MACHINE LEARNING IN ATLAS

Savannah Thais

Yale University

SYNPA 2017, Gwangju, 11/17/17

Outline

1. Introduction
2. Particle ID with Machine Learning
    1. W and top classification
3. Jet Images and Computer Vision
    1. Basics
    2. Quark vs gluon jet classification
    3. Understanding what the network is learning
4. Adversarial Networks
    1. Basics
    2. Jet Image Generation
    3. Decorrelation Studies
5. Recursive Neural Networks
    1. Basics
    2. Jet Classification
    3. B-tagging

Machine Learning Introduction

- Allows an algorithm to learn patterns without being explicitly programmed
- Focus of this talk: algorithms trained on simulated data where truth values are known
  - Some studies using unlabeled data, beyond the scope of this talk
- Many different algorithms exist, general procedure is:
  - Initial algorithm parameters are random
  - Simulated data is fed through the algorithm
  - Predicted classification is compared to truth classification, error is quantified according to a loss function
  - Error is back-propagated to adjust the algorithm parameters
  - This process is repeated until stopping criteria is reached
- Important considerations:
  - Dependency on variables used and algorithm hyperparameters
  - Must check for overtraining

Machine Learning for HEP

- Can utilize information not available in cut-based techniques:
  - Variables whose distributions over lap (exploit shape differences)
  - Non-linear correlations between input variables
  - Low-level variables
- Can reduce dependence on systematic uncertainties
- Can mitigate effects of simulation mis-modeling
- Often much faster than current techniques

Generally better performance on classification and similar tasks

HEP Environments

L1 Trigger	High Level Trigger	Offline Computing	Analysis
• Makes very fast decision (10 μs) • Based on coarse reconstruction in ROI • Implemented at hardware level	• Slightly longer decision time (30 ms) • Based on simplified global reconstruction • Implemented at software level	• Even longer computation time (20 s) • Reconstructs, identifies, isolates, and calibrates all particles	• Computation time (approximately) not important • Code written by individual groups • Some centrally produced algorithms

HEP Environments

L1 Trigger → High Level Trigger → Offline Computing → Analysis

L1 Trigger
- Makes very fast decision (10 μs)
- Based on coarse reconstruction in ROI
- Implemented at hardware level

High Level Trigger
- Slightly longer decision time (30 ms)
- Based on simplified global reconstruction
- Implemented at software level

Offline Computing
- Even longer computation time (20 s)
- Reconstructs, identifies, isolates, and calibrates all particles

Analysis
- Computation time (approximately) not important
- Code written by individual groups
- Some centrally produced algorithms

Simulation
- Used to tune algorithms at all stages
- Computationally expensive to do well
- Known mis-modeling problems and generator dependencies

HEP Environments

L1 Trigger

- Makes very fast decision (10 μs)
- Based on coarse reconstruction in ROI
- Implemented at hardware level

High Level Trigger

- Slightly longer decision time (30 ms)
- Based on simplified global reconstruction
- Implemented at software level

Offline Computing

- Even longer computation time (20 s)
- Reconstructs, identifies, isolates, and calibrates all particles

Analysis

- Computation time (approximately) not important
- Code written by individual groups
- Some centrally produced algorithms

Simulation

- Used to tune algorithms at all stages
- Computationally expensive to do well
- Known mis-modeling problems and generator dependencies

Particle ID with Machine Learning

Paper here

$W^{\pm}$ and Top Quarks

Want to separate hadronically decaying $W^{\pm}$ and top quarks from general QCD jet background

Training data construction:

1. Reconstruct jets with standard anti-k_t algorithm and trimming
2. Calculate jet substructure variables
3. Reconstruct 'truth jets' from long-lived particles
4. Match jets with truth jets and original truth particles to get labels

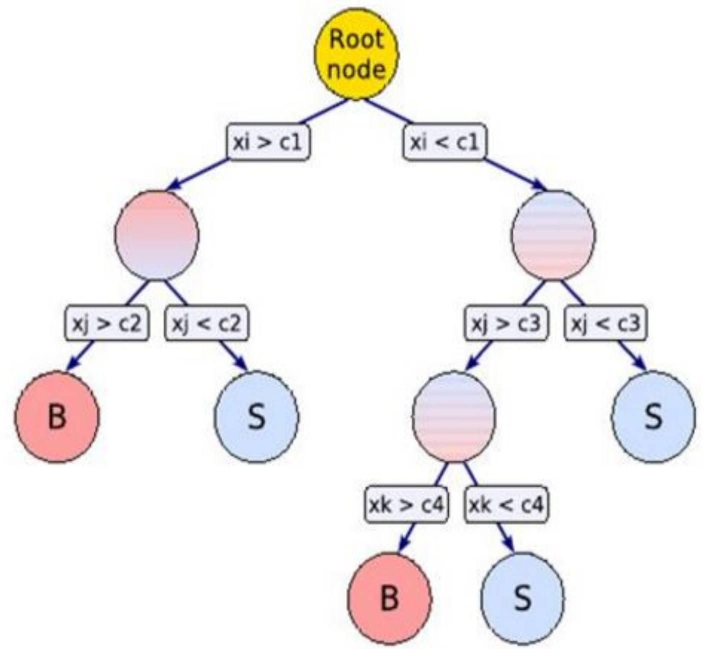
Training variables:

Observable	Variable	Used For
Energy Correlation Ratios	ECF_1, ECF_2, ECF_3 C_2,D_2	top,W
N-subjettiness	τ_1,τ_2,τ_3 τ_{21},τ_{32}	top,W
Center of Mass Observables	Fox Wolfram (R_2^{FW}) Sphericity (S) Thrust (T_{MIN},T_{MAJ})	W W W
Splitting Measures	Z_{CUT} μ_{12} $\sqrt{d_{12}}$,$\sqrt{d_{23}}$	W W top,W
Planar Flow	$\mathcal{P}$	W
Dipolarity	$\mathcal{D}$	W
Angularity	a_3	W
Aplanarity	A	W
KtDR	$KtDR$	W
Qw	Q_w	top

Boosted Decision Trees

Combine many shallow decision trees into a boosted forest

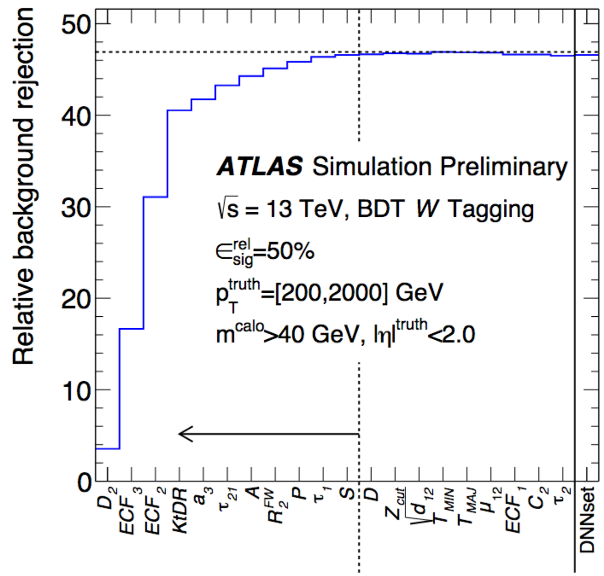
1. All events (signal and background) are equally weighted and mixed at the top of first tree
2. At each branching, an optimal cut is found to separate S and B
3. When algorithm stops (due to predefined # of branches or events per node) each node is assigned an S or B label
4. Boosted: all incorrectly classified events are given a higher weight for next tree
5. Final classifier is the weighted average of the forest



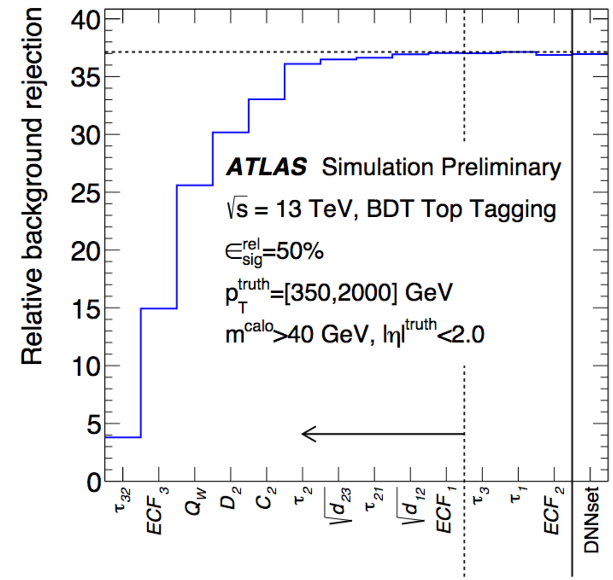
BDT Training

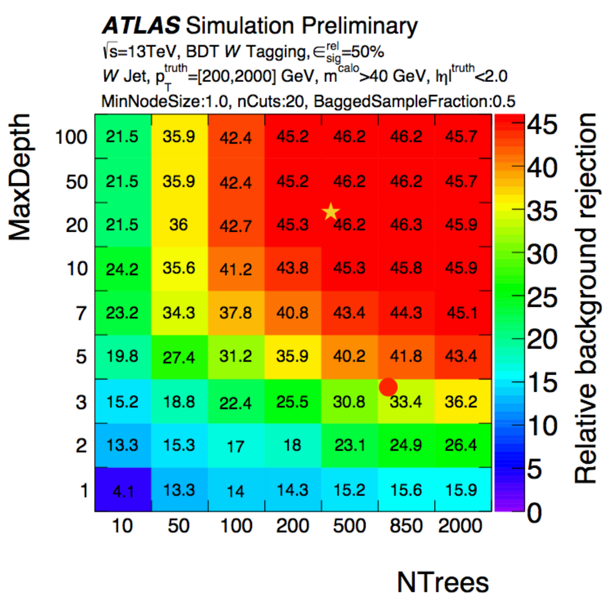
Iteratively add variables to pick best set

$W^{\pm}$



Top





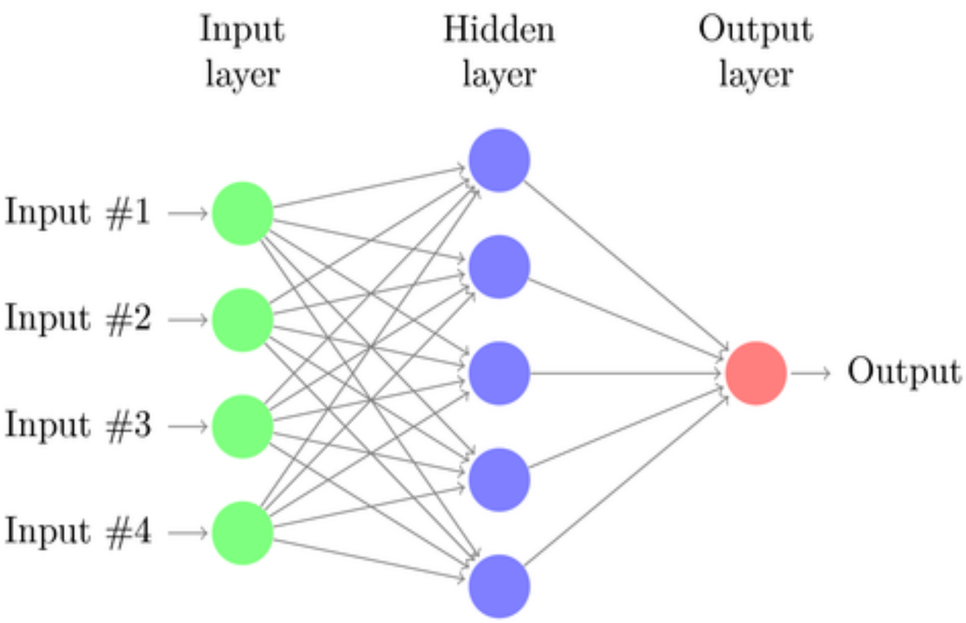
Scan hyper-parameter space to optimize

Check for over-training with cross validation

Neural Networks

Based on biological networks: a collection of connected nodes that pass information downstream

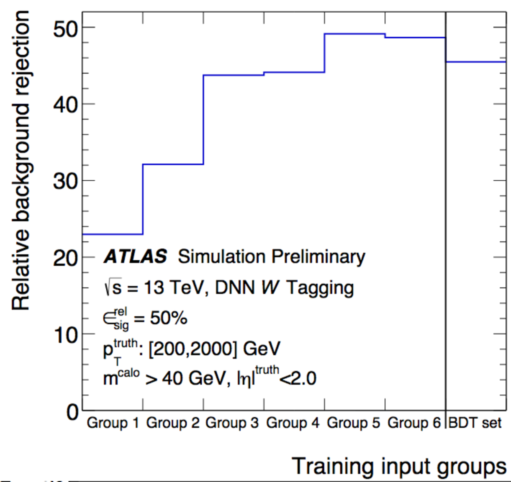
1. Define a structure of multiple layers, each with different numbers of nodes
2. Define a (initially random) matrix for dimensional transformation between the layers
3. Feed data through the network to predict a classification probability (0 to 1)
4. Back-propagate error through the network and changing the matrix values



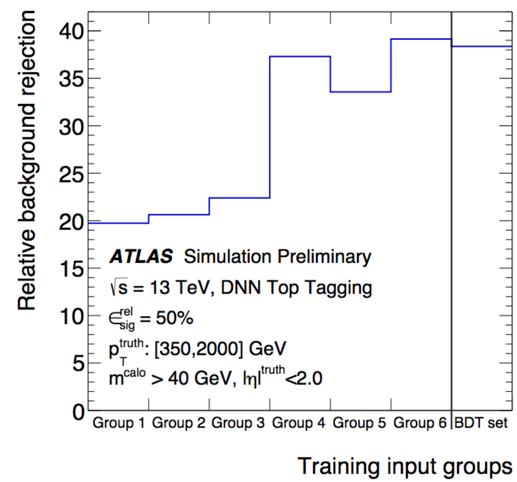
NN Training

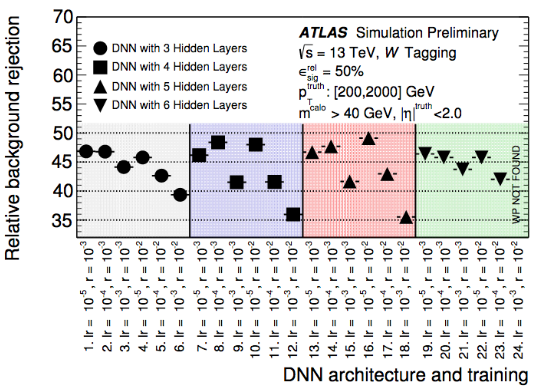
Iteratively add variables in groups to pick best set

