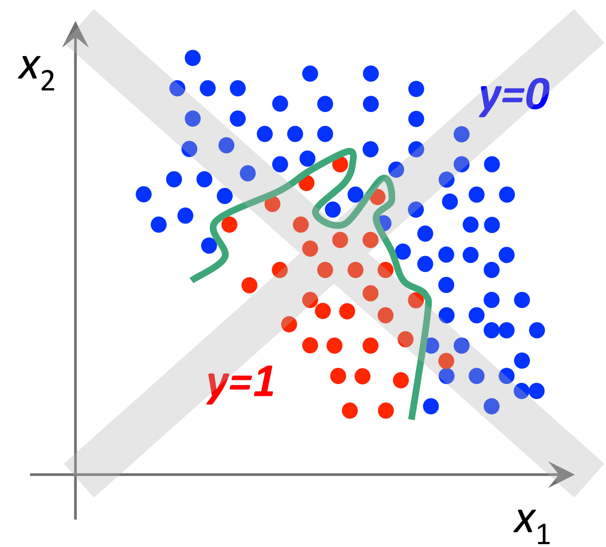
Linear discriminant



Nonlinear discriminant

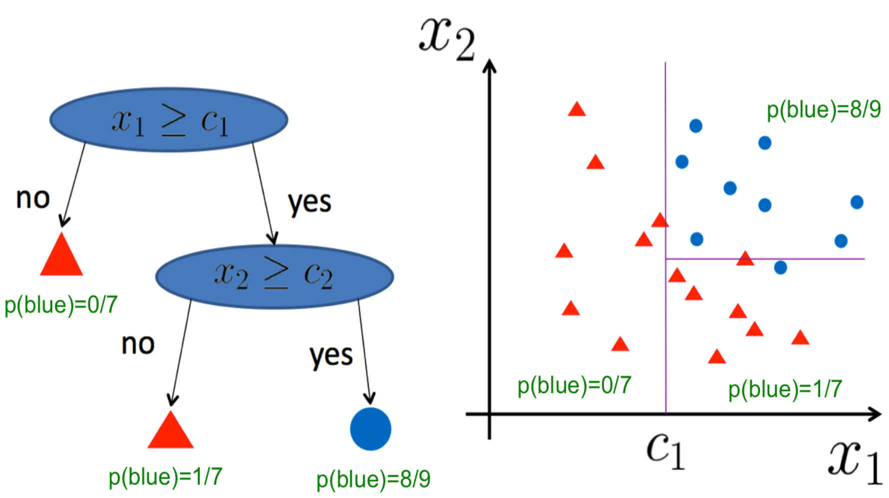
- Learn a function to separate different classes of data

- Avoid over-fitting:
  - Learning too fined details about your training sample that will not generalize to unseen data



Machine Learning in High Energy Physics

- Many recent application 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
    http://jmlr.csail.mit.edu/papers/volume15/delgado14a/delgado14a.pdf
  - *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

Decision Trees

Optimizing a Decision Tree

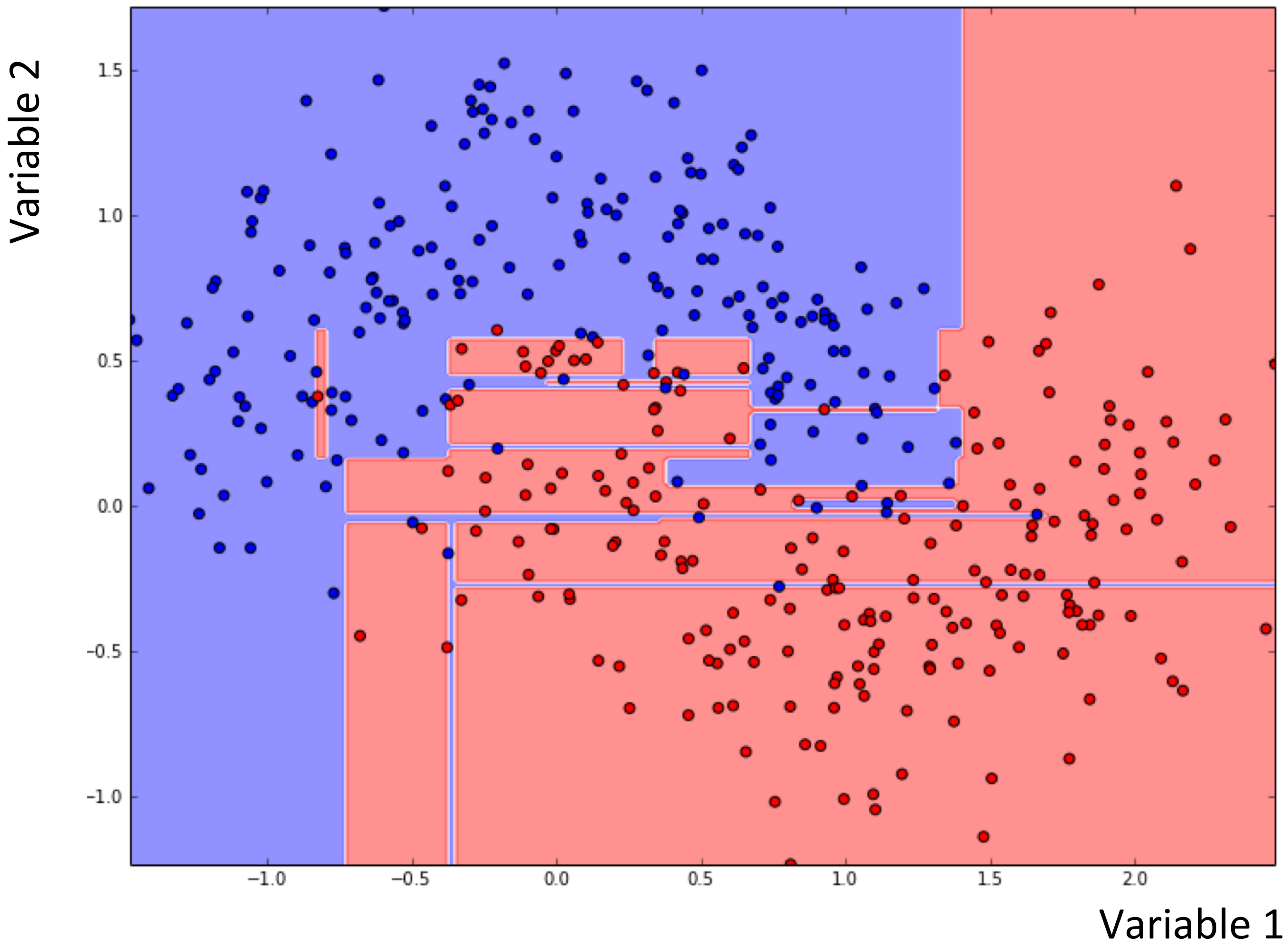
- Building an optimal decision tree is an NP-complete problem
  - Hard to find a global optimization for all splittings at the same time

- 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 = Σ_{leaf} Impurity(leaf)*size(leaf)
  - Typical leaf impurity functions:
  - Gini = p*(1-p)
  - Entropy = -p*log(p) – (1-p)*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): $(1/n_i)\ \Sigma_{i \text{ in leaf}} (y_i - m)^2$
    - 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

Overfitting



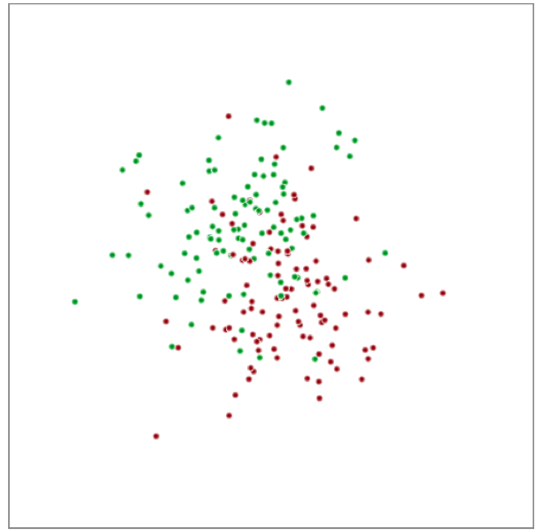
- Single decision trees can quickly overfit
- Especially when increasing the depth of the tree

Ensemble Methods

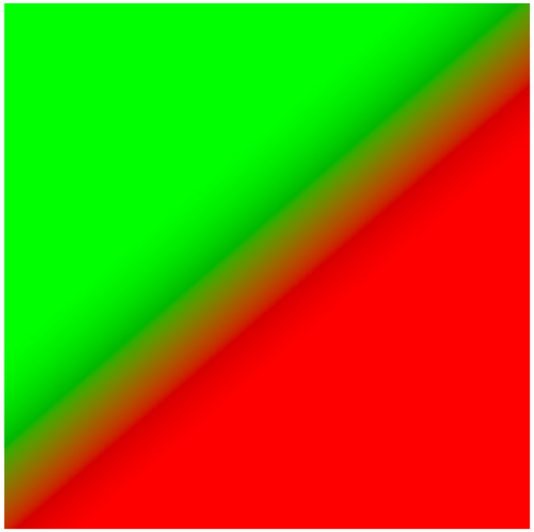
- Combine many decision trees, use the ensemble for prediction
- Averaging: $D(x) = \dfrac{1}{N_{tree}} \sum_{i=1}^{N_{tree}} d_i(x)$
  - **Random Forest**, averaging combined with:
    - **Bagging:** Only use a subset of events for each tree training
    - **Feature subsets**: Only use a subset of features for each tree
- Boosting (weighted voting): $D(x) = \sum_{i=1}^{N_{tree}} \alpha_i d_i(x)$
  - 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

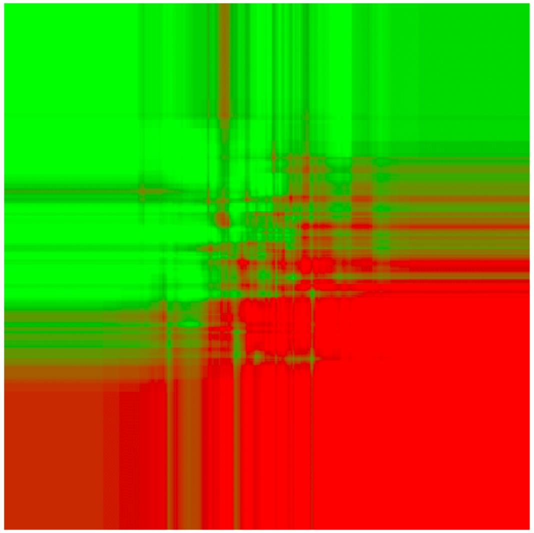
- Ensembles of trees tend to work very well
  - Relatively simple
  - Relatively easy to train
  - Tend not to overfit (especially random forests)
  - Work with different feature types: continuous, categorical, etc.



