Machine Learning:
A new toolbox for Theoretical Physics

Juan Rojo

VU Amsterdam & Theory group, Nikhef

D-ITP Advanced Topics in Theoretical Physics

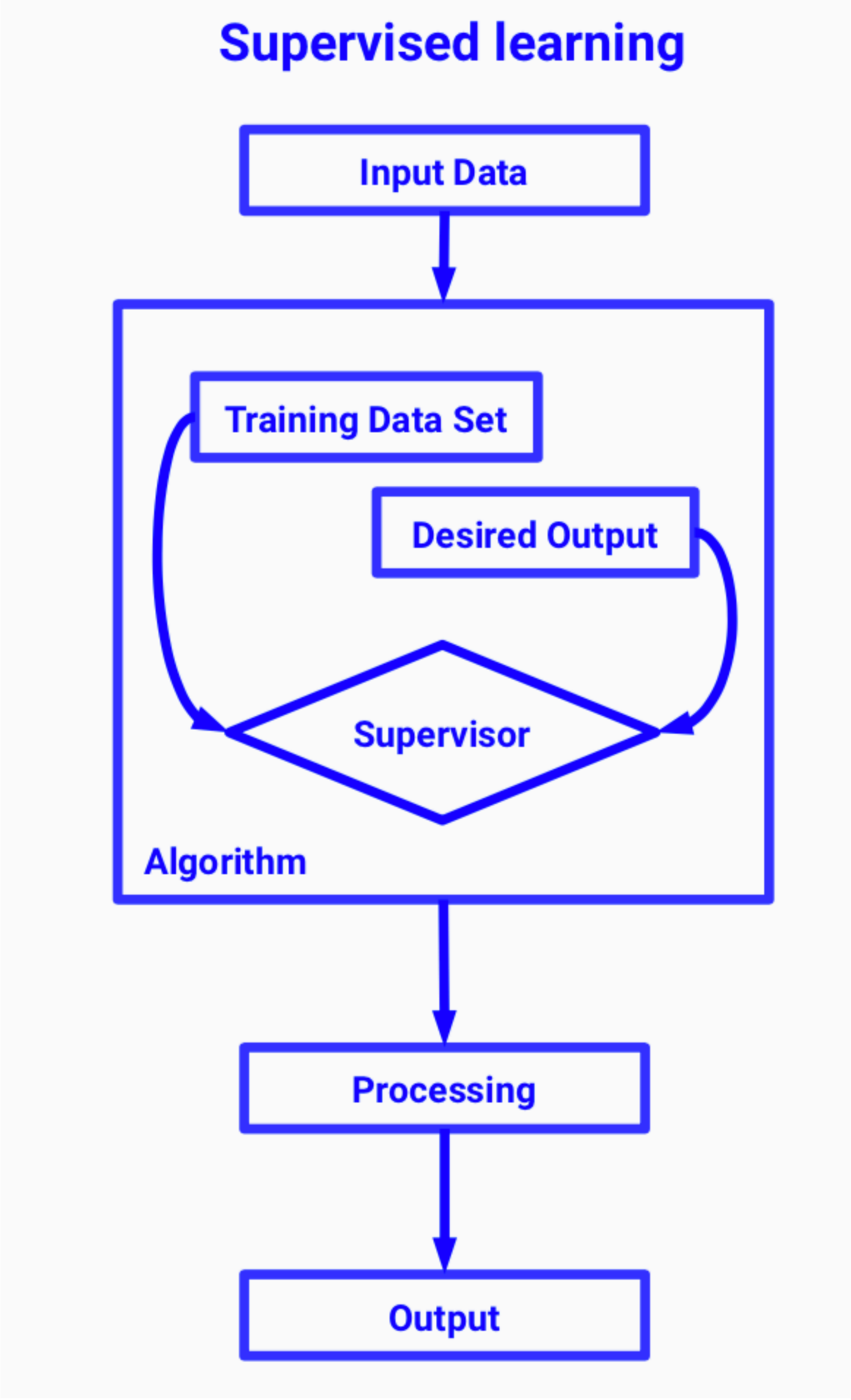
09/12/2019

Recap from Lectures 1+2+3

Learning to learn

Machine Learning algorithms can be divided into several classes, including

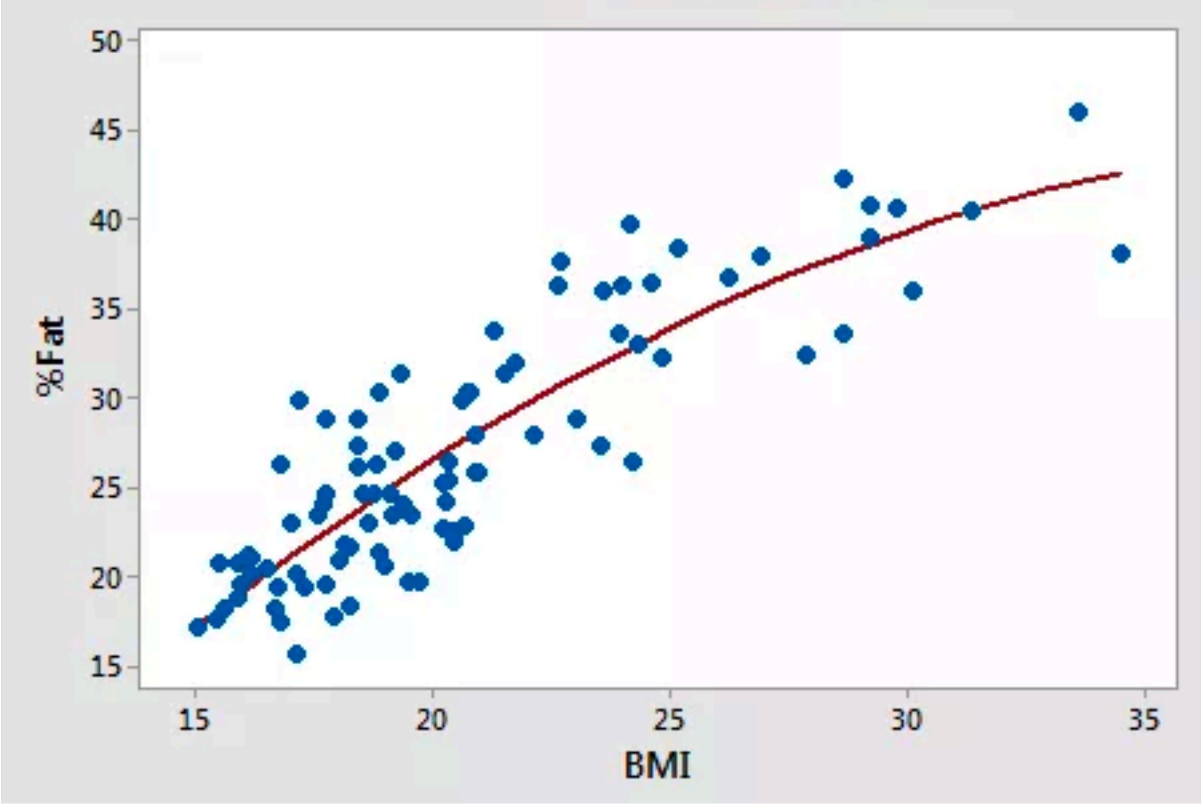
- **Supervised Learning**:
  regression, classification, …

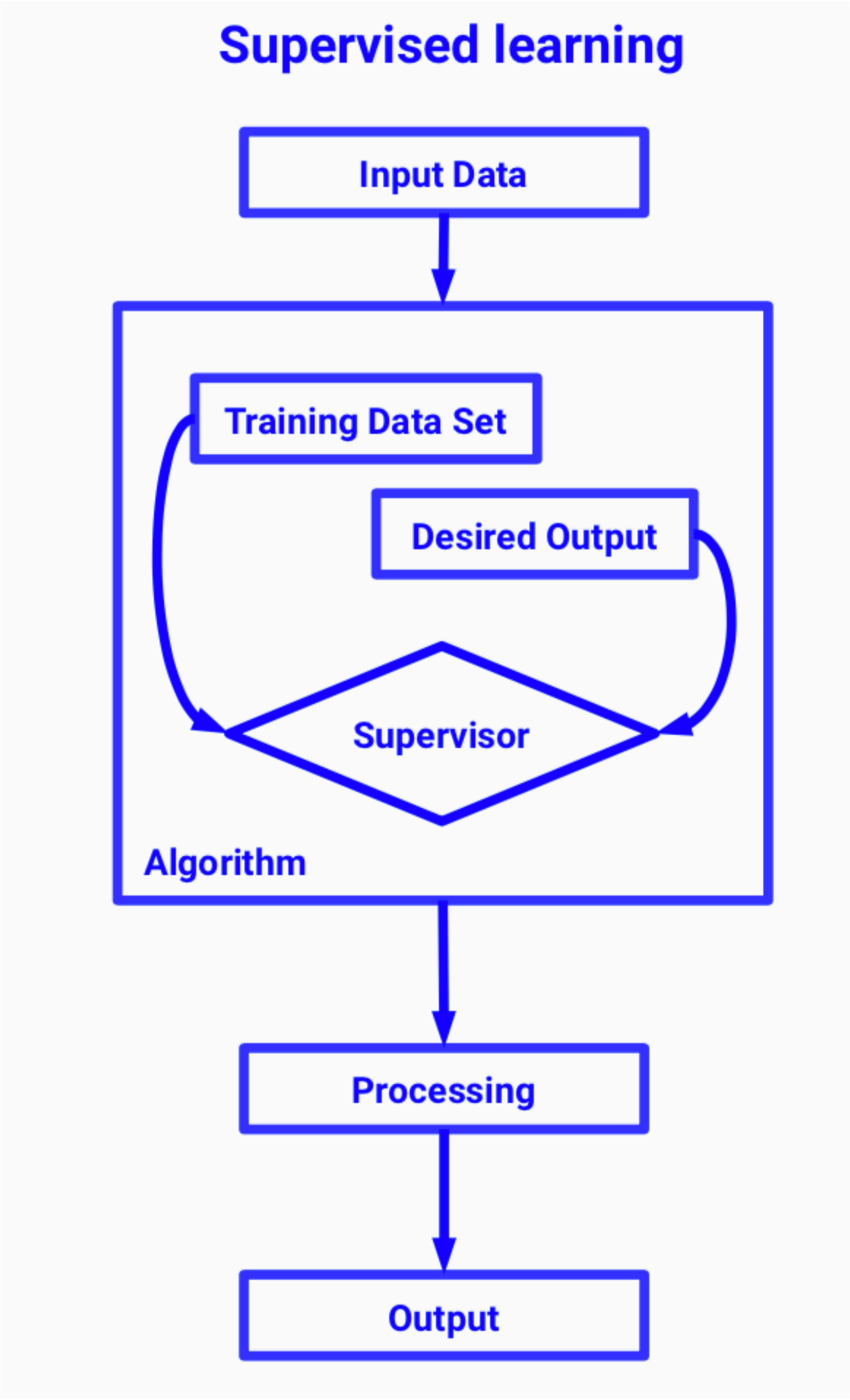


Learning to learn

Machine Learning algorithms can be divided into several classes, including

- **Supervised Learning**: regression, classification, …



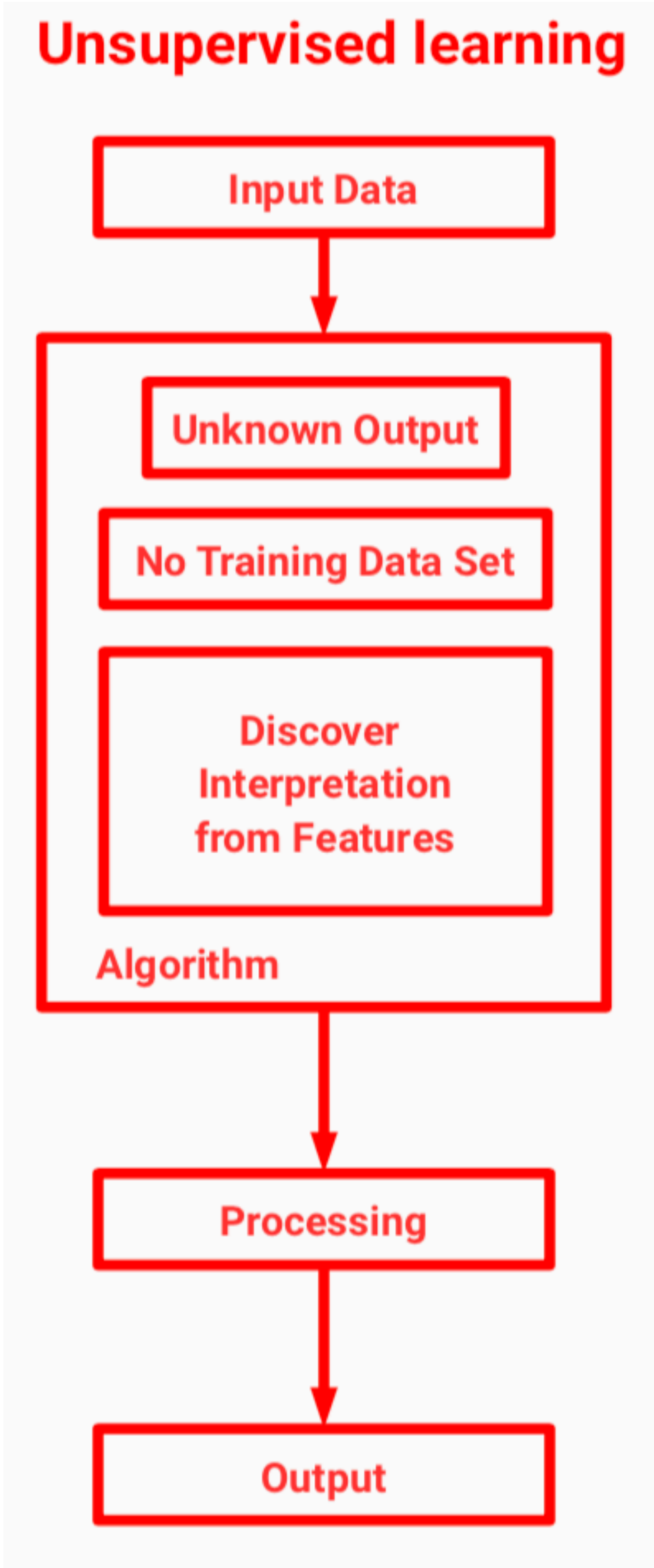


Learning to learn

Machine Learning algorithms can be divided into several classes, including

- **Supervised Learning**:
  regression, classification, …

- **Unsupervised Learning**:
  clustering, data dimensional reduction, ….

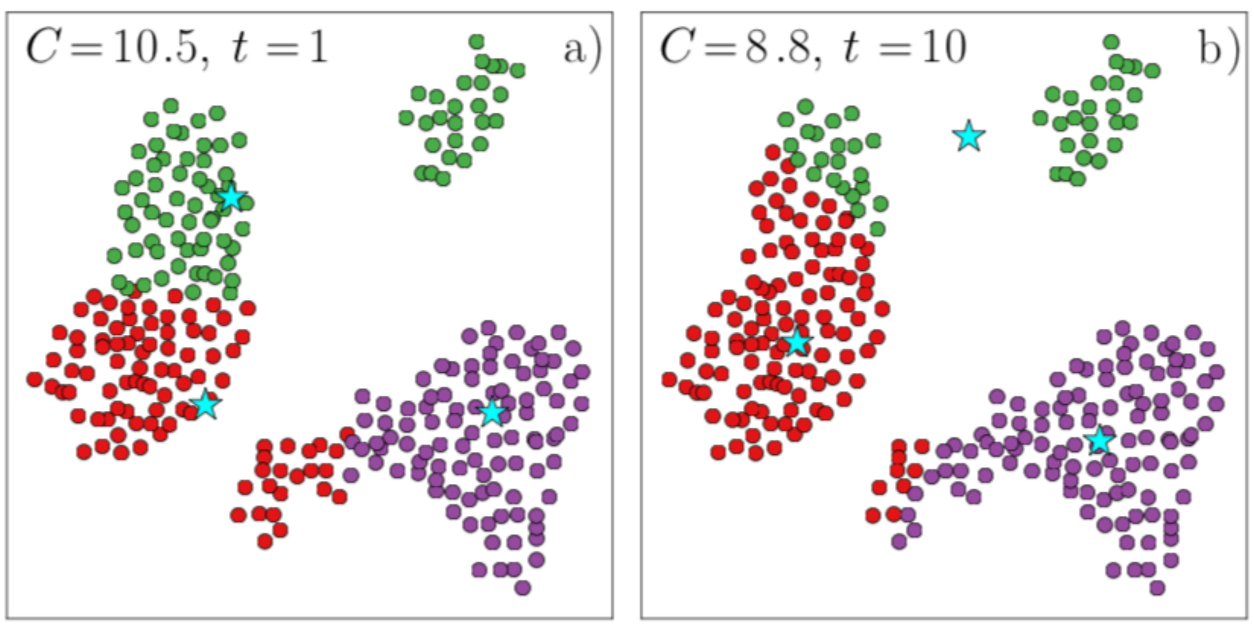


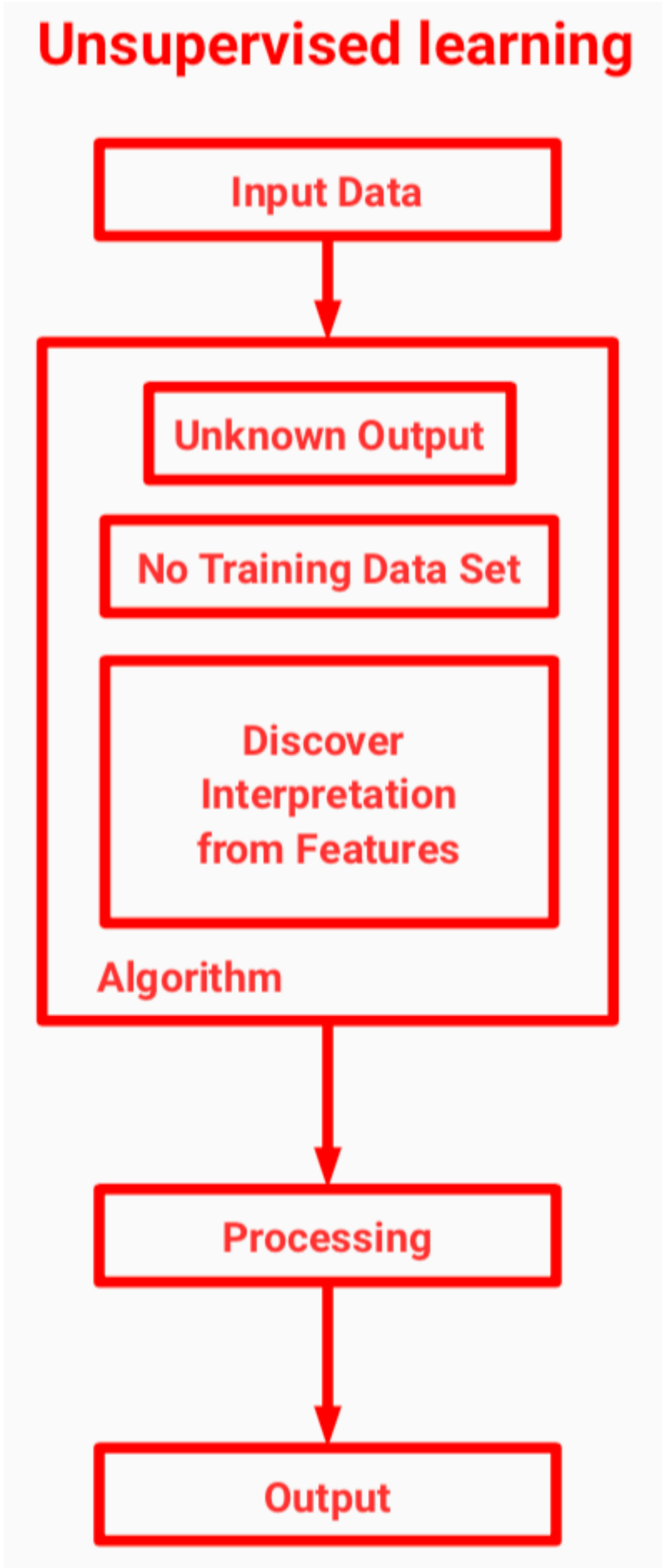
Learning to learn

Machine Learning algorithms can be divided into several classes, including

- **Supervised Learning**: regression, classification, …

- **Unsupervised Learning**: clustering, data dimensional reduction, ….