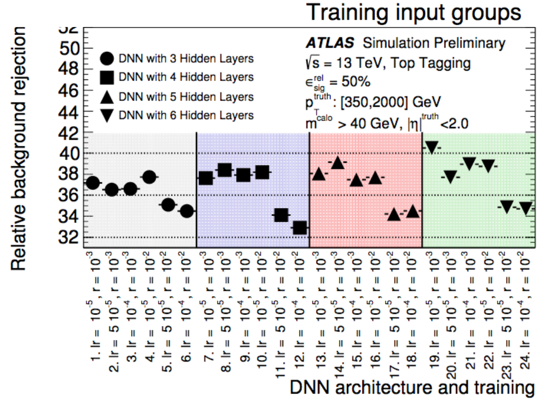
W±



Top

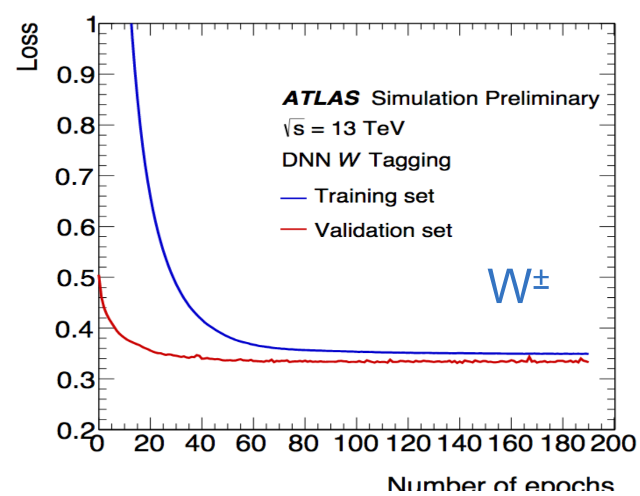






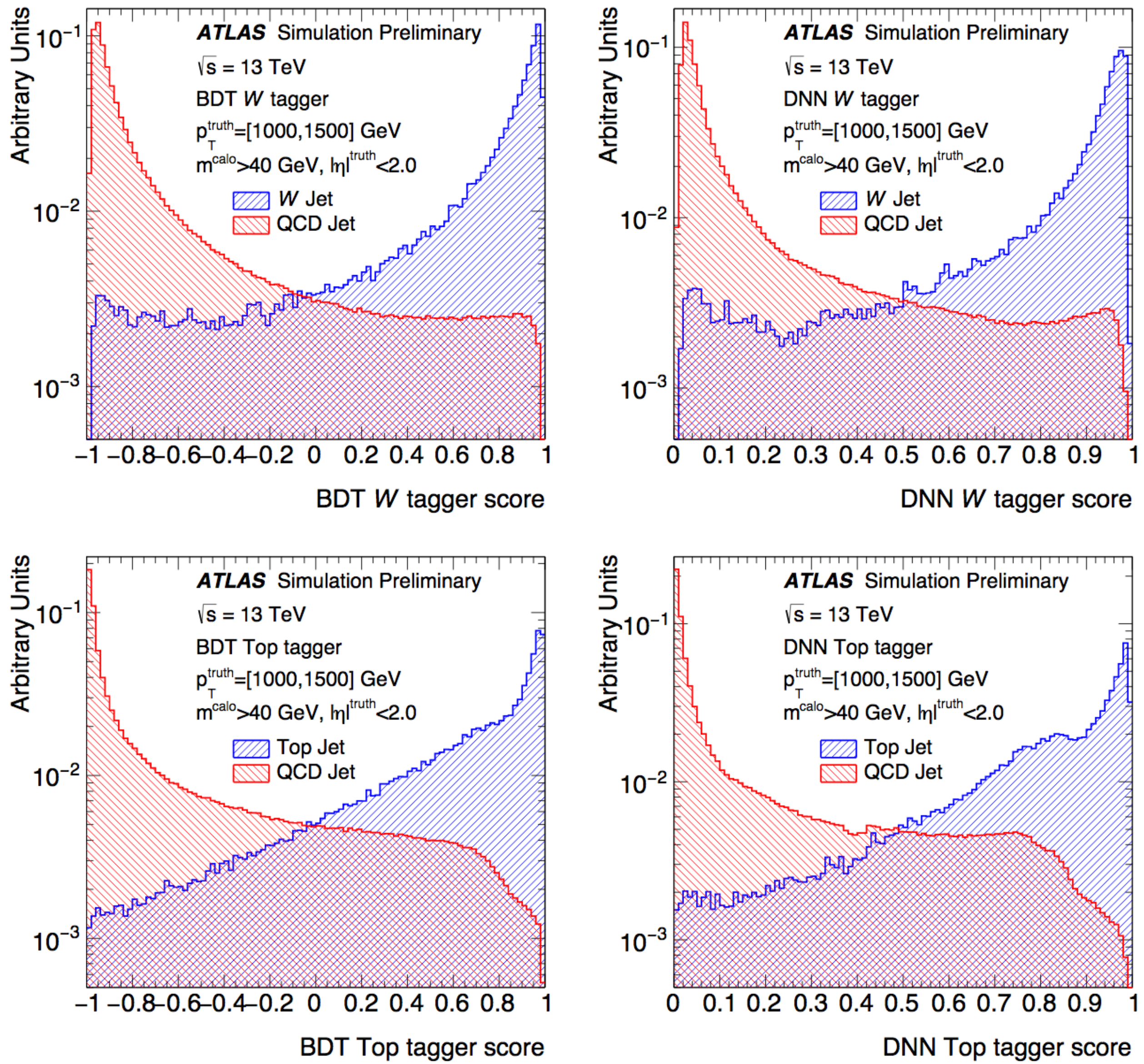
Scan hyper-parameter space to optimize

Check for over-training with cross validation



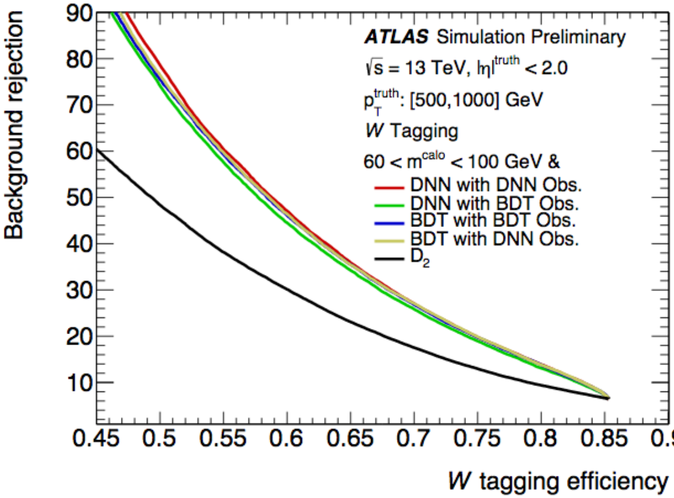
Results

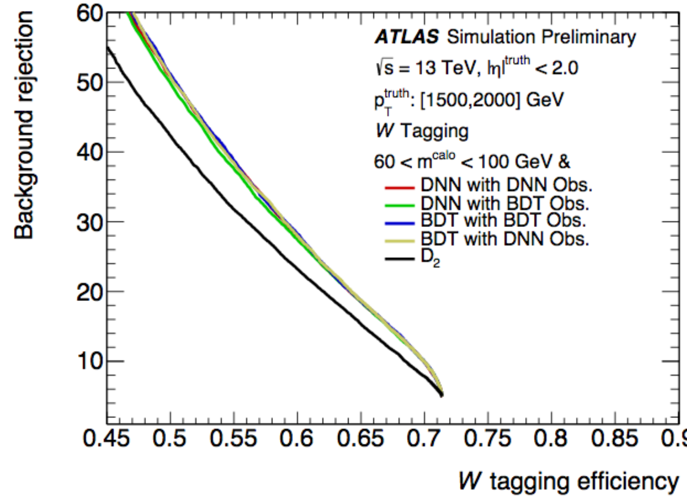
- Trained algorithms produce discriminant distributions
- Select cut for desired efficiency



Results

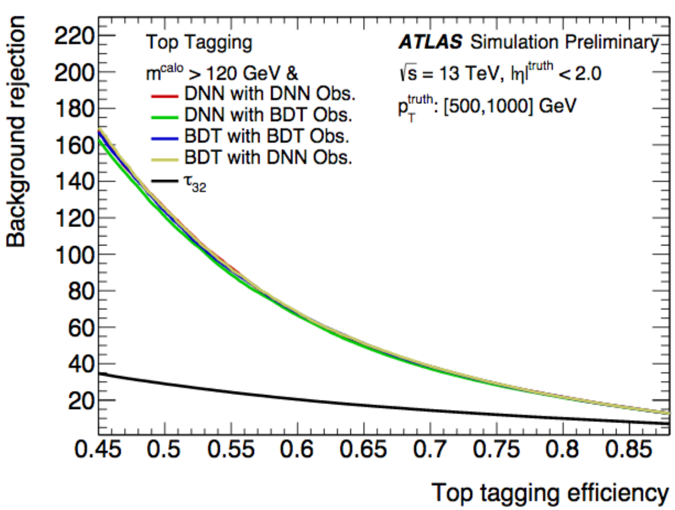
W±

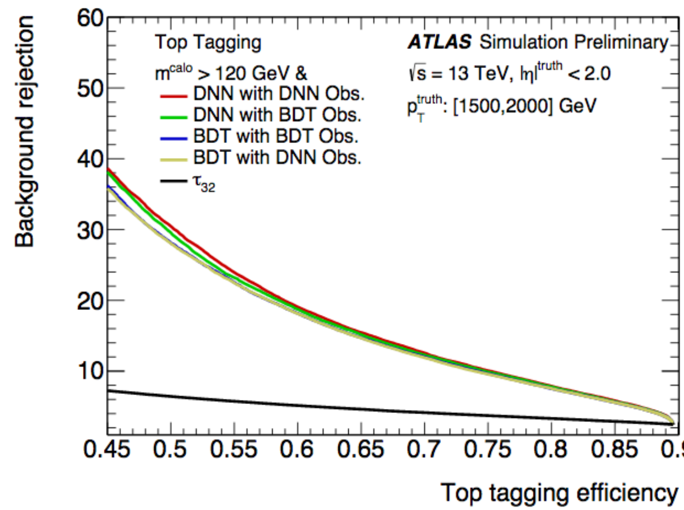




BDTs and NNs outperform cuts on physics motivated variables

Top



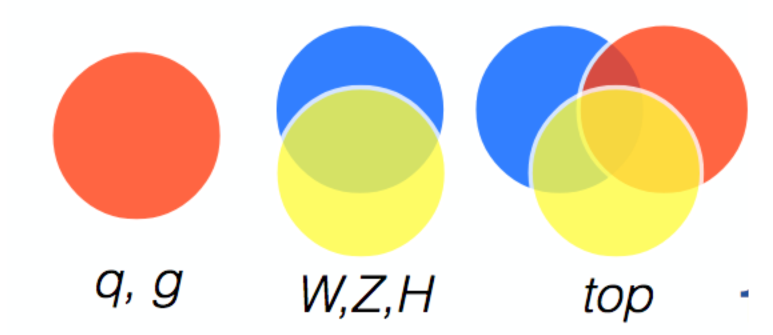


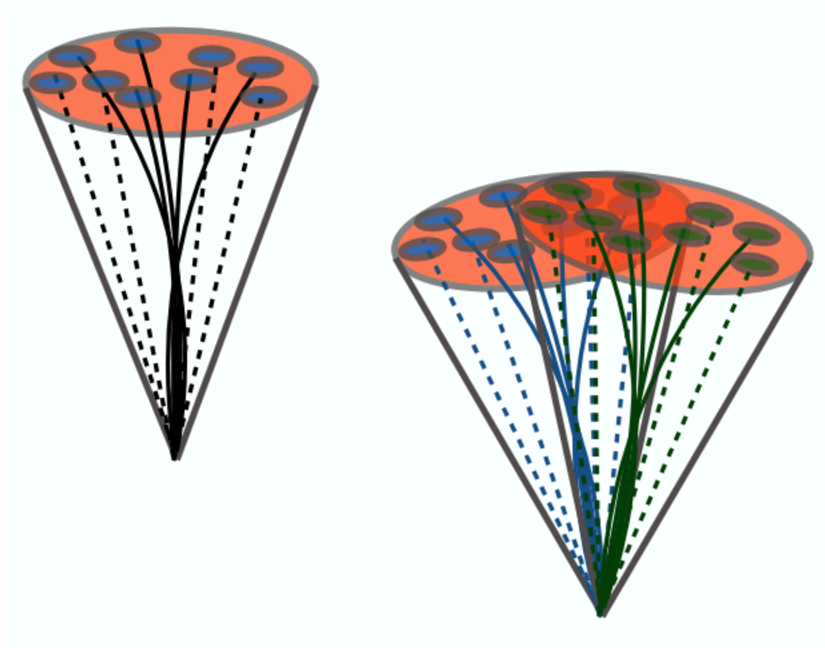
BDTs and NNs perform similarly even when trained with other variables