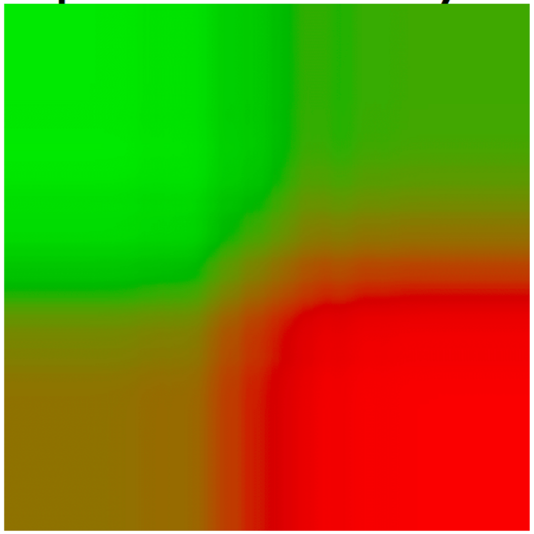
data



optimal boundary

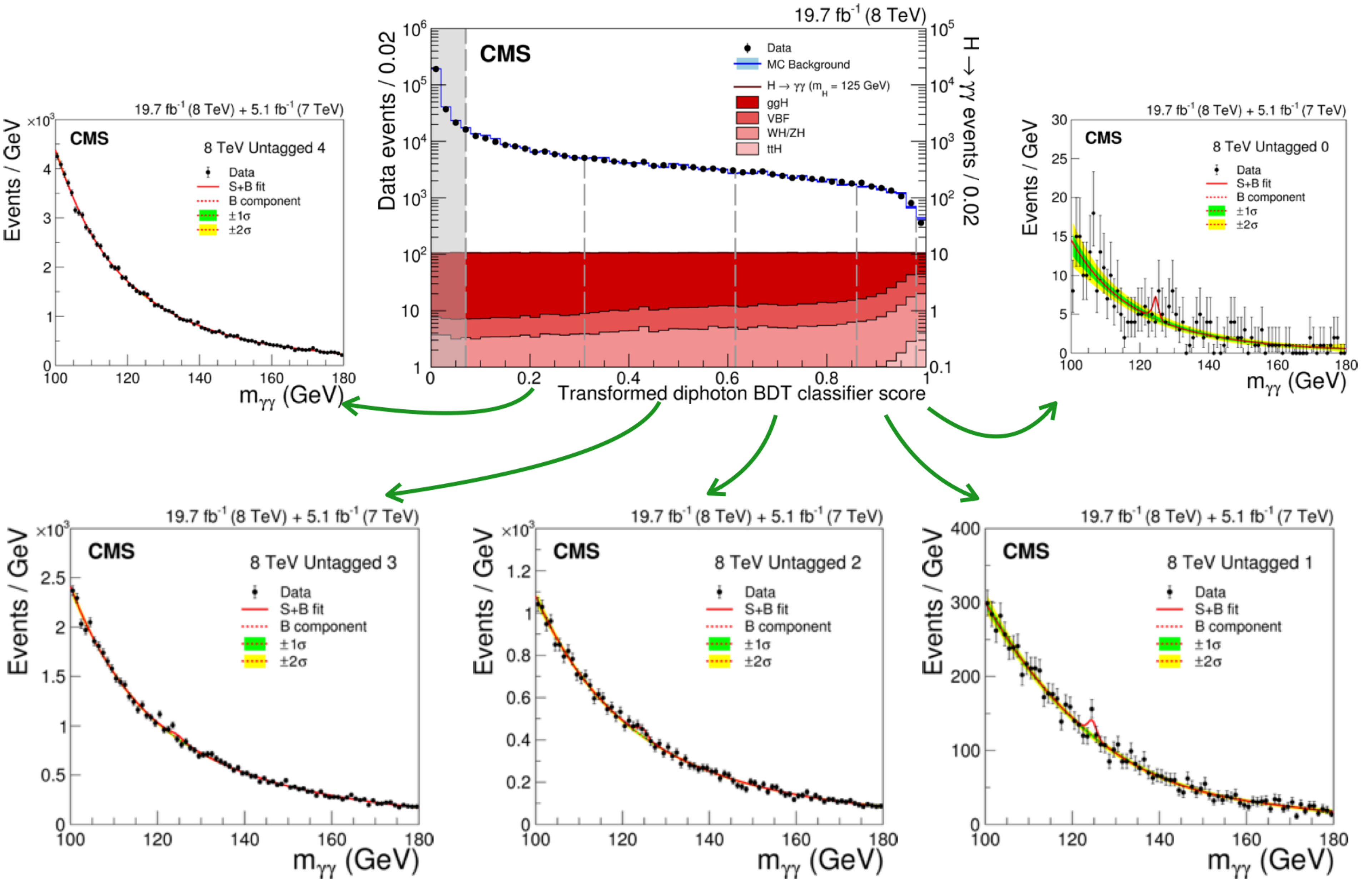


50 trees



2000 trees

CMS h→γγ (8 TeV)

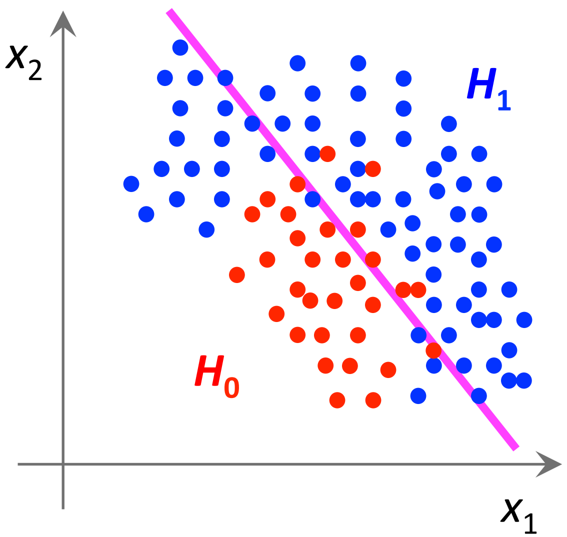


19.7 fb-1 (8 TeV)
CMS
Data
MC Background
H → γγ (mH = 125 GeV)
ggH
VBF
WH/ZH
ttH
Data events / 0.02
H → γγ events / 0.02
Transformed diphoton BDT classifier score
19.7 fb-1 (8 TeV) + 5.1 fb-1 (7 TeV)
8 TeV Untagged 4
8 TeV Untagged 3
8 TeV Untagged 2
8 TeV Untagged 1
8 TeV Untagged 0
S+B fit
B component
±1σ
±2σ
Events / GeV
mγγ (GeV)

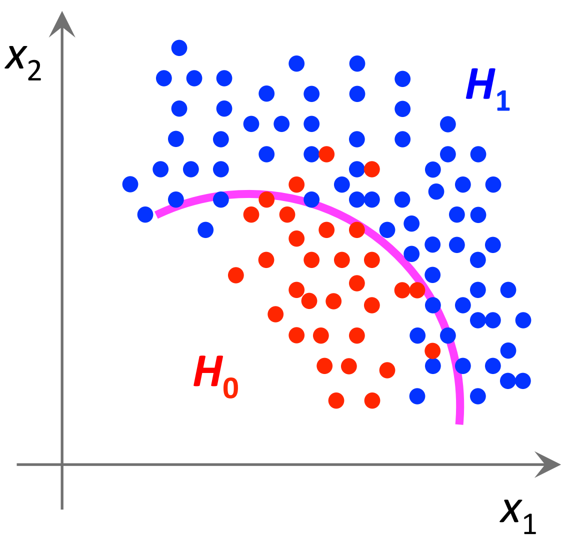
Neural Networks

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: $f(X) = W_n(W_{n-1} (\dots W_1 X)) = UX$, where $U = \Pi_i W_i$
- Linear functions can only correctly classify linearly separable data!
- For complex datasets, need nonlinearities to properly learn data structure



Linear Classifier



Non-linear Classifier

Neural Networks and Local Minima

- Large NN's difficult to train…trapping in local minimum?