Learning to learn

Machine Learning algorithms can be divided into several classes, including

- **Supervised Learning**: regression, classification, …
- **Unsupervised Learning**: clustering, data dimensional reduction, ….
- **Reinforcement learning**: efficiently react to changing environment

Learning to learn

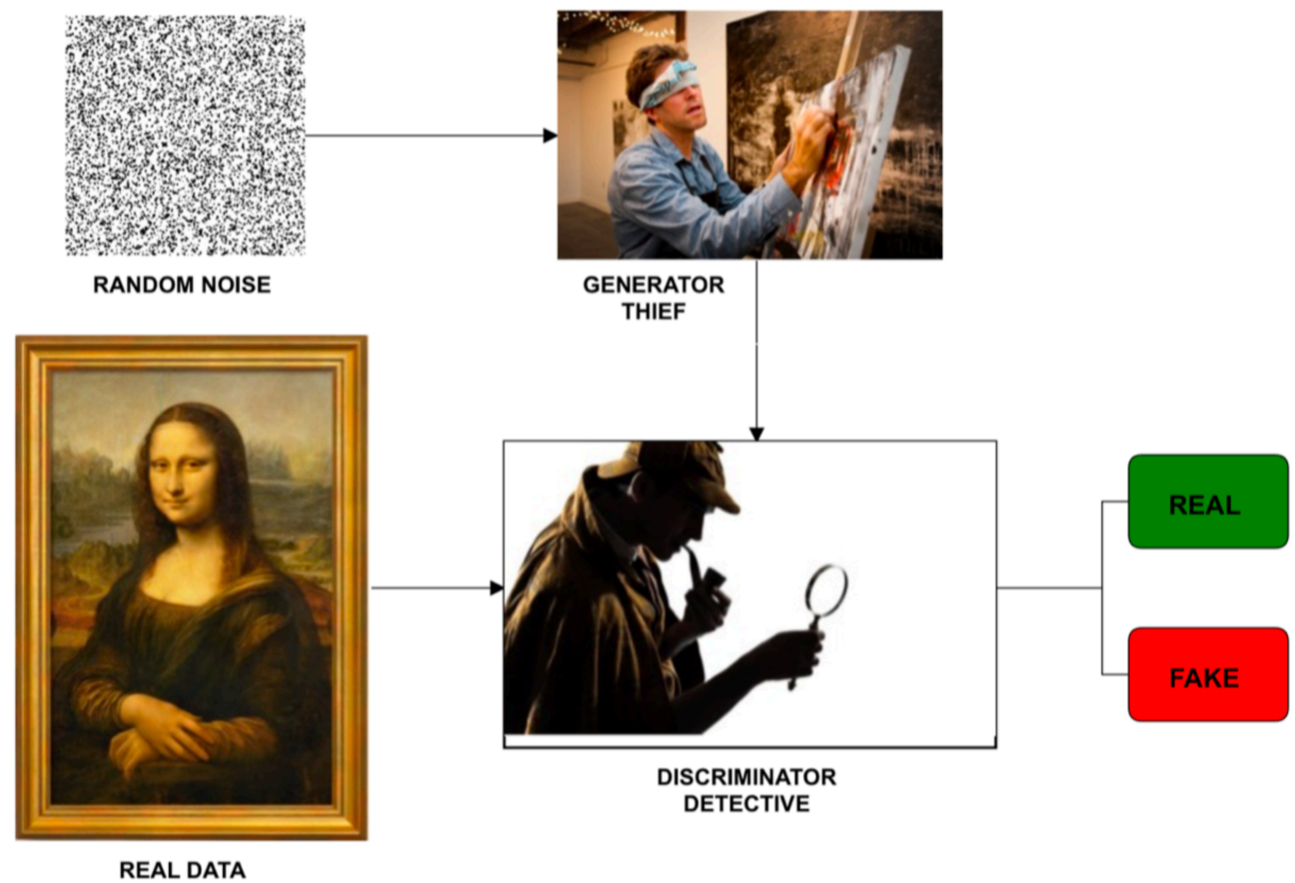
Machine Learning algorithms can be divided into several classes, including

- **Reinforcement learning**: efficiently react to changing environment

Learning to learn

Machine Learning algorithms can be divided into several classes, including

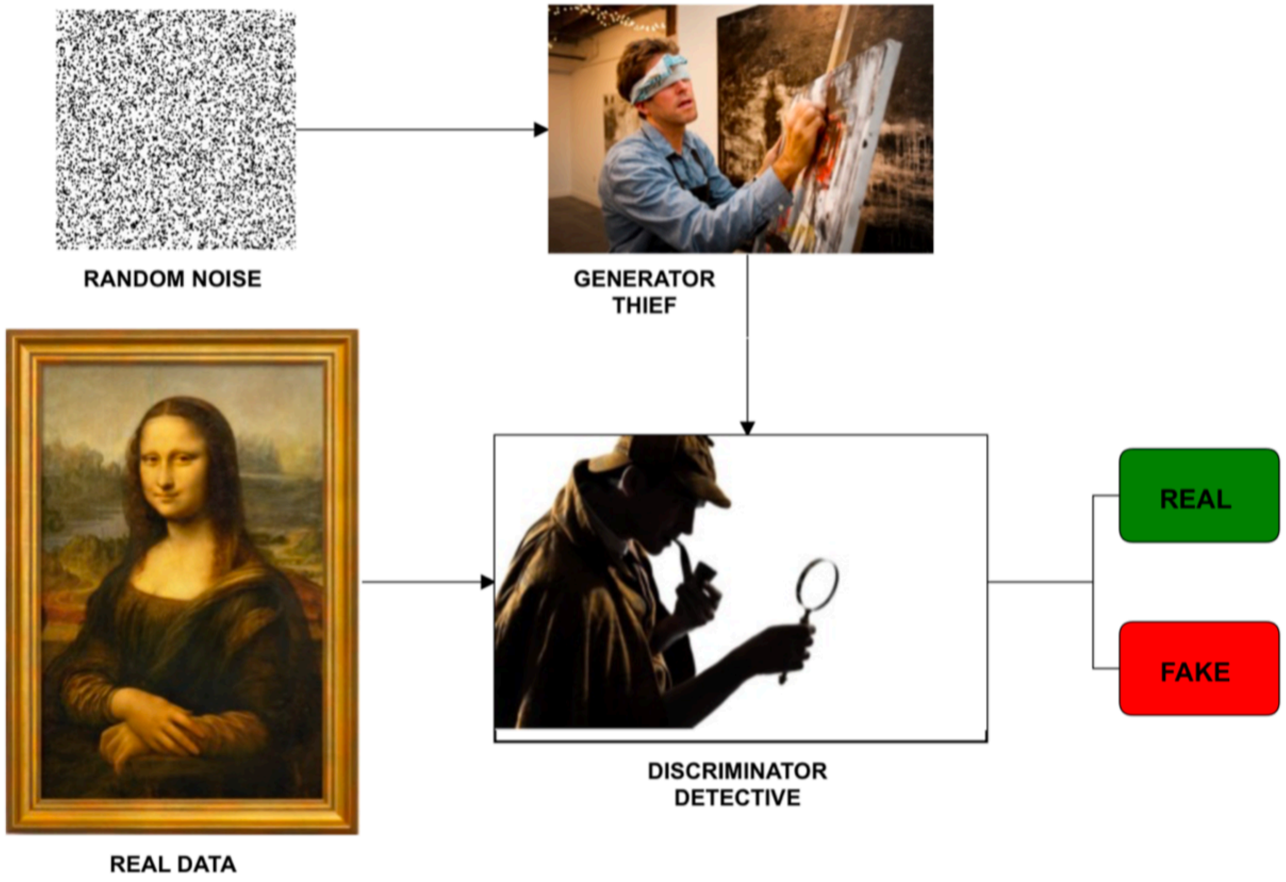
- **Supervised Learning**: regression, classification, …
- **Unsupervised Learning**: clustering, data dimensional reduction, ….
- **Reinforcement learning**: efficiently react to changing environment
- **Adversarial learning**: learn by internal competition



this lecture!

Learning to learn

Machine Learning algorithms can be divided into several classes, including



- **Adversarial learning**: learn by internal competition

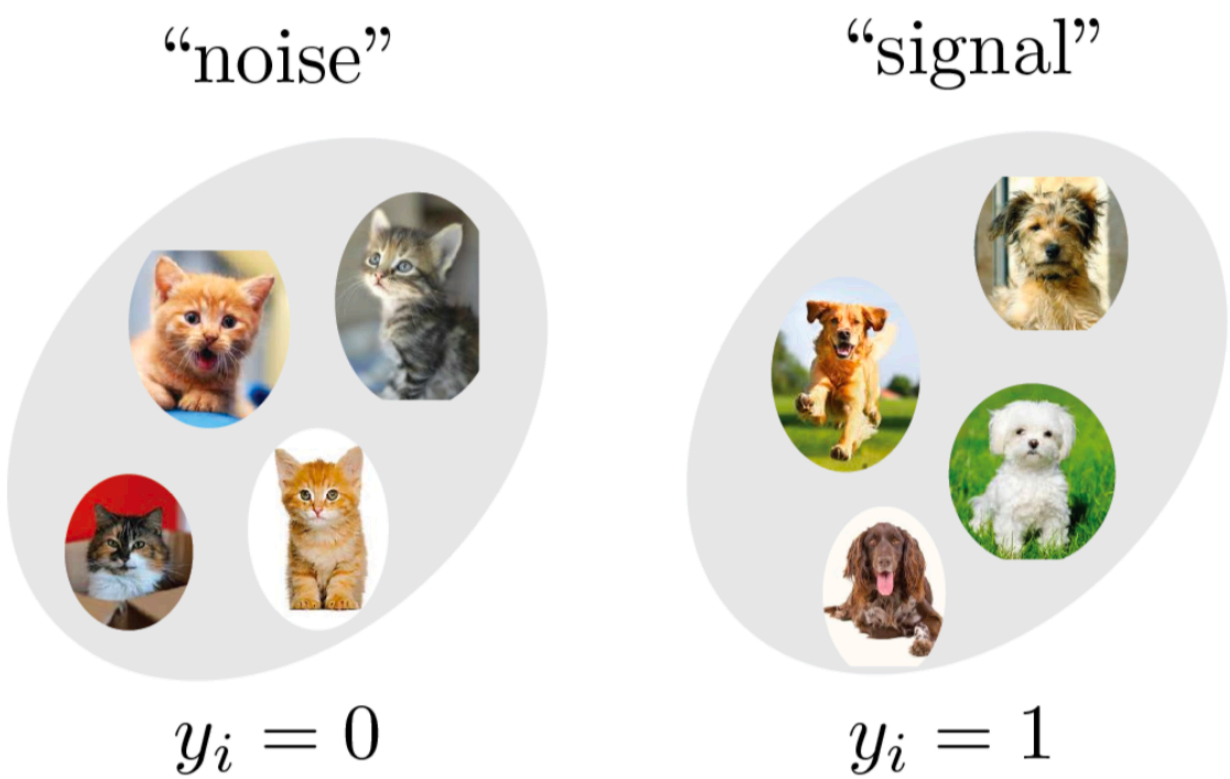
Today's lecture

- Logistic regression: the Higgs pair production case
- Energy-based models and Boltzmann learning
- Generative Models and Adversarial Learning
- Convolutional Neural Networks
- Applications of Machine Learning to Theoretical Physics

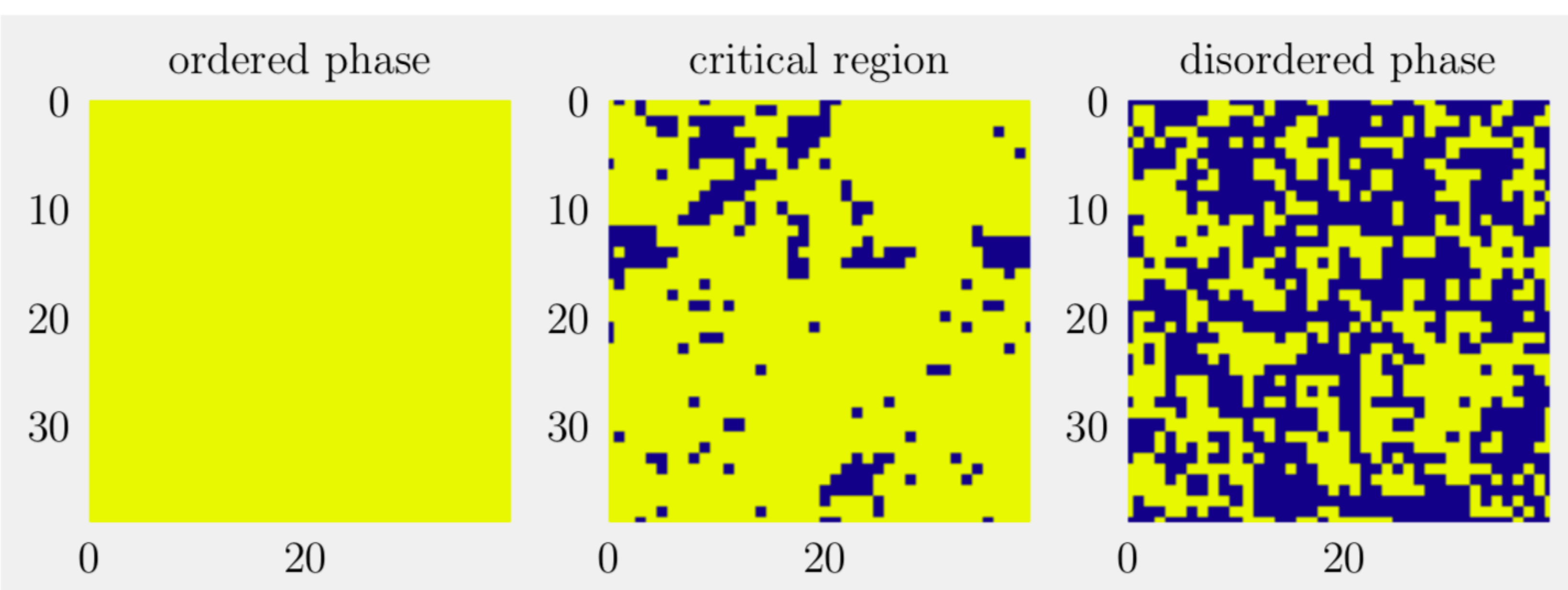
Case Study:
Higgs Pair Production

Logistic regression

Relevant for Machine Learning applications where outcomes are discrete variables, eg. **categories in classification problems**



can we tell apart cats from dogs?



can we identify the phase (ordered/disordered) of spin configurations in 2D Ising?

Logistic regression

cost function for logistic regression derived from Maximum Likelihood Estimation (MLE): choose parameters that maximise the probability of seeing the observed data

$$\mathscr{L}(\boldsymbol{\theta}\,|\,\mathscr{D}) = \prod_{i=1}^{n} P(y_i\,|\,\boldsymbol{x}_i, \boldsymbol{\theta}) = \prod_{i=1}^{n} \left(\sigma(\boldsymbol{x}_i^T\boldsymbol{\theta})\right)^{y_i} \times \left(1 - \sigma(\boldsymbol{x}_i^T\boldsymbol{\theta})\right)^{1-y_i}$$

Since the cost function is the negative log-likelihood, we find that for logistic regression

$$E(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left(-y_i \log \sigma(\boldsymbol{x}_i^T\boldsymbol{\theta}) - (1-y_i)\log(1-\sigma(\boldsymbol{x}_i^T\boldsymbol{\theta})\right)$$

cross-entropy

model for the classification probability

The parameters of the model are determined by minimising the cross-entropy

$$\widehat{\theta} = \arg\,\min_{\theta}\left\{ \sum_{i=1}^{n} \left(-y_i \log \sigma(\boldsymbol{x}_i^T\boldsymbol{\theta}) - (1-y_i)\log(1-\sigma(\boldsymbol{x}_i^T\boldsymbol{\theta})\right) \right\}$$

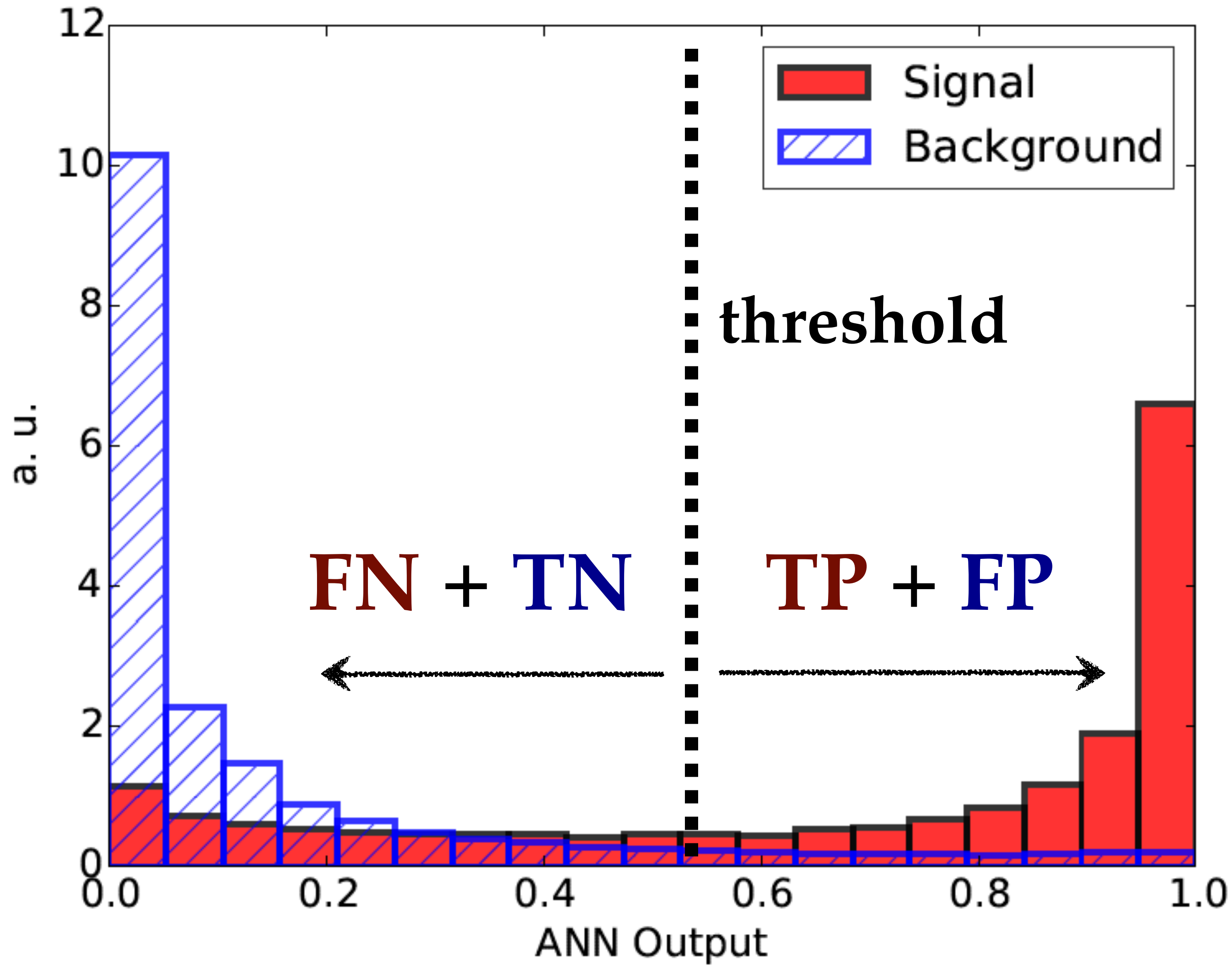
note that no analytic solution is possible, and numerical methods are required

one can use more complex models to parametrise the probabilities, such as NNs

ROC curve

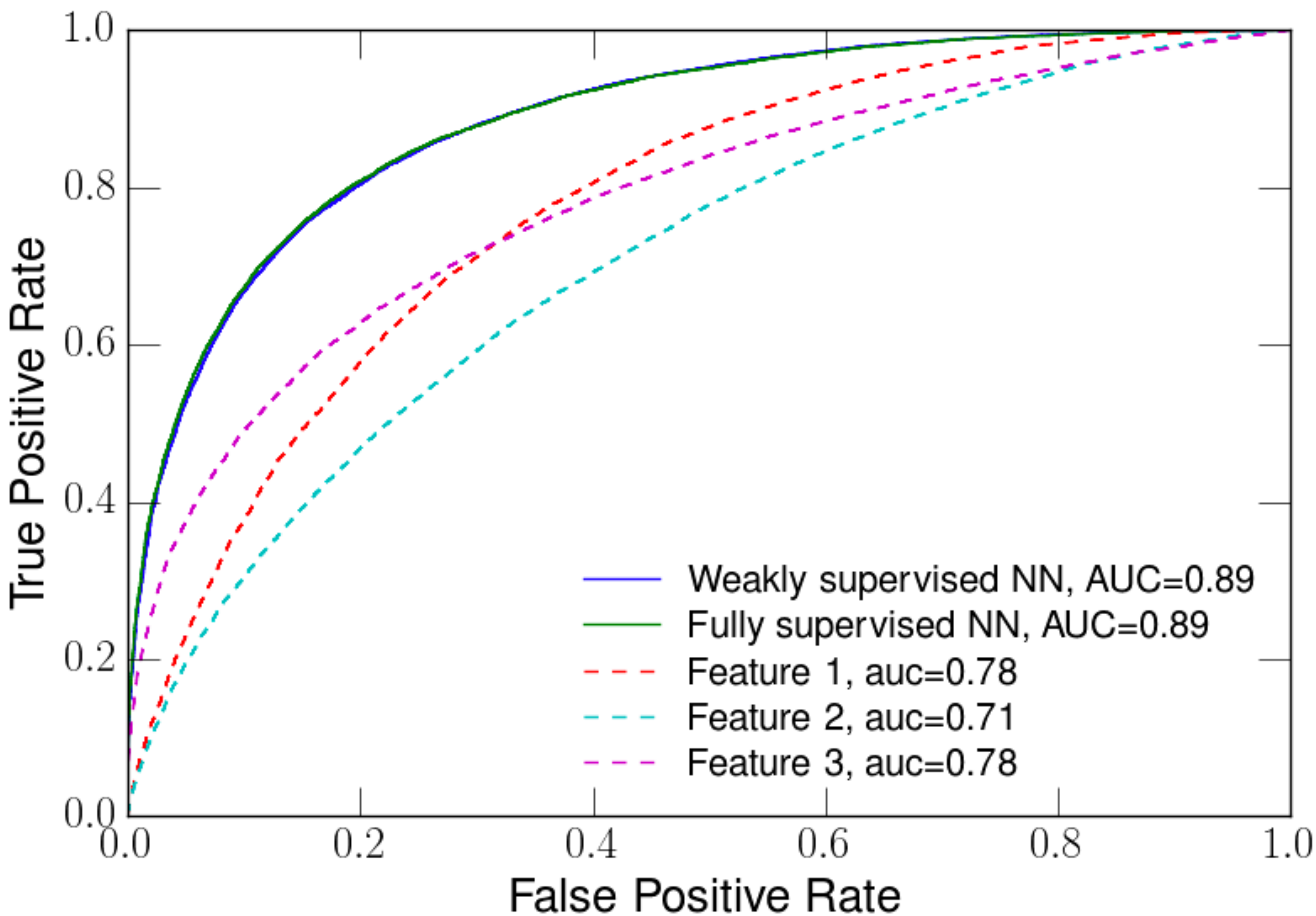
In most ML classification problems there is a **threshold** that can be varied to decide at what output of the model a given data point is assigned to each category:

- ☑ A conservative threshold will **maximise TP**, but also FP might be large
- ☑ An aggressive threshold **reduces FP** but then FN might be large.

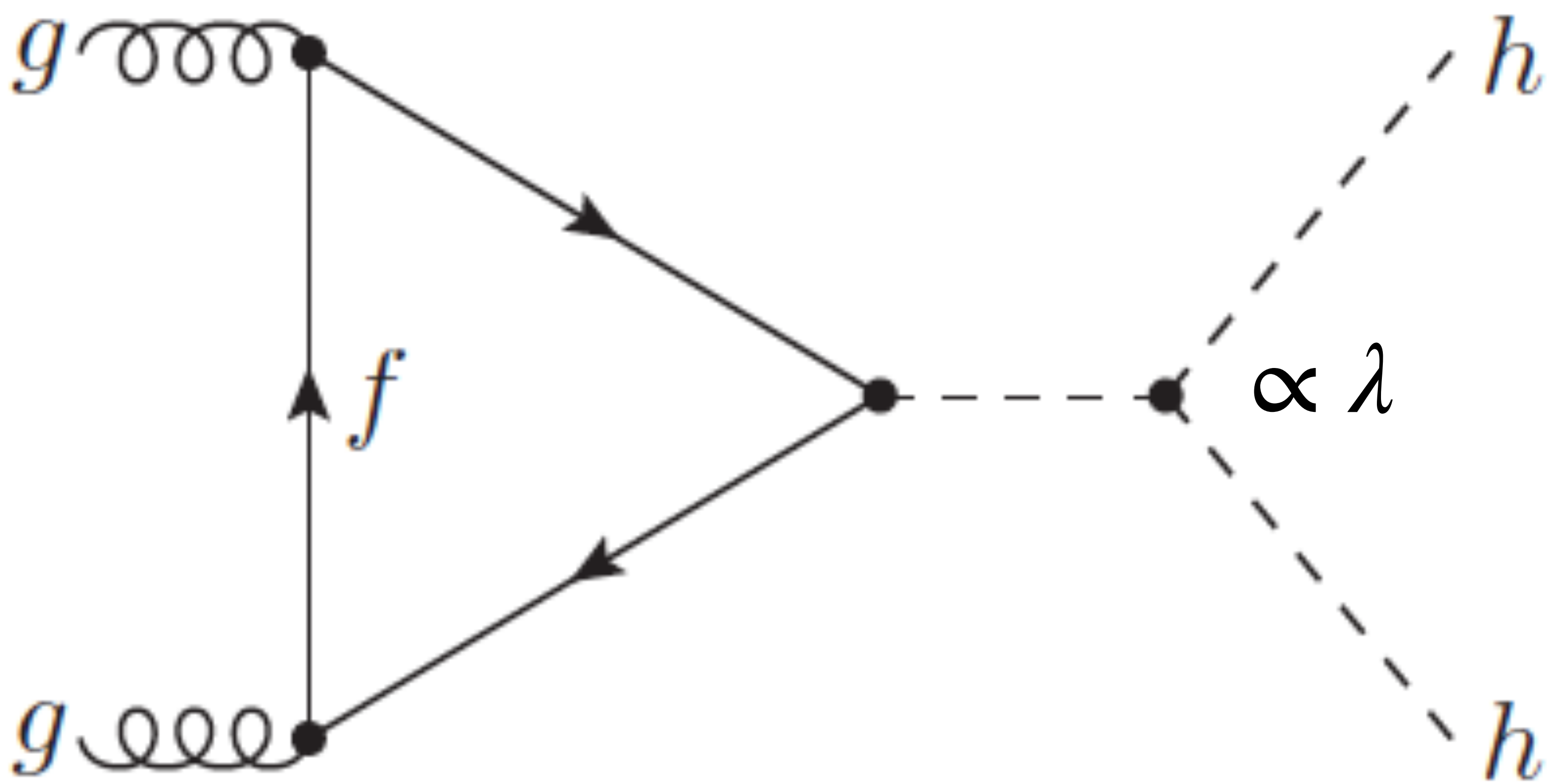


ROC curve

$$\text{Recall} = \text{True Positive Rate} = \frac{\text{TP}}{\text{TP} + \text{FN}} \qquad \text{False Positive Rate} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$