Jet Images and Computer Vision

Jets in ATLAS

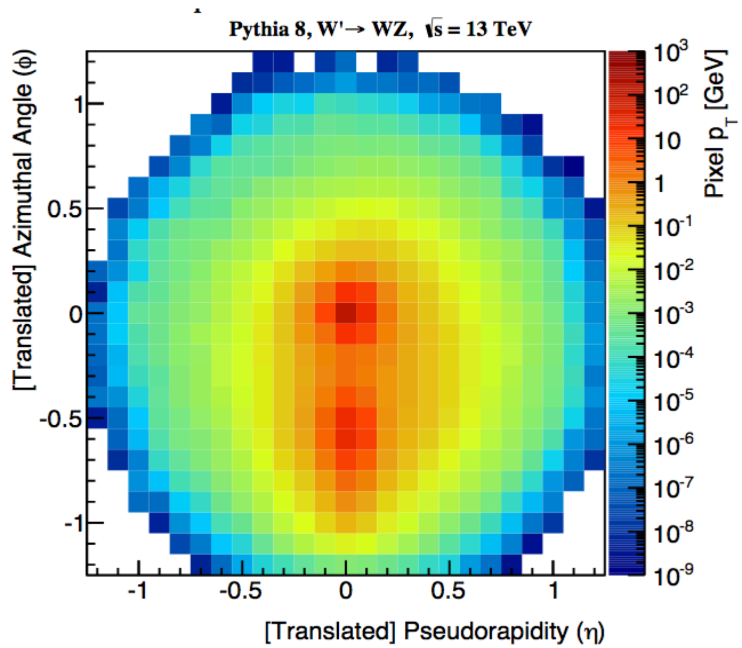
- Cone-like showers of quarks and gluons that produce more particles all close to each other
- Can come from QCD processes or boosted bosons and tops
- Typically identified using constructed variables that describe substructure inside jet

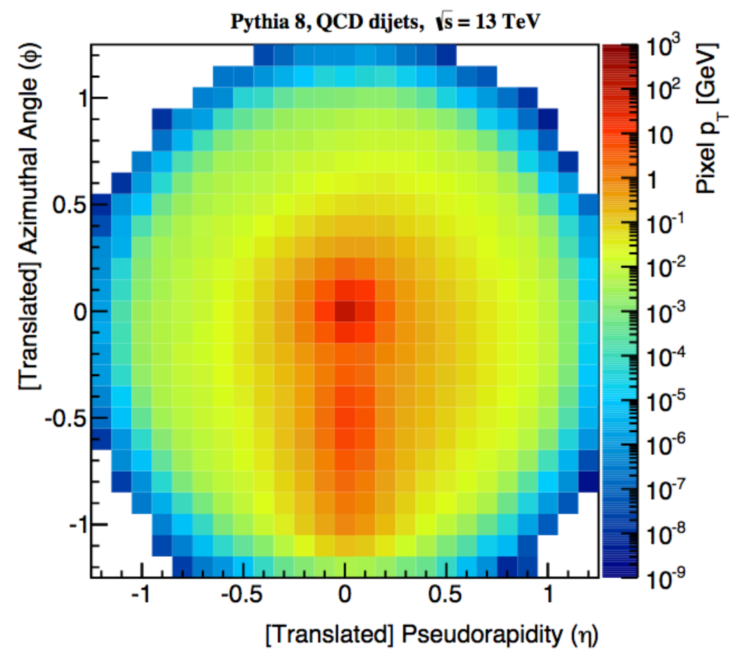




Jet Images

Cells in the calorimeter become pixels in an image





1. Center the image on largest energy deposit
2. Select a fixed window size around center
3. Color pixel according to energy deposited in that cell

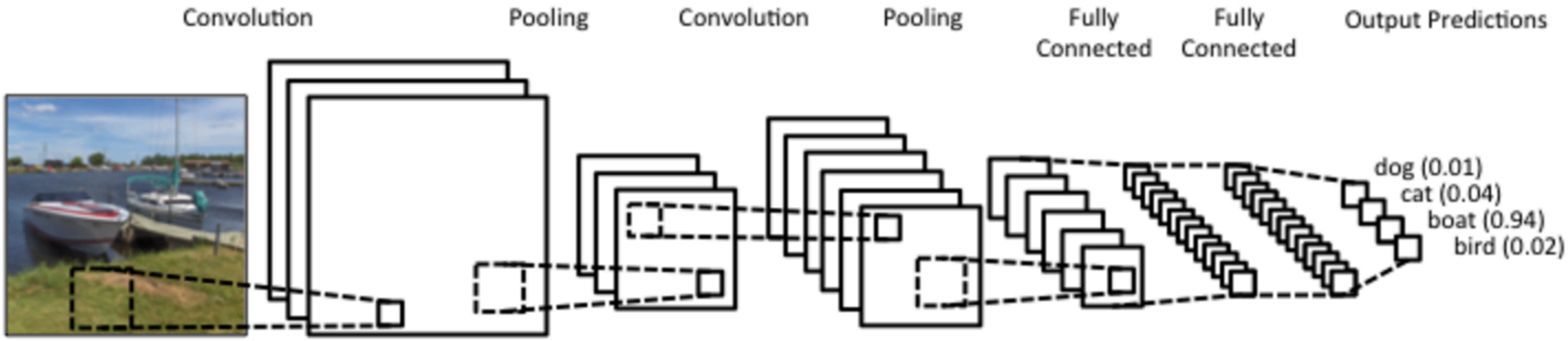
Note: jet images are sparser than images in other computer vision applications and do not have well defined edges → introduces new difficulties

Convolutional Neural Networks

A type of deep NN typically used for image processing

Consist of some combination of 3 layer types:

- Convolution: a set of learnable filters (kernels) that are convolved across the width and height of input data using a sliding window
- Pooling: provides non-linear down-sampling by combining the outputs of several neurons
- Fully Connected: traditional NN layer

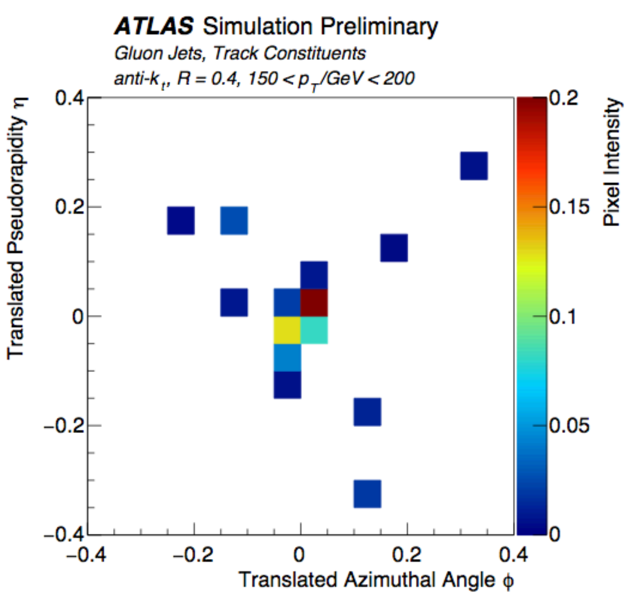


Paper Here

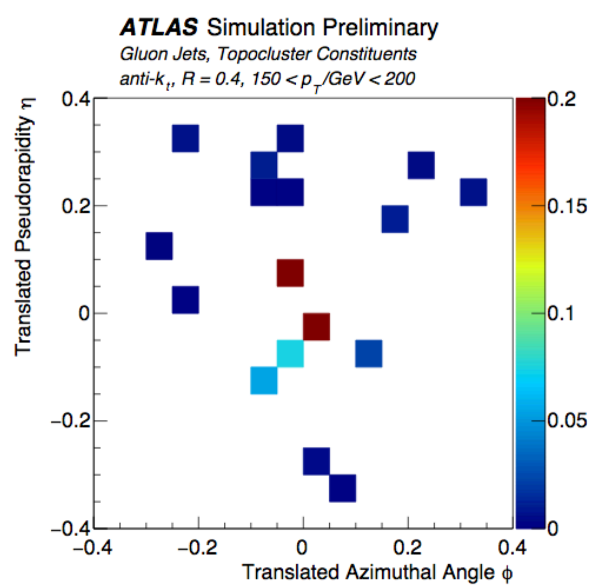
Quark vs Gluon Jet ID: Data

Looked at 3 ways to calculate pixel energy:

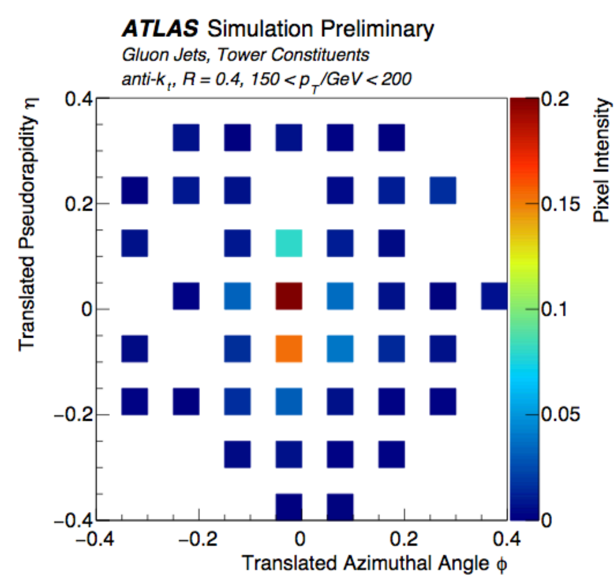
- Topo-clusters: groups of energy deposits, used for jet clustering
- Calo-towers: fixed size division of calorimeter projected onto grid
- Tracks: tracks associated to jets with ghost-association



Tracks



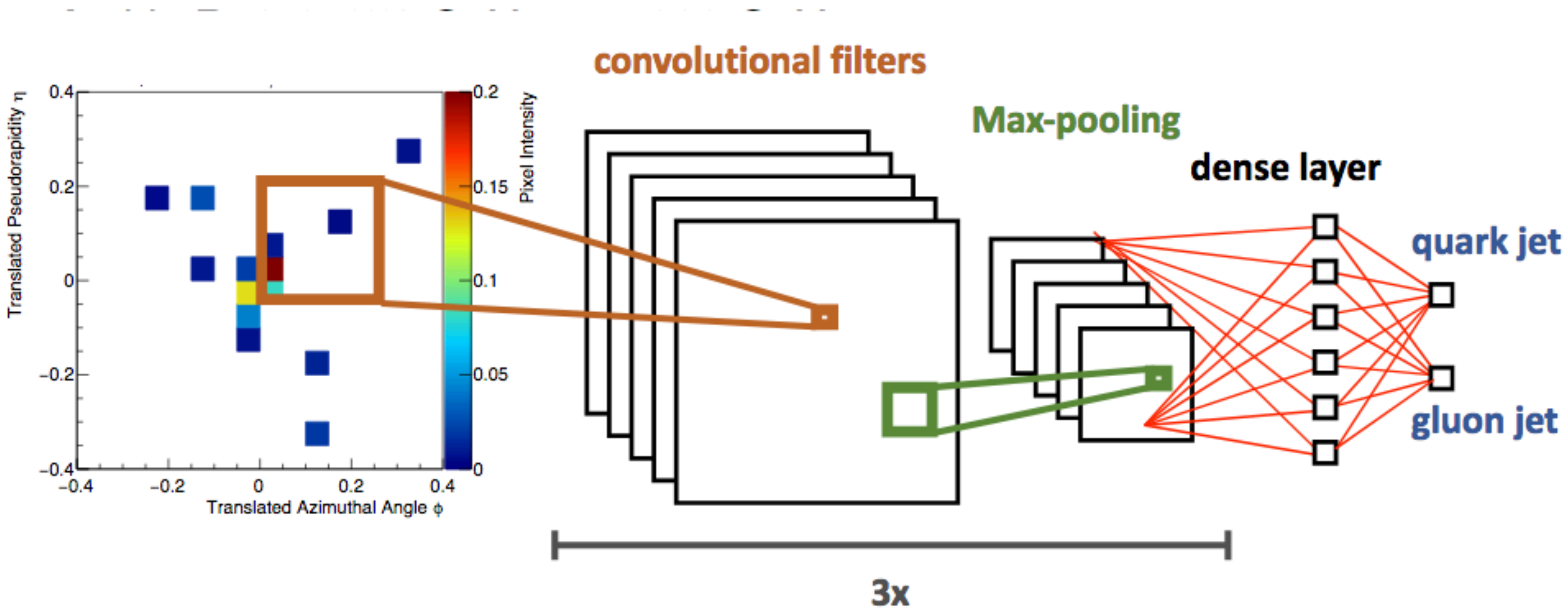
Topoclusters



Towers

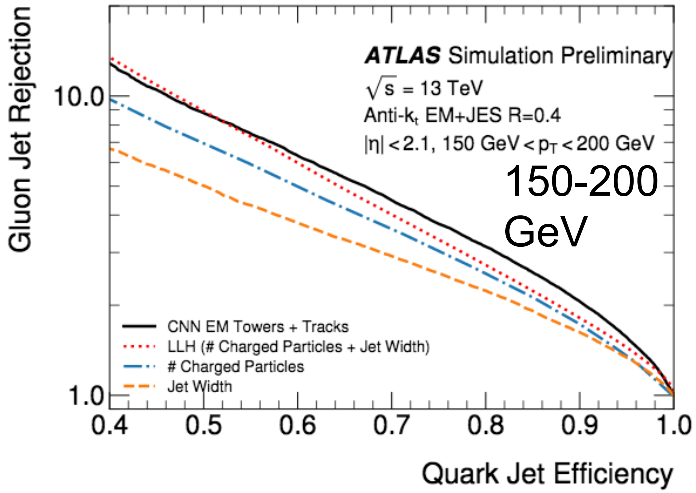
Also pre-processed images to exploit space-time symmetries (in backup)

