HISS Summer School

Machine Learning: Intro and Overview

Prof. Gregor Kasieczka
gregor.kasieczka@uni-hamburg.de

What's special about Physics?

Different fields, unified by **digitisation challenges:**

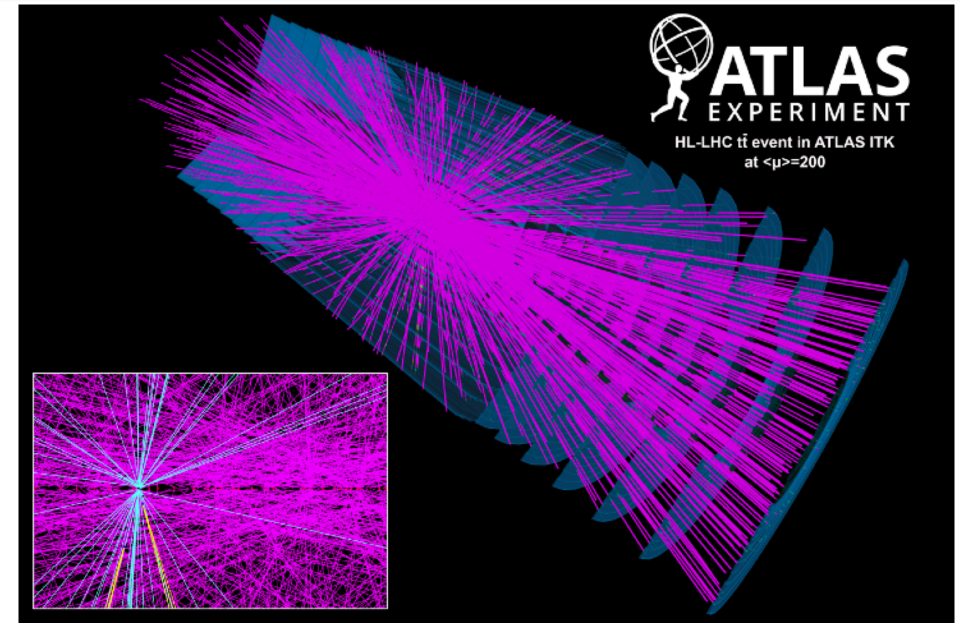
- **Immense data volumes** with high complexity
- Weak signals overshadowed by much larger background processes
- Large data-throughput requiring custom solutions

The planned Square Kilometre Array (SKA) telescope will produce one exabyte of raw data/day (~10 PB after compression)

What’s special about Physics?

Different fields, unified by **digitisation challenges:**

- Immense data volumes **with high complexity**
- Weak signals overshadowed by much larger background processes
- Large data-throughput requiring custom solutions

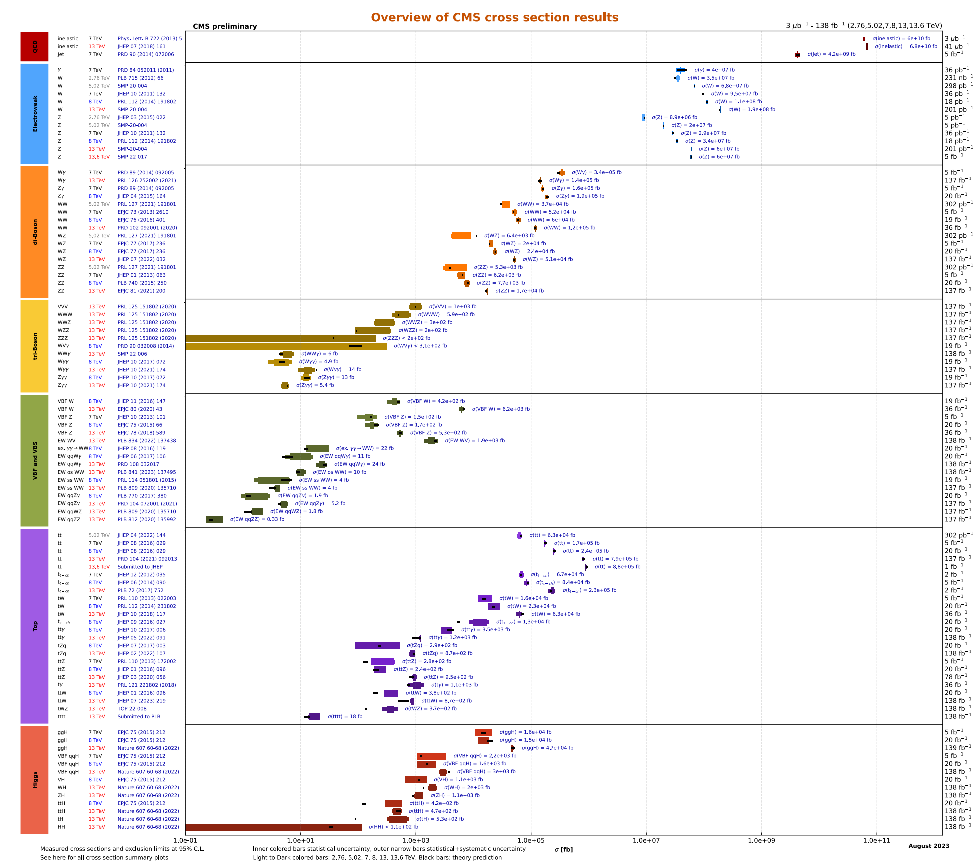


At the High Luminosity LHC, overlapping collisions mean that thousands of trajectories need to be reconstructed in parallel

What’s special about Physics?

Different fields, unified by **digitisation challenges:**

- Immense data volumes with high complexity
- **Weak signals overshadowed by much larger background processes**
- Large data-throughput requiring custom solutions