Higgs Pair Production

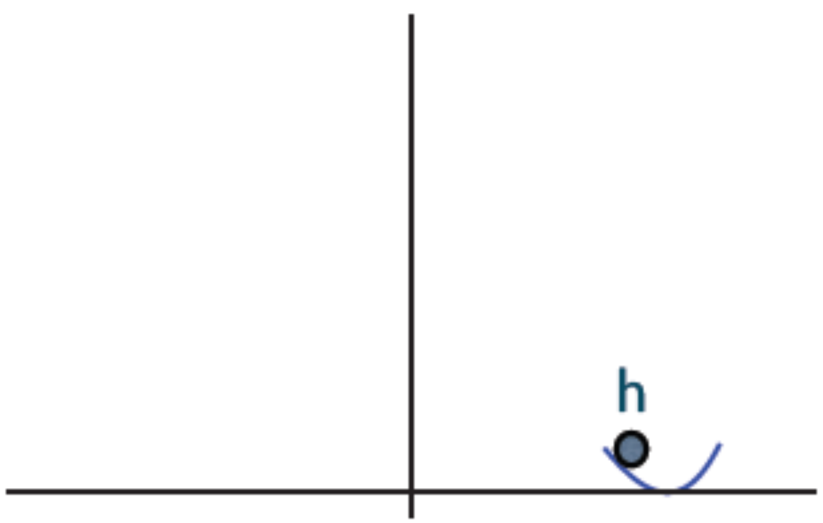


$$V(h) = m_h^2 h^\dagger h + \frac{1}{2}\lambda(h^\dagger h)^2$$

Due to small rates in the SM, only final states where at least one of the Higgs bosons decays into a **bb pair** are experimentally accessible at the LHC

Higgs Pair Production

- Current measurements in single Higgs production probe **Higgs potential close to minimum**
- Double Higgs production essential to **reconstruct full potential** and clarify EWSB mechanism
- In the SM the Higgs potential is fully *ad-hoc*: **many other EWSB mechanisms** conceivable



Higgs mechanism

$$V(h) = m_h^2 h^\dagger h + \frac{1}{2}\lambda(h^\dagger h)^2$$

Coleman-Weinberg mechanism

$$V(h) \to \frac{1}{2}\lambda(h^\dagger h)^2 \log\left[\frac{(h^\dagger h)}{m^2}\right]$$

Each possibility associated to **completely different EWSB mechanism**, with crucial implications for the **hierarchy problem**, the structure of quantum field theory, and **New Physics at the EW scale**

Arkani-Hamed, Han, Mangano, Wang, arxiv:1511.06495

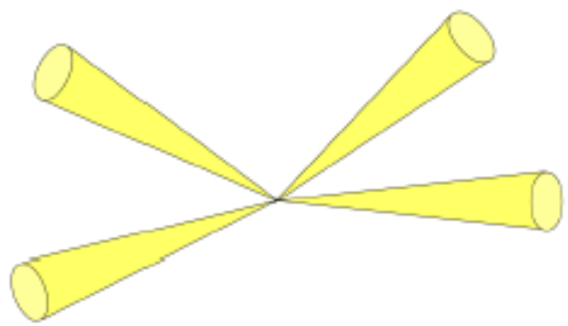
ggF di-Higgs in the 4b final state

- The **4b final state** offers largest rates but overwhelming QCD multijet backgrounds
- Made competitive requiring the **di-Higgs system to be boosted** and exploiting kinematic differences between signal and QCD background with **jet substructure**
- **Boosted b-tagging** by ghost-associating mass-drop tagged large-*R* jets with b-tagged small-*R* jets
- **Tagging these events** is challenging due to the very high rate of QCD multijets but doable
- **Scale-invariant tagging:** event-by-event classification depending on final state topology

Process	Generator	N_{evt}	σ_{LO} (pb)	K-factor
$pp \to hh \to 4b$	`MadGraph5_aMC@NLO`	1M	$6.2 \cdot 10^{-3}$	2.4 (NNLO+NNLL [18, 19])
$pp \to b\bar{b}b\bar{b}$	`SHERPA`	3M	$1.1 \cdot 10^{3}$	1.6 (NLO [63])
$pp \to b\bar{b}jj$	`SHERPA`	3M	$2.7 \cdot 10^{5}$	1.3 (NLO [63])
$pp \to jjjj$	`SHERPA`	3M	$9.7 \cdot 10^{6}$	0.6 (NLO [77])
$pp \to t\bar{t} \to b\bar{b}jjjj$	`SHERPA`	3M	$2.5 \cdot 10^{3}$	1.4 (NNLO+NNLL [78])

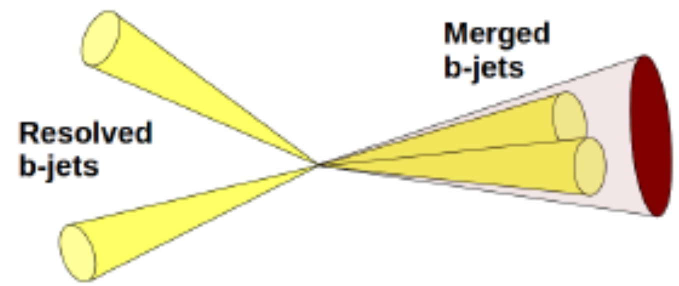
Analysis strategy

Resolved



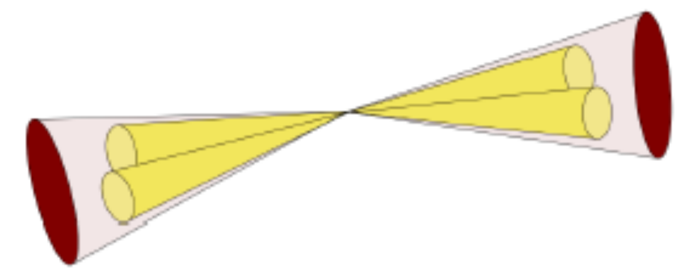
- ≥ 4 b-tagged small-R jets
- Higgs reconstruction from leading 4 jets
- Choice that minimises mass difference between dijet systems

Intermediate



- $= 1$ large-R jet (Higgs-tagged + b-tagged) (leading Higgs)
- ≥ 2 b-tagged small-R jets
- $\Delta R > 1.2$ w.r.t. large-R jet
- Higgs reconstruction from leading 2 small-R jets
- Choice that minimises mass difference of dijet system and large-R jet

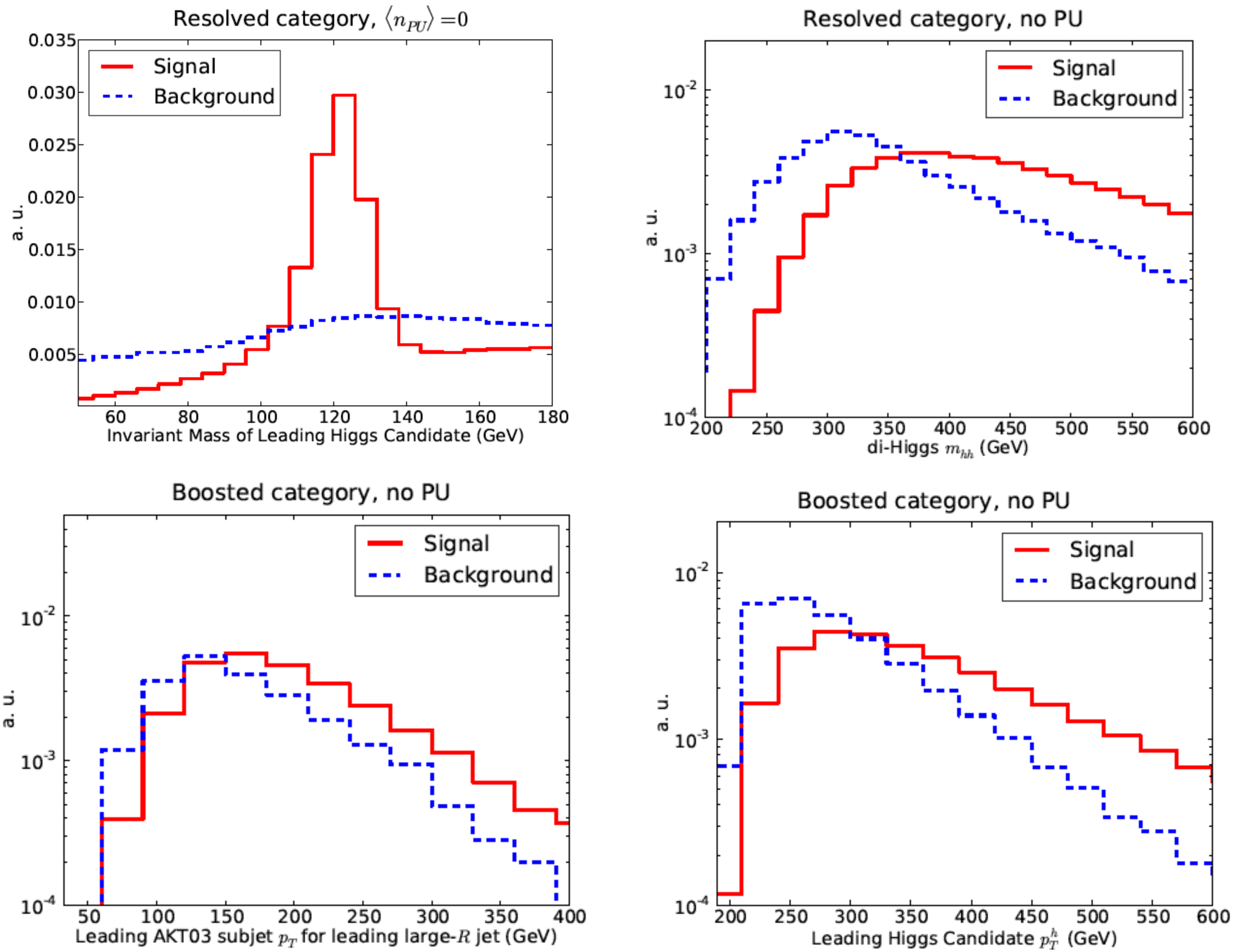
Boosted



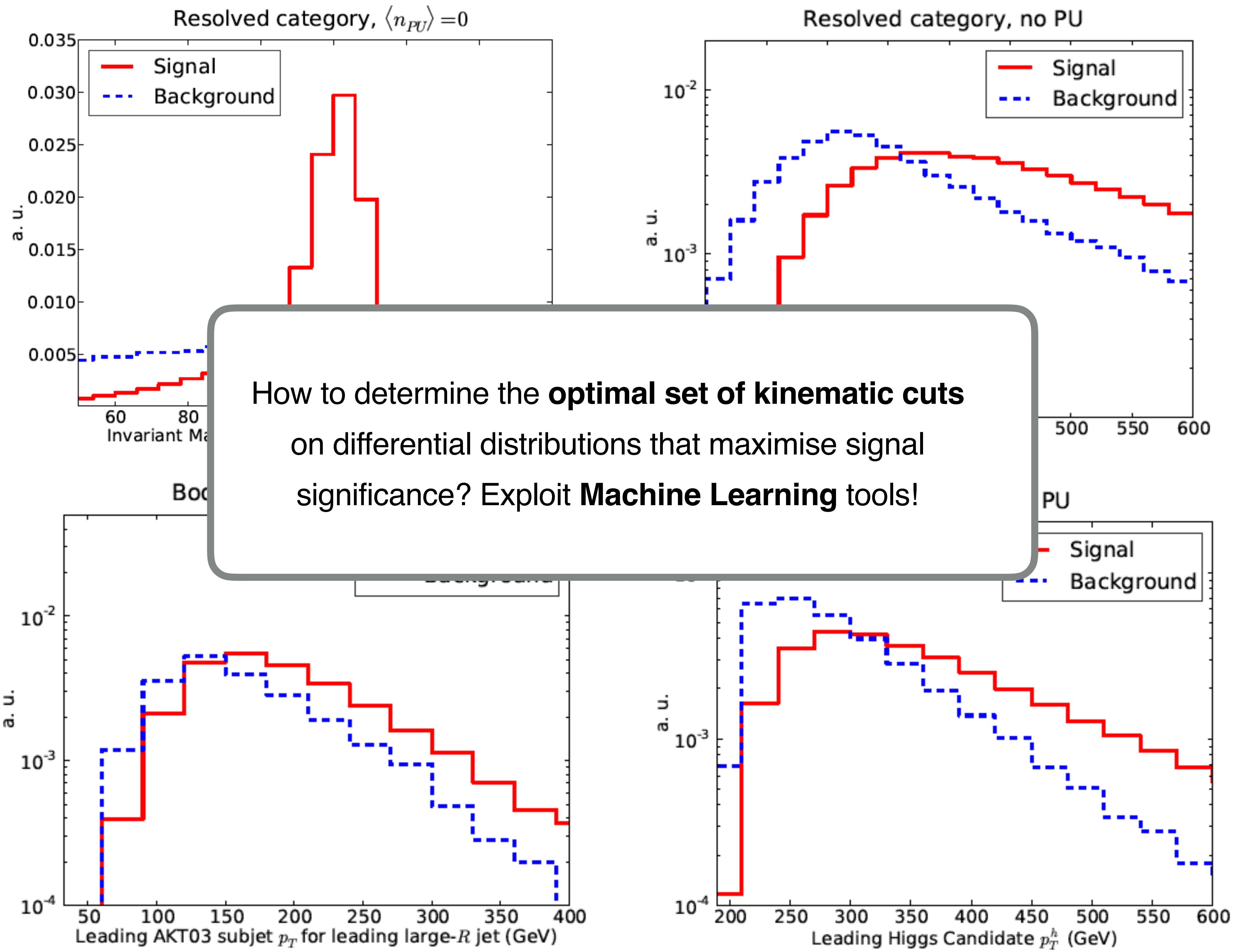
- ≥ 2 large-R jets (Higgs-tagged + b-tagged)
- Leading two jets taken as Higgs candidates

+ **Loose Higgs mass window cut**: $|m_{h,j} - 125 \text{ GeV}| < 40 \text{ GeV}$, $j = 1, 2$
+ **Rank categories** by $S/\sqrt{B}$ to make them **exclusive**: boosted > intermediate > resolved

Results of cut-based analysis



Results of cut-based analysis



How to determine the **optimal set of kinematic cuts** on differential distributions that maximise signal significance? Exploit **Machine Learning** tools!

Neural Network Discriminator

Input

Higgs pT

Higgs m

di-Higgs m

ECF

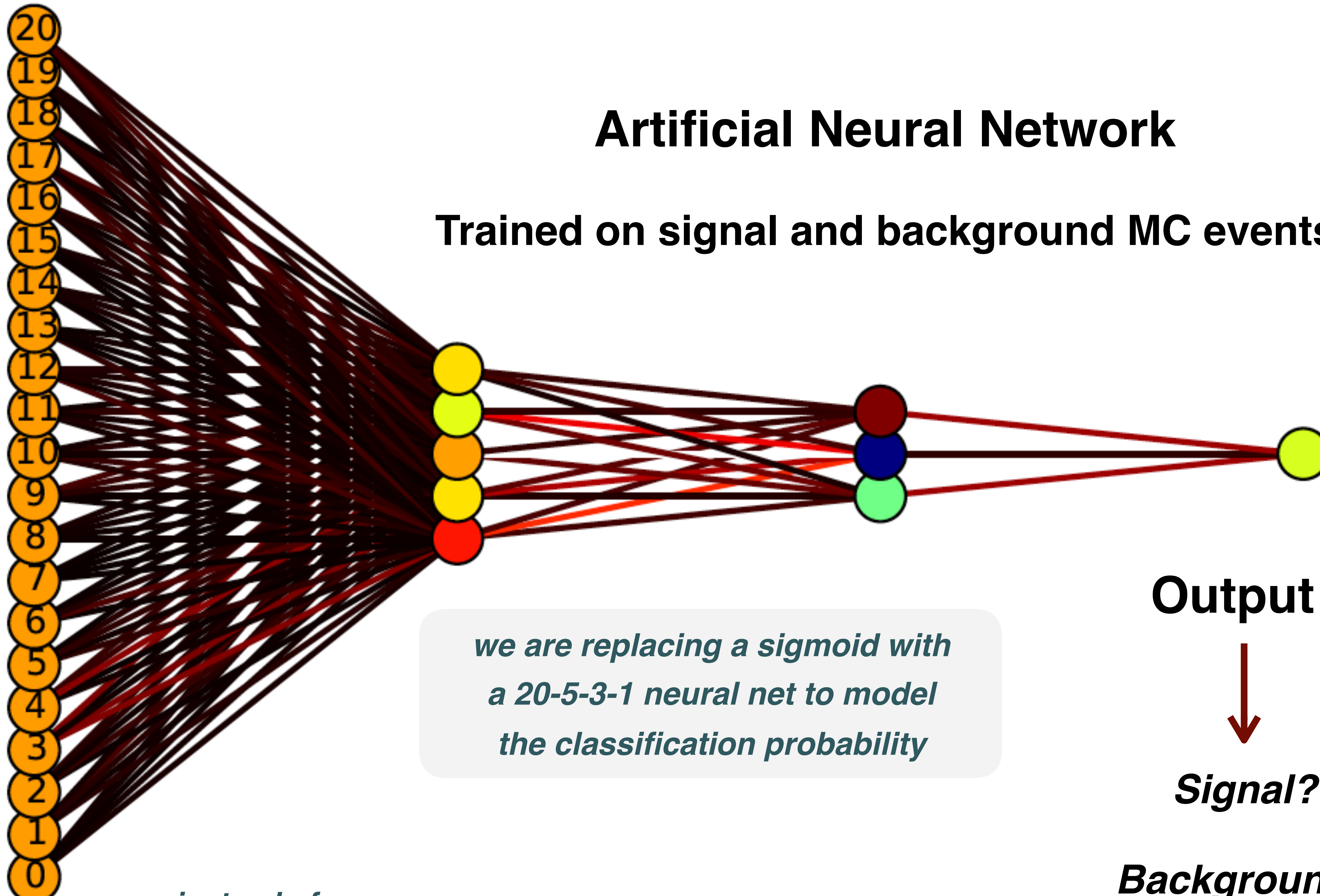
τ_{12}

Subjet pT

.....

Artificial Neural Network

Trained on signal and background MC events



we are replacing a sigmoid with a 20-5-3-1 neural net to model the classification probability

Output

Signal?

Background?

instead of

$$P(y_i = 1 \,|\, \boldsymbol{x}_i, \boldsymbol{\theta}) = \frac{1}{1 + e^{-\boldsymbol{x}_i^T \boldsymbol{\theta}}}$$

now we use

$$P(y_i = 1 \,|\, \boldsymbol{x}_i, \boldsymbol{\theta}) = \mathrm{NN}(\boldsymbol{x}_i; \boldsymbol{\theta})$$

Neural Network Discriminator

Input

Higgs pT

Higgs m

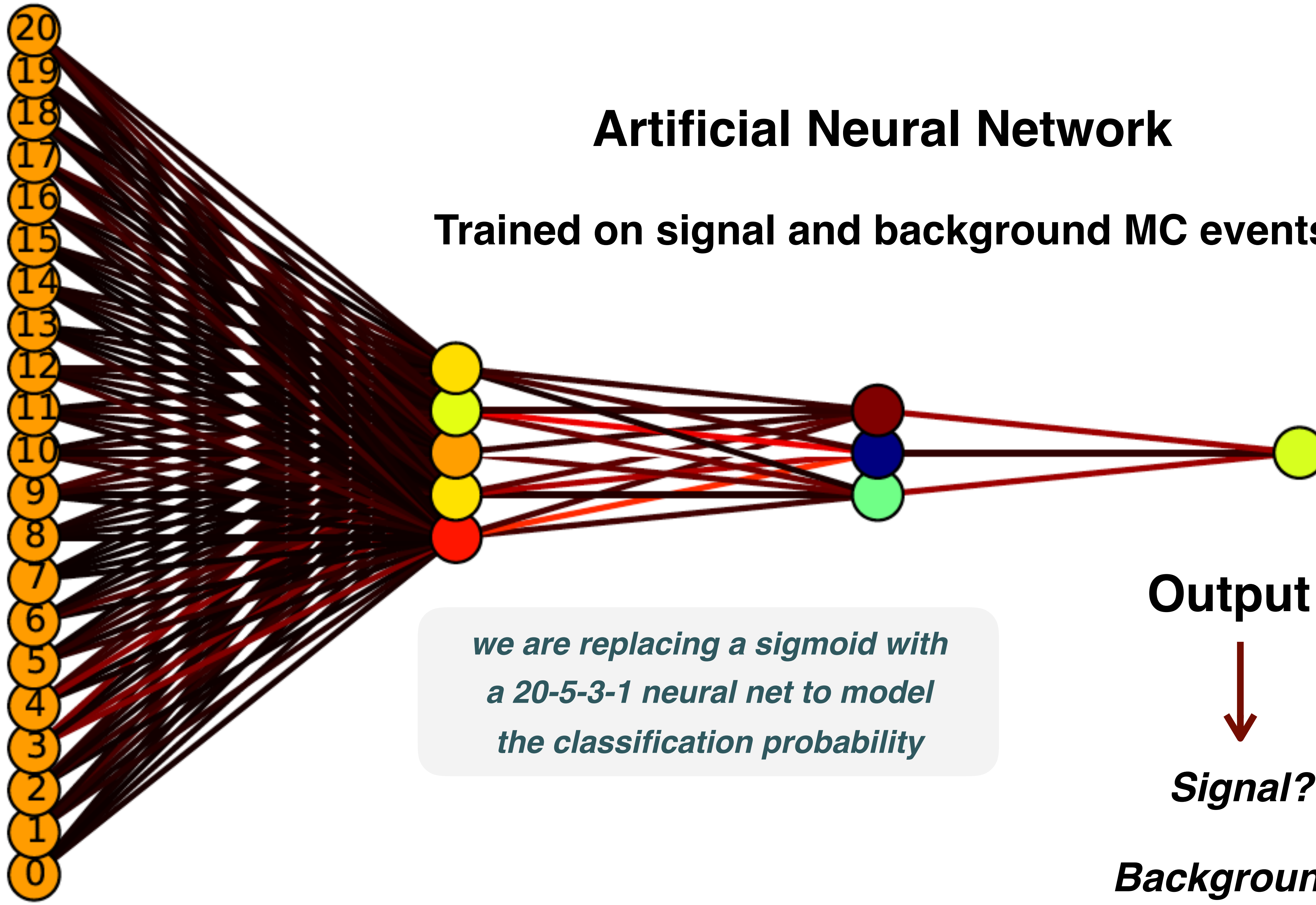
di-Higgs m

ECF

τ_{12}

Subjet pT

.....