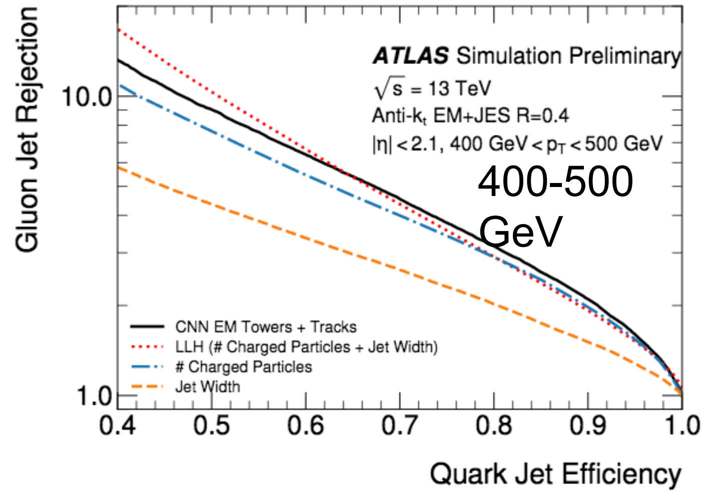
Quark vs Gluon Jet ID: Network



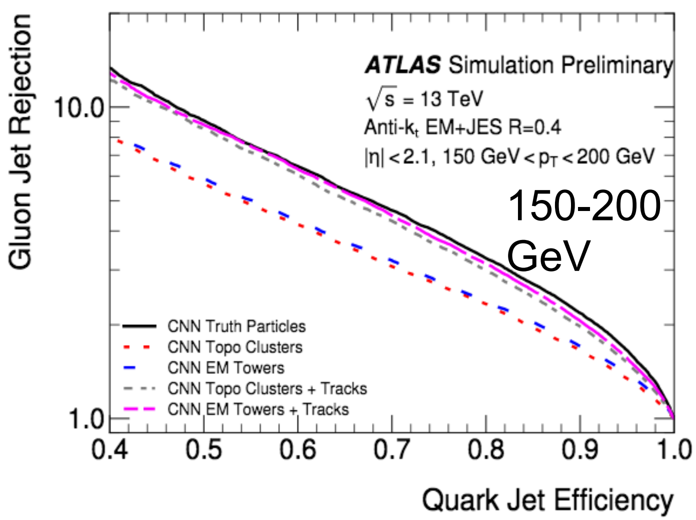
- 3 convolution and max pooling combinations
- Final output is softmax probability of being quark jet or gluon jet

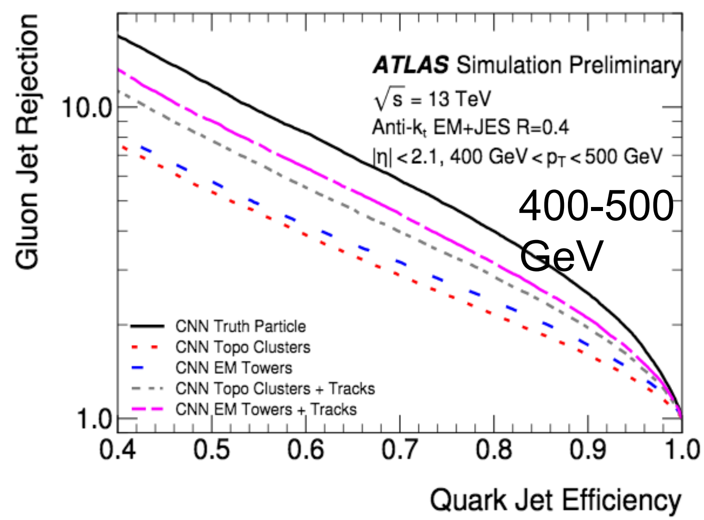
Quark vs Gluon Jet ID: Results





Comparisons to physics motivated variables





Comparisons of pixelization schemes

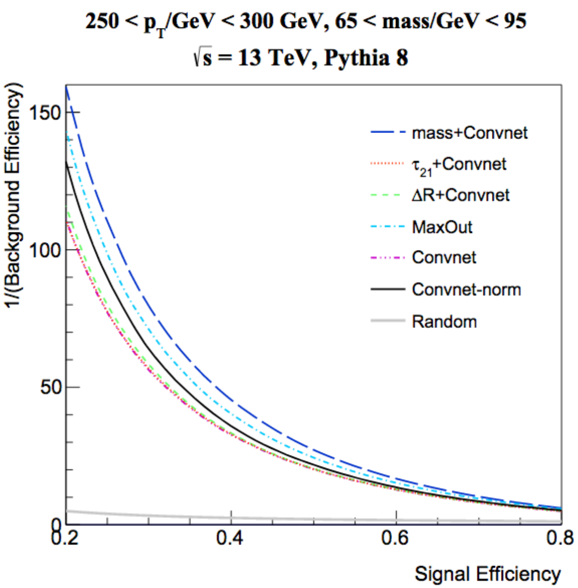
Paper Here

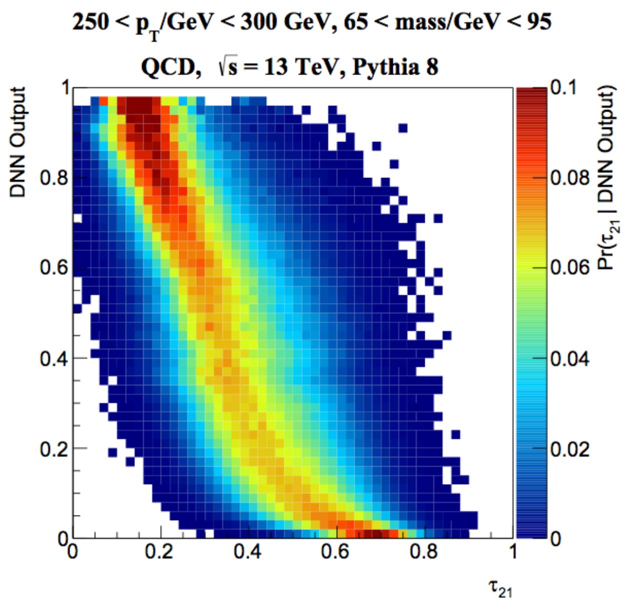
What Is It Learning?

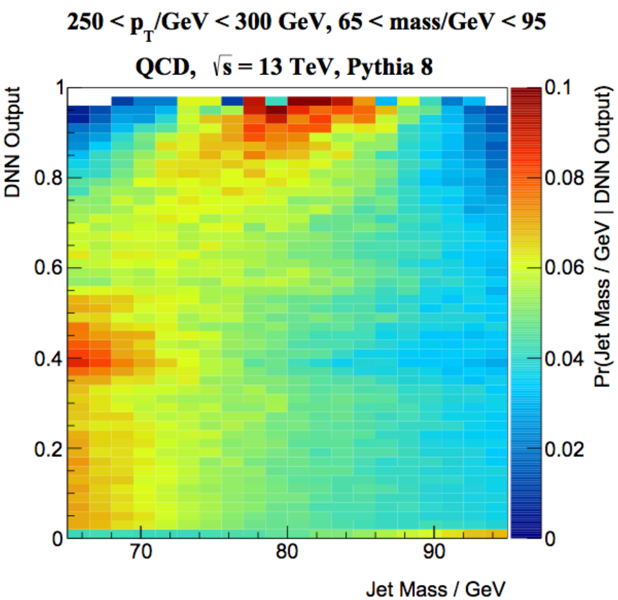
Now look at a study on separating W jets from QCD jets

Combine DNN with physics variables

Look at correlation of DNN output with physics variables

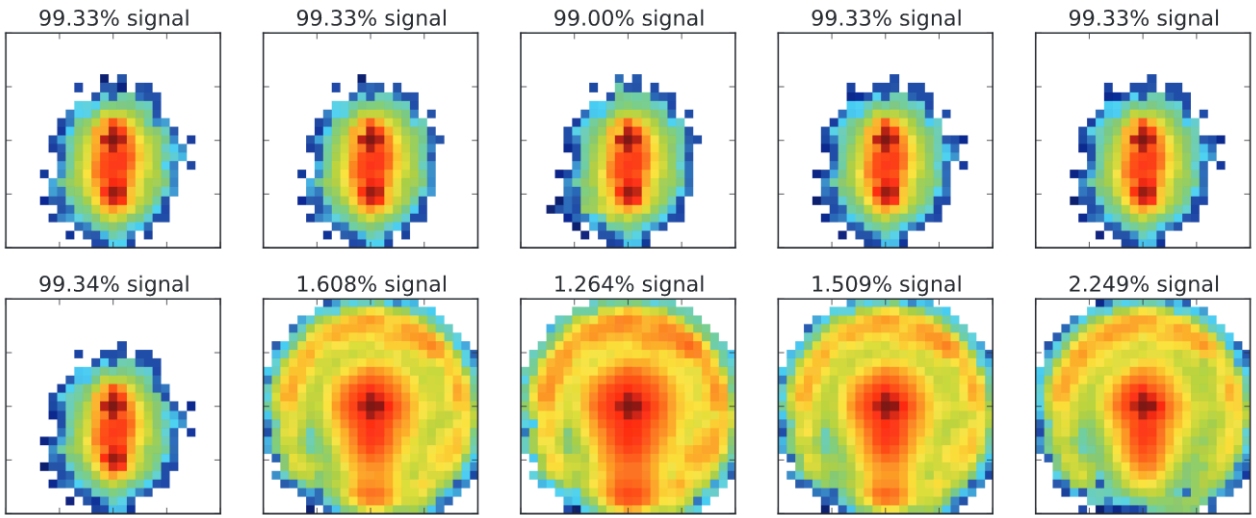




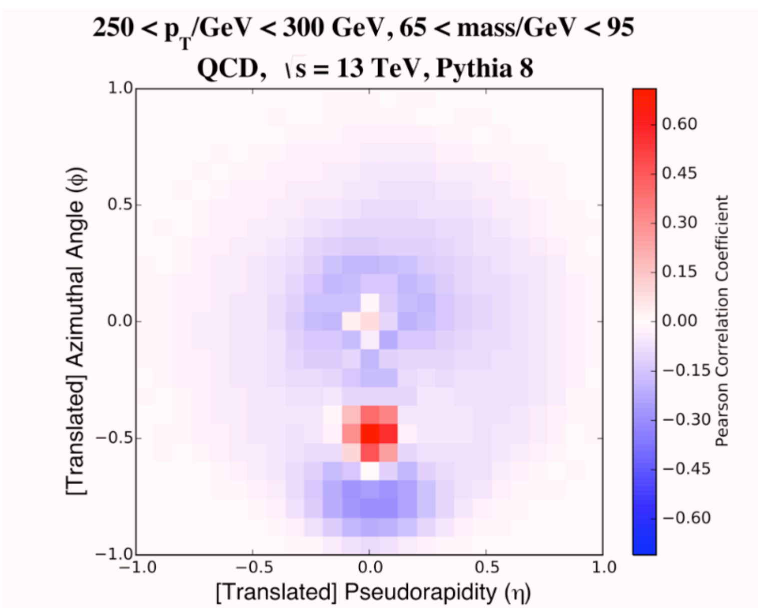


Network learns most variables, but doesn't entirely learn jet mass

What Is It Learning?



Look at average of 500 most activating images for different nodes

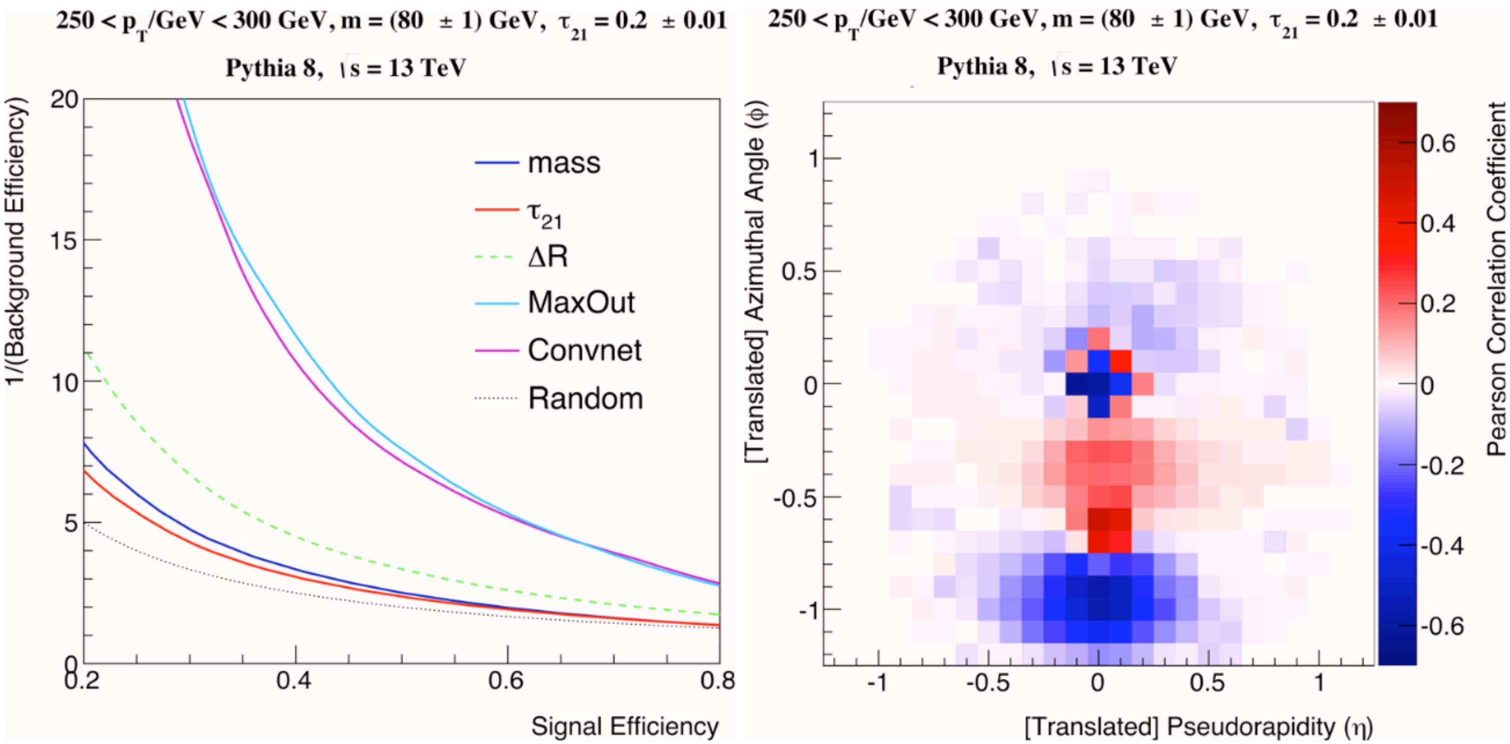


Look at correlation of each pixel with classification output

Learning that QCD background has wider radiation and W has 2 clear prongs!

