Deep-Learned Top Taggers from Images & Lorentz Invariance

Anja Butter, Gregor Kasieczka, Tilman Plehn,
Michael Russel, Torben Schell

ML in Jet Physics Workshop
2017-12-12

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Top Tagging

- Identify hadronically decaying top quarks from QCD jets
- Mainly used for new physics searches & SM analyses
- p_T large enough so that the top quark is contained in one jet (boosted topology)
- Many QCD inspired variables available
  - jet mass = top mass
  - also find W mass inside jet
  - n-subjettiness (3-prong structure of the top)
- → *Top tagging is a well understood problem to gain an understanding of what DNNs learn*

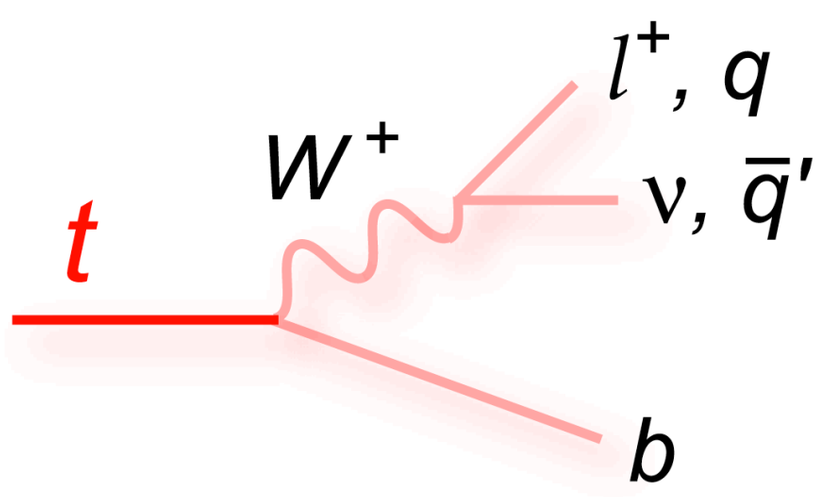


Image Based Top Tagging

Deep-learning Top Taggers or The End of QCD?
GK, T Plehn, M Russell, T Schell
JHEP 05 (2017) 006

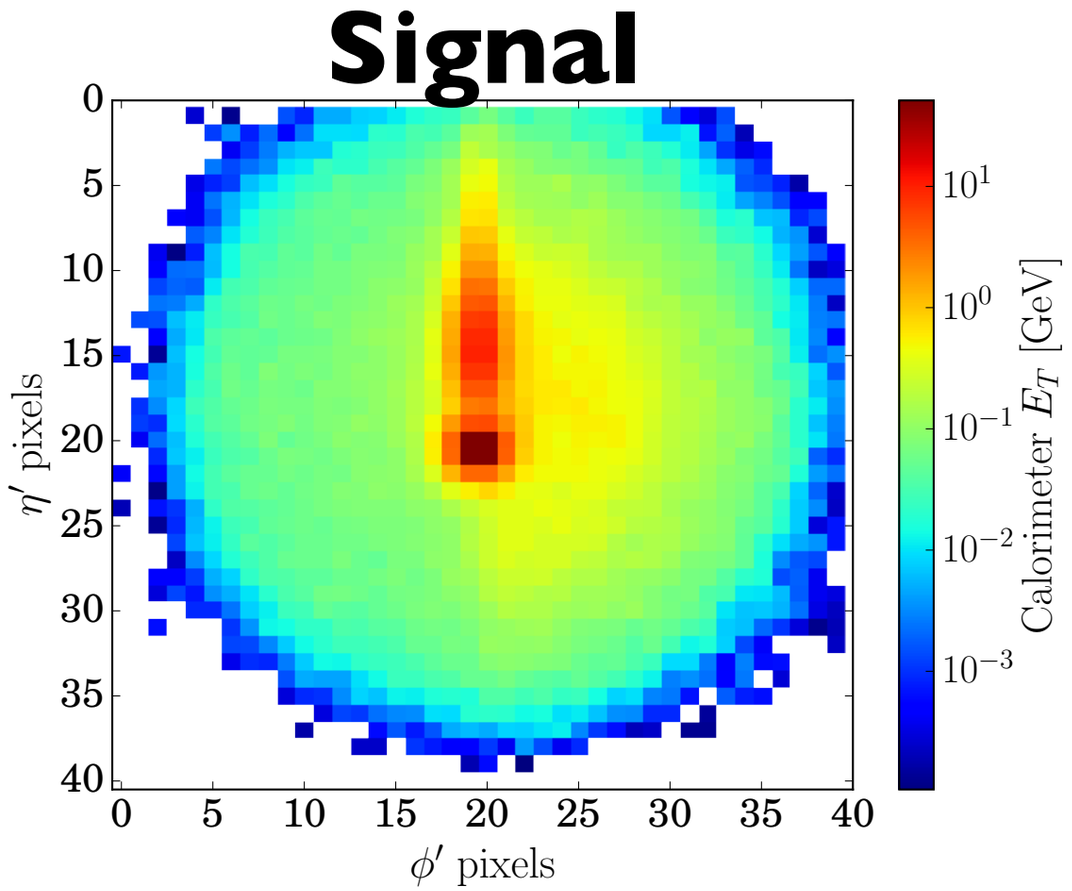
Image approach

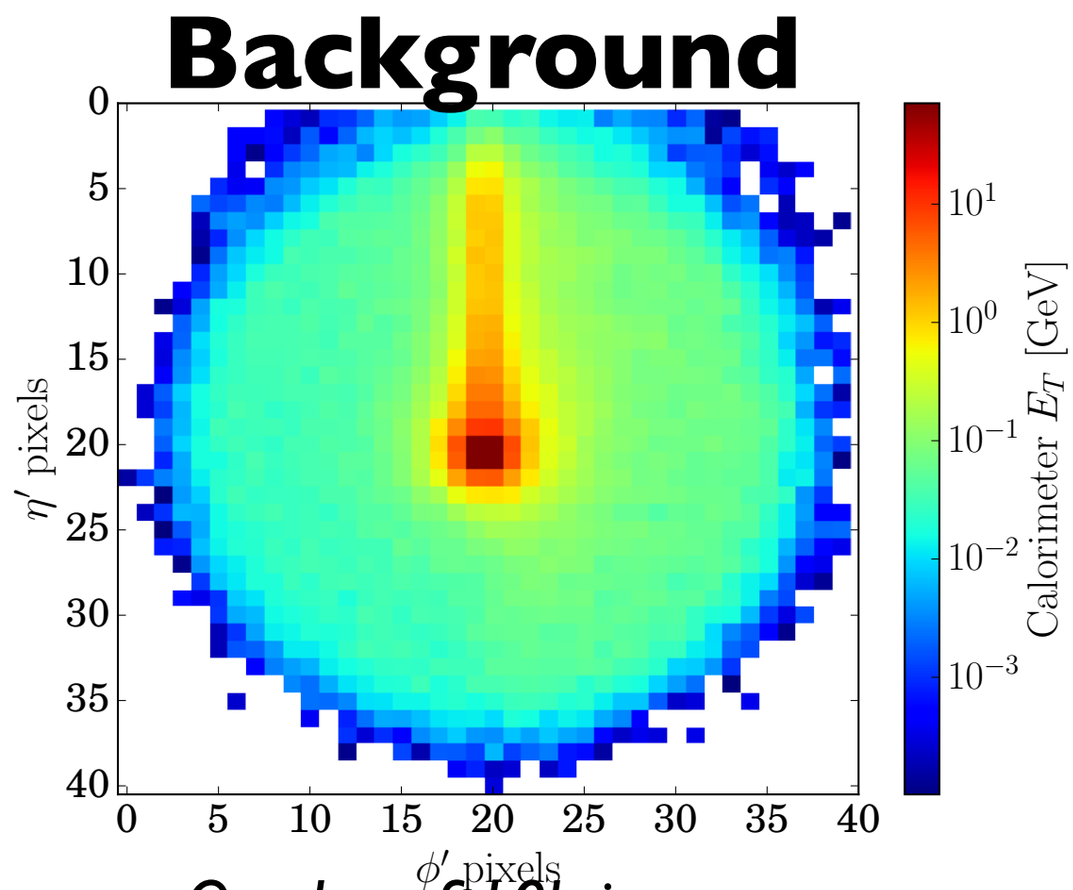
- Jets = 2d grayscale images:
  - 1 pixel = 0.1 in eta, 5 degree in phi
  - pixel energy: calorimeter ET
- Preprocessing (for illustration!)
  - Center maximum
  - Rotate so that second maximum is 12 o'clock
  - Flip so that third maximum is on the right side
  - Crop to 40x40 pixels

<u>*Origins:*</u>

Jet-Images: Computer Vision Inspired Techniques for Jet Tagging
J Cogan, M Kagan, E Strauss, A Schwartzman
arXiv:1407.5675

Jet-Images -- Deep Learning Edition
Ld Oliveira, M Kagan, L Mackey, B Nachman, A Schwartzman
JHEP 1607 069



