ML landscape of top tagging

Gregor Kasieczka
(gregor.kasieczka@uni-hamburg.de)

Jets @ LHC

June 1st, 2021

CLUSTER OF EXCELLENCE
QUANTUM UNIVERSE

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

PIER
Partnership of Universität Hamburg and DESY

Emmy Noether-Programm
Deutsche Forschungsgemeinschaft
DFG

Bundesministerium für Bildung und Forschung

Overview

SciPost Physics Submission

The Machine Learning Landscape of Top Taggers

G. Kasieczka (ed)[1], T. Plehn (ed)[2], A. Butter[2], K. Cranmer[3], D. Debnath[4], B. M. Dillon[5], M. Fairbairn[6], D. A. Faroughy[5], W. Fedorko[7], C. Gay[7], L. Gouskos[8], J. F. Kamenik[5,9], P. T. Komiske[10], S. Leiss[1], A. Lister[7], S. Macaluso[3,4], E. M. Metodiev[10], L. Moore[11], B. Nachman,[12,13], K. Nordström[14,15], J. Pearkes[7], H. Qu[8], Y. Rath[16], M. Rieger[16], D. Shih[4], J. M. Thompson[2], and S. Varma[6]

1 Institut für Experimentalphysik, Universität Hamburg, Germany
2 Institut für Theoretische Physik, Universität Heidelberg, Germany
3 Center for Cosmology and Particle Physics and Center for Data Science, NYU, USA
4 NHECT, Dept. of Physics and Astronomy, Rutgers, The State University of NJ, USA
5 Jozef Stefan Institute, Ljubljana, Slovenia
6 Theoretical Particle Physics and Cosmology, King's College London, United Kingdom
7 Department of Physics and Astronomy, The University of British Columbia, Canada
8 Department of Physics, University of California, Santa Barbara, USA
9 Faculty of Mathematics and Physics, University of Ljubljana, Ljubljana, Slovenia
10 Center for Theoretical Physics, MIT, Cambridge, USA
11 CP3, Universitéxx Catholique de Louvain, Louvain-la-Neuve, Belgium
12 Physics Division, Lawrence Berkeley National Laboratory, Berkeley, USA
13 Simons Inst. for the Theory of Computing, University of California, Berkeley, USA
14 National Institute for Subatomic Physics (NIKHEF), Amsterdam, Netherlands
15 LPTHE, CNRS & Sorbonne Université, Paris, France
16 III. Physics Institute A, RWTH Aachen University, Germany

gregor.kasieczka@uni-hamburg.de
plehn@uni-heidelberg.de

July 24, 2019

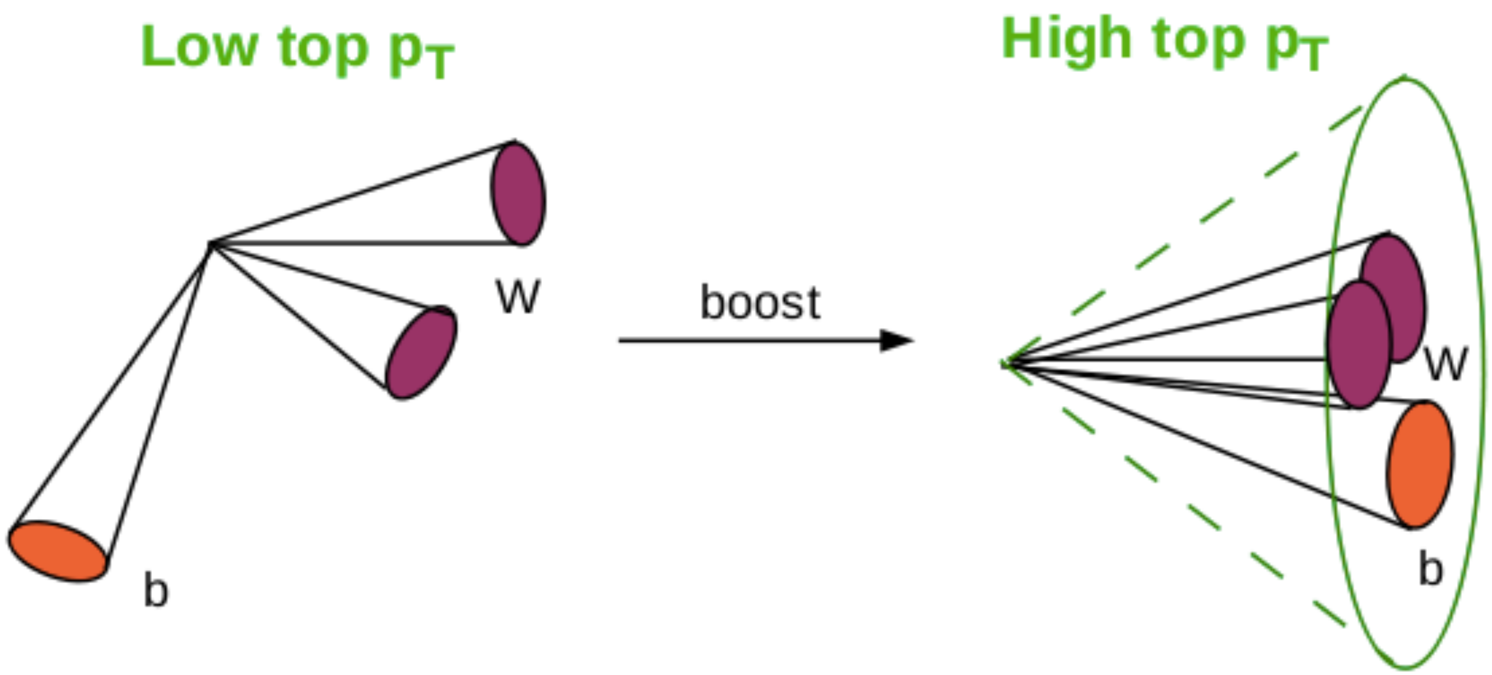
Abstract

Based on the established task of identifying boosted, hadronically decaying top quarks, we compare a wide range of modern machine learning approaches. Unlike most established methods they rely on low-level input, for instance calorimeter output. While their network architectures are vastly different, their performance is comparatively similar. In general, we find that these new approaches are extremely powerful and great fun.

- Tagging hadronically decaying tops is a well established benchmark for ML at LHC
- Broad comparison of methods in 1902.09914
- Outline
  - Introduction of task / dataset
  - Landscape of taggers
  - Going beyond

Introduction

Heavy Resonance Tagging



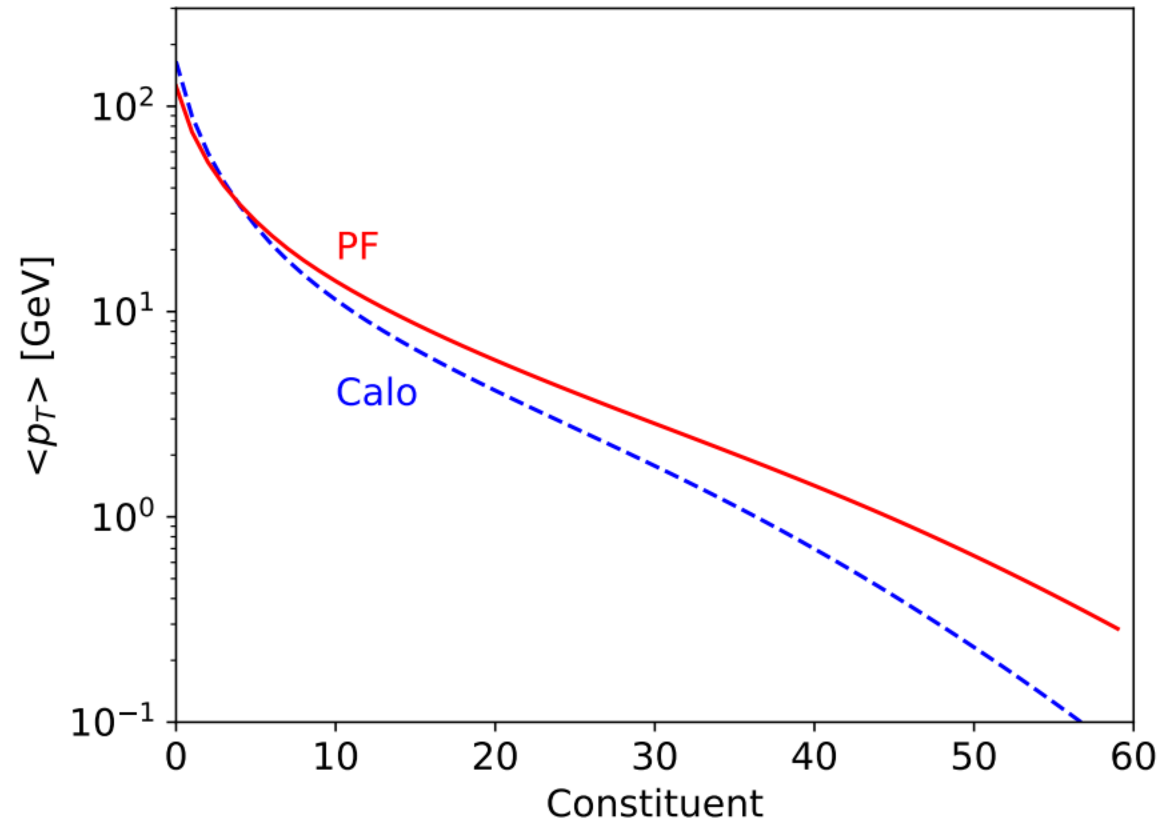
- Hadronically decaying top/Higgs/W/Z
- Contained in one (large-R) jet
  - m/pT >= ~1
- How to distinguish from light quark/gluon jets (and from each other)
- Used for new physics searches (and SM studies)

Classical handles:

- Mass
  e.g., *using a grooming algorithm*
- Centers of hard radiation
  e.g., *n-subjettiness or energy correlation functions*
- Flavour
  b tagging of large-R jets or subjets
- Combinations

➡ **Well-defined problem with simple performance metric:**
Great environment to test algorithms

Dataset



For consistent tests, need a **common dataset**:

- Based on 1701.08784
- Pythia simulated light quark+ gluon (background) vs hadronically decaying top quarks (signal) with pT = 550..650 GeV
- Delphes simulation, simple particle flow (PF)
- FastJet, AntiKt R=0.8, truth-matched
- 1.2M training examples, 400k each for testing and validation
- Store up too 200 constituent four-vectors of leading pT jet

Starting from four-vectors allows a large number of approaches to be compared *(more on that soon)*

Limitations / possible improvements

- Inclusion of pile-up
- Track/vertex information
- Statistics
- Realistic detector model
- Systematic uncertainties

Data available at:
https://desycloud.desy.de/index.php/s/llbX3zpLhazgPJ6

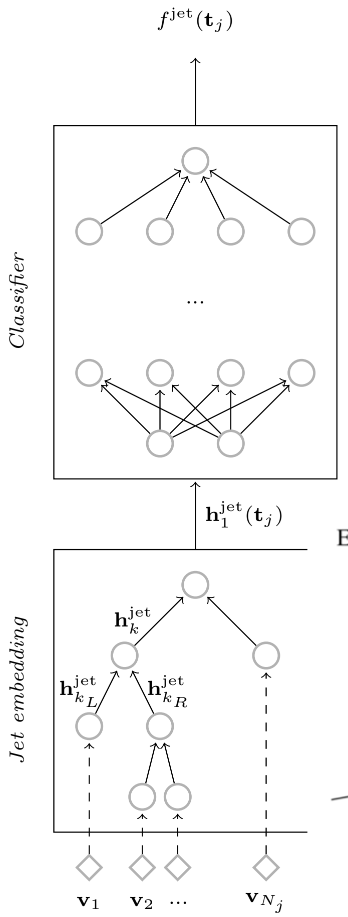
Machine Learning Mini-Intro

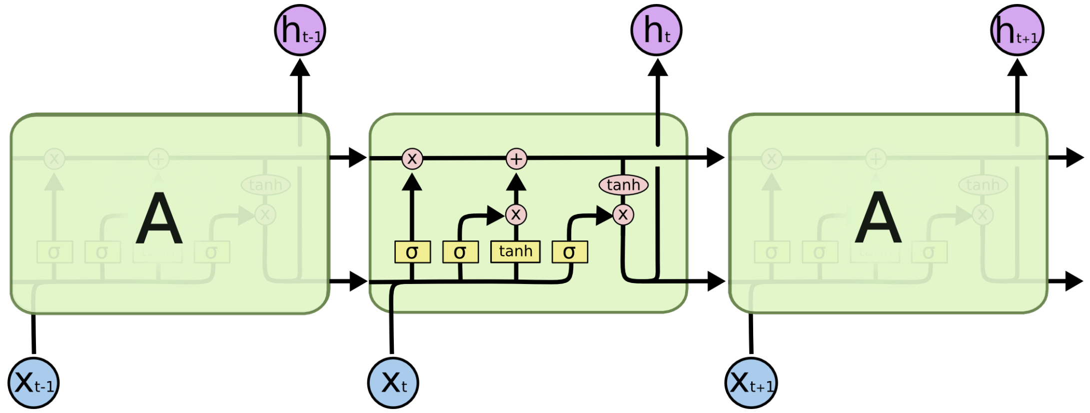
- Formulate task as a minimisation problem and solve

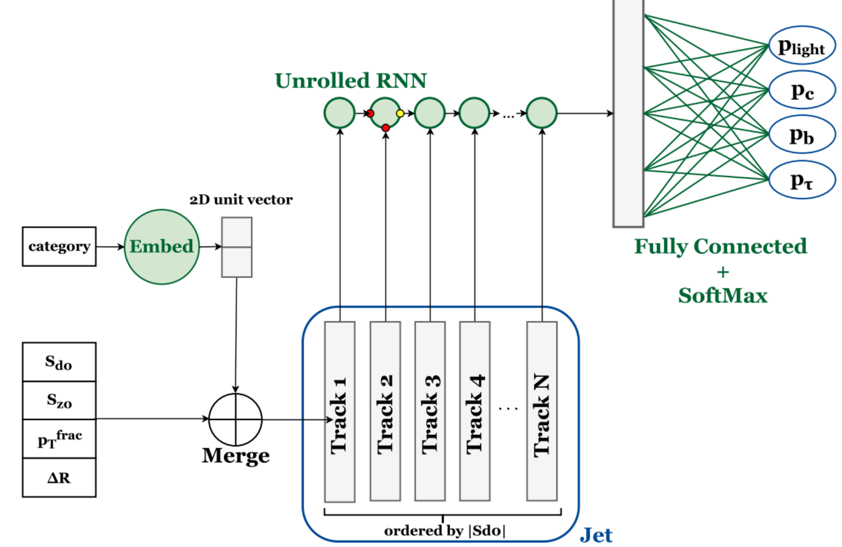
$$\theta^* = \mathrm{argmin}_\theta \, \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})} \left[\mathcal{L}(f_\theta(\mathbf{x}), \mathbf{x})\right]$$

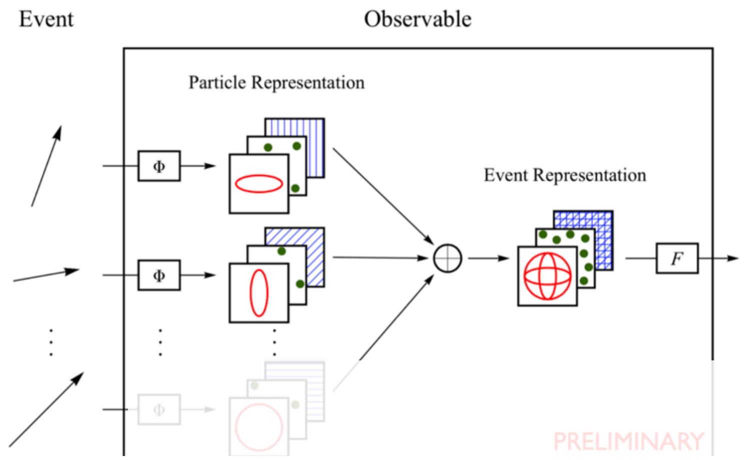
$$\text{Loss function } \mathcal{L}$$
$$\text{Neural network } f_\theta$$
$$\text{Parameters } \theta$$
$$\text{Opt. Parameters } \theta^*$$
$$\text{Data } \mathbf{x}$$
$$\text{Data distribution } p(\mathbf{x})$$

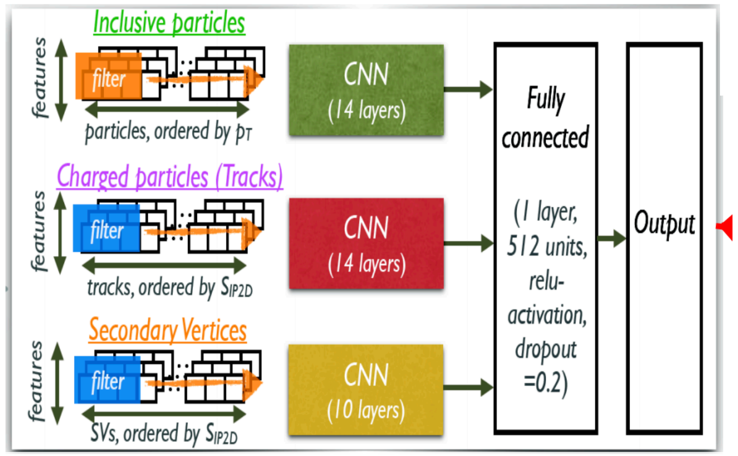
- Neural networks are a convenient way of building expressive functions with many tuneable parameters (10s to millions) that can efficiently be optimised via gradient descent
- Loss function to distinguish two classes: cross-entropy
  - *(We'll come back to that)*
- If networks have many parameters:
  - *Interesting choices how to structure them (architecture)*
  - *Which ways of connecting the nodes in a neural network work well for physics data?*