What Is It Learning?

Restrict phase space to eliminate power of substructure variables



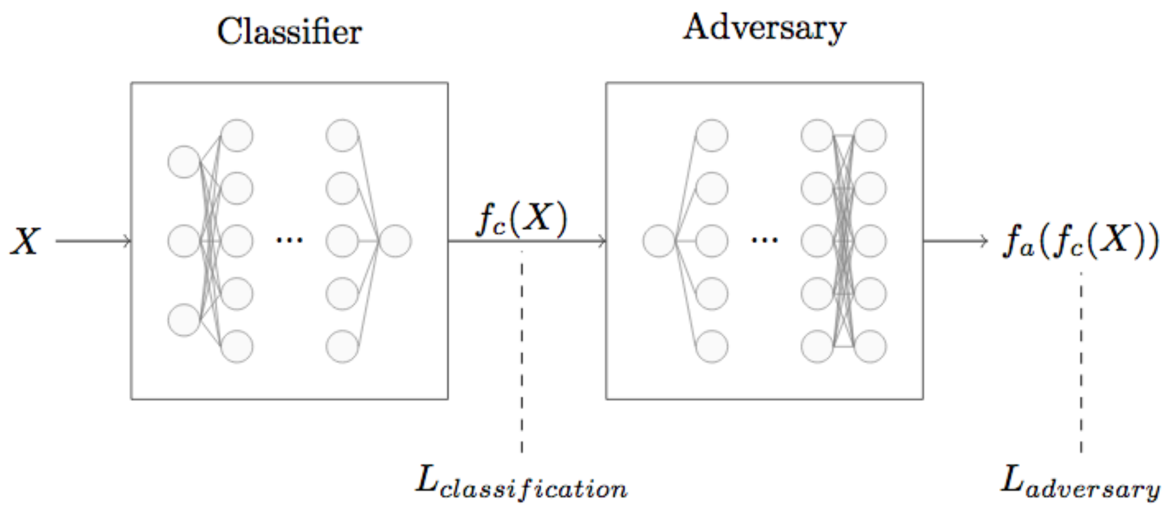
Network is learning additional information outside of substructure!

Adversarial Networks

Adversarial Networks

- Pit 2 networks against each other in a non-cooperative game
- Adversary network takes output of main task network and tries to predict something from it
- Loss function becomes combination of competing objectives

$$\boxed{E(\theta_f, \theta_r) = \mathcal{L}_f(\theta_f) - \mathcal{L}_r(\theta_f, \theta_r)}$$



Simulations in ATLAS

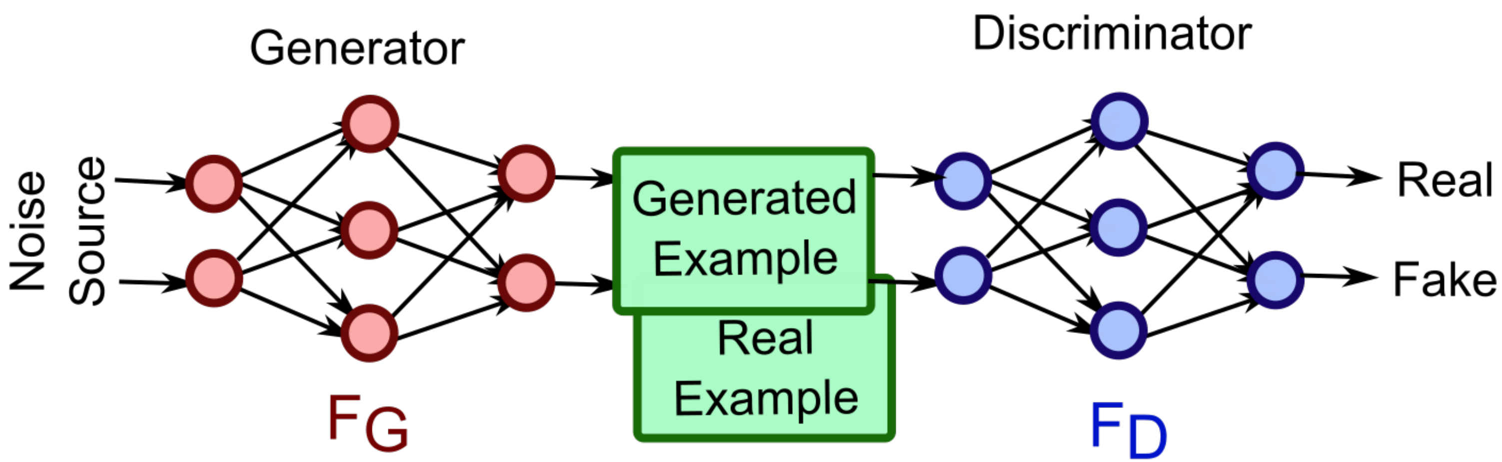
- Full simulations in ATLAS are very computationally expensive (if done well)
- FASTSim reduces CPU time, but is also less accurate
- Many analyses need lots of high quality simulations to optimize their design → currently no good solution

Can we use ML to solve this?

Paper Here

Generative Adversarial Networks

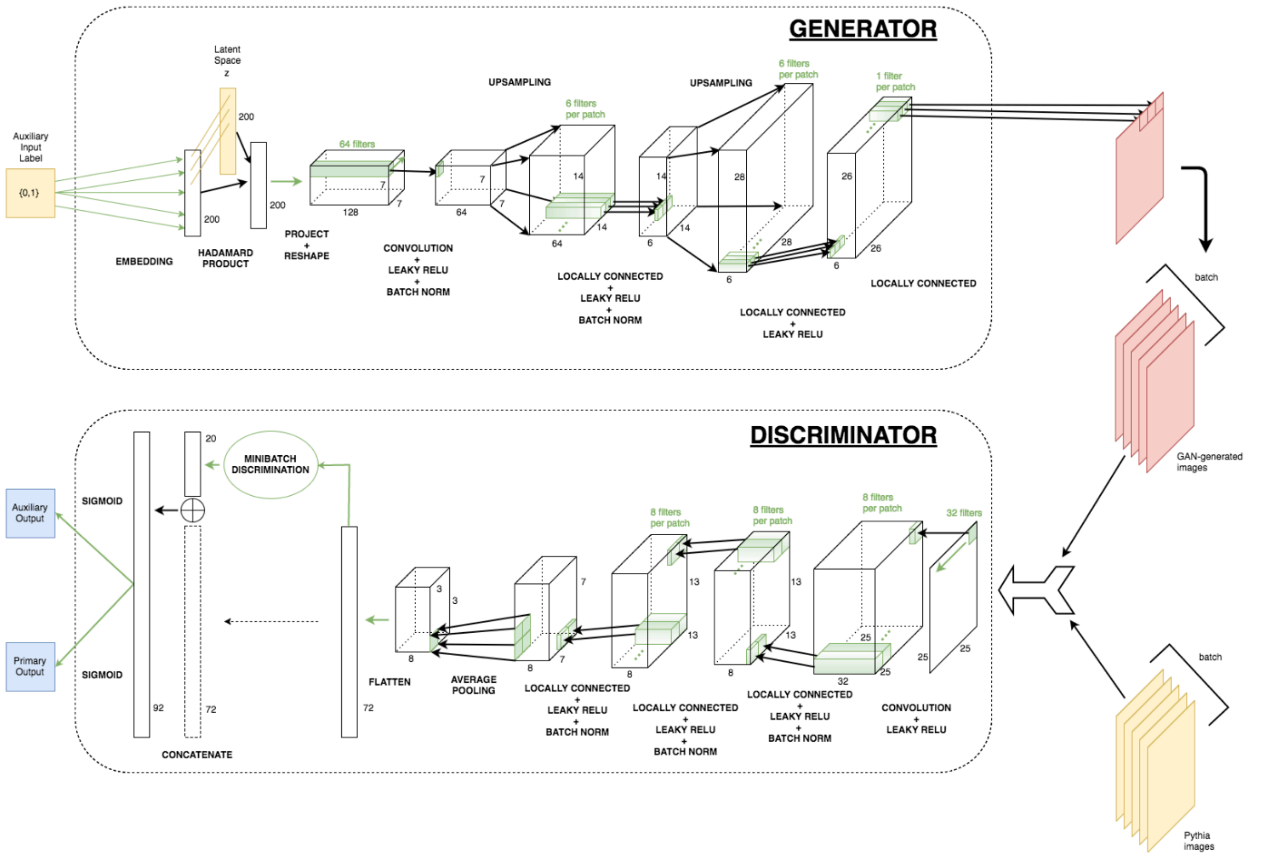
- GANs pit a generator G against a discriminator D
  - G tries to generate physics simulations from random noise input
  - D tries to separate simulations from G from Pythia simulations
- First ATLAS study is generating jet images
- Common problem with GANs is mode collapse: G learns one small feature that is maximally confusing to D
  - Can alleviate this by adding an auxiliary task to D
  - In this study, auxiliary task is distinguishing W jets from QCD jets



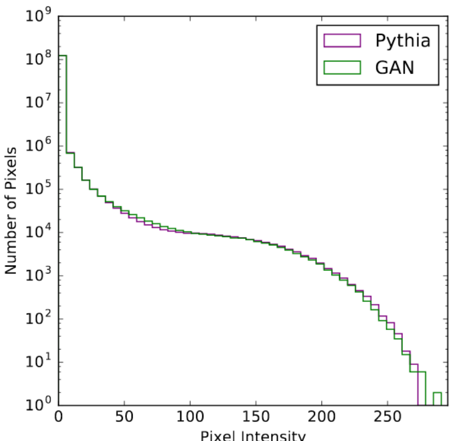
GAN Architecture

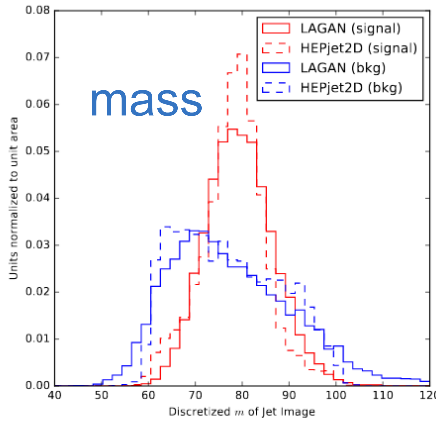
For HEP tasks, create a location aware GAN (LAGAN) with:

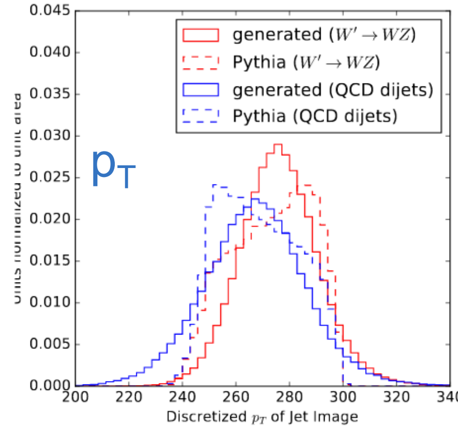
- Locally connected layers
- Rectified Linear Units in last layer to create sparsity
- Batch normalization to help stabilize
- Minibatch discrimination to enforce sparsity and high dynamic range

