



# Machine Learning in Particle Physics

going beyond classification and regression

Michael Spannowsky  
IPPP, Durham University







# Humans vs Machines

- **2015 Image Classification:**

- K. He et al (Microsoft Research), *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*, 1502.01852

- **2016 Go:**

- Alpha Go (D. Silver et al, *Mastering the game of Go with deep neural networks and tree search*, Nature 529, pp484–489 and D. Silver et al *Mastering the game of Go without human knowledge*, Nature 550, pp354–359)

- **2016 Speech recognition:**

- W. Xiong et al (Microsoft Research) *Achieving Human Parity in Conversational Speech Recognition*, 1610.05256

- **2017 Poker** (heads-up no-limits Texas Hold'em):

- N Brown and T Sandholm, *Superhuman AI for heads-up no-limit poker: Libratus beats top professionals*, Science 359, Issue 6374, pp418-424

- **2018 Translation** (Chinese-English)

- H H Awadalla et al (Microsoft AI & Research) *Achieving Human Parity on Automatic Chinese to English News Translation*

- **20?? Particle Physics**



# Machine Learning

=

The scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead.  
(Wikipedia)

In other words: Algorithm-based extraction of correlations between input and output

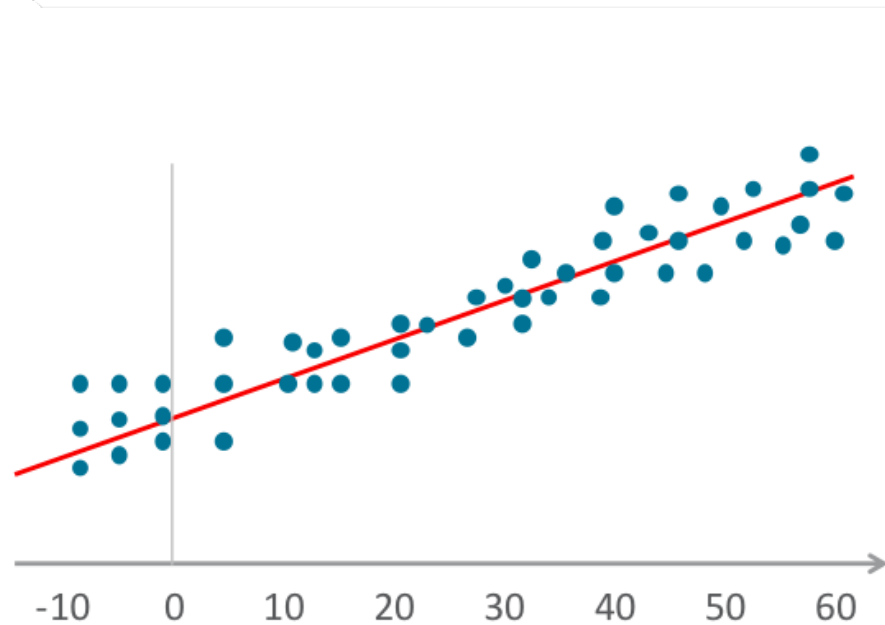
In particle physics, mostly used in context of  
**Regression and Classification**

(upcoming clustering and anomaly detection)

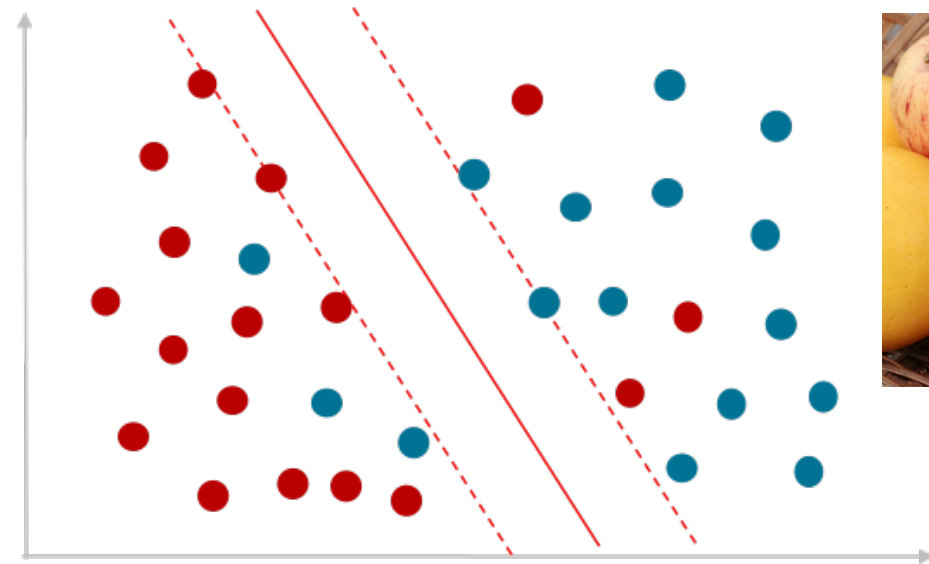


# Supervised

## Regression

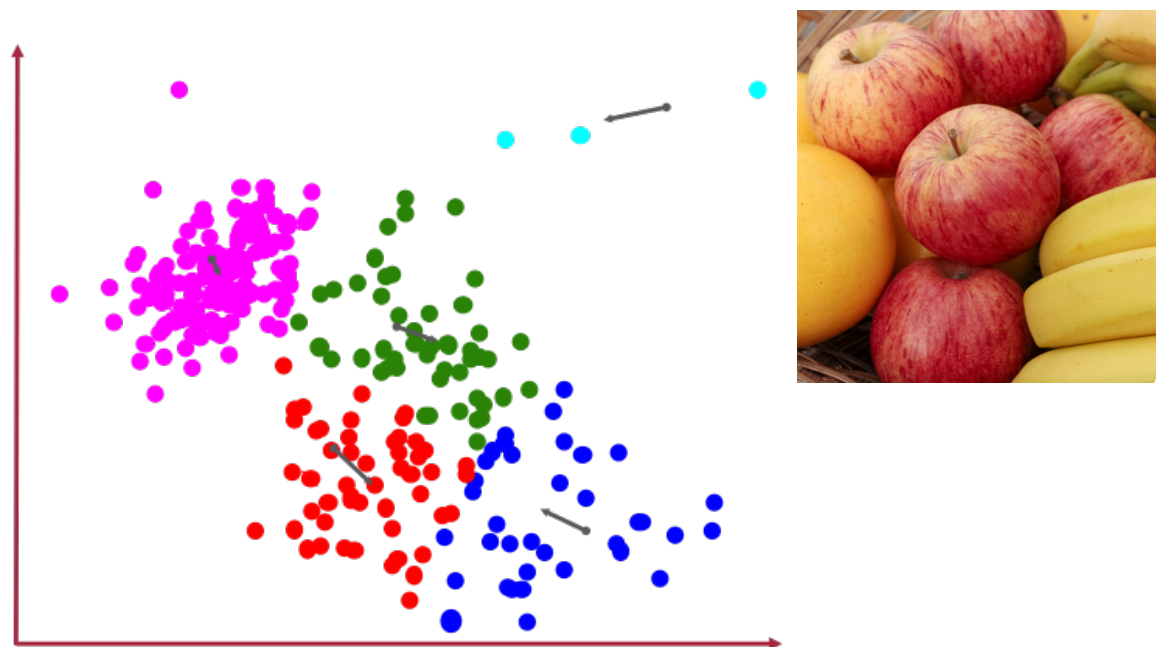


## Classification



# Unsupervised

## Clustering

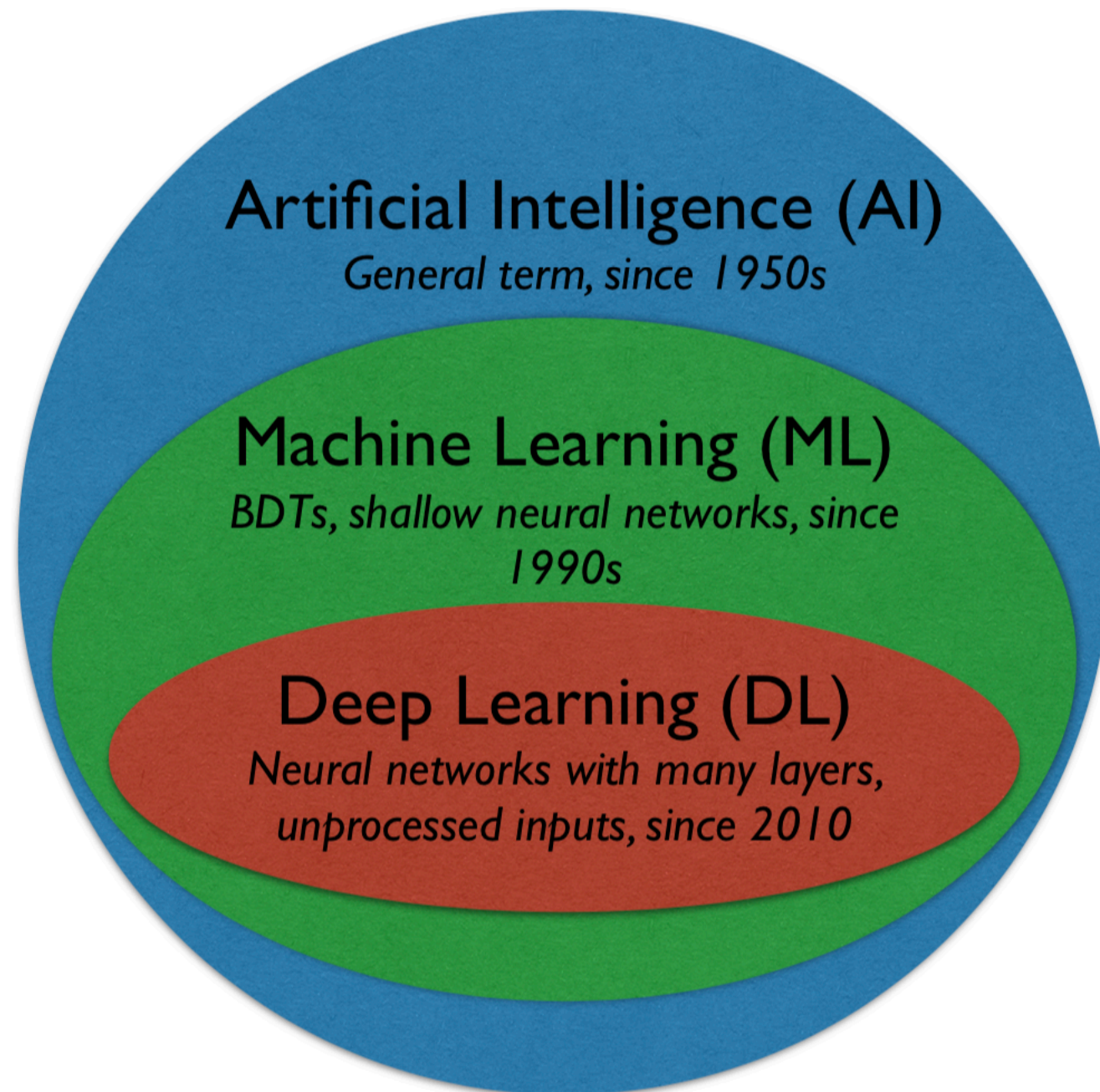


## Anomaly Detection





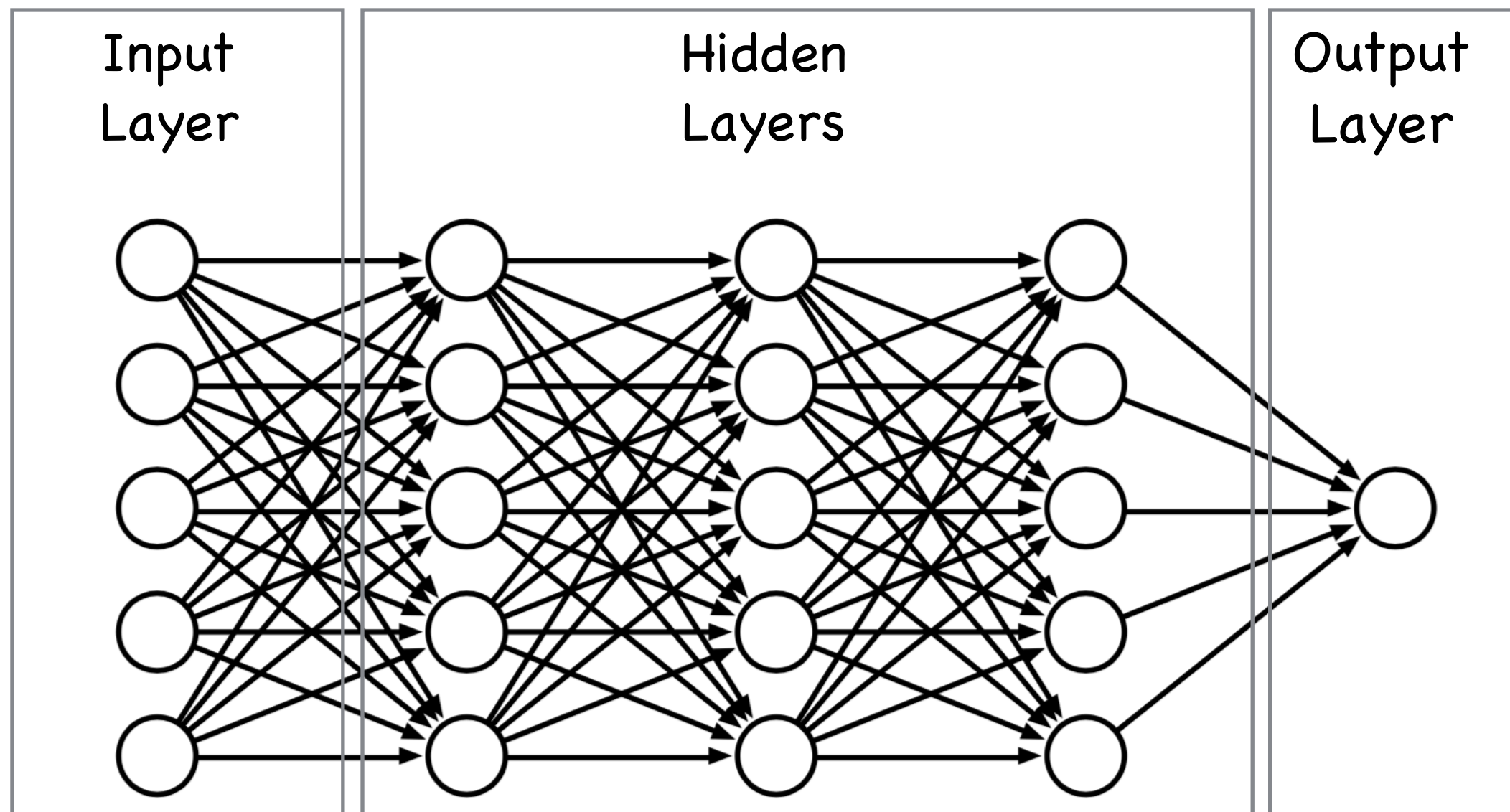
# Not new, but HIP - even in particle physics



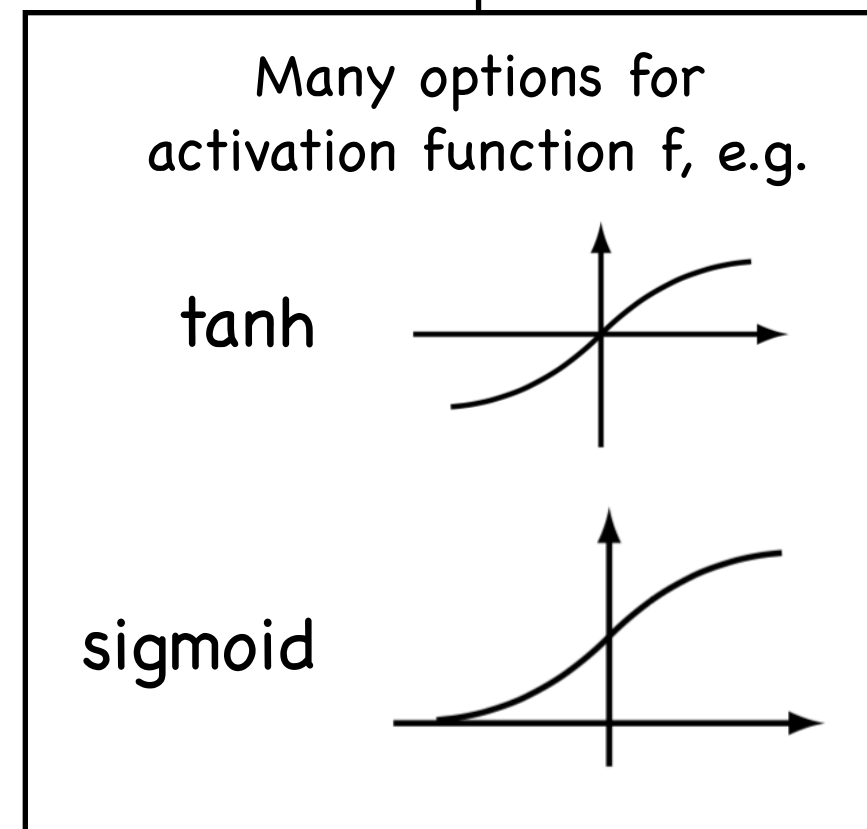
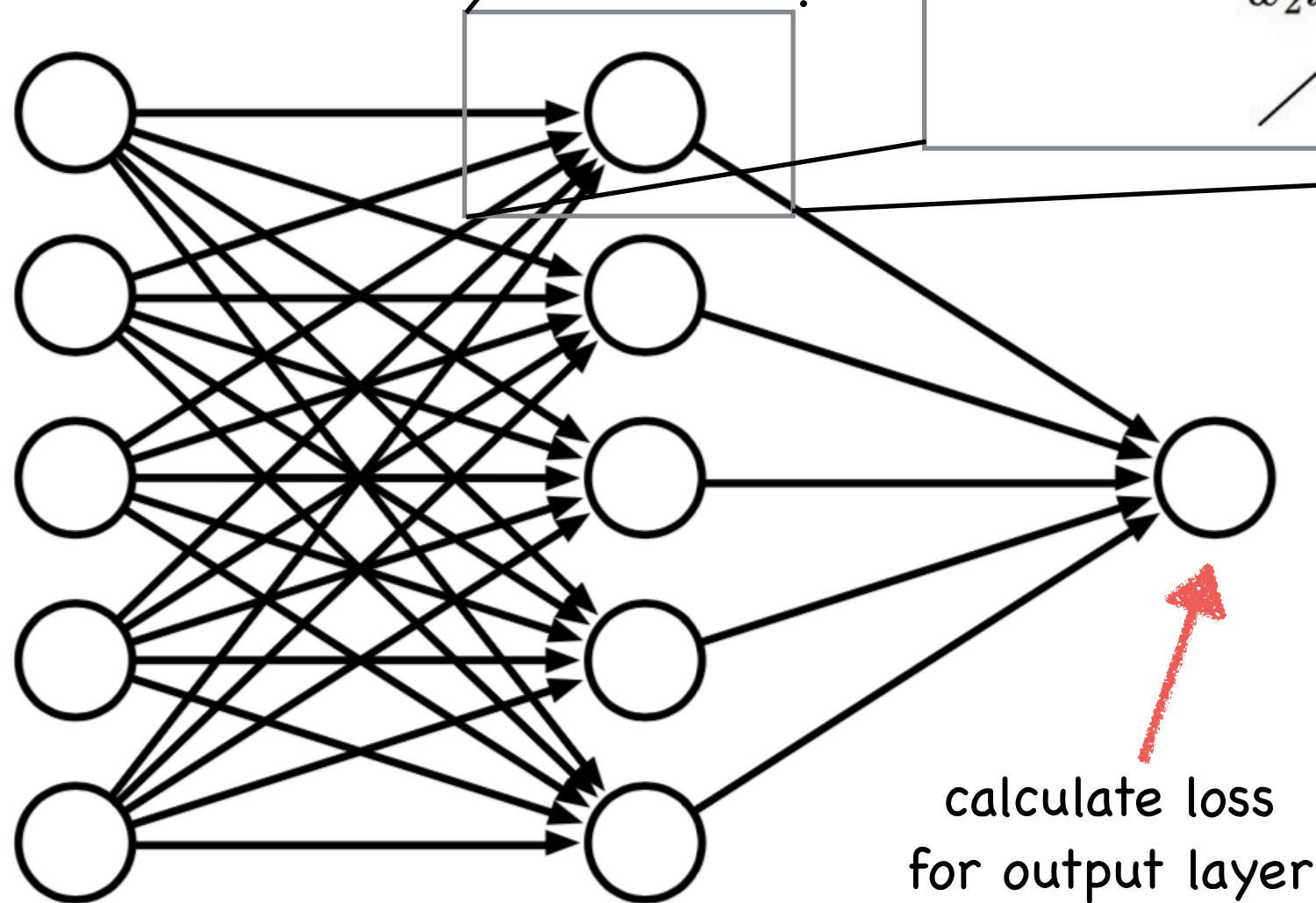
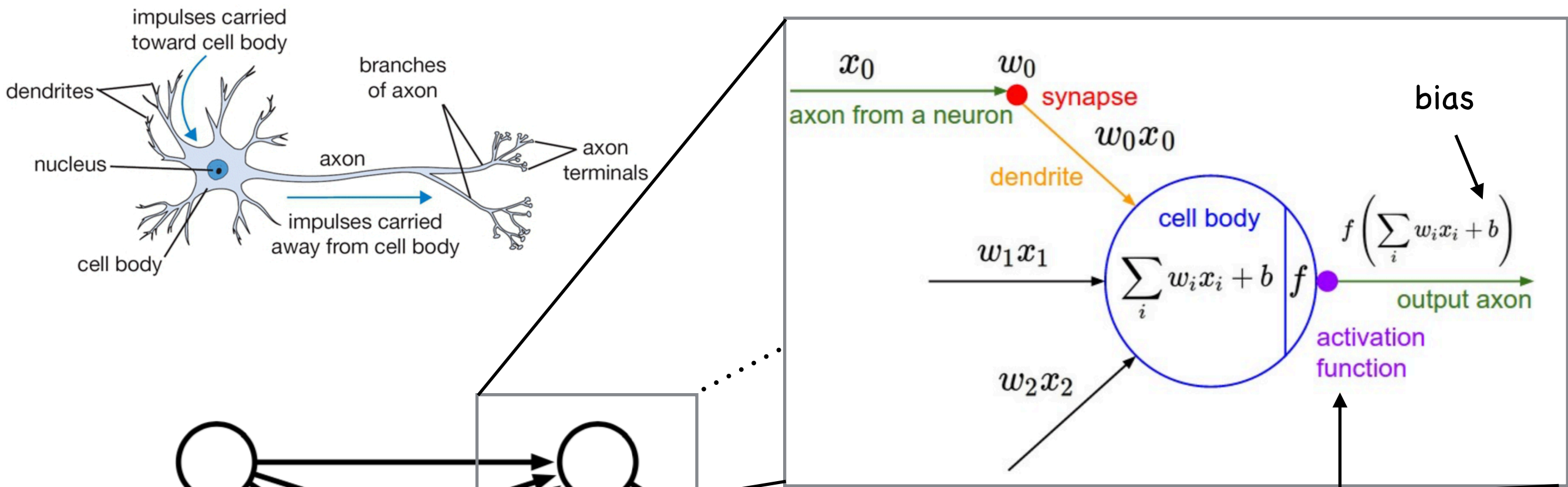