Artificial Neural Network

Trained on signal and background MC events

we are replacing a sigmoid with a 20-5-3-1 neural net to model the classification probability

Output

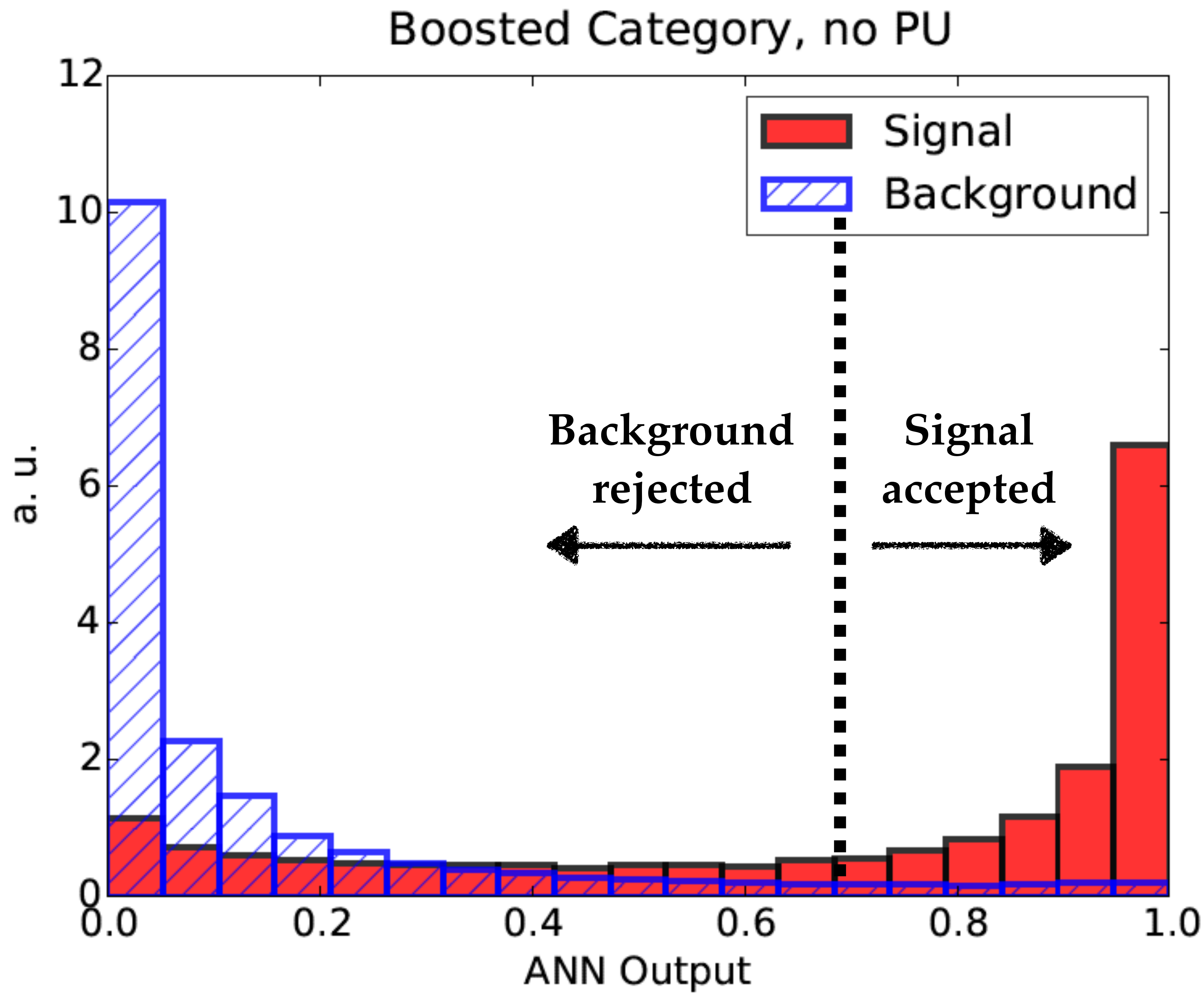
Signal?

Background?

$$\hat{\boldsymbol{\theta}} = \arg\ \min_{\theta} \left\{ \sum_{i=1}^{n} \left(-y_i \log \mathrm{NN}(\boldsymbol{x}_i; \boldsymbol{\theta}) \right) - (1 - y_i) \log \left(1 - \mathrm{NN}(\boldsymbol{x}_i; \boldsymbol{\theta}) \right) \right\}$$

ML discriminator

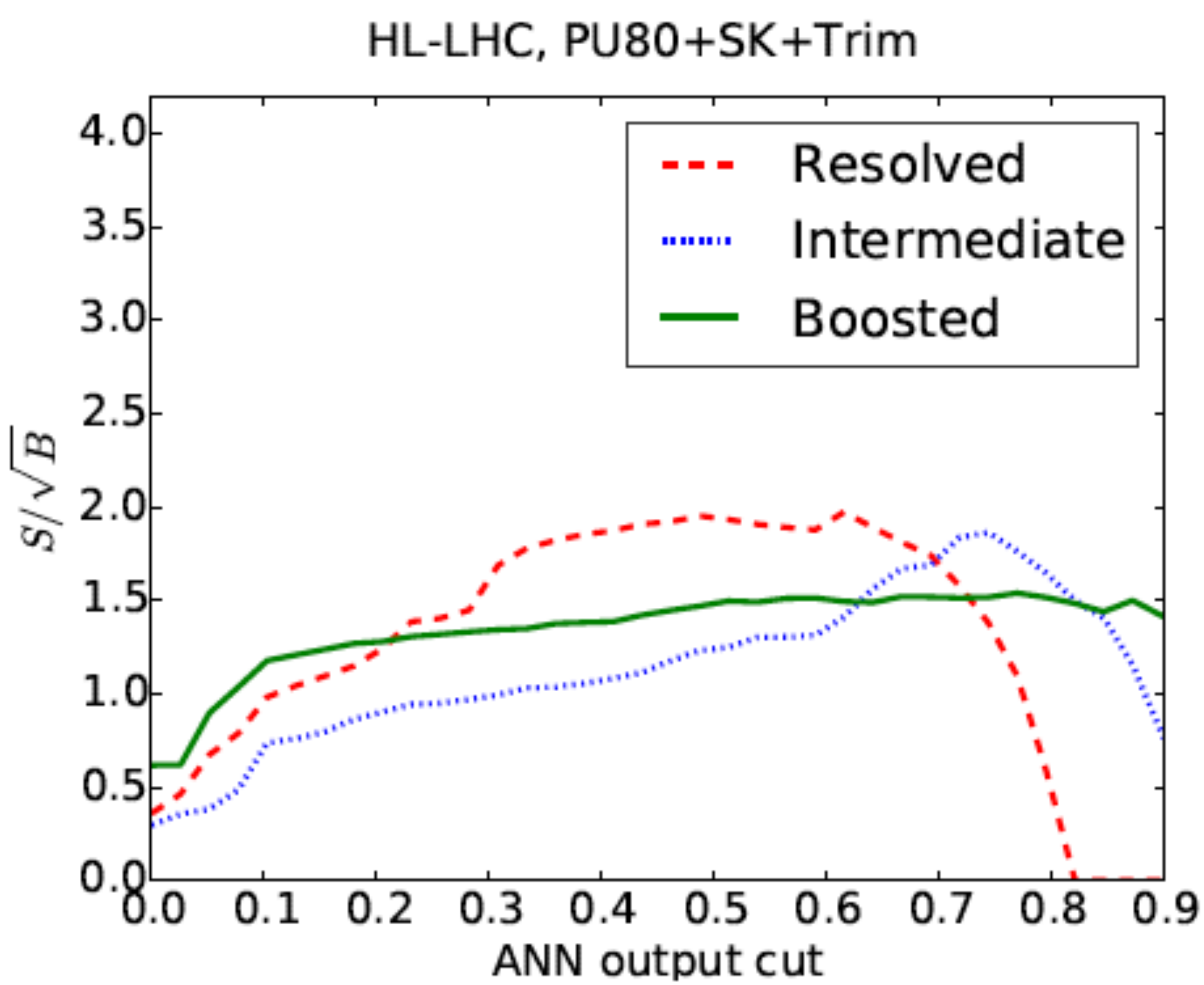
Optimal signal/background discrimination from ML-driven combination of kinematical info



Results

The total combined significance is enough to **observe Higgs pair production in the 4b final state** at the HL-LHC. Substantial improvement if reducible backgrounds (fakes) can be eliminated

$$\left(\frac{S}{\sqrt{B}}\right)_{\rm tot} \simeq 3.1\ (1.0) \qquad \left(\frac{S}{\sqrt{B_{4b}}}\right)_{\rm tot} \simeq 4.7\ (1.5)\,, \quad \mathcal{L} = 3000\ (300)\ {\rm fb}^{-1}$$



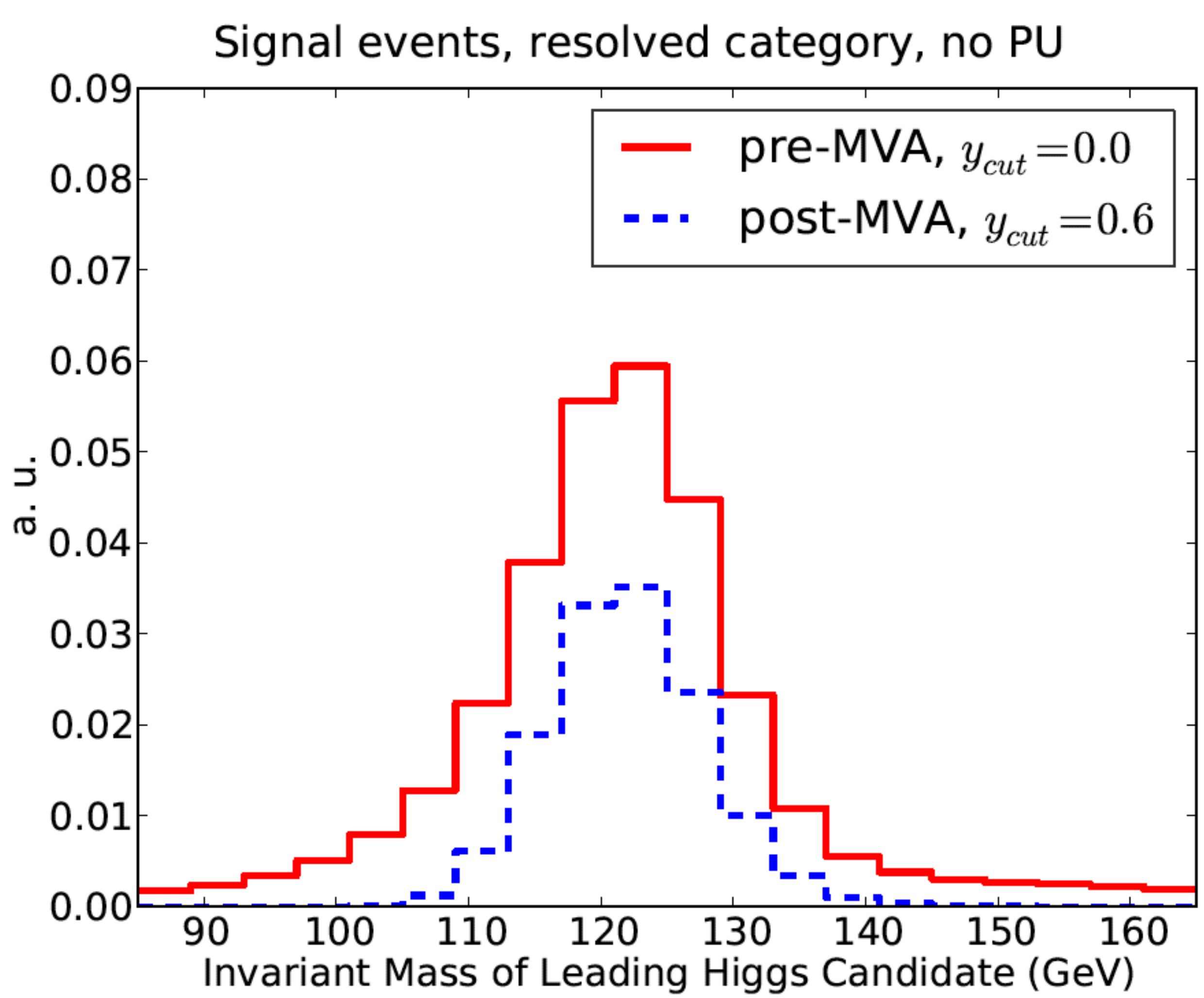
Pre-ML result

Opening the black box

- ML tools often criticised as **black boxes**, with little understanding of inner working
- ANNs are simply a **set of combined kinematical cuts**, nothing mysterious in them!
- Plot kin distributions after and before the ANN cut to determine the **effective kinematic cuts**
- This info enough to **perform a cut-based analysis** with similar signal significance

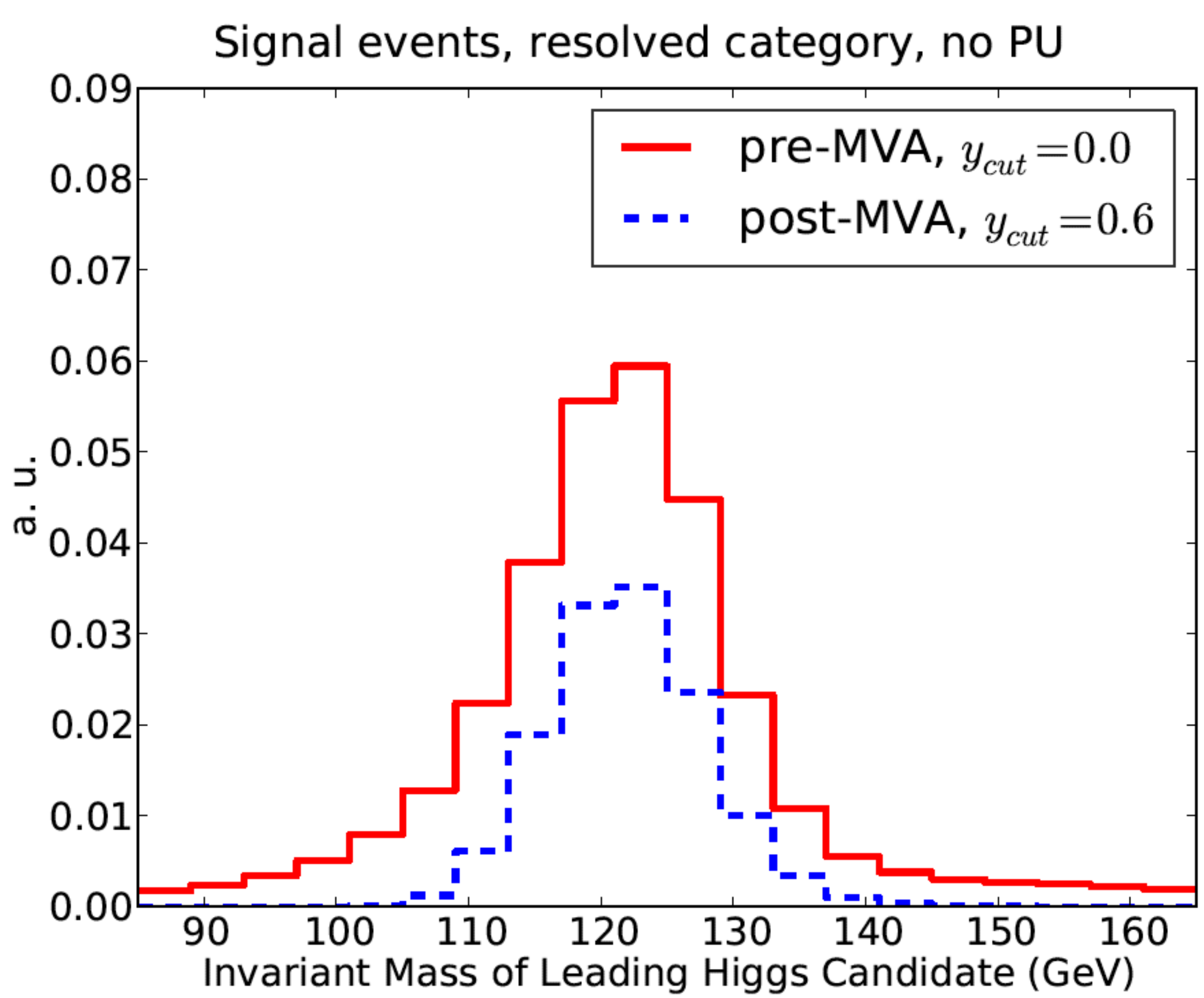
Opening the black box

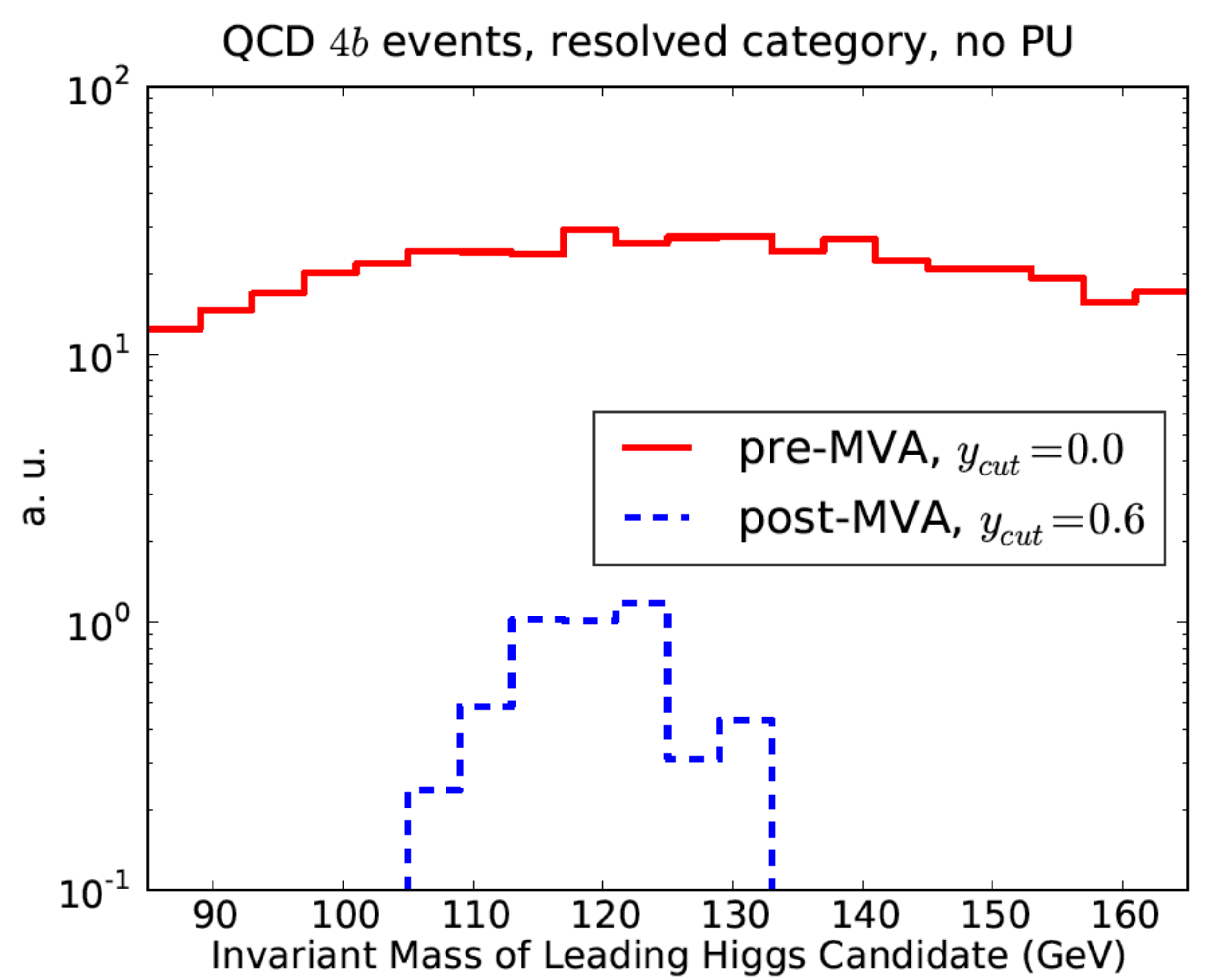
- ML tools often criticised as **black boxes**, with little understanding of inner working
- ANNs are simply a **set of combined kinematical cuts**, nothing mysterious in them!
- Plot kin distributions after and before the ANN cut to determine the **effective kinematic cuts**
- This info enough to **perform a cut-based analysis** with similar signal significance



Opening the black box

- ML tools often criticised as **black boxes**, with little understanding of inner working
- ANNs are simply a **set of combined kinematical cuts**, nothing mysterious in them!
- Plot kin distributions after and before the ANN cut to determine the **effective kinematic cuts**
- This info enough to **perform a cut-based analysis** with similar signal significance





*Higgs-like peak **sculpted** in QCD background*

Tutorial 4: Exercise 2b

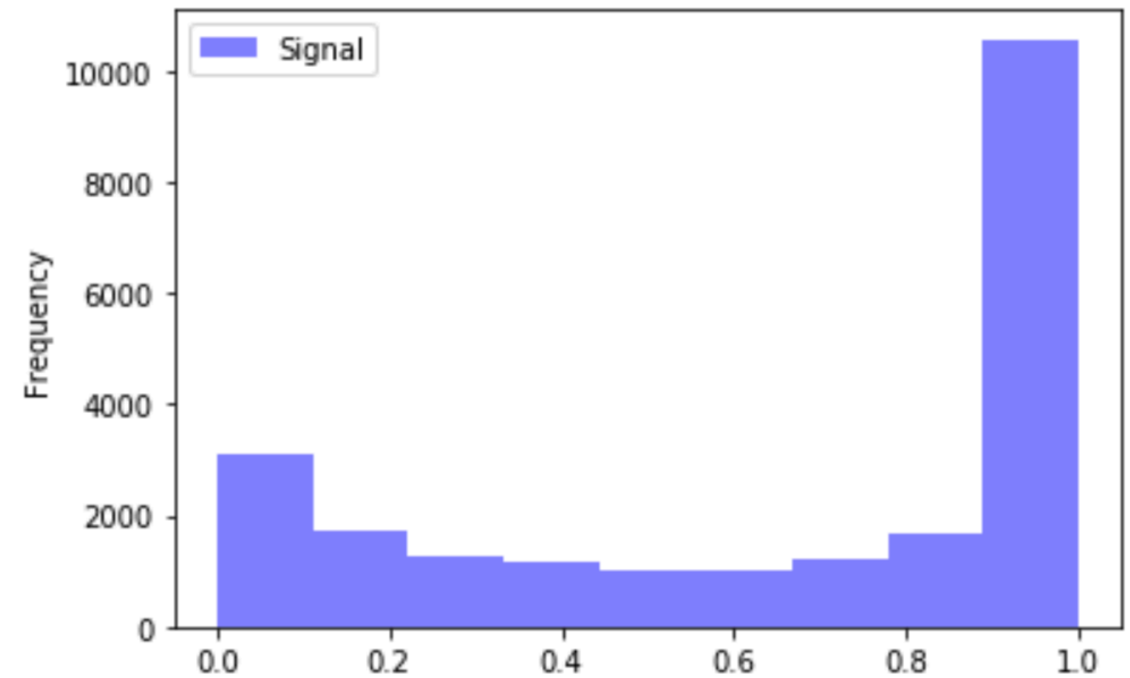
starting point is the **Python script** that you will find in

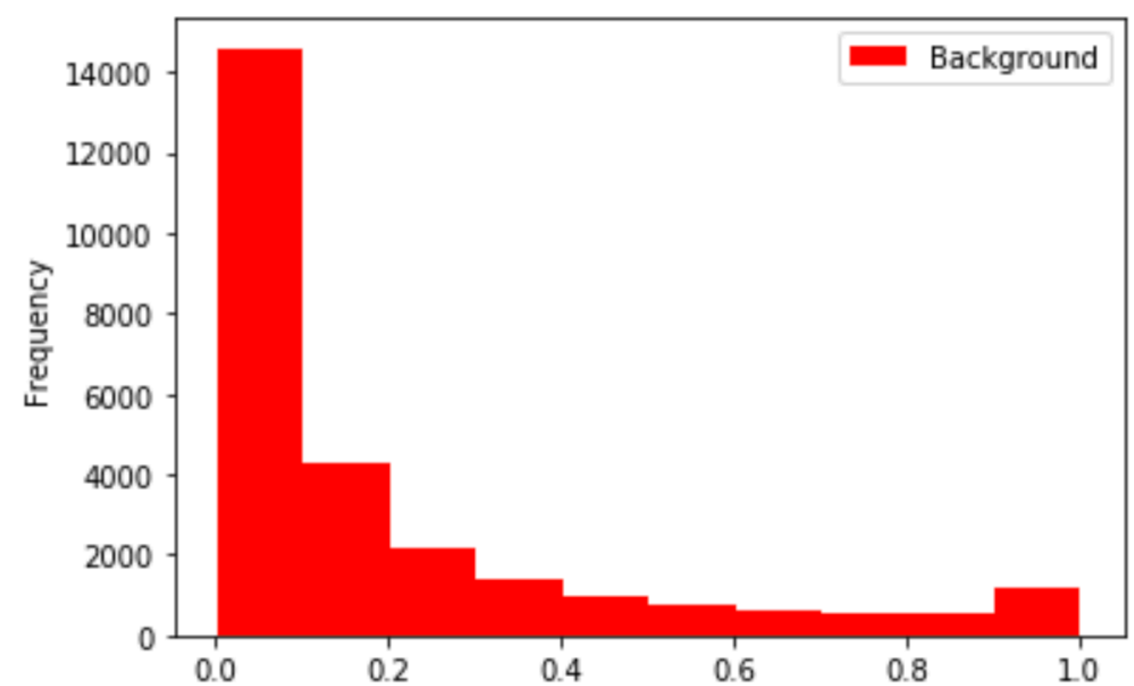
https://github.com/juanrojochacon/ml-ditp-attp/blob/master/Tutorials/Tutorial2/

- Train a **classifier** to classify signal events, **supersymmetric particle production**, with respect to background events at the LHC

- Determine the settings that lead to the **best discrimination** quantified by the largest ratio of TPR to NPR

- How does the performance of the classifier depend as we vary the size of the training dataset? Or on the number of input features used?