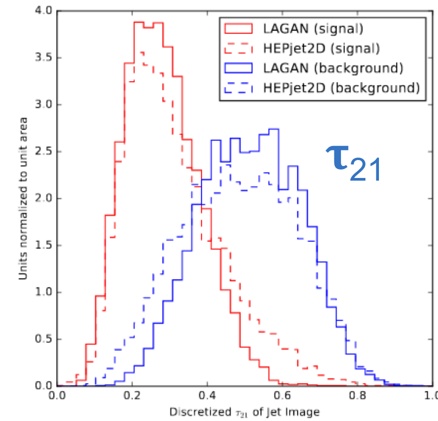


GAN Results

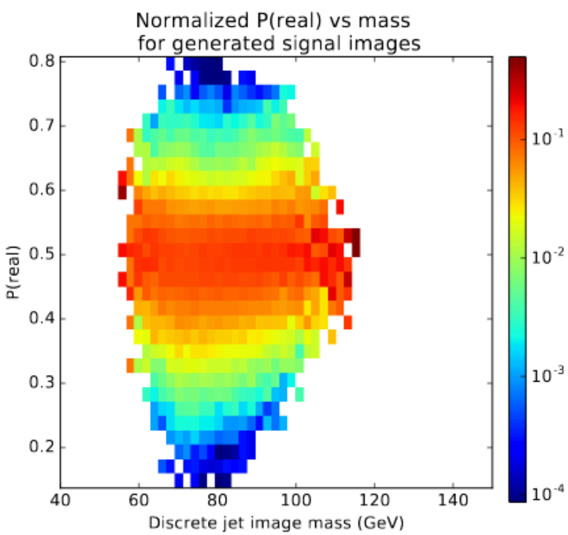


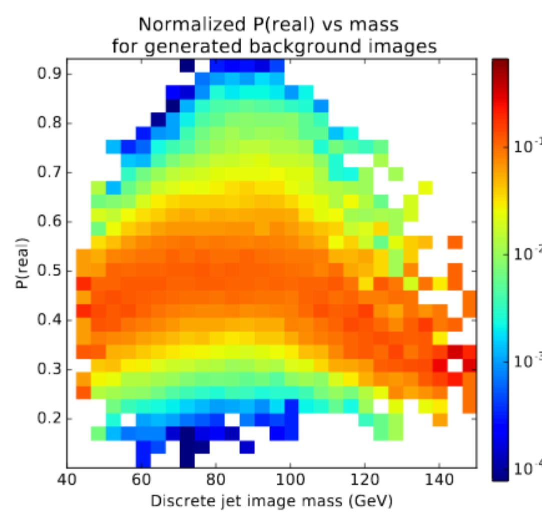






Accurately reproduces pixel intensity and substructure variable distributions





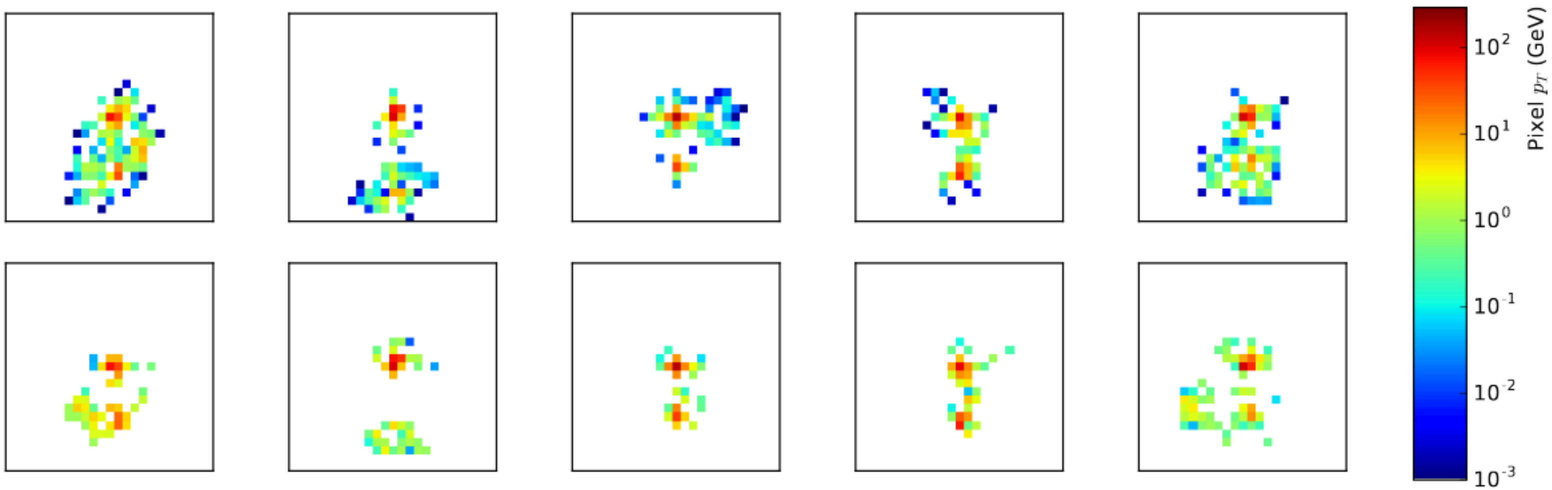
Training converges to stable point where D gives 1/2

What is the GAN Learning?

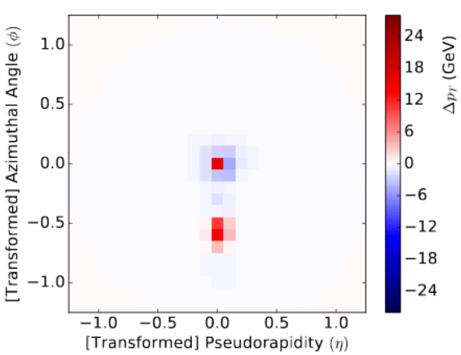
Random Pythia Jets and their nearest GAN neighbors

Pythia

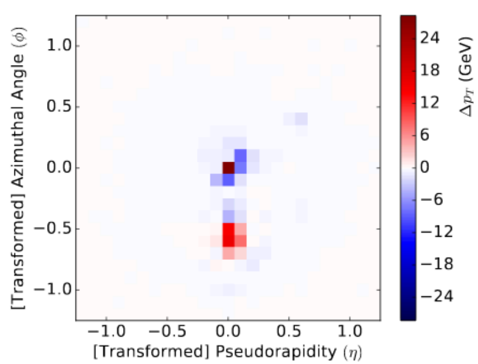
GAN



Signal and Background Correlations

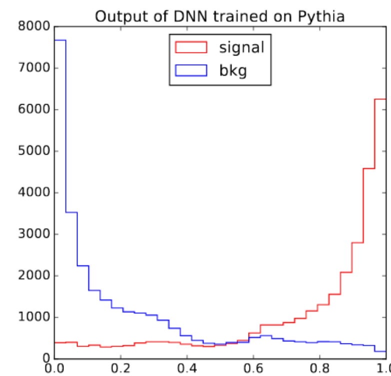


Pythia

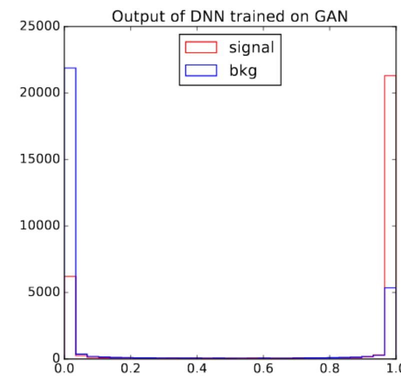


GAN

DNN Output



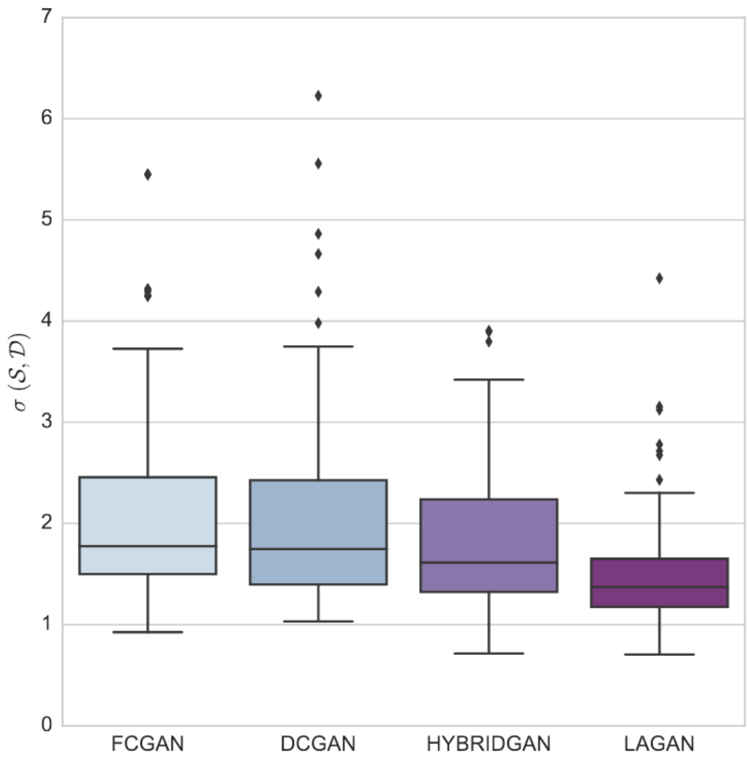
Pythia

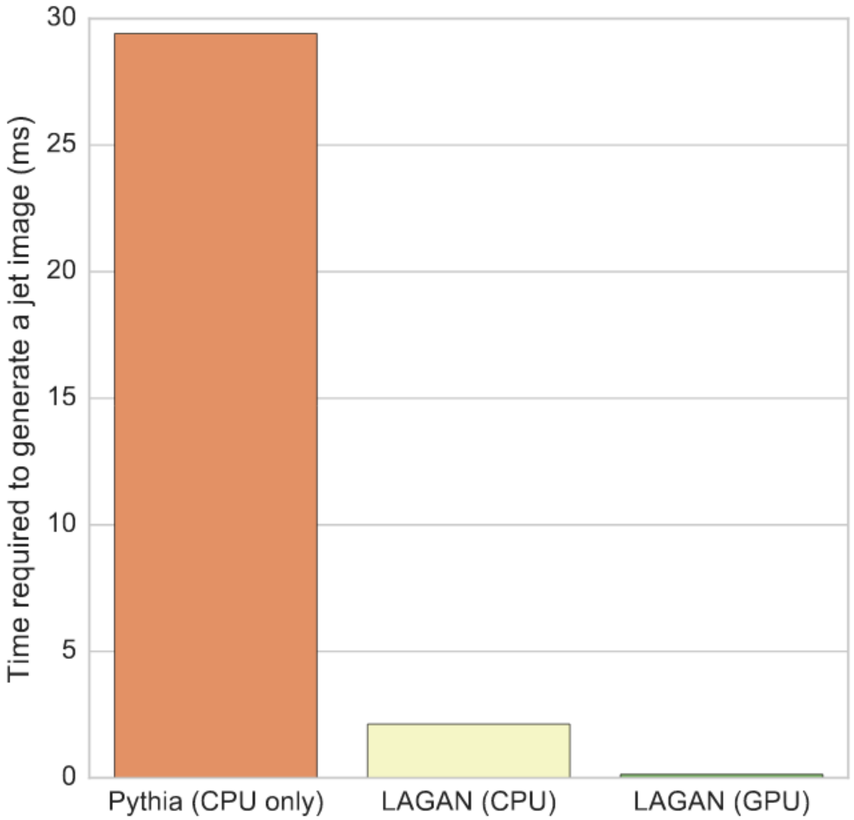


GAN

Learning images well while not memorizing Pythia distributions, but also learning to produce easier to discriminate images

GAN Performance and Speed



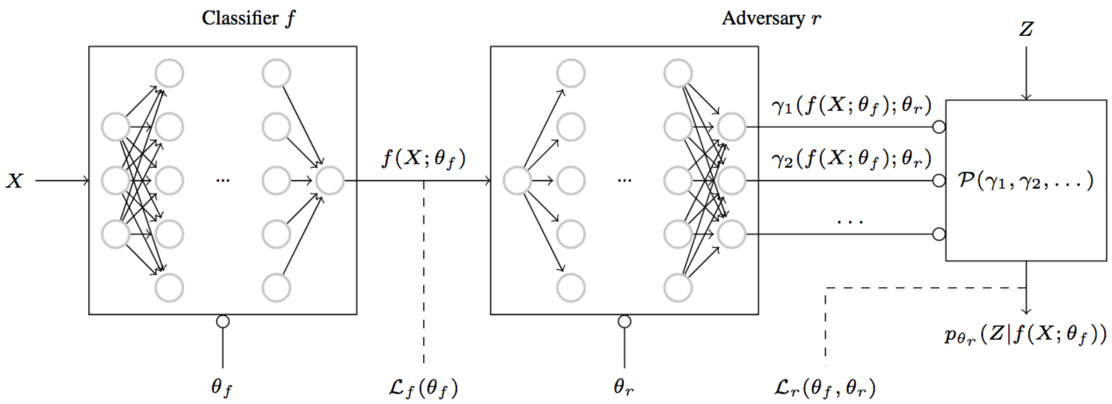


LAGAN performs better than other common GAN architectures

LAGAN is an order of magnitude faster even on CPU

Imposing Constraints

- Outside of simulation generation, can use ANNs to impose physics driven constraints on training
- Big challenge in HEP is robustness with respect to systematic uncertainties and changing conditions
- To train a discriminator robust to or de-correlated from a physics variable, train adversary to reproduce this variable from the output of the classifier



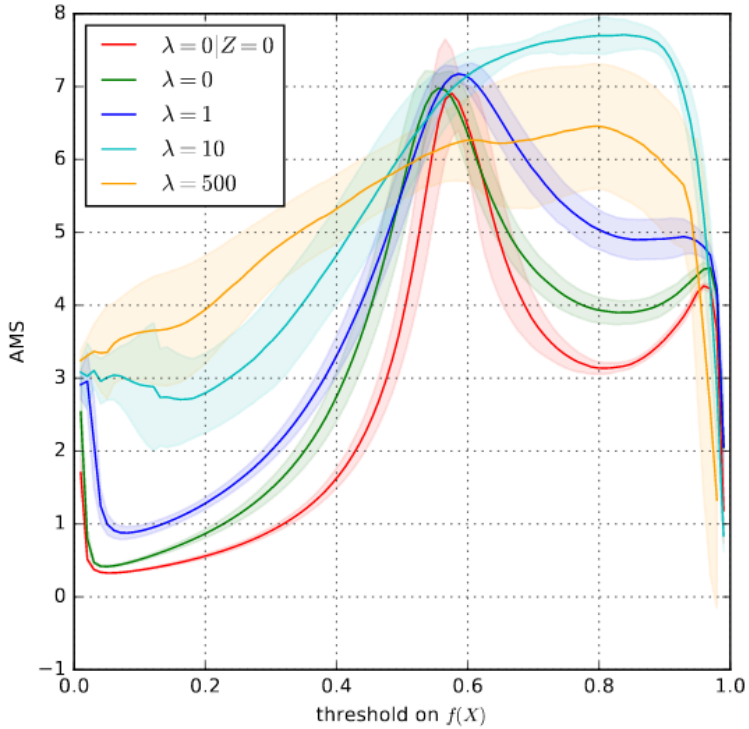
Optimizing both goals concurrently is impossible, so introduce weighting parameter:

$$L_{\text{tagger}} = L_{\text{classification}} - \lambda L_{\text{adversary}}$$

Paper Here

Reducing Pileup Dependence

- Can introduce nuisance parameter representing pileup
  - First study is discretized: Z=0 for no pileup Z=1 for 50
- Primary task: distinguishing W jets from QCD jets
- Adversary task: predicting Z from primary output