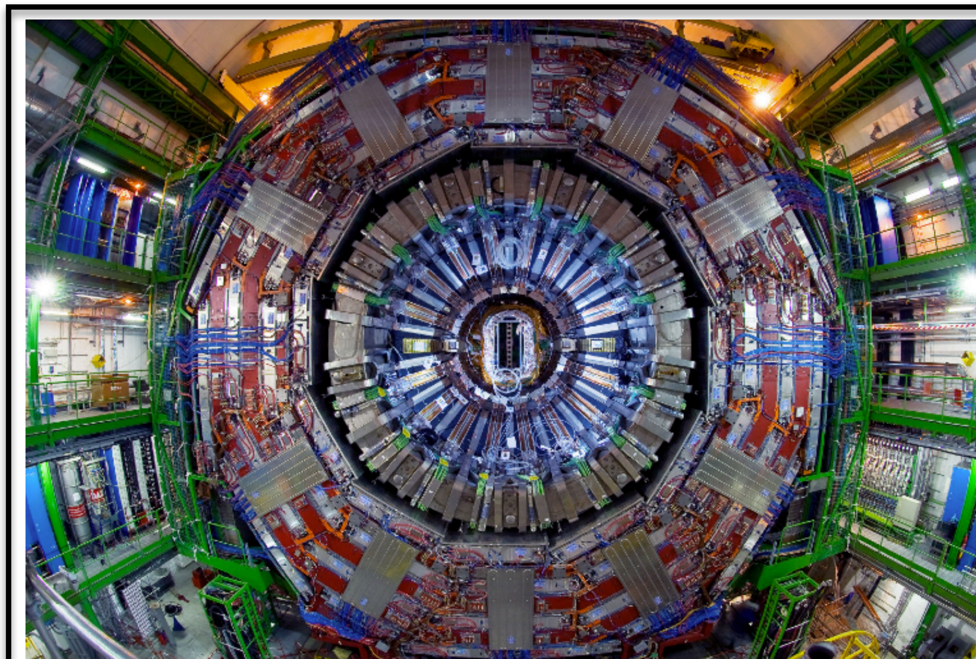


Rare interesting processes only occur in 1 of 10 billion events in particle physics experiments

What’s special about Physics?

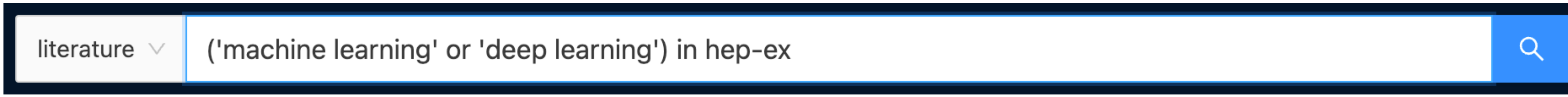
Different fields, unified by **digitisation challenges:**

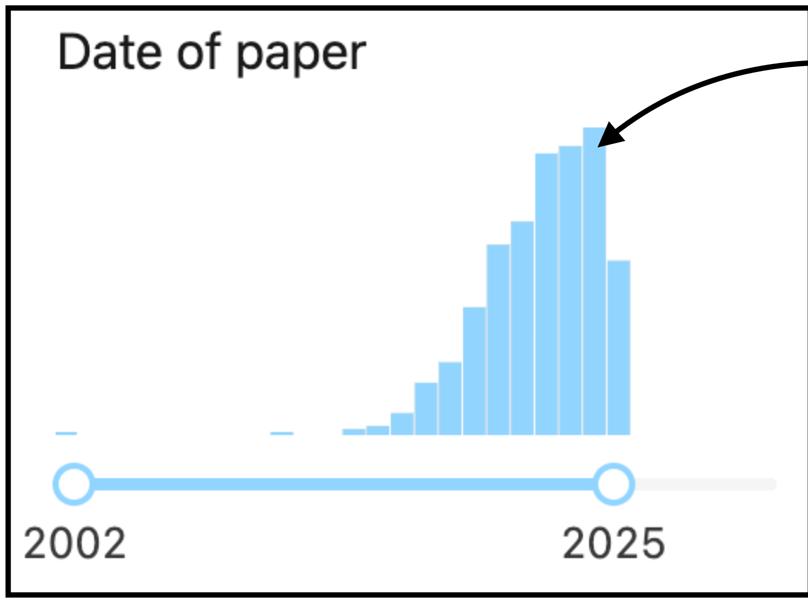
- Immense data volumes with high complexity
- Weak signals overshadowed by much larger background processes
- **Large data-throughput requiring custom solutions**



Trigger systems filter 40 million particle collisions/second at experiments at the Large Hadron Collider (LHC) at CERN