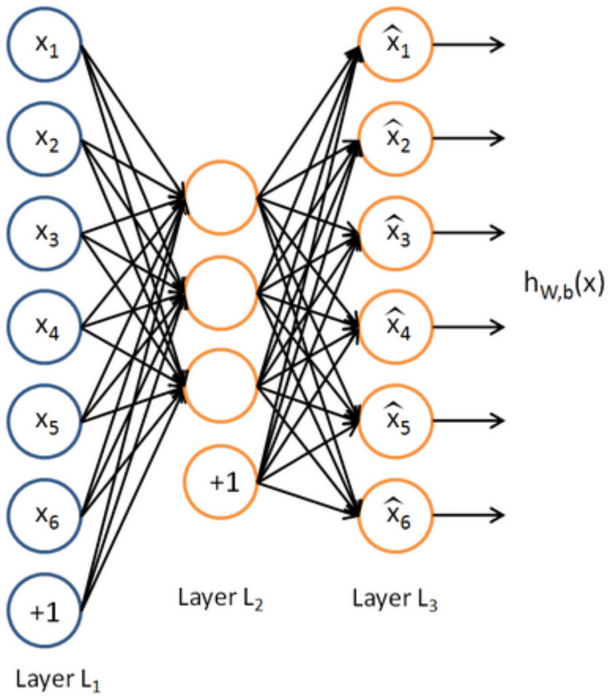
- Not in large neural networks *https://arxiv.org/abs/1412.0233*
  – Most local minima equivalent, and resonable
  – Global minima may represent overtraining
  – Most bad (high error) critical points are saddle points (different than small NN's)

Weight Initializations and Training Procedures

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

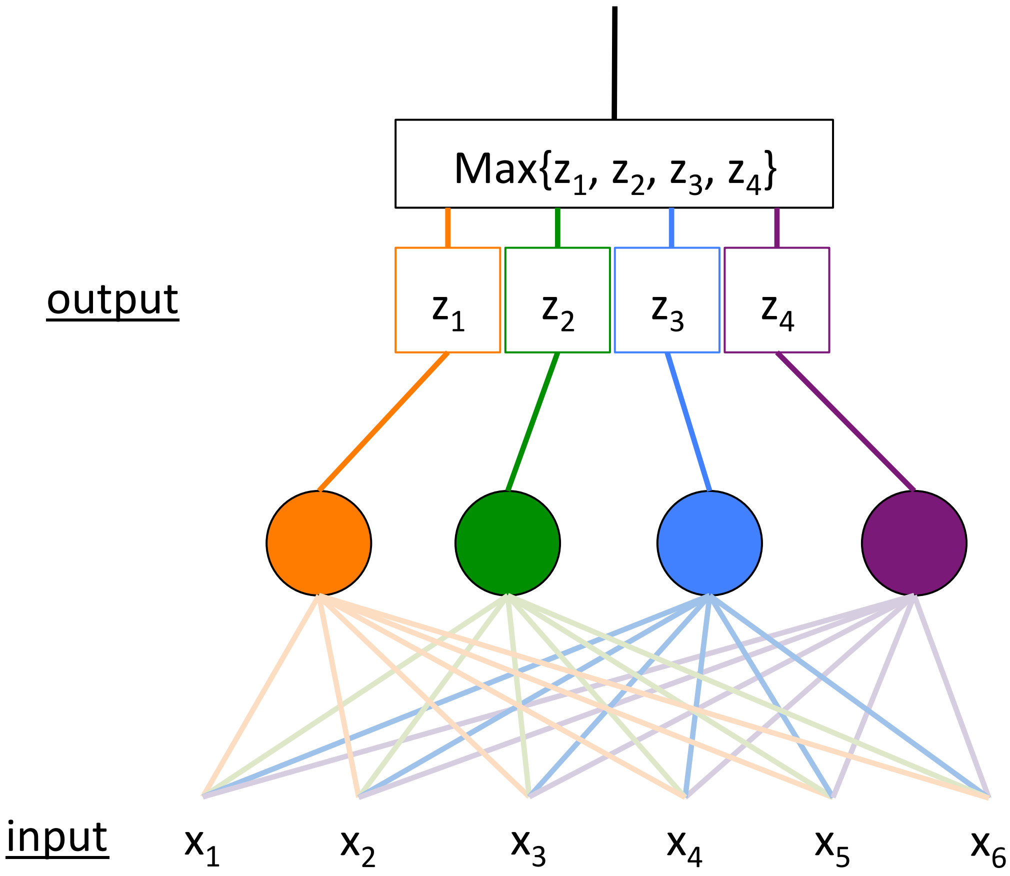
- Now initialize such that node outputs are normally distributed

- 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…



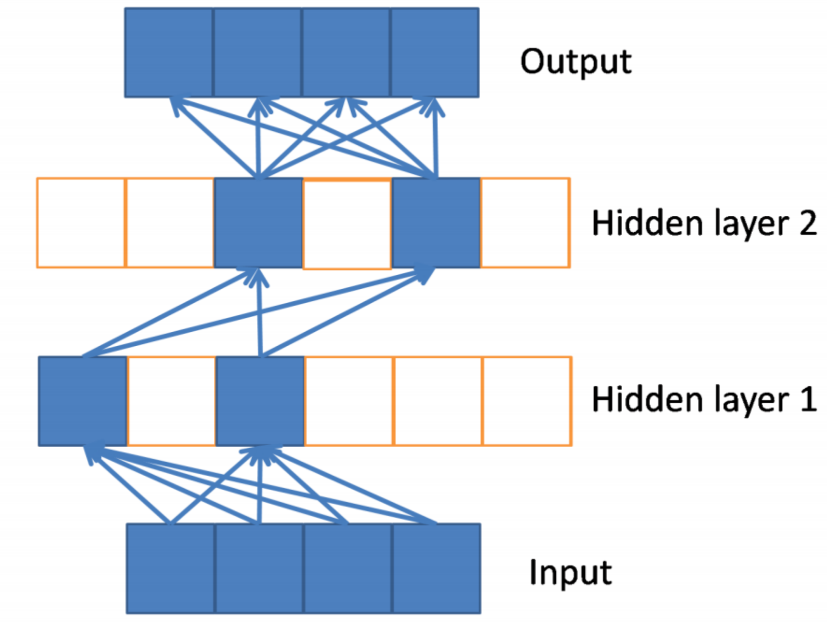
- Adversarial training, invented 2014
  - Will potential HEP applications later

MaxOut



Hidden layer
Different Colors represent different weights W*x

ReLU Networks



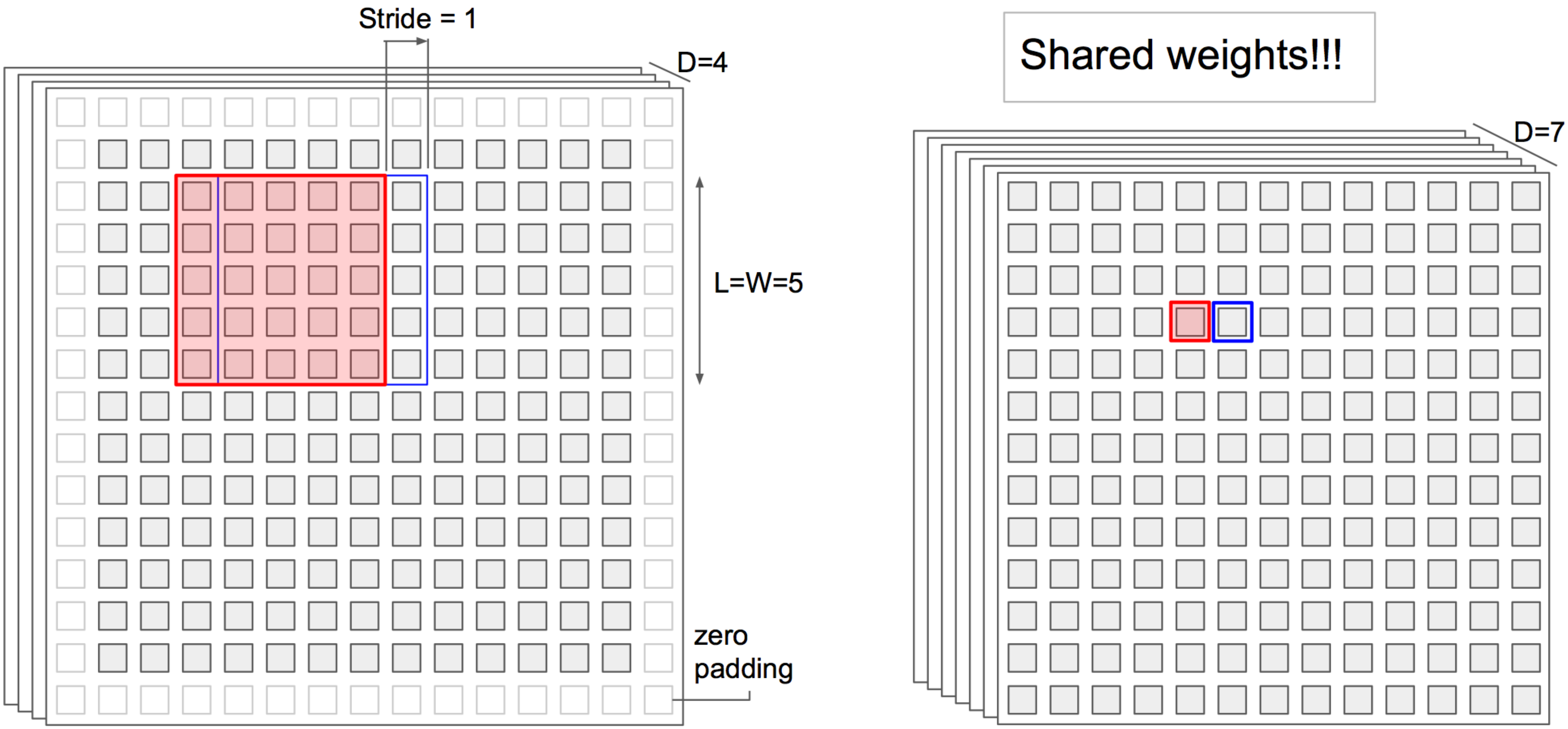
http://www.jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf