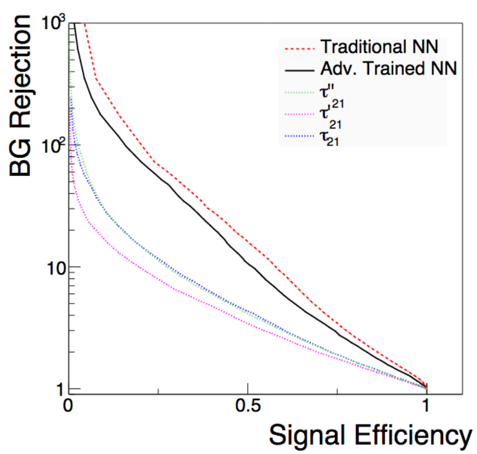


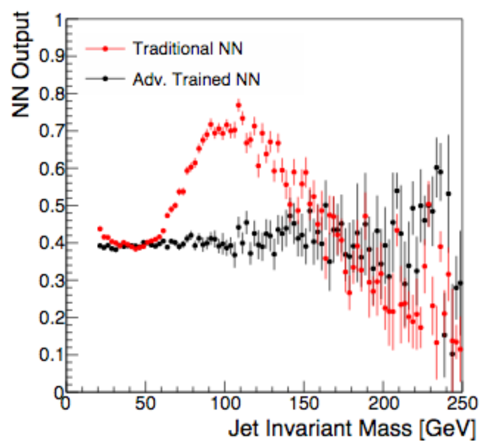
Trading classification accuracy for robustness to pileup increases final significance

Jet Mass Decorrelation

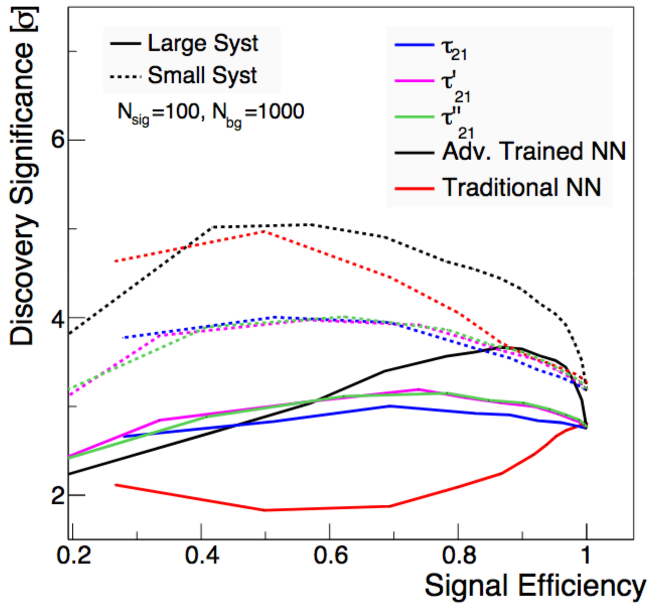
Paper Here

- Many jet tagging procedures distort jet mass distribution
  - Increases uncertainty in background modeling
  - Decreases significance of final results
- Primary task: distinguish W jets from QCD jets
- Adversary task: reproduce jet mass from primary output





ANN less efficient than regular NN, <u>but</u> also not mass dependent



ANN provides better final significance!

Recurrent Neural Networks

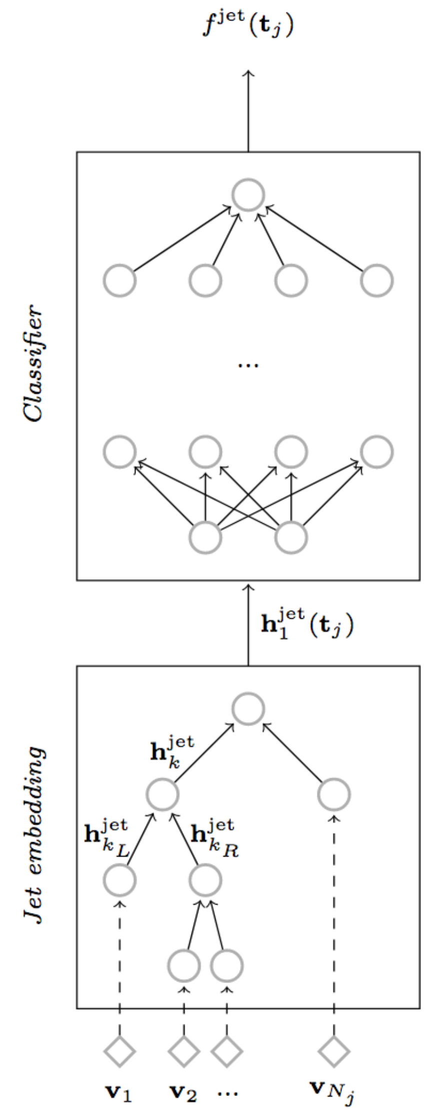
Recurrent Neural Networks

- RNNs take in time ordered data
- Basic unit is a cell with some internal state
  - Initial state is 0
  - At each training step, a new event is fed in and combined with the current internal state
  - Combination rules are learned during training
- Allow for embedding variable length information into a fixed length space while maintaining information from ordering
  - The output embedded vector can then be fed to a classifier

Paper Here

RNN for Jet ID: Concept

- RNNs widely used for language processing, can extend this to jet construction:
  - The particles in a jet should follow some order determined by QCD
  - 4 momentum of particles are the 'words' and the ordered clustering into jets are the 'sentences'



Ordered jets are embedded into a binary tree, weights of the tree are learned by the RNN (bottom up)

$$\mathbf{h}_k^{\text{jet}} = \begin{cases} \mathbf{u}_k & \text{if } k \text{ is a leaf} \\ \sigma\left(W_h \begin{bmatrix} \mathbf{h}_{k_L}^{\text{jet}} \\ \mathbf{h}_{k_R}^{\text{jet}} \\ \mathbf{u}_k \end{bmatrix} + b_h \right) & \text{otherwise} \end{cases}$$

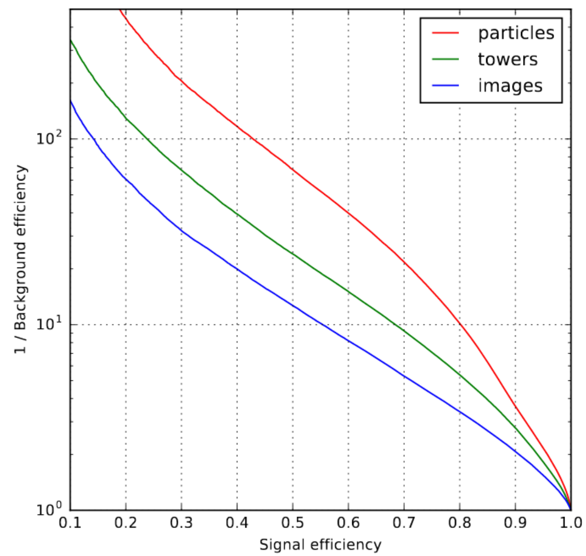
$$\mathbf{u}_k = \sigma\left(W_u g(\mathbf{o}_k) + b_u \right)$$

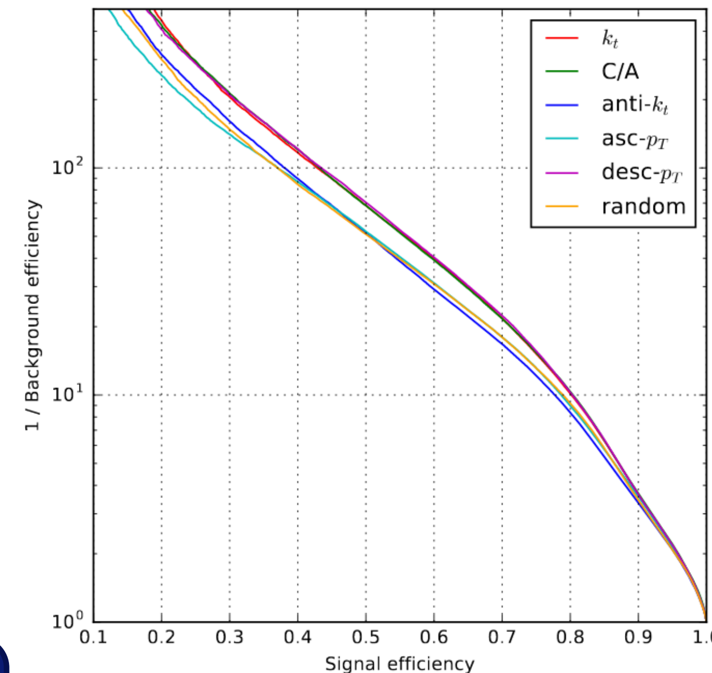
$$\mathbf{o}_k = \begin{cases} \mathbf{v}_{i(k)} & \text{if } k \text{ is a leaf} \\ \mathbf{o}_{k_L} + \mathbf{o}_{k_R} & \text{otherwise} \end{cases}$$

RNN for Jet ID: Results

- Applied to distinguishing W jets from QCD jets
- Looked at using information from pre-processed images and raw p_T information calo towers or individual particles

Input	Architecture	ROC AUC
	Projected into images	
towers	MaxOut	**0.8418**
towers	k_t	0.8321 ± 0.0025
towers	k_t (gated)	0.8277 ± 0.0028
	Without image preprocessing	
towers	τ_{21}	0.7644
towers	mass + τ_{21}	0.8212
towers	k_t	0.8807 ± 0.0010
towers	C/A	0.8831 ± 0.0010
towers	anti-k_t	0.8737 ± 0.0017
towers	asc-p_T	0.8835 ± 0.0009
towers	desc-p_T	**0.8838 ± 0.0010**
towers	random	0.8704 ± 0.0011
particles	k_t	0.9185 ± 0.0006
particles	C/A	**0.9192 ± 0.0008**
particles	anti-k_t	0.9096 ± 0.0013
particles	asc-p_T	0.9130 ± 0.0031
particles	desc-p_T	0.9189 ± 0.0009
particles	random	0.9121 ± 0.0008





Best RNNs ~ MaxOut with images, but faster and easier to train

Better with particle and towers than images → information lost in images

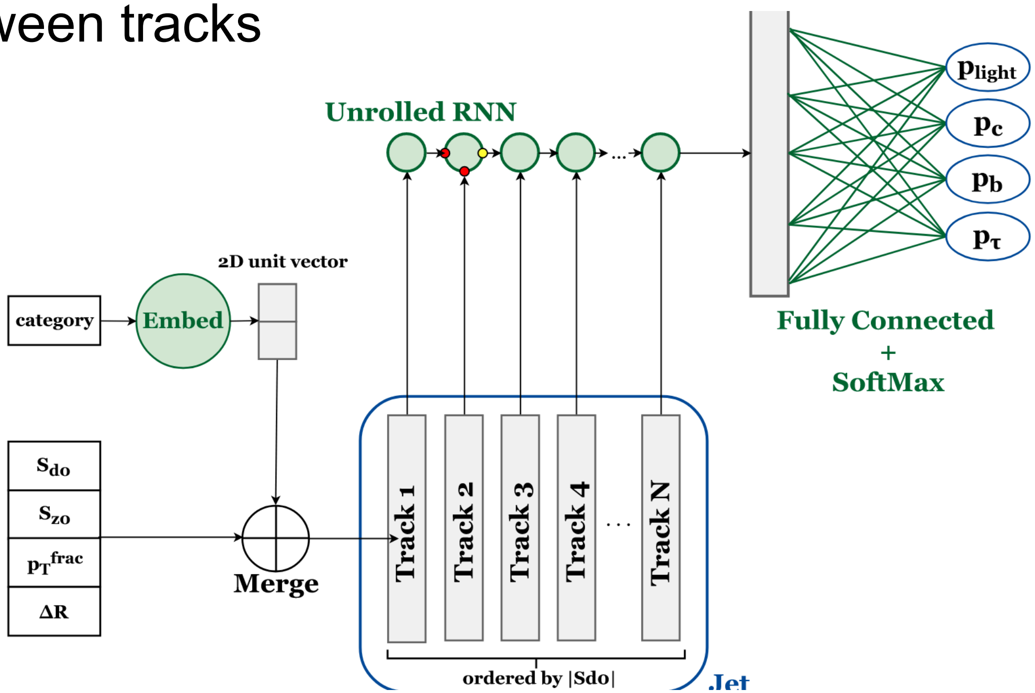
The algorithm used to order matters: k_T and desc-p_T best

RNN for B-tagging: Concept

Paper Here

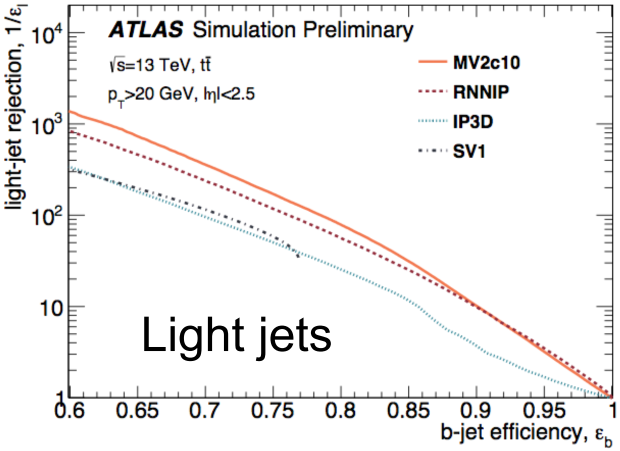
- Current b-tagging uses impact parameter (IP) information from tracks and secondary vertex information
  - Combined in a BDT for final application
- Current IP algorithm (IP3D) applies a LH to tracks to predict if they came from a certain flavor particle
  - Neglects correlations between tracks