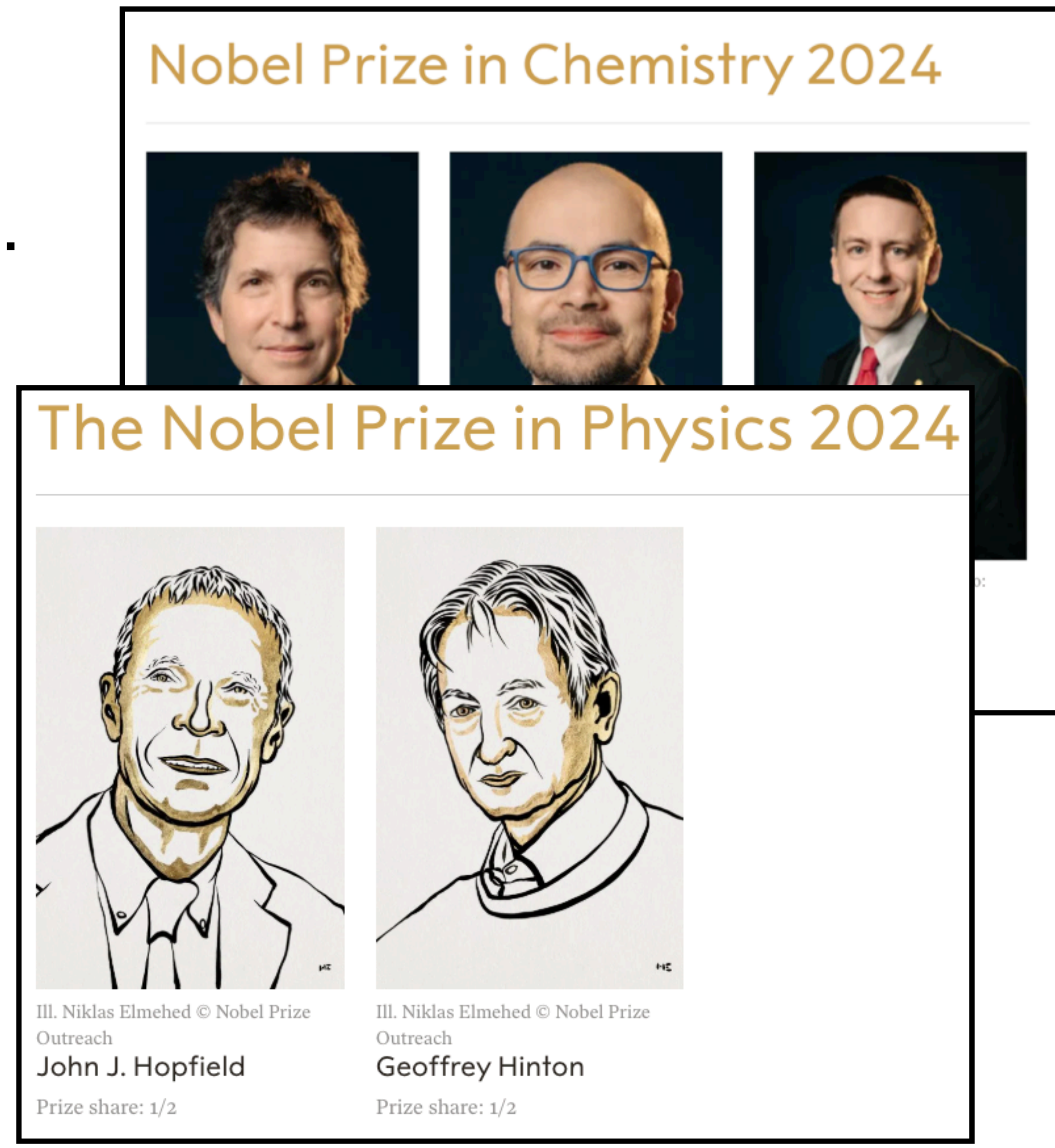
State of AI in Physics





200+ papers in 2024

Similar for nucl-ex, astro-ph, hep-ph, …



Rapid rise of AI in fundamental physics

Transition from concepts to applications

Recognition of AI work as Nobel-prize worthy

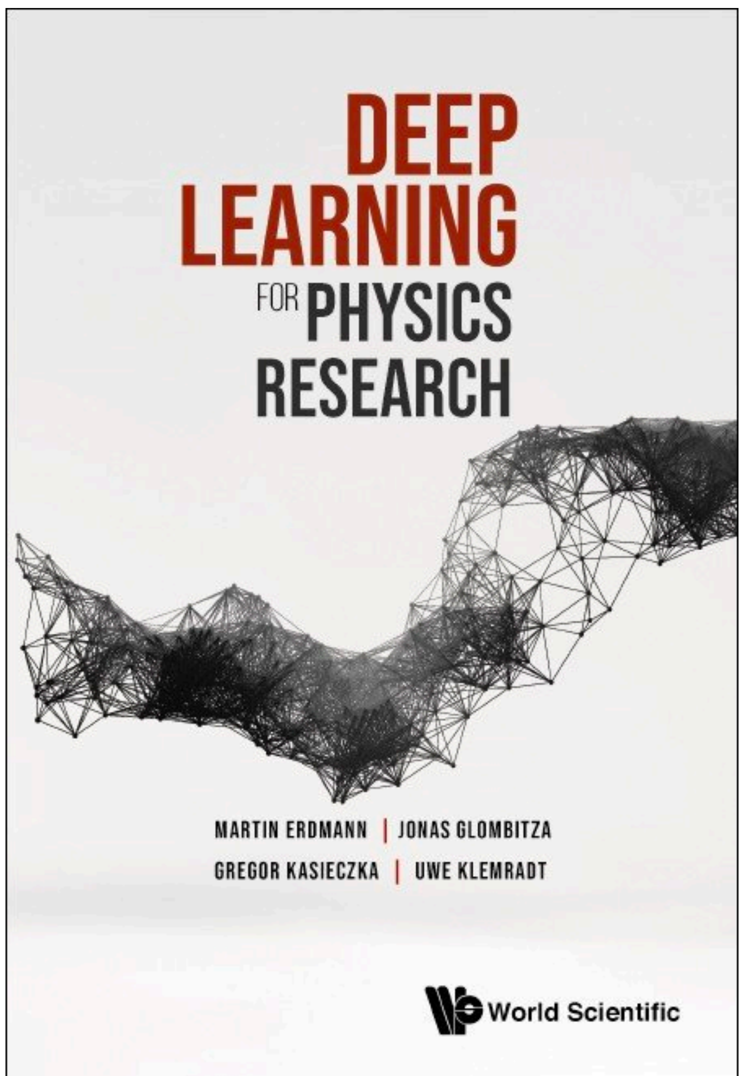
See also https://iml-wg.github.io/HEPML-LivingReview/

Plan

- Lecture 1:
  - Basics of Neural Networks
  - Classification example: Top Tagging
- Lecture 2:
  - Generative Models
  - Anomaly Detection