- 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

Convolutions in 2D

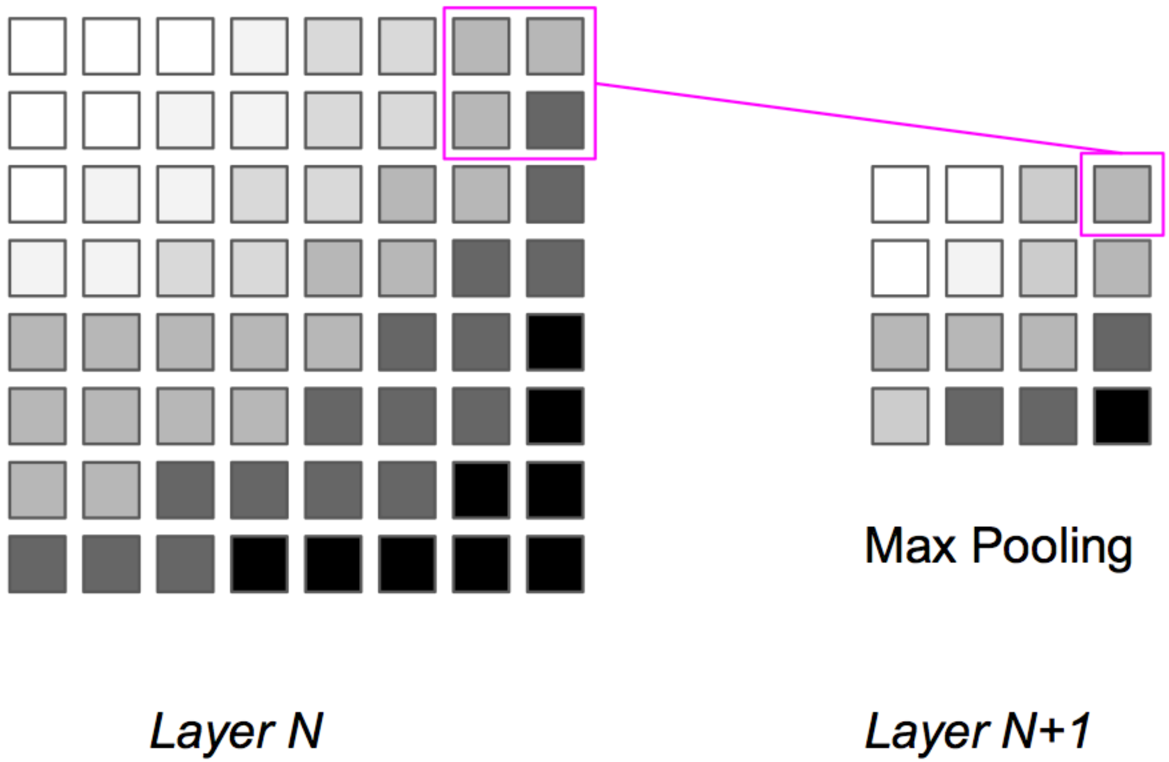


Input image

Convolved image

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

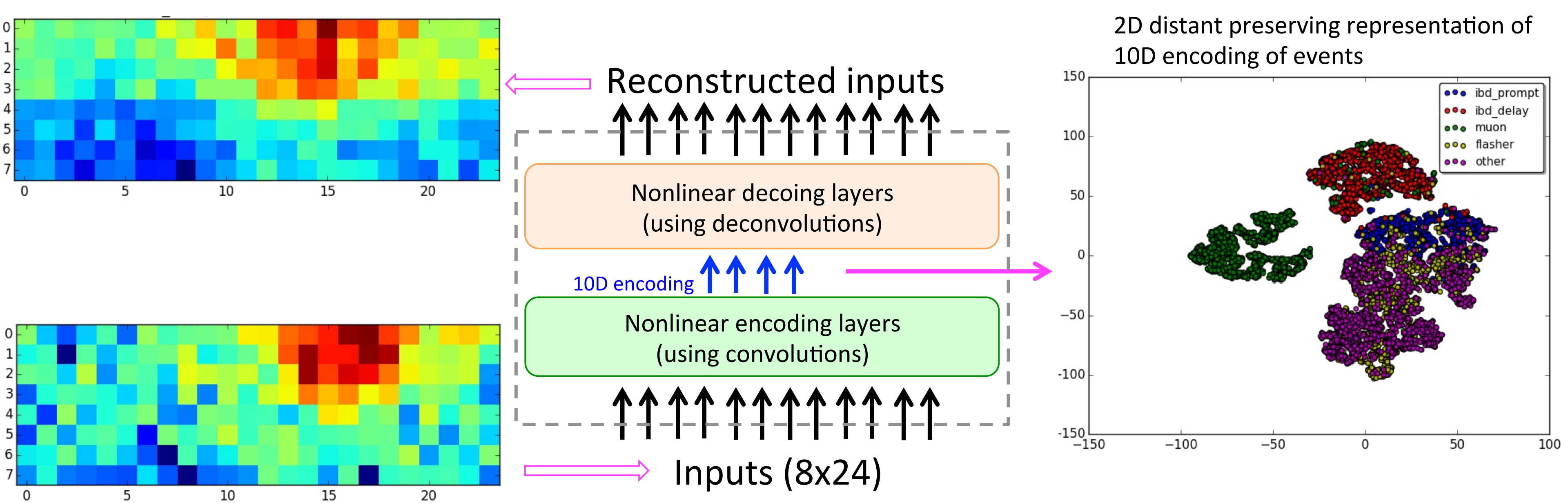
Max Pooling



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

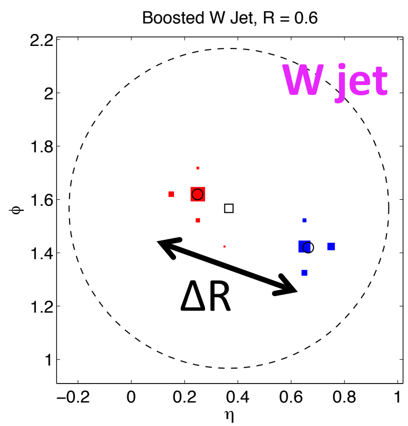
Daya Bay Neutrino Experiment

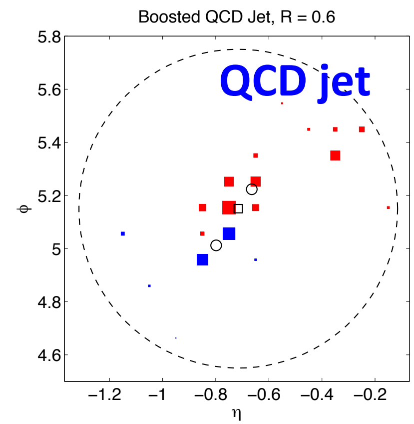
- Aim to reconstruct inverse β-decay interactions from scintillation light recorded in 8x24 PMT's
- Study discrimination power using CNN's
  - Supervised learning → observed excellent performance (97% accuracy)
  - Unsupervised learning: ML learns itself what is interesting!



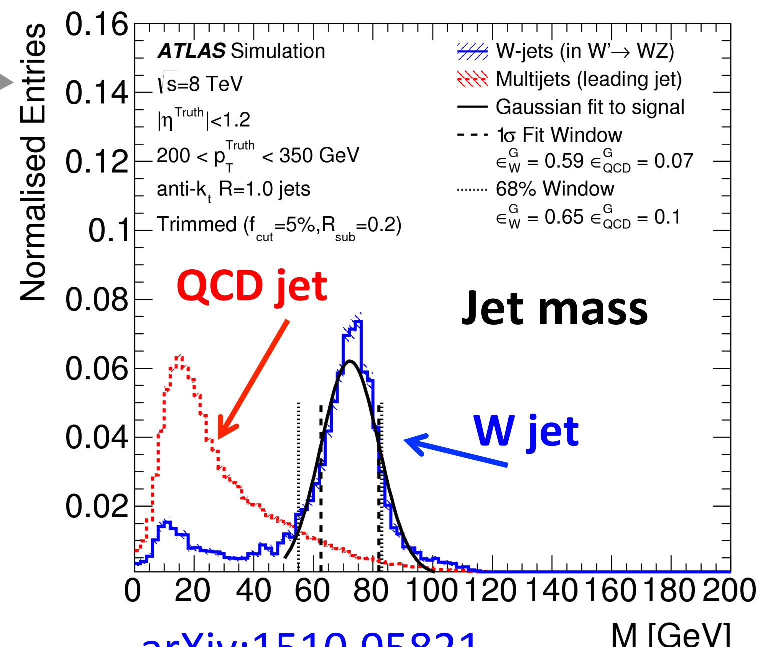
Jet tagging using jet substructure

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





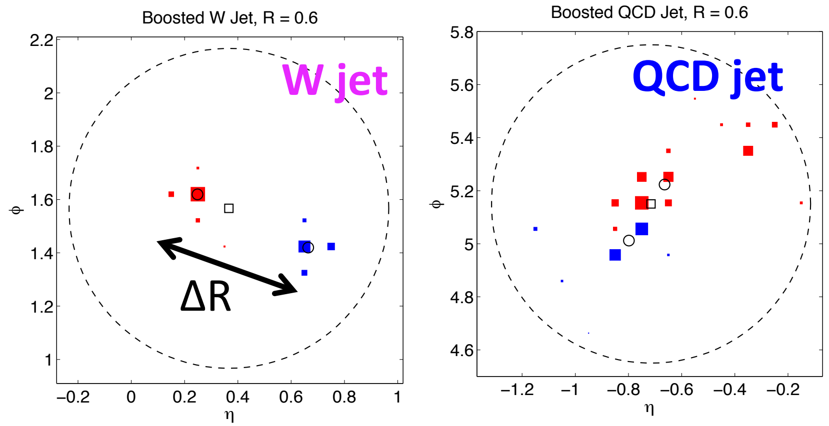
- Typical physics inspired variables exploit differences in:
  - **Jet mass**
  - **N-prong structure**:
    - 1-prong (QCD)
    - 2-prong (W,Z,H)
    - 3-prong (top)
  - **Radiation pattern:**
    - Soft gluon emission
    - Color flow