# Machine-Learning with Deep Neural Networks

- Regression and Classification = correlating input and output
- Various applications in particle physics









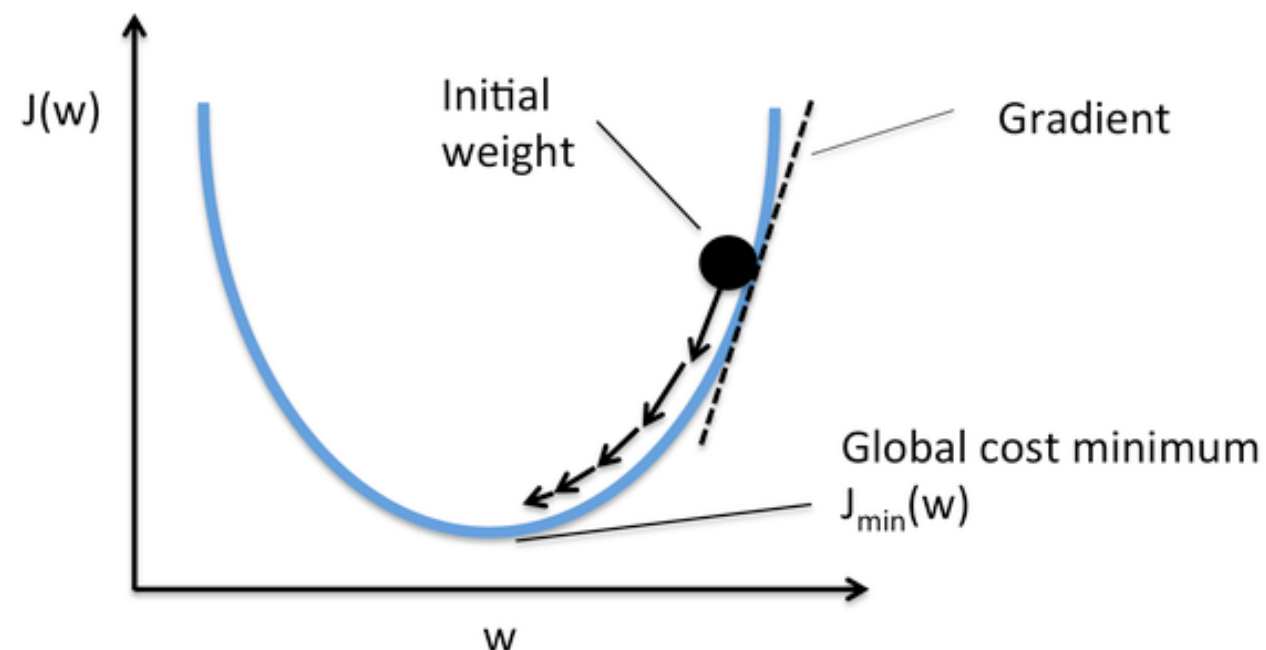
# Gradient descent

- After forward propagation, ie. establishing the weights for all nodes (including the output node), we evaluate the **loss function**, to establish the error we are making.

Loss function = difference between predicted and true function,

$$\text{e.g. } E(y, y') = \frac{1}{2} |y - y'|^2$$

- Gradient descent: change network such, that you move towards the error minimum.
- Compute gradient  $\rightarrow$  get direction towards error minimum.



# Learning via backpropagation

- Backpropagation is method to compute the partial derivative of the loss function  $E(y, y')$ . It is about determining how changing the weights impact the overall loss in the NN

- variation of loss with respect to weight  $w_k$  of NN is

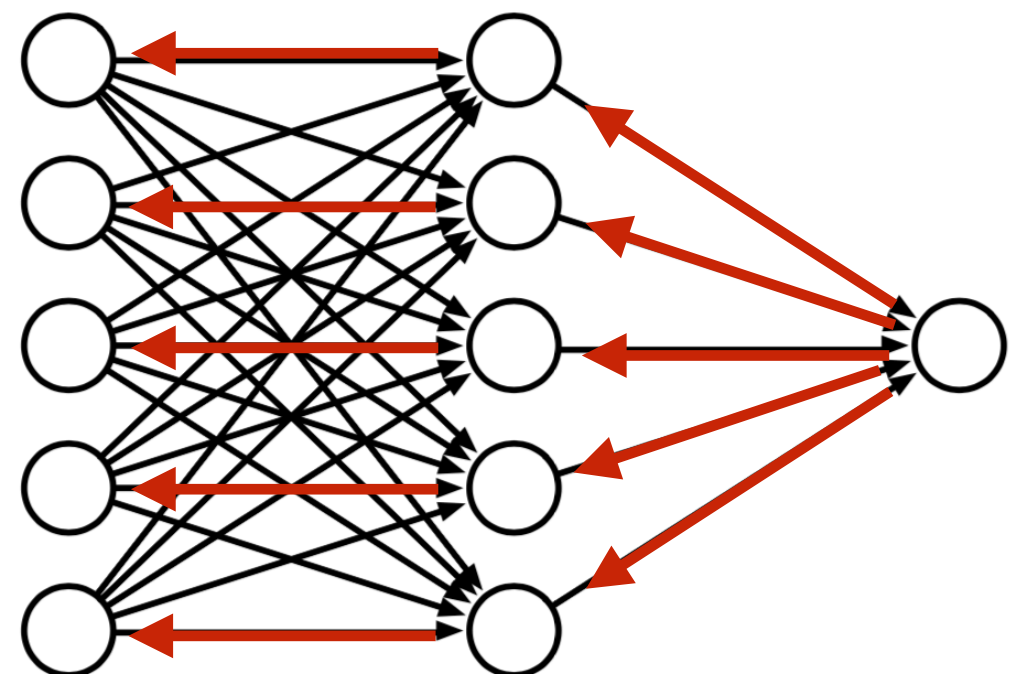
$$\frac{dE}{dw_k} = \frac{dE}{dy} \frac{dy}{ds} \frac{ds}{dw_k} \quad \text{with}$$

comb of weights  $s = \sum_k w_k h_k$   
activation function  $y$

- weights of network adjusted by learning rate  $\mu$

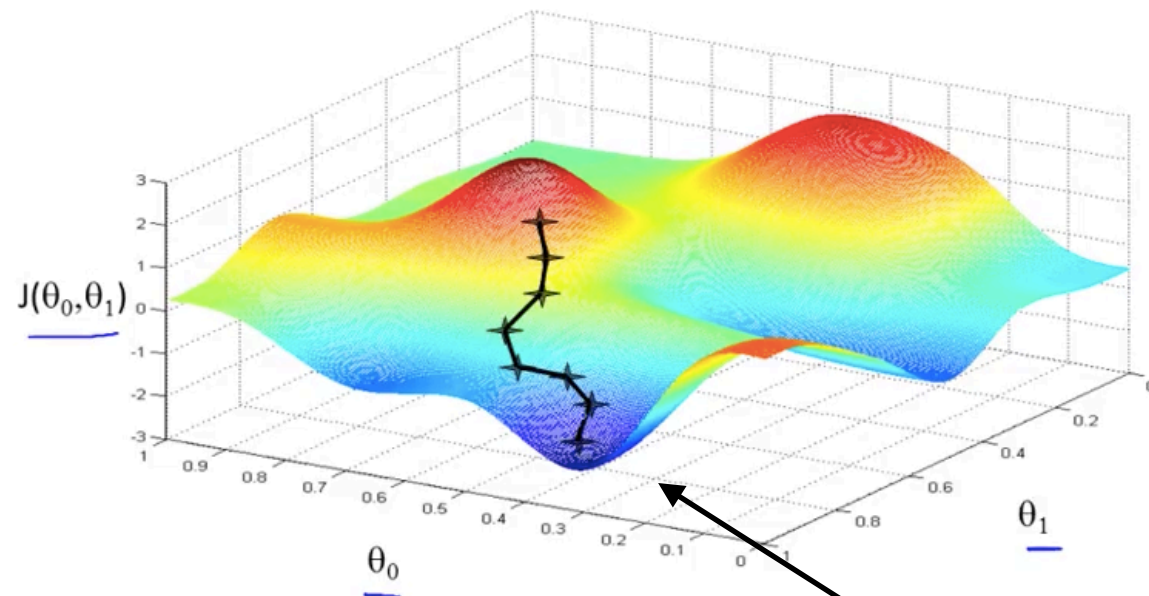
$$\Delta w_k = -\mu \frac{dE}{dy} \frac{dy}{ds} \frac{ds}{dw_k}$$

- New network weights reduce value of loss function





# Neural Net

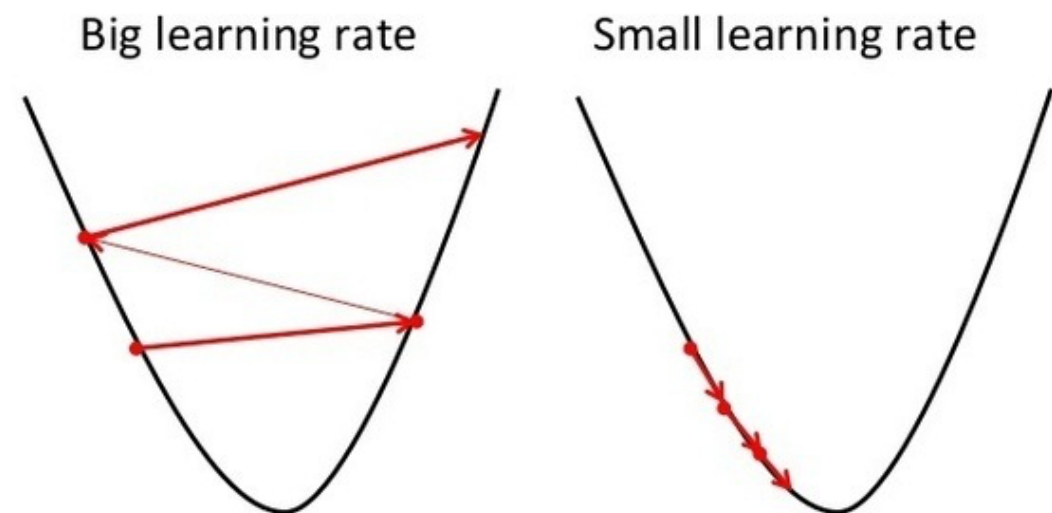


All a Neural Net does is give an analytic expression for the loss function, and then attempts to minimise the loss function using gradient descent and backpropagation

Overall, optimisation problem. Need to optimise hyperparameters to problem, e.g. learning rate

...

## Gradient Descent



# Application often to jet physics, i.e. top, W, Z, H vs lf jets

order of recombination defined by some metric, e.g. kT distance

$$d_{j_1 j_2} = \frac{\Delta R_{j_1 j_2}^2}{D^2} \min(p_{T,j_1}^2, p_{T,j_2}^2)$$