Resources

Free online: https://www.deeplearningbook.org/

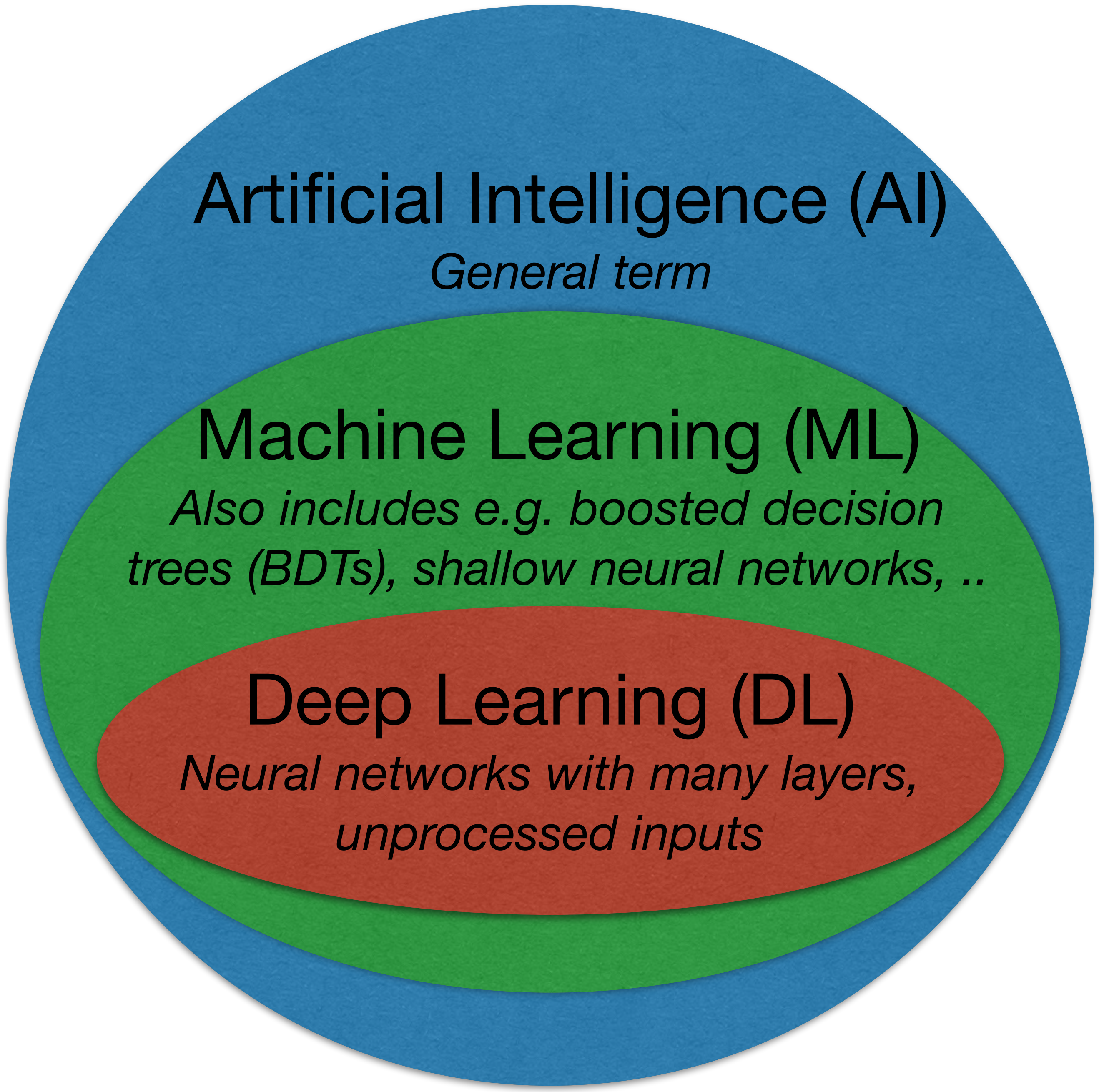


https://www.worldscientific.com/worldscibooks/10.1142/12294#t=aboutBook
(available in UHH library)

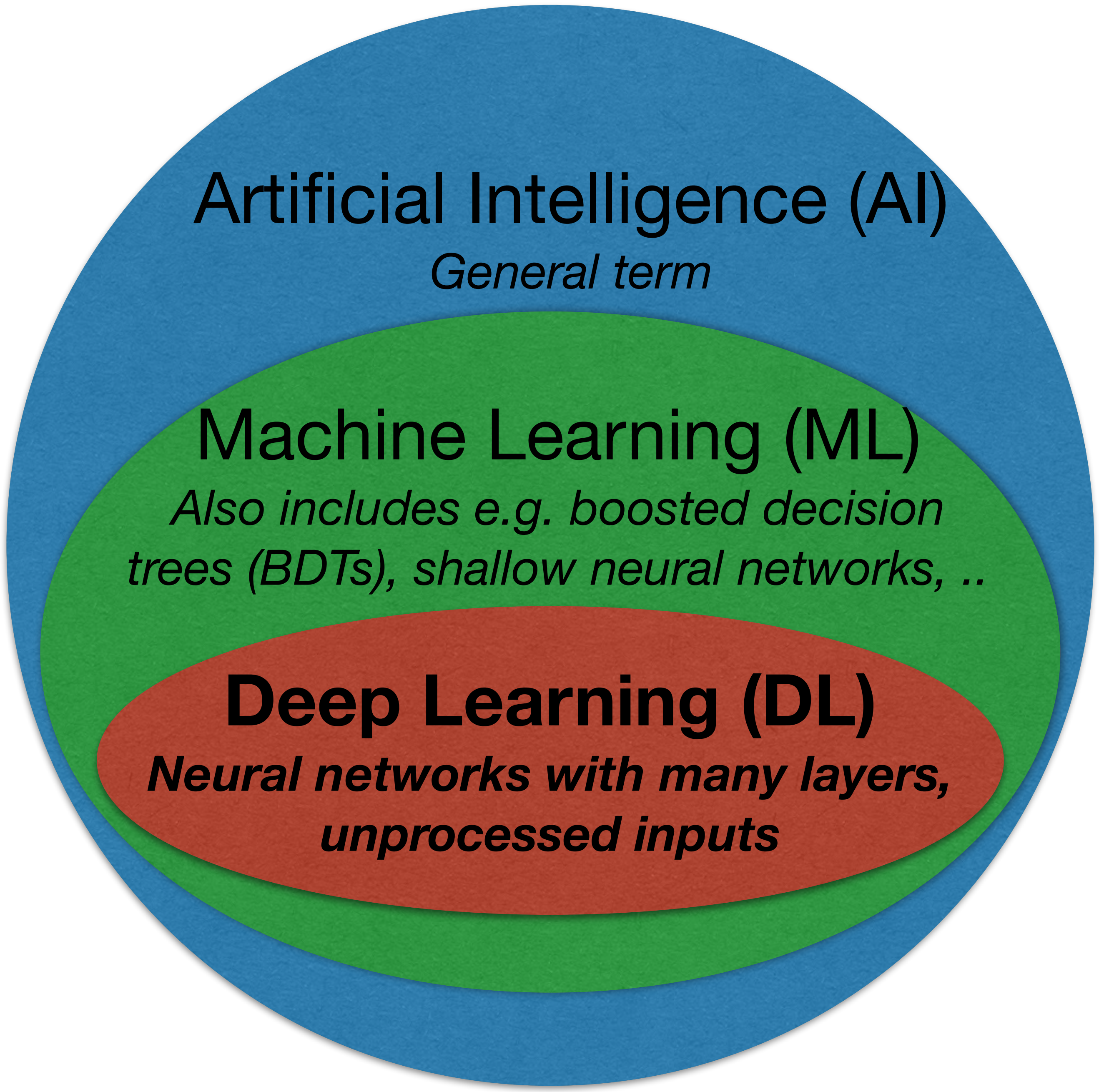
Lecture Slides

Learning like a machine

Terminology



Terminology



Basic idea

- Classical approach:
  - Write a sequence of instructions to solve a specific task
  - e.g.:
    - Tracking algorithm
    - Jet clustering
    - Calculation of physical observables

```
    if (trackerTopology.pxfSide(detId) == 1) {
      trackingLayer._side = TrackingLayer::Side::NEG_ENDCAP;
    } else if (trackerTopology.pxfSide(detId) == 2) {
      trackingLayer._side = TrackingLayer::Side::POS_ENDCAP;
    } else {
      throw cms::Exception("FastSimulation/Tracking")
          << "Tracker hit for seeding in FPix seems neither on positive nor on negative disk side: "
          << trackerTopology.print(detId).c_str();
    }
    trackingLayer._layerNumber = trackerTopology.pxfDisk(detId);
  }
  //TIB
  else if (subdet == StripSubdetector::TIB) {
    trackingLayer._subDet = TrackingLayer::Det::TIB;
    trackingLayer._side = TrackingLayer::Side::BARREL;
    trackingLayer._layerNumber = trackerTopology.tibLayer(detId);
  }
  //TID
  else if (subdet == StripSubdetector::TID) {
    trackingLayer._subDet = TrackingLayer::Det::TID;
    if (trackerTopology.tidSide(detId) == 1) {
      trackingLayer._side = TrackingLayer::Side::NEG_ENDCAP;
    } else if (trackerTopology.tidSide(detId) == 2) {
      trackingLayer._side = TrackingLayer::Side::POS_ENDCAP;
    } else {
      throw cms::Exception("FastSimulation/Tracking")
          << "Tracker hit for seeding in TID seems neither on positive nor on negative disk side: "
          << trackerTopology.print(detId).c_str();
    }
    trackingLayer._layerNumber = trackerTopology.tidWheel(detId);
  }
  //TOB
  else if (subdet == StripSubdetector::TOB) {
```

Basic idea

- Machine learning approach:
  - Rephrase task as a minimisation problem..
  - ..and “simply” solve:

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

  - Will now go step-by-step to understand the underlying ideas, focusing on neural networks.