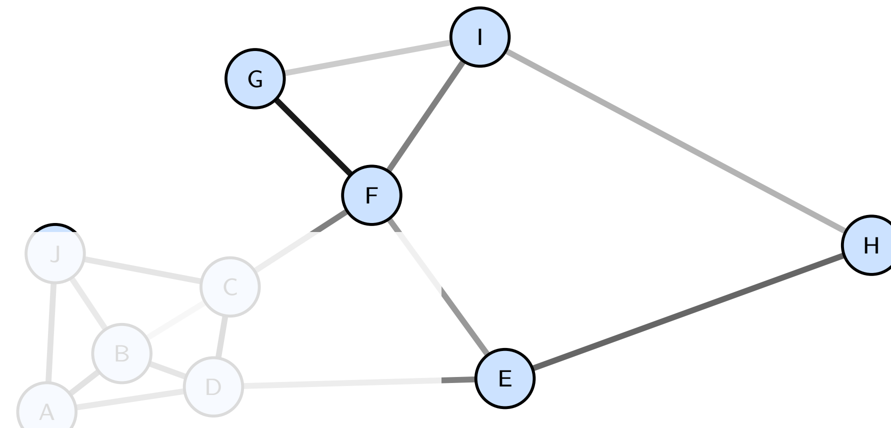




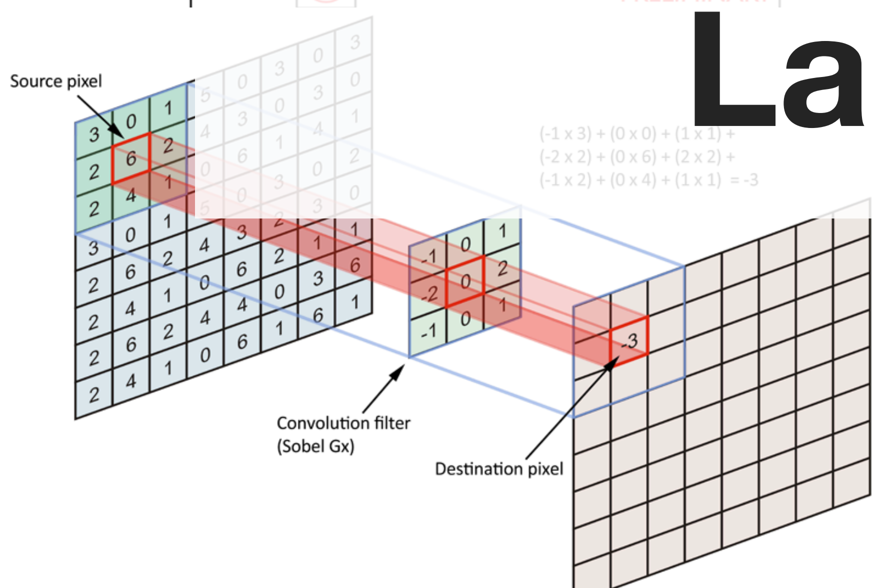


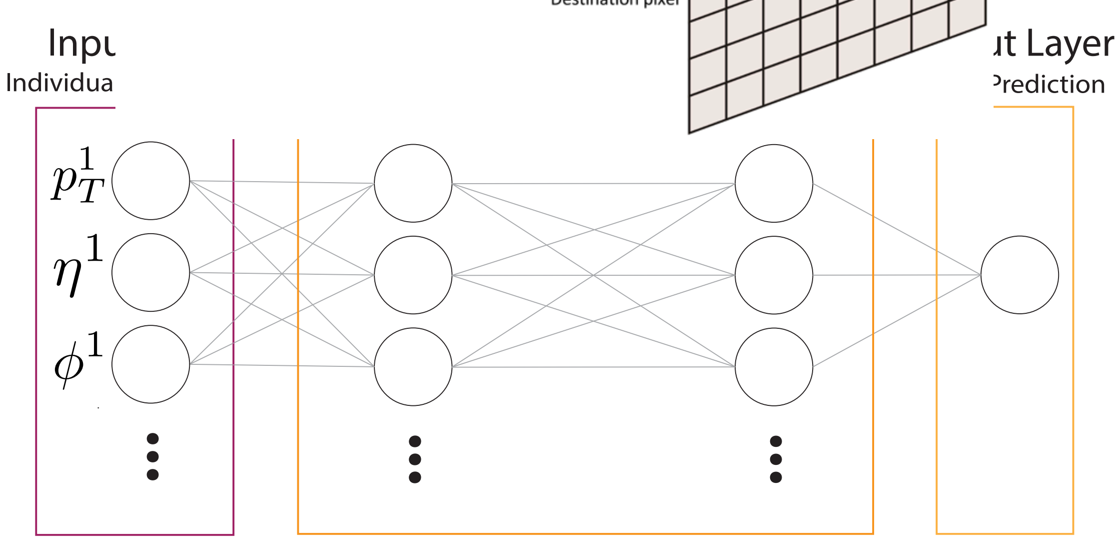






Landscape





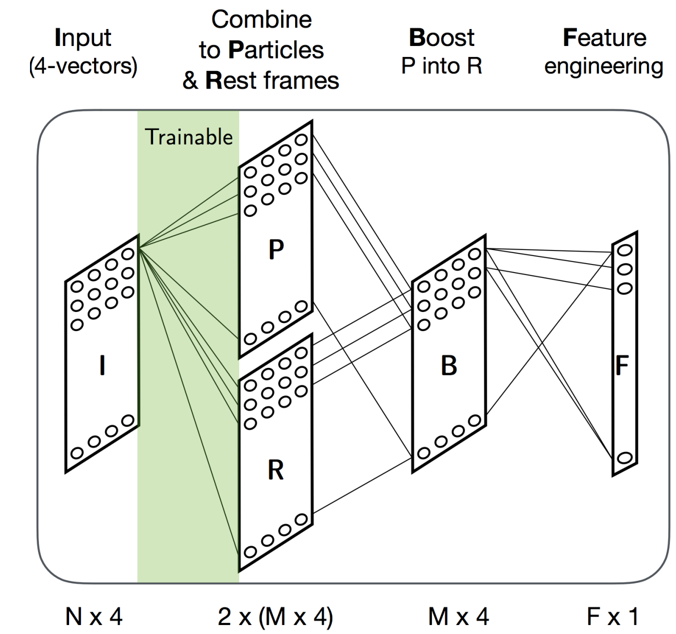
$$k_{\mu,i} = \begin{pmatrix} E_0 & E_1 & \cdots & E_N \\ p_{x,0} & p_{x,1} & \cdots & p_{x,N} \\ p_{y,0} & p_{y,1} & \cdots & p_{y,N} \\ p_{z,0} & p_{z,1} & \cdots & p_{z,N} \end{pmatrix}$$

Combination Layer (**CoLa**): create linear combinations:

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

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

$$\widetilde{k}_{\mu,i} \to \sum_j (\widetilde{k}_i - \widetilde{k}_j)_\mu (\widetilde{k}_i - \widetilde{k}_j)_\nu \eta^{\mu\nu} B_{ij}$$



Methods included

CNN [16]
ResNeXt [31]

TopoDNN [18]
Multi-body N-subjettiness 6 [24]
Multi-body N-subjettiness 8 [24]
TreeNiN [43]
P-CNN
ParticleNet [47]

LBN [19]
LoLa [22]
LDA [54]
Energy Flow Polynomials [21]
Energy Flow Network [23]
Particle Flow Network [23]

Images / Convolutional networks

Fully connected network on 4-vectors

Fully connected network on n-subjettiness

Recurrent analysis of clustering tree

1D Convolution on 4 vectors

Graphs

Explicit use of Lorentz Boost

Dirichlet analysis

New set of substructure observables

Set based

- Highlighting some select architectures
- See 1902.09914 and refs therein for full picture

Methods included

CNN [16]
ResNeXt [31]

TopoDNN [18]
Multi-body N-subjettiness 6 [24]
Multi-body N-subjettiness 8 [24]
TreeNiN [43]
P-CNN
ParticleNet [47]

LBN [19]
LoLa [22]
LDA [54]
Energy Flow Polynomials [21]
Energy Flow Network [23]
Particle Flow Network [23]

Images / Convolutional networks

Fully connected network on 4-vectors

Fully connected network on n-subjettiness

Recurrent analysis of clustering tree

1D Convolution on 4 vectors

Graphs

Explicit use of Lorentz Boost

Dirichlet analysis

New set of substructure observables

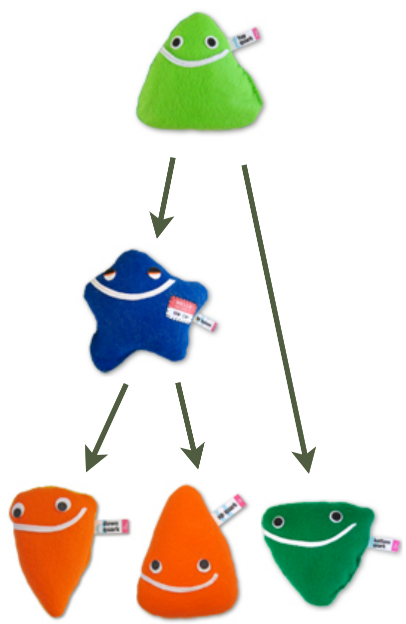
Set based

- Highlighting some select architectures
- See 1902.09914 and refs therein for full picture

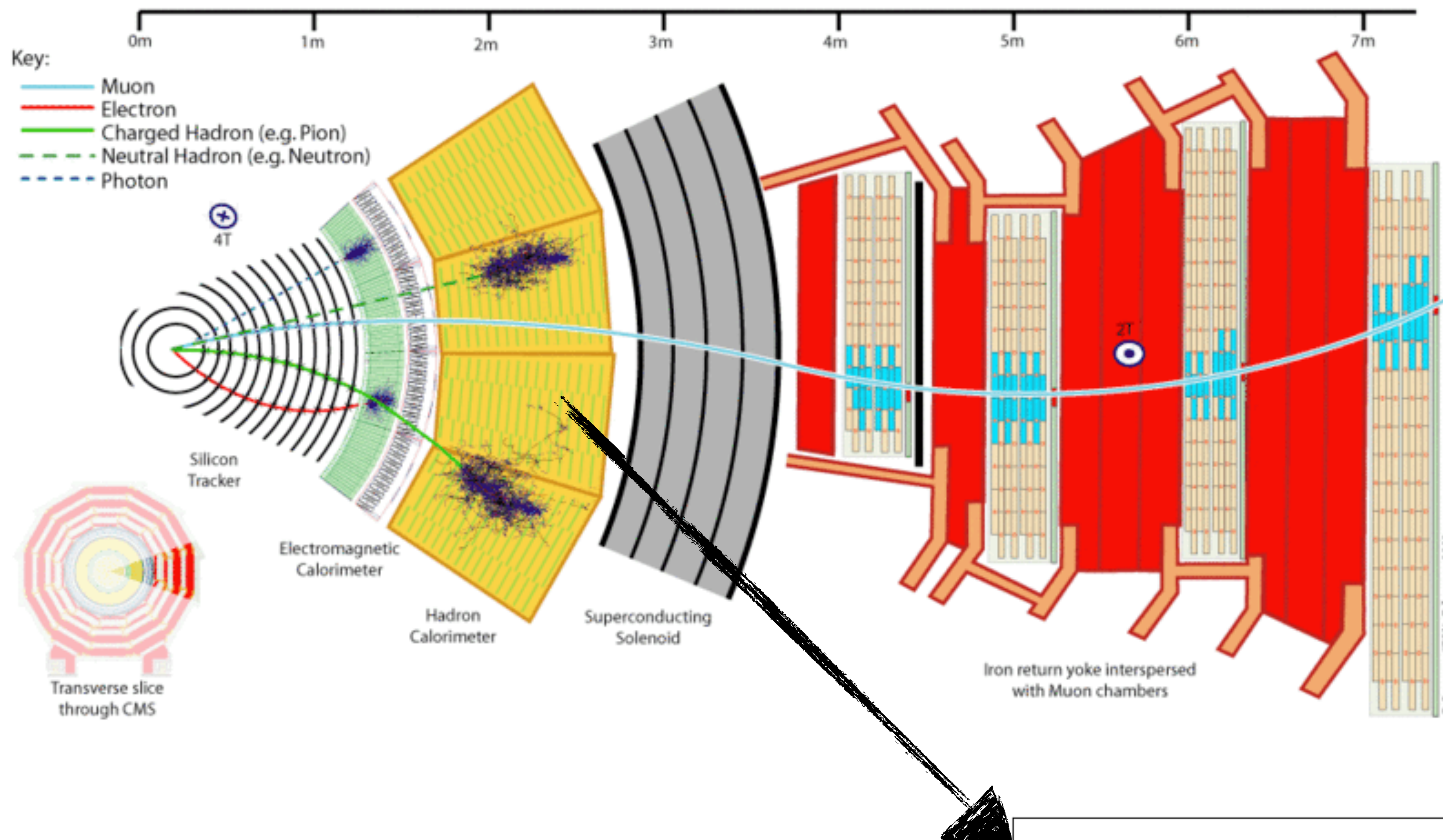
Jet Images

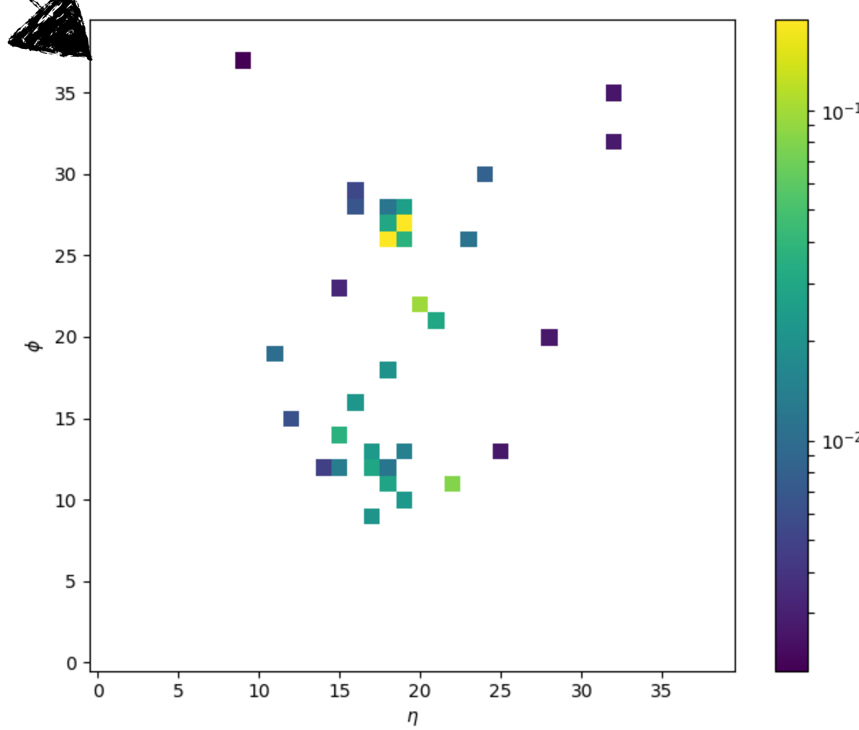
(jet images by C Daza)