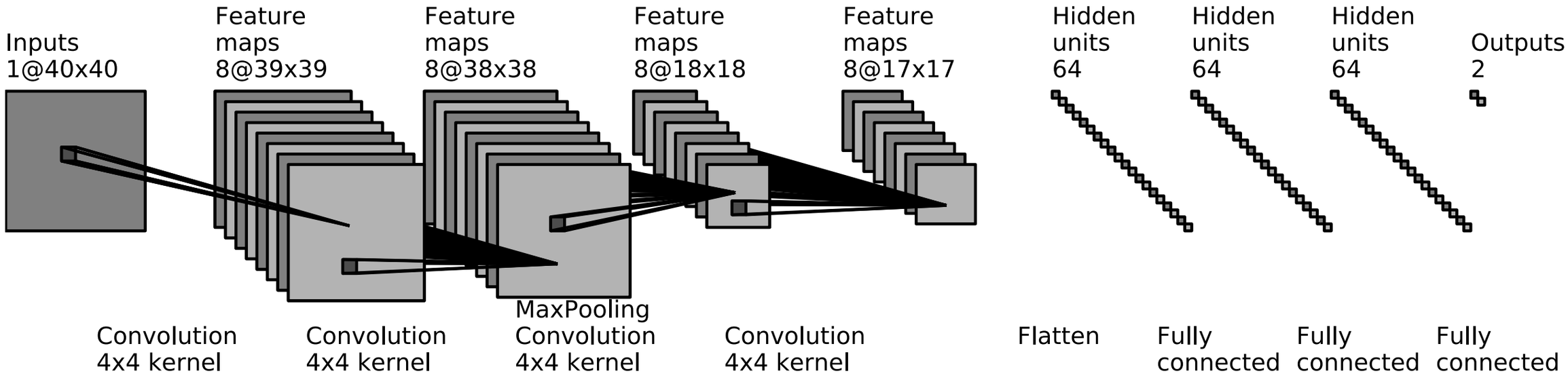


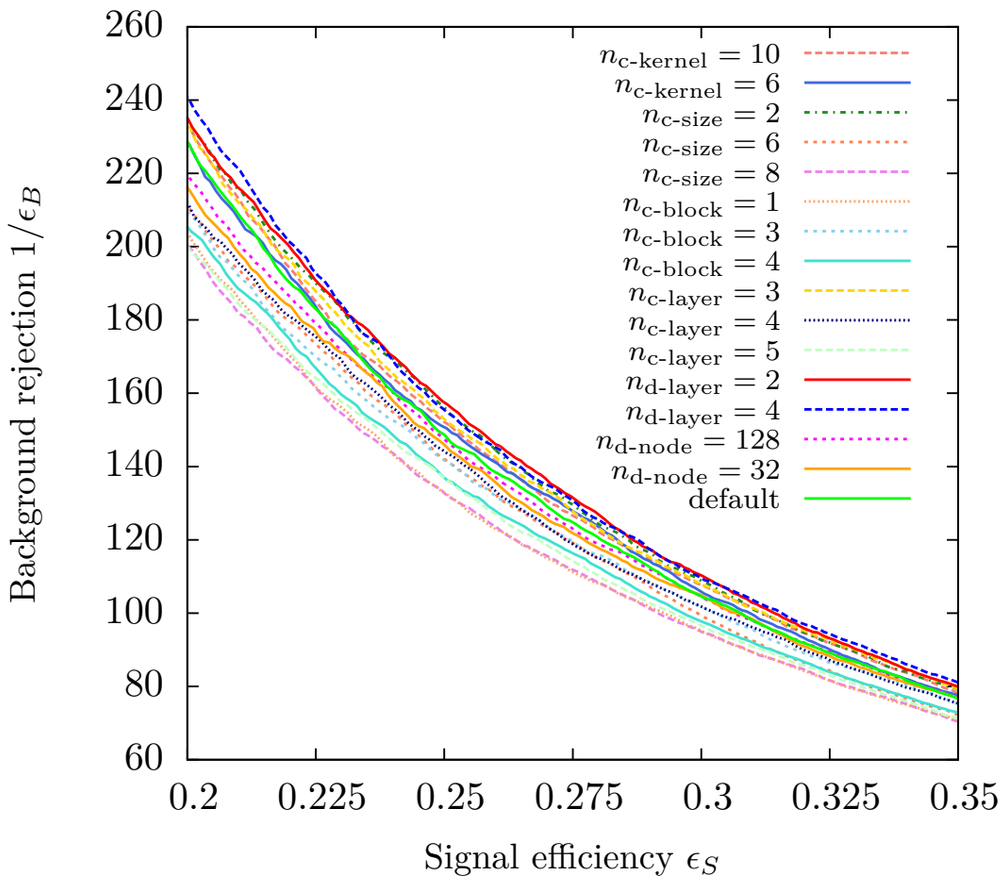
Overlay of 10k images

Technical interlude

- 14 TeV hadronic ttbar vs QCD, both simulated with Pythia 8
- Delphes 3 detector simulation
- Cluster with Anti-kT(R=1.5), recluster with Cambridge/Aachen (R=1.5)
- Jet p_T: 350..450 GeV, |eta| < 1.0
- Signal is truth matched to top within Delta R < 1.2
- Samples (signal+background):
  - 150k+150k for training
  - 150k+150k for checks during training
  - 300k+300k for final test

Network architecture





Iterative tuning of hyper parameters.

Performance

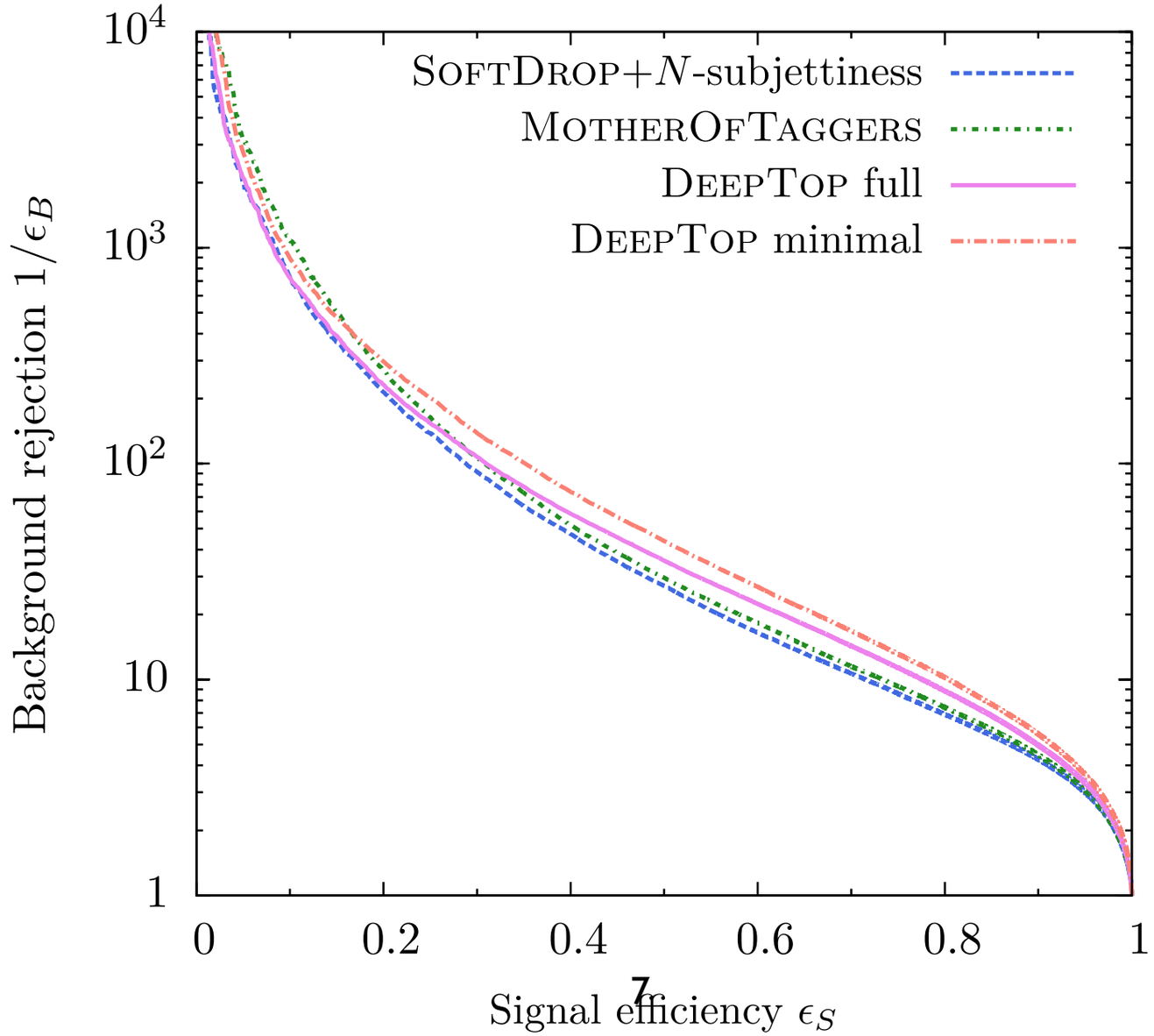
- Train a BDT on a set of standard tagging variables

SoftDrop + n-subjettiness:

$$\{ m_{\mathrm{sd}}, m_{\mathrm{fat}}, \tau_2, \tau_3, \tau_2^{\mathrm{sd}}, \tau_3^{\mathrm{sd}} \}$$

MotherOfTaggers:

$$\{ m_{\mathrm{sd}}, m_{\mathrm{fat}}, m_{\mathrm{rec}}, f_{\mathrm{rec}}, \Delta R_{\mathrm{opt}}, \tau_2, \tau_3, \tau_2^{\mathrm{sd}}, \tau_3^{\mathrm{sd}} \}$$



Technical Interlude II

- Training done on Piz Daint nodes equipped with *Nvidia Tesla P100 GPUs* at CSCS
- Also worked with Amazon cloud *Nvidia Tesla K80*
  (very easy and convenient, but they want money!)
- Network trains in O(some hours)
  - *Depends on N(events), network architecture, learning parameters. Can probably optimize a bit*
- Use
  - Keras with Theano back-end for DNN
  - scikit-learn for BDT
  - NumPy and Pandas for processing & storage

Beyond Images

Deep-learned Top Tagging from Lorentz Invariance
A Butter, GK, T Plehn, M Russell
arXiv: 1707.08966

Motivation

- Jet Images
  - straight-forward encoding of spatial relation
  - potential loss of information from pixelation
  - problem of sparse images
  - difficult to add more information* (tracking)
- Recursive networks**
  - can work directly with particles
  - need to learn relation between four-vectors

Deep learning in color: towards automated quark/gluon jet discrimination
PT Komiske, EM Metodiev, MD Schwartz
JHEP 1701

***QCD-Aware Recursive Neural Networks for Jet Physics*
G Louppe, K Cho, C Becot, K Cranmer
arXiv:1702.00748

Let's create a neural network architecture designed for particle physics!