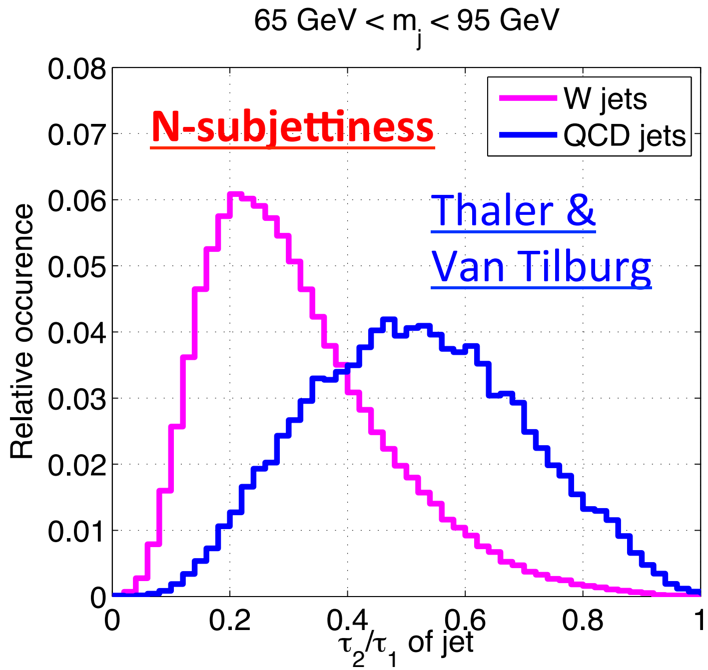
arXiv:1510.05821

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

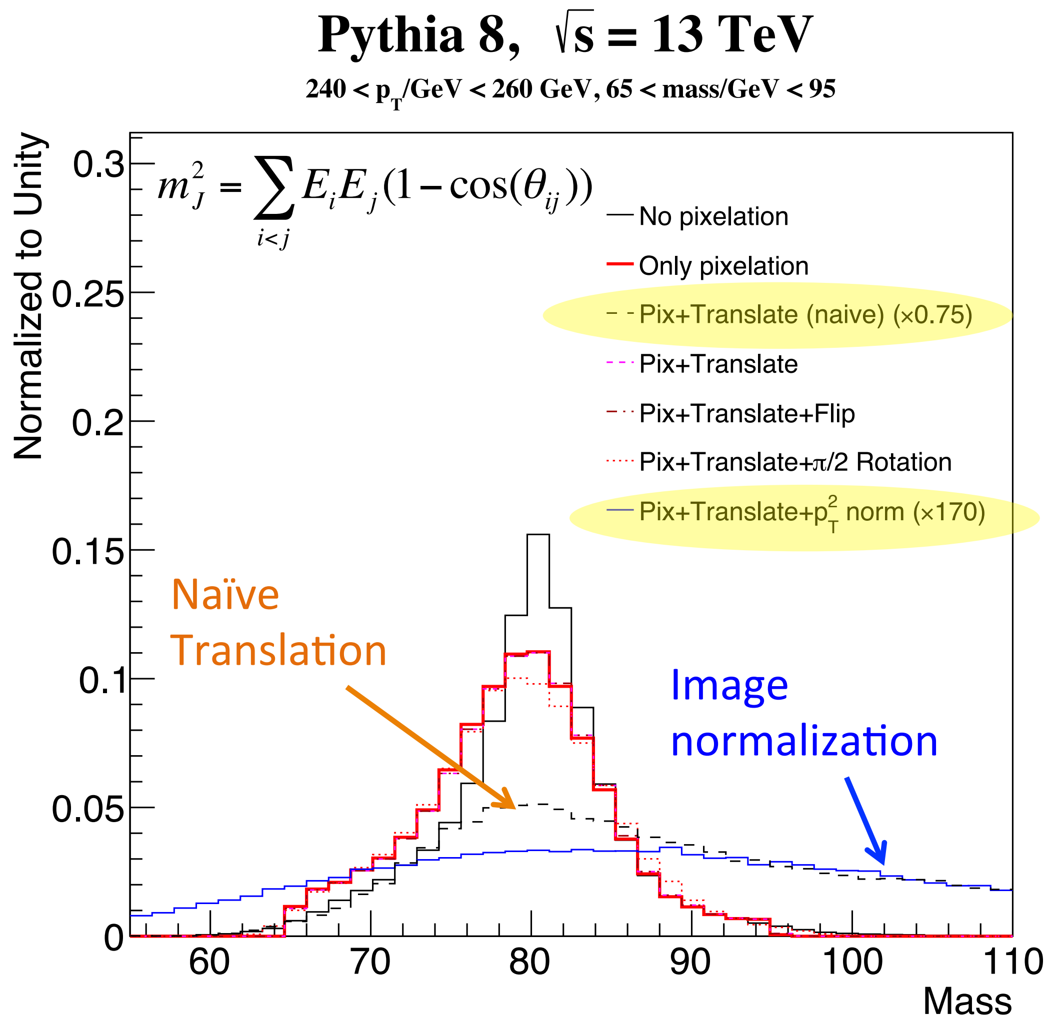


$$\tau_N = \frac{1}{d_0} \sum p_{T,k} \min\{\Delta R_{k,axis-1}, \dots, \Delta R_{k,axis-n}\}$$

Pre-processing and space-time symmetries

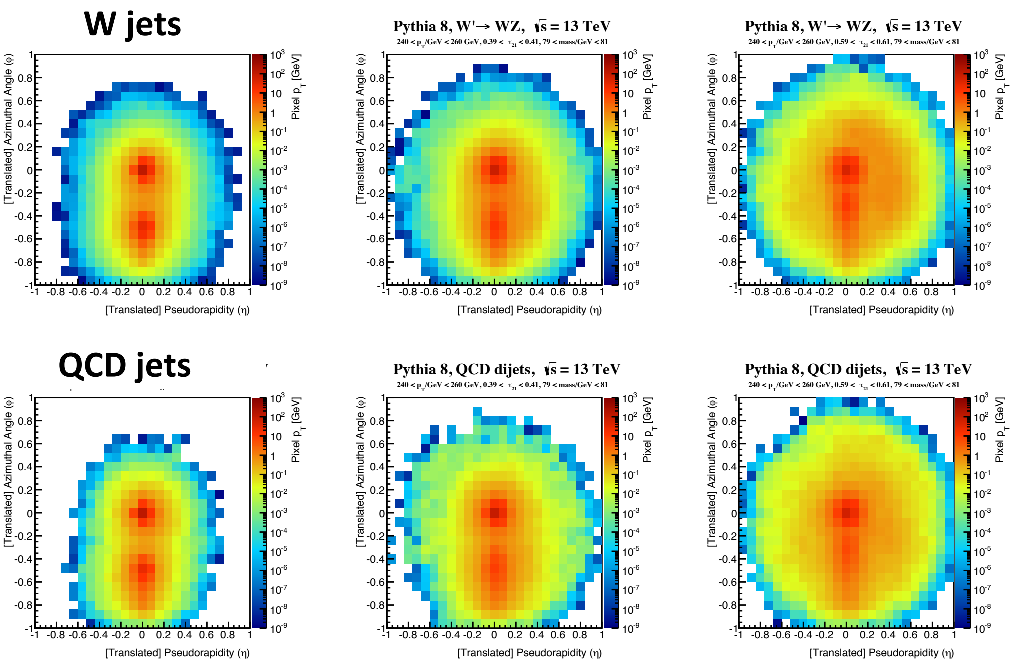
Pre-processing steps may not be Lorentz Invariant

- Translations in η 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



Restricted phase space

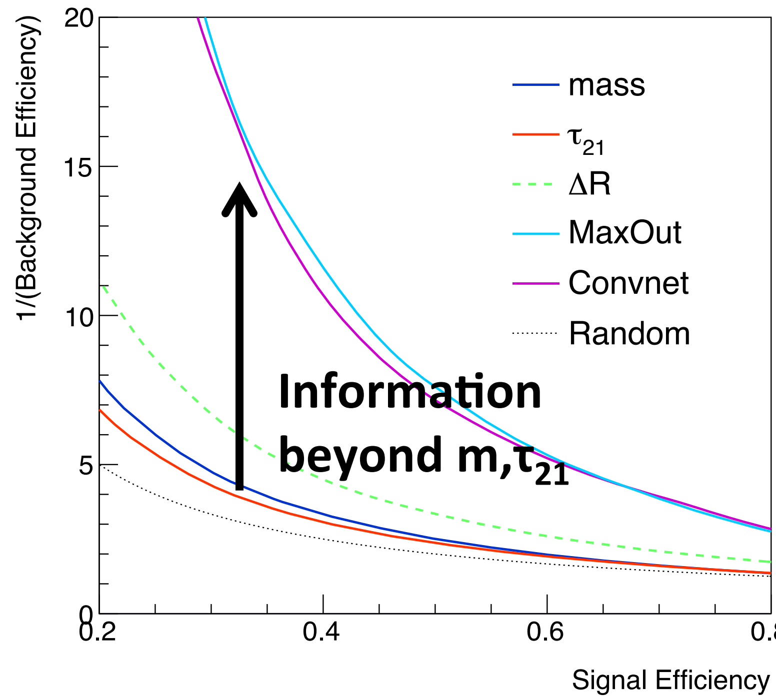
2-prong τ_{21} 1-prong



[0.19, 0.21] [0.39, 0.41] [0.59, 0.61]