General Strategy

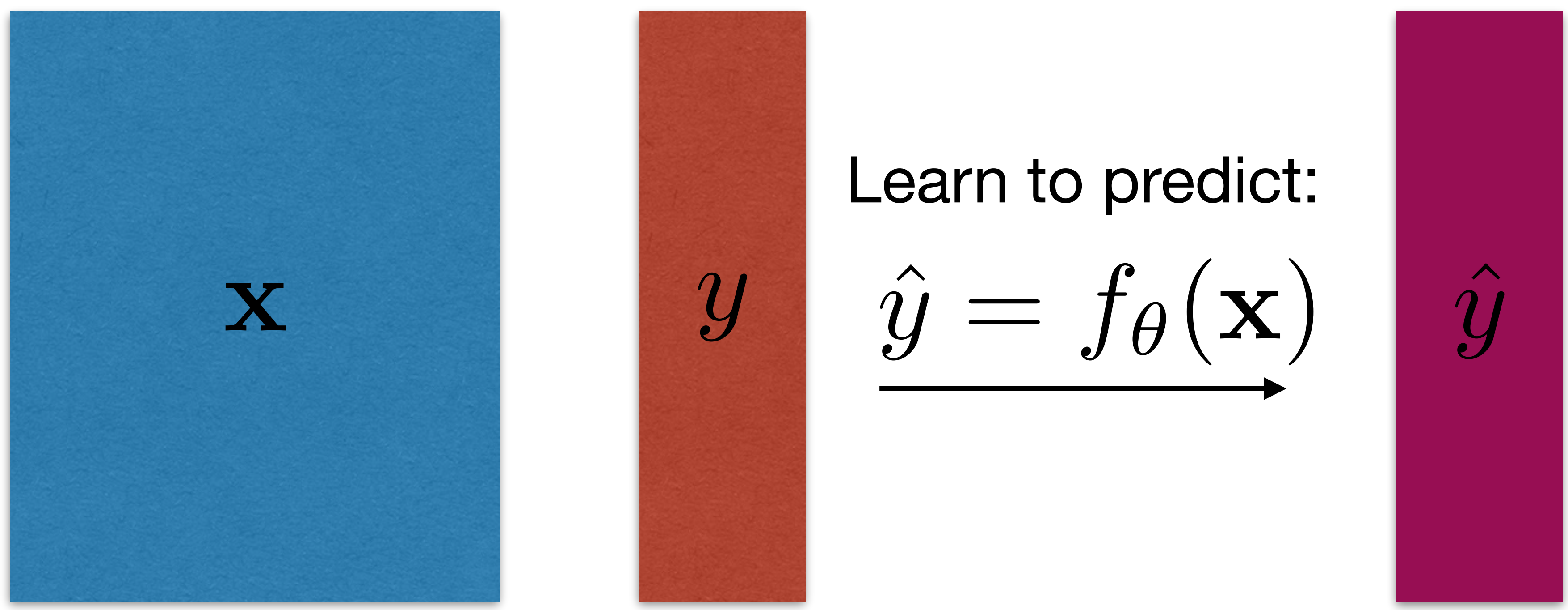
$\text{Loss function } \mathcal{L}$

- Define an optimisation target (loss function)

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

Regression: Minimize mean squared error:

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

Classification Tasks

Distinguish a pair of classes (binary) or several (multi-class).

Images

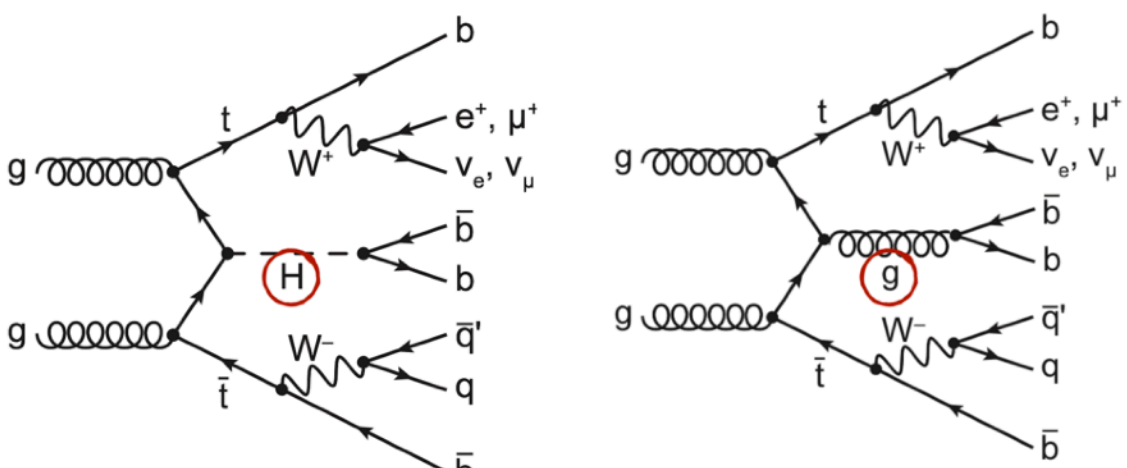
Loss function: Cross entropy

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

Classification Tasks

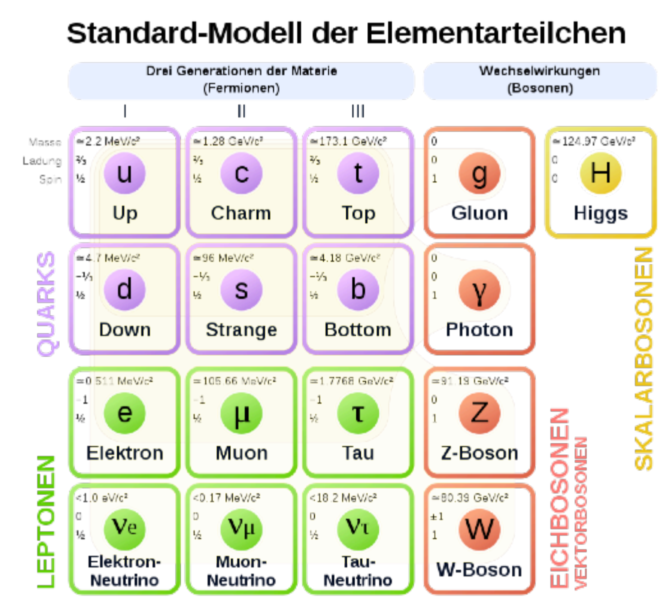
Distinguish a pair of classes (binary) or several (multi-class).