Approach

Input is a p_T sorted list of Lorentz four-vectors:
(calo towers or particle flow objects)

$$k_{\mu,i} = \begin{pmatrix} E_0 & E_1 & \cdots & E_N \\ p_{x,0} & p_{x,1} & \cdots & p_{x,N} \\ p_{y,0} & p_{y,1} & \cdots & p_{y,N} \\ p_{z,0} & p_{z,1} & \cdots & p_{z,N} \end{pmatrix}$$

Combination Layer (**CoLa**): create linear combinations: $k_{\mu,i} \xrightarrow{\text{CoLa}} \widetilde{k}_{\mu,j} = k_{\mu,i}\, C_{ij}$

Lorentz Layer (**LoLa**): Use resulting matrix to extract physics features.
Main assumption is the Minkowski metric

Fully connected layers for final output

CoLa

- Goal: Allow network to reconstruct substructure axes (top, W, hard subjets, ..) by summing constituents
- (M - (N)) x N trainable weights

$$k_{\mu,i} \xrightarrow{\text{CoLa}} \widetilde{k}_{\mu,j} = k_{\mu,i}\, C_{ij}$$

$$C = \begin{pmatrix} 1 & 0 & \cdots & 0 & C_{1,N+2} & \cdots & C_{1,M} \\ 0 & 1 & & \vdots & C_{2,N+2} & \cdots & C_{2,M} \\ \vdots & \vdots & \ddots & 0 & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & C_{N,N+2} & \cdots & C_{N,M} \end{pmatrix}$$

Diagonal matrix (pass-through constituents)

trainable linear combinations

LoLa

- Transforms M Lorentz-vectors into M vectors with P components

$$\tilde{k}_j \xrightarrow{\text{LoLa}} \hat{k}_j = \begin{pmatrix} m^2(\tilde{k}_j) \\ p_T(\tilde{k}_j) \\ w_{jm}^{(E)}\ E(\tilde{k}_m) \\ w_{jm}^{(d)}\ d_{jm}^2 \end{pmatrix}$$

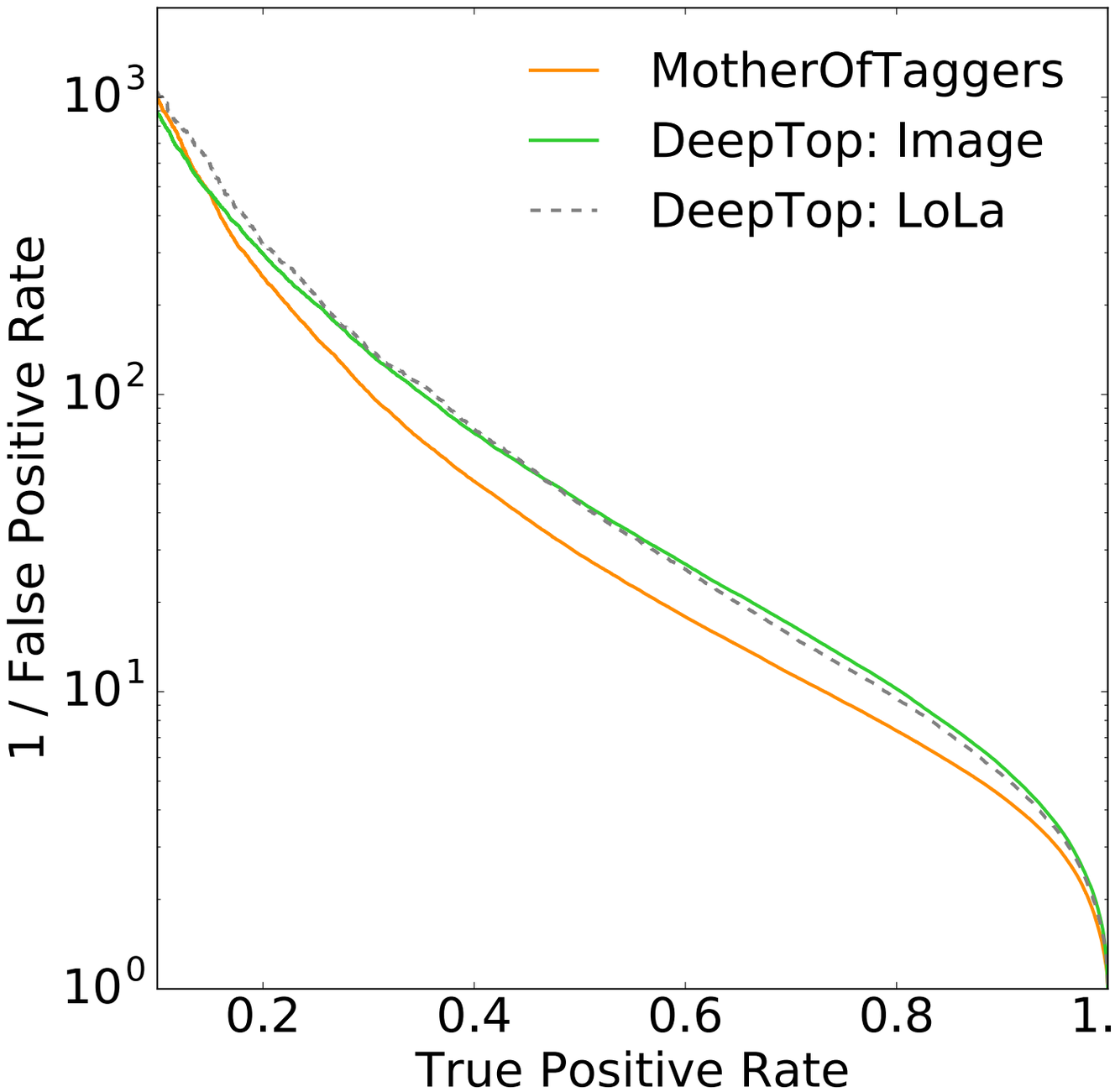
$$d_{jm}^2 = (\tilde{k}_j - \tilde{k}_m)_\mu\ g^{\mu\nu}\ (\tilde{k}_j - \tilde{k}_m)_\nu \qquad \eta_{\mu\nu} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Four copies:
2 sums over m
2 minima over m

Technical Interlude III

- Layers implemented in Python using Keras backend and raw Theano functions
- Simple fully connected network after LoLa:
  - Two fully connected layers with 100/50 nodes
- Approximately 200k signal and 200k background events for training
- Trained with Adam optimiser (learning rate: 0.001)
- Categorical cross entropy loss function

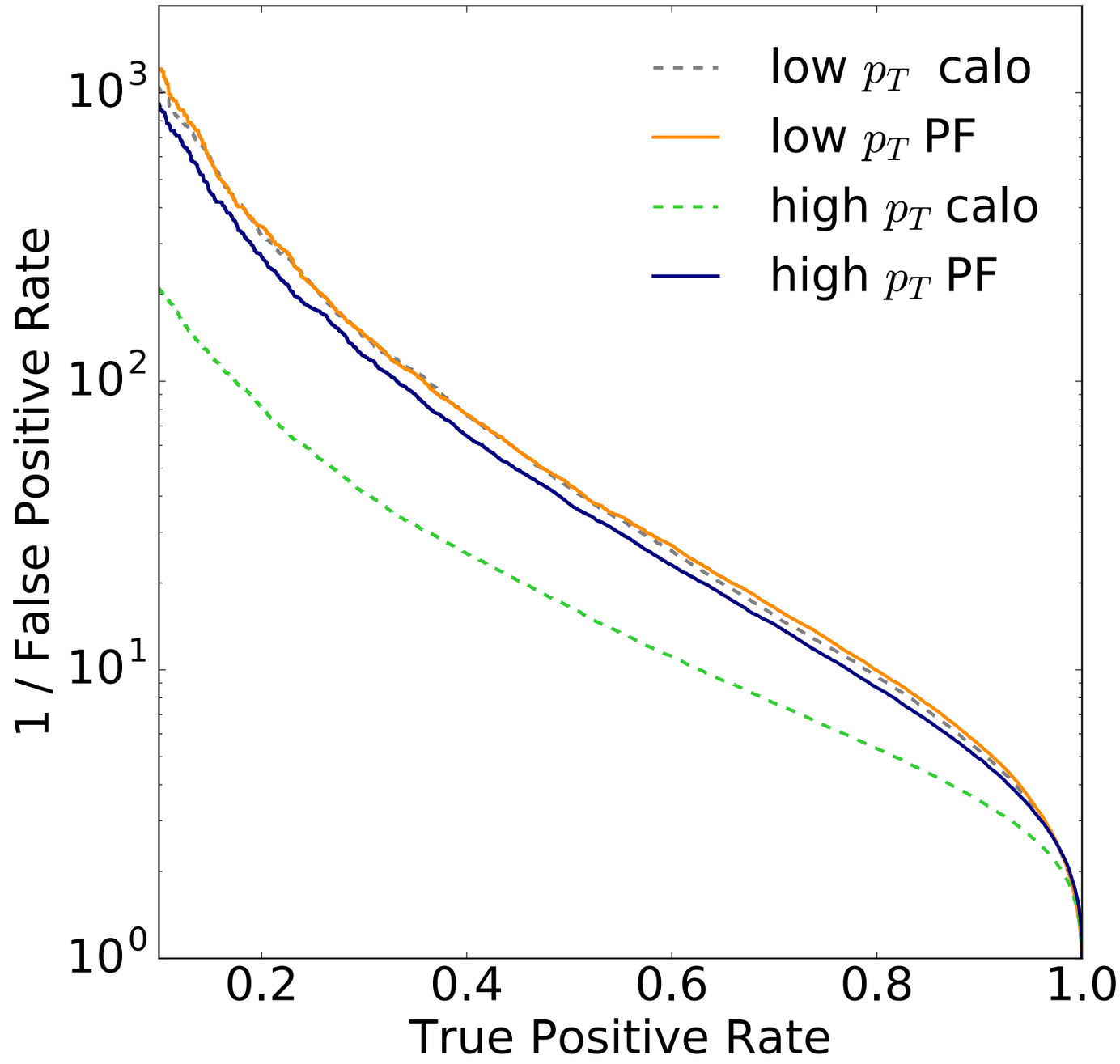
Does it work?