79 < m < 81 GeV

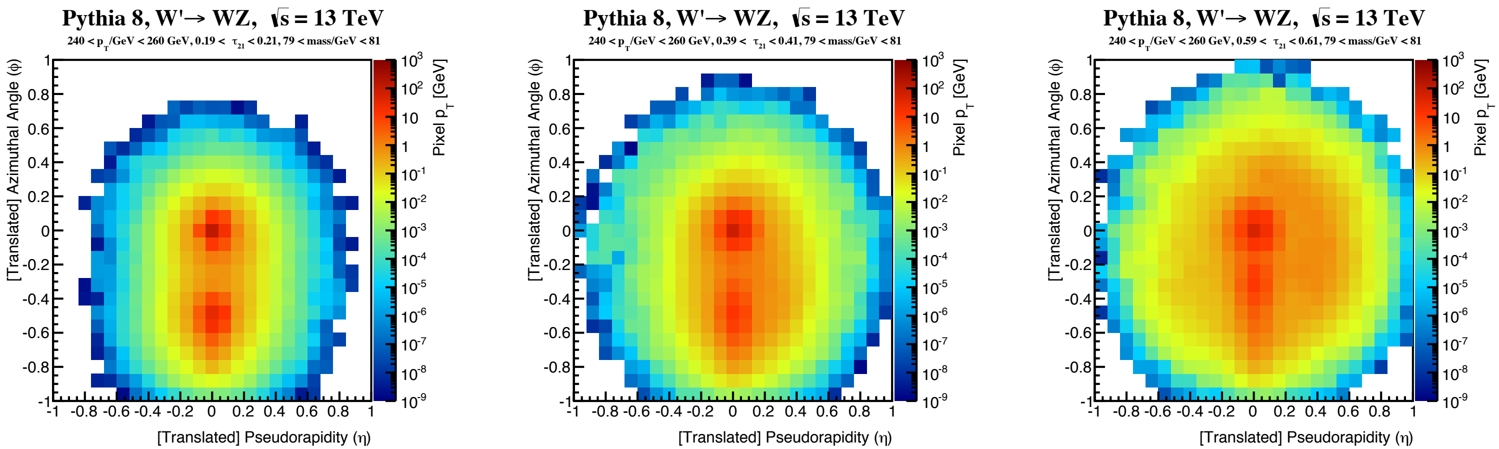
$0.19 < \tau_{21} < 0.21$



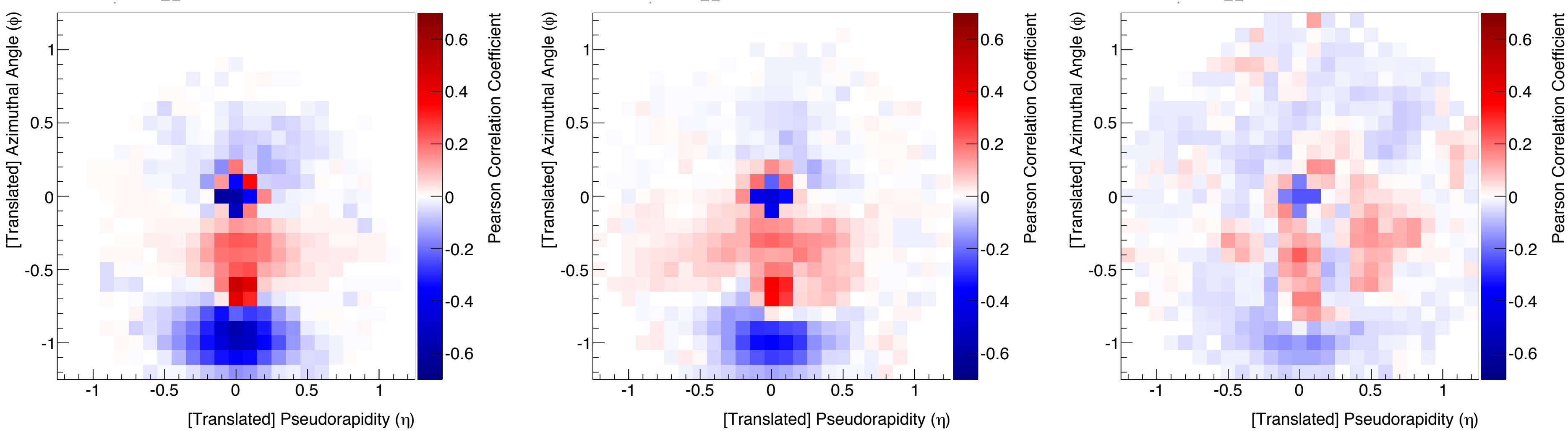
Restrict the phase space in very small mass and τ_{21} bins:

Improvement in discrimination from new, unique, information learned by the network

Deep correlation jet images

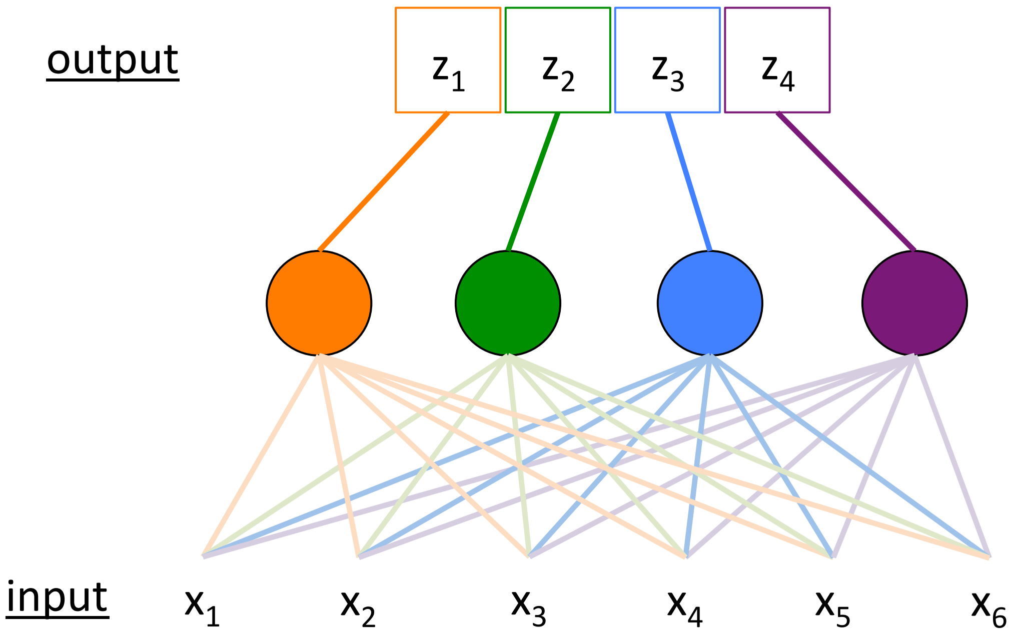


0.19 $<\tau_{21}<$ 0.21 0.39 $<\tau_{21}<$ 0.41 0.59 $<\tau_{21}<$ 0.61



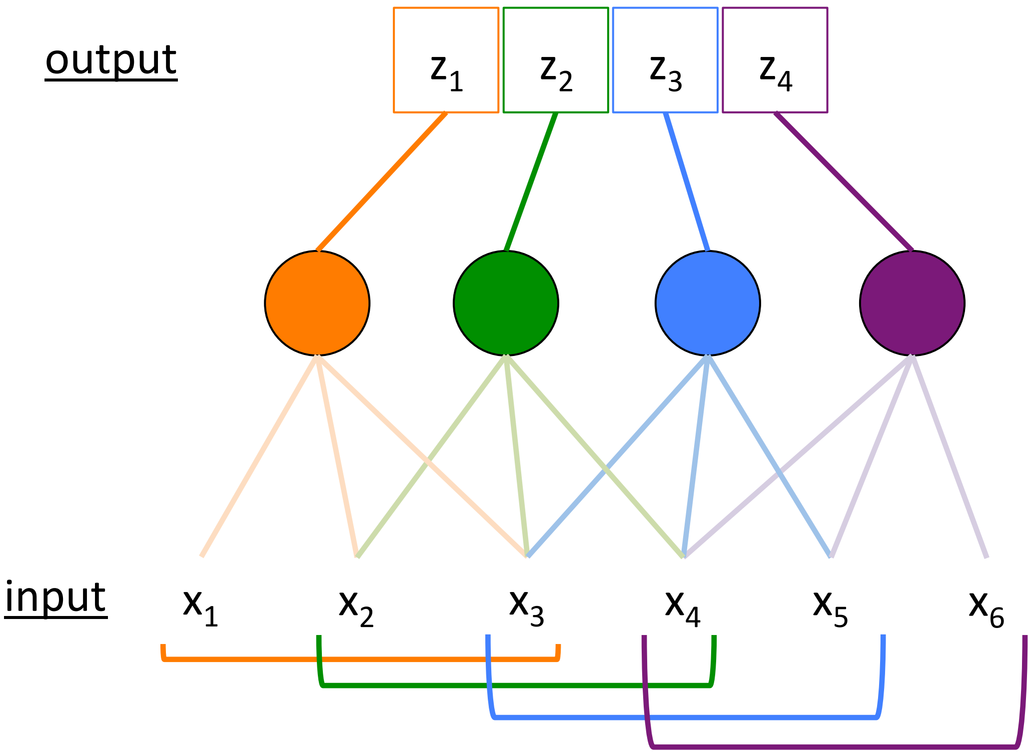
Spatial information indicative of radiation pattern for W and QCD: where in the image the network is looking for discriminating features