Top Quark



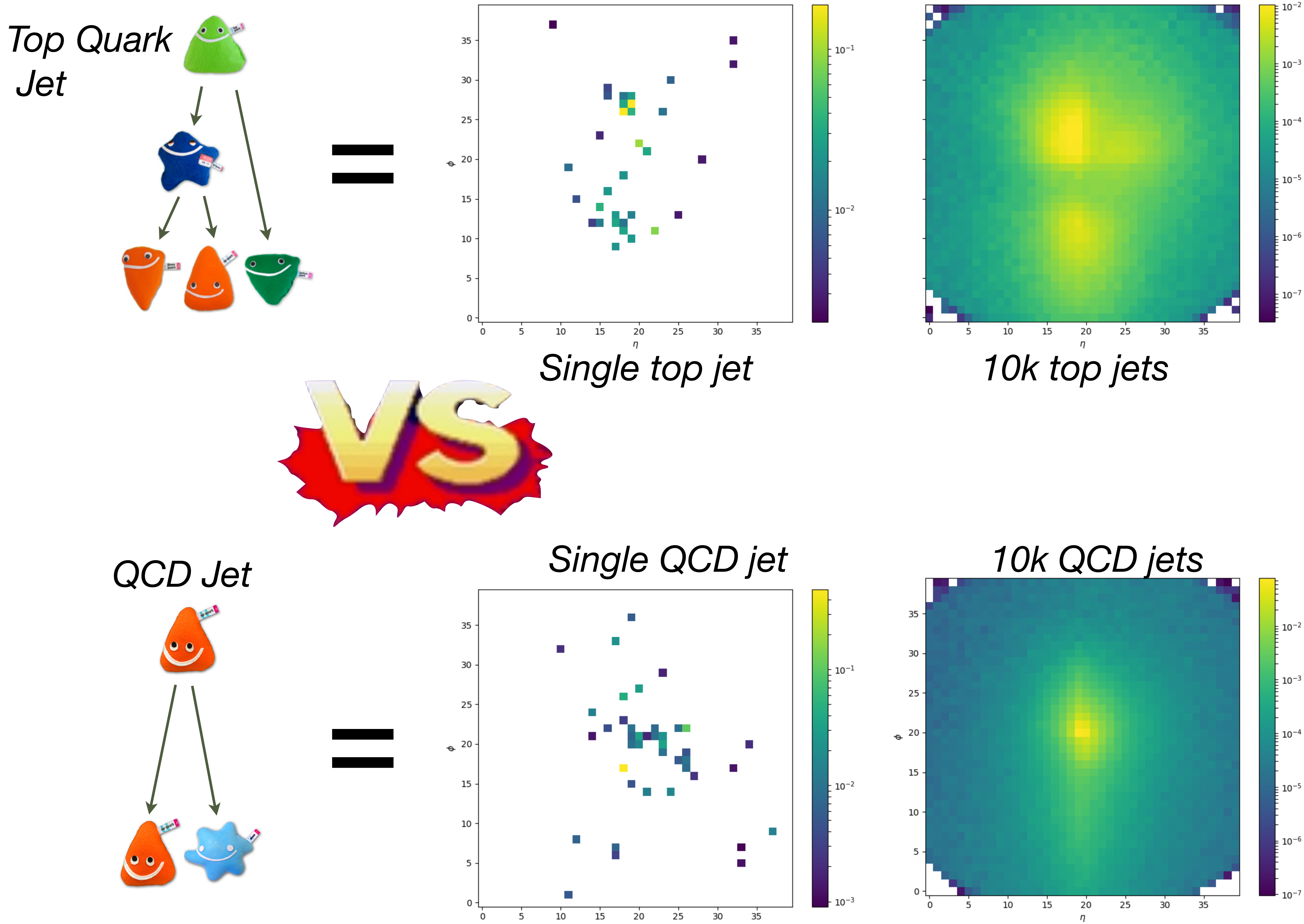
+





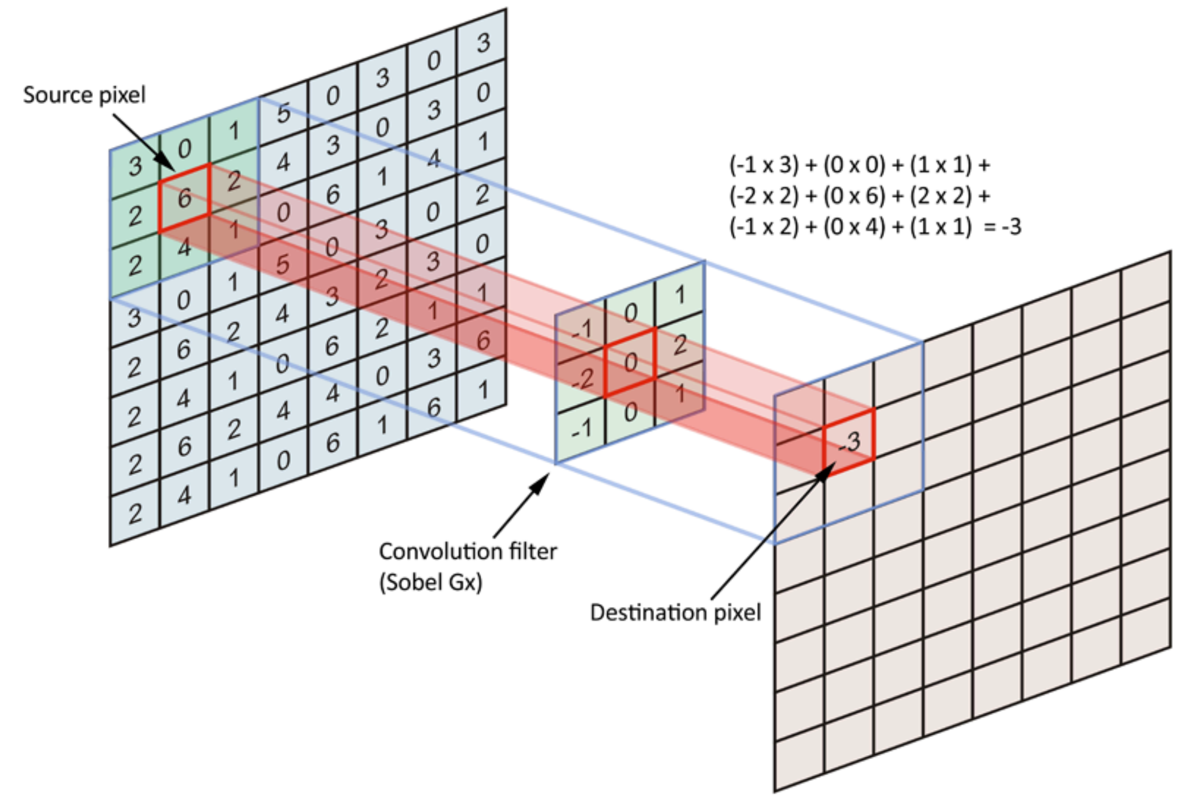
- Treat jets as images: 1407.5675, 1501.05968, 1511.05190, 1612.01551, 1701.08784, 1803.00107,….
  - Popular and done before deep learning

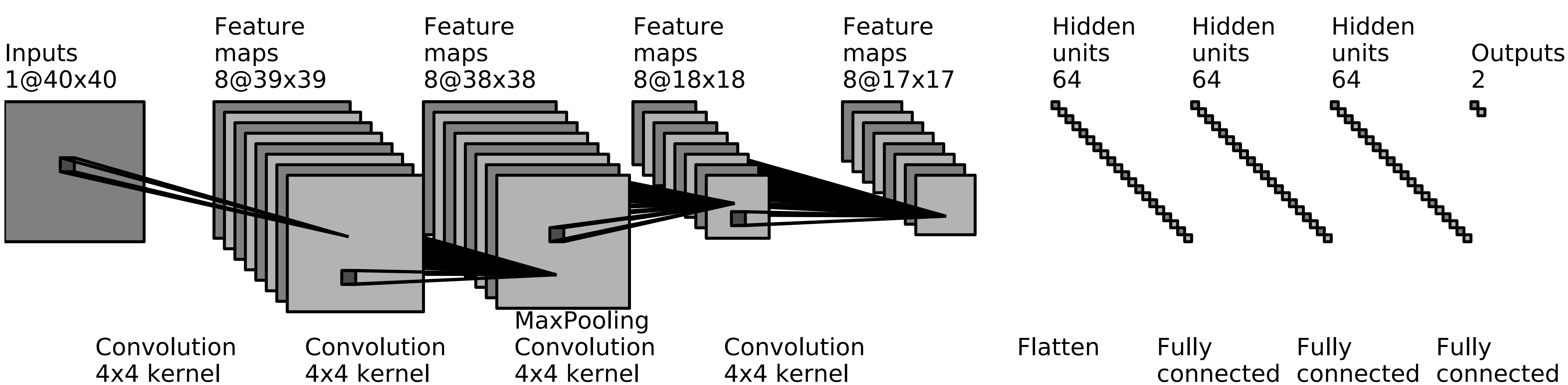
- Measure particle energies in calorimeter
- Image preprocessing
  - center, rotate, mirror, pixelate, trim, normalise


Top Quark Jet
=
Single top jet
10k top jets
VS
QCD Jet
=
Single QCD jet
10k QCD jets
η
ϕ

Convolutional network

- Analyse grid-like data with convolutional networks
  - Same architectures as for computer vision
- Accounts for locality (correlation of nearby pixels) and *translation invariance*
- Potential limitation due to sparsity/pixelisation for high resolution data
  - No strong effect observed in this study
  - Careful how to pre-process (1803.00107)





Architecture from 1701.08784

Methods included

CNN [16]
ResNeXt [31]

TopoDNN [18]
Multi-body N-subjettiness 6 [24]
Multi-body N-subjettiness 8 [24]
TreeNiN [43]
P-CNN
ParticleNet [47]

LBN [19]
LoLa [22]
LDA [54]
Energy Flow Polynomials [21]
Energy Flow Network [23]
Particle Flow Network [23]

Images / Convolutional networks

Fully connected network on 4-vectors

Fully connected network on n-subjettiness

Recurrent analysis of clustering tree

1D Convolution on 4 vectors

Graphs

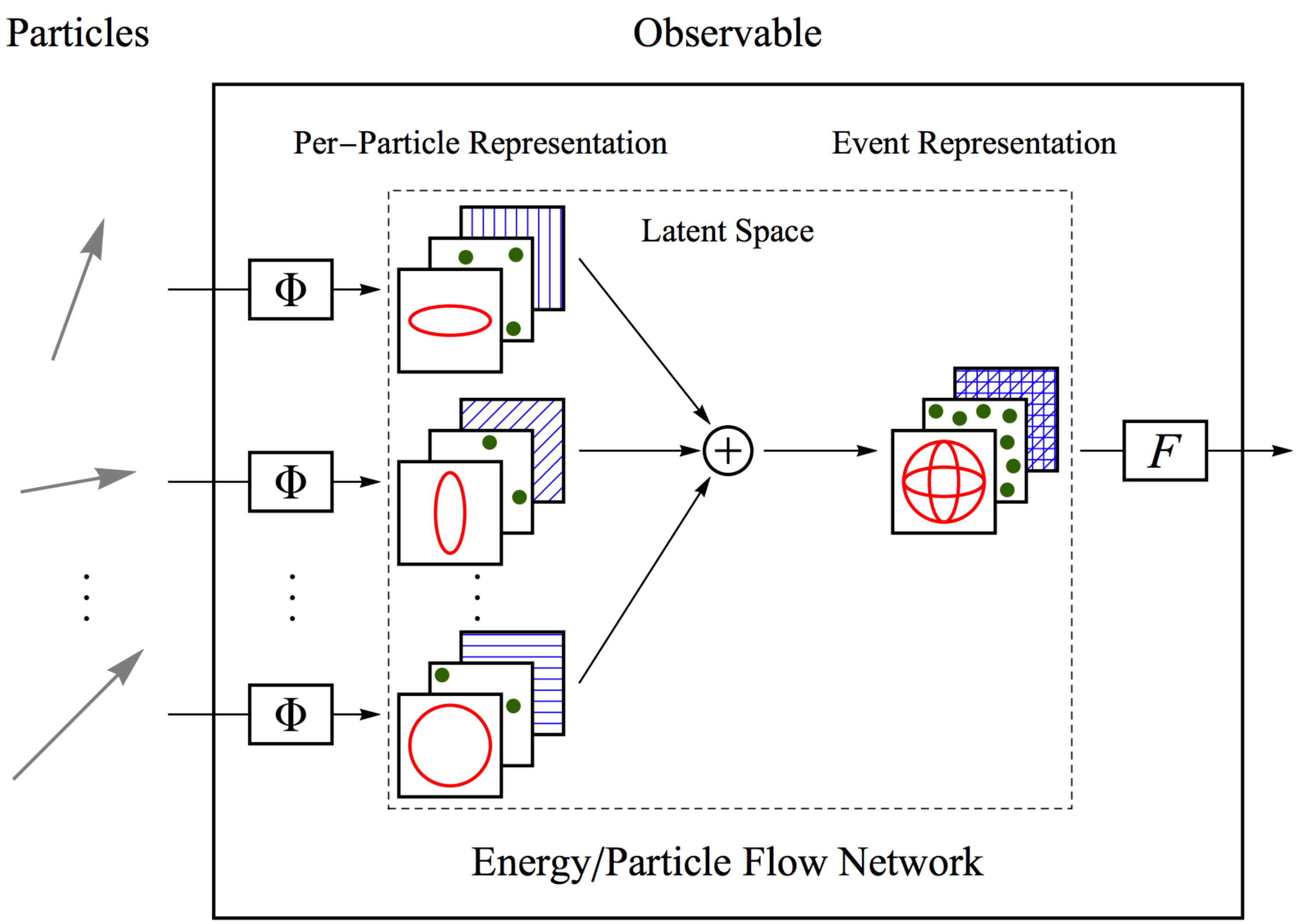
Explicit use of Lorentz Boost

Dirichlet analysis

New set of substructure observables

Set based

Deep Sets



- To reduce pre-processing, might want to work with four-vector inputs of particles
- How to make independent from ordering of four vectors?
  - Use permutation invariance of sum
  - →Deep set architecture (1703.06114)
  - Apply to jets: energy flow network (EFN) / particle flow network (PFN) (1810.05165)
- Simple and straightforward to implement but limited use of neighbourhood information

General :

$$\text{PFN:} \quad F\left(\sum_{i=1}^{M} \Phi(p_i)\right)$$

IRC safe:

$$\text{EFN:} \quad F\left(\sum_{i=1}^{M} z_i \Phi(\hat{p}_i)\right)$$

1810.05165

Methods included

CNN [16]
ResNeXt [31]

TopoDNN [18]
Multi-body N-subjettiness 6 [24]
Multi-body N-subjettiness 8 [24]
TreeNiN [43]
P-CNN
ParticleNet [47]

LBN [19]
LoLa [22]
LDA [54]
Energy Flow Polynomials [21]
Energy Flow Network [23]
Particle Flow Network [23]