Jet

Subjects

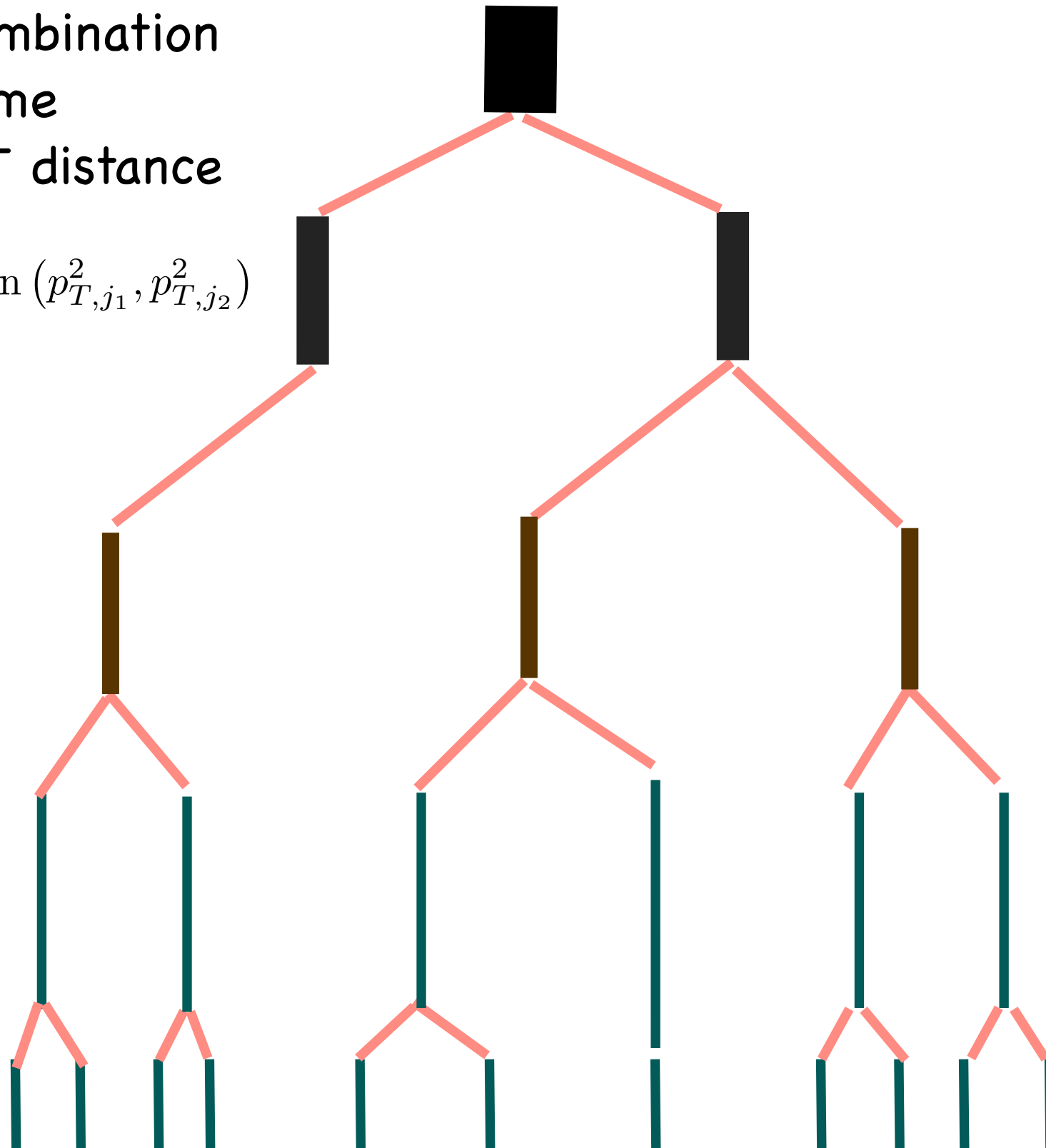
Jet constituents

First AEI BSM Conference Jeju

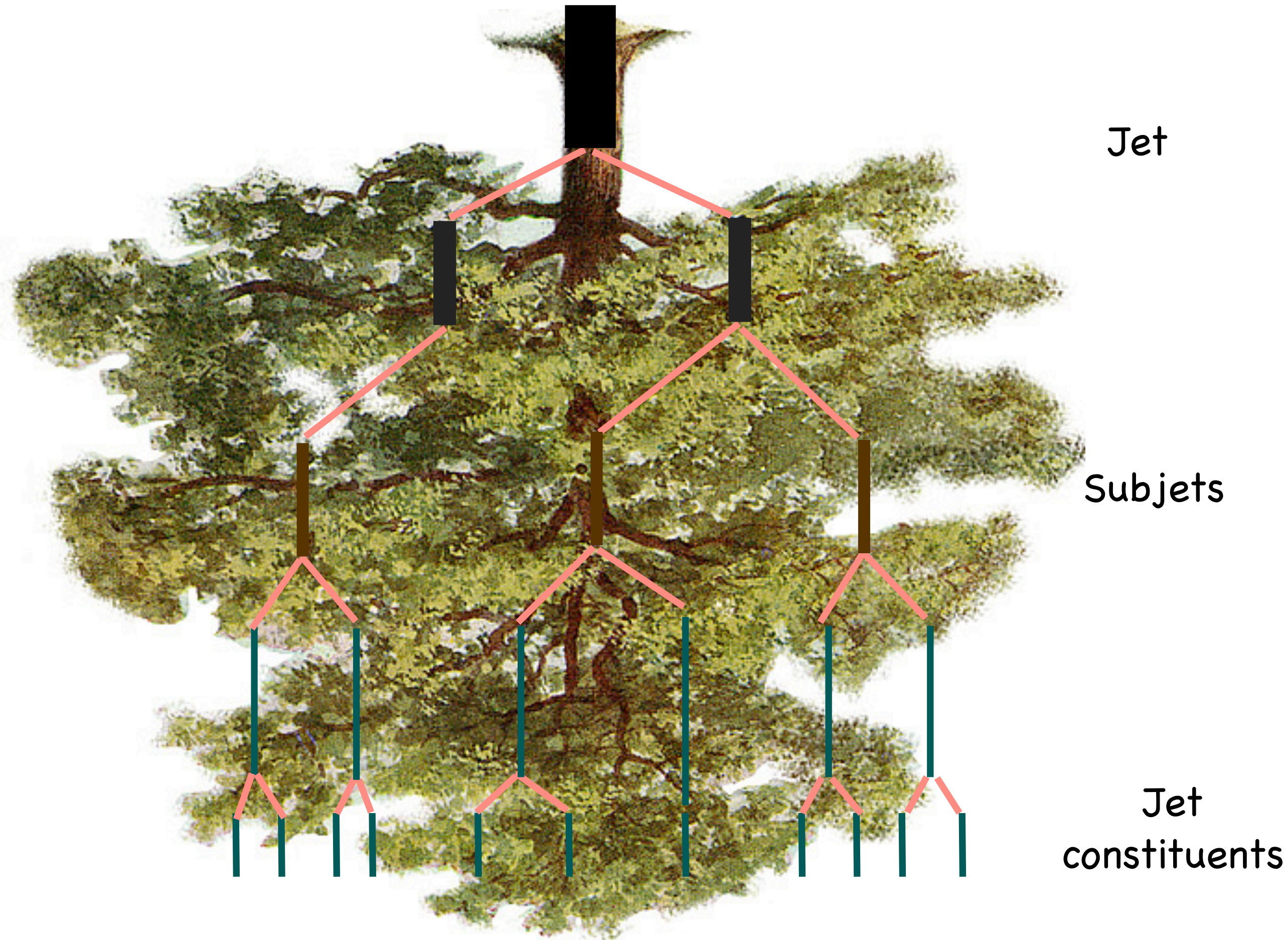
122

Michael Spannowsky

07.11.2019







Jet

Subjects

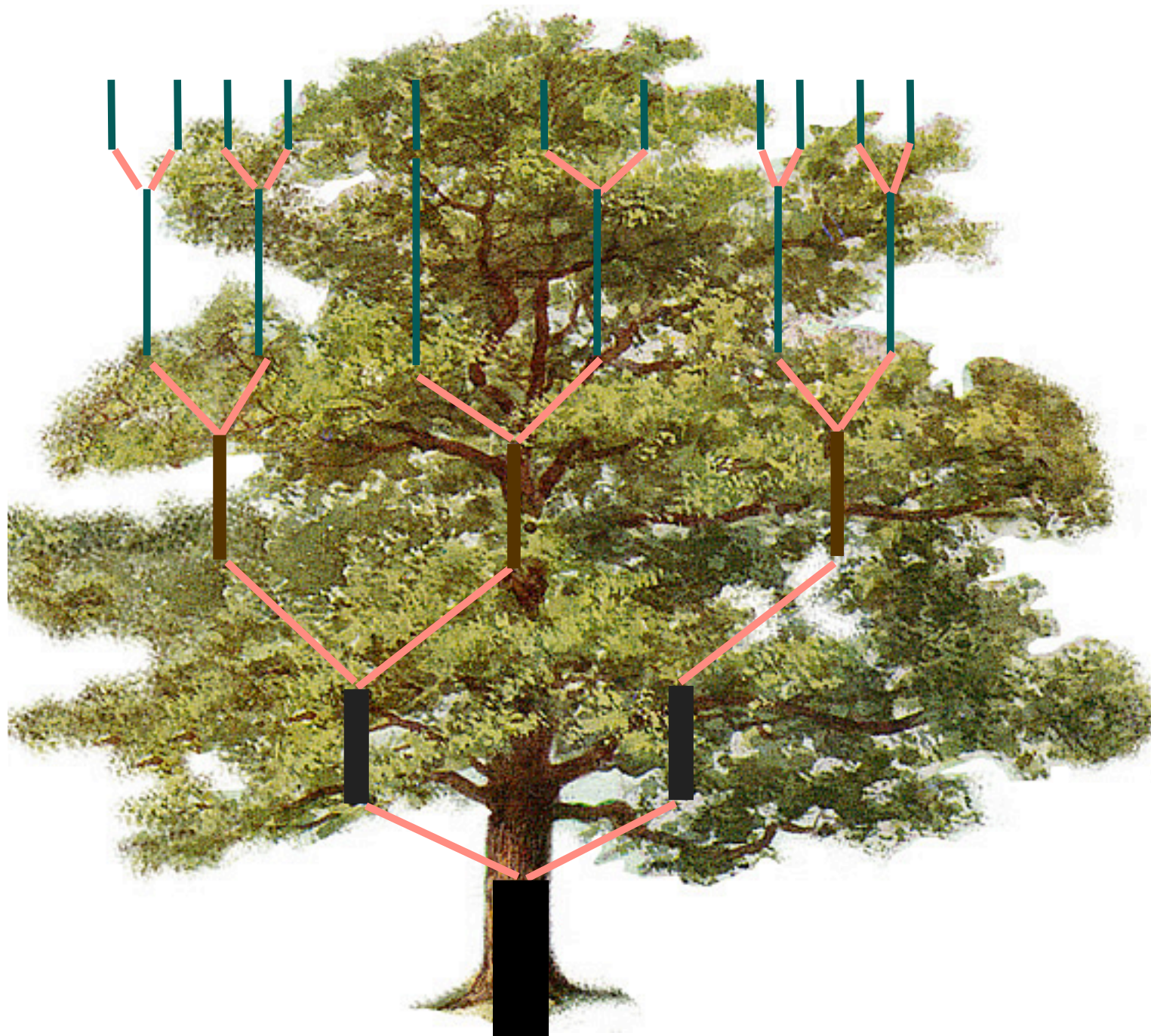
Jet constituents



Intrinsic scales and energy sharing of jet substructure different

→ Can be used to discriminate QCD jet from resonance jet

W/Z/H-boson jet



QCD jet





# What are some of the common Neural Nets

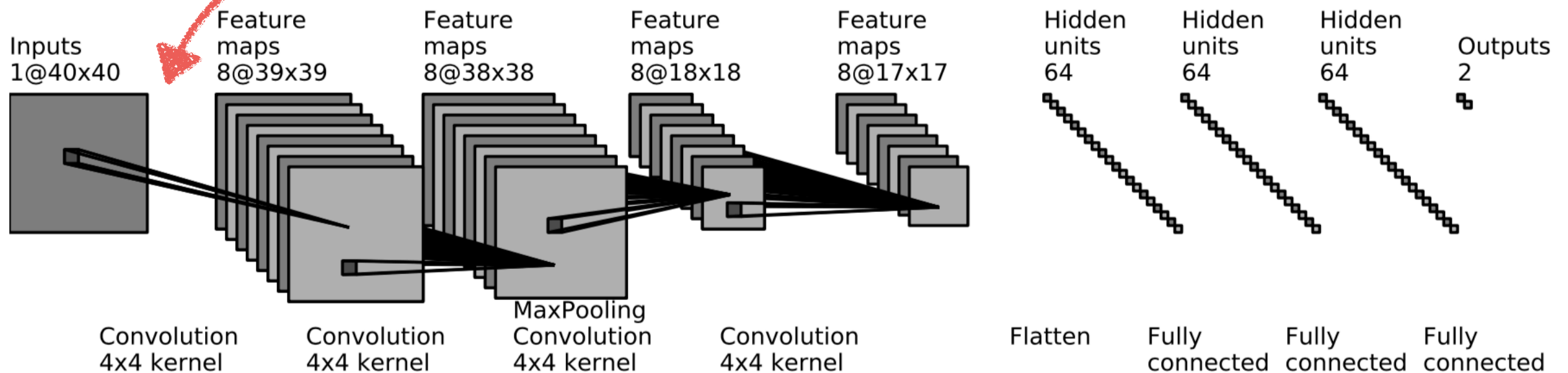
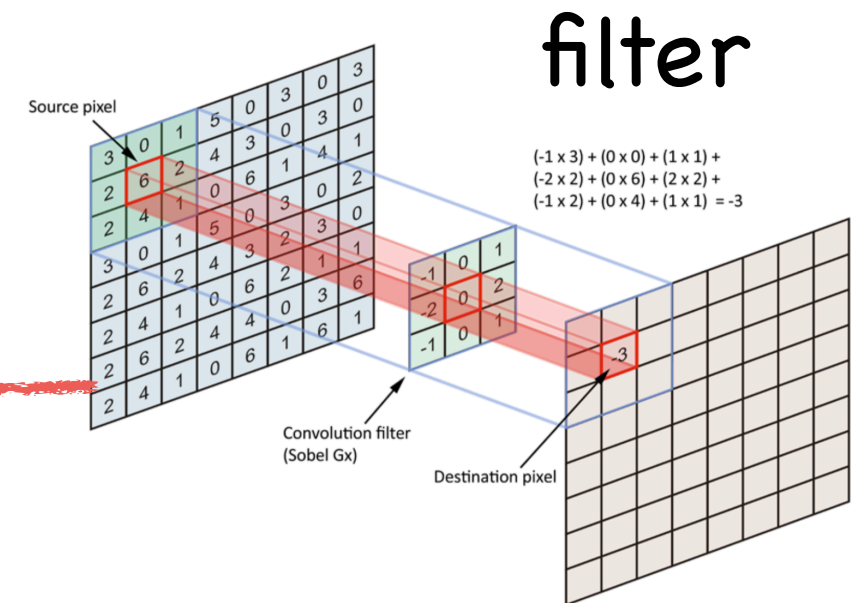
## Supervised

- (D)NN

- Convolutional NN

Often used for image recognition

Standard setup:

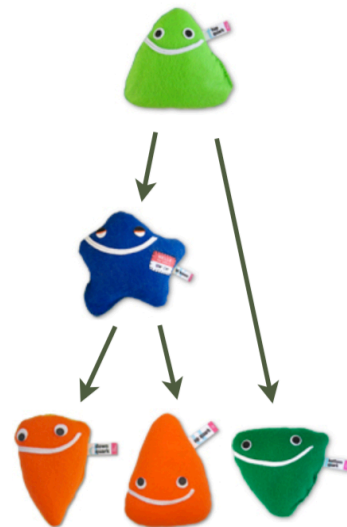


Allows to be sensitive on translationally invariant features

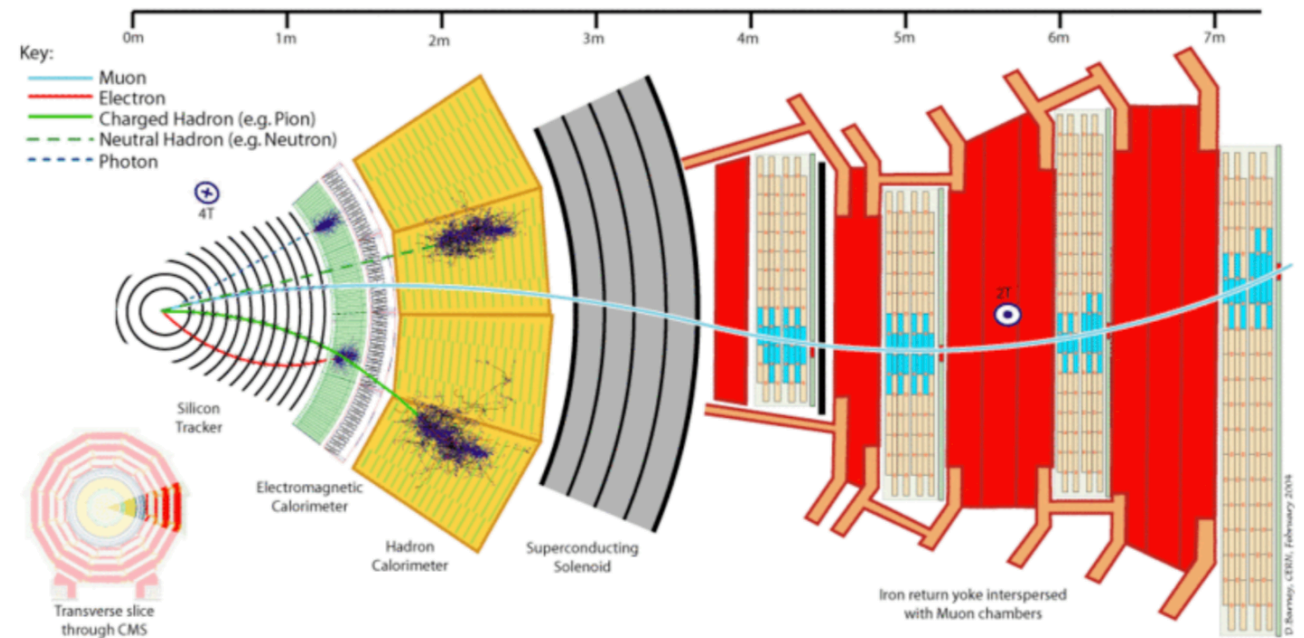
CNN has a sense of distance and orientation

# Application to top tagging

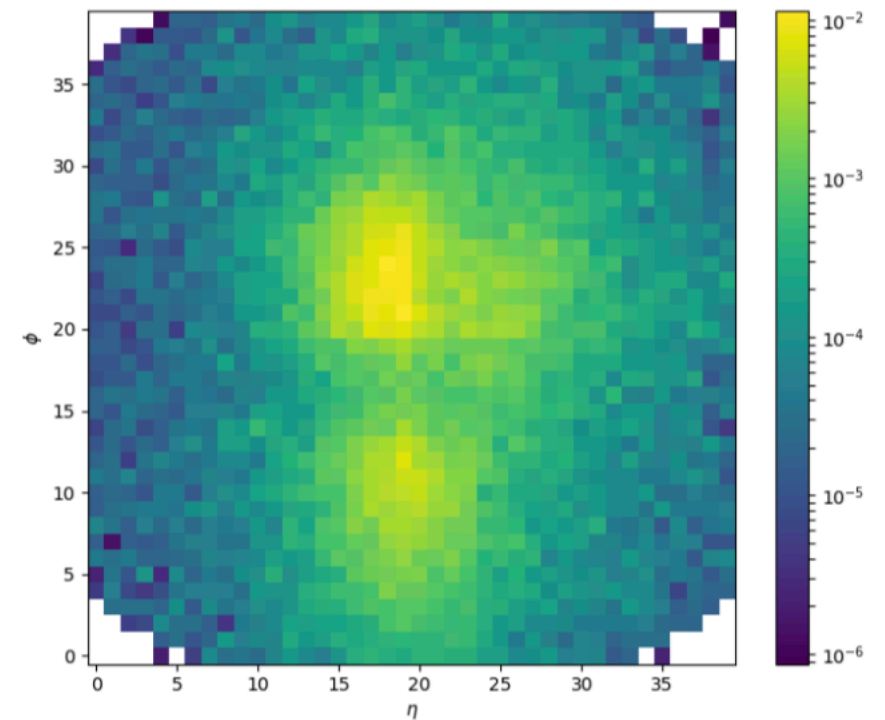
Top Quark



+



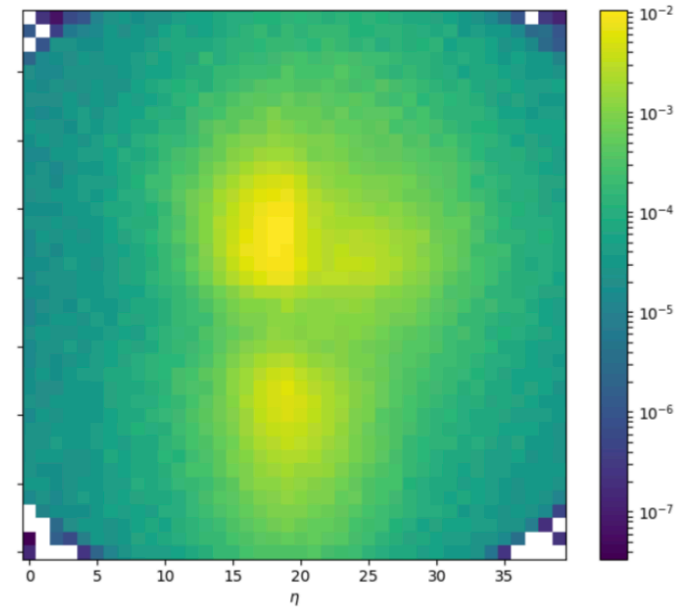
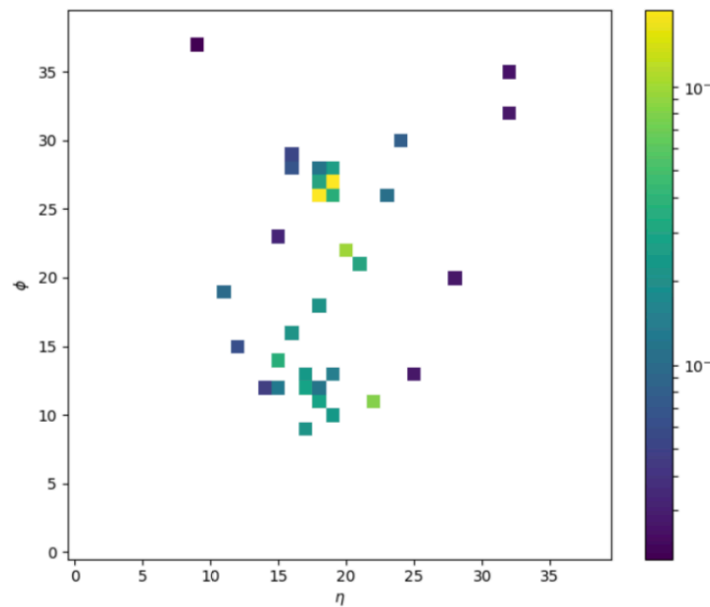
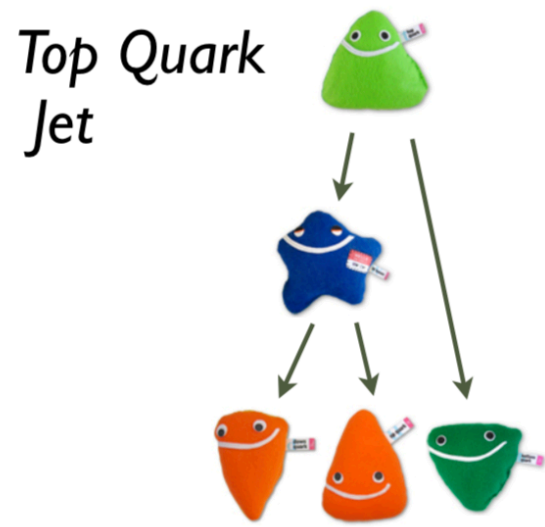
1. Measure particle energies in calorimeter
2. Reconstruct jet
3. Image preprocessing: center, rotate mirror, pixelate, trim, normalise



1000 image average

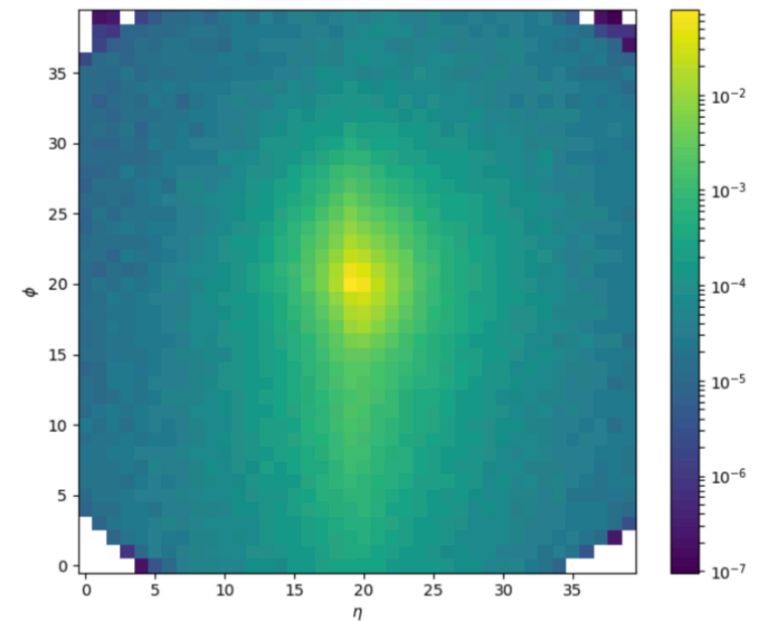
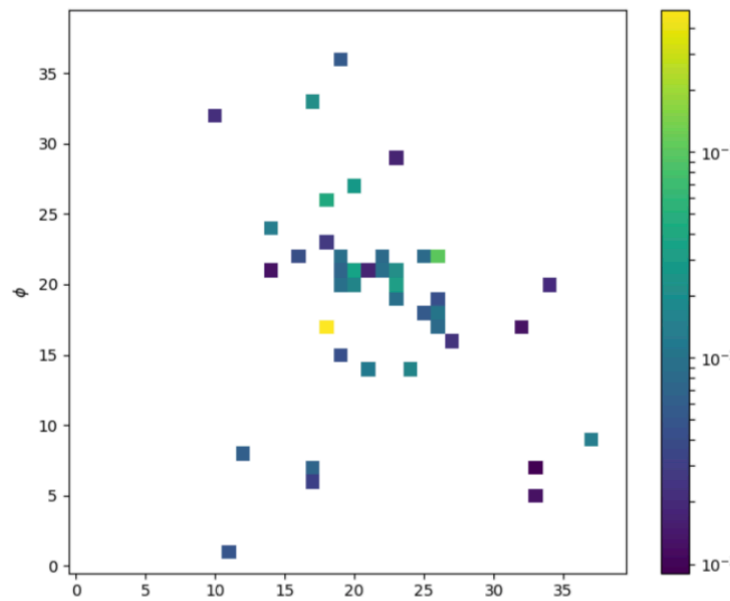
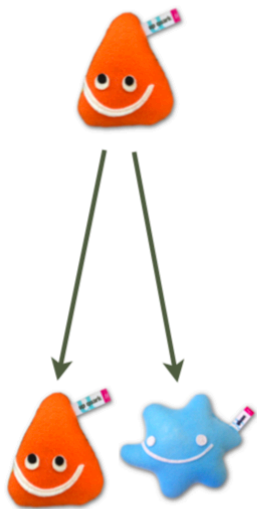


However, on event-by-event basis not so clear-cut...



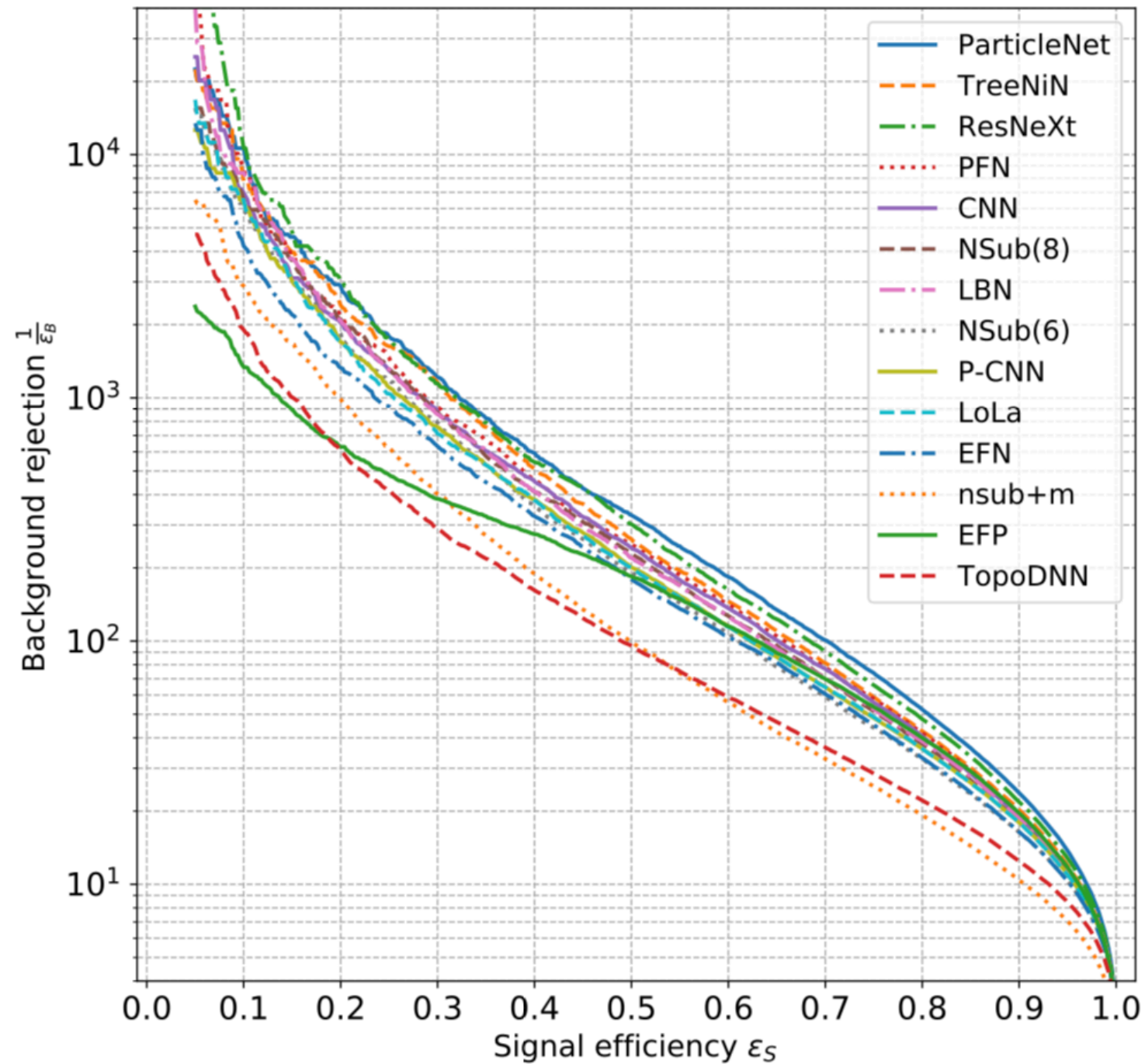
VS

QCD Jet



tough call... problems: sparsity of image

# Top tagging performance comparison of various NN approaches

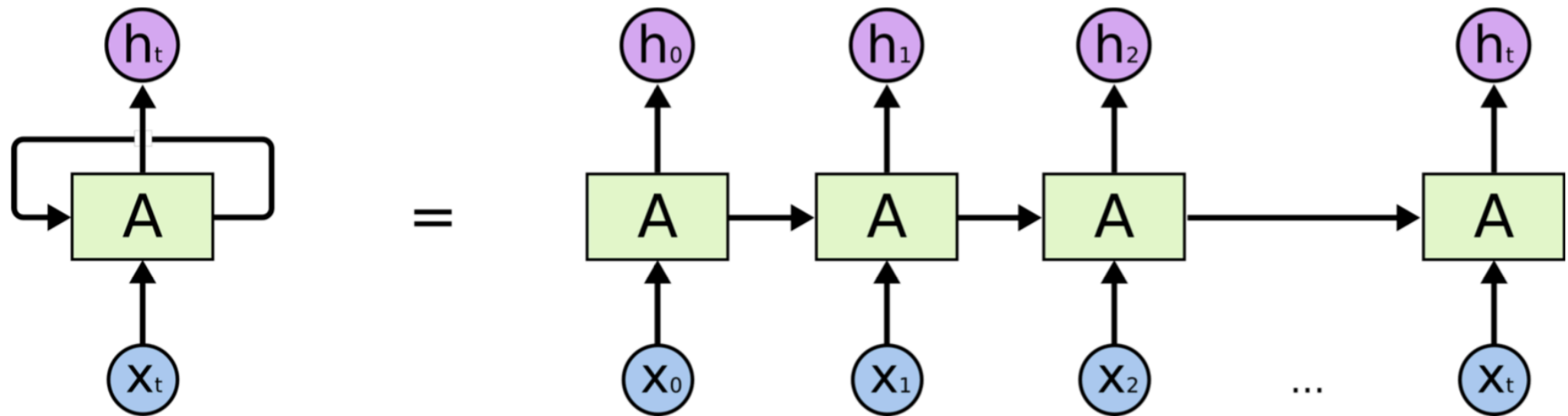


[1902.09914]