At low pT:
Comparable to image approach

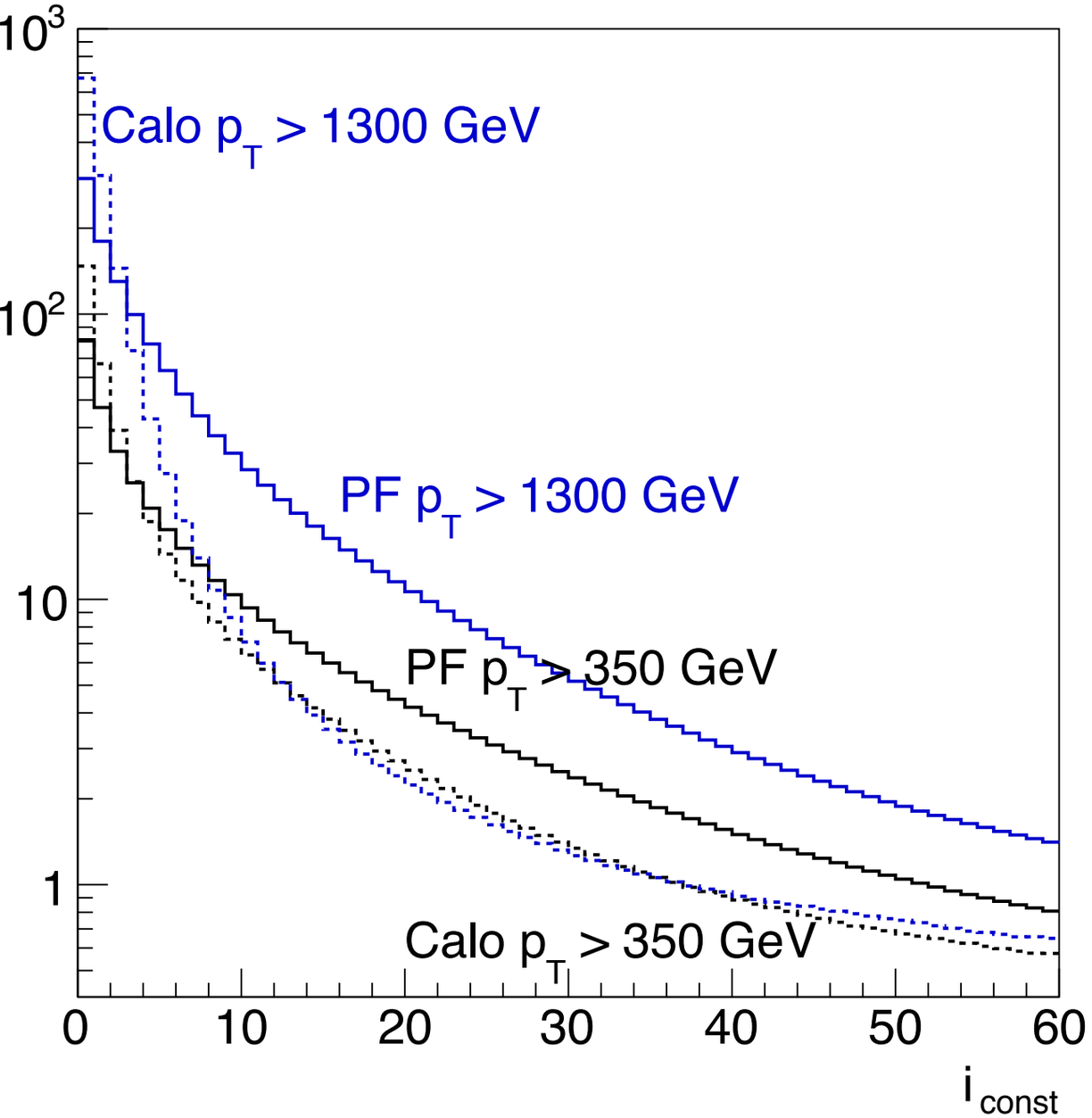
Resolution?

low pT: 350-450 GeV
high pT: 1300-1400 GeV



At high pT:
Clear gain from ability to use PF with tracking precision

Which Information?



# Constituents	Particle Flow AOC	Calo Tower AOC
10	0.9193(2)	0.9254(14)
20	0.9303(5)	0.9303(6)
30	0.9307(35)	0.9312(3)
40	0.9340(1)	0.9321(2)
50	0.9342(4)	0.9310(4)
60	0.9336(5)	0.9316(5)

(Low pT)

Learning the metric

- LoLa uses Minkowski metric
- Can make trainable with the Ansatz:

$$\mathrm{diag}(-1, 1, 1, 1) \rightarrow \mathrm{diag}(K, L, M, N)$$

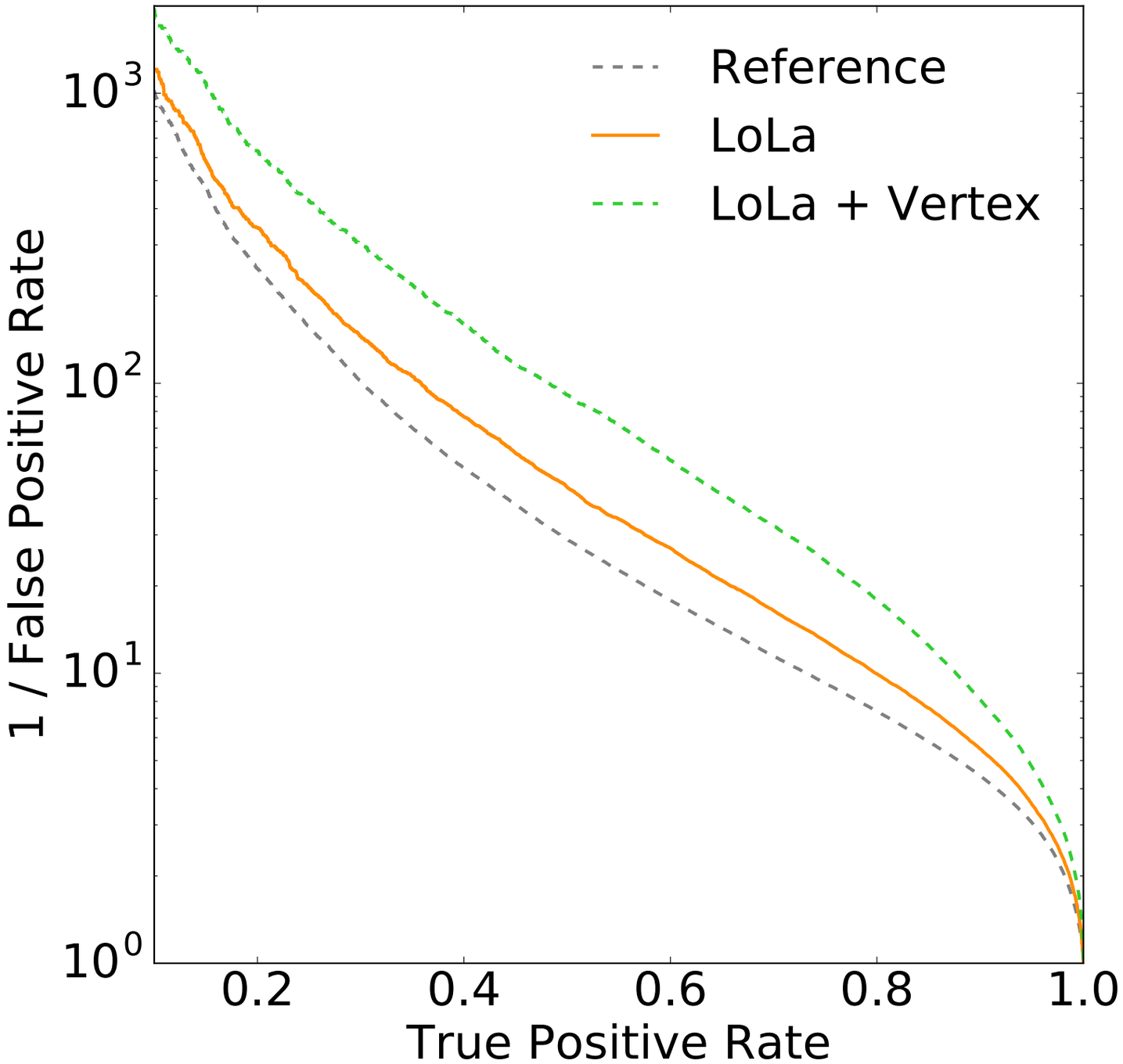
- Result: $g = \mathrm{diag}(\quad 0.99 \pm 0.02,$
  $-1.01 \pm 0.01, -1.01 \pm 0.02, -0.99 \pm 0.02)$

- We learn a +/- signature:
  - Difference between time and space-like coordinates understood!
- We learn similar absolute values:
  (overall normalisation does not matter)
  - Isotropy of space!

Where is the information?

- Replace pT ordering with random order:
  - 0.5% worse AOC
- Only mass
  - 24% worse AOC
- Only first three rows
  - 1.6% worse AOC
- Only last row
  - 2 sum, 2min: 0.3% worse AOC
  - 1 sum: 1.2% worse AOC

$$\tilde{k}_j \xrightarrow{\text{LoLa}} \hat{k}_j = \begin{pmatrix} m^2(\tilde{k}_j) \\ p_T(\tilde{k}_j) \\ w_{jm}^{(E)}\, E(\tilde{k}_m) \\ w_{jm}^{(d)}\, d_{jm}^2 \end{pmatrix}$$



$$\begin{pmatrix} E \\ p_x \\ p_y \\ p_z \end{pmatrix} \rightarrow \begin{pmatrix} E \\ p_x \\ p_y \\ p_z \\ v_x \\ v_y \\ v_z \end{pmatrix}$$

Vertex information successfully included in LoLa architecture

Caveat:

Result with Delphes vertex information for tracks.

Need to get a realistic resolution

Architecture

- Wide range of architectures used for top tagging:
  - **Fully connected** networks
  - **Image** recognition
  - **Natural language** processing
  - **Physics** based
  - **Hybrid**

- We should do a detailed comparison study on the same events to see what is learned where
- Need a large sample with well described b-tag information!