Event Topologies



signal *difficult background*

Particles



Loss function: Cross entropy

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

Cross Entropy

- Example: Binary classification
  - 2 Classes with labels: $y_k = 0, 1$
  - Build the NN such that we the output is a probability: $\hat{y}(x_k) \in [0,1]$
- -> Bernoulli Trial with probabilities p and $\mathrm{q} = 1 - \mathrm{p}$
- Likelihood:

$$L = \prod_k^N [\, y_k \cdot \hat{y}(x_k) + (1 - y_k) \cdot (1 - \hat{y}(x_k)) \,]$$

- We want to minimize the average negative log-likelihood:

$$\boxed{-\frac{1}{N}\log(L) = -\frac{1}{N}\sum_k^N [y_k \cdot \log(\hat{y}(x_k)) + (1 - y_k) \cdot \log(1 - \hat{y}(x_k))]}$$

Jacob Bernoulli 1654-1705

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.

$$\mathbf{x}$$

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)

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

General Strategy

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

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

General Strategy

Loss function $\mathcal{L}$

Neural network f_θ

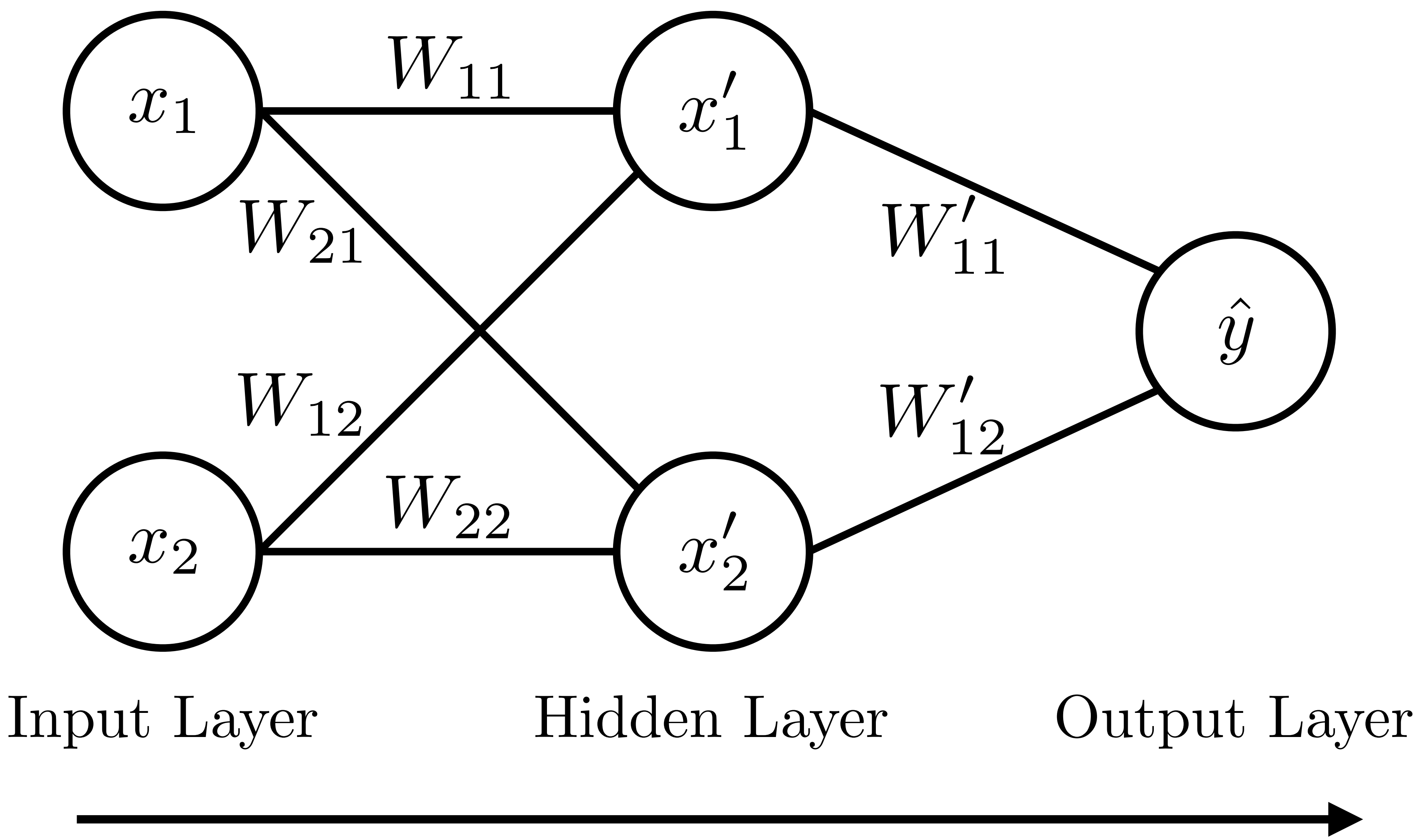
Parameters θ

- Define an optimisation target (loss function)
- Choose a non-linear, expressive, parametrised function (e.g. a neural network)

Neural networks

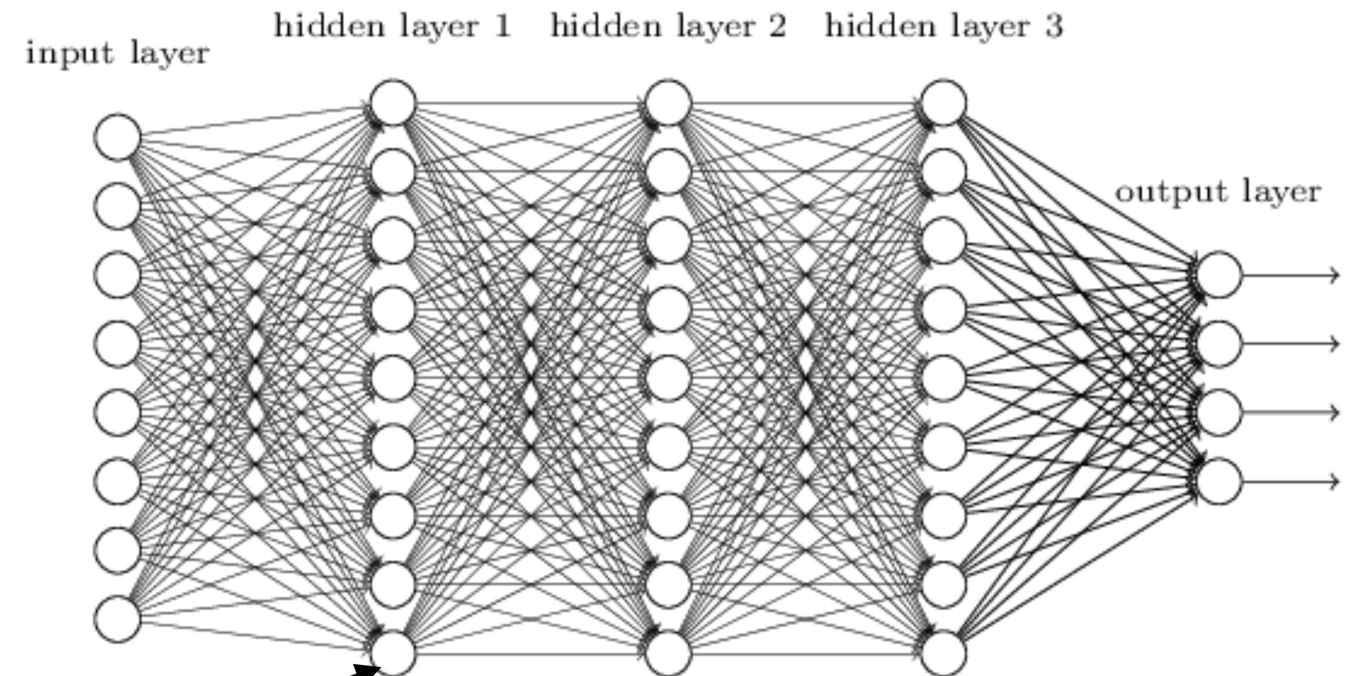
We need an expressive function
(universal approximator)
with tuneable parameters and useful
implicit biases