Typical Neural Network Hidden Layer



Hidden layer
Different Colors represent
different weights W*x

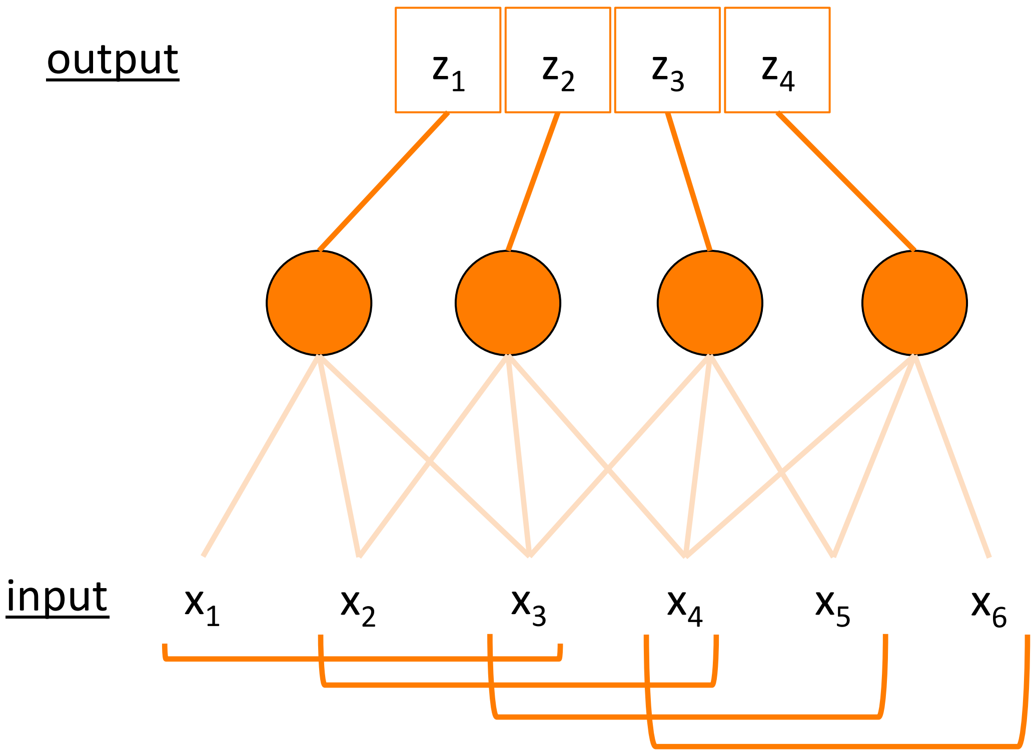
Local Connectivity



Hidden layer
Different Colors represent different weights W*x

Local connectivity: each neuron has a small "field of view" of a few inputs

Shared Weights → Convolutions



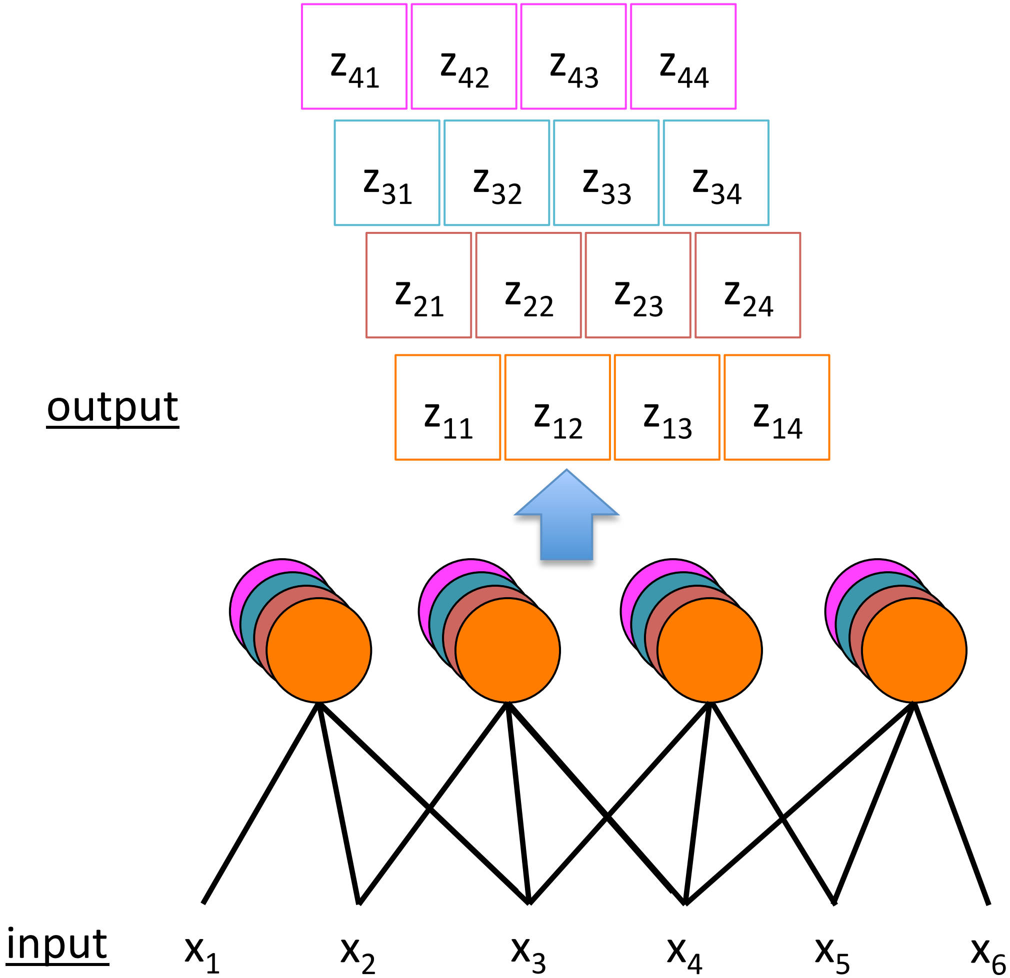
<u>Hidden layer</u>

Different Colors represent different weights W*x

Shared weights: each neuron uses the same weights…

Effect → the neuron is scanned over different fields of view → **Convolution**

Convolutional Layer



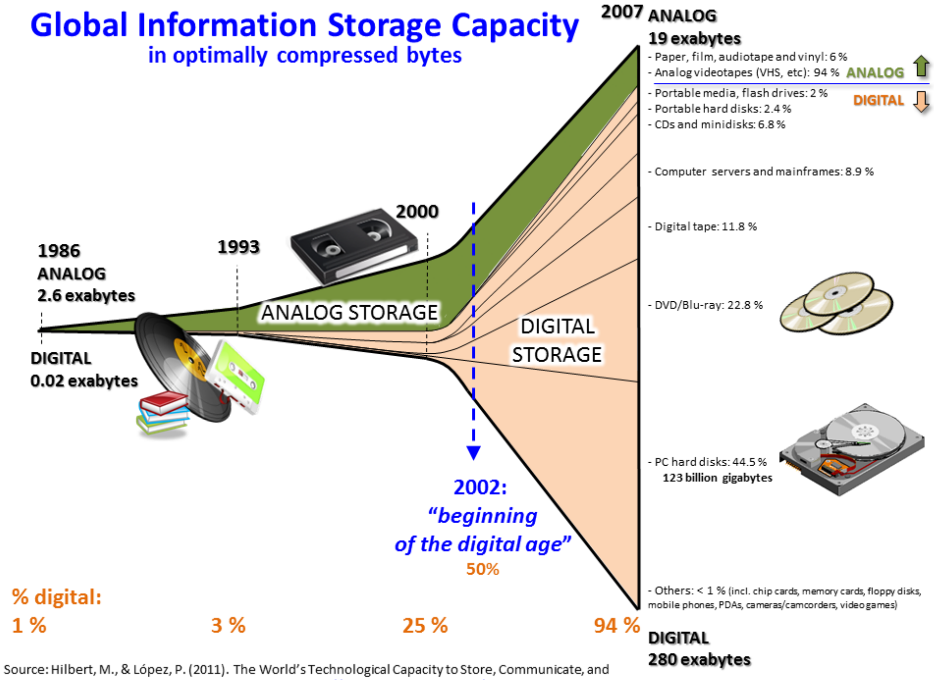
<u>Hidden layer</u>
Different Colors represent different weights W*x

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

Why did it take so long to train DNN's?



Source: Hilbert, M., & López, P. (2011). The World's Technological Capacity to Store, Communicate, and Compute Information. *Science*, 332(6025), 60 –65. http://www.martinhilbert.net/WorldInfoCapacity.html

- 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…

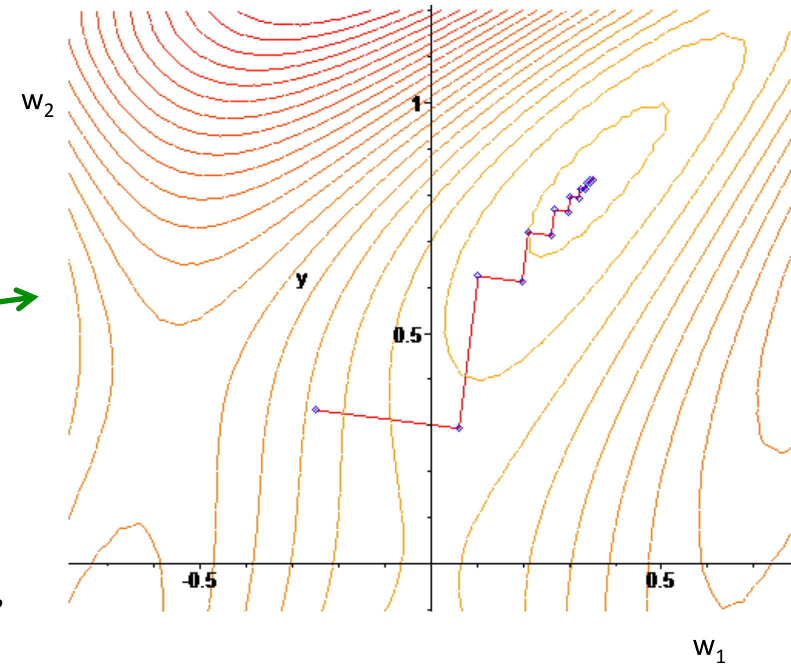
Training Improvements

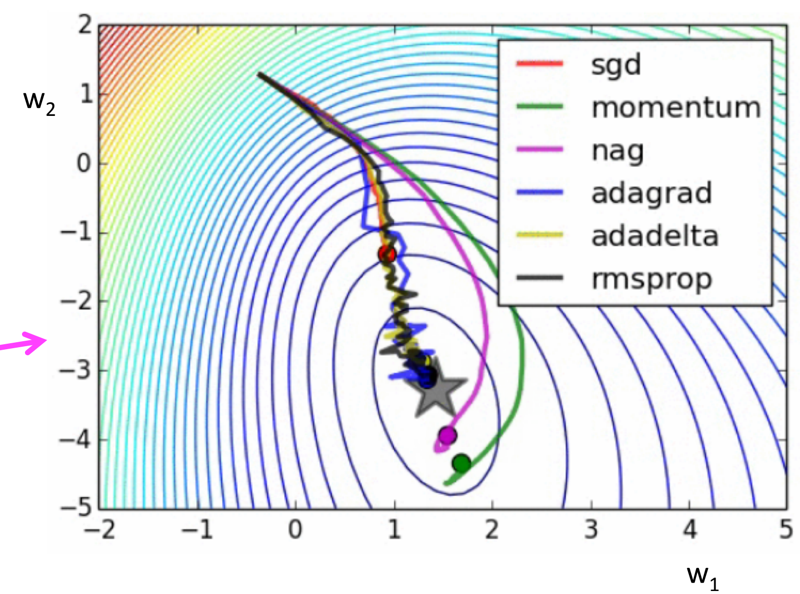
- Gradient descent is computationally costly (since we compute gradient over full training set)