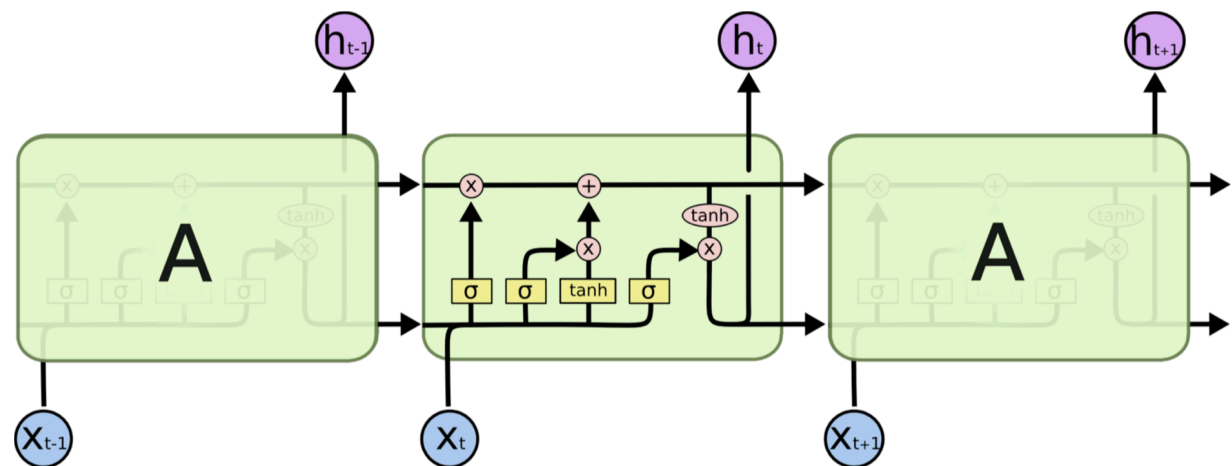
• Recurrent NN

for prediction only one output though



Long Short-Term Memory (LSTM)

Often used for language processing

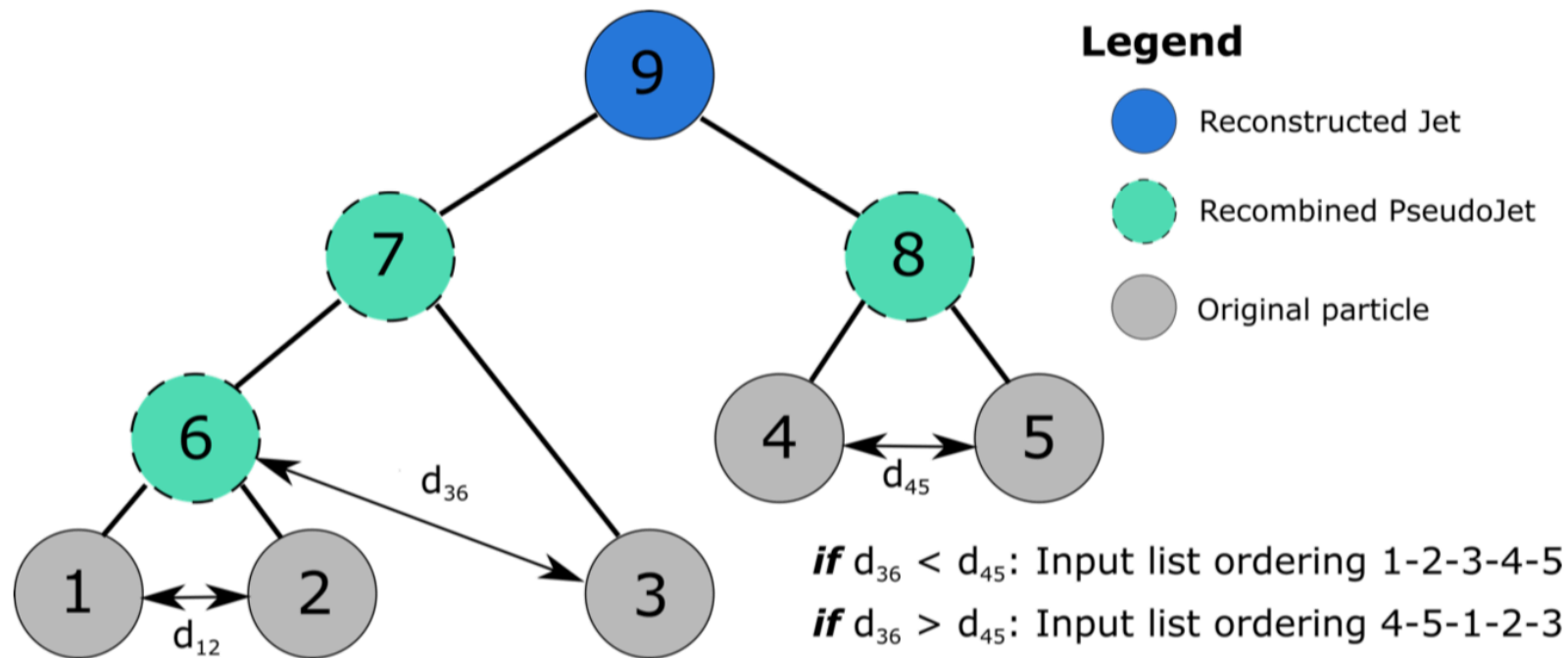


RNN (LSTM) has a sense of time and a memory

Good for variable-length inputs, as can be stopped after n iterations

# Application to top tagging with RNN

LSTM width of 128, fully connected layer of 64 nodes



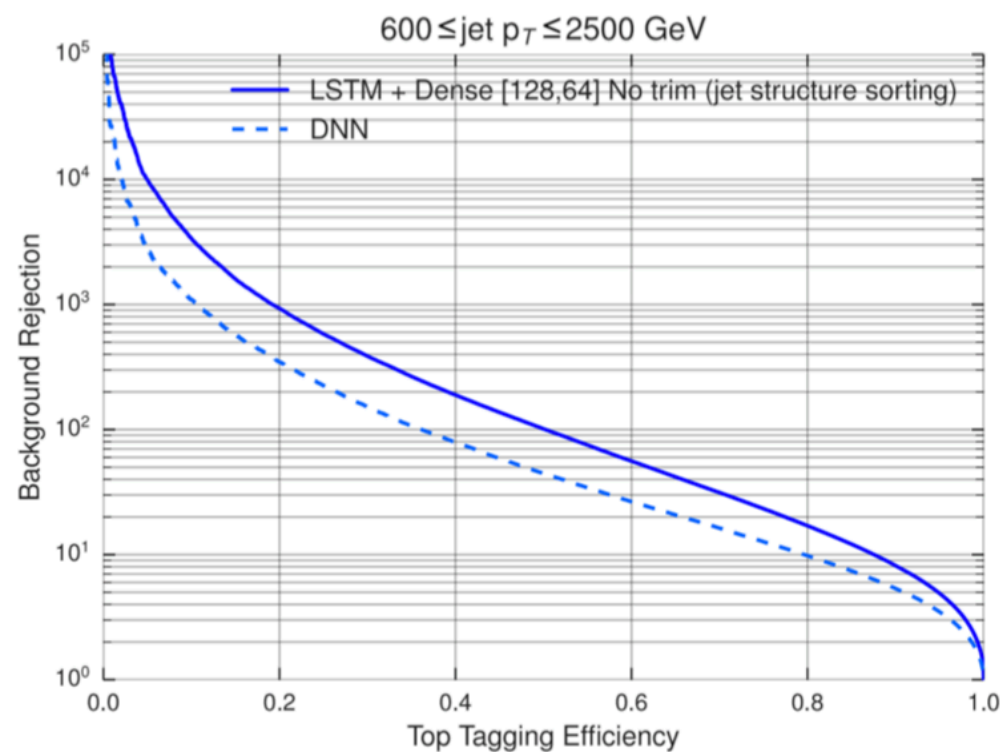
[Egan, Fedorko, Lister, Pearkes, Gay '17]

Jet cluster algorithm defines ordered sequence (timing)

recombination history

$\leftrightarrow$

ordering of sentence



Shows good performance compared to DNN

see also [Louppe, Cho, Becot, Cranmer '17]



# Unsupervised

## • autoencoder

autoencoder used for anomaly detection

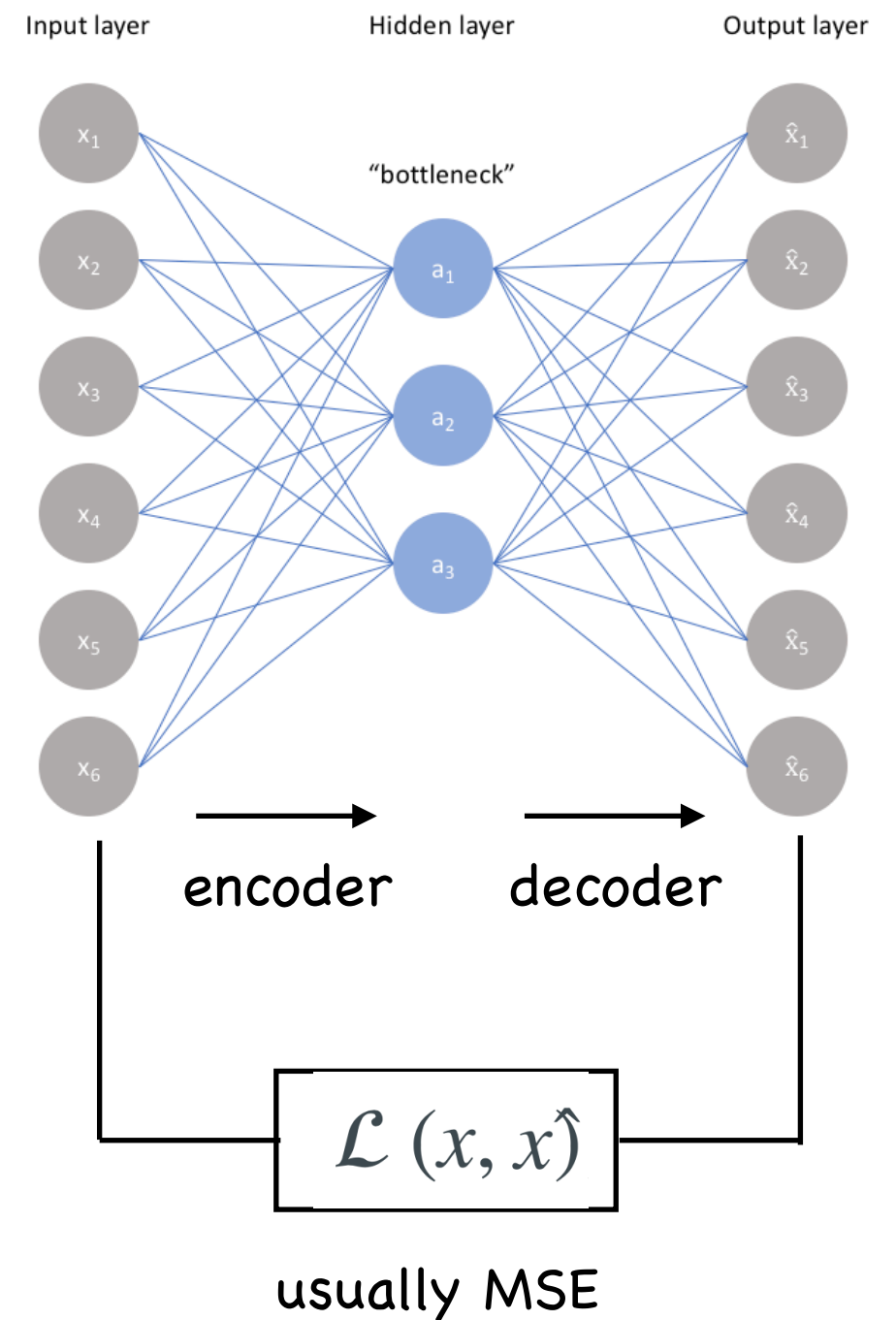
in first step input is encoded into information bottleneck

between input/output layer and bottleneck can be many hidden layers  
-> highly non-linear

after bottleneck decoding step

Reconstructed output is then compared with input via loss-function (usually MSE)

NN is trained such that input and output high degree of similarity



# "The strange death of theory"



Frankfurter  
Allgemeine  
Zeitung

23.01.2017

or is it?

# Pros and Cons of ML in Particle Theory

- Particle theory in some sense **worst** and **best** place to use ML
- MVA well motivated to extract correlations without existing theory, i.e. **stock trade**
- In particle physics we established gauge theories, thus, we have existing theory to predict connection of 'input with output'
- Current pheno approach:

We take first-principle QFT:  $\mathcal{L} = \mathcal{L}_{EW} + \mathcal{L}_{QCD} + \mathcal{L}_{Higgs}$

Put it into an event generator to generate pseudo-data

Then a smart physicist or MVA comes up with way to access the Lagrangian we put in in the first place

[Soper, MS '14]

[Ferreira de Lima, Mattelaer, MS '17]

[Prestel, MS '19]



Seems like an unnecessary detour...

More suitable: **All-order Matrix Element Method**



# Training ML on pseudo-data

- MVAs will optimise for – according to MC – most sensitive exclusive phase space regions
  - theory uncertainties difficult to control
- Full event generators are mashup of different parts that are partly tuned, i.e. hard interaction, UE, ISR,...
- Highly computationally intensive. If you want to template correlations of say 7 particles:
  - Time estimate:
    - 7 microjets, each 4-momentum components divided into only 10 bins
    - $10^{28}/7! \sim 10^{24}$  configurations
    - If MC takes 1 ms per event →  $10^{13}$  years to have 1 hit per config.

# Training ML on data only

- Less plagued by theoretical uncertainties
- But only possible if objects to reconstruct or events to measure already in data.

problem for new or rare physics:

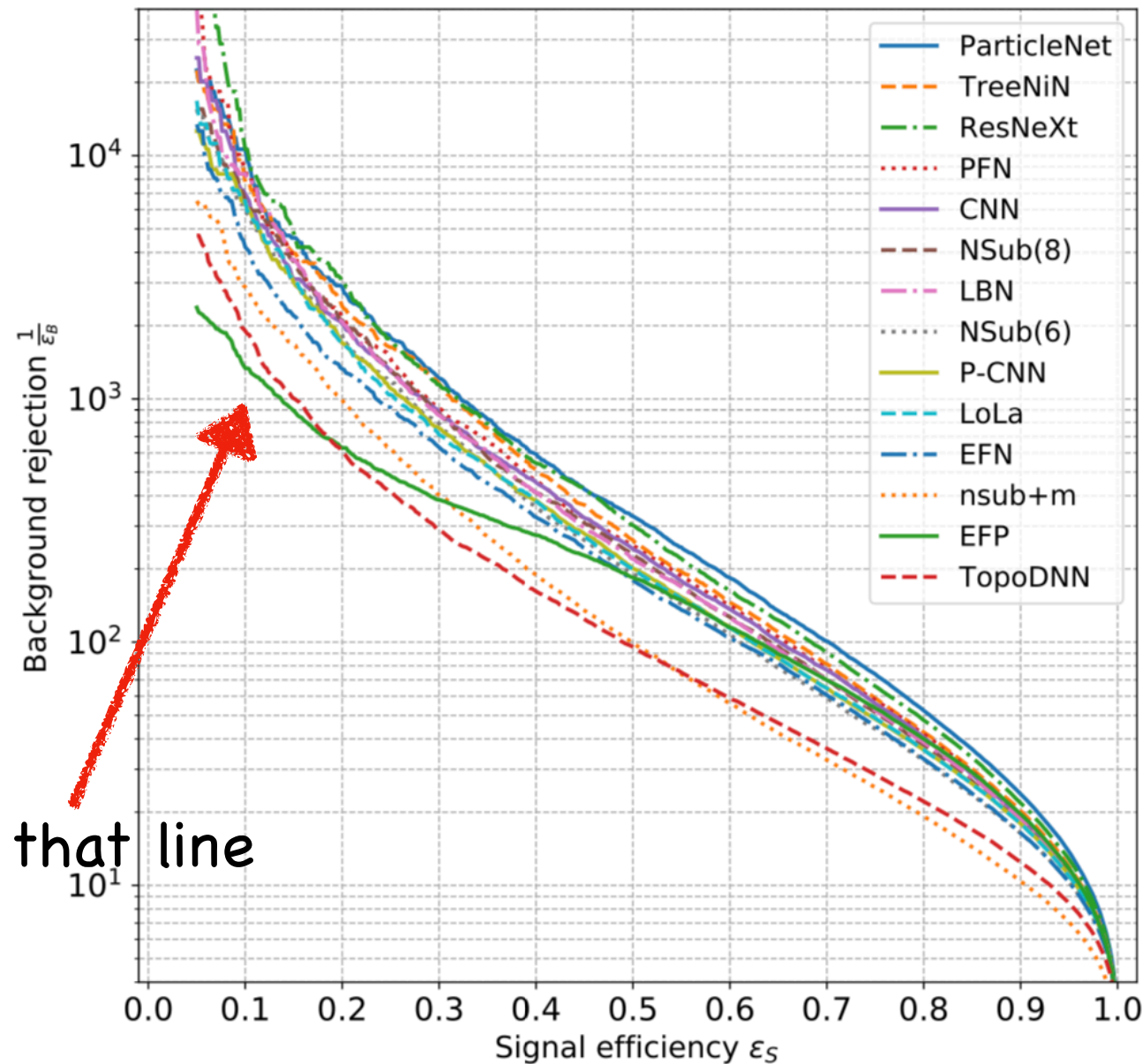
- ➔ gluino-tag
- ➔ axion-tag
- ➔  $pp \rightarrow HH$
- ➔ ...

- Everything done purely on data without theory cross-check has 0 safety margins...

➔ now gone 2 TeV excess in ATLAS and CMS might be an example

[Goncalves, Krauss, MS '15]

# Top tagging performance comparison of various NN approaches



[1902.09914]

Attention to that line

High performance in tagging efficiencies can be achieved.  
However, be careful in taking these efficiencies at face value

Current ROC curves do not come with uncertainty bands



# Decorrelation of nuisance parameters

Solution: use a pivotal quantity as a classifier [DeGroot, Schervish '75]

Pivot = quantity whose distribution is invariant with nuisance parameters

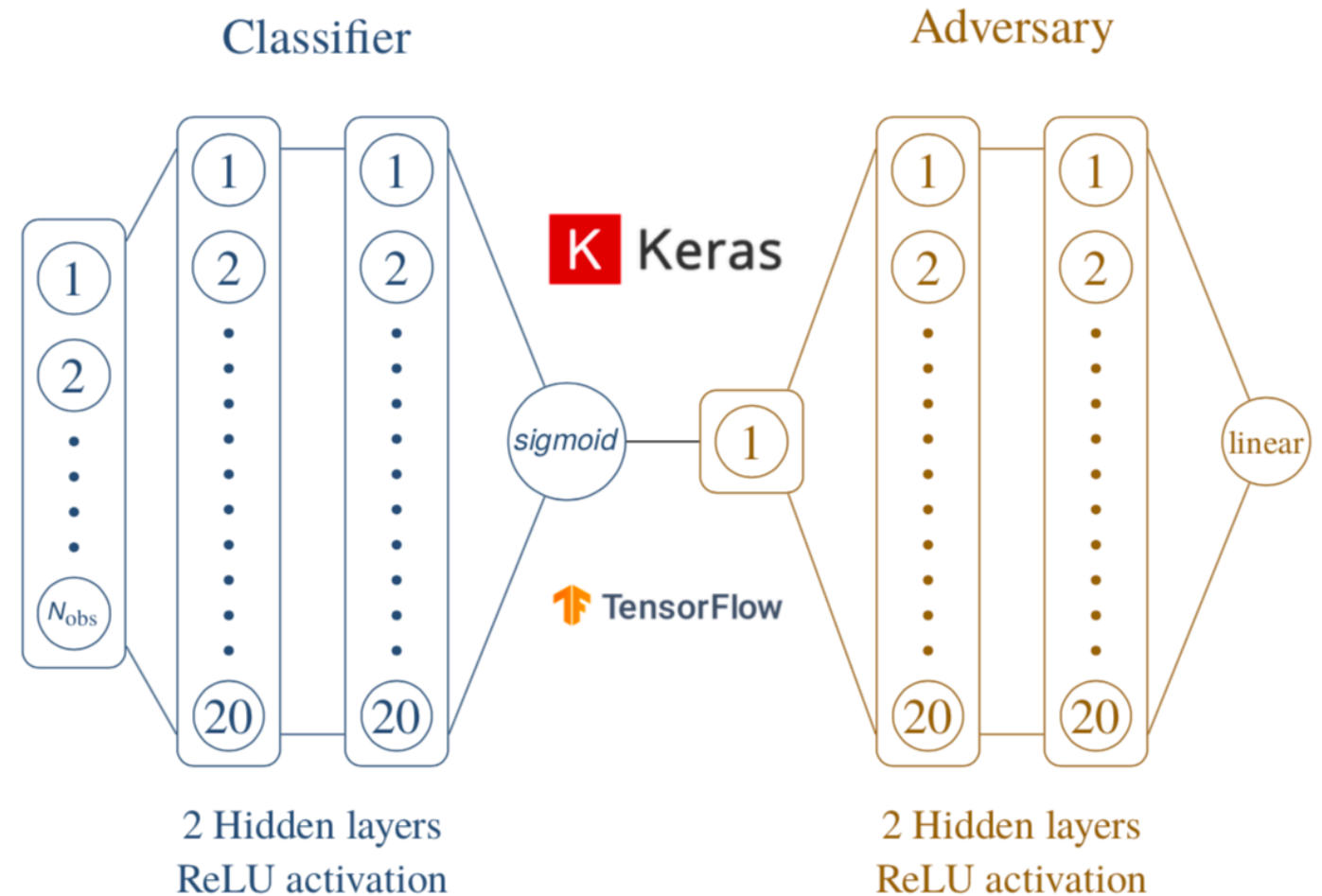
- training method for pivotal classifier using adversarial networks [Louppe, Kagan, Cranmer '16]
- showed that it works for systematic uncertainties which affect classification on an event-by-event basis (e.g. event reconstruction uncertainties)
- use pivotal classifier in the context of theoretical scale uncertainties affecting the event sample as a whole [Englert, Galler, Harris, MS '19]

# implementation of adversarial

- Connect both networks through loss function

$$L = L_{\text{Class}} - K L_{\text{Adv}}$$

- Adversarial will force the Classifier output to look the same for each 'smearing class'