Energy-Based Models and Boltzmann Learning

Generative Models

Most ML models discussed here (Supervised NNs, logistic regression, ensemble models) are **discriminative:** designed to identify **differences between groups of data**

e.g. cats vs dogs discrimination

these models cannot carry some tasks such as **drawing new examples** from an unknown probability distribution: for this we need **generative models**

e.g. learn how to draw new examples of cat and dog images

e.g. generate new samples of a given phase of the Ising model

generative models are Machine Learning techniques that allows to learn **how to generate new examples** similar to those found in a training dataset

Here we consider **energy-based generative models**: close connection with statistical physics

hence the term Boltzmann Learning

Maximum Entropy Generative Models

basic concept is the **Shannon information-theoretic entropy**, which quantifies the statistical uncertainty one has about a random variable drawn from a probability distribution

$$S_p = \mathrm{Tr}_{\boldsymbol{x}} p(\boldsymbol{x}) \log \boldsymbol{x}$$

sum/integral over all possible values of variable

assume we have a **set of models**, functions of ***x***, whose average should coincide with some observed values. What should be their **underlying prob dist**?

$$\{f_i(\boldsymbol{x})\} \qquad \langle f_i \rangle_{\mathrm{obs}}$$

models ***observations***

Principle of Maximum Entropy: choose the probability distribution with the largest uncertainty (Shannon entropy) subject to the observational constraints

$$\langle f_i \rangle_{\mathrm{model}} = \int d\mathbf{x} f_i(\mathbf{x}) p(\mathbf{x}) = \langle f_i \rangle_{\mathrm{obs}}$$

the selected distribution is the one that makes admits the most ignorance beyond the stated data

Maximum Entropy Generative Models

this condition can be expressed as a **Lagrange Multiplier problem** by minimising:

$$\mathscr{L}[p] = -S_p + \sum_i \lambda_i \left(\langle f_i \rangle_{\rm obs} - \int d\boldsymbol{x}\, f_i(\boldsymbol{x}) p(\boldsymbol{x}) \right) + \gamma \left(1 - \int d\boldsymbol{x} p(\boldsymbol{x}) \right)$$

Shannon entropy *observational constraints* *normalisation*

whose solution gives us the **Maximum Entropy distribution**

$$p(\boldsymbol{x}) = \frac{\exp\left(\sum_i \lambda_i f_i(\boldsymbol{x}) \right)}{\int d\boldsymbol{x}\ \exp\left(\sum_i \lambda_i f_i(\boldsymbol{x}) \right)} = \frac{\exp\left(\sum_i \lambda_i f_i(\boldsymbol{x}) \right)}{Z}$$

partition function

$$E = -\sum_i \lambda_i f_i(\boldsymbol{x})$$

which is nothing but the **Boltzmann distribution in statistical mechanics,** and where the parameters of the distribution are fixed by the observations

$$\partial_{\lambda_i} \log Z = \langle f_i \rangle_{\rm data}$$

Energy-based Generative Models

these MaxEnt models can be used to **infer the underlying probability distributions** from a finite set of observations, which subsequently can be used to generate new instances

training an energy-based generative model: using the data to infer the model parameters

$$E\left(\boldsymbol{x};\boldsymbol{\theta}\right) = -\sum_i \theta_i f_i(\boldsymbol{x})$$

as in Supervised Learning, we need to specify a **cost function,** which however in the case of generative models is much subtler: what defines a good model?

the most useful method is to **maximise the log-likelihood** of the training set

$$\widehat{\boldsymbol{\theta}} = \arg\ \min_{\theta} \mathscr{L}(\{\boldsymbol{\theta}\})$$

$$\mathscr{L}(\{\boldsymbol{\theta}\}) = \left\langle \log p_\theta(\boldsymbol{x}) \right\rangle_{\rm data} = -\langle E(\boldsymbol{x};\boldsymbol{\theta})\rangle_{\rm data} - \log Z(\{\boldsymbol{\theta}\})$$

where we have used that the **generative probability distribution** is of the Boltzmann form and that the partition function does not depend on the data

Boltzmann machines

The training of **energy-based generative models** proceeds usually via SGD

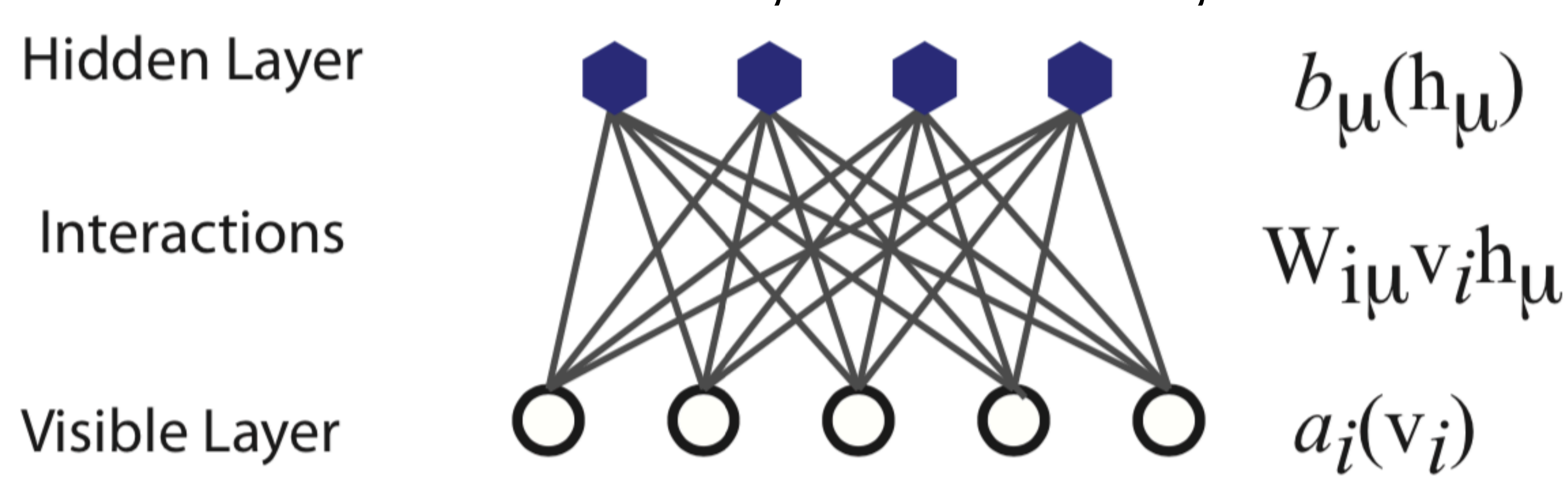
MaxEnt generative models are defined by the choice of the **energy**

$$p(\boldsymbol{x}) = \frac{\exp\left(\sum_i \lambda_i f_i(\boldsymbol{x})\right)}{\int d\boldsymbol{x}\, \exp\left(\sum_i \lambda_i f_i(\boldsymbol{x})\right)} = \frac{\exp\left(\sum_i \lambda_i f_i(\boldsymbol{x})\right)}{Z} = \frac{\exp\left(-E(\boldsymbol{x};\boldsymbol{\lambda})\right)}{Z}$$

one can construct various **other generative models** with different choices of the energy

e.g. Restricted Boltzmann machines

$$E(\boldsymbol{x};\boldsymbol{\theta}) = -\sum_i a_i(v_i) - \sum_\mu b_\mu(h_\mu) - \sum_{i\mu} W_{i\mu} v_i h_\mu$$



Generative Models & Adversarial Learning

The Kullback-Leibler divergence

The KL divergence, a measure of the similarity between two probability distributions *p(**x**)* and *q(**x**)* plays an important role in machine learning applications

$$D_{KL}\left(p\,||\,q\right) = \int d\boldsymbol{x}\, p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} \qquad D_{KL}\left(q\,||\,p\right) = \int d\boldsymbol{x}\, q(\boldsymbol{x}) \log \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}$$

which can be symmetrised to construct a **squared metric** (distance)

$$D_{JS}\left(p\,||\,q\right) = \frac{1}{2}\left(D_{KL}\left(p \,\middle|\middle|\, \frac{p+q}{2}\right) + D_{KL}\left(q \,\middle|\middle|\, \frac{p+q}{2}\right)\right)$$

Jensen-Shannon divergence

The KL-divergence is **positive-definite**, and only vanishes when *p(**x**)*= q*(**x**)*

$$D_{KL}\left(p\,||\,q\right) \geq 0$$

in general the integral cannot be computed and one needs to sample the two probability distributions by means of a suitable binning

The Kullback-Leibler divergence

In **generative models** one deals with two probability distributions (data and model), which we would like to have as similar as possible

$$p_\theta(\boldsymbol{x}) \qquad\qquad p_{\rm data}(\boldsymbol{x})$$

from model *from training data*

however subtleties about how we define **similarity** have large implications for the model training

maximising the **log-likelihood of the data under the model** is the same as **minimising the KL divergence** between the data distribution and the model distribution

$$D_{KL}\left(p_{\rm data}\,||\,p_\theta\right) = \int d\boldsymbol{x}\, p_{\rm data}(\boldsymbol{x}) \log \frac{p_{\rm data}(\boldsymbol{x})}{p_\theta(\boldsymbol{x})}$$

$$= \int d\boldsymbol{x}\, p_{\rm data}(\boldsymbol{x}) \log p_{\rm data}(\boldsymbol{x}) - \int d\boldsymbol{x}\, p_{\rm data}(\boldsymbol{x}) \log p_\theta(\boldsymbol{x})$$

$$= S[p_{\rm data}] - \left\langle \log p_\theta \right\rangle_{\rm data}$$

The Kullback-Leibler divergence

maximising the **log-likelihood of the data under the model** is the same as **minimising the KL divergence** between the data distribution and the model distribution

$$\Big\langle \log p_\theta(\boldsymbol{x}) \Big\rangle_{\rm data} = S[p_{\rm data}] - D_{KL}\left(p_{\rm data} \,||\, p_\theta\right)$$

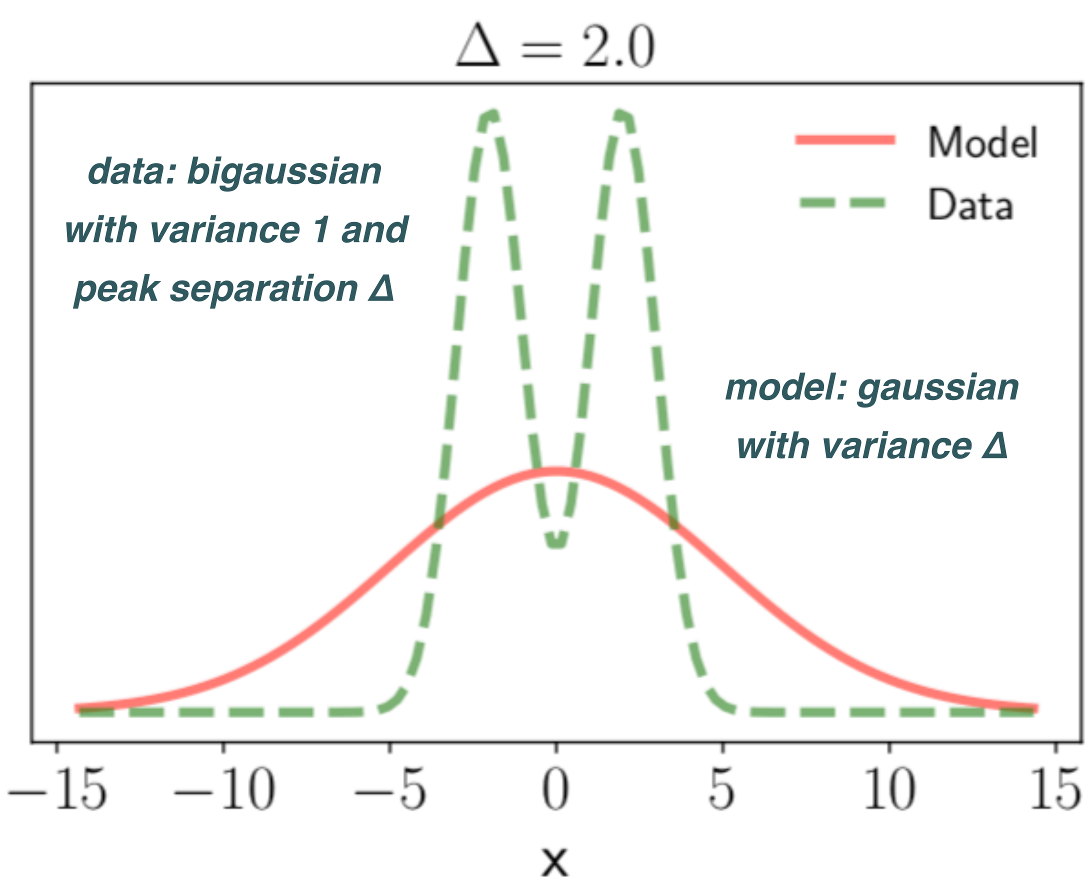
Log-likelihood of data under model

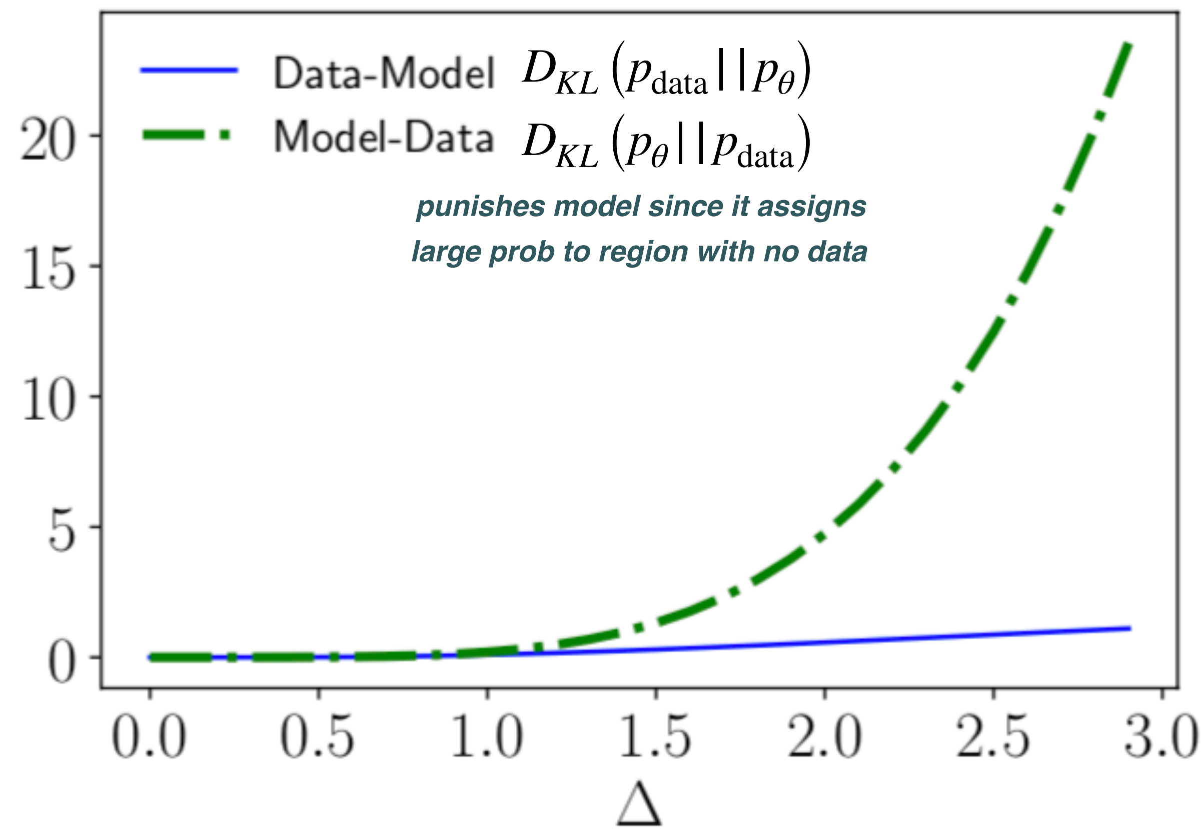
entropy of data: independent of model parameters

KL-divergence

Similarity

Similarity between probability distributions is a subtle concept





$$D_{KL}\left(p_{\rm data}||p_\theta\right) \longrightarrow$$ misses important information when comparing the data and theory probability distributions

Adversarial Learning

(draw values of θ accordingly to p_θ and compare with data)

(1) $D_{KL}\left(p_{\text{data}}||p_\theta\right)$ → ***Calculable using sampling***

(2) $D_{KL}\left(p_\theta||p_{\text{data}}\right)$ → ***Large when model over-weights low-density regions near real peaks***

→ ***but not calculable since p_{data} unknown ….***

In **Adversarial Learning** we achieve a similar goal as that of minimising ***(2)*** by training a **discriminator** to distinguish between real data points and samples from the model

By punishing the model for generating points that can be easily discriminated from the data, Adversarial Learning decreases the **weight of regions in the model space that are far away from data points,** regions that inevitably arise when maximising the likelihood

Generative adversarial networks

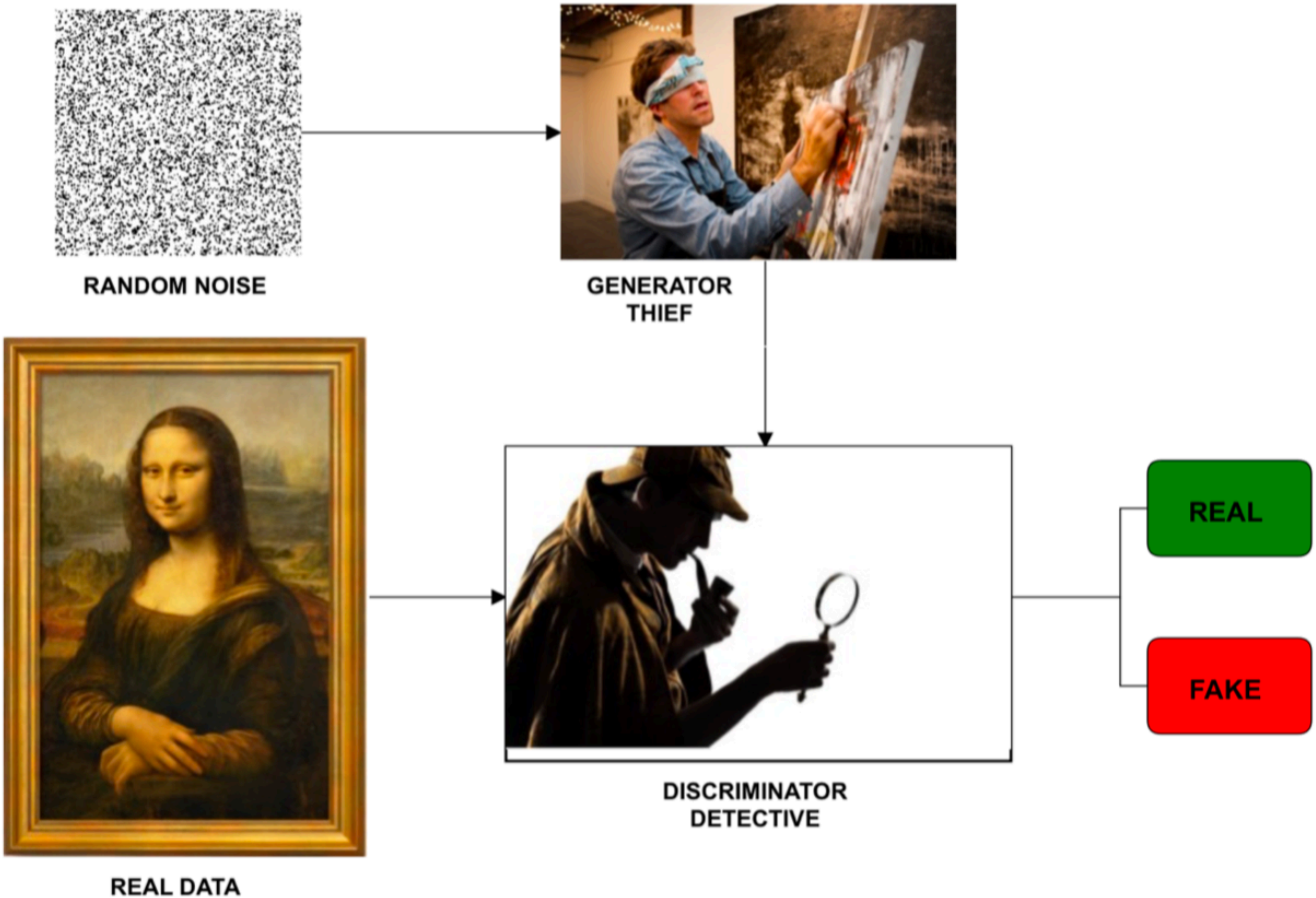
Generative Adversarial Networks (GANs) are deep neural network architectures, composed by two independent NNs which **compete against each other**

- ***(1)*** A **generator G** NN that creates (samples) pseudo-data by inferring the probability distribution associated to the training dataset

- ***(2)*** A **discriminator D** NN which determines the probability of a given sample arises from the actual training data rather than having been produced by **G**

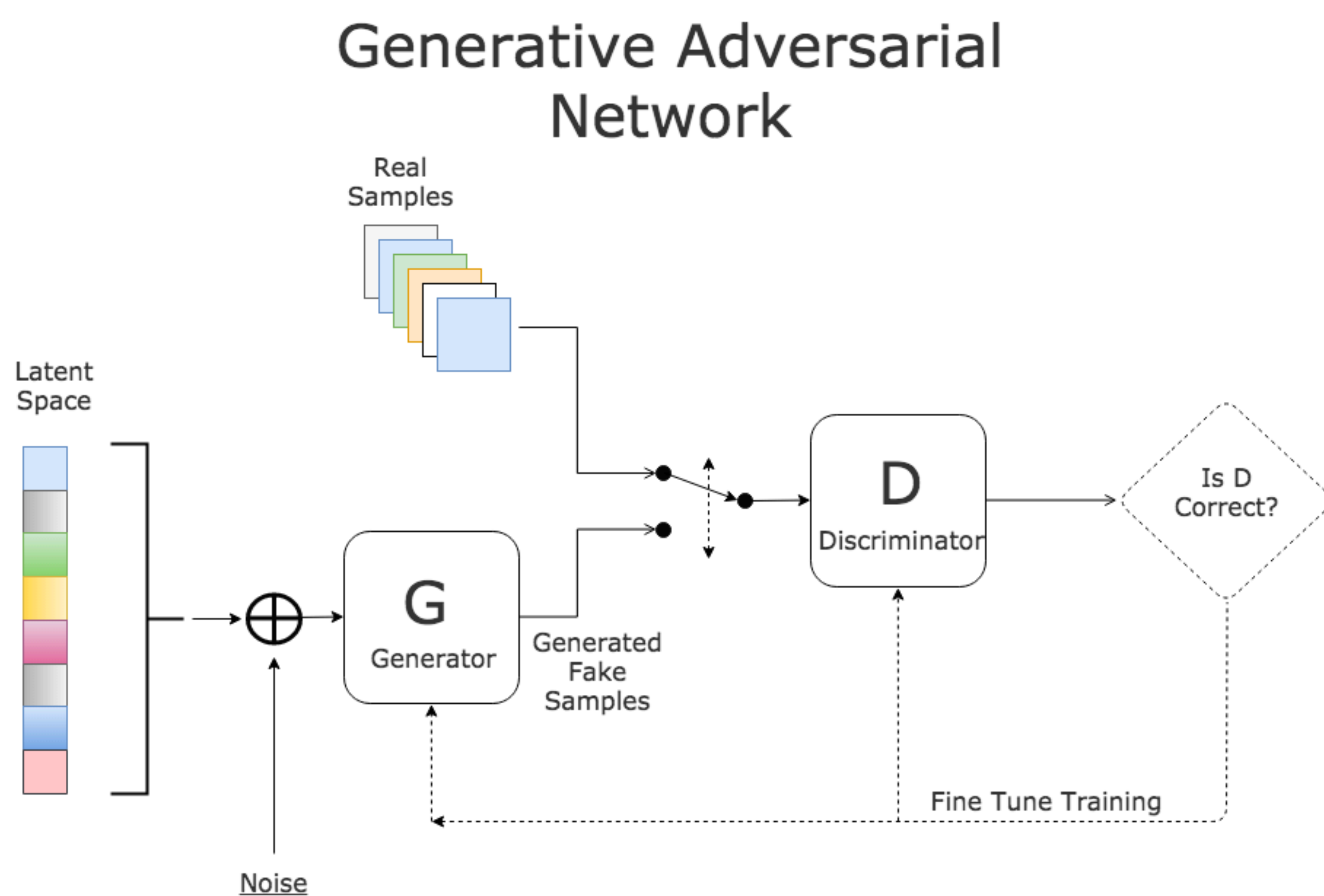
the generator network **G** should be trained to maximise the probability that the discriminator network **D** makes a mistake: that is, **G** should generate pseudo-data samples that are virtually **indistinguishable** from the actual data

Generative adversarial networks



Generative Adversarial Networks

- New architecture for an **unsupervised neural network training** (unlabelled samples)
- Based on two **independent nets** that work separately and act as **adversaries:**
  - the **Discriminator (D)** undergoes training and plays the role of classifier
  - the **Generator (G)** and is tasked to generate random samples that **resemble real samples** with a twist rendering them as fake samples.