Images / Convolutional networks

Fully connected network on 4-vectors

Fully connected network on n-subjettiness

Recurrent analysis of clustering tree

1D Convolution on 4 vectors

Graphs

Explicit use of Lorentz Boost

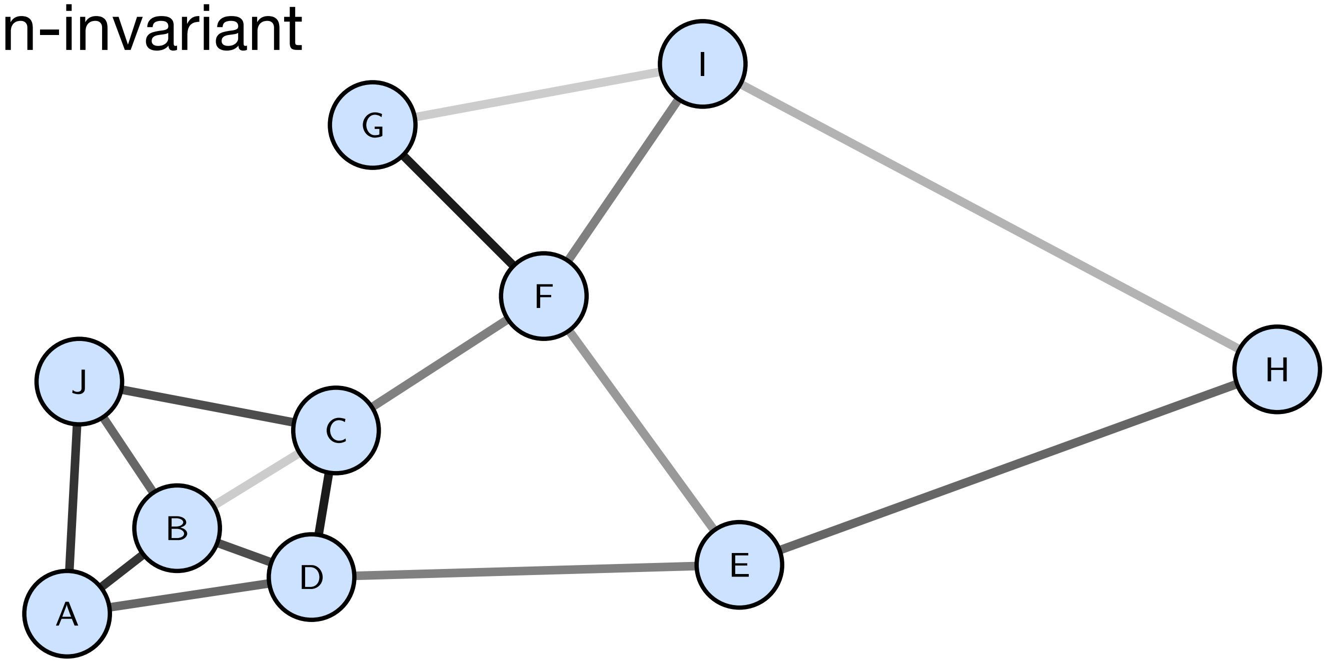
Dirichlet analysis

New set of substructure observables

Set based

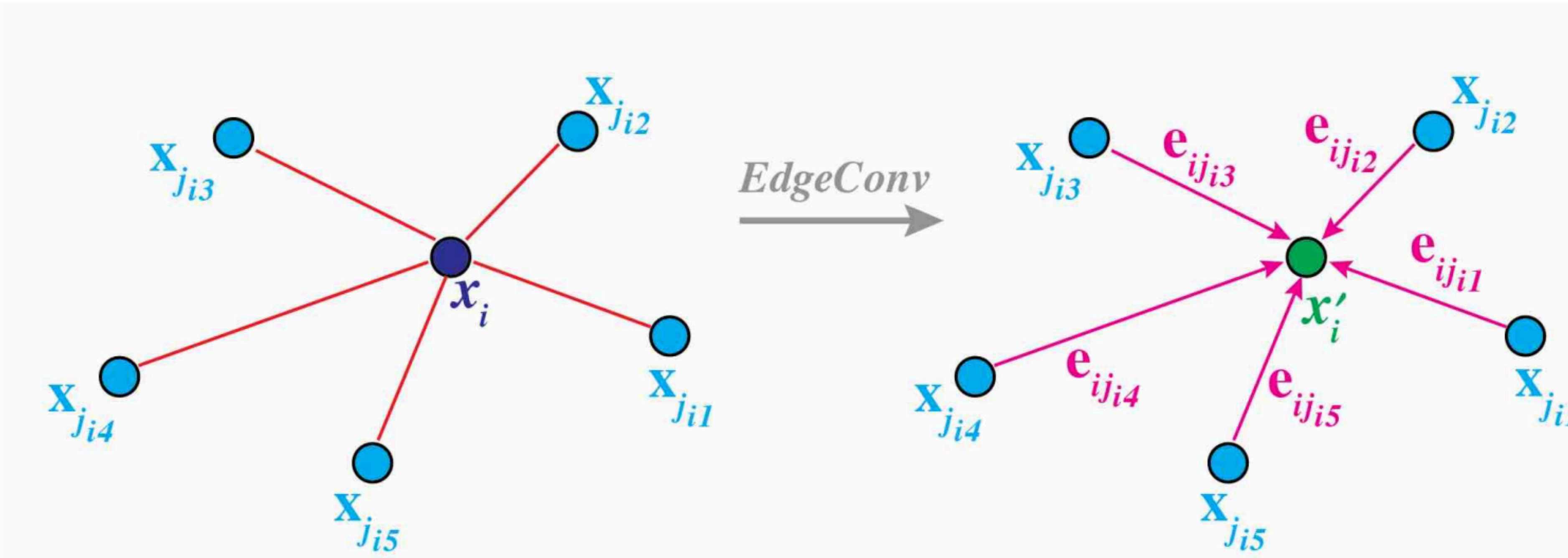
Graphs

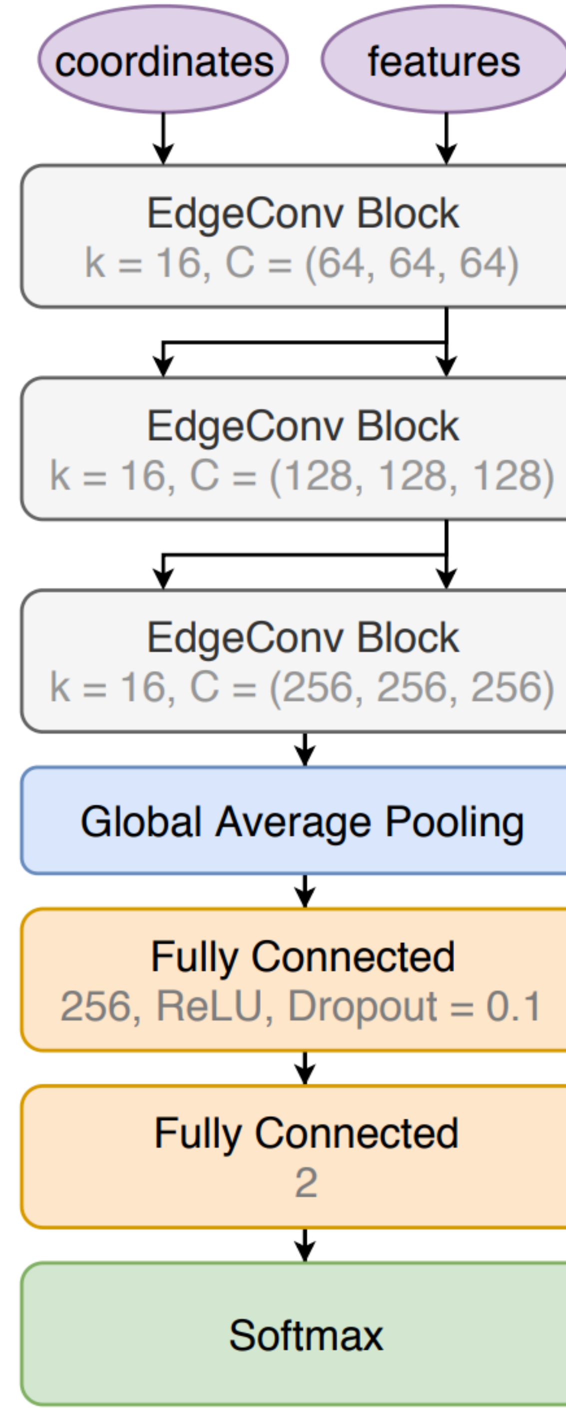
- Combine locality of images with permutation-invariant handling of four-vectors
  →Graphs
- How to build a graph
  - Vertex: particle (e.g., four-vector)
  - Edge: distance (for example geometric)
- Works with:
  - Data that naturally comes as a graph (e.g. a decay sequence)
  - Data embedded in some geometric space (point cloud)
- Active development of graphs on CS side, increasing number of physics applications: 1902.08570, 1902.07987, 1908.05318, 2008.03601, 2103.16701, 2101.08578, ….



1902.08570

Closer look





- Interactions of particles with its nearest neighbours
  - Initially in physical space, later in learned space

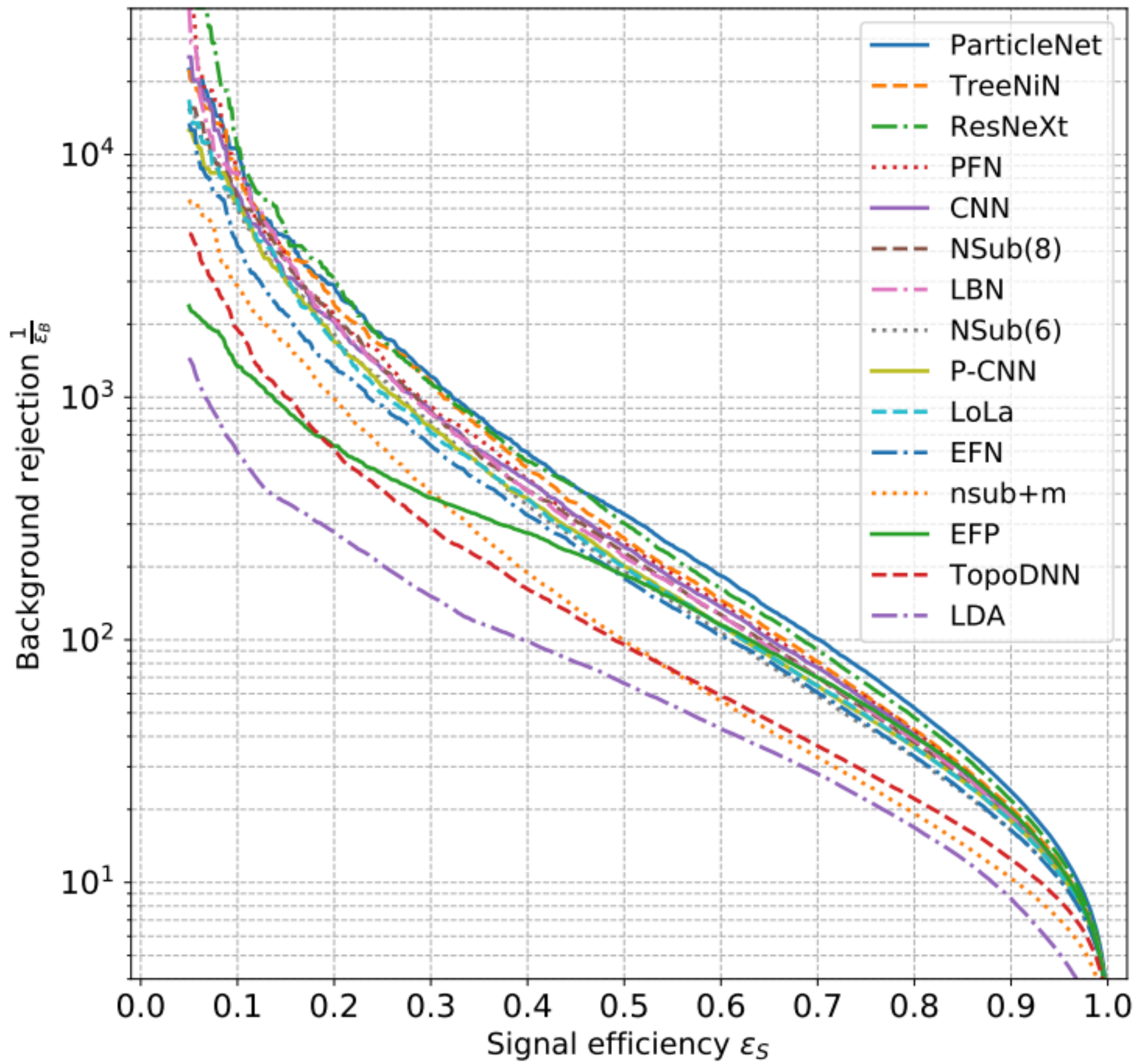
$$\boldsymbol{x}'_i = \mathop{\square}_{j=1}^{k} \boldsymbol{h}_{\boldsymbol{\Theta}}(\boldsymbol{x}_i, \boldsymbol{x}_{i_j})$$

Neural network

$$\boldsymbol{h}_{\boldsymbol{\Theta}}(\boldsymbol{x}_i, \boldsymbol{x}_{i_j}) = \bar{\boldsymbol{h}}_{\boldsymbol{\Theta}}(\boldsymbol{x}_i, \boldsymbol{x}_{i_j} - \boldsymbol{x}_i)$$

Aggregation function (sum or max)

Results



General gain of ~2x compared to baseline (mass+few n-subjettiness variables)

Results

	AUC	Acc
CNN [16]	0.981	0.930
ResNeXt [31]	0.984	0.936
TopoDNN [18]	0.972	0.916
Multi-body N-subjettiness 6 [24]	0.979	0.922
Multi-body N-subjettiness 8 [24]	0.981	0.929
TreeNiN [43]	0.982	0.933
P-CNN	0.980	0.930
ParticleNet [47]	0.985	0.938
LBN [19]	0.981	0.931
LoLa [22]	0.980	0.929
LDA [54]	0.955	0.892
Energy Flow Polynomials [21]	0.980	0.932
Energy Flow Network [23]	0.979	0.927
Particle Flow Network [23]	0.982	0.932

- Strongest performance from ParticleNet (Graph based)
- Close field of well-performing approaches

Results

	AUC	Acc	$1/\epsilon_B$ ($\epsilon_S = 0.3$)		
			single	mean	median
CNN [16]	0.981	0.930	914±14	995±15	975±18
ResNeXt [31]	0.984	0.936	1122±47	1270±28	1286±31
TopoDNN [18]	0.972	0.916	295±5	382± 5	378 ± 8
Multi-body N-subjettiness 6 [24]	0.979	0.922	792±18	798±12	808±13
Multi-body N-subjettiness 8 [24]	0.981	0.929	867±15	918±20	926±18
TreeNiN [43]	0.982	0.933	1025±11	1202±23	1188±24
P-CNN	0.980	0.930	732±24	845±13	834±14
ParticleNet [47]	0.985	0.938	1298±46	1412±45	1393±41
LBN [19]	0.981	0.931	836±17	859±67	966±20
LoLa [22]	0.980	0.929	722±17	768±11	765±11
LDA [54]	0.955	0.892	151±0.4	151.5±0.5	151.7±0.4
Energy Flow Polynomials [21]	0.980	0.932	384		
Energy Flow Network [23]	0.979	0.927	633±31	729±13	726±11
Particle Flow Network [23]	0.982	0.932	891±18	1063±21	1052±29

- Gains from ensembles (averaging network predictions)
- Not really news for fans of BDTs

Results

	AUC	Acc	$1/\epsilon_B$ ($\epsilon_S = 0.3$) single	mean	median
CNN [16]	0.981	0.930	914±14	995±15	975±18
ResNeXt [31]	0.984	0.936	1122±47	1270±28	1286±31
TopoDNN [18]	0.972	0.916	295±5	382± 5	378 ± 8
Multi-body N-subjettiness 6 [24]	0.979	0.922	792±18	798±12	808±13
Multi-body N-subjettiness 8 [24]	0.981	0.929	867±15	918±20	926±18
TreeNiN [43]	0.982	0.933	1025±11	1202±23	1188±24
P-CNN	0.980	0.930	732±24	845±13	834±14
ParticleNet [47]	0.985	0.938	1298±46	1412±45	1393±41
LBN [19]	0.981	0.931	836±17	859±67	966±20
LoLa [22]	0.980	0.929	722±17	768±11	765±11
LDA [54]	0.955	0.892	151±0.4	151.5±0.5	151.7±0.4
Energy Flow Polynomials [21]	0.980	0.932	384		
Energy Flow Network [23]	0.979	0.927	633±31	729±13	726±11
Particle Flow Network [23]	0.982	0.932	891±18	1063±21	1052±29
GoaT	0.985	0.939	1368±140		1549±208

Slight gains from combining all taggers - limited orthogonally

Results

	AUC	Acc	$1/\epsilon_B$ ($\epsilon_S = 0.3$)			#Param
			single	mean	median	
CNN [16]	0.981	0.930	914±14	995±15	975±18	610k
ResNeXt [31]	0.984	0.936	1122±47	1270±28	1286±31	1.46M
TopoDNN [18]	0.972	0.916	295±5	382± 5	378 ± 8	59k
Multi-body N-subjettiness 6 [24]	0.979	0.922	792±18	798±12	808±13	57k
Multi-body N-subjettiness 8 [24]	0.981	0.929	867±15	918±20	926±18	58k
TreeNiN [43]	0.982	0.933	1025±11	1202±23	1188±24	34k
P-CNN	0.980	0.930	732±24	845±13	834±14	348k
ParticleNet [47]	0.985	0.938	1298±46	1412±45	1393±41	498k
LBN [19]	0.981	0.931	836±17	859±67	966±20	705k
LoLa [22]	0.980	0.929	722±17	768±11	765±11	127k
LDA [54]	0.955	0.892	151±0.4	151.5±0.5	151.7±0.4	184k
Energy Flow Polynomials [21]	0.980	0.932	384			1k
Energy Flow Network [23]	0.979	0.927	633±31	729±13	726±11	82k
Particle Flow Network [23]	0.982	0.932	891±18	1063±21	1052±29	82k
GoaT	0.985	0.939	1368±140		1549±208	35k

More parameters do not automatically give more performance

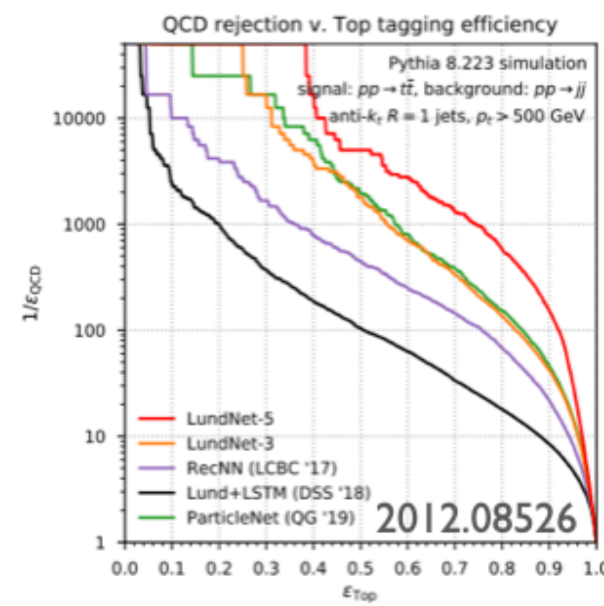
Summary so far

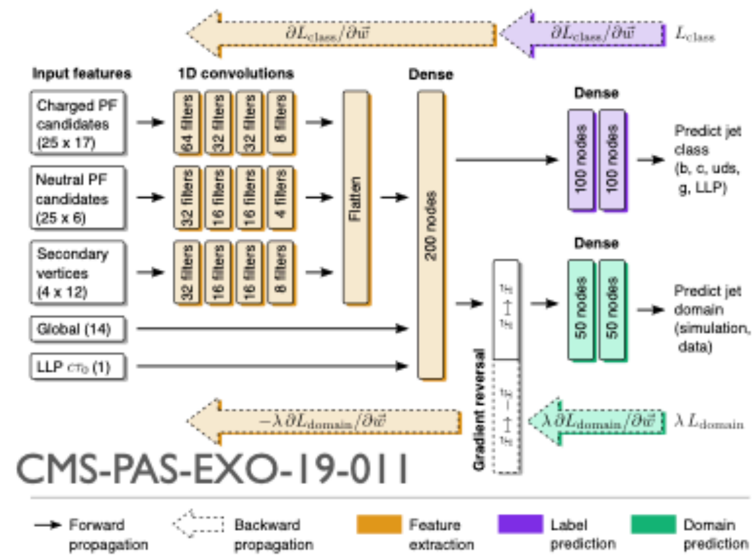
- Simple top-tagging problem as useful benchmark
- Comparison of different network architectures and data representations
- General large gain from more complex networks compared to traditional approaches
- Graph networks perform best, but dense field of good performances
- Other criteria will be more relevant for use:
    - Speed, stability, ease of training, …

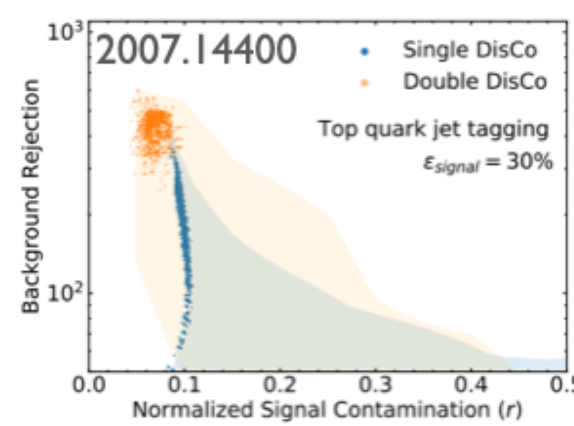
Beyond

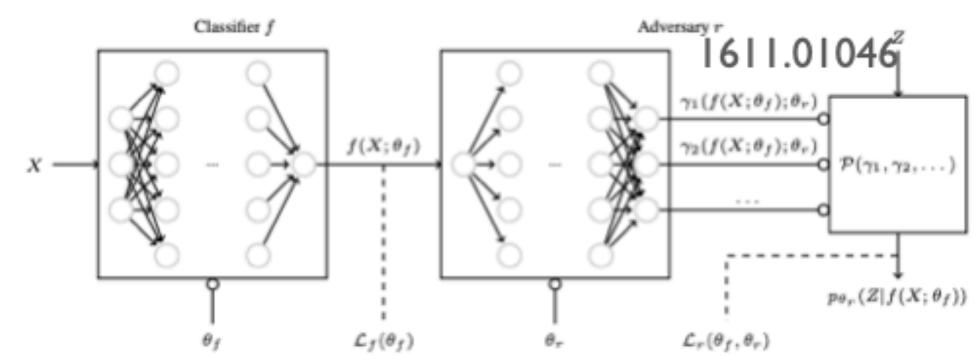
Are we done?

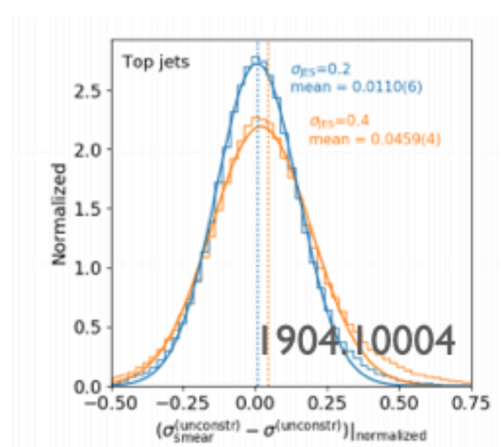
- No (!)