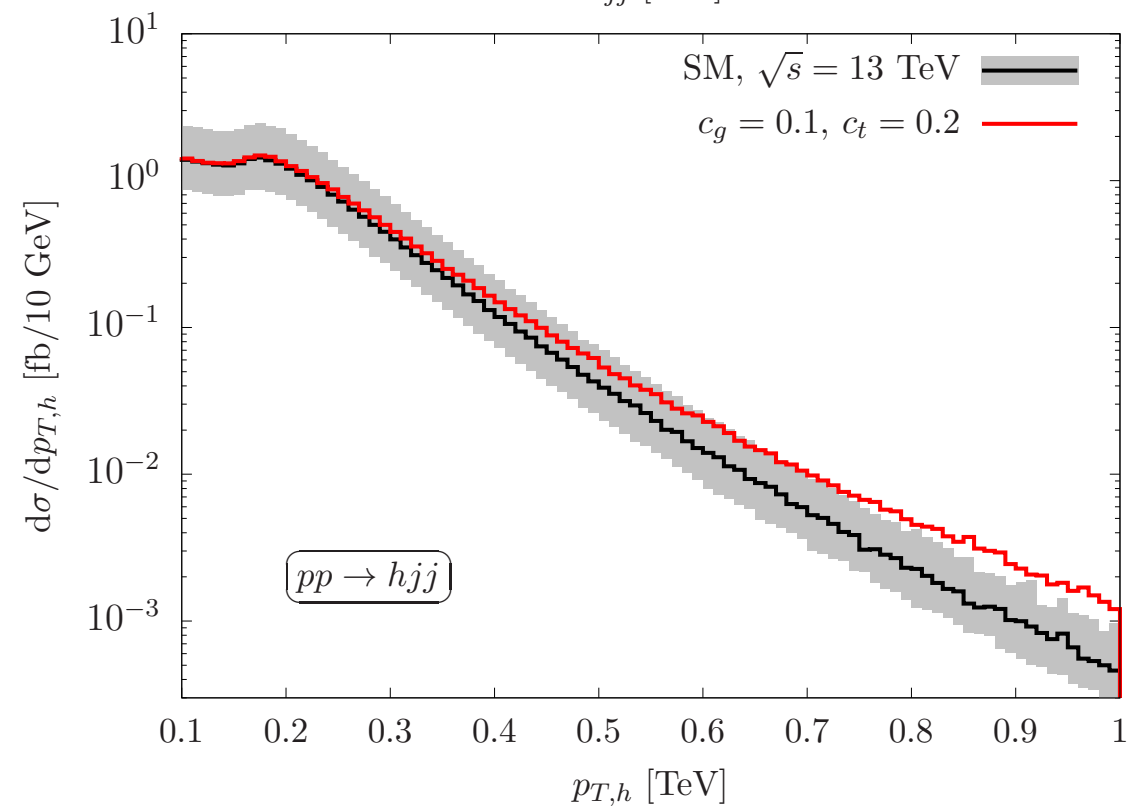
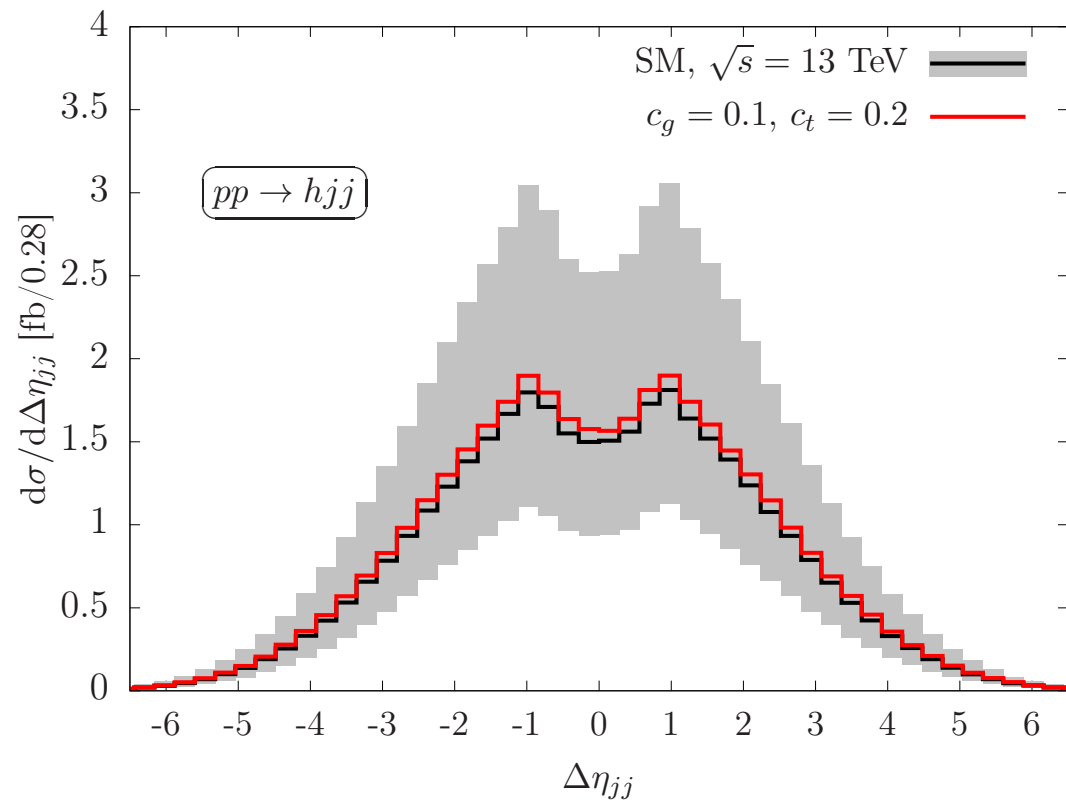
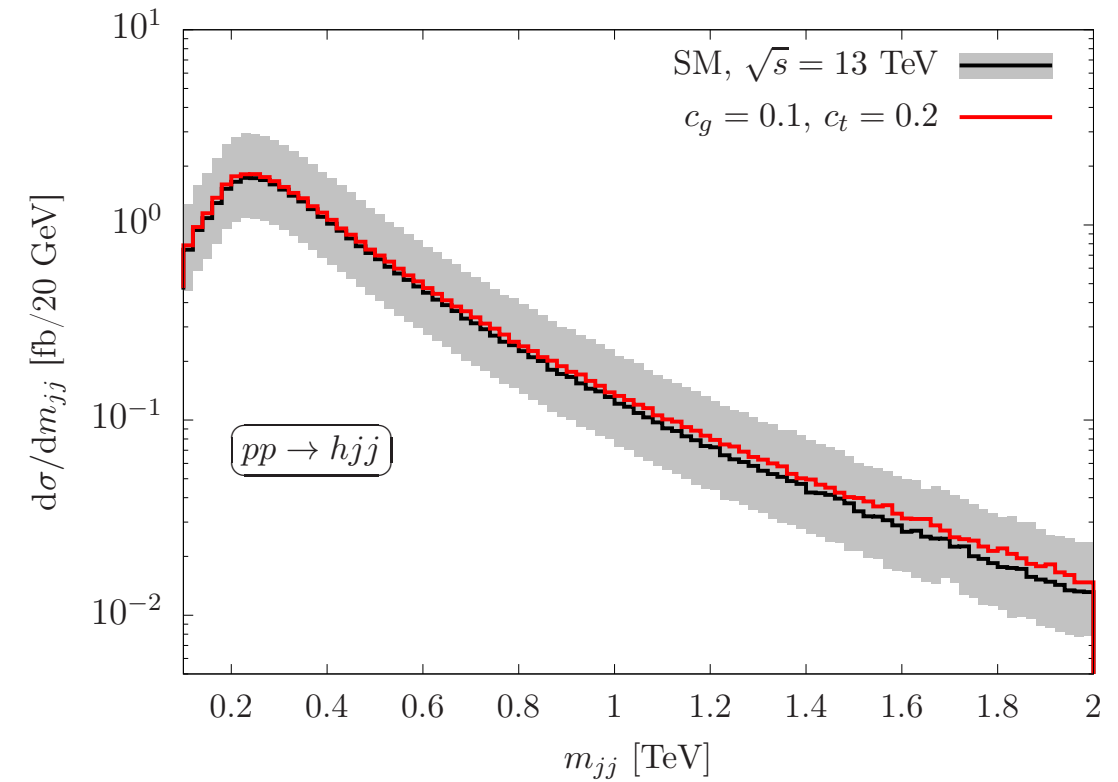
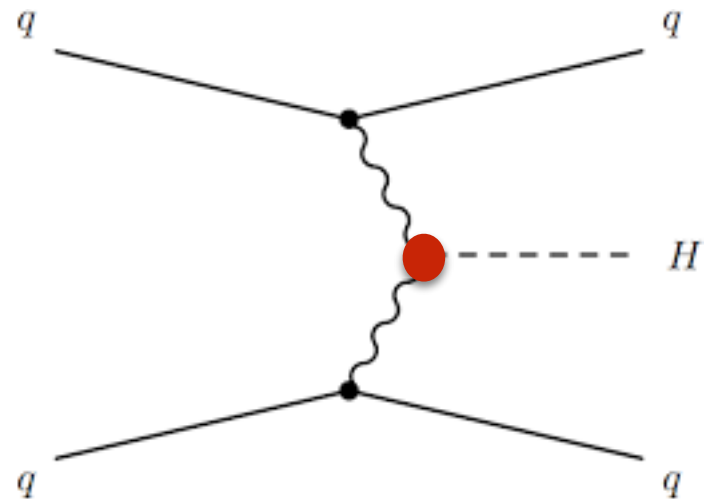
- Both networks have to be trained and improve simultaneously -> ping-pong training

- Approach can be used for systematic and theoretical uncertainties

[Louppe, Kagan, Cranmer '16]

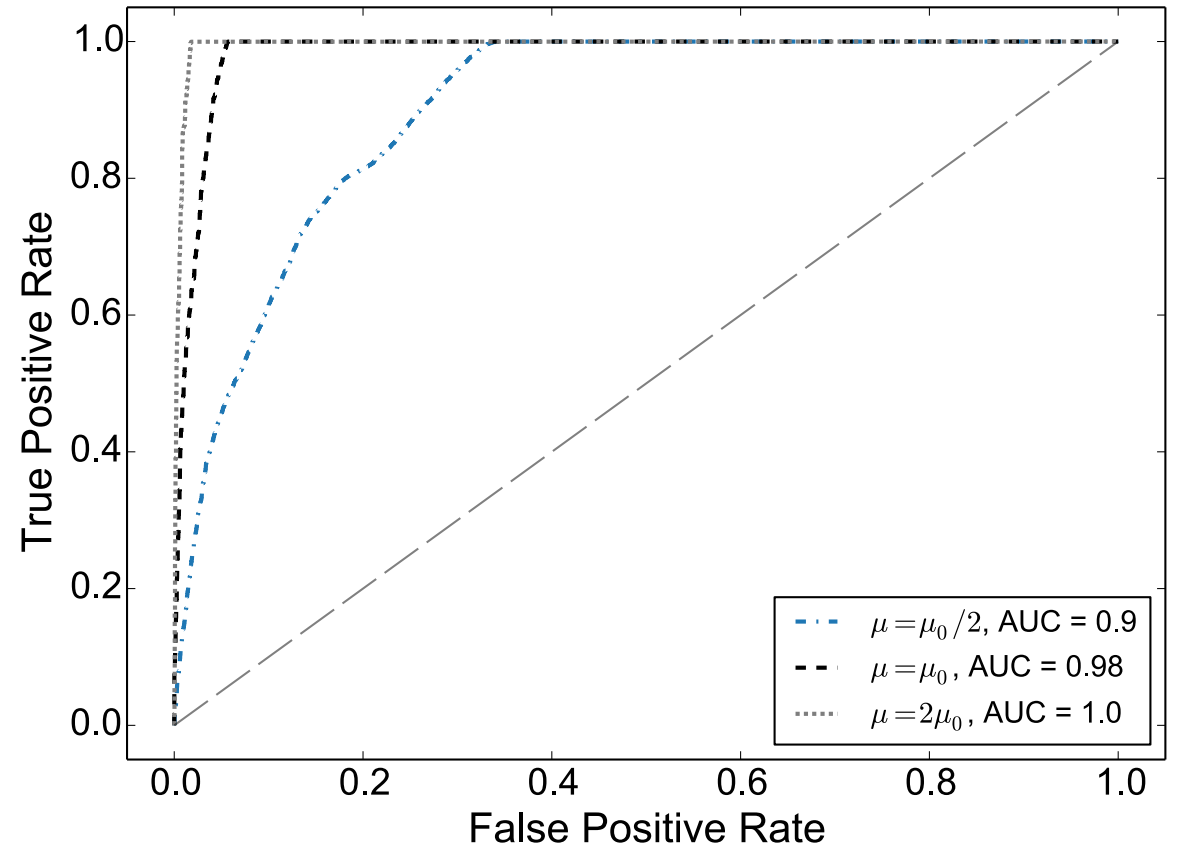
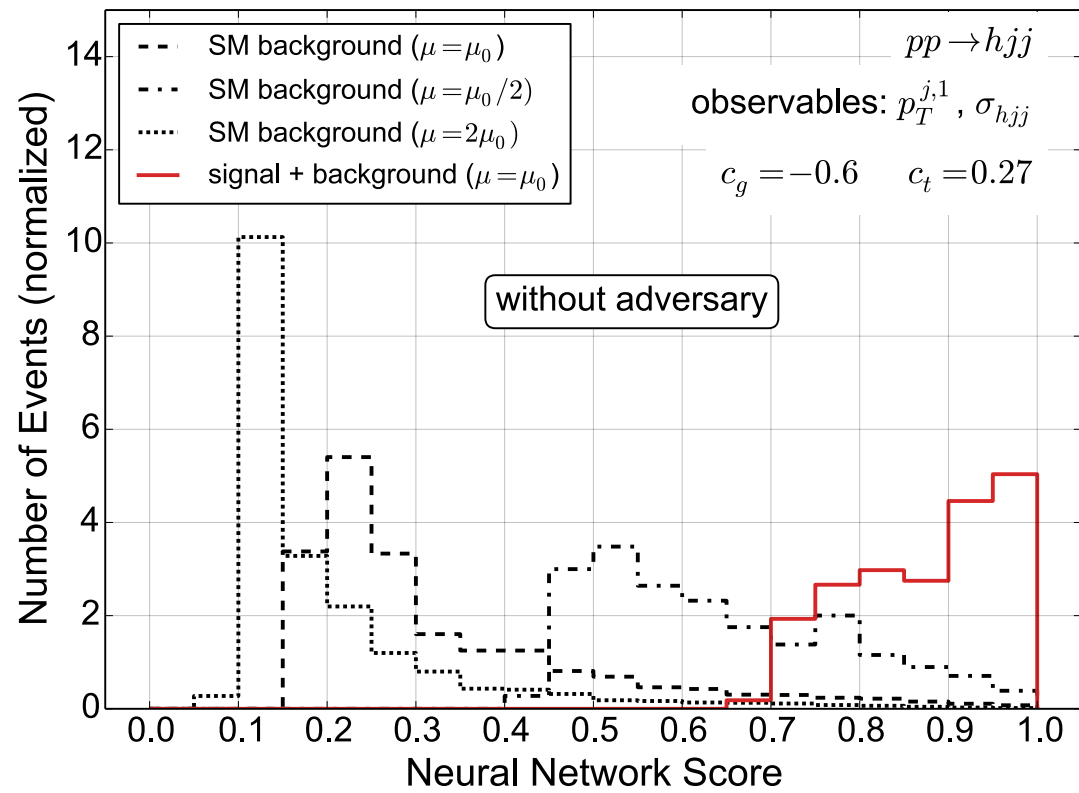
[Englert, Galler, Harris, MS '19]

# Example: EFT hypothesis test in H+jets

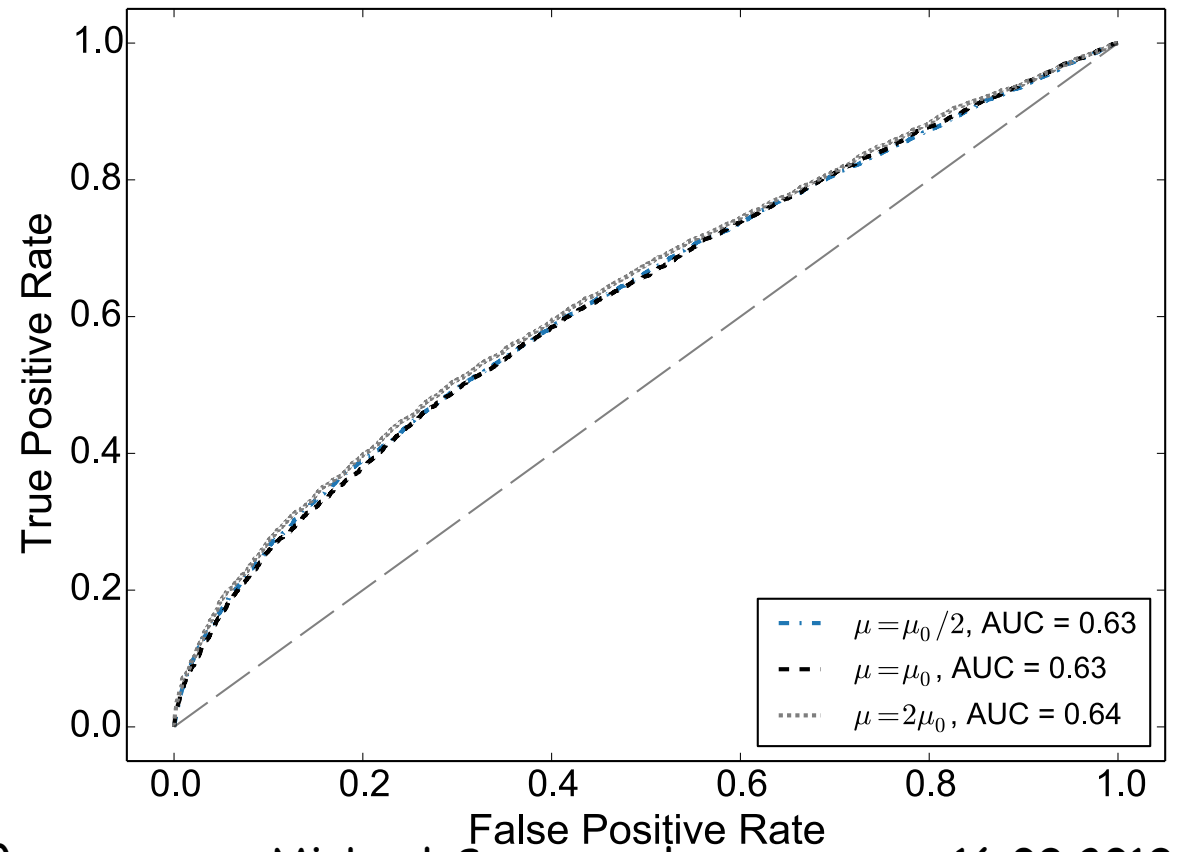
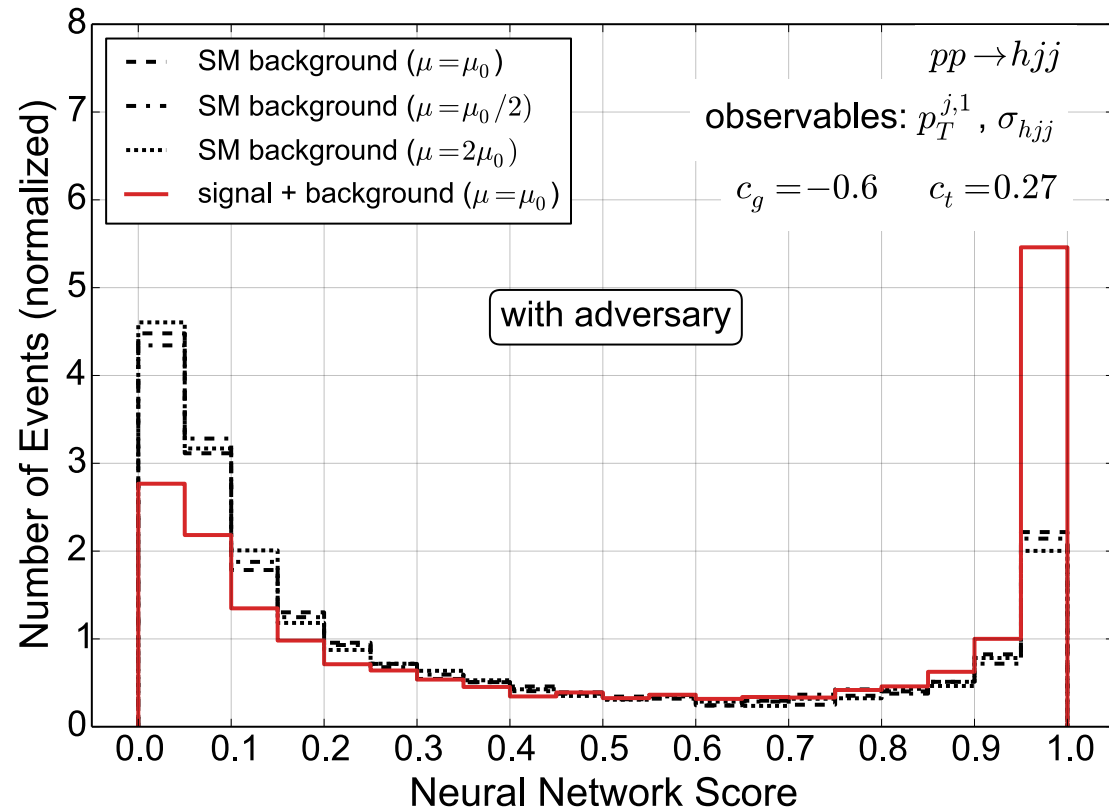