- 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

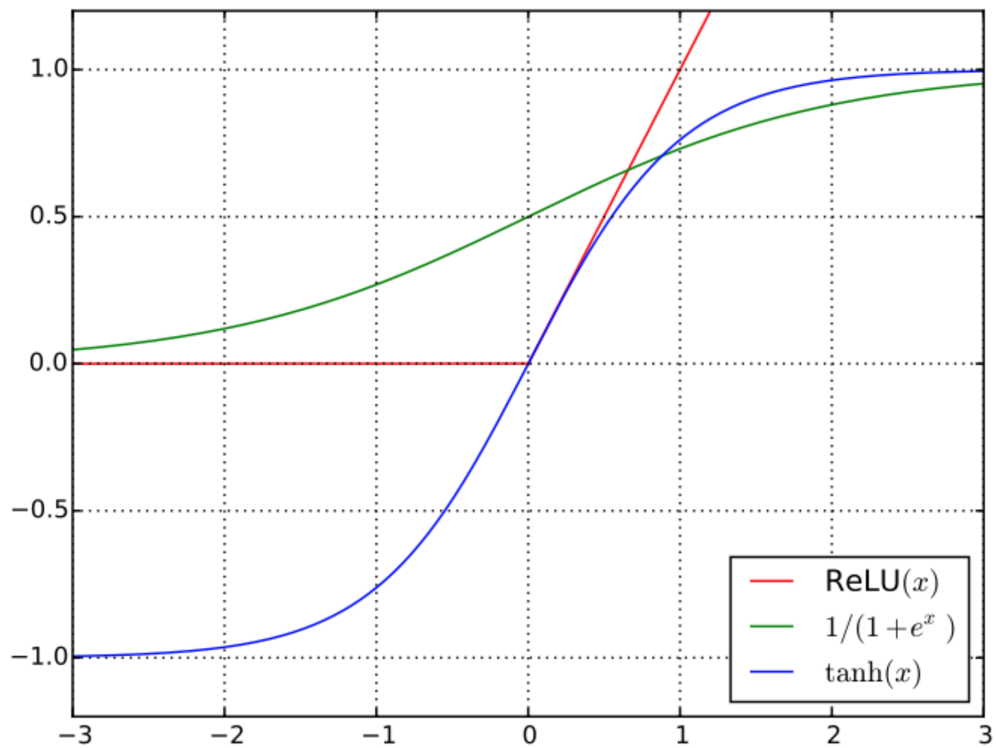
  - Scales well with dataset and model size
  - But can have some convergence difficulties

  - Improvements include: Momentum, RMSprop, AdaGrad, …





Better Activation Functions



- **Vanishing gradient problem**
  - Derivative of sigmoid:

$$\frac{\partial\sigma(x)}{\partial x} = \sigma(x)(1-\sigma(x))$$

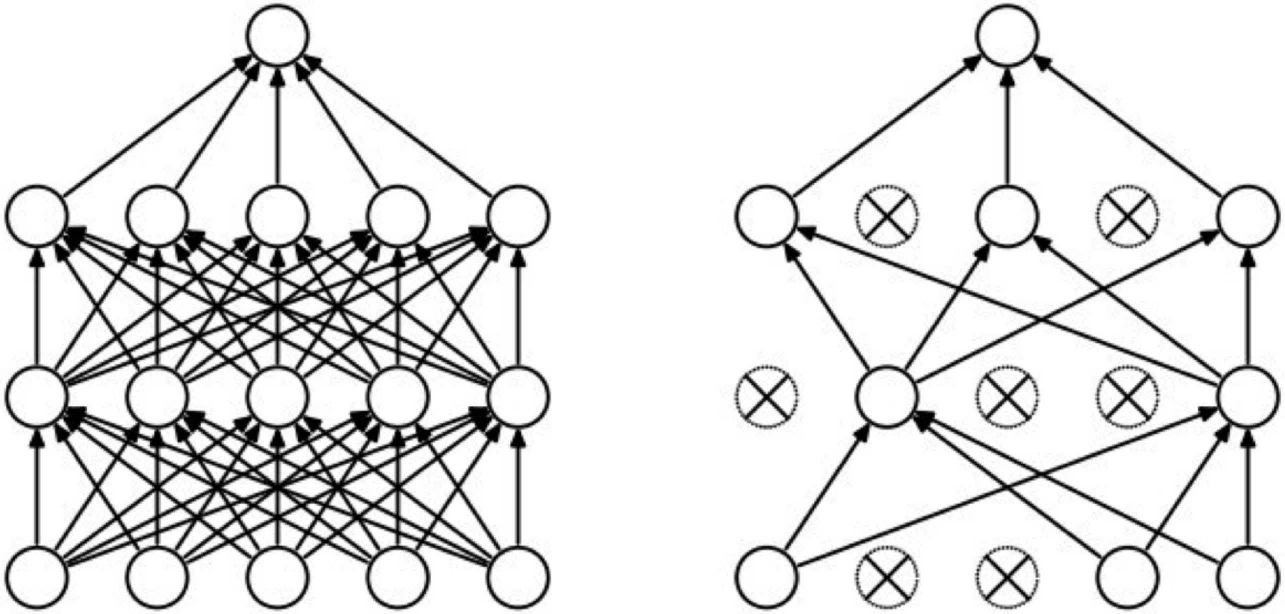
  - Nearly 0 when x is far from 0!
  - Gradient descent impossible!

- **Rectified Linear Unit (ReLU)**
  - ReLU(x) = max{0, x}
  - Derivative is constant!

$$\frac{\partial \mathrm{Re}LU(x)}{\partial x} = \begin{cases} 1 & \text{when } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

  - ReLU gradient doesn't vanish

Better Regularization Inside the Network



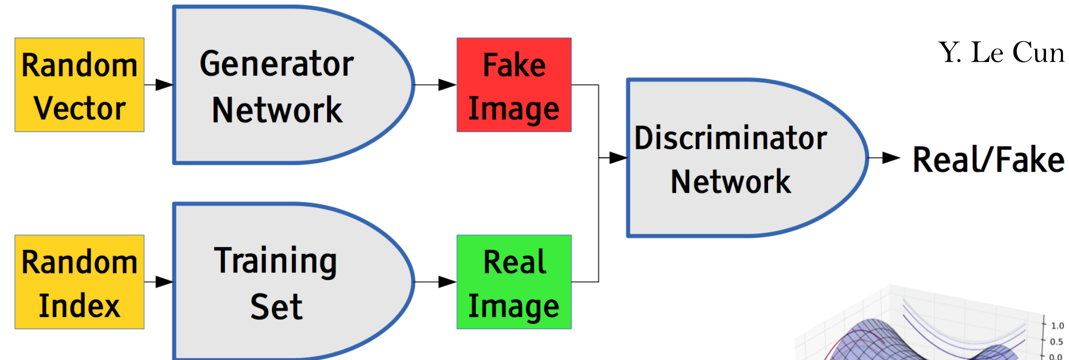
(a) Standard Neural Net (b) After applying dropout.

- Dropout
  - Randomly remove nodes during training
  - Avoid co-adaptation of nodes
  - Essentially a large model averaging procedure

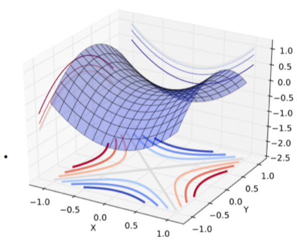
New way to train networks… Potential for HEP?

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

Generative Adversarial Nets



Y. Le Cun

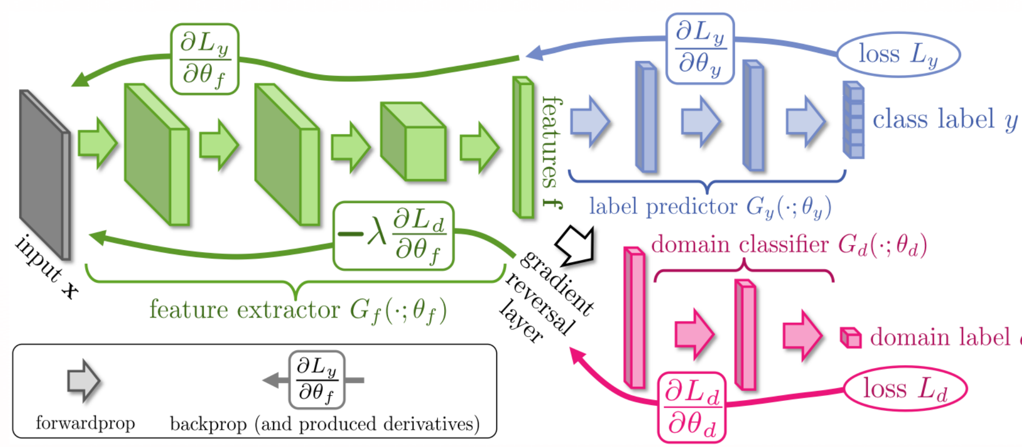


$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

https://arxiv.org/abs/1406.2661

- 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?

Gradient Reversal Layers and Domain Adaptation



http://arxiv.org/abs/1409.7495