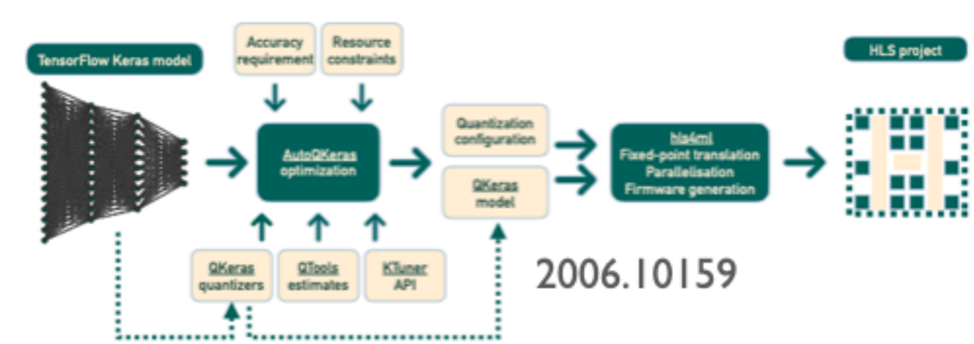






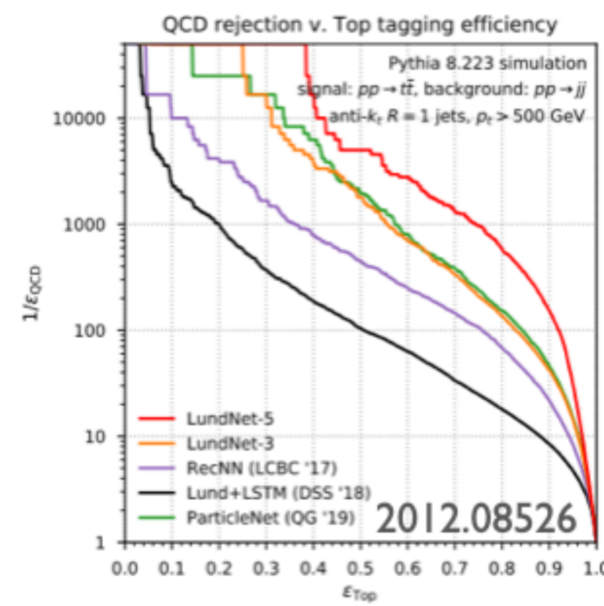


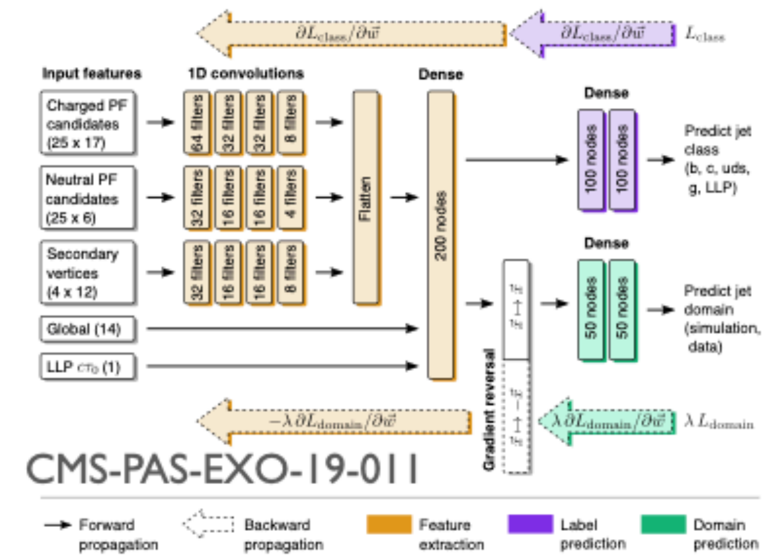


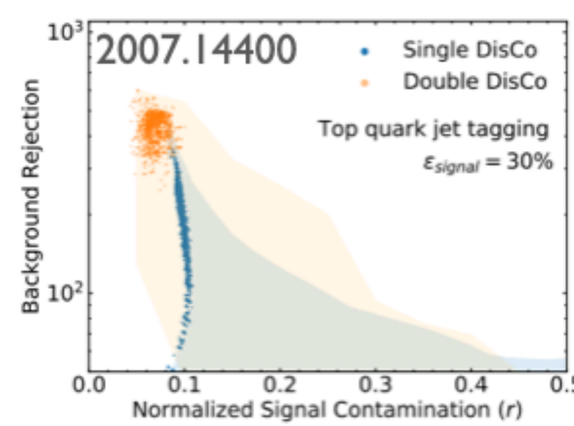


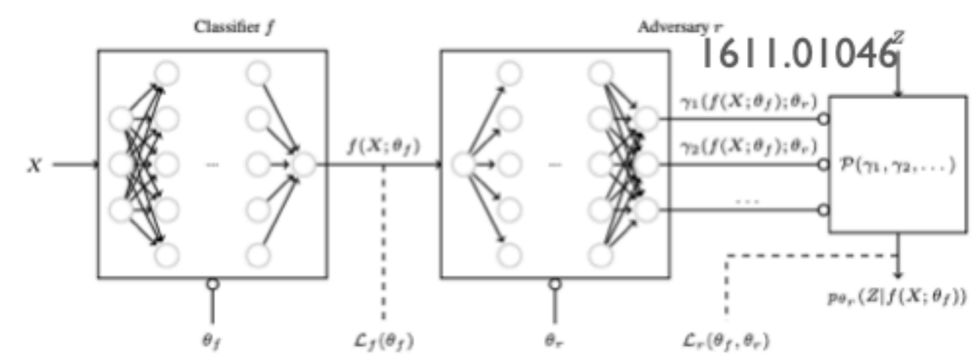
Are we done?

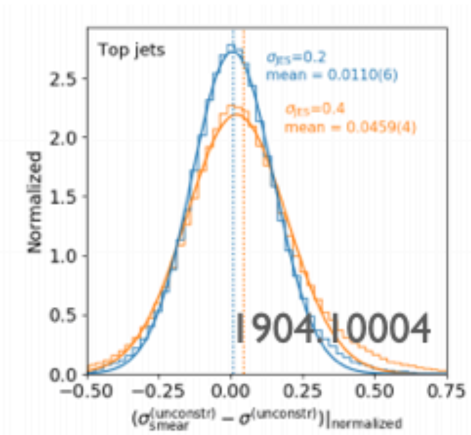
- No (!)
- Need:
  - **Higger accuracy (easy to measure, many results)**

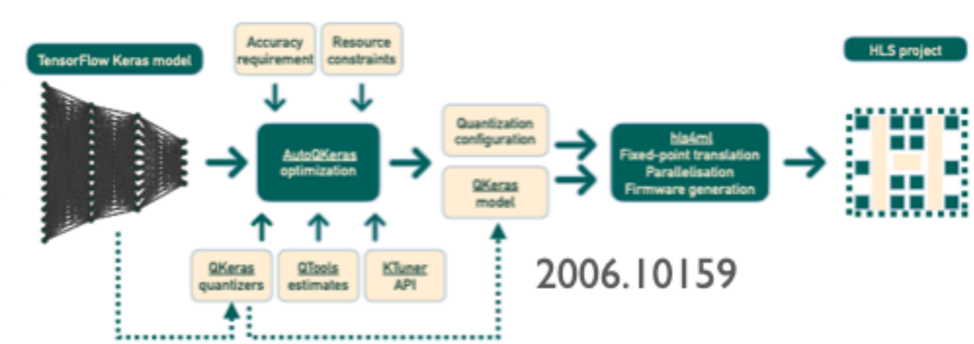






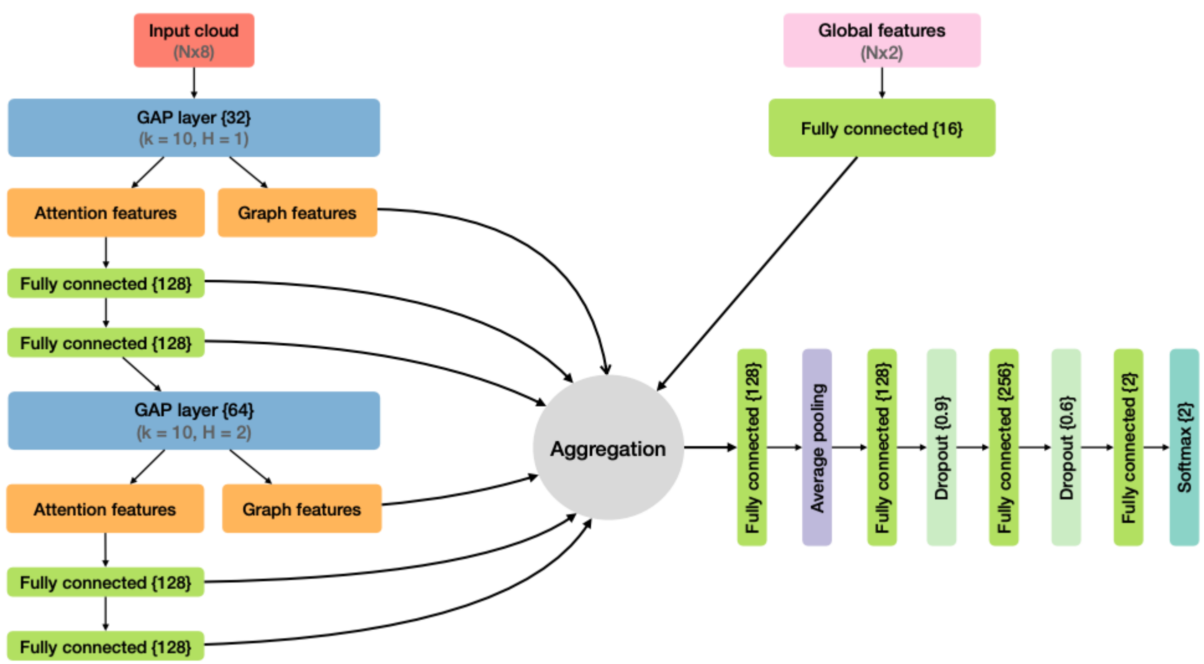


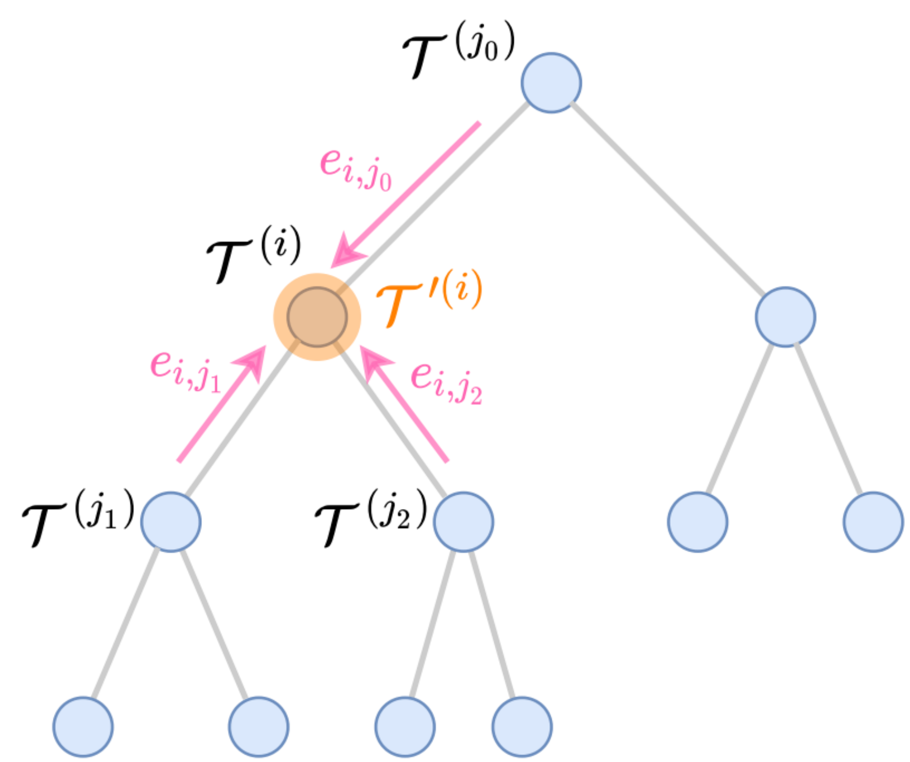




New Ideas

- Work on improving taggers continues
  - Use attention mechanism in graphs to decide which particles are most relevant for given task (Mikuni, Canelli, 2001.05311)

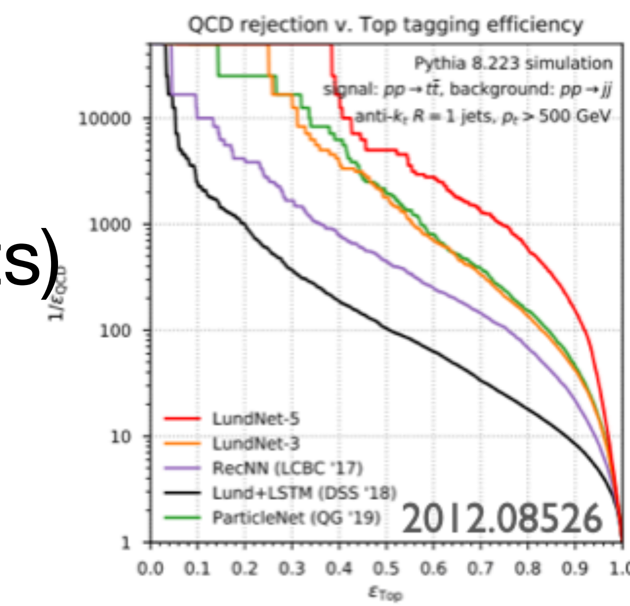


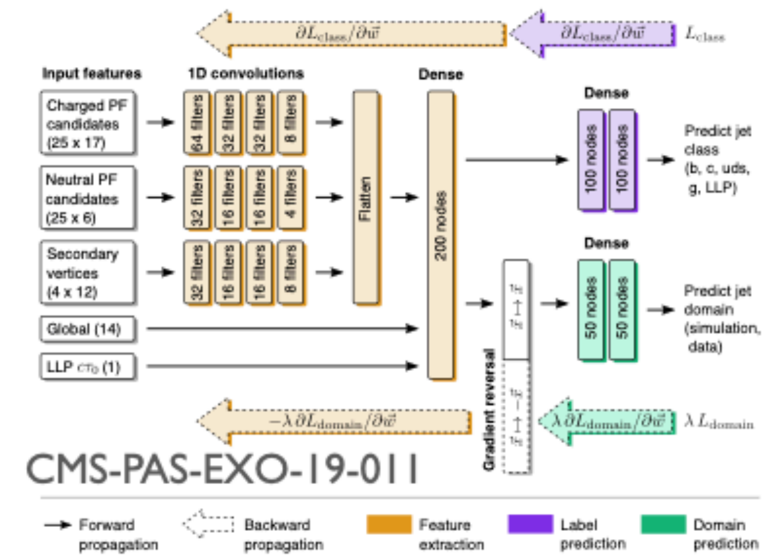


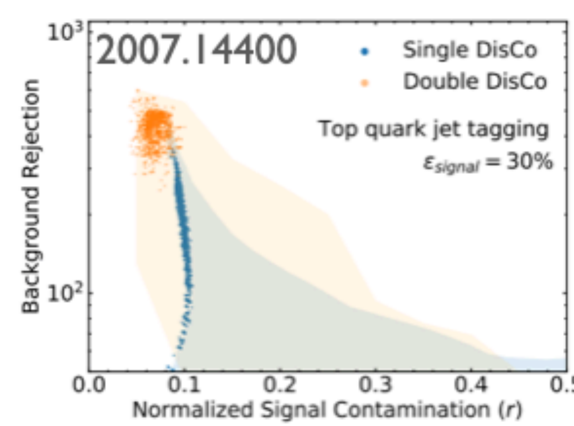
- Apply graph-architecture to jet clustering history in the Lund plane (Dreyer, Qu, 2012.08526)

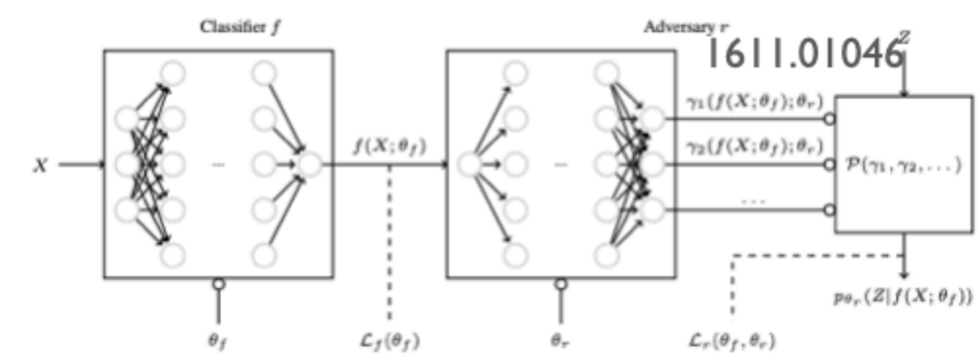
Are we done?