# no adversarial



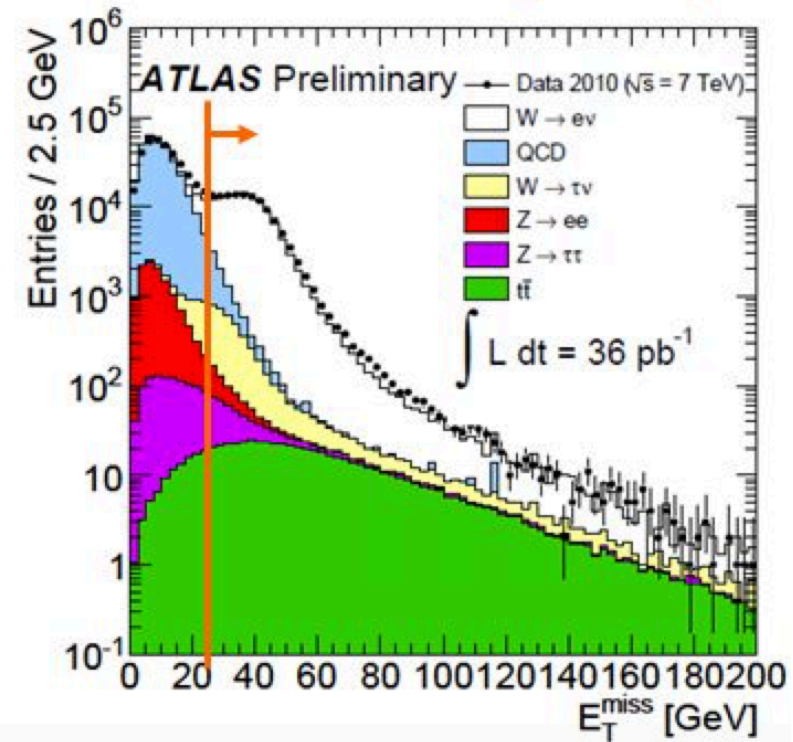
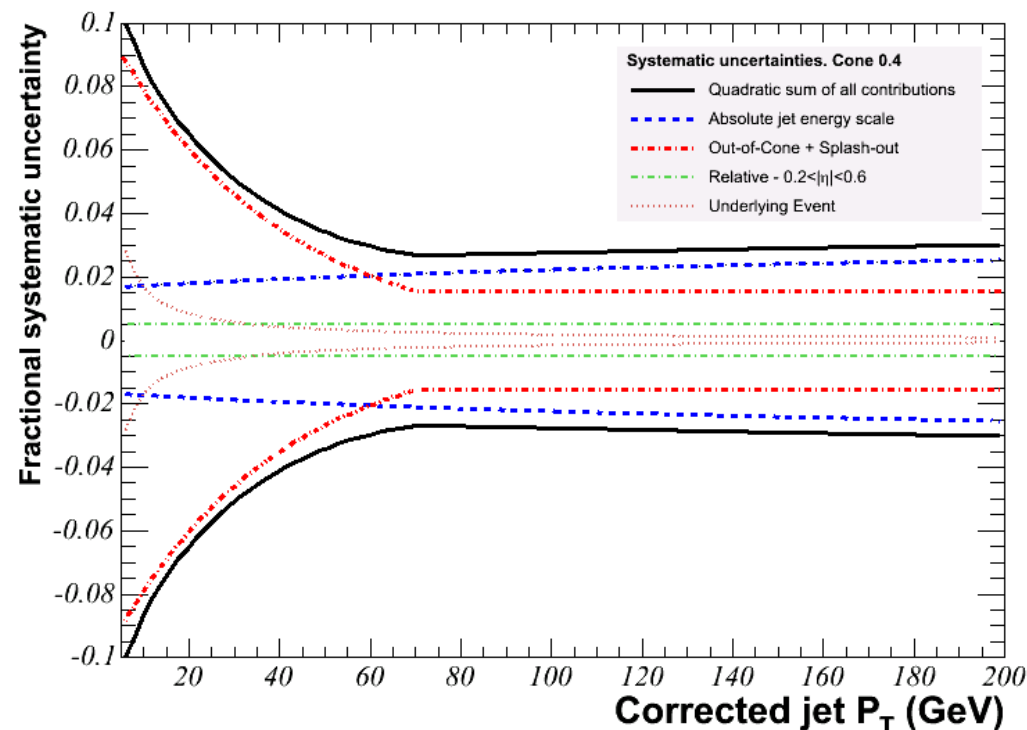
# with adversarial



- Remember, we never wanted to train on MC
  - We want to discover new physics without knowing what we are looking for
- ➔ Anomaly detection via unsupervised learning



systematic (experimental) uncertainties last remaining uncertainties, e.g. energy smearing of jets or MET

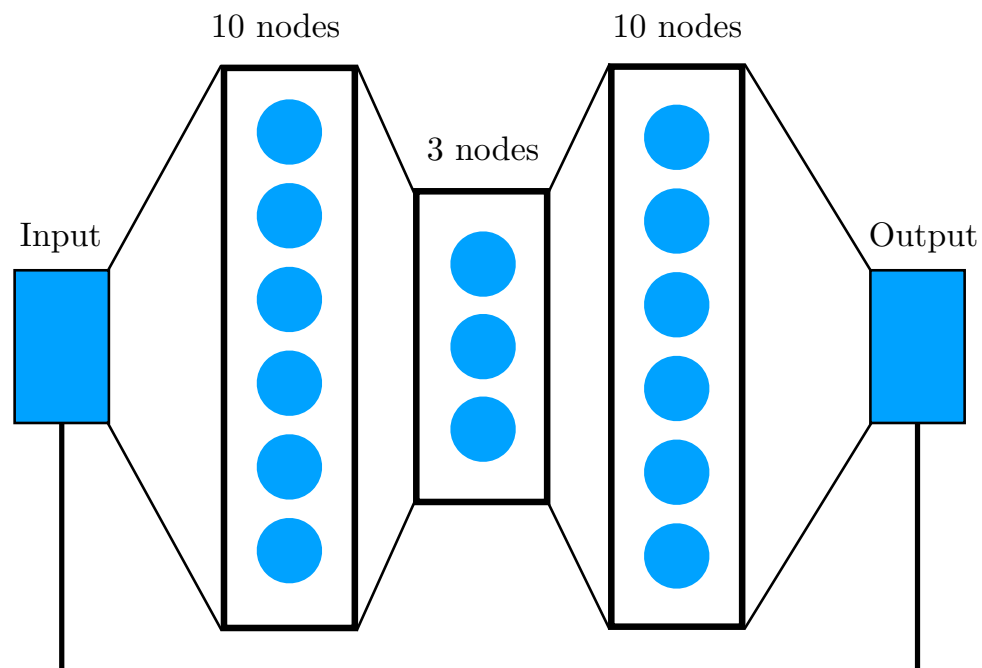




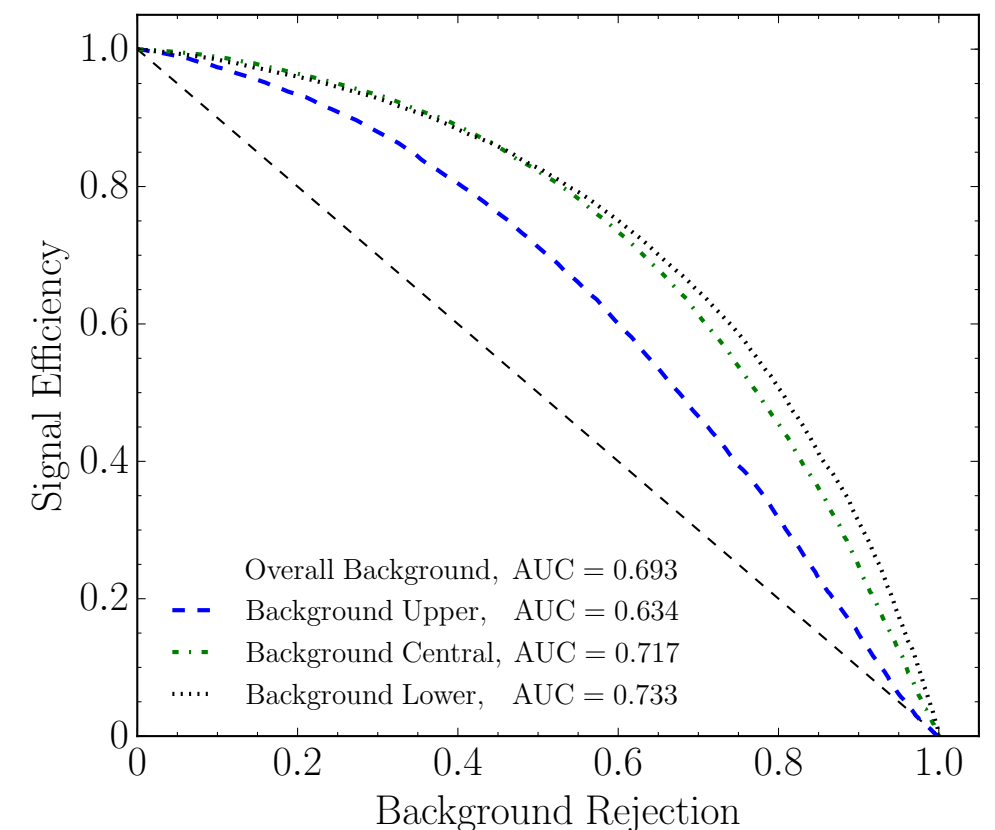
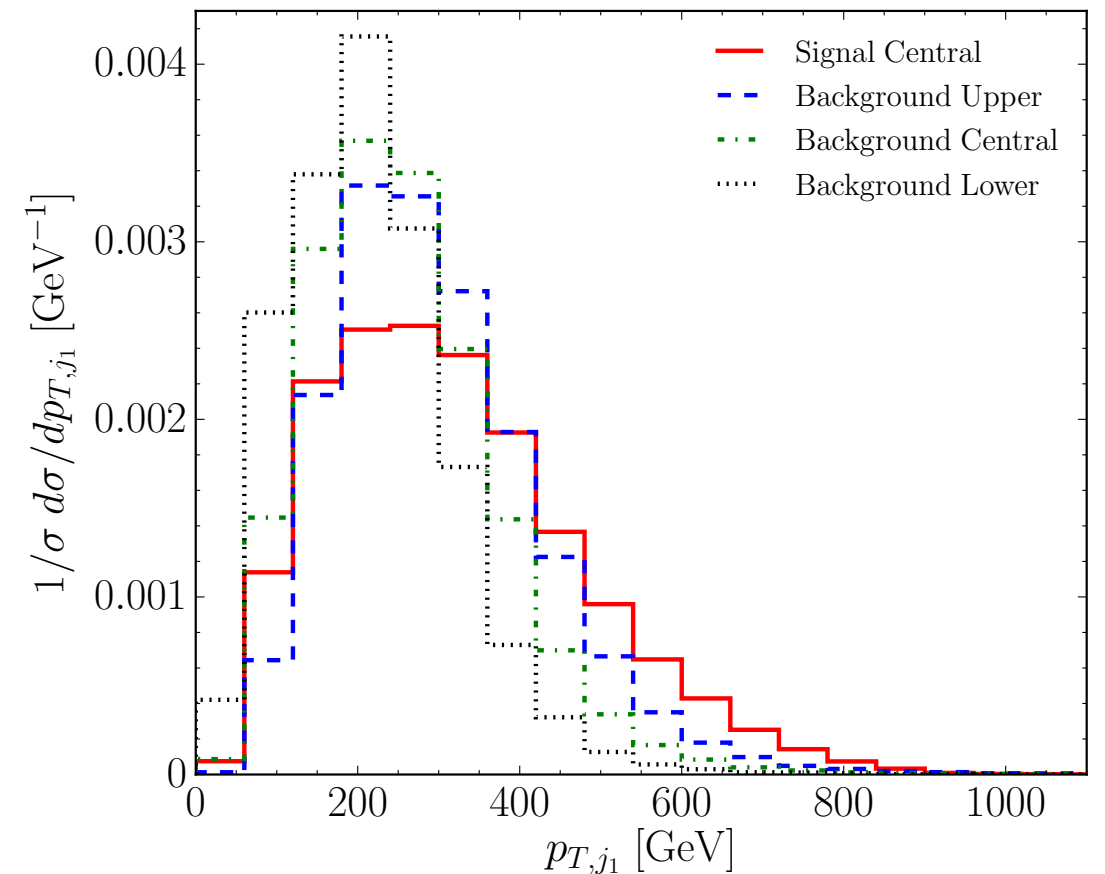
# Uncertainties via adversarial for autoencoder (unsupervised)

signal:  $pp \rightarrow Z' \rightarrow tt$       bkg:  $pp \rightarrow tt$

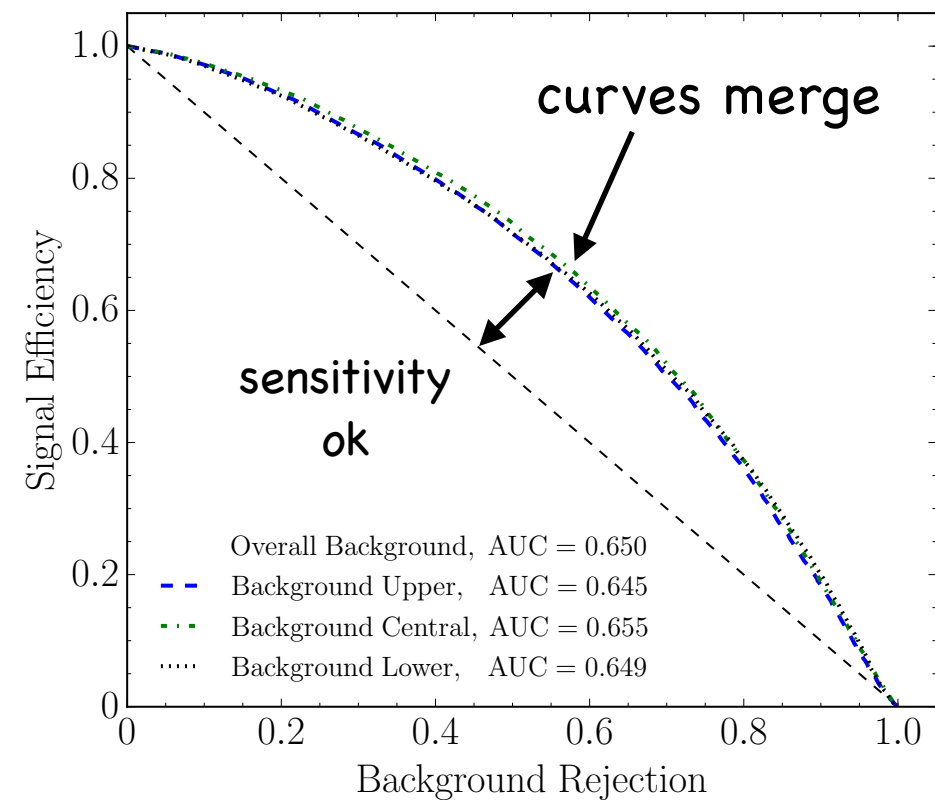
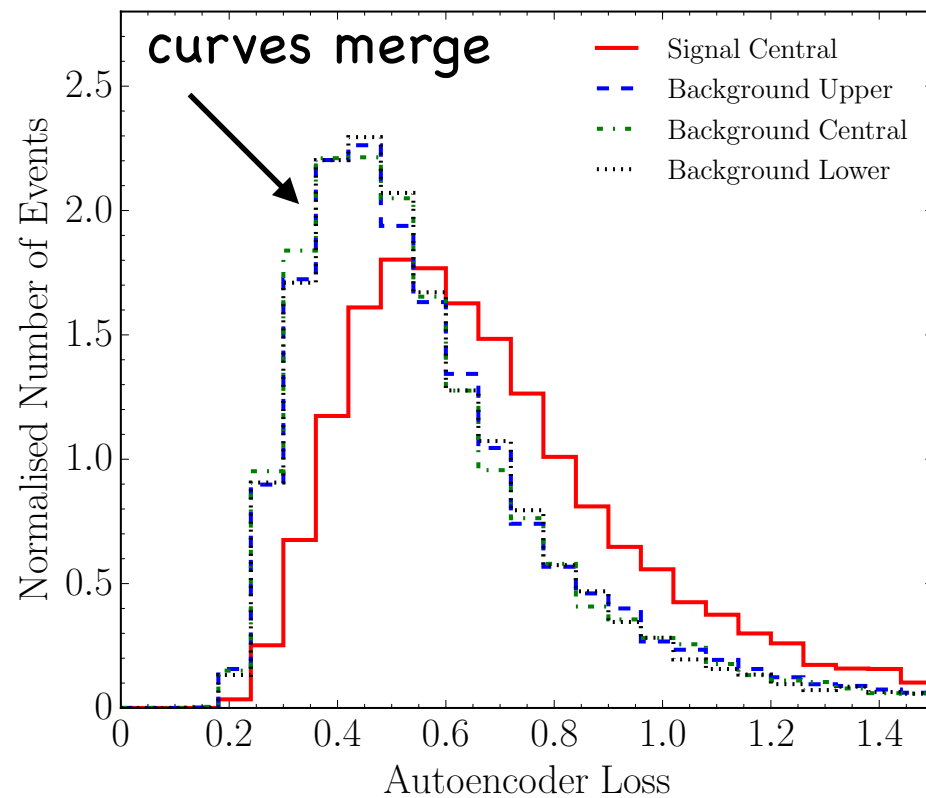
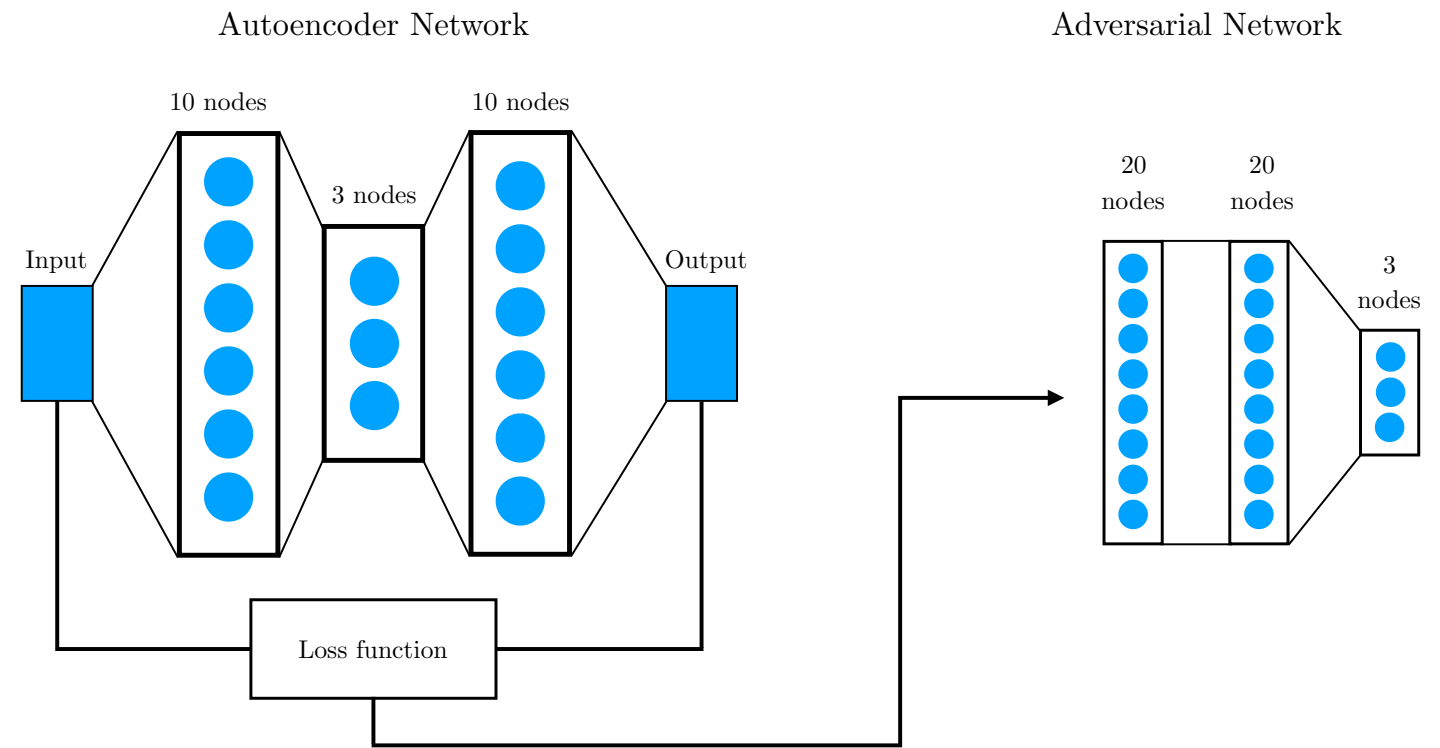
- to benchmark sensitivity first without adversarial
- only train on bkg (anomaly detection)
- syst. uncertainties via jet, MET smearing



## smearing jet distributions for bkg

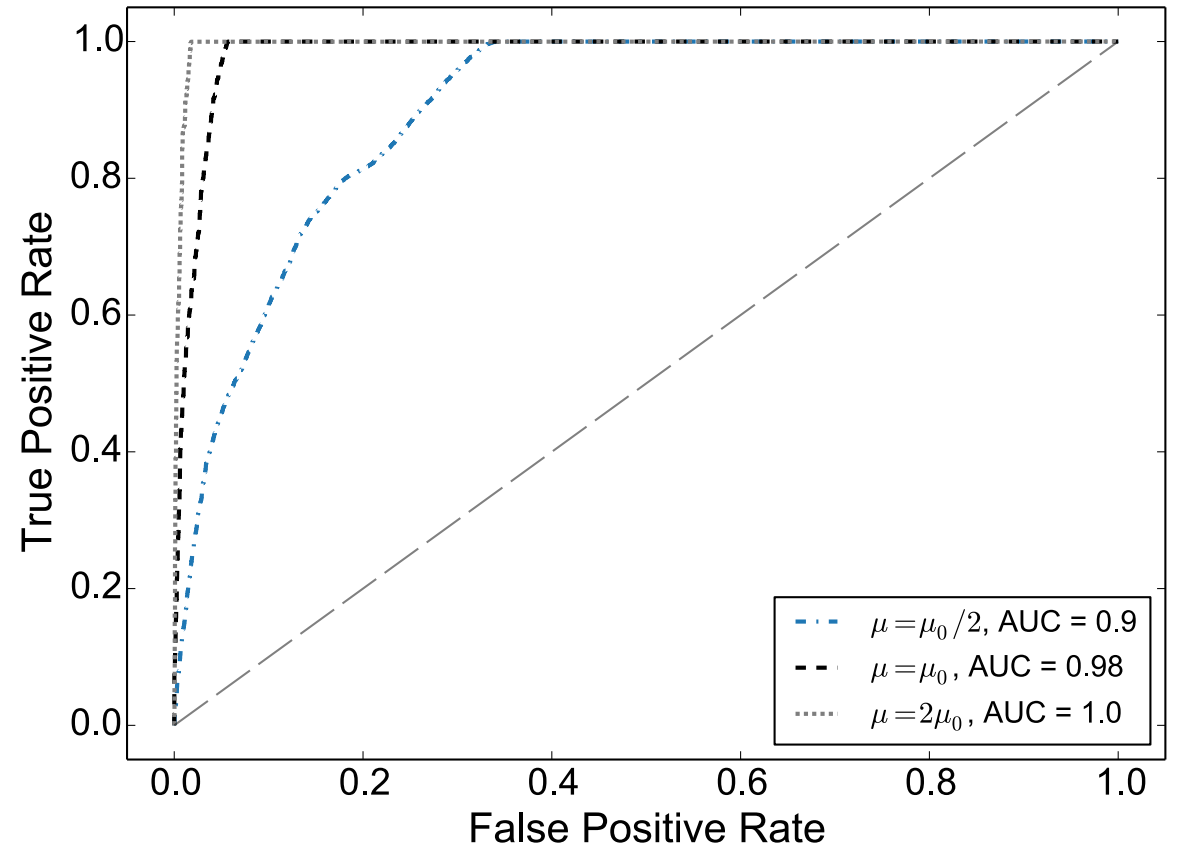
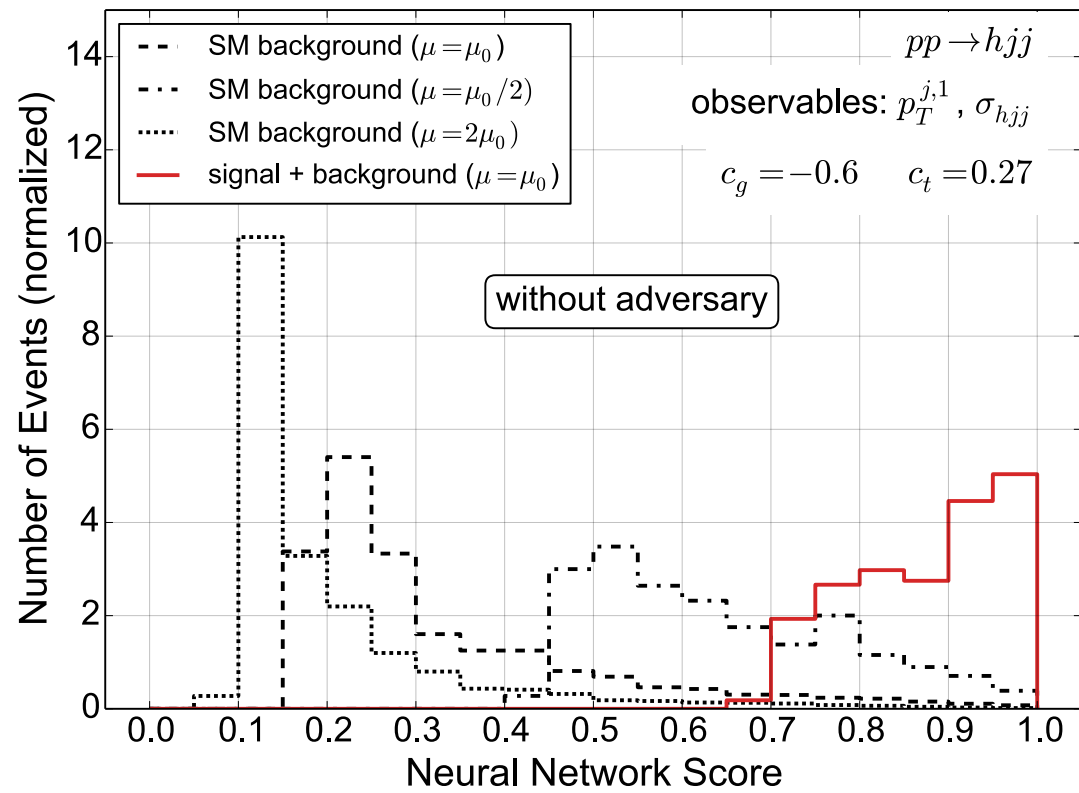


- We can remove dependence on smearing by applying an adversary that will try to classify the direction of smearing
- The two networks are in a zero-sum game - an increase in adversary performance will result in the autoencoder being penalised.