Generative Adversarial Networks

GAN training

As with other NN architectures one uses GD to train GANs, but now one has to **update sequentially** the model parameters of both **G** and **D**

- Take a sample of *N* data points from the training set

$$\{\boldsymbol{x}_n\}_{n=1}^N \qquad \boldsymbol{x}_n = (x_{n,1}, x_{n,2}, \ldots, x_{n,p})$$

p = number of features per sample

- Produce a sample of *N* pseudo-data points from generator **G** (at ite$_0$ this is random noise)

$$\{\boldsymbol{z}_n\}_{n=1}^N \qquad \boldsymbol{z}_n = (z_{n,1}, z_{n,2}, \ldots, z_{n,p})$$

- Evaluate the cost function: since we are dealing with binary classification (true/false) the appropriate cost function is the **cross-entropy**

$$C\left(\boldsymbol{\theta}_D, \boldsymbol{\theta}_G\right) = \frac{1}{N}\sum_{n=1}^{N}\left(\log D(\boldsymbol{x}_i) + \log(1 - D(G(\boldsymbol{z}_i)))\right)$$

NN params of G

output of D when input a real data sample

output of D when input a ``fake'' data sample produced by G

NN params of D

- Train **D** using GD to maximise its discrimination capability

GAN training

- Evaluate the cost function: since we are dealing with binary classification (true/false) the appropriate cost function is the **cross-entropy**

$$C\left(\boldsymbol{\theta}_D, \boldsymbol{\theta}_G\right) = \frac{1}{N}\sum_{n=1}^{N}\left(\log D(\boldsymbol{x}_i) + \log(1 - D(G(\boldsymbol{z}_i)))\right)$$

- Train **D** using GD to maximise its discrimination capability

$$\mathbf{v}_t = \eta_t \nabla_{\theta_D} C(\boldsymbol{\theta}_{D,t}, \boldsymbol{\theta}_G)\,, \qquad \boldsymbol{\theta}_{D,t+1} = \boldsymbol{\theta}_{D,} - \mathbf{v}_t$$

- At this point **D** can tell apart data from pseudo-data pretty well, so we need to train **G** to generate better (closer to the training set) pseudo-data samples
- Produce a sample of *N* pseudo-data points from the generator **G** $\{z_n\}_{n=1}^{N}$

$$C\left(\boldsymbol{\theta}_D, \boldsymbol{\theta}_G\right) = \frac{1}{N}\sum_{n=1}^{N}\log(1 - D(G(z_i)))$$

output of D (now with its parameters fixed)

$$\mathbf{v}_t = \eta_t \nabla_{\theta_G} C(\boldsymbol{\theta}_{D,t}, \boldsymbol{\theta}_G)\,, \qquad \boldsymbol{\theta}_{G,t+1} = \boldsymbol{\theta}_{G,t} - \mathbf{v}_t$$

GAN training

the generator and discriminator are sequentially trained and iterated until convergence is achieved, at this point **D** cannot tell apart the pseudo-data from **G** from the real data

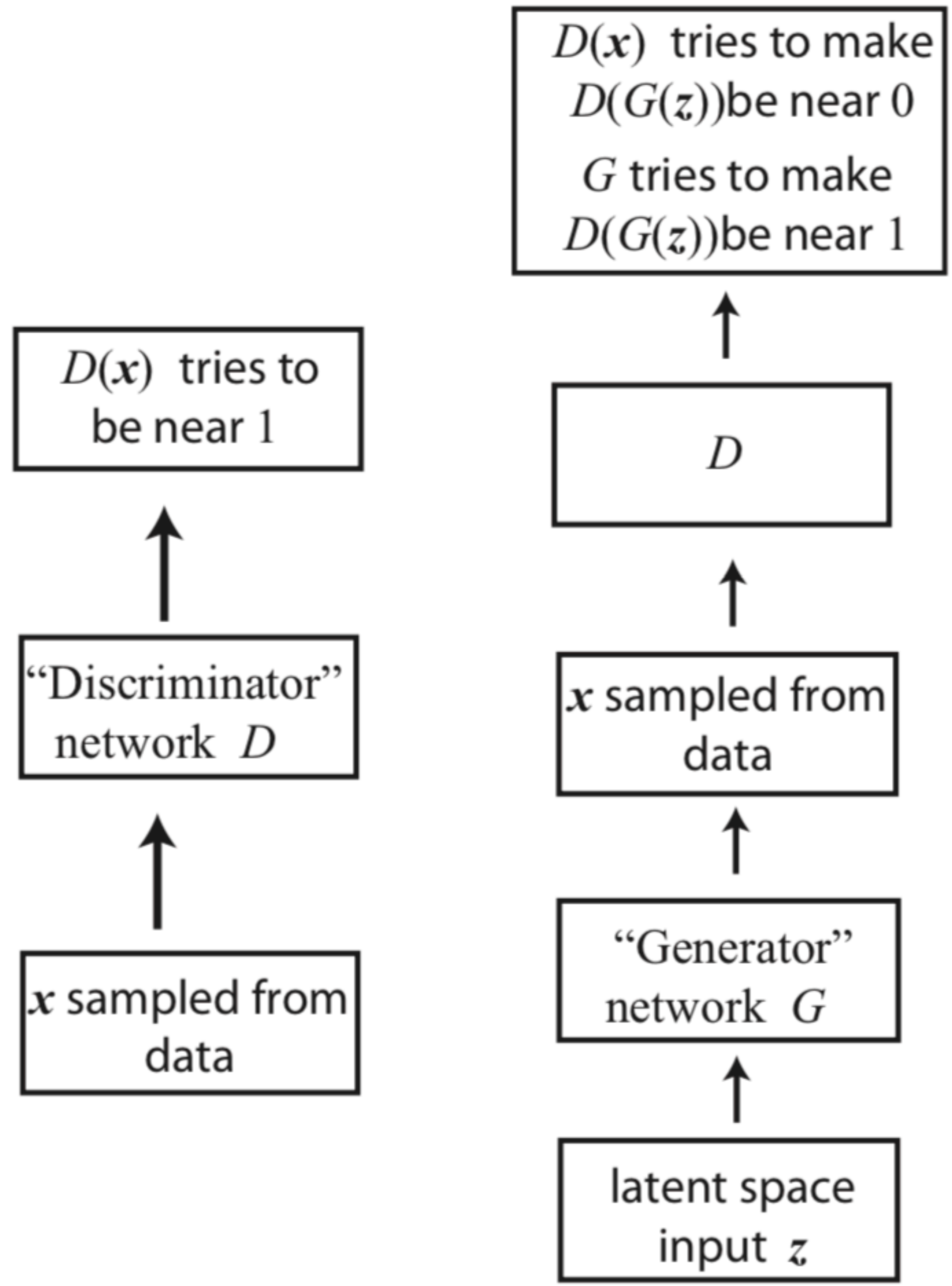


Image generation with GANs

https://thispersondoesnotexist.com/

Image generation with GANs

[***https://thispersondoesnotexist.com/***](https://thispersondoesnotexist.com/)

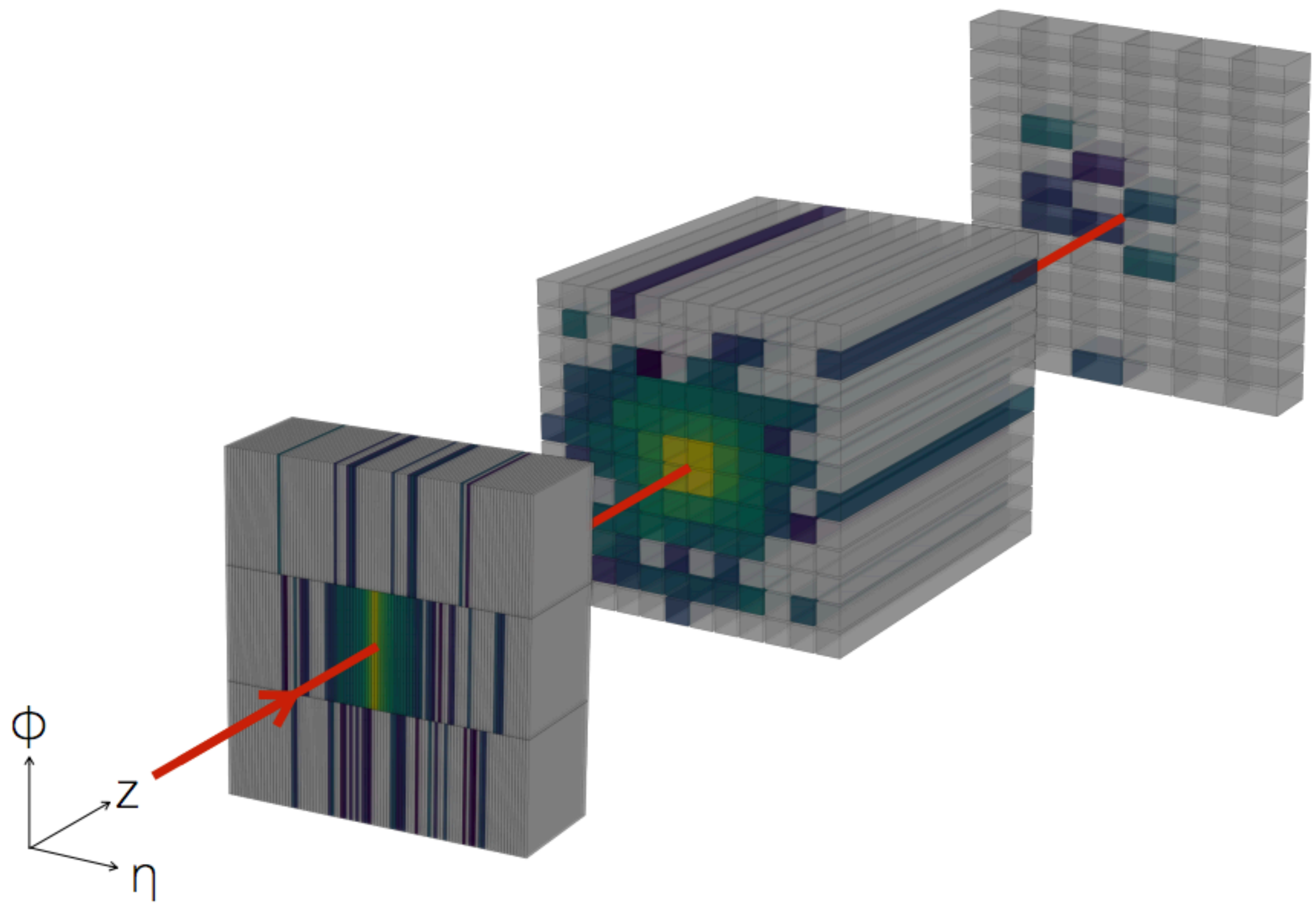
Generative Adversarial Networks

GANs for detector simulation in HEP

- Modelling accurately the response of detectors with the **propagation of high energy particles** is an essential task for present and future HEP experiment

- **Detector simulation at the LHC** is a very CPU-intensive task, dominated by modelling of particle showers inside calorimeters

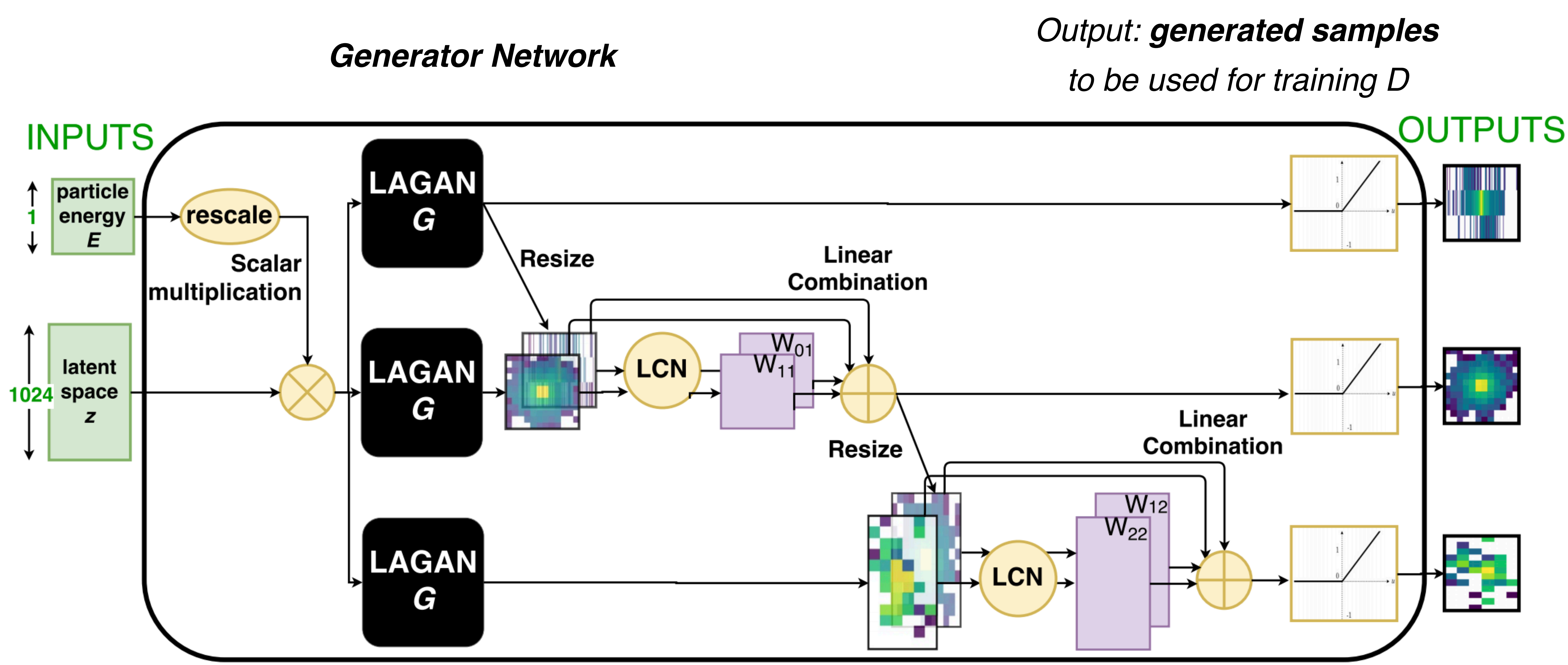
- **Generative Adversarial Networks** can speed up detector simulation by orders of magnitude



*Task: to efficiently model the **propagation of high energy particles** (and their interaction) within the layers of electromagnetic and hadronic calorimeters*

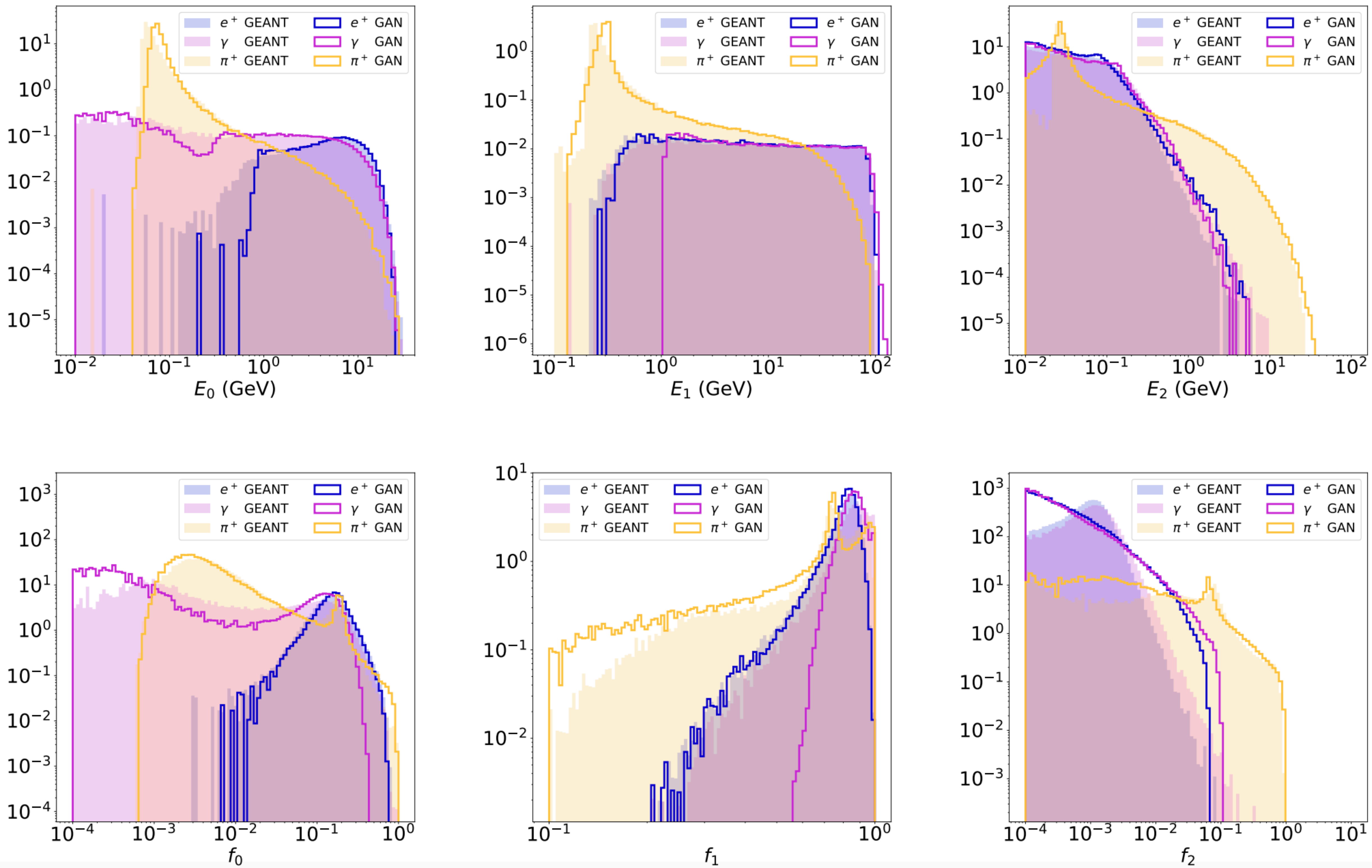
GANs for detector simulation in HEP

- Use GANs as a tool to **speed up full simulation of particle showers** in a HEP calorimeter
- The generator G learns a map from **a latent space** to space of **generated samples** for training
- Carefully understanding the **underlying physics of particle propagation** in a detector is crucial to optimise the training strategy, *e.g.* relationships between neighbouring detector layers



Paganini et al. 17

GANs for detector simulation in HEP



GANs for detector simulation in HEP

Simulator	Hardware	Batch Size	ms/shower
Geant4	CPU	N/A	1772
CaloGAN	CPU	1	13.1
		10	5.11
		128	2.19
		1024	2.03
	GPU	1	14.5
		4	3.68
		128	0.021
		512	0.014
		1024	0.012

Speed-up by several orders of magnitude, specially when running in GPUs

Convolutional Neural Networks

Convolutional Neural Networks

Like physical systems, many datasets and supervised learning tasks also possess additional **symmetries and structure** what can (and should) be exploited

e.g. we want to train a classifier to identify pictures of cats. What **high-level features** must one learn first?

Convolutional Neural Networks

Like physical systems, many datasets and supervised learning tasks also possess additional **symmetries and structure** what can (and should) be exploited

e.g. we want to train a classifier to identify pictures of cats. What **high-level features** must one learn first?