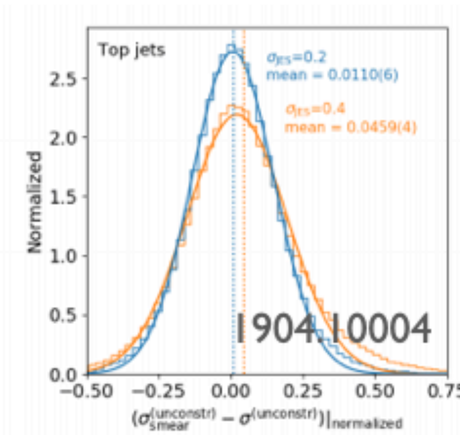
- No (!)
- Need:
  - Higger accuracy
    (easy to measure, many results)
  - **Better stability**
    (domain adaptation issue)

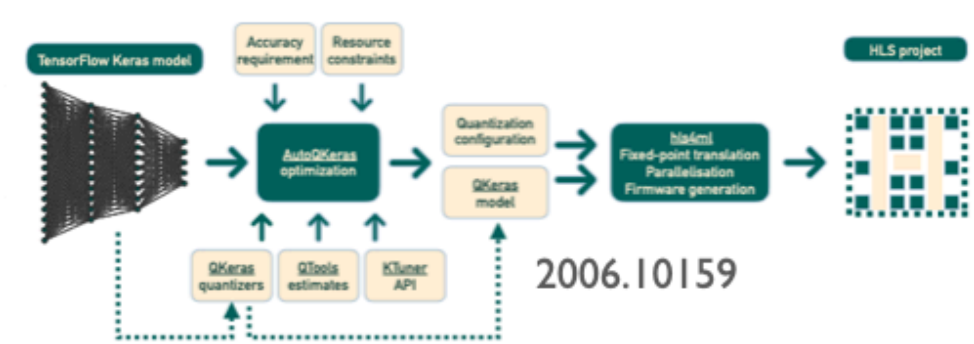






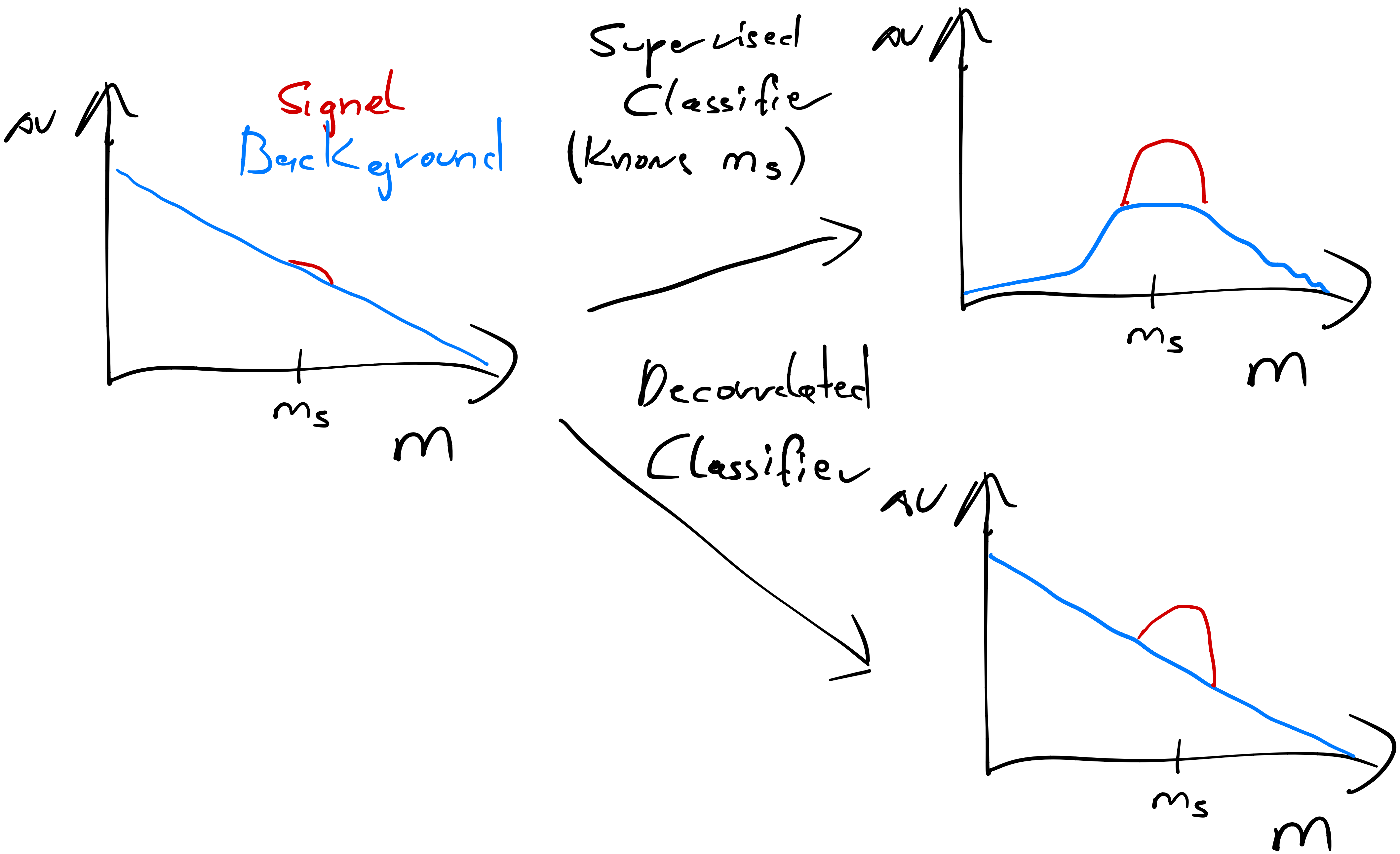






Decorrelation

- Reduce impact of other variables on analysis result
- Remove correlation of classifier output with another variable



How?

- Adversarial training (two competing classifiers) is default approach
  - Unstable / difficult to train
- Find a regulariser term that fulfils the same goal but allows simple training to convergence
  - Use distance correlation (DisCo)

$$\min_{\theta_{\rm clf}} L_{\rm clf}(y(\theta_{\rm clf})) + \lambda C_{\rm reg}(y(\theta_{\rm clf}), m)$$

$$\mathrm{dCorr}^2(X,Y) = \frac{\mathrm{dCov}^2(X,Y)}{\mathrm{dCov}(X,X)\mathrm{dCov}(Y,Y)}$$

$$L = L_{classifier}(\vec{y}, \vec{y}_{true}) + \lambda\,\mathrm{dCorr}^2(\vec{m}, \vec{y})$$

$$x_{jk} = |X_j - X_k|$$ Distances of all examples in batch for classifier output

$$y_{jk} = |Y_j - Y_k|$$ ... for variable to decorrelate

$$\hat{x}_{jk} = x_{jk} - \overline{x}_{j\cdot} - \overline{x}_{\cdot k} + \overline{x}_{\cdot\cdot}$$

$$\hat{y}_{jk} = y_{jk} - \overline{y}_{j\cdot} - \overline{y}_{\cdot k} + \overline{y}_{\cdot\cdot}$$

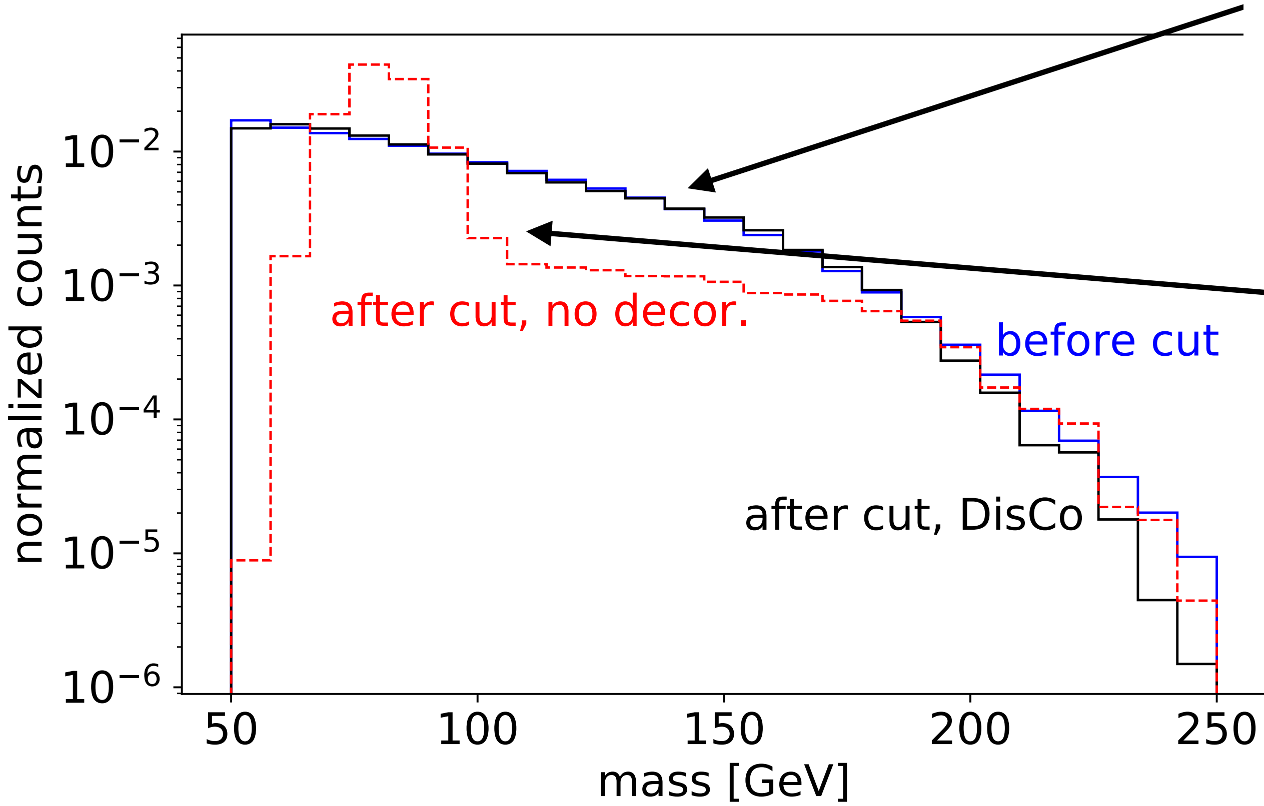
Center distributions

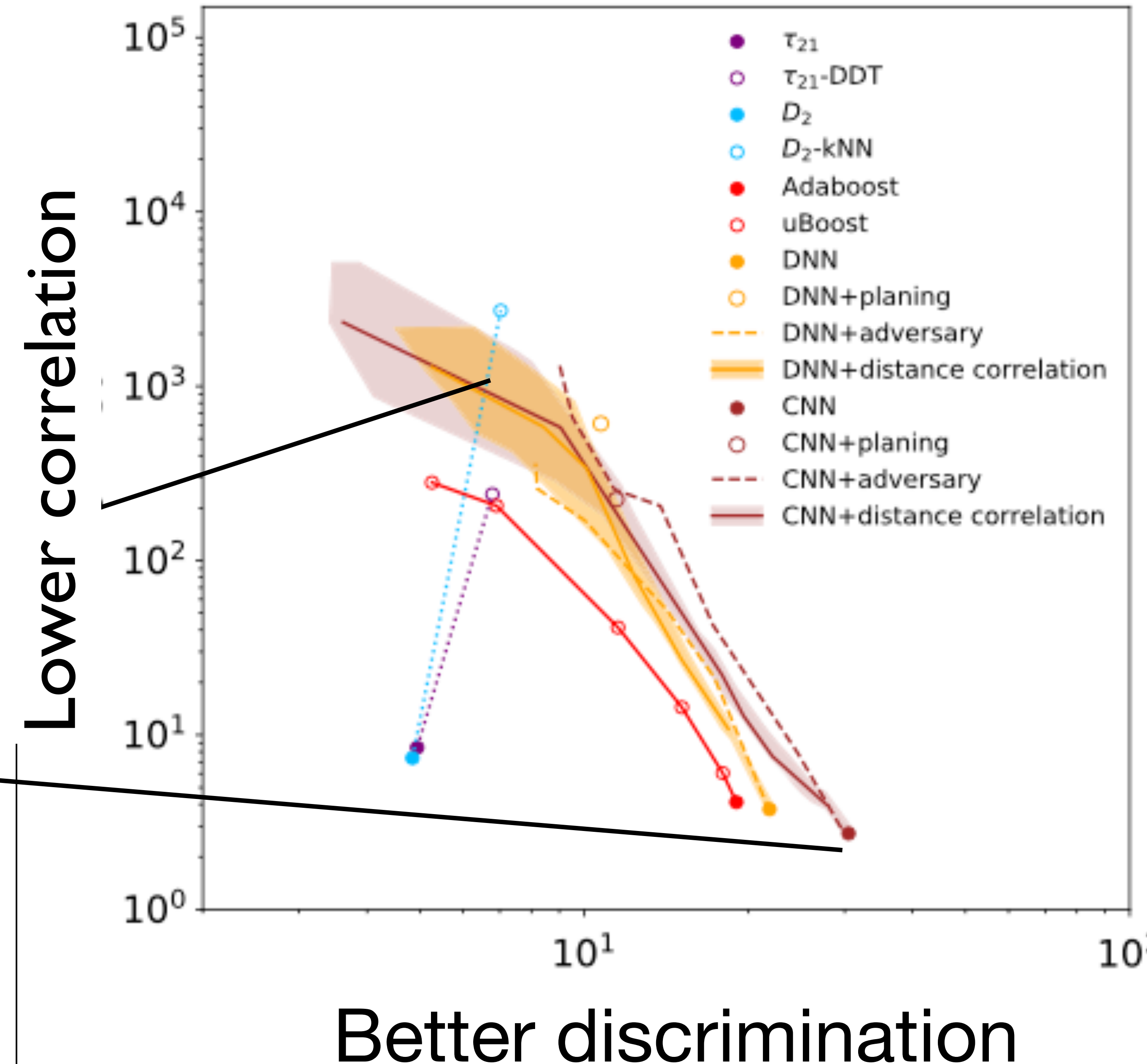
$$\mathrm{dCov}^2 = \frac{1}{n}\sum_j \sum_k \hat{x}_{jk}\hat{y}_{jk}$$

And calculate average product per batch

Result

Our recast of ATLAS study for tagging boosted hadronic W jets using jet substructure

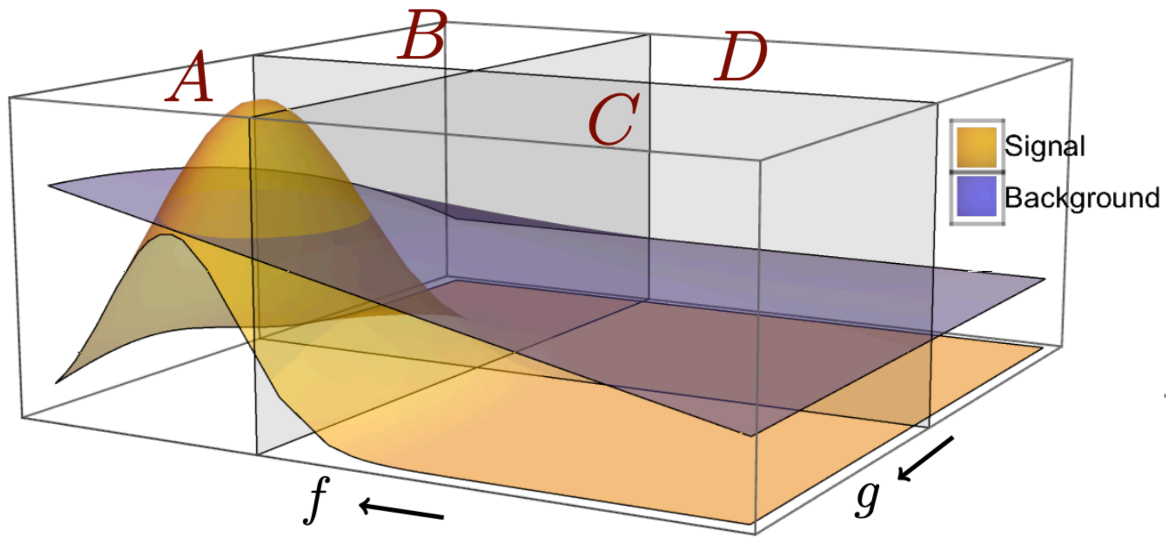




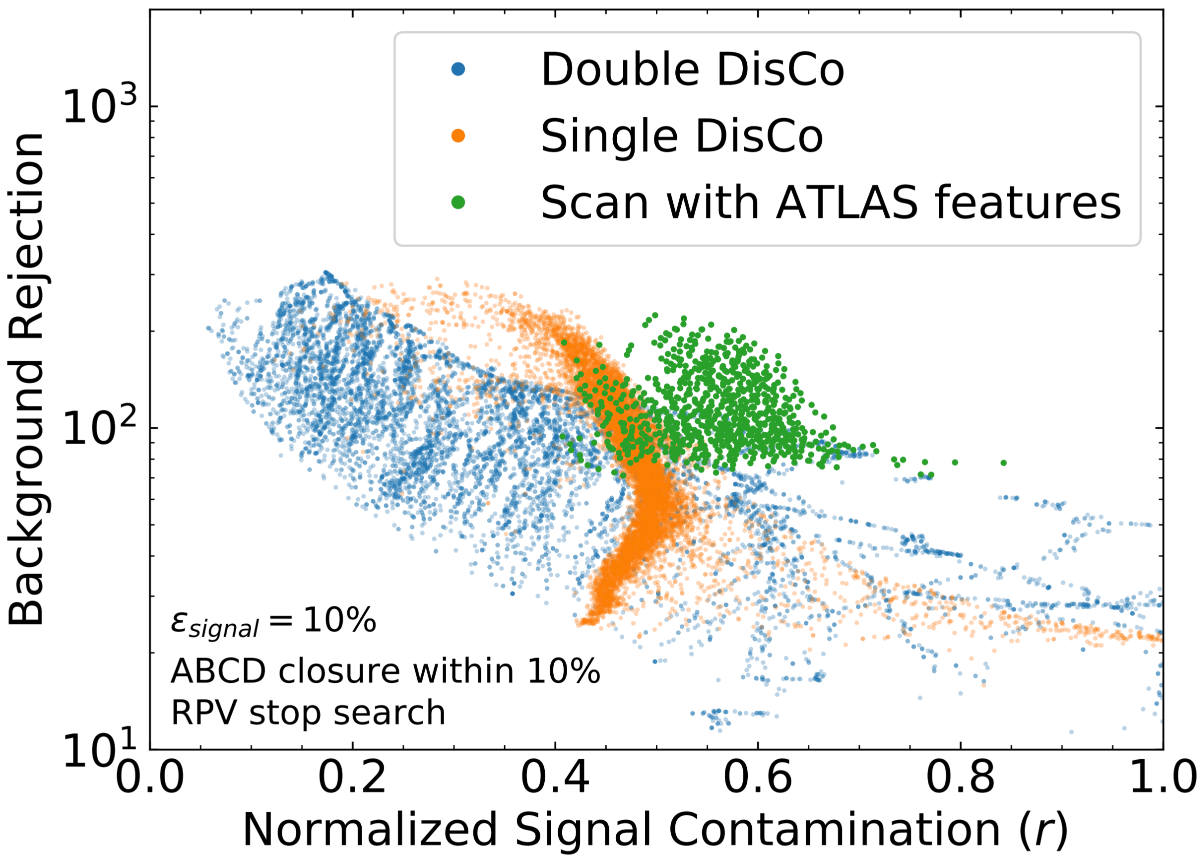
Decorrelation using DisCo achieves same performance as adversarial method, easier to train