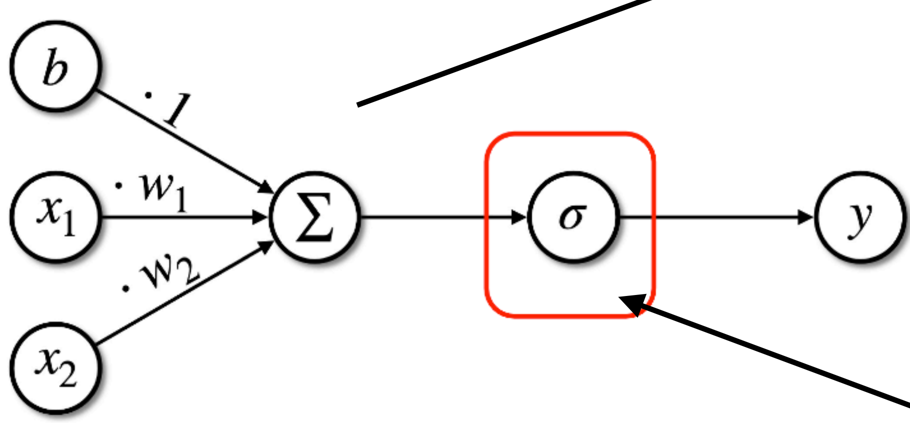
$$x'_1 = f(W_1 1 \cdot x_1 + W_1 2 \cdot x_2)$$



Neural Networks

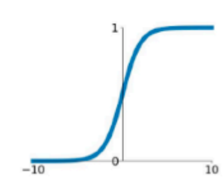
We need an expressive function
(universal approximator)
with tuneable parameters:
Neural networks



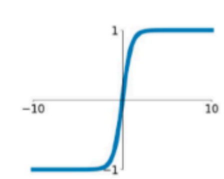


Activation Functions

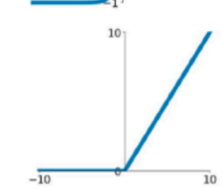
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



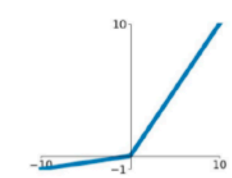
tanh
$\tanh(x)$



ReLU
$\max(0, x)$

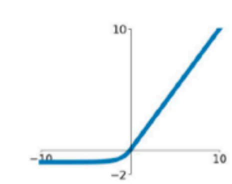


Leaky ReLU
$\max(0.1x, x)$

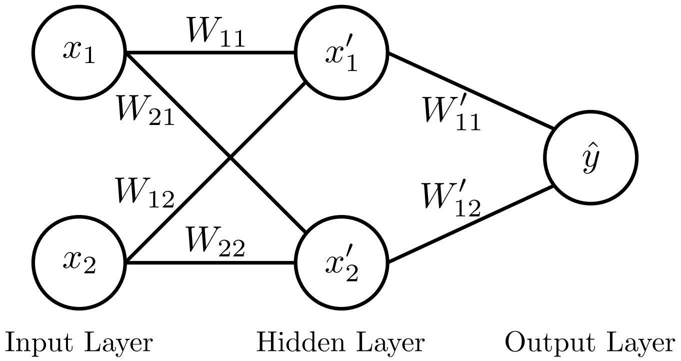


Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

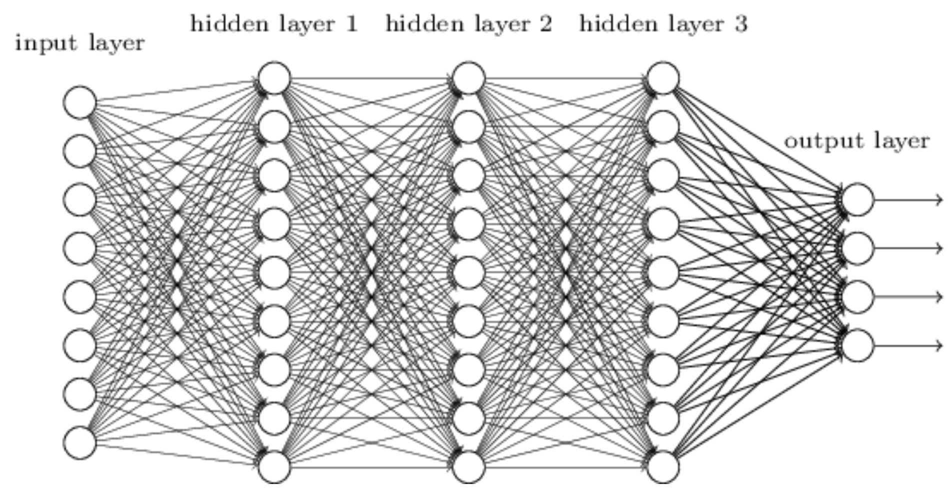
ELU
$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$



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
		$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 128\\3\times3, 128, C{=}32\\1\times1, 256\end{bmatrix}\times3$
conv3	28×28	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 256\\3\times3, 256, C{=}32\\1\times1, 512\end{bmatrix}\times4$
conv4	14×14	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 512\\3\times3, 512, C{=}32\\1\times1, 1024\end{bmatrix}\times6$
conv5	7×7	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 1024\\3\times3, 1024, C{=}32\\1\times1, 2048\end{bmatrix}\times3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		$\mathbf{25.5}\times10^6$	$\mathbf{25.0}\times10^6$
FLOPs		$\mathbf{4.1}\times10^9$	$\mathbf{4.2}\times10^9$

175 billion weights: *2020 GPT-3 text model (GPT-4 ~**1.8 trillion weights**)*

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}

General Strategy

Loss function $\mathcal{L}$ ✓

Neural network f_θ ✓

Parameters θ ✓

Opt. Parameters θ^*

Data $\mathbf{x}$

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

- Define an optimisation target (loss function)
- Choose a non-linear, expressive, parametrised function (e.g. a neural network)
- Use training data to optimise parameters, then apply to new examples

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

How do networks learn?