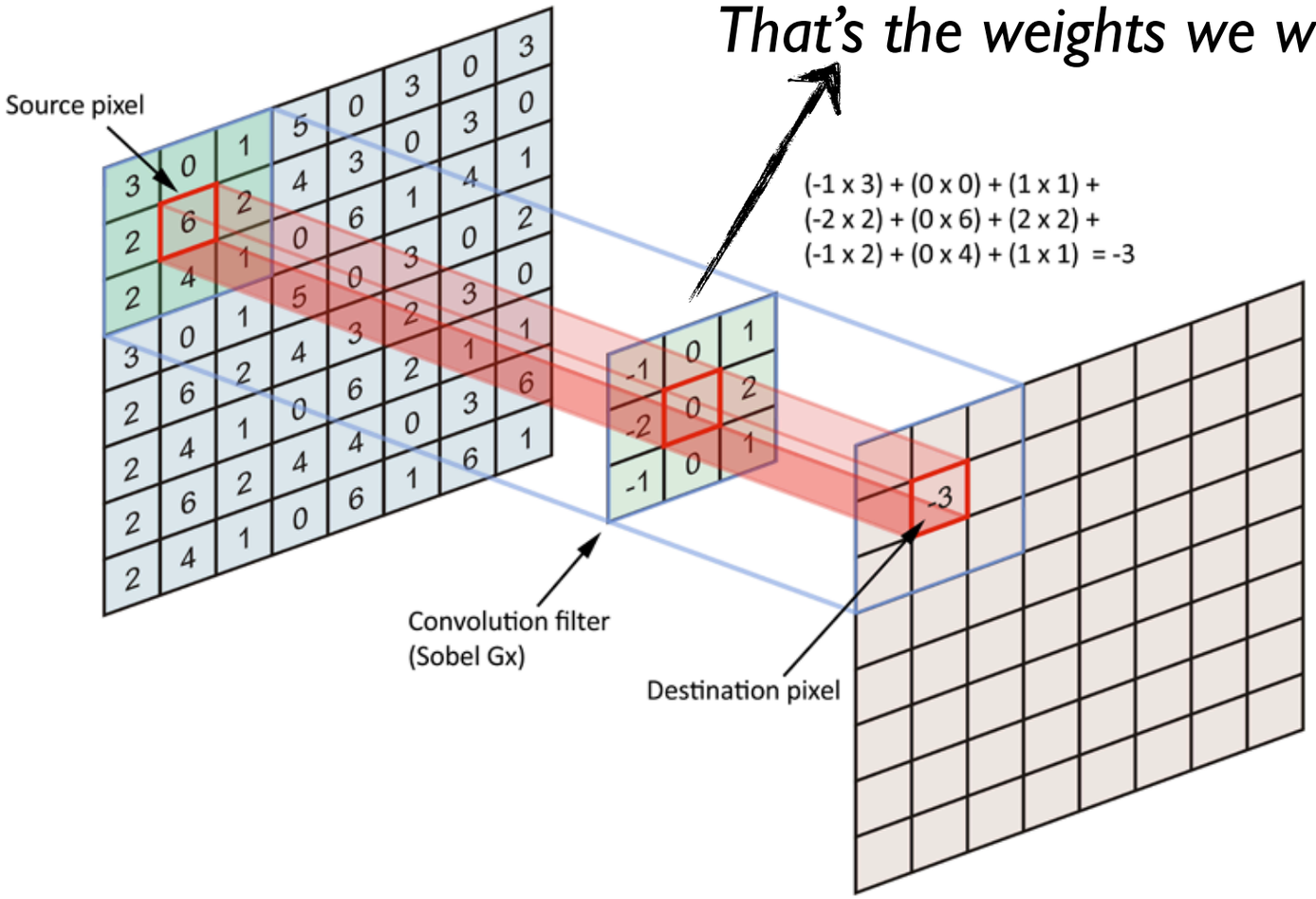
Closing

- Gain understanding into DNN reconstruction using top tagging:
  - Simple image based approach easily outperforms standard taggers
  - Preprocessing hurts performance
  - Probe what is learned
- Create novel, physics-motivated architecture:
  - Gives (*some of*) the learned weights a physical interpretation
  - Performance on par with image approach
  - No information loss due to preprocessing, can directly work with tracks
  - Can add extra properties (4-vector -> "N-vector")

Thank You!

Bonus Slides

Convolutional Network



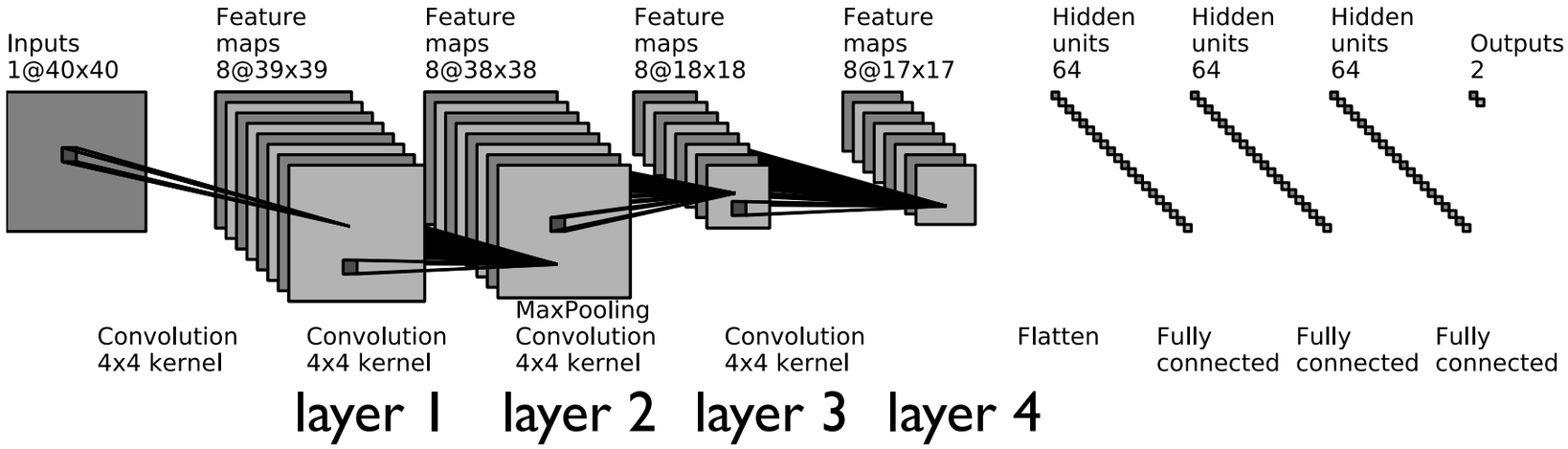
Fold a mask with the input to get output
What is learned are the parameters of the mask

Convolutional (conv) layer

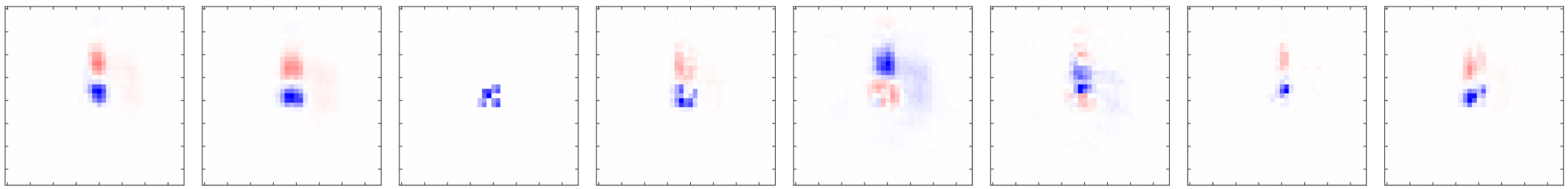
- ***How to build a convolutional network***
  - Chain multiple conv layers
  - Reduce image resolution in between
  - Use multiple masks per layer
  - Add linear ANN in the end

(This is still a network. We just use a fancy idea to decide which nodes to connect to each other)