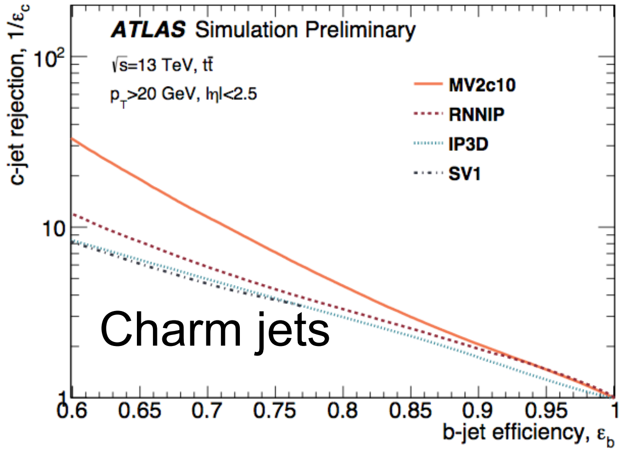
RNN application treats tracks as variable length sequence to embed



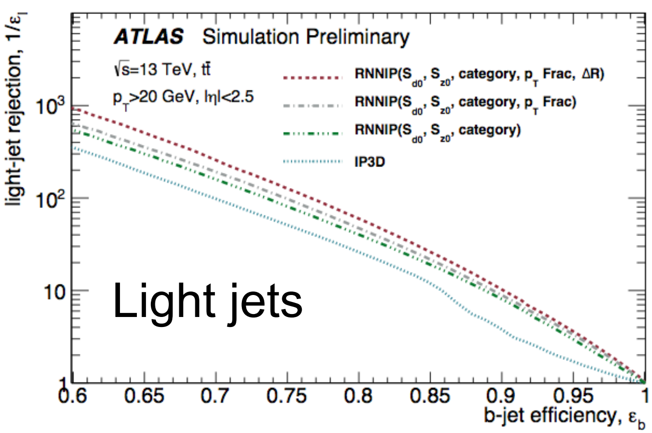
RNN for B-tagging: Results

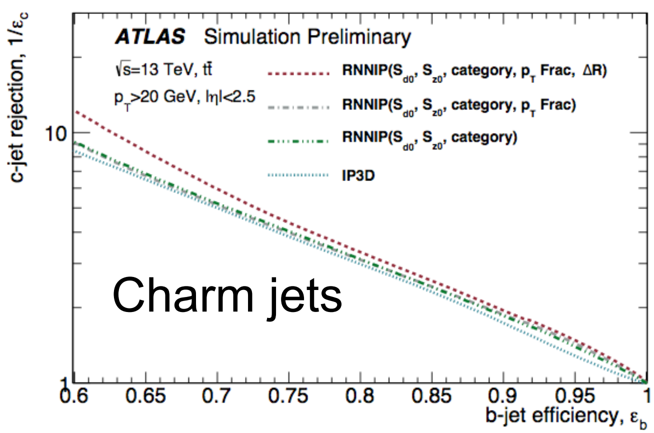
RNN outperforms IP3D, almost as well as combined BDT





Including substructure variables further improves RNN





RNN could replace IP3D and improve b-tagging accuracy!

Additional Studies

- Unsupervised mixture modeling and weakly labeld learning (improved quark vs gluon jets discrimination)
- Bonsai trees for triggers (improved accuracy and speed)
- DNNs for exotic particle searches (analysis classification)
- Reweighting/calibration with BDTs
- Studying parton shower modeling dependence in jet images and eliminating scale dependence
- Reinterpretation of LHC data for BSM searches based on theory parameters (what should the LHC events look like)
- Other particle IDs (taus, photons)
- Color studies with CNNs (additional information by separating energy contributions from different particle types)
- CNNs for EM Particle ID (my work!)
- LHC work summarized here

Conclusions

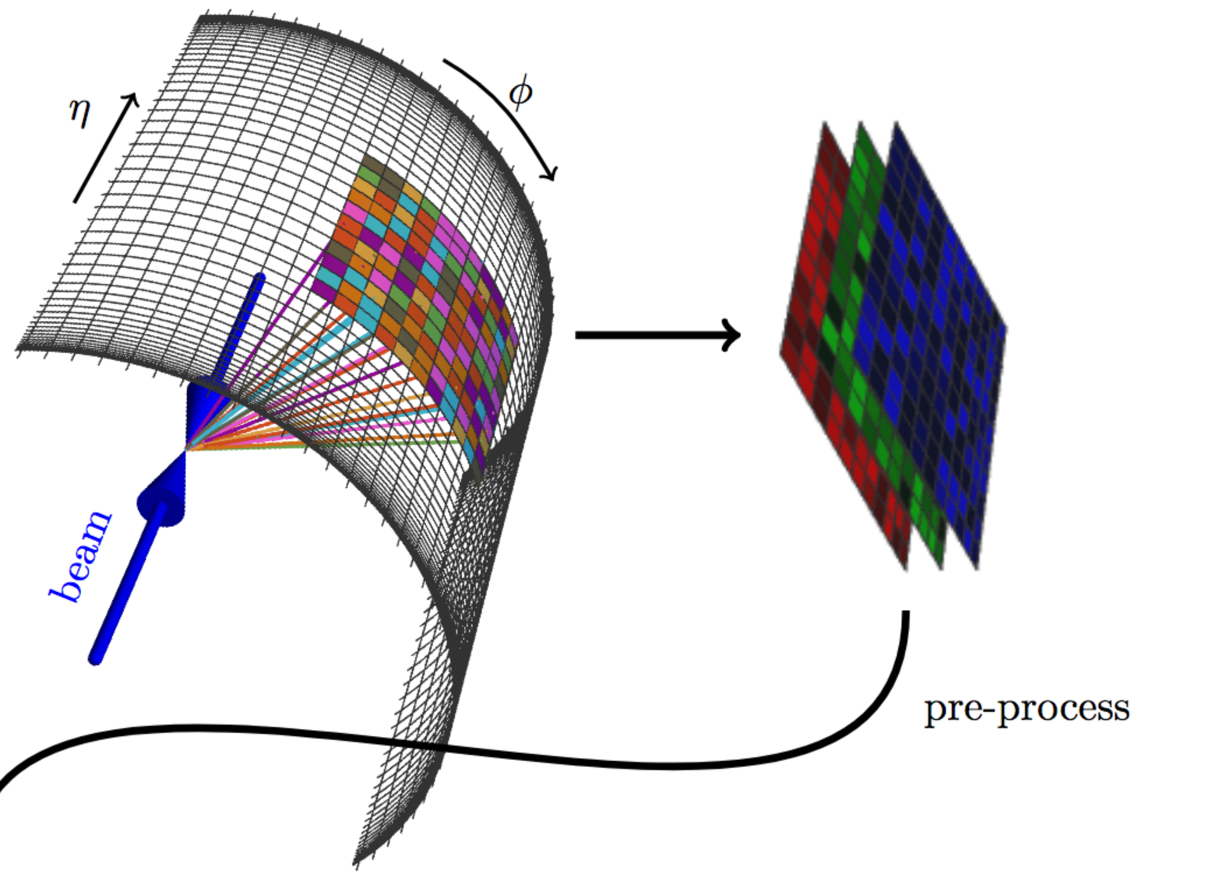
- Machine learning outperforms physics motivated techniques in many applications
- Can be applied to all stages of LHC physics
- Complexity of events and dependence on pileup will only increase as we move to HL LHC
  - Need to develop better triggers, taggers, and reduce pileup dependence
- Many exciting areas for continued research and collaboration with industry to use cutting edge ML techniques!

Backup

Variable Grouping in BDT Training

	W-Boson Tagging Observable Groups						
Observable	1	2	3	4	5	6	7 (BDT)
ECF_1			○	○	○	○	
ECF_2			○	○	○	○	○
ECF_3			○	○	○	○	○
C_2	○	○			○	○	
D_2	○	○			○	○	○
τ_1			○	○	○	○	○
τ_2			○	○	○	○	
τ_{21}	○	○			○	○	○
R_2^{FW}		○	○	○	○	○	○
$\mathcal{S}$		○	○	○	○	○	○
$\mathcal{P}$					○	○	○
$\mathcal{D}$					○	○	
a_3			○	○	○	○	○
A			○	○	○	○	○
T_{MIN}		○		○		○	
T_{MAJ}		○		○		○	
Z_{CUT}					○	○	
μ_{12}					○	○	
$\sqrt{d_{12}}$	○	○	○		○	○	
$KtDR$					○	○	○

Projection onto Calo Towers

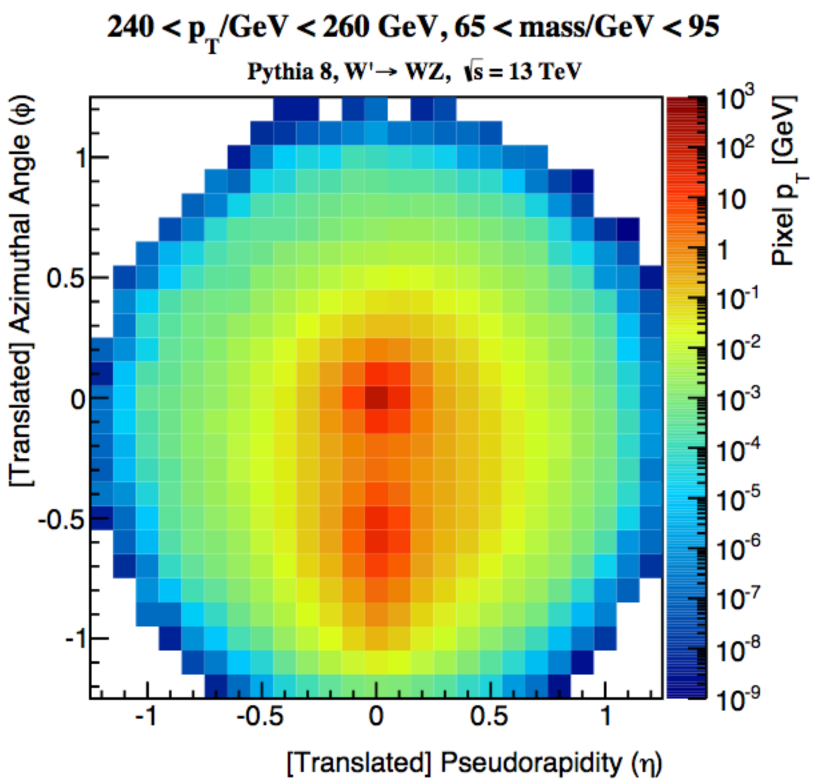


Paper Here

$W^{\pm}$ vs QCD Jet ID: Data

Want to separate boosted $W^{\pm}$ jets from QCD background

- Restricted study to 250-300 GeV jet p_T, and 65-95 GeV jet mass
- Images formed using calo-tower technique, 25x25 pixel images
- Pre-processed with translation, rotation, and parity flip

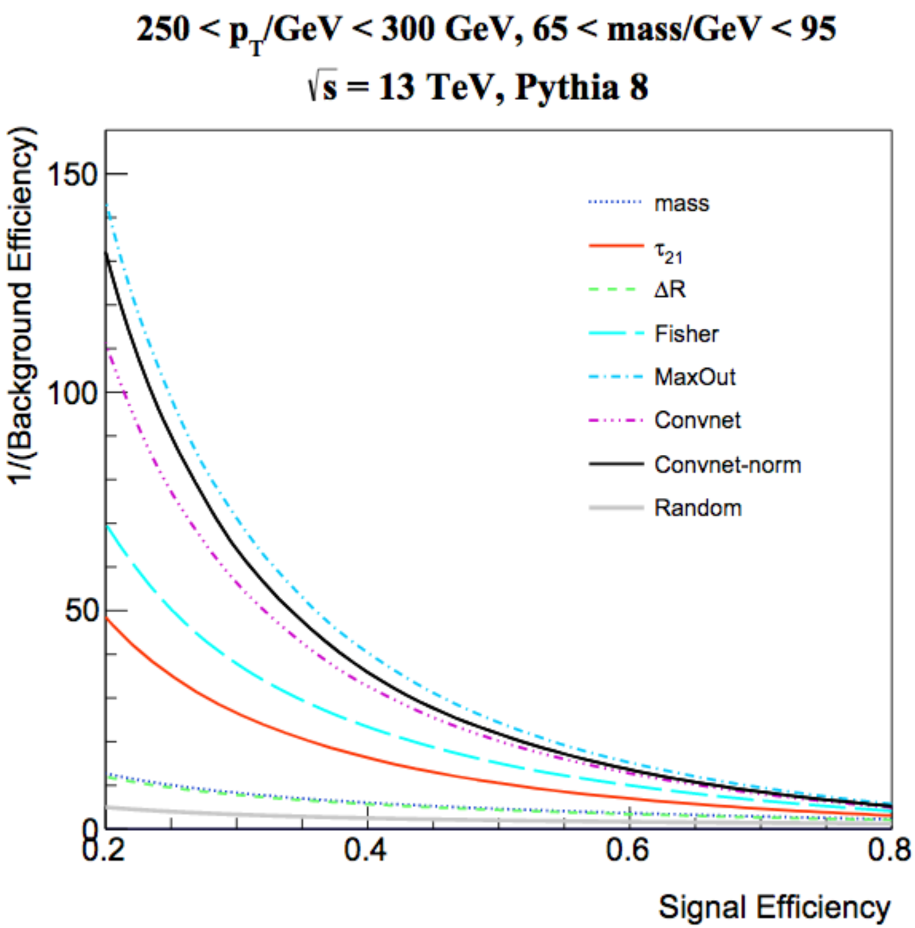


$W^{\pm}$ vs QCD Jet ID: Architecture

Compared performance of 2 network types:

1. CNN:
   - 3 convolution, max pooling, and dropout layer combinations
   - 11x11 kernels in first layer, 3x3 in other layers
   - 1 densely connected layer
   - Output layer of sigmoid classification
2. MaxOut:
   - 2 Maxout layers: value of node is max of all inputs
   - 2 fully connected layers
   - Output layer of sigmoid classification

$W^{\pm}$ vs QCD Jet ID: Results



DNNs outperform physics motivated variables!

Bjet track Correlations