2001.05310

ABCDisco



$$N_A = \frac{N_B N_C}{N_D}$$



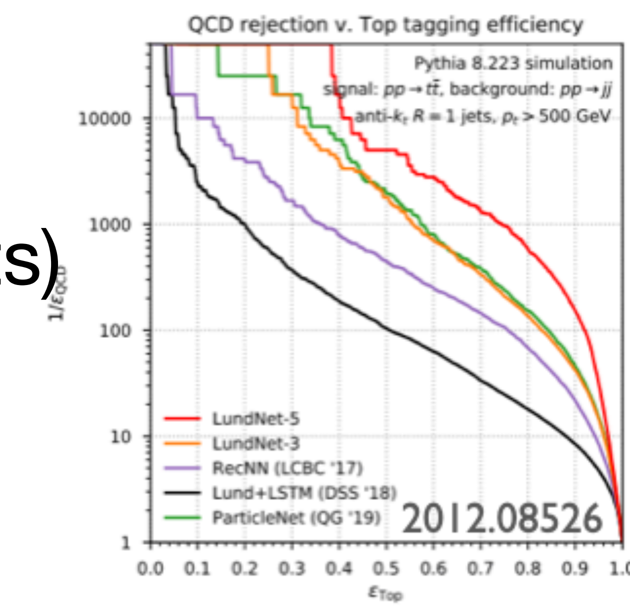
- ABCD method used for background estimation
  - Need two variables so that signal is localised but that are independent for background
- Can we use Disco to train NN for either one or both variables?
  - Recast ATLAS RPV SUSY search for paired dijet resonances (2 squarks to jets)
  - Analysis done using high level kinematic features and angles

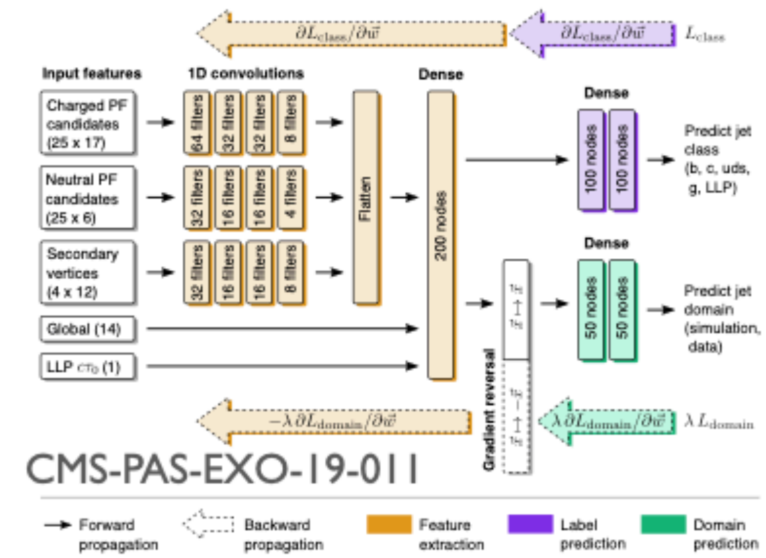
$$\mathcal{L}[f,g] = \mathcal{L}_{\text{classifier}}[f(X),y] + \mathcal{L}_{\text{classifier}}[g(X),y] + \lambda\,\text{dCorr}^2_{y=0}[f(X),g(X)]$$

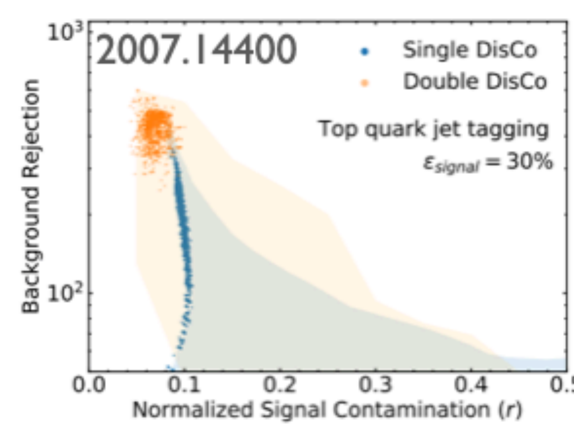
2007.14400

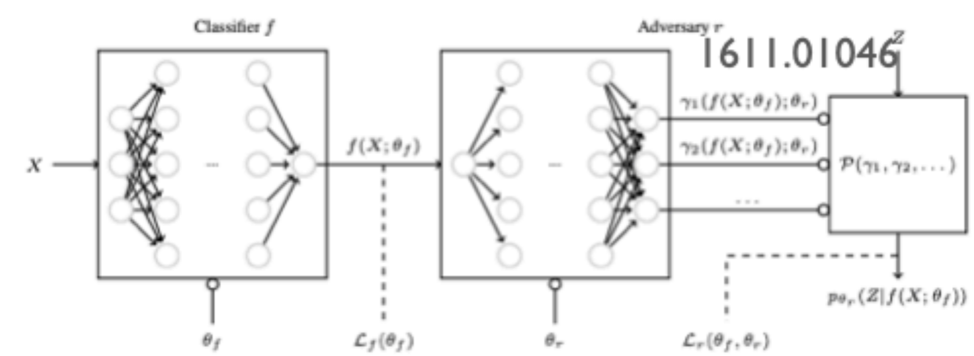
Are we done?

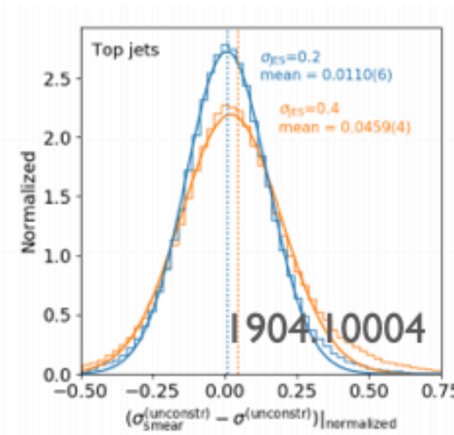
- No (!)
- Need:
  - Higger accuracy
    (easy to measure, many results)
  - Better stability
    (domain adaptation issue)
  - **More control over uncertainties**

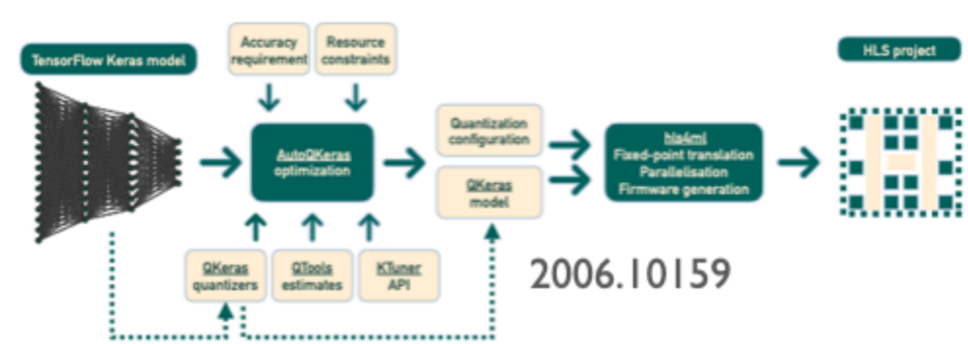






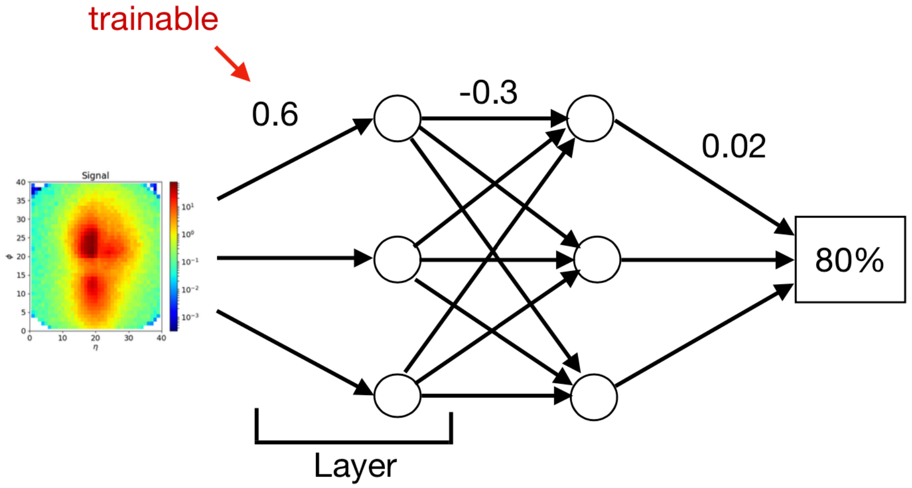




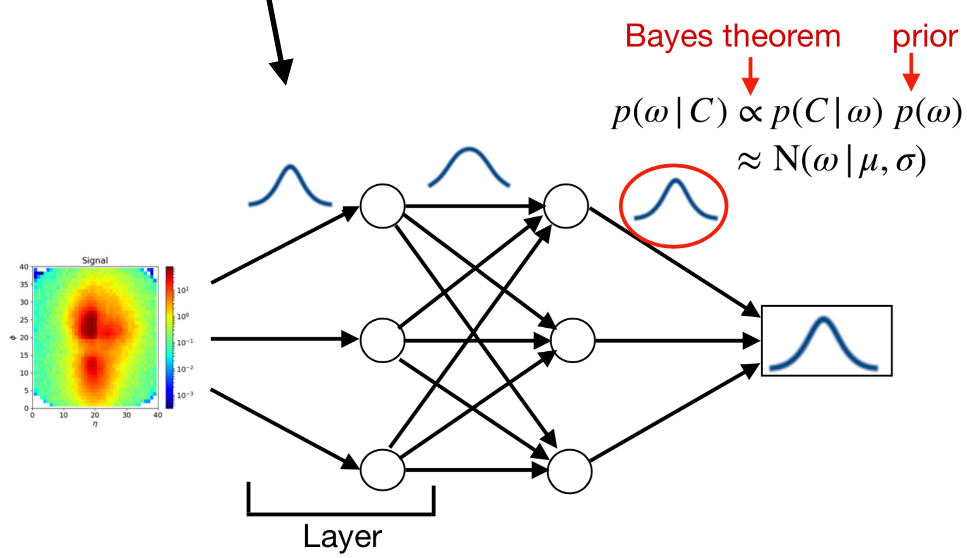


Bayesian Networks

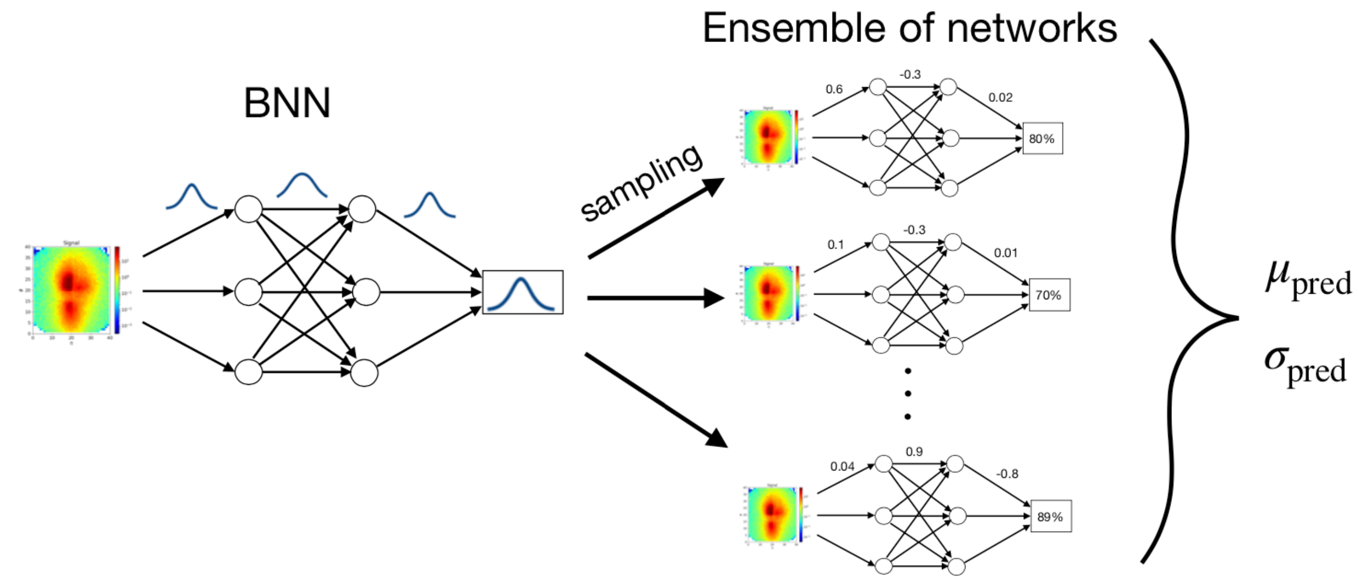
Goal: Quantify uncertainty due to limited trainings statistics



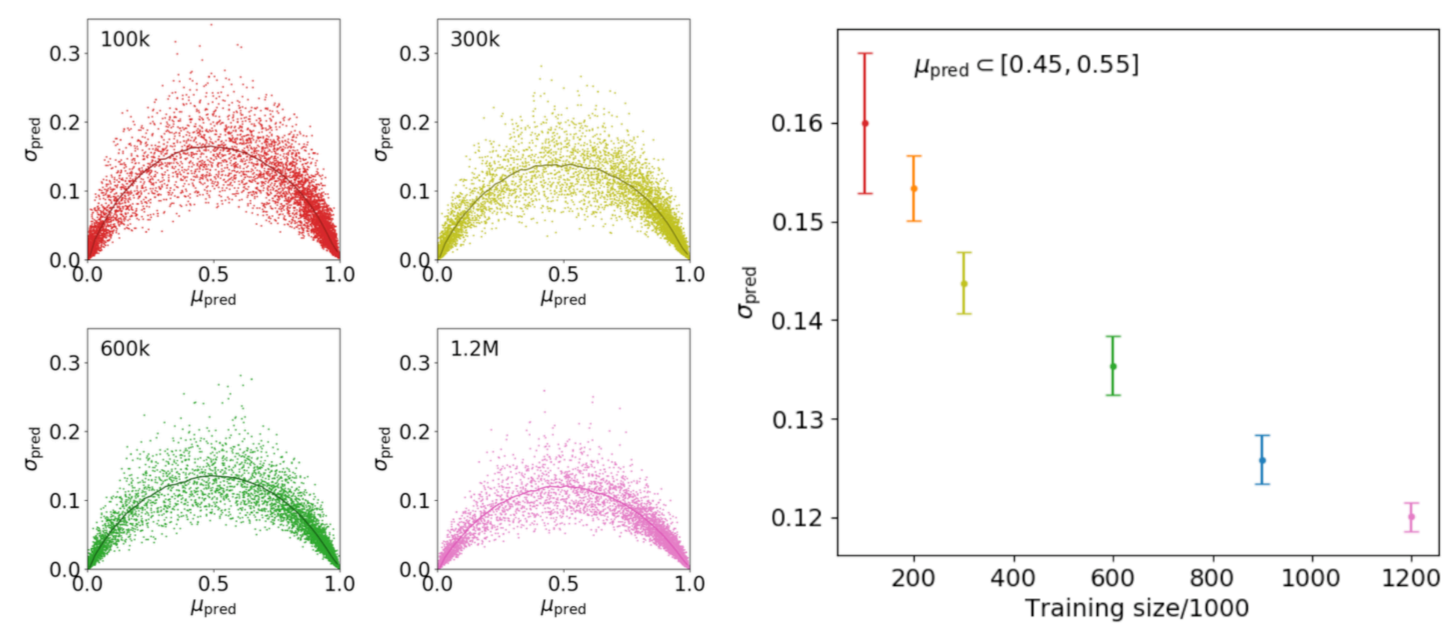
1. *Replace weights with Gaussian PDFs*



2. *Sample network to get prediction+uncertainty*



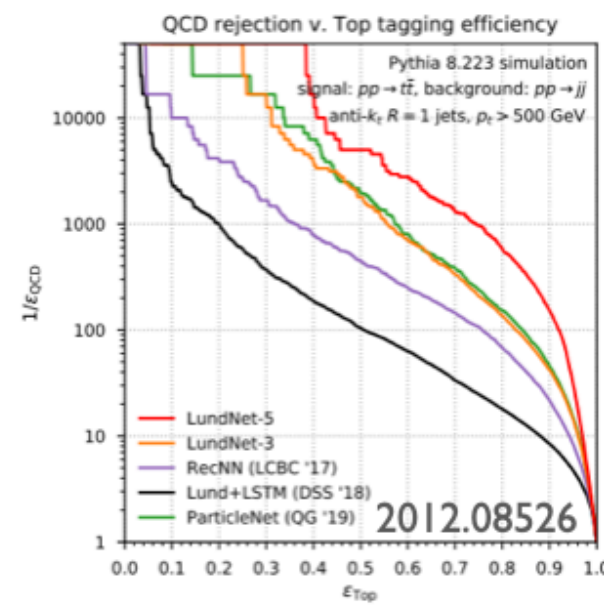
3. *Capture effect of training statistics*