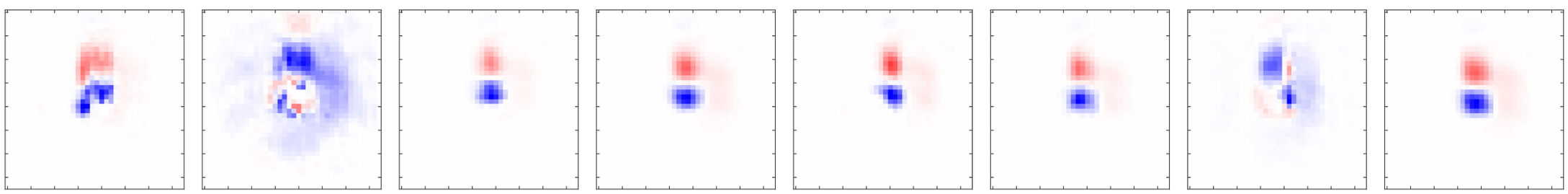
Intermediate output



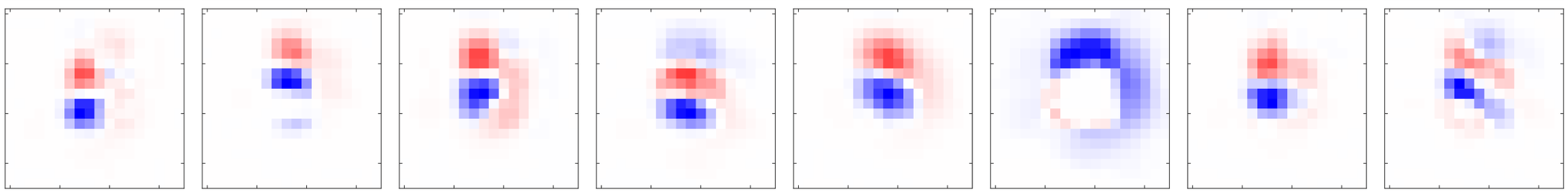
layer 1



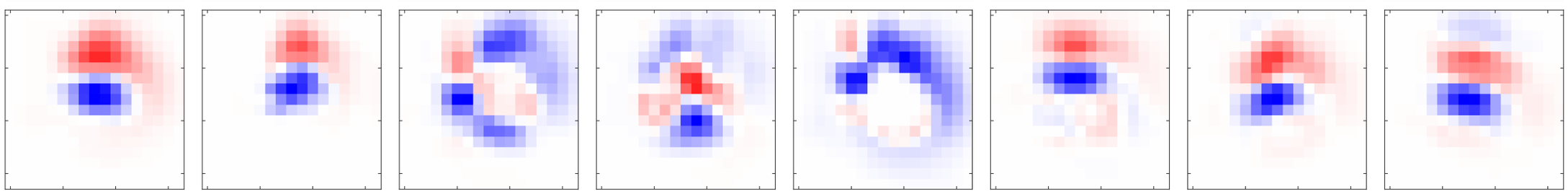
layer 2



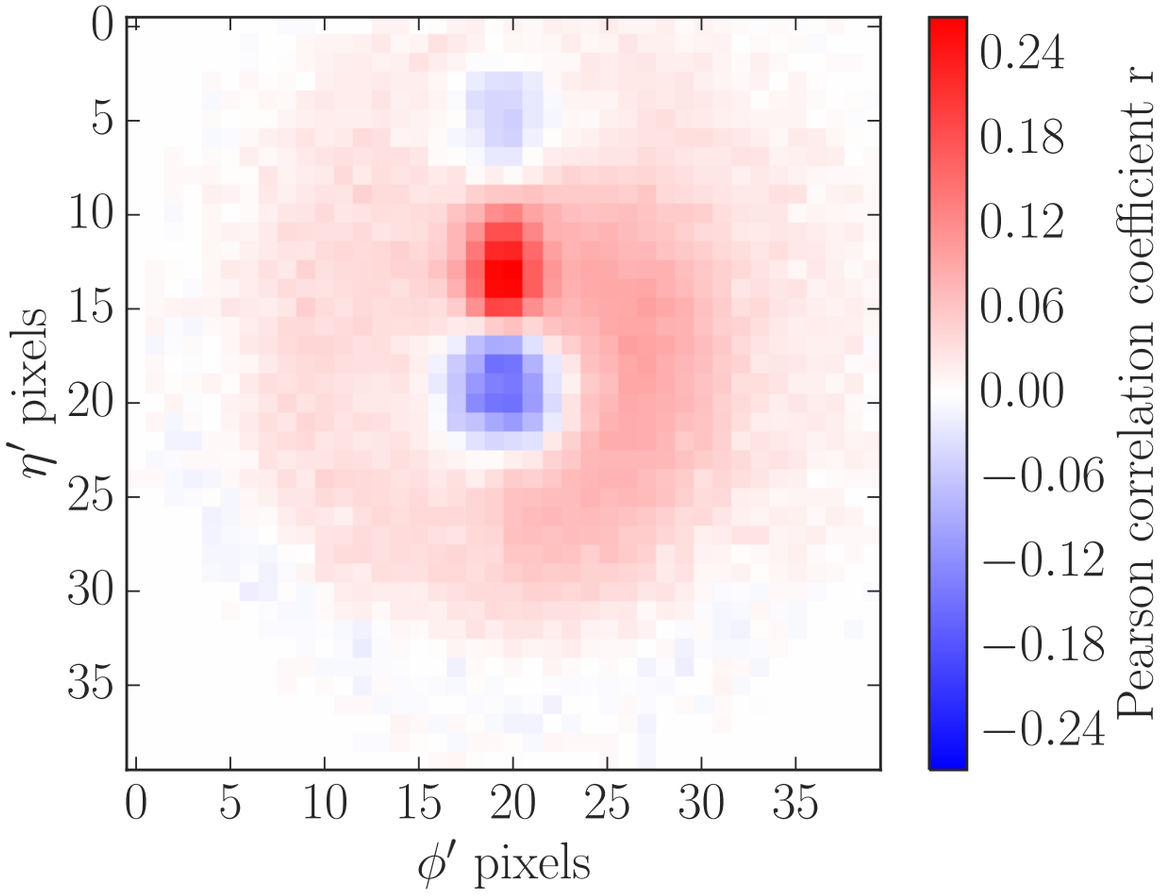
layer 3



layer 4



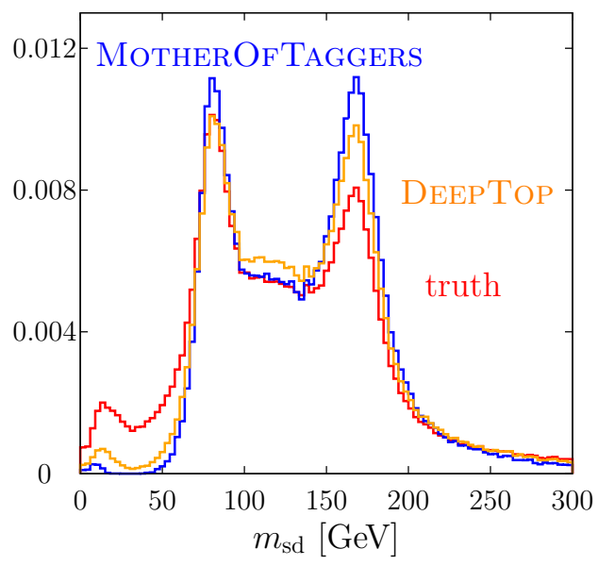
Pearson correlation coefficient



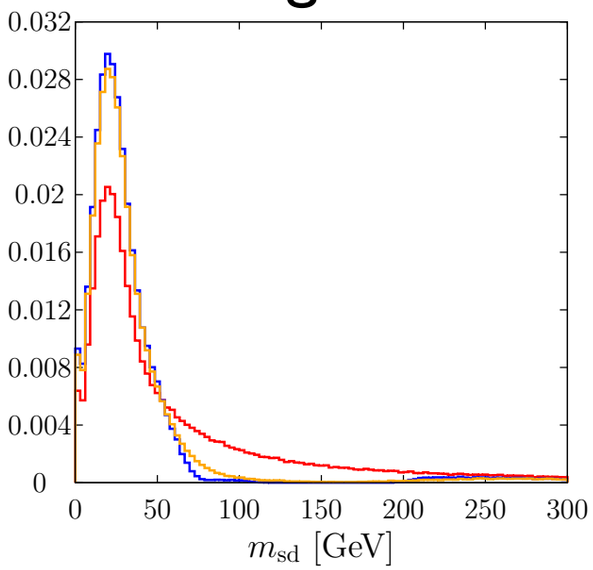
$$r_{ij} = \frac{\sum_{\text{images}} \left(x_{ij} - \bar{x}_{ij}\right)\left(y - \bar{y}\right)}{\sqrt{\sum_{\text{images}} \left(x_{ij} - \bar{x}_{ij}\right)^2} \sqrt{\sum_{\text{images}} \left(y - \bar{y}\right)^2}}$$