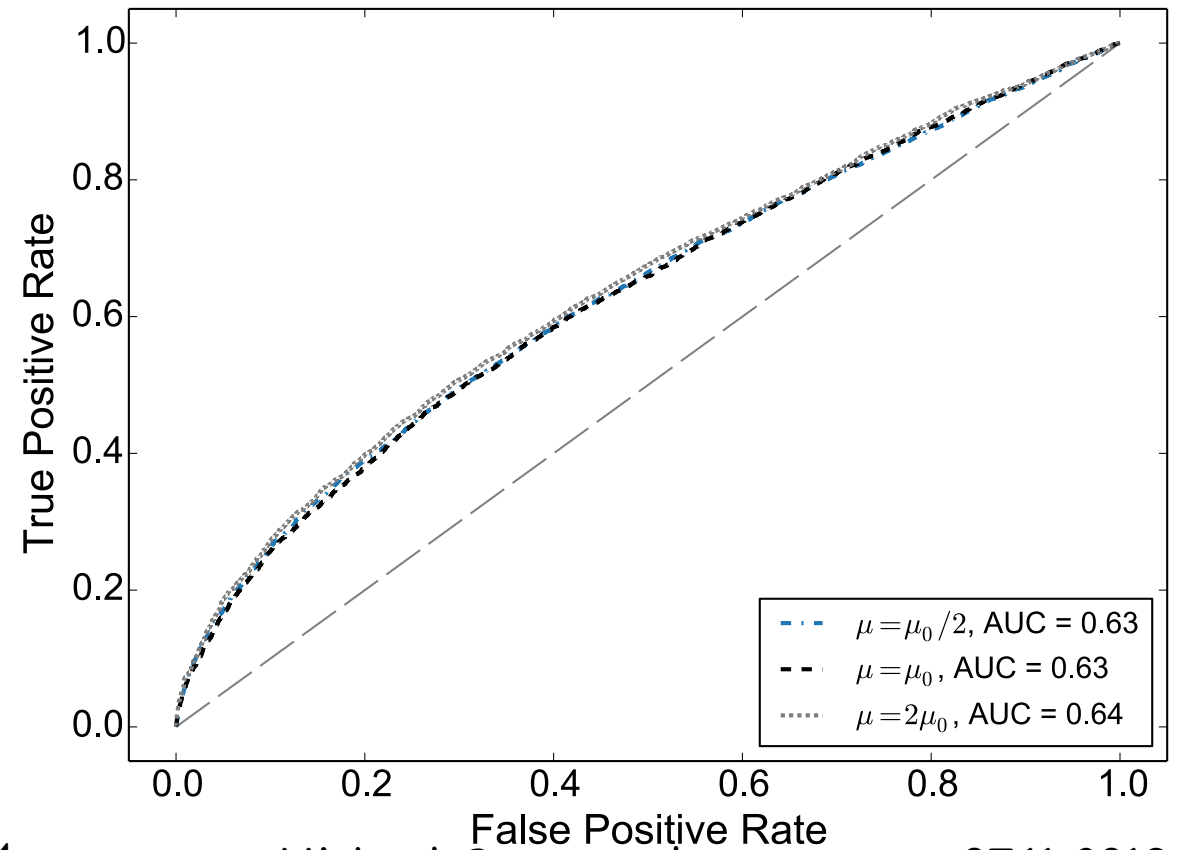
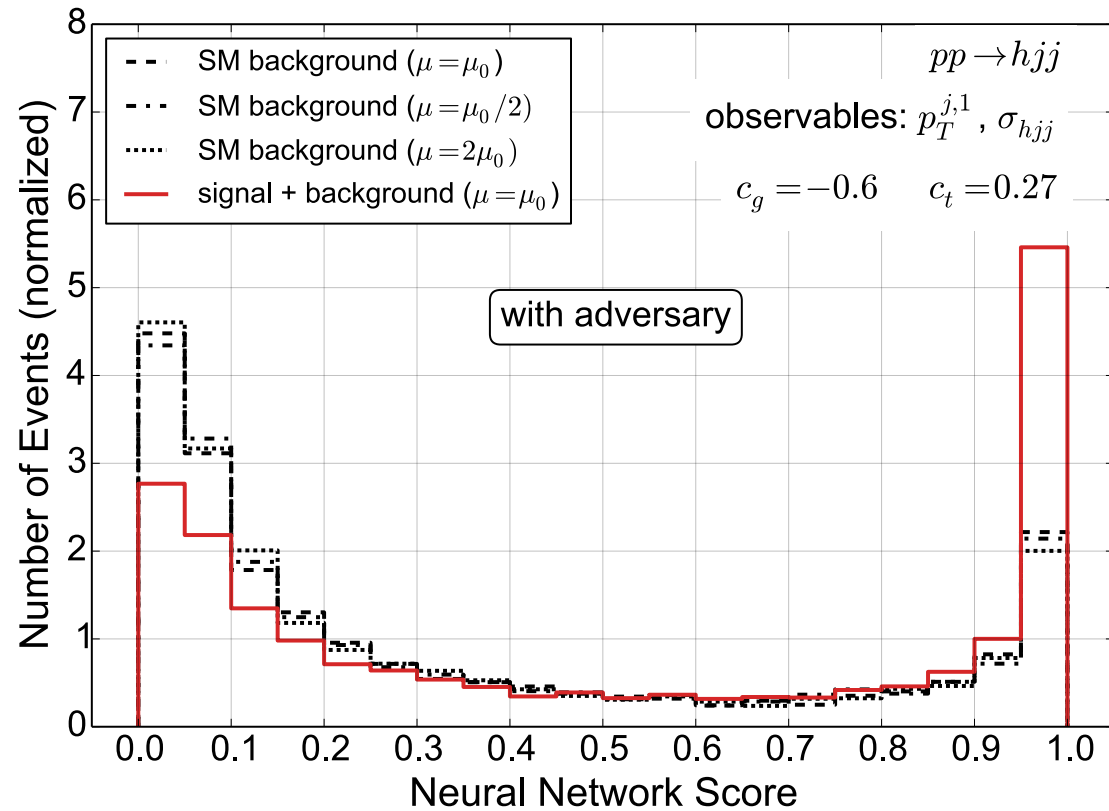


➔ Robust, yet sensitive, anomaly detection with adversarial autoencoders

# no adversarial



# with adversarial





## Going beyond classification and regression

- All previous examples and methods for classification and regression – either supervised or unsupervised
- If theorist should do that is up for debate... development of methods is done in data science (not particle theory)  
-> only track available is heavy user
- **Question:** Can we use ML (NN) such that we obtain new physics results – instead of extracting correlations we hide in pseudo data in the first place?
- **Answer:** Yes, neural nets are great numerical solvers!

[Piscopo, MS, Waite '19]

[Hornik et al '89]

## Universal approximation theorem:

[Hornik '91]

**feed-forward** network with a single hidden layer containing a finite number of **neurons** can approximate **continuous functions** on **compact subsets** of  $\mathbb{R}^n$ , under mild assumptions on the activation function.

## Expression of the output of a neural net:

$$N_m(\vec{x}, \{w, \vec{b}\}) = \sum_{k,n} w_{mk}^f g(w_{kn}^h x_n + b_k^h) + b_m^f$$

Annotations for the equation:
 

- $N_m$ : m outputs
- $k, n$ : single hidden layer with k units
- $w_{kn}^h$ : n inputs
- $w_{mk}^f$ : final
- $b_k^h$ : hidden

Our method can solve any equation that can be cast as an **optimisation problem**, i.e. can be brought into the form

$$\mathcal{F}_m(\vec{x}, \phi_m(\vec{x}), \nabla \phi_m(\vec{x}), \dots, \nabla^j \phi_m(\vec{x})) = 0$$

# Application to differential equations

(but tested also on integral equations)

Build the full function, here a DE into the loss function, incl boundary conditions

$$\mathcal{L}(\{w, \vec{b}\}) = \frac{1}{i_{\max}} \sum_{i,m} \hat{\mathcal{F}}_m(\vec{x}^i, \hat{\phi}_m(\vec{x}^i), \dots, \nabla^j \hat{\phi}_m(\vec{x}^i))^2 \\ + \sum_{\text{B.C.}} (\nabla^p \hat{\phi}_m(\vec{x}_b) - K(\vec{x}_b))^2,$$

identify trial solution with network output  $\hat{\phi}_m(\vec{x}) \equiv \check{N}_m(\vec{x}, \{w, \vec{b}\})$

Build the full function, here a DE into the loss function, incl. boundary conditions

- Each part of the NN is differentiable (activation functions are chosen such)  
-> derivatives can be obtained analytically (no loss of precision)
- Hyperparameters can be optimised for speed of convergence and accuracy:  
activation function, nr hidden layers, width of NN, learning rate, epochs...



## Example simple ODE

I first order: 
$$\frac{d\phi}{dx} + \left( x + \frac{1 + 3x^2}{1 + x + x^3} \right) \phi - x^3 - 2x - x^2 \frac{1 + 3x^2}{1 + x + x^3} = 0,$$

with boundary condition  $\phi(0) = 1$  in domain  $x \in [0, 2]$

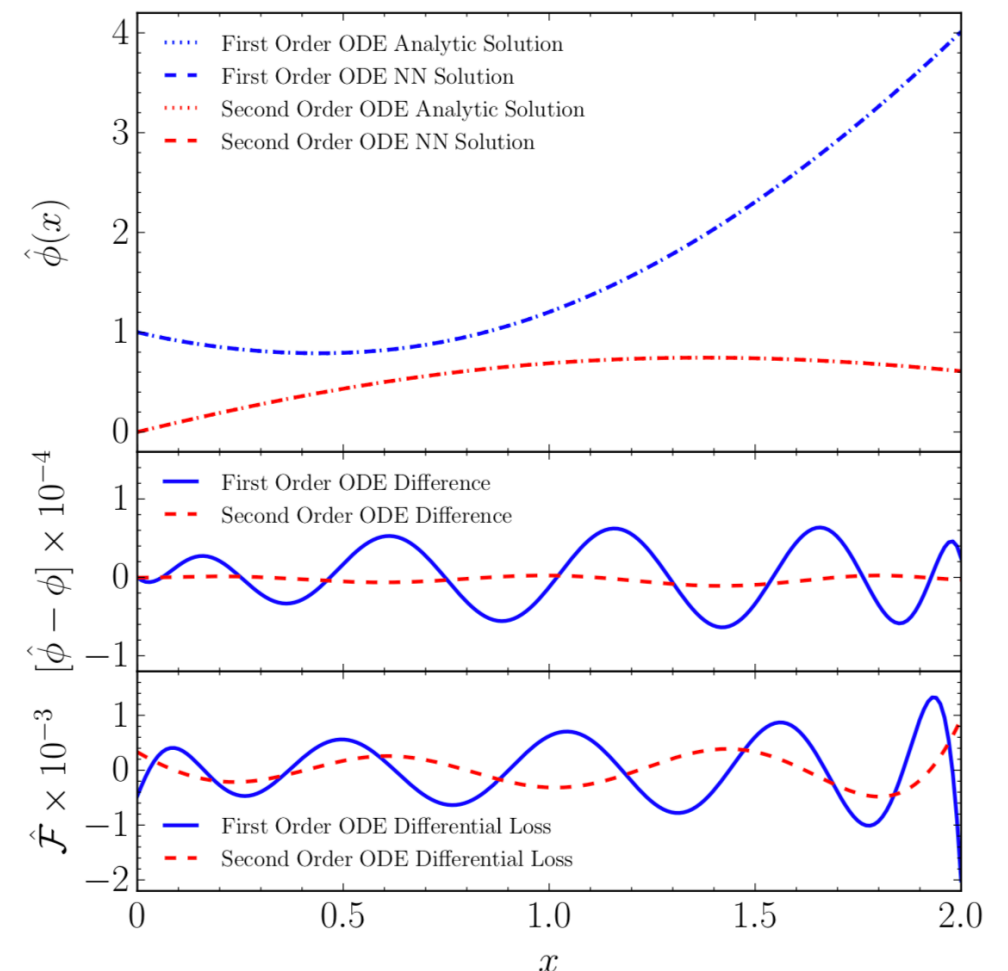
II second order: 
$$\frac{d^2\phi}{dx^2} + \frac{1}{5} \frac{d\phi}{dx} + \phi + \frac{1}{5} e^{-\frac{x}{5}} \cos x = 0$$

with boundary conditions  $\phi(0) = 0$  and

$$\frac{d}{dx}\phi(0) = 1 \text{ in domain } x \in [0, 2]$$

comparison between  
true and NN solution

true solution independent measure how well  
functional is solved in each point of domain



# Second example PDE

(but tested many more)

Differential equation:  $\nabla^2 \phi - e^{-x}(x - 2 + y^3 + 6y) = 0$

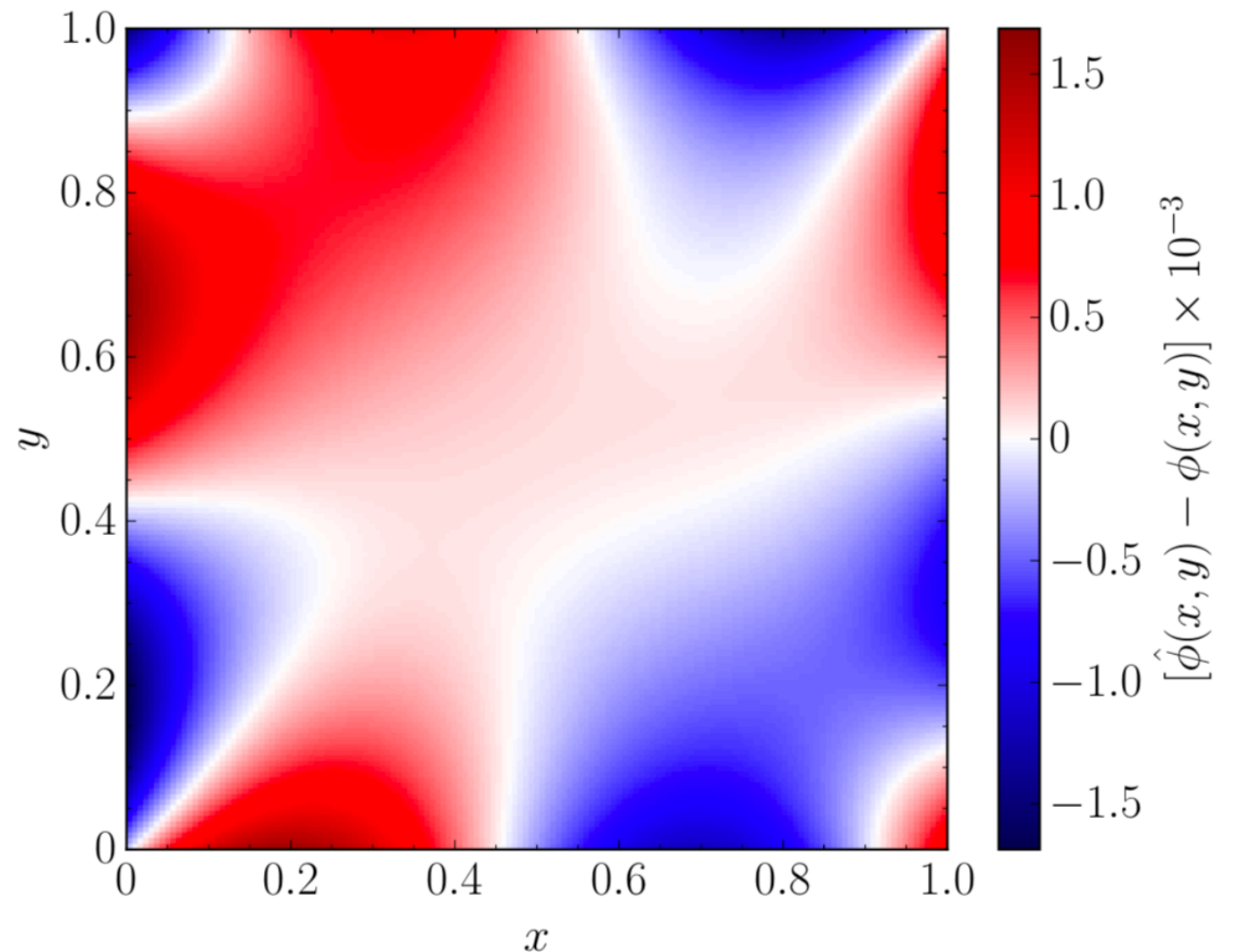
boundary conditions

$$\begin{aligned} \phi(0, y) &= y^3, & \phi(1, y) &= (1 + y^3)e^{-1}, \\ \phi(x, 0) &= xe^{-x}, & \phi(x, 1) &= e^{-x}(x + 1), \end{aligned}$$

domain  $(x, y) \in [0, 1] \times [0, 1]$

- Here deeper network pays off!
- Method also used for coupled PDE

difference between analytic and numeric solution



# A true physics problem to solve:

## Semiclassical calculations for bubbles and phase transitions

Need to find stationary points of Euclidean action:

$$S = \int d\tau d^3x \left[ \frac{1}{2} \left( \frac{\partial\phi}{\partial\tau} \right)^2 + \frac{1}{2} (\nabla\phi)^2 + V(\phi) \right]$$

Potential energy

Transition from false to true via tunnelling. Bubble can nucleate anywhere, with nucleation rate per unit volume:

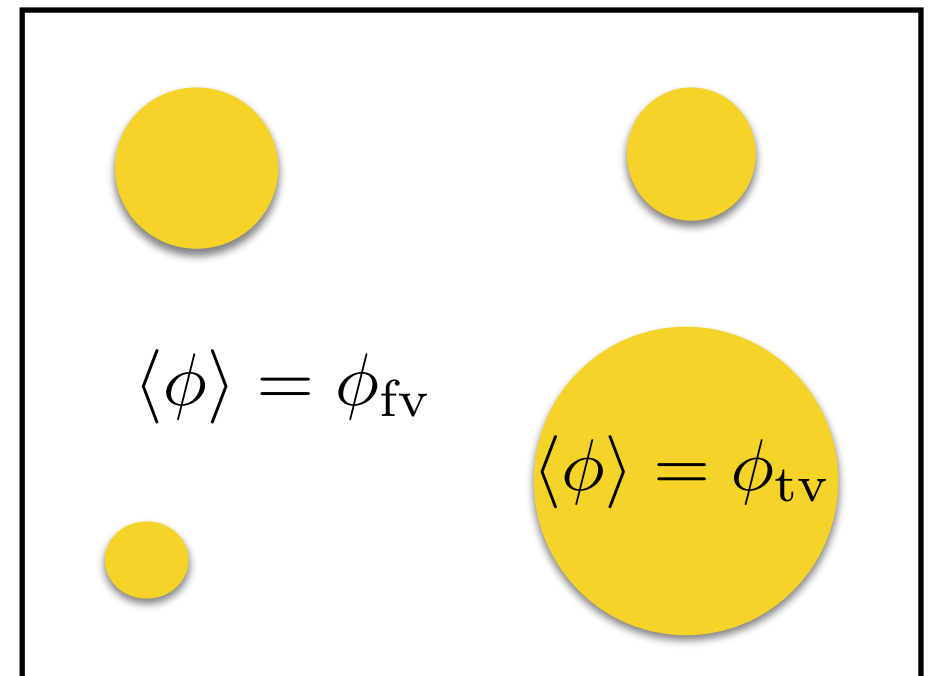
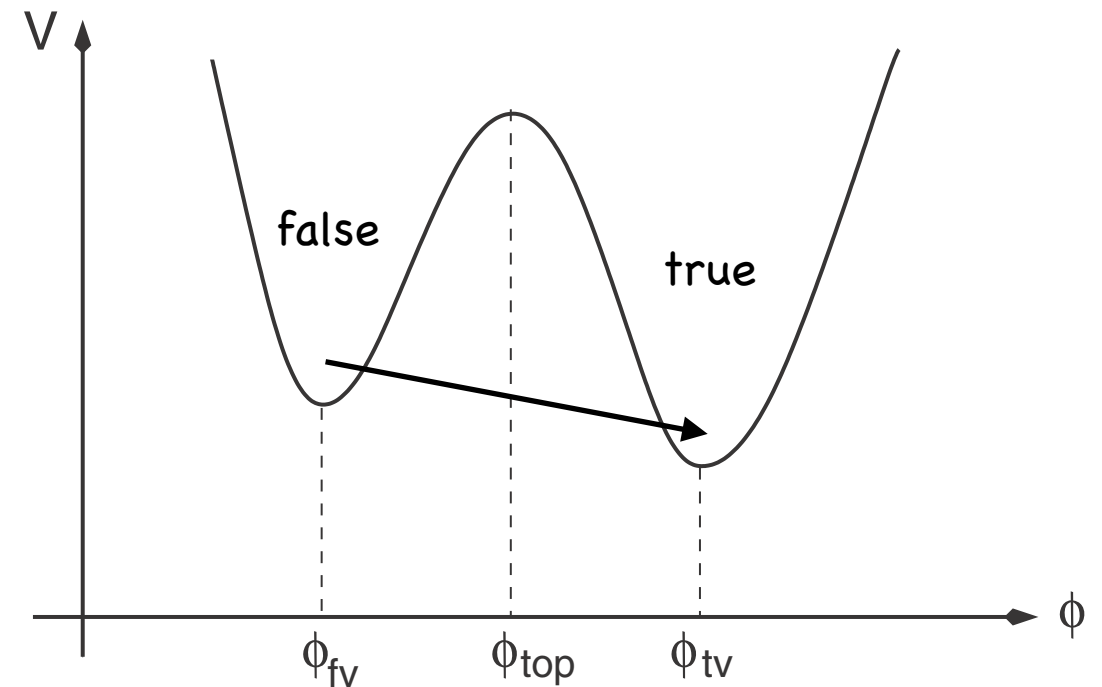
$$\frac{\Gamma}{\mathcal{V}} = Ae^{-B} \quad B = S(\phi_b) - S(\phi_{fv})$$