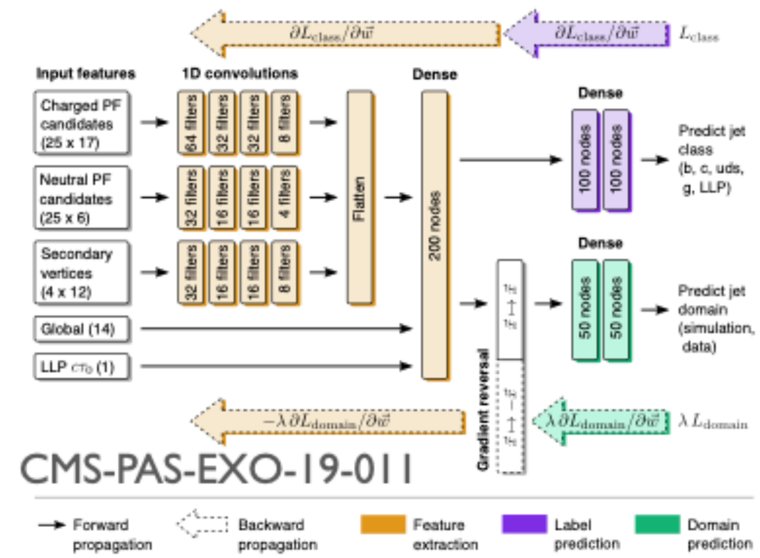


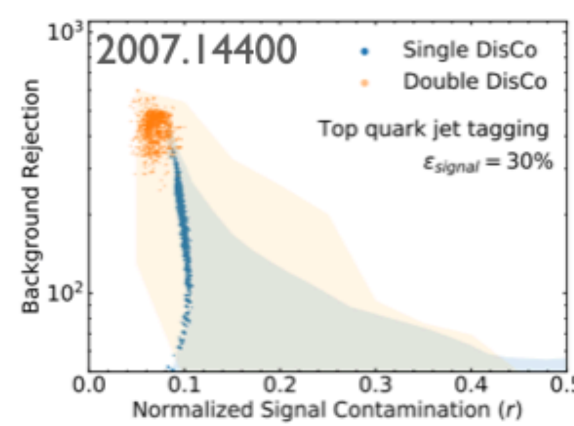
1904.10004, 2003.11099

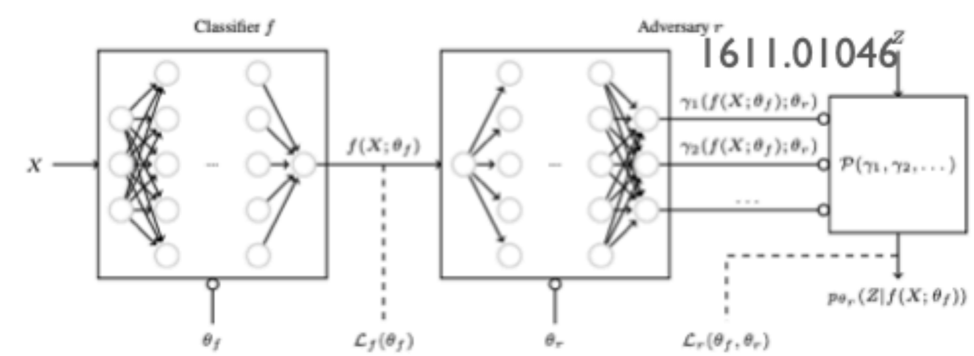
Are we done?

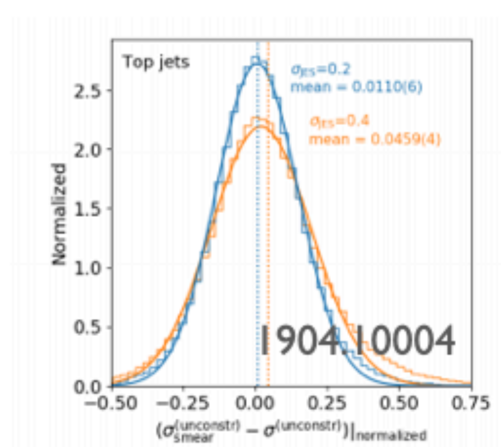
- No (!)
- Need:
  - Higger accuracy
    (easy to measure, many results)
  - Better stability
    (domain adaptation issue)
  - More control over uncertainties
  - Resource efficient implementations
  - Experimental integration
  - **Theoretical understanding / explainability**
  - **More holistic learning**
  - **Problems beyond supervised learning**
  - ….

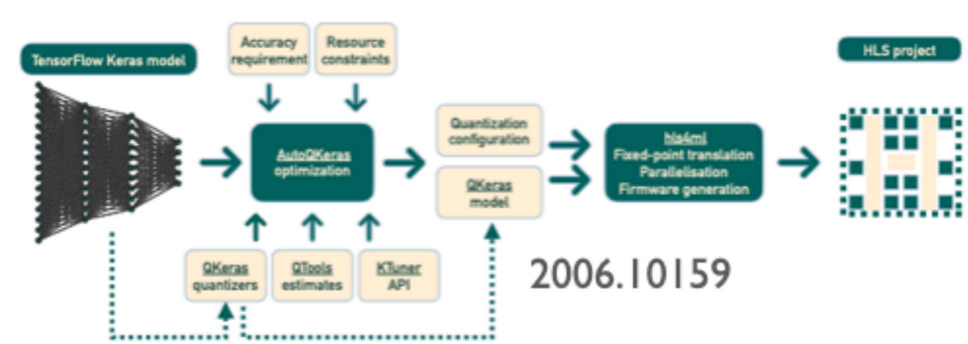












Closing

- Deep Learning in fundamental physics rapidly developing solutions to a wide range of problems
    - Object and Event classification
    - Anomaly detection
    - Robustness and uncertainties
    - Fast reconstruction and simulation
- (Sub-)Jet Physics is leading the way in many regards
    - ***Exciting to see what else we can do!!***

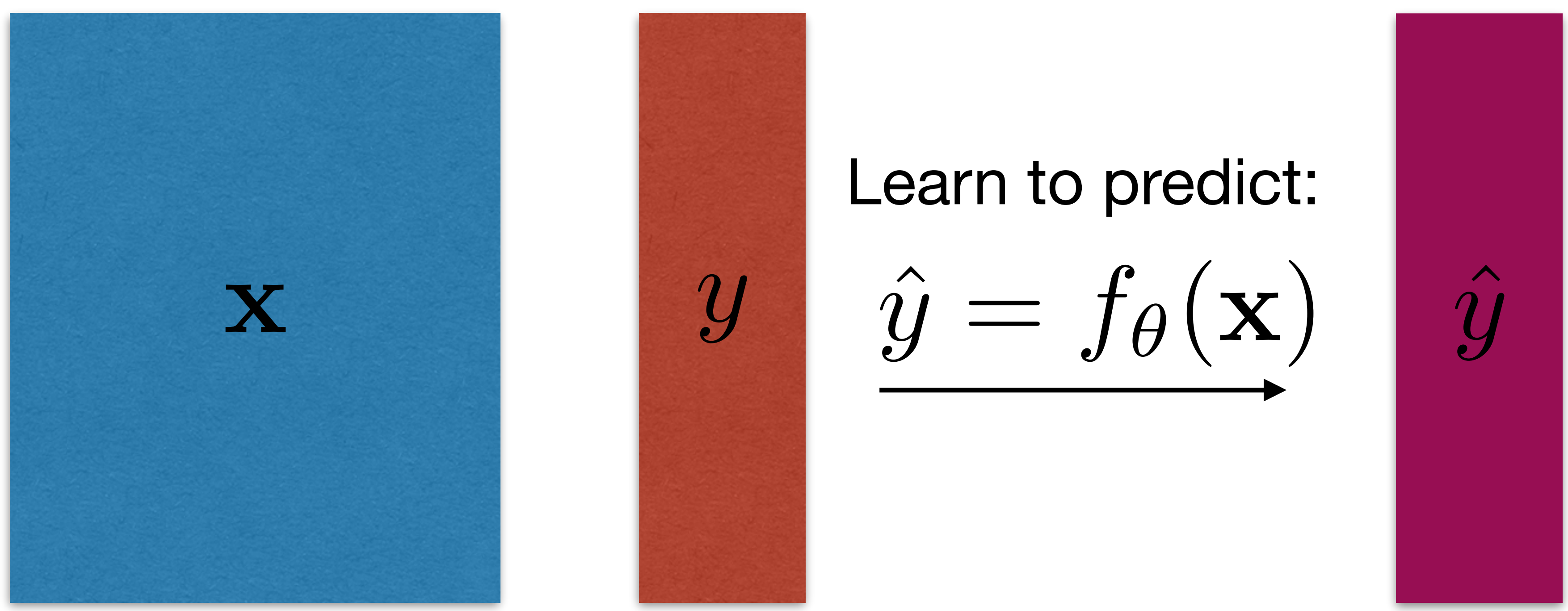
Thank you!

Backup

Loss function: Supervised

Supervised Learning:
Attempt to infer some target *(truth label)*:
classification, regression (often also clustering/inference)

Use training data with known labels
(often from Monte Carlo simulation)



observable features such as kinematics, tracks,…

truth label
(e.g. true energy)

predicted energy

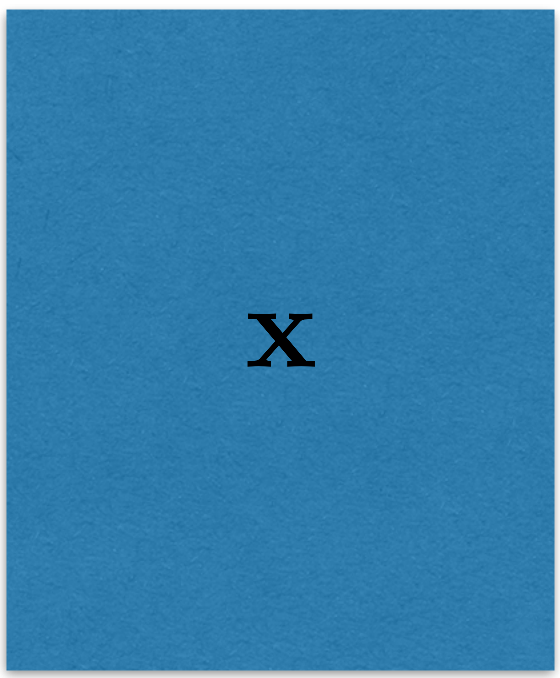
Classification: Minimize cross entropy

$$\mathcal{L} = -y \log(\hat{y}) - (1-y)\log(1-\hat{y})$$

Loss function: Unsupervised

Unsupervised Learning:
No target, learn the probability distribution (directly from data)

Can use for sampling, anomaly detection, unfolding, …



Learn to predict:

$$\hat{p}(\mathbf{x}) = f_\theta(\mathbf{x})$$

$$p(\mathbf{x})$$

True probablity density

Distribution learning: Maximise likelihood (minimize log-likelihood):

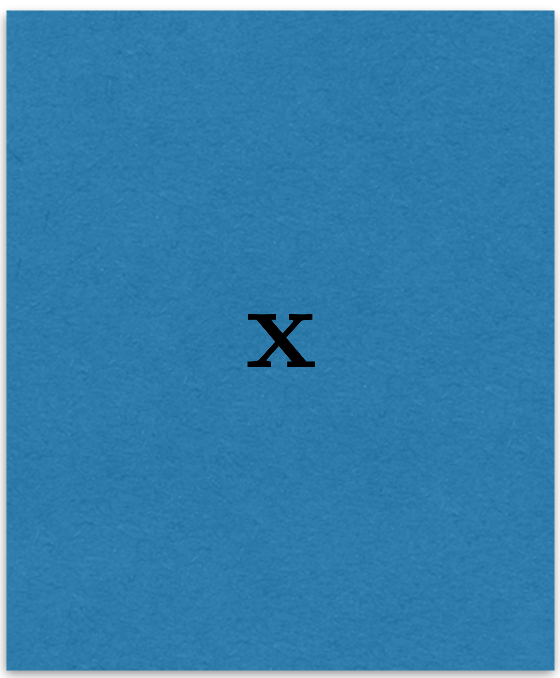
$$\mathcal{L} = -\log\left(\hat{p}(\mathbf{x})\right)$$

Loss function: Unsupervised

Unsupervised Learning:
No target, learn the probability distribution (directly from data)

Can use for sampling, anomaly detection, unfolding, …

There also exists a number of other less-than-supervised approaches (weakly supervised learning, semi-supervised learning, …) Not so important for now.



Learn to predict:

$$\hat{p}(\mathbf{x}) = f_\theta(\mathbf{x})$$

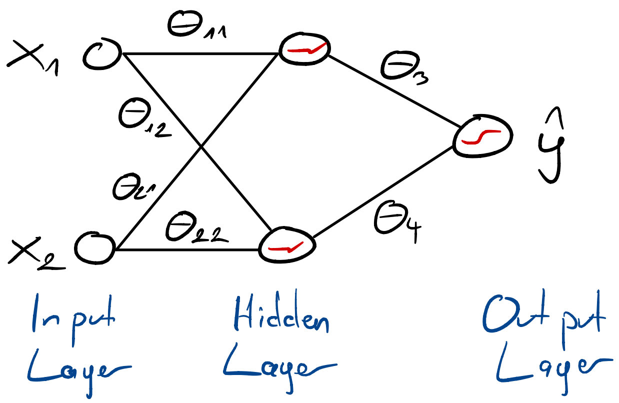
$$p(\mathbf{x})$$

True probablity density

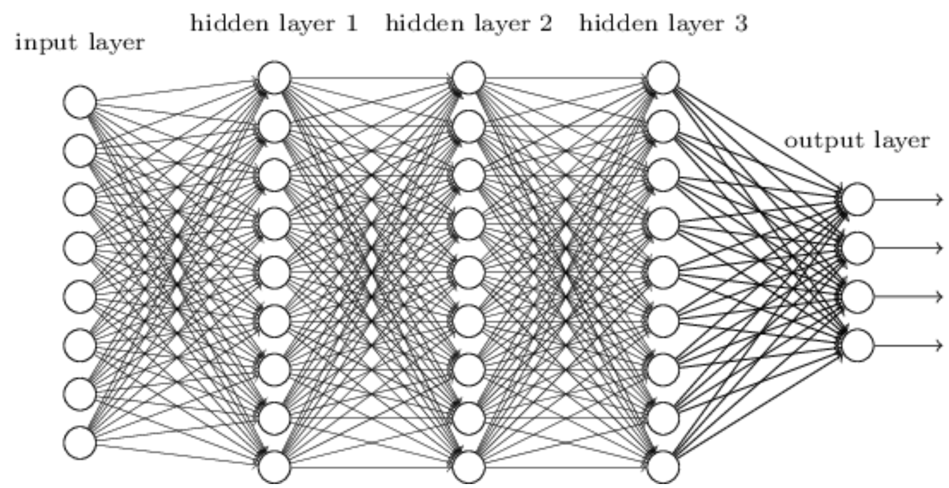
Distribution learning: Maximise likelihood (minimize log-likelihood):
(either directly or with approximations)

$$\mathcal{L} = -\log\left(\hat{p}(\mathbf{x})\right)$$

Complexity



6 weights



300 weights

Deep Learning:
Complex network + low level inputs

25 million weights:
2016 state of the art for image classification

stage	output	ResNet-50		ResNeXt-50 (32×4d)	
conv1	112×112	7×7, 64, stride 2		7×7, 64, stride 2	
conv2	56×56	3×3 max pool, stride 2		3×3 max pool, stride 2	
		[1×1, 64 3×3, 64 1×1, 256]	×3	[1×1, 128 3×3, 128, C=32 1×1, 256]	×3
conv3	28×28	[1×1, 128 3×3, 128 1×1, 512]	×4	[1×1, 256 3×3, 256, C=32 1×1, 512]	×4
conv4	14×14	[1×1, 256 3×3, 256 1×1, 1024]	×6	[1×1, 512 3×3, 512, C=32 1×1, 1024]	×6
conv5	7×7	[1×1, 512 3×3, 512 1×1, 2048]	×3	[1×1, 1024 3×3, 1024, C=32 1×1, 2048]	×3
	1×1	global average pool 1000-d fc, softmax		global average pool 1000-d fc, softmax	
# params.		**25.5**$\times 10^6$		**25.0**$\times 10^6$	
FLOPs		**4.1**$\times 10^9$		**4.2**$\times 10^9$	

175 billion weights: *2020 GPT-3 text model*

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

How do networks learn?

- *Backpropagation + Gradient descent*
- Important: Loss function needs to be differentiable
  - (Or find a differentiable approximation)
- Pass input (x_1, x_2, …) to networks
- From output calculate loss function
  Find gradient of loss function with respect to weights
- Use gradient to find new weights

$$\theta_{t+1} = \theta_t - \eta \frac{\partial \mathcal{L}}{\partial \theta_t} = \theta_t - \eta \nabla \mathcal{L}$$

Learning rate

- Practically, this is taken care of by an optimiser algorithm
  (e.g. Adam as default)