Sliced Masses

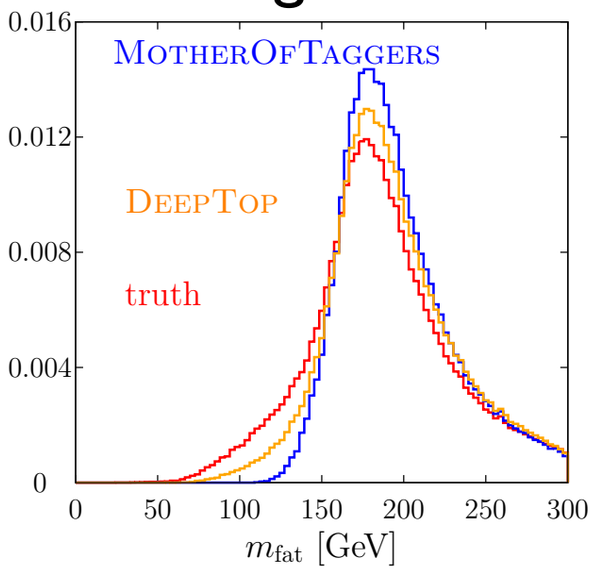
Signal



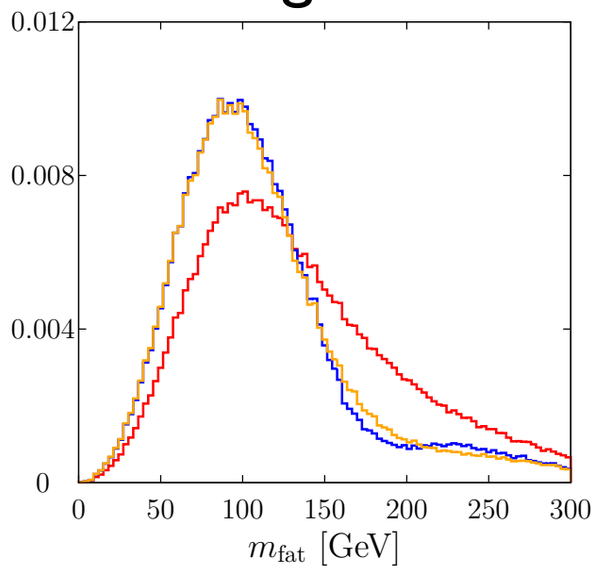
Background



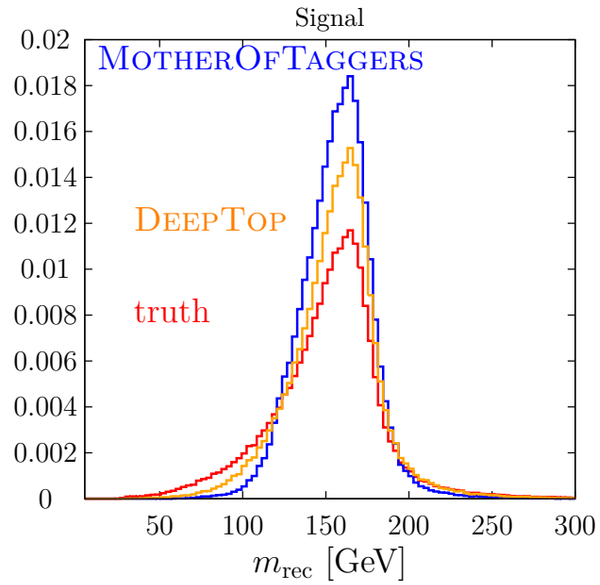
Signal



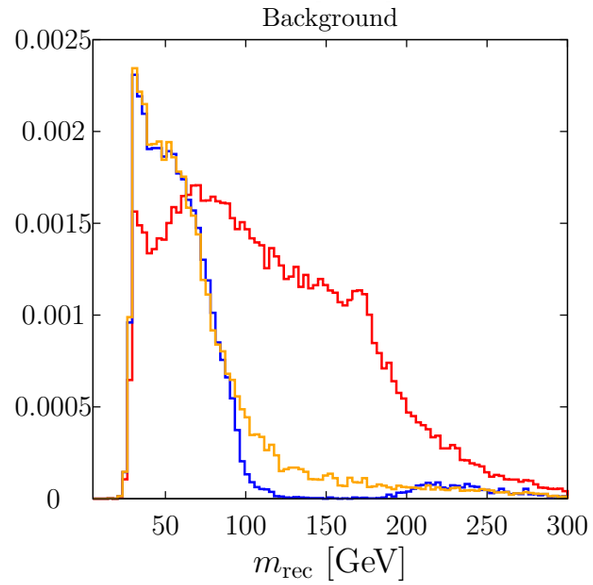
Background



Signal



Background



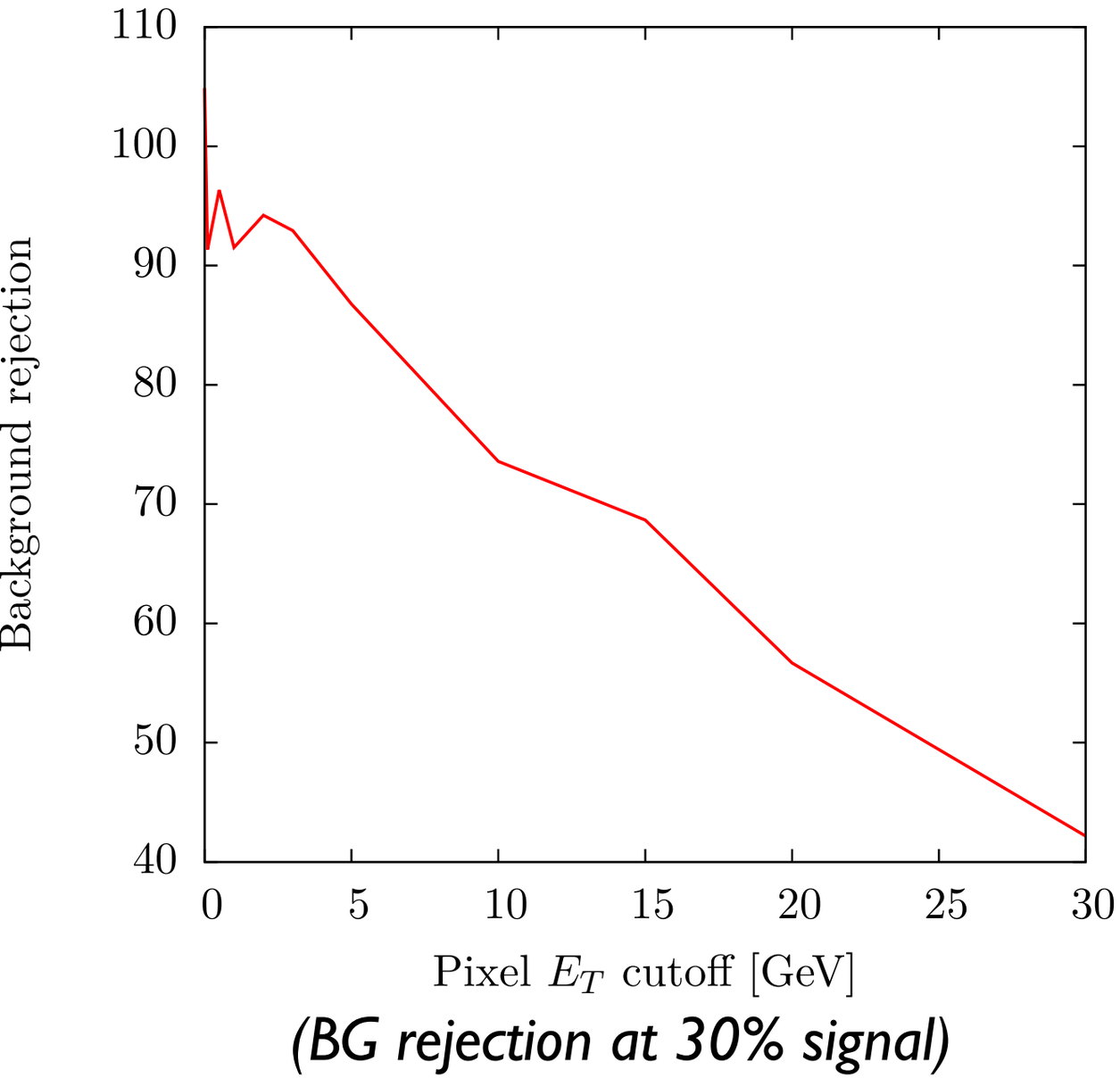
Compare:

Selection on truth

Selection using BDT (<0.2 / >0.8)

Selection using DNN (<0.2 / >0.8)

Pixel threshold

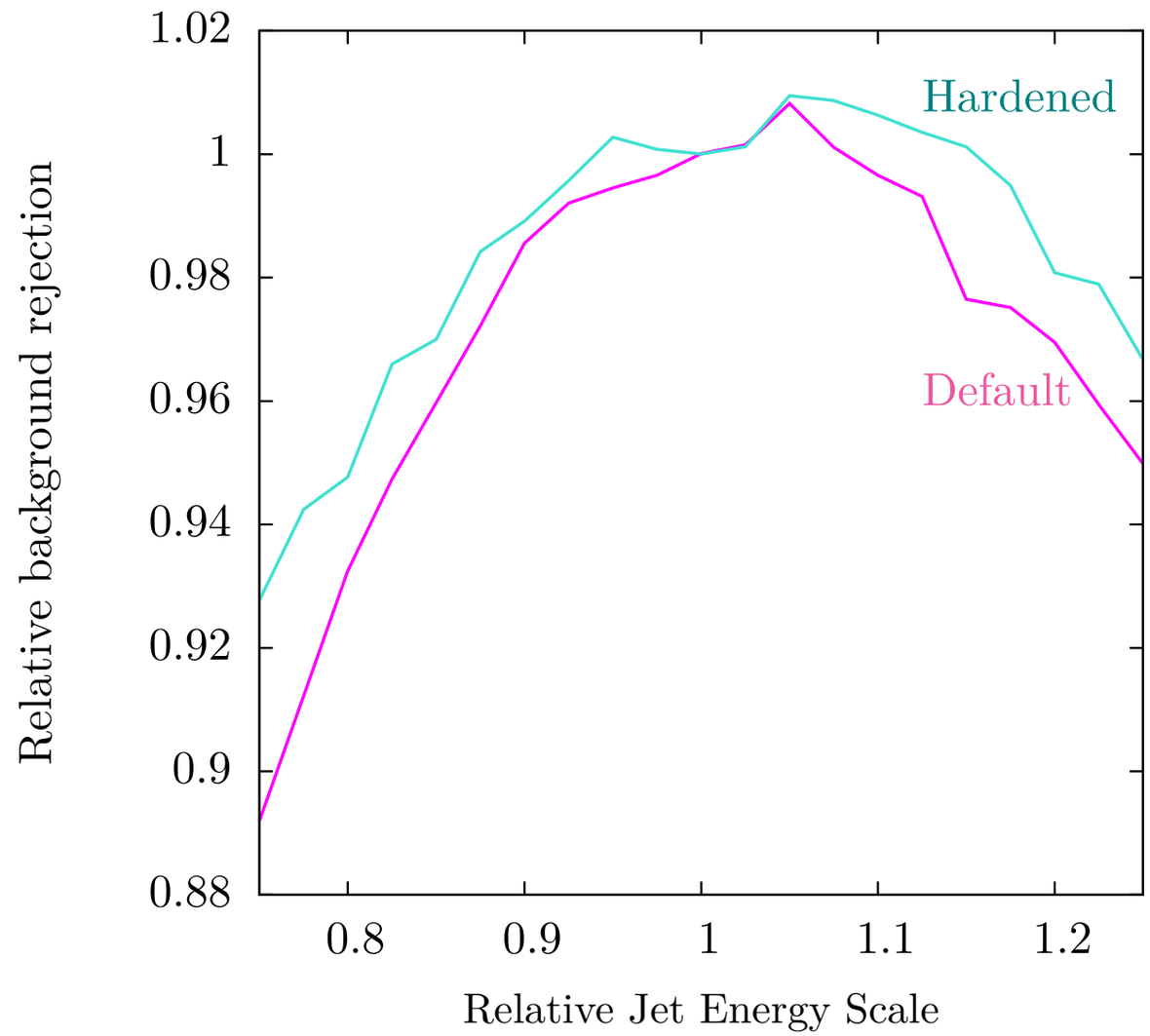


(BG rejection at 30% signal)

- Train and test network on images with higher pixel threshold
(and set values below threshold to zero)
- We are not sensitive to very soft information (this is good)