- *The features that define ``cat" are local in the picture: whiskers, tail, paws ….: **locality***

- *Cats can be anywhere in the image: **translational invariance***

- *Relative position of features must be respected (eg whiskers and tail shoaled appear in opposite sides of ``cat"): **rotational invariance***

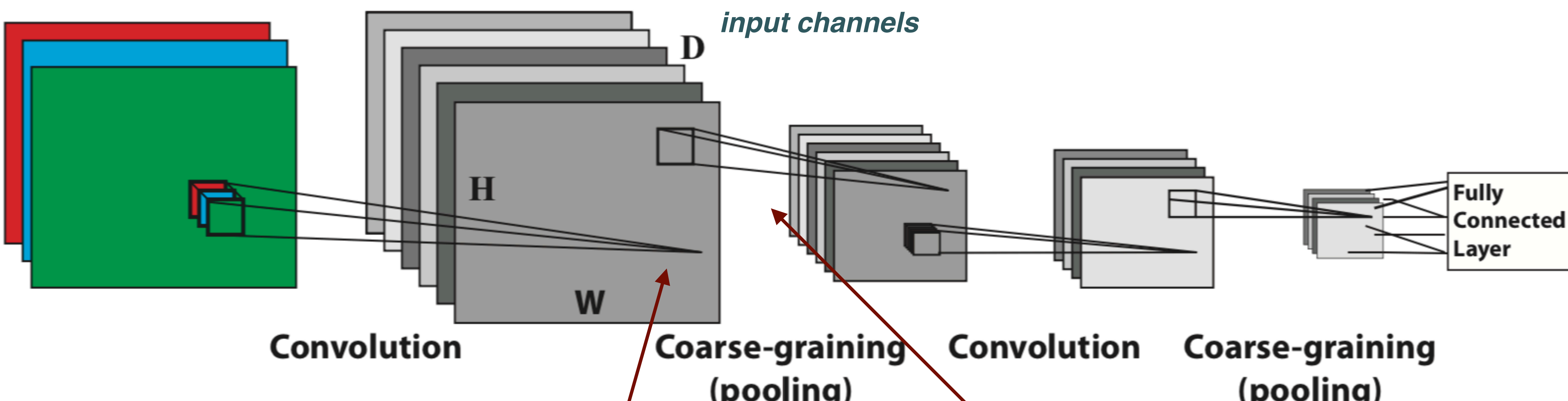
Our classifier should exhibit all these high-level features

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are architectures that take **advantage of this additional high-level structures** that all-to-all coupled networks fail to exploit

A CNN is a translationally invariant neural network that respects locality of the input data

Depth: number of input channels



D=3 for RGB images
Height (H) and Width (W) determined by number of pixels

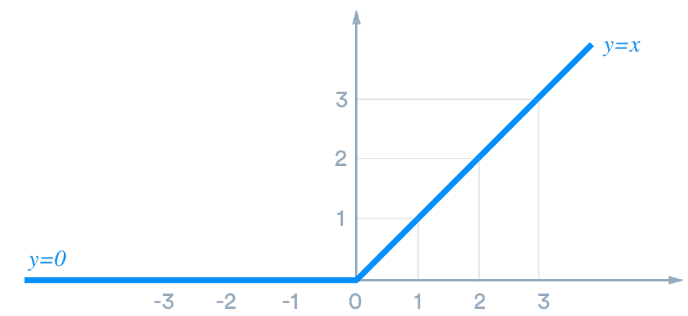
neuron activation state: convolution with local spatial filter (eg 3 x 3 pixel grid)

pooling layers reduce H, W while preserving D

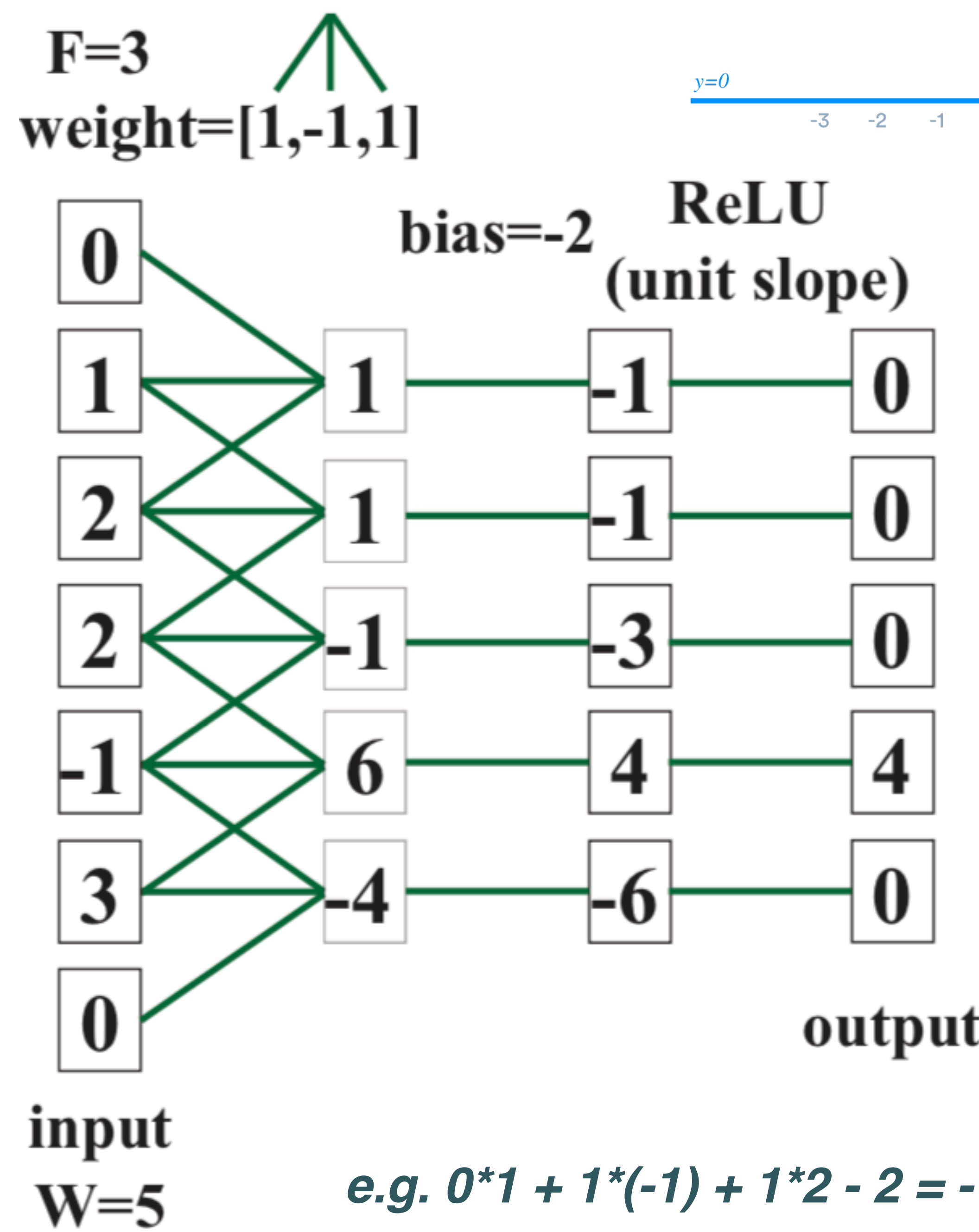
Convolutional Neural Networks

CNNs are composed by two kinds of layers

Convolution of input with **filters**



size-3 Filter



example of convolutional layer

note that convolution changes the depth, but not the height and width of the network

*e.g. 0*1 + 1*(-1) + 1*2 - 2 = -1*

Convolutional Neural Networks

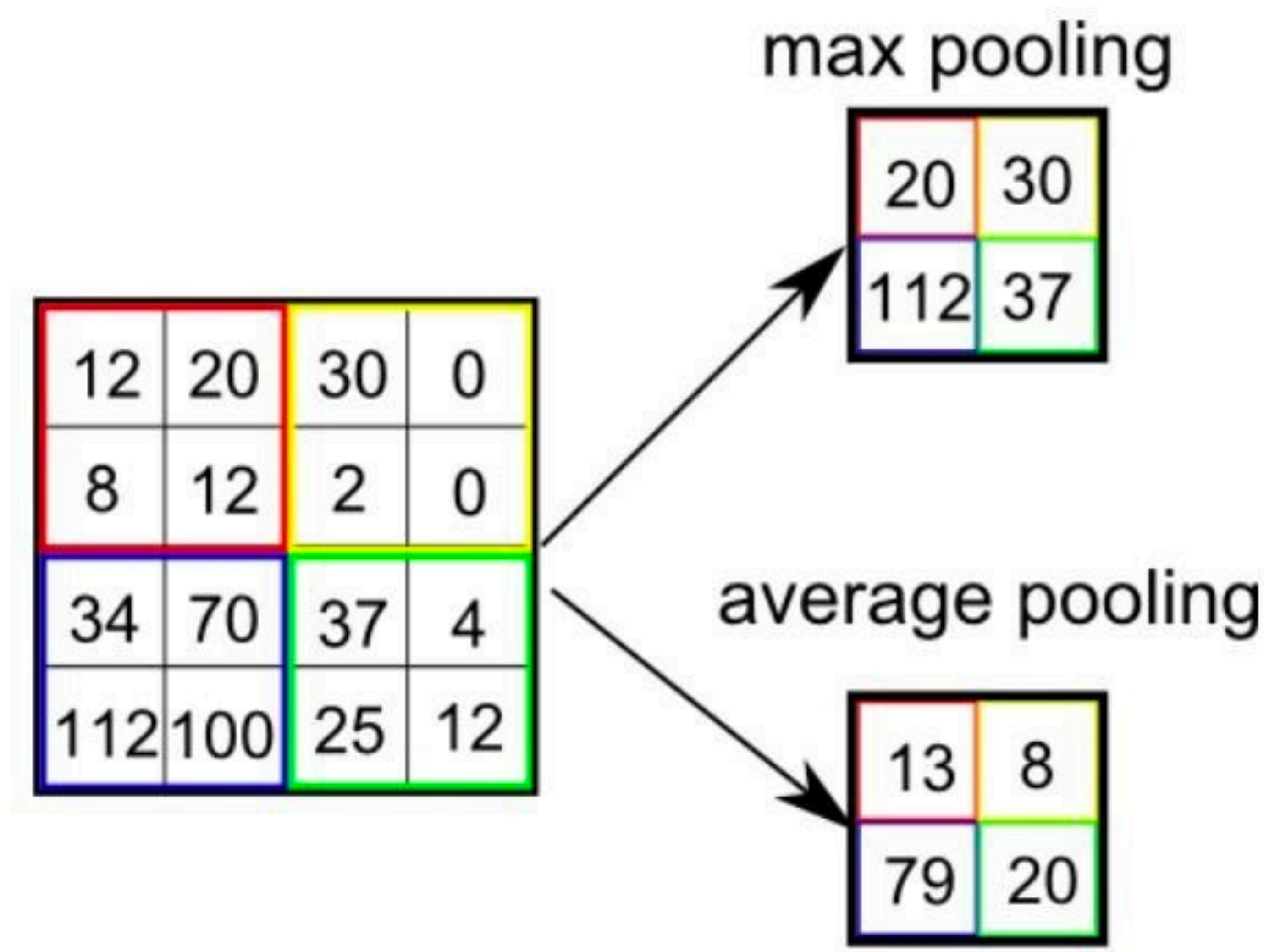
CNNs are composed by two kinds of layers

Convolution layer of input with **filters**

Pooling layer that coarse-grains the input while maintaining locality and spatial structure

e.g. **MaxPool**, the spatial dimensions are coarse-grained by replacing a small region by single neuron whose output is maximum value of the output in the region

in average pooling, one averages over output in region



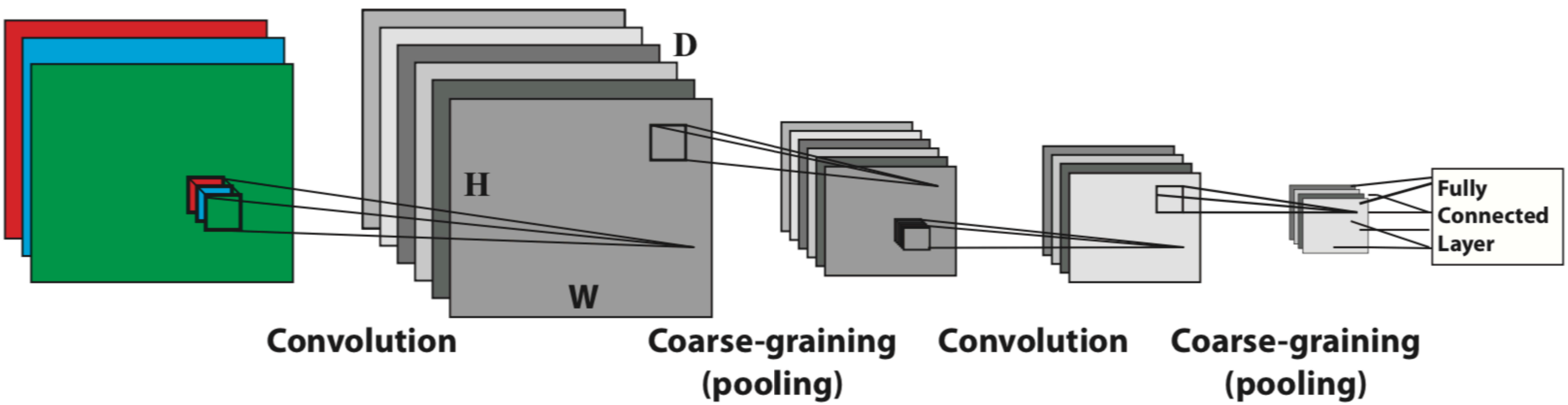
Convolutional Neural Networks

CNNs are composed by two kinds of layers

Convolution layer of input with **filters**

Pooling layer that coarse-grains the input while maintaining locality and spatial structure

the convolution and max-pool layers are followed by an **all-to-all connected layer and a high-level classifier**, so that one can train CNNs using the standard backpropagation algorithm



Convolutional Neural Networks

CNNs are composed by two kinds of layers

Convolution layer of input with **filters**

Pooling layer that coarse-grains the input while maintaining locality and spatial structure

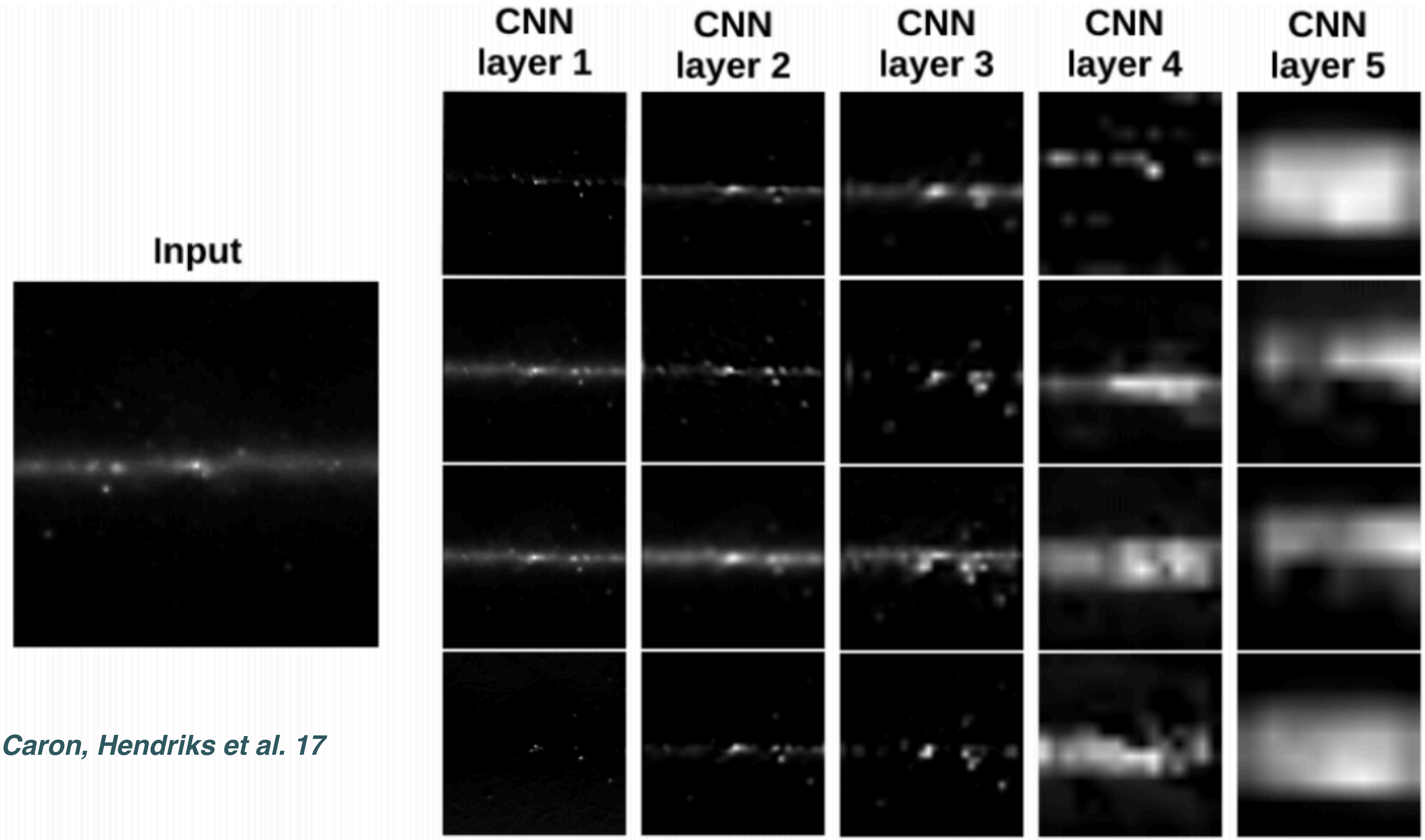
the convolution and max-pool layers are followed by an **all-to-all connected layer and a high-level classifier**, so that one can train CNNs using the standard backpropagation algorithm

note that only problems characterised by a **spatial locality** are amenable to CNNs: for example the 2D Ising dataset can be studied with CNNs, but not the SUSY dataset

How CNNs work

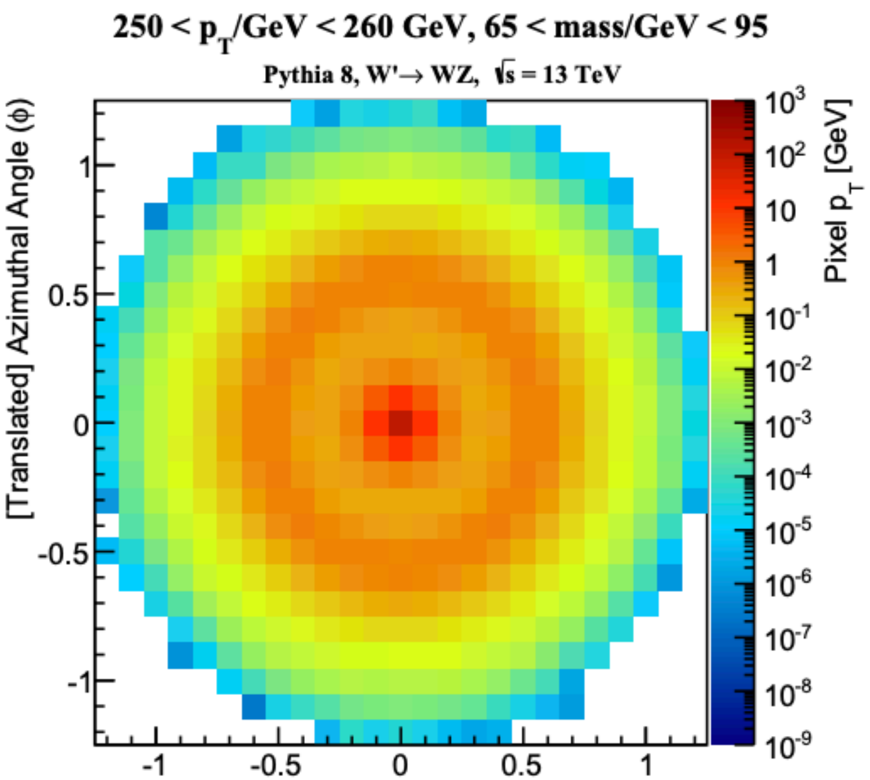
Application to Dark Matter searches

Use CNNs to discriminate between **point sources** (astrophysical origin) and **diffuse flux** (dark matter) in galactic centre images, to compare with predictions of DM models

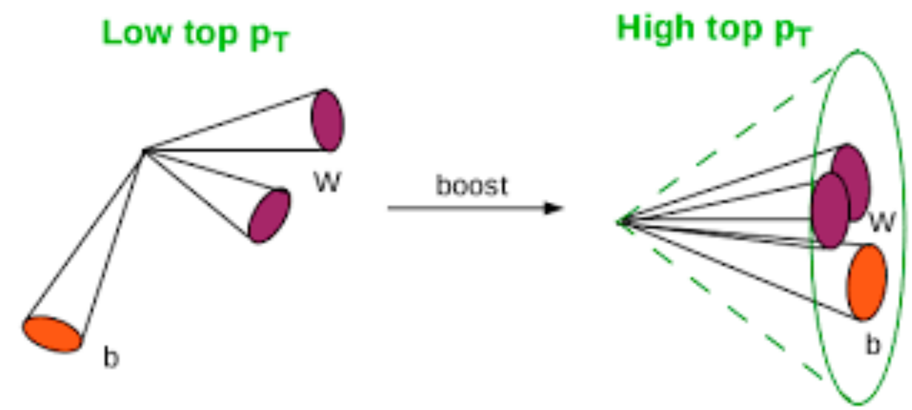


Caron, Hendriks et al. 17

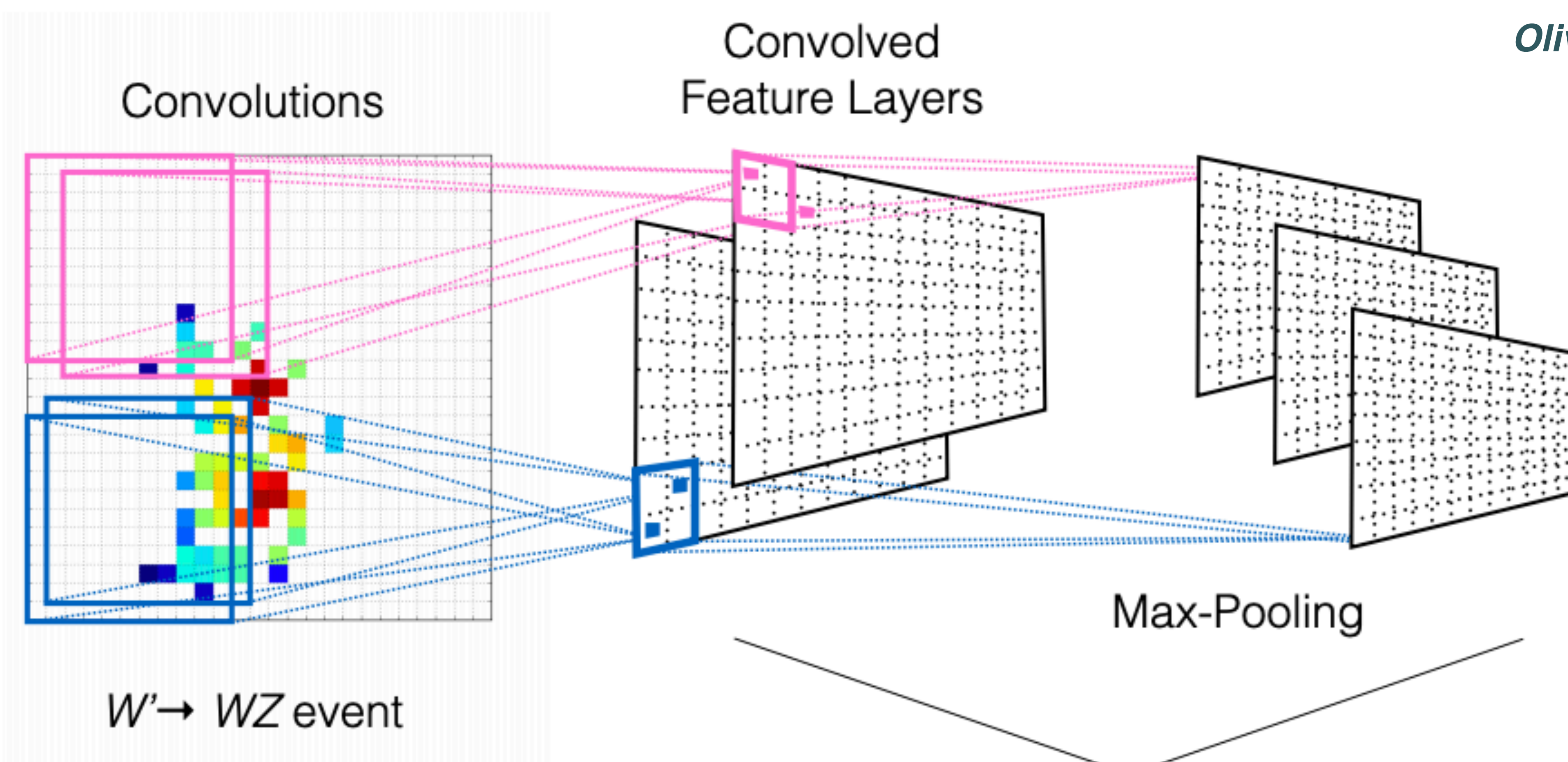
Application to LHC physics



Train CNNs on jet **images** (energy deposits in detector) to discriminate between signal and background events