Growth of bubble via classical equation of motion

$$\left( -\frac{\partial^2}{\partial t^2} + \nabla^2 \right) \phi = \frac{\partial}{\partial\phi} V(\phi)$$

Methods to calculate bubble nucleation:

- Thin-wall approximation
- Polygon approximation [Guada, Maiezza, Nemevsek '18]
- over/undershoot method
- **Neural-Net approach** [Piscopo, MS, Waite '19]



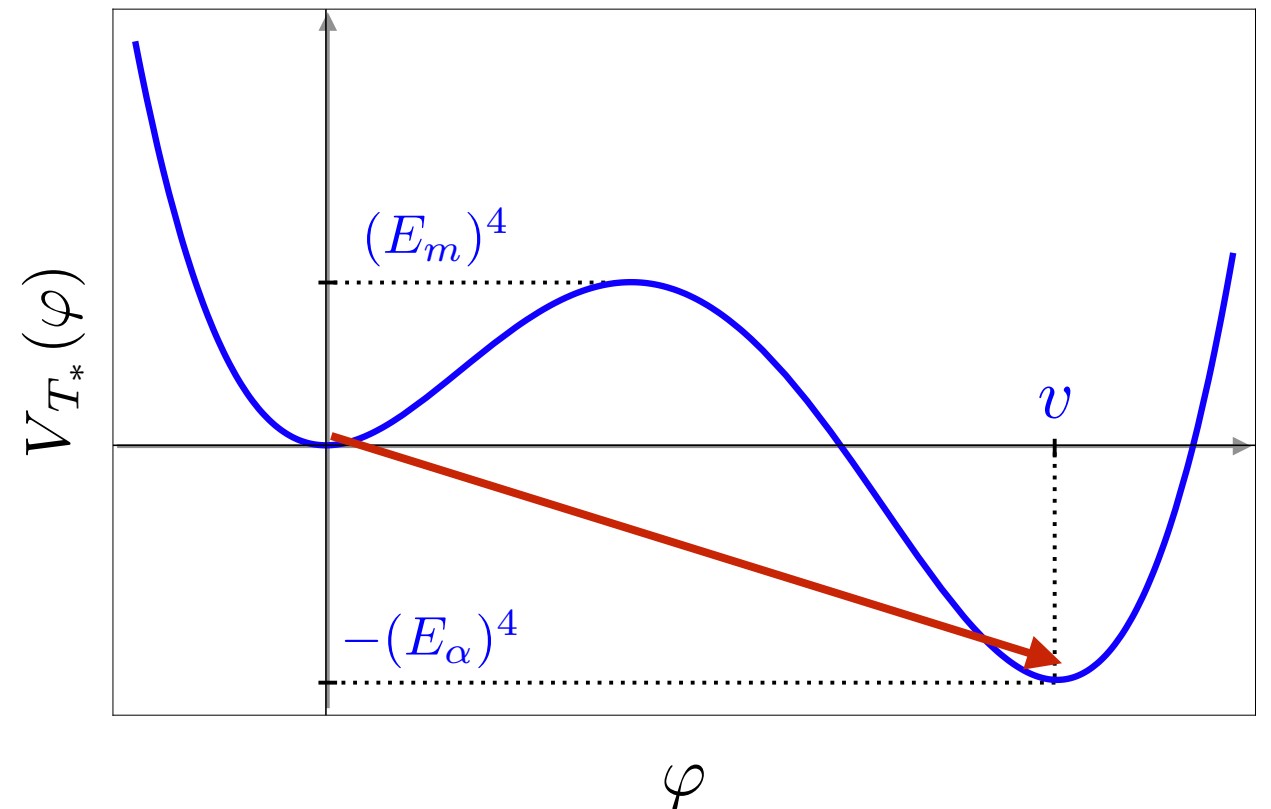


## Bubble nucleation rate

$$\Gamma(T) = A(T)e^{-S_3(T)/T}$$

## Duration of phase transition to Hubble rate

$$\frac{\beta}{H} = \left[ T \frac{d}{dT} \left( \frac{S_3(T)}{T} \right) \right] \Big|_{T=T_*}$$



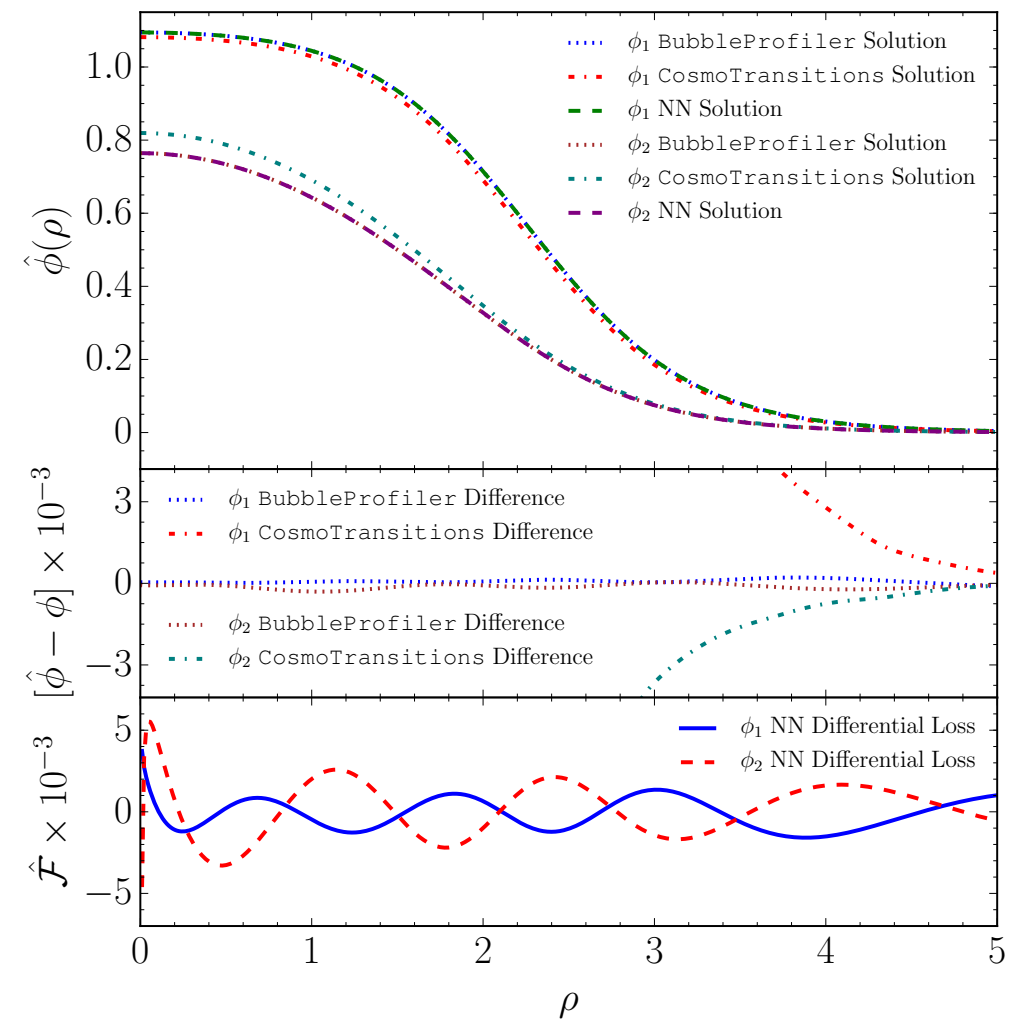
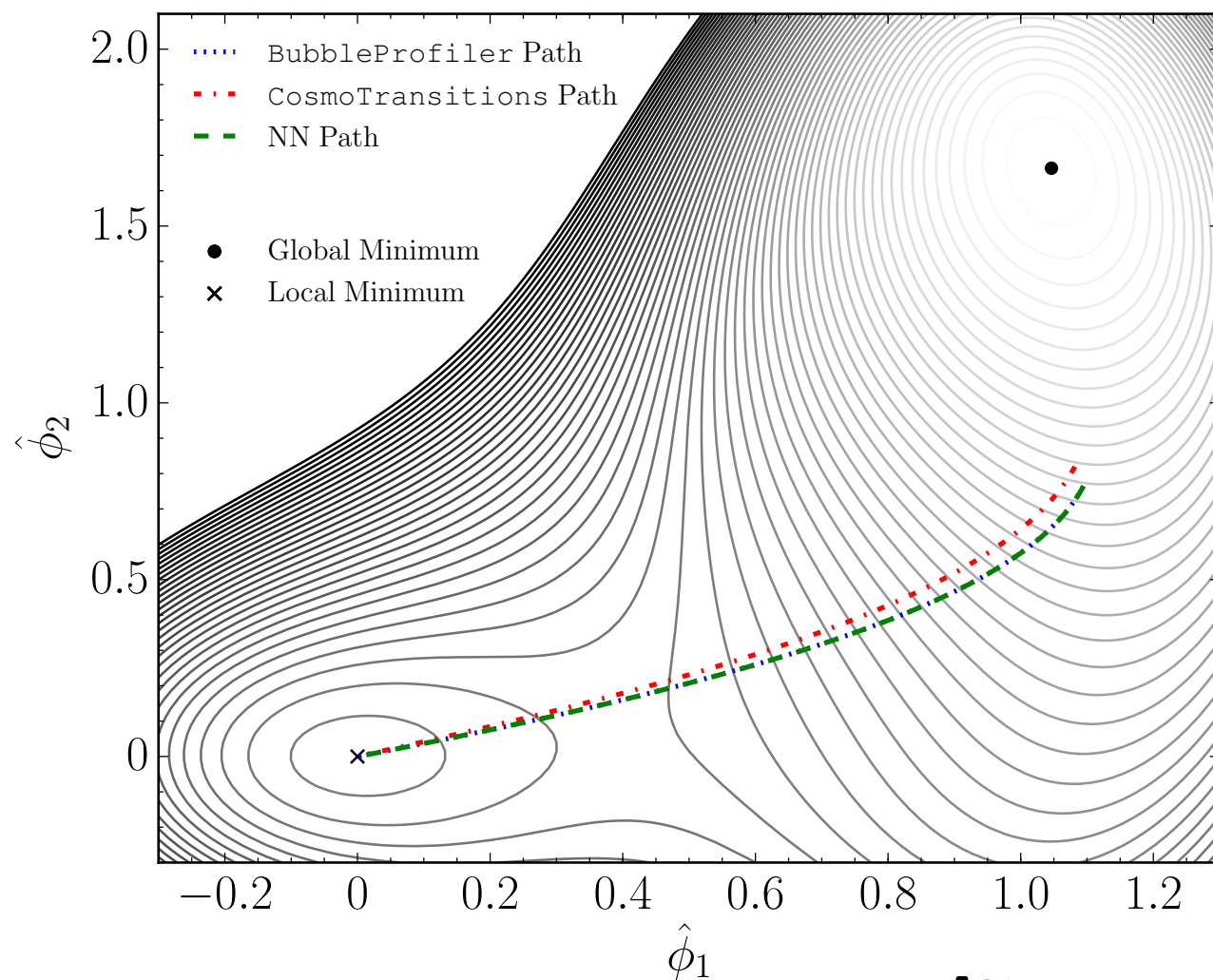
## Probability for a single bubble to nucleate within the horizon volume

$$P(T_*) = \int_{\infty}^{T_*} \frac{dT}{T} \left( \frac{2\zeta M_{\text{Pl}}}{T} \right)^4 \exp \left[ -\frac{1}{T} S_3^{\text{cl}}(T) \right] \quad \text{with} \quad \zeta^{-1} = 4\pi \sqrt{\pi g_*(T)/45} \quad [\text{Linde '83}]$$

To be of  $O(1)$  exponential factor should be compensated by prefactor

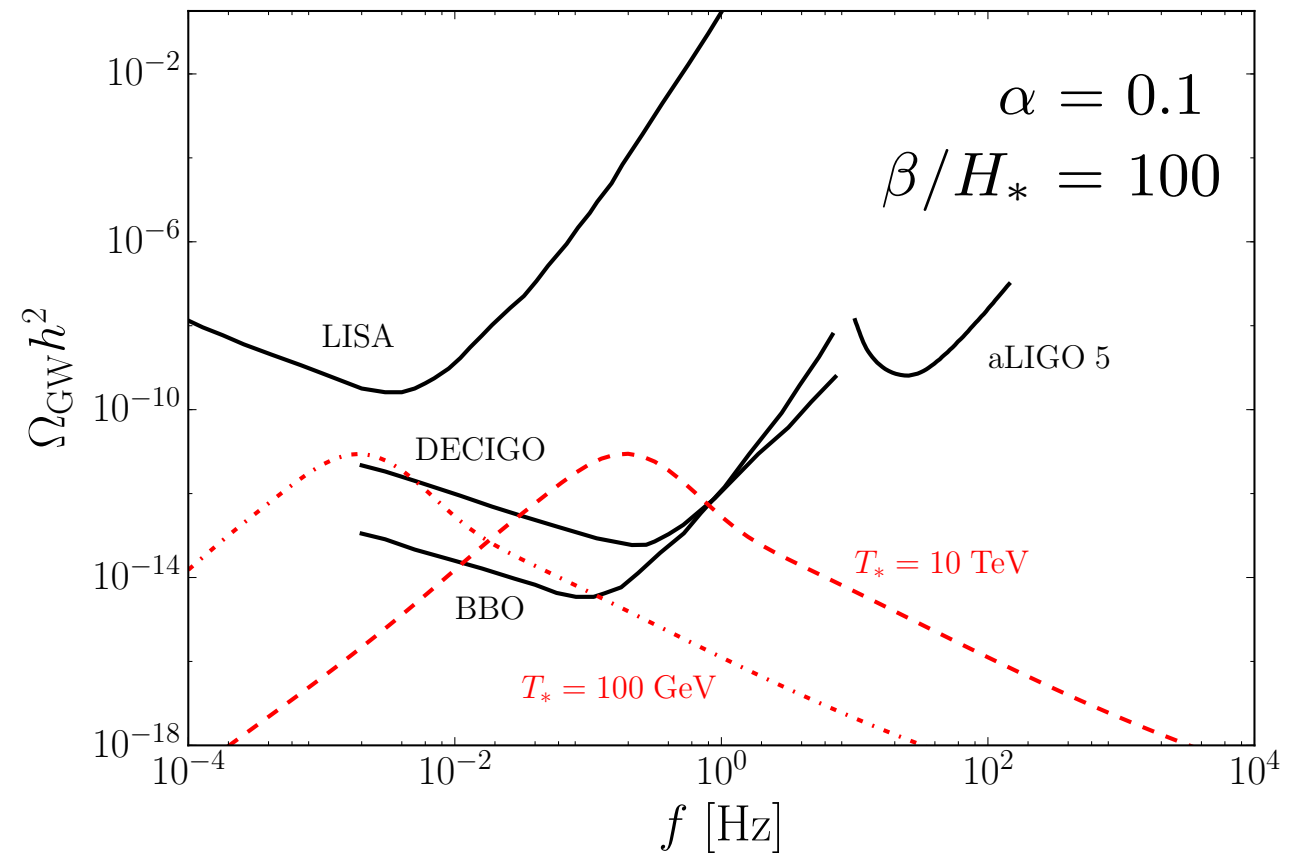
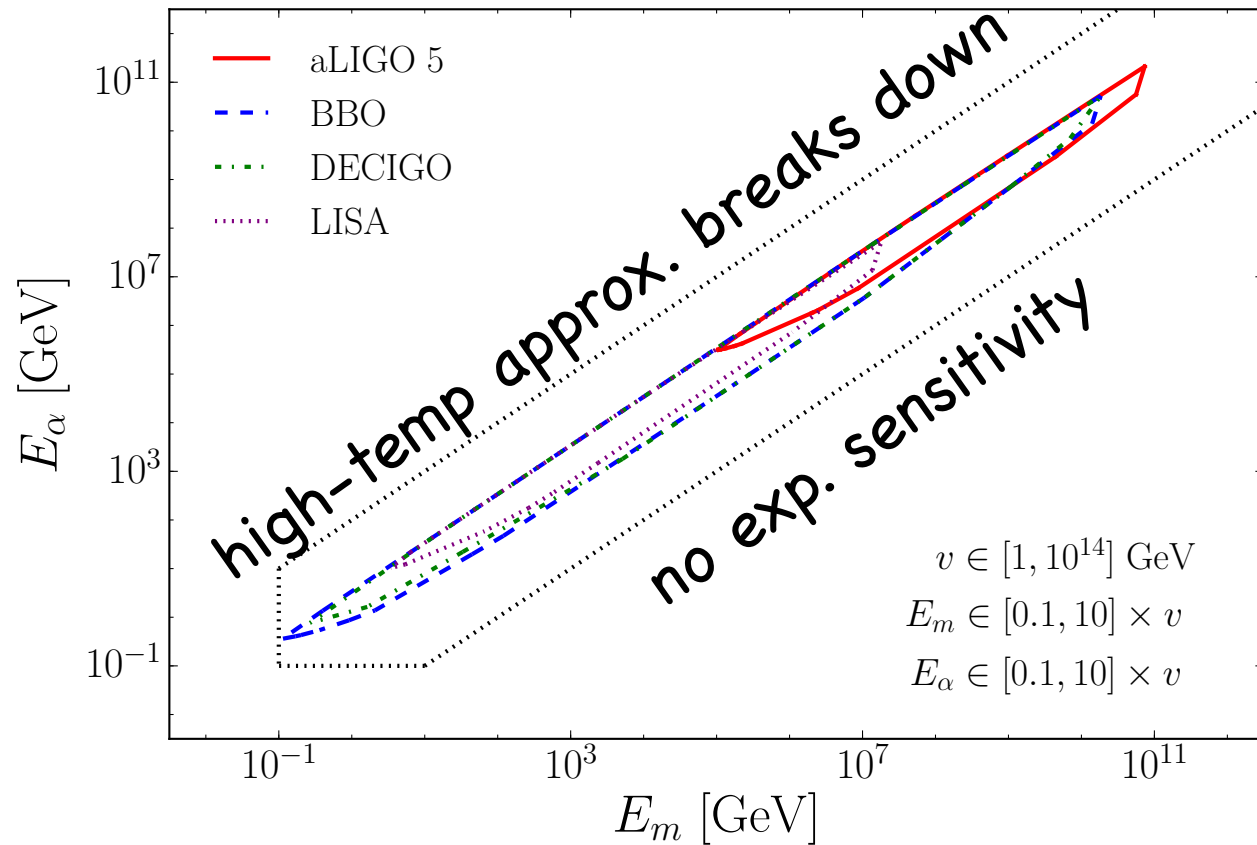
$$P(T_*) \simeq 1 \quad \Rightarrow \quad \frac{1}{T_*} S_3^{\text{cl}}(T_*) \simeq 140 - 4 \log \frac{T_*}{100 \text{ GeV}} \simeq 100$$

# Field profiles and tunnelling path



[Piscopo, MS, Waite '19]

- NN-approach often more reliable and more precise than dedicated PT solver, e.g. CosmoTransitions, BubbleProfiler
- $\hat{\mathcal{F}}$  is measure for quality of the solution
- Many applications beyond phase transitions



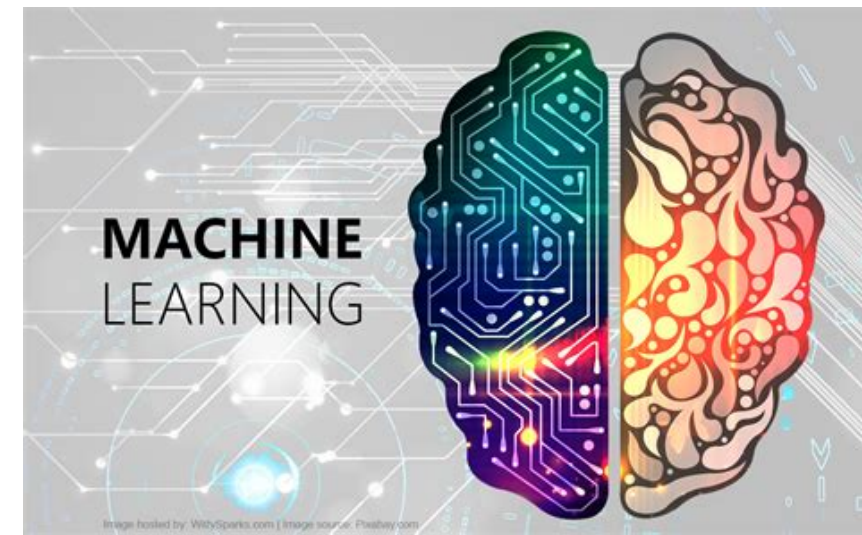
- All experiments (existing and future) are sensitive to the region  $E_\alpha \sim (0.1 - 10) \times E_m$
- aLIGO has sensitivity to high temperatures, LISA to ELW temperatures and large  $E_\alpha$
- Can check for generic models without using CosmoTransitions et. al. using approach and data from: [Chala, Khoze, MS, Waite '19]

<https://www.ippp.dur.ac.uk/~mspannow/gravwaves.html>





## Summary



- Machine Learning methods are powerful tools to address relevant physics questions, i.e. finding correlations in multi-dim parameter spaces, regression, anomaly detection, but also as numerical solvers
- Particle Physicist: Working on ML == probably more like using ML
- For theorists ML can be powerful in providing solutions to numerical problems