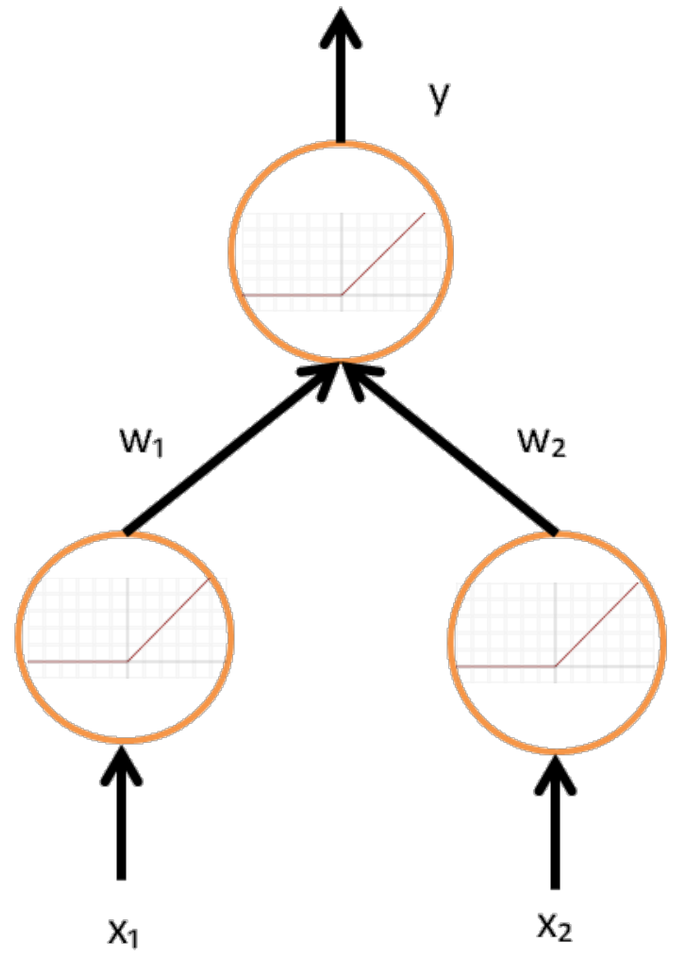
Detector effects

- *Default*
  - Train network on JES=1.0
  - Test network on images with JES scaled up/down
- *Hardened*
  - Train network while randomly smearing JES
  - Test network on images with JES scaled up/down



Control what the network is sensitive to

A Very Simple Network



$$y = f(f(x_1)w_1 + f(x_2)w_2)$$

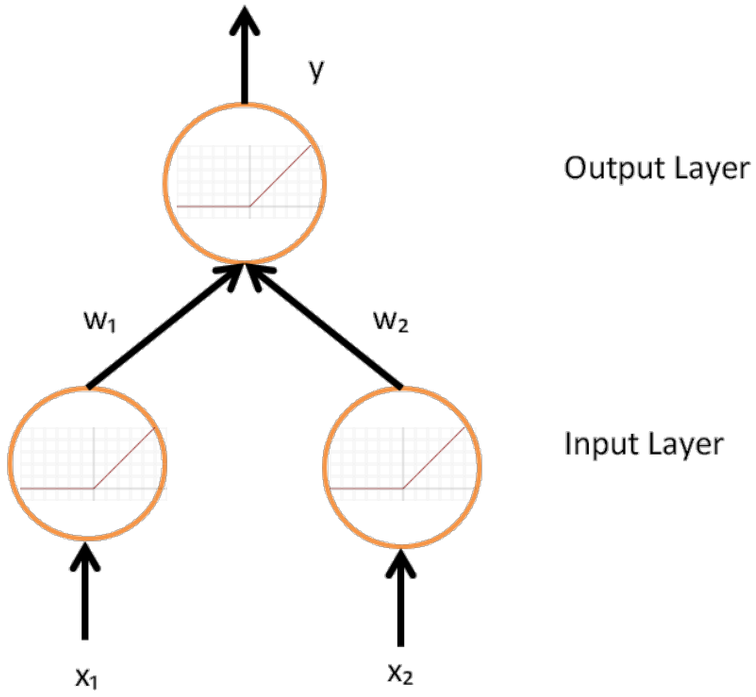
$$f(x) = \Theta(x) \cdot x$$

How do networks learn?

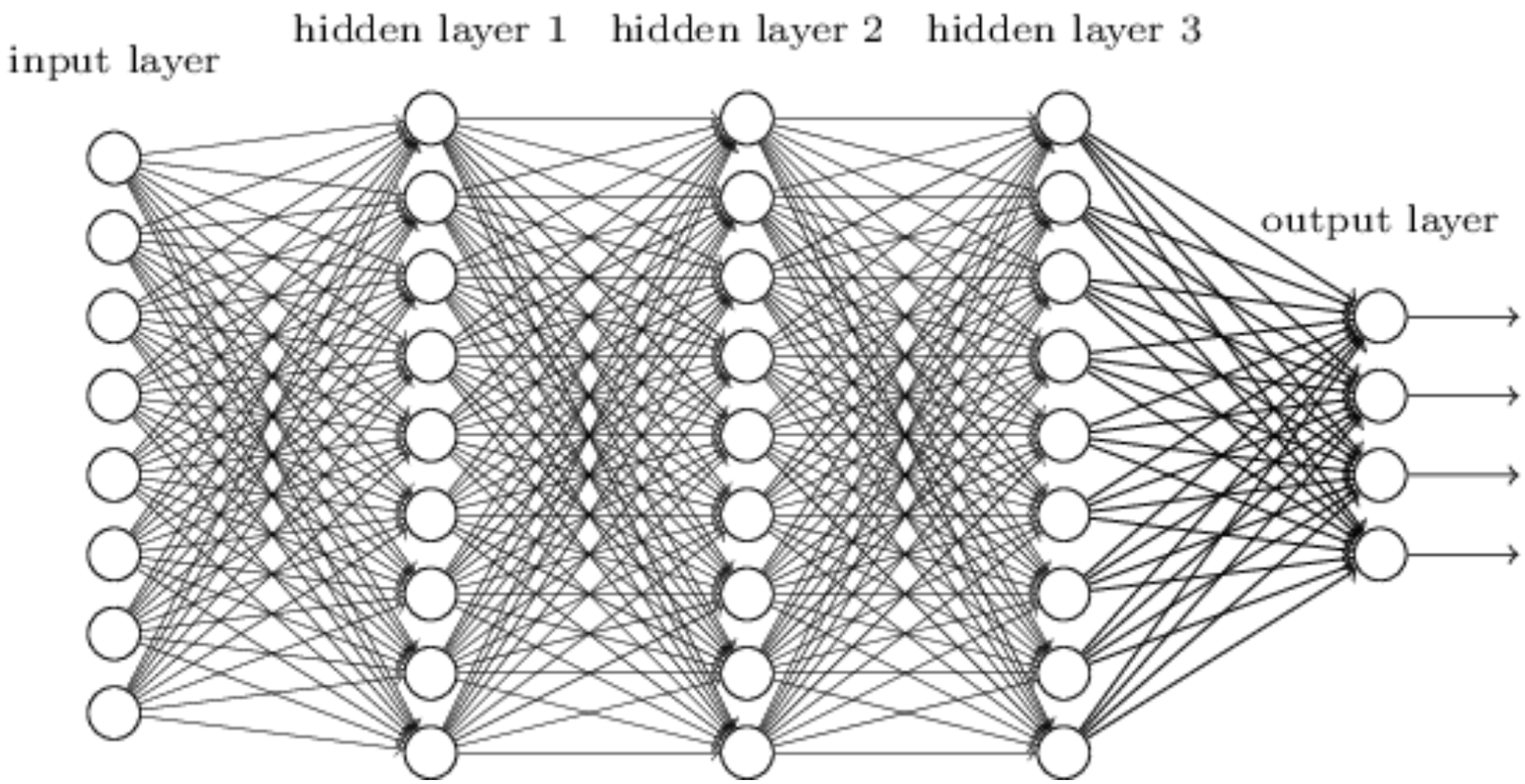
- *Backpropagation + Gradient descent*
- Pass input (x_1, x_2) to ANN
- Calculate output (y) and difference to true value ($\hat{y}$)
  This is the loss function L
- Find gradient of loss function with respect to weights
- Use gradient to find new weights

$$L(y, \hat{y}) = (y - \hat{y})^2$$

$$w_i' = w_i + \alpha \cdot \frac{\partial L}{\partial w_i}$$

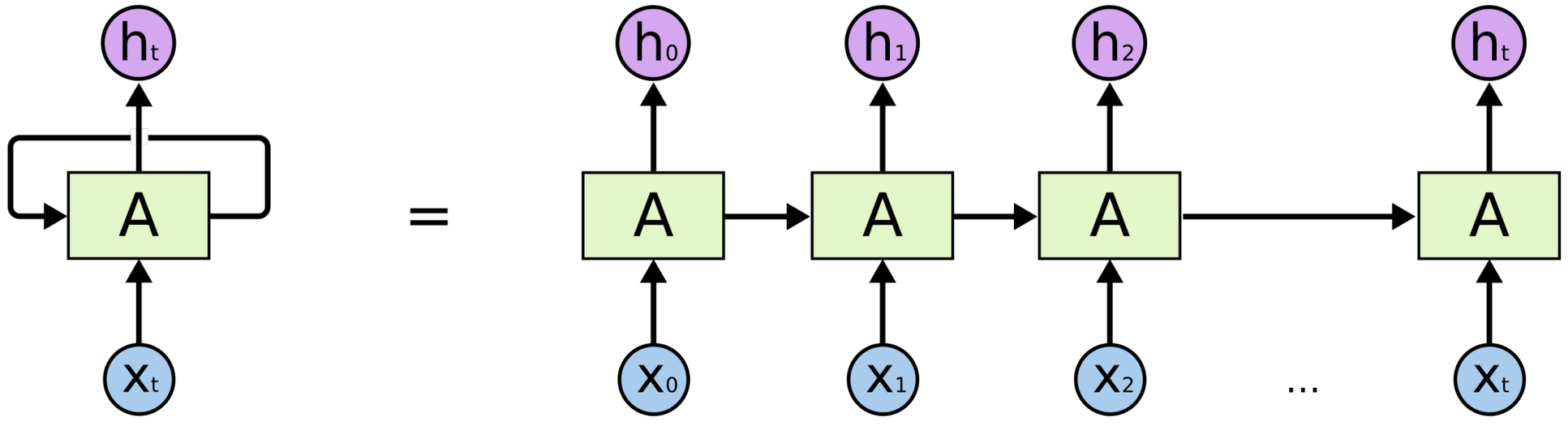


Fully Connected



- *Classical* ANN
- Most generic structure
- Many weights, inefficient
- Can we use the symmetry of the problem to simplify matters?

Recursive



- Inspired by natural language processing
- Work with a sequence of inputs
- Inputs can change the state of the cell (*Long Short Term Memory*)