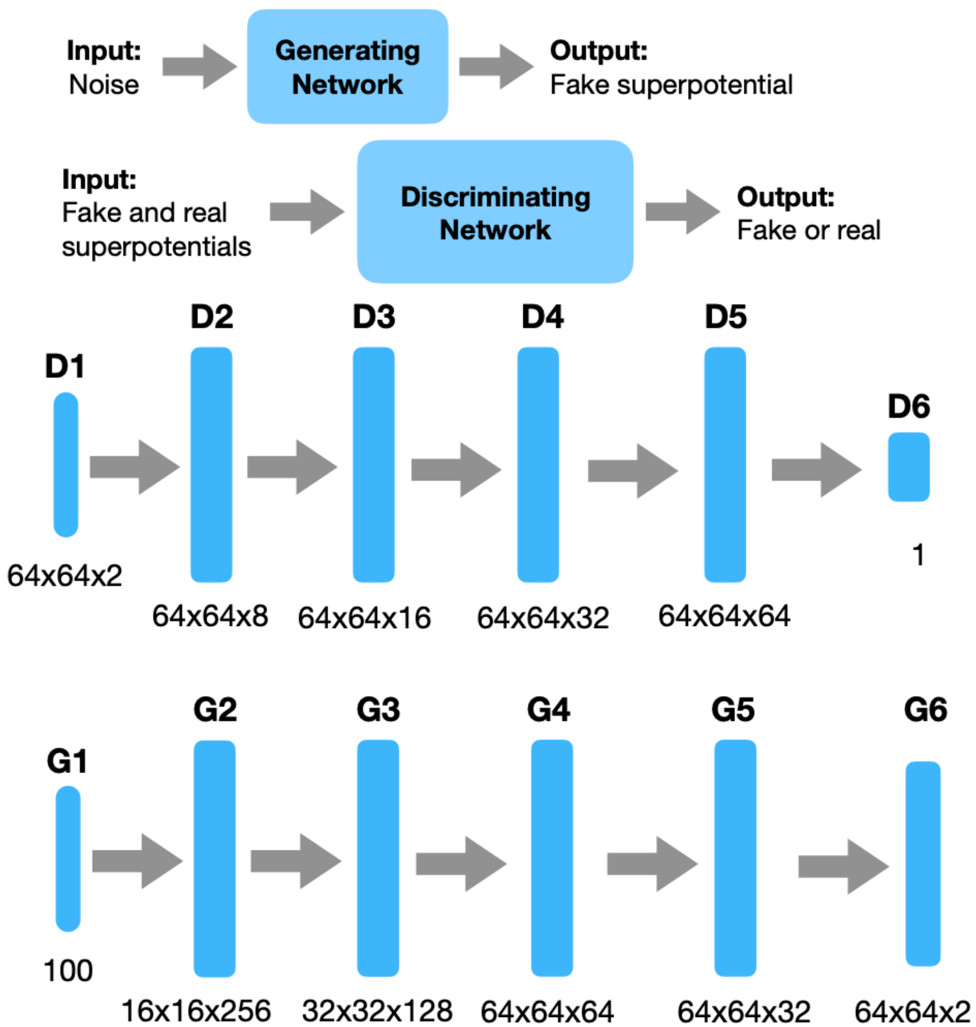
Oliveira et al. 17



earning

(More) Applications of ML to Theoretical Physics

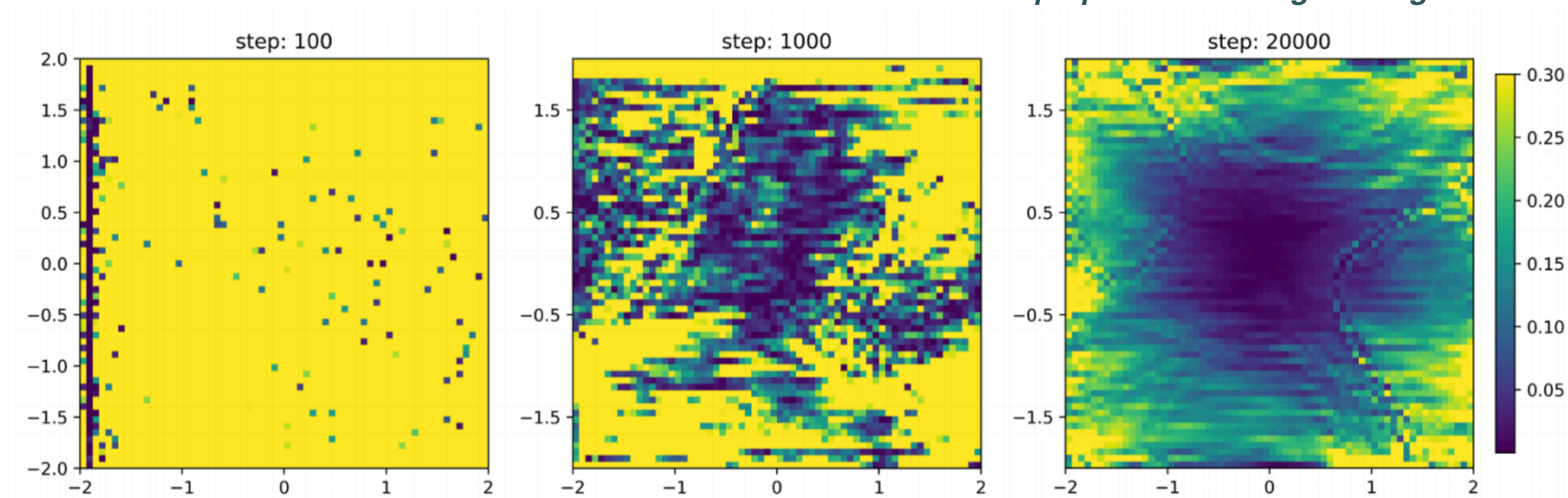
GANs for generating EFT models



- Starting from examples of **known SUSY QFTs** (with generating superpotentials) train GANs to generate other, equally consistent, theories
- After training the generator outputs new superpotentials that correspond to **new SUSY theories** not present in the training sample

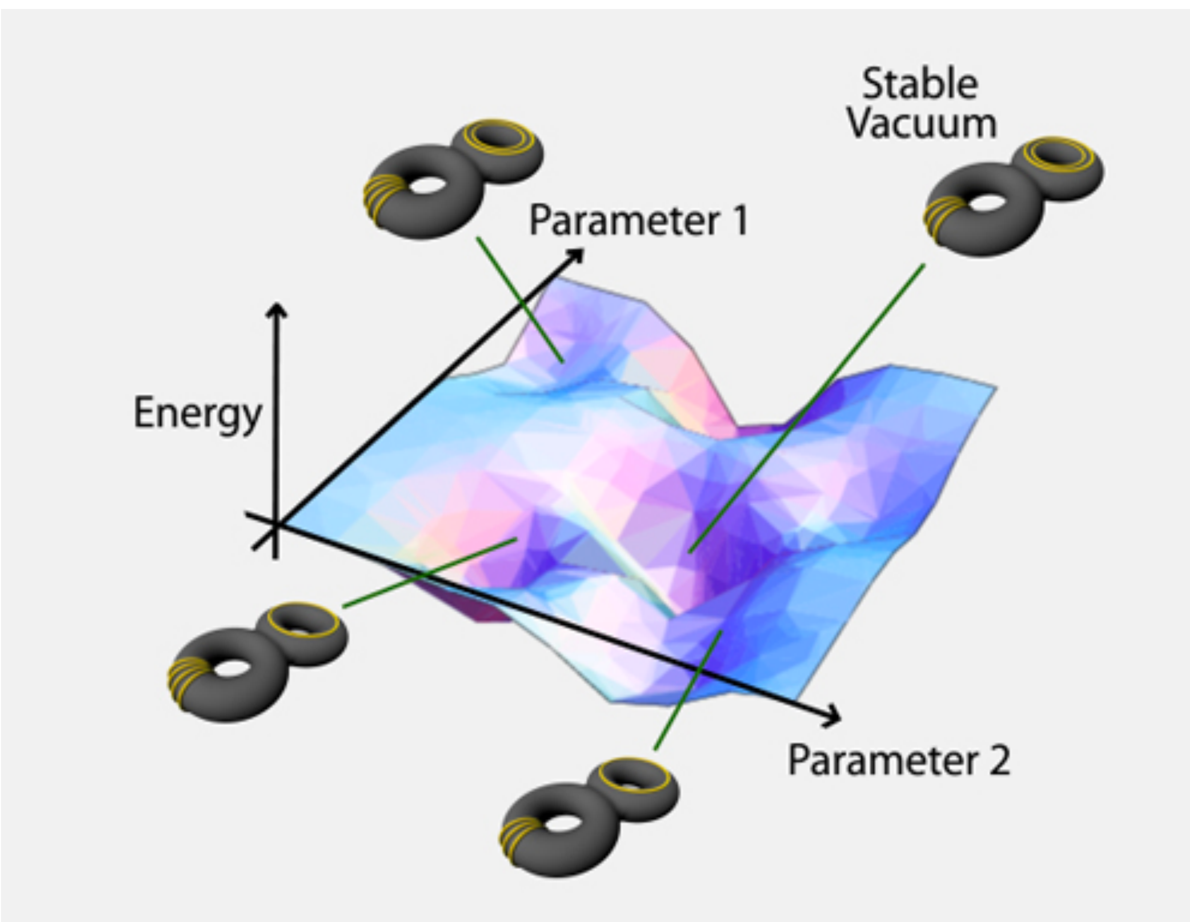
Erbin & Krippendorf 18

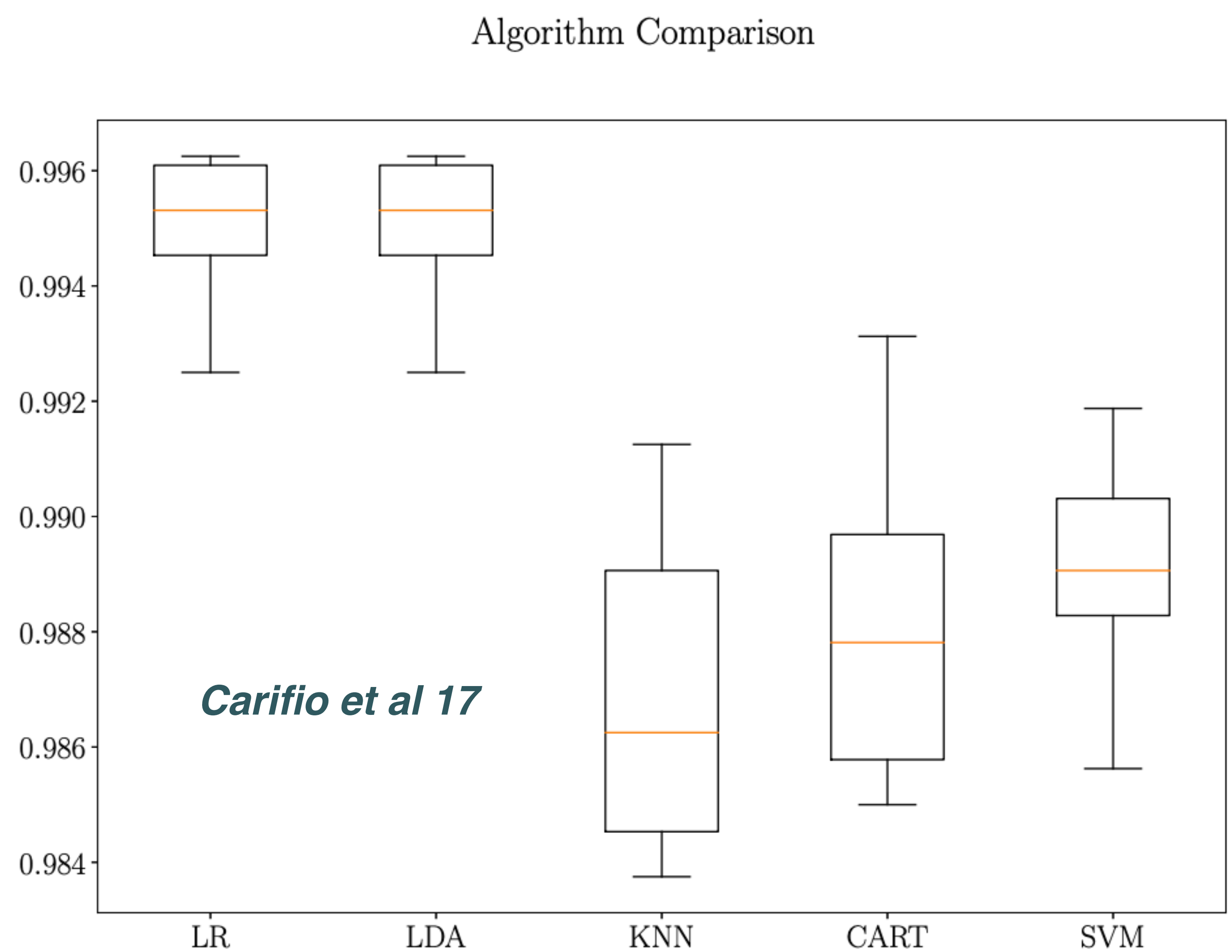
evolution of superpotential during training



Machine Learning the String Landscape

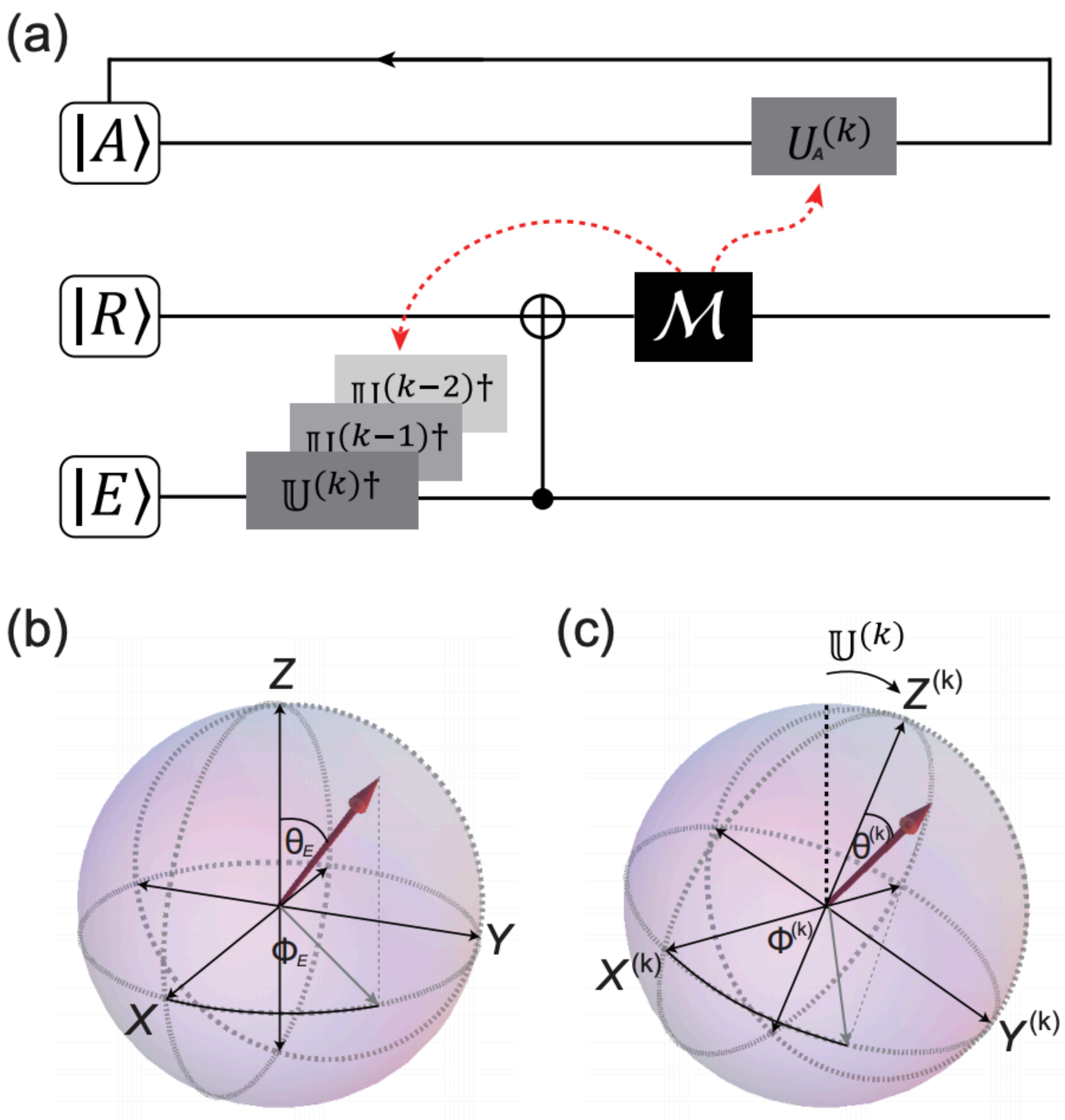
- Numerical exploration of the string theory landscape
- **Formulation and verification of conjectures** about the properties of the landscape
- Using regression, K-means clustering, decision trees, Linear Discriminant Analysis …..

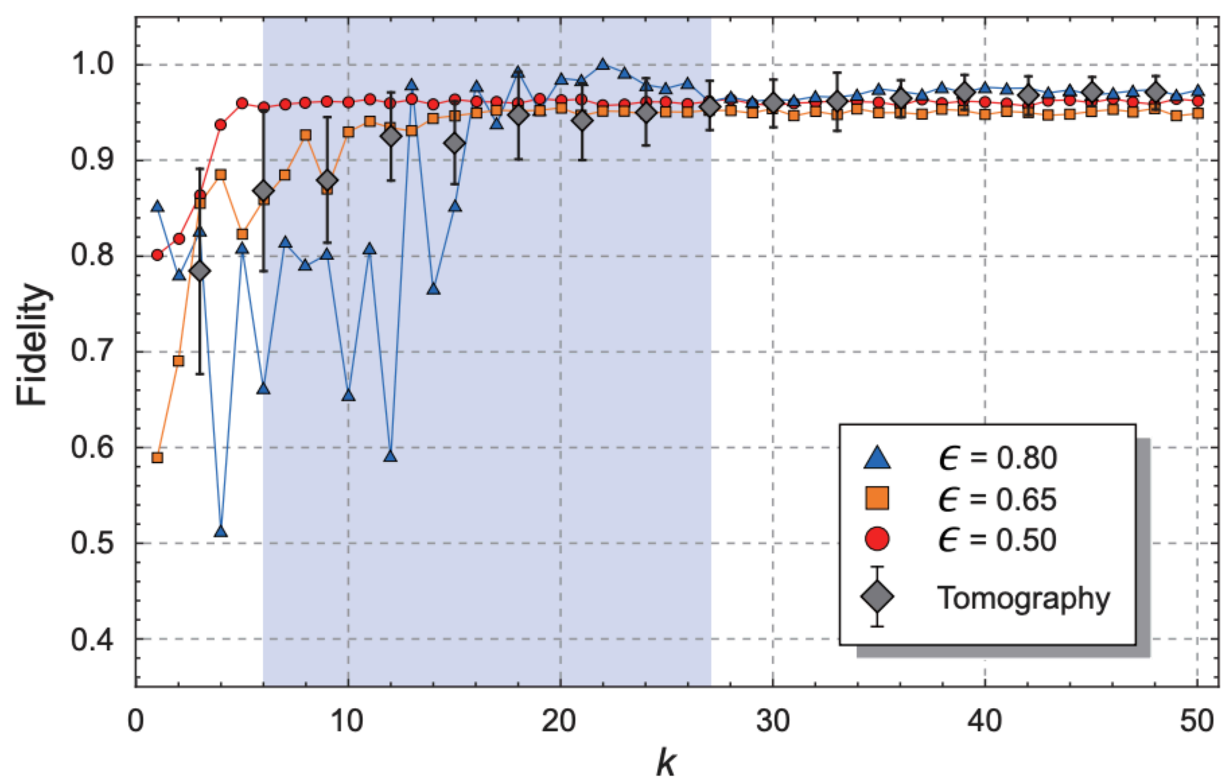




Qubit Reconstruction with Reinforcement Learning

to reconstruct an unknown photonic quantum state with a limited amount of copien, one employs a **semi-quantum reinforcement learning approach** to adapt one qubit state, an “agent”, to an unknown quantum state, an “environment”





Yu et al 19

More Examples

Machine learning and the physical sciences

Giuseppe Carleo

Center for Computational Quantum Physics, Flatiron Institute, 162 5th Avenue, New York, NY 10010, USA[*]

Ignacio Cirac

Max-Planck-Institut fur Quantenoptik, Hans-Kopfermann-Straße 1, D-85748 Garching, Germany

Kyle Cranmer

Center for Cosmology and Particle Physics, Center of Data Science, New York University, 726 Broadway, New York, NY 10003, USA

Laurent Daudet

LightOn, 2 rue de la Bourse, F-75002 Paris, France

Maria Schuld

University of KwaZulu-Natal, Durban 4000, South Africa National Institute for Theoretical Physics, KwaZulu-Natal, Durban 4000, South Africa, and Xanadu Quantum Computing, 777 Bay Street, M5B 2H7 Toronto, Canada

Naftali Tishby

The Hebrew University of Jerusalem, Edmond Safra Campus, Jerusalem 91904, Israel

Leslie Vogt-Maranto

Department of Chemistry, New York University, New York, NY 10003, USA

Lenka Zdeborová

Institut de physique théorique, Université Paris Saclay, CNRS, CEA, F-91191 Gif-sur-Yvette, France[†]

Contents

I. Introduction
- A. Concepts in machine learning
  - 1. Supervised learning and neural networks
  - 2. Unsupervised learning and generative modelling
  - 3. Reinforcement learning

II. Statistical Physics
- A. Historical note
- B. Theoretical puzzles in deep learning
- C. Statistical physics of unsupervised learning
  - 1. Contributions to understanding basic unsupervised methods
  - 2. Restricted Boltzmann machines
  - 3. Modern unsupervised and generative modelling
- D. Statistical physics of supervised learning
  - 1. Perceptron and GLMs
  - 2. Physics results on multi-layer neural networks
  - 3. Information Bottleneck
  - 4. Landscapes and glassiness of deep learning
- E. Applications of ML in Statistical Physics
- F. Outlook and Challenges

III. Particle Physics and Cosmology
- A. The role of the simulation
- B. Classification and regression in particle physics
  - 1. Jet Physics
  - 2. Neutrino physics
  - 3. Robustness to systematic uncertainties
  - 4. Triggering
  - 5. Theoretical particle physics
- C. Classification and regression in cosmology
  - 1. Photometric Redshift
  - 2. Gravitational lens finding and parameter estimation
  - 3. Other examples
- D. Inverse Problems and Likelihood-free inference
  - 1. Likelihood-free Inference
  - 2. Examples in particle physics
  - 3. Examples in Cosmology
- E. Generative Models
- F. Outlook and Challenges

IV. Many-Body Quantum Matter
- A. Neural-Network quantum states
  - 1. Representation theory
  - 2. Learning from data
  - 3. Variational Learning
- B. Speed up many-body simulations
- C. Classifying many-body quantum phases
  - 1. Synthetic data
  - 2. Experimental data
- D. Tensor networks for machine learning
- E. Outlook and Challenges

V. Quantum computing
- A. Quantum state tomography
- B. Controlling and preparing qubits
- C. Error correction

VI. Chemistry and Materials
- A. Energies and forces based on atomic environments
- B. Potential and free energy surfaces
- C. Materials properties
- D. Electron densities for density functional theory
- E. Data set generation
- F. Outlook and Challenges

VII. AI acceleration with classical and quantum hardware
- A. Beyond von Neumann architectures
- B. Neural networks running on light
- C. Revealing features in data
- D. Quantum-enhanced machine learning
- E. Outlook and Challenges

Course evaluation

The **marking** of this module is composed by **two parts**

- Writing of a **short report** (4 pages) about a specific **application of Machine Learning algorithms** to a physics problem that you find interesting or relevant for your research. This report can be written individually or in groups of at most 3 students. Including possible **code examples** is encouraged but not required

You need to send by email this report by **Friday 13th December**

- **Presentation** of the contents of this report on **Monday 16th:** 10 min + discussion

Only students that write and present their report will obtain a **pass** for the course, provided their **quality is deemed sufficient**

The examination is only required if you aim to **obtain ECs** from this course!