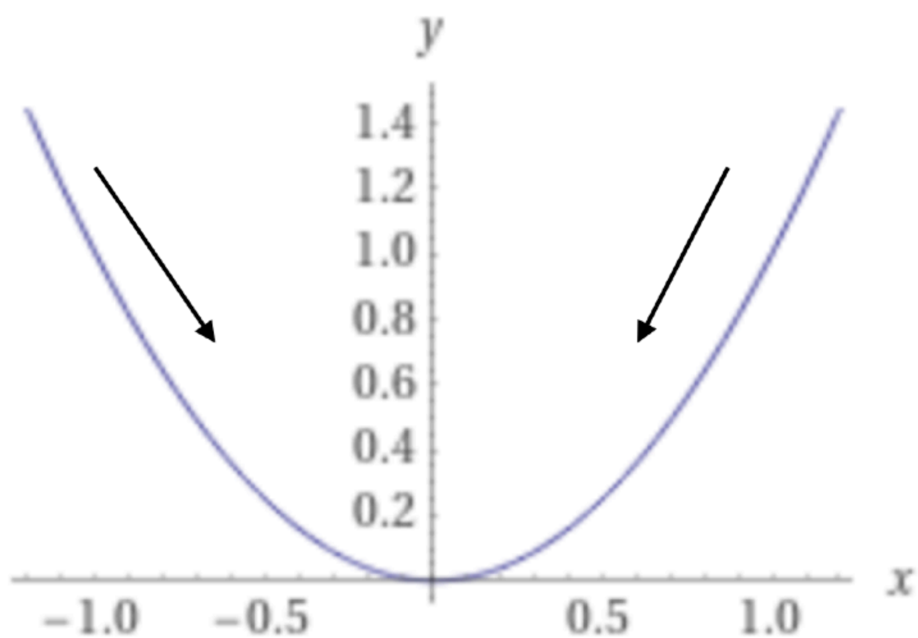
- *Backpropagation + Gradient descent*
- Important: Loss function needs to be differentiable
  - (Or find a differentiable approximation)
- Pass input (x_1, x_2, …) to networks
- From output calculate loss function
  Find gradient of loss function with respect to weights
- Use gradient to find new weights

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

Learning rate

- In practice, handled by optimise algorithm (e.g. Adam)

How do networks learn?



$$f(x) = x^2$$

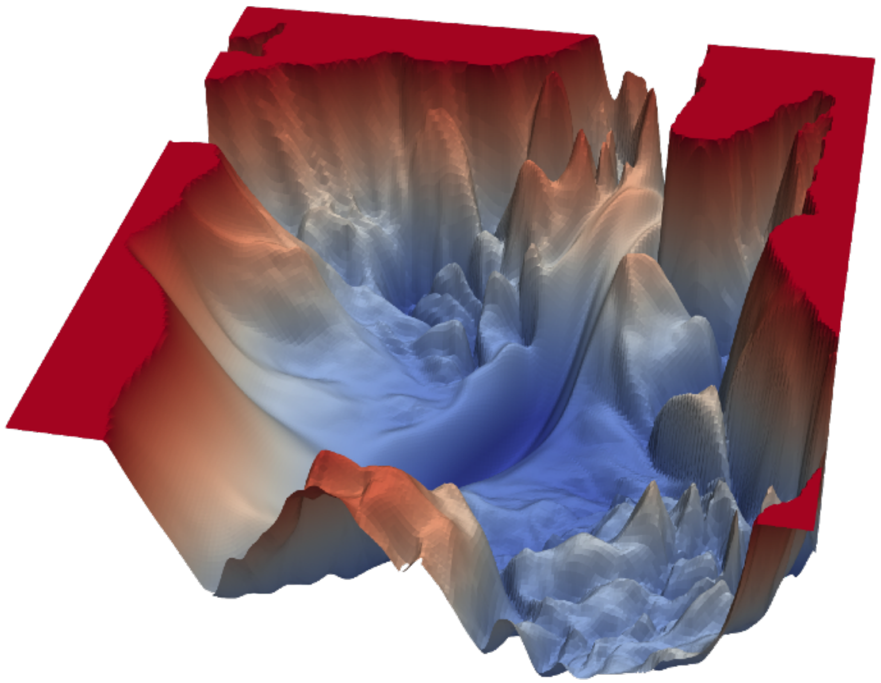
$$f'(x) = 2x$$

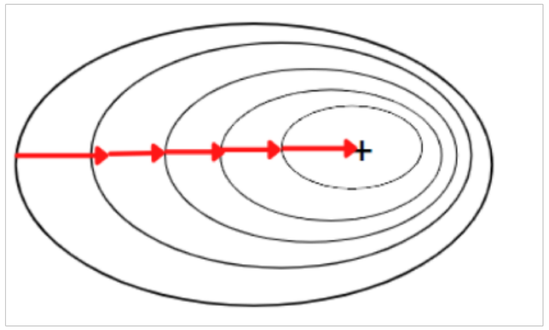
$x < 0 \implies$ -$f'(x) > 0$ $\qquad$ $x > 0 \implies$ -$f'(x) < 0$

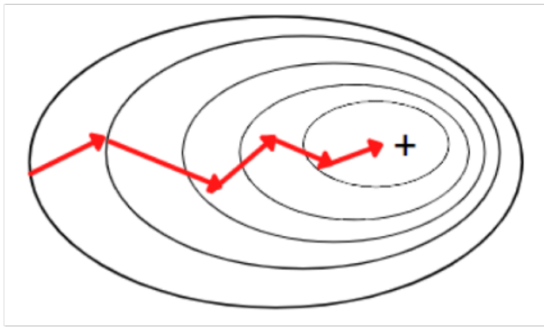
"Go right" $\qquad$ "Go left"

Batches and Epochs

- The loss depends on O(1k-1b) parameters
- To minimize it these parameters need to be adjusted
- For large networks the loss landscape can get very complicated with many local minima
- Better convergence
  - Multiple passes (epochs) over training data
  - Split training data in batches with a minimization call and model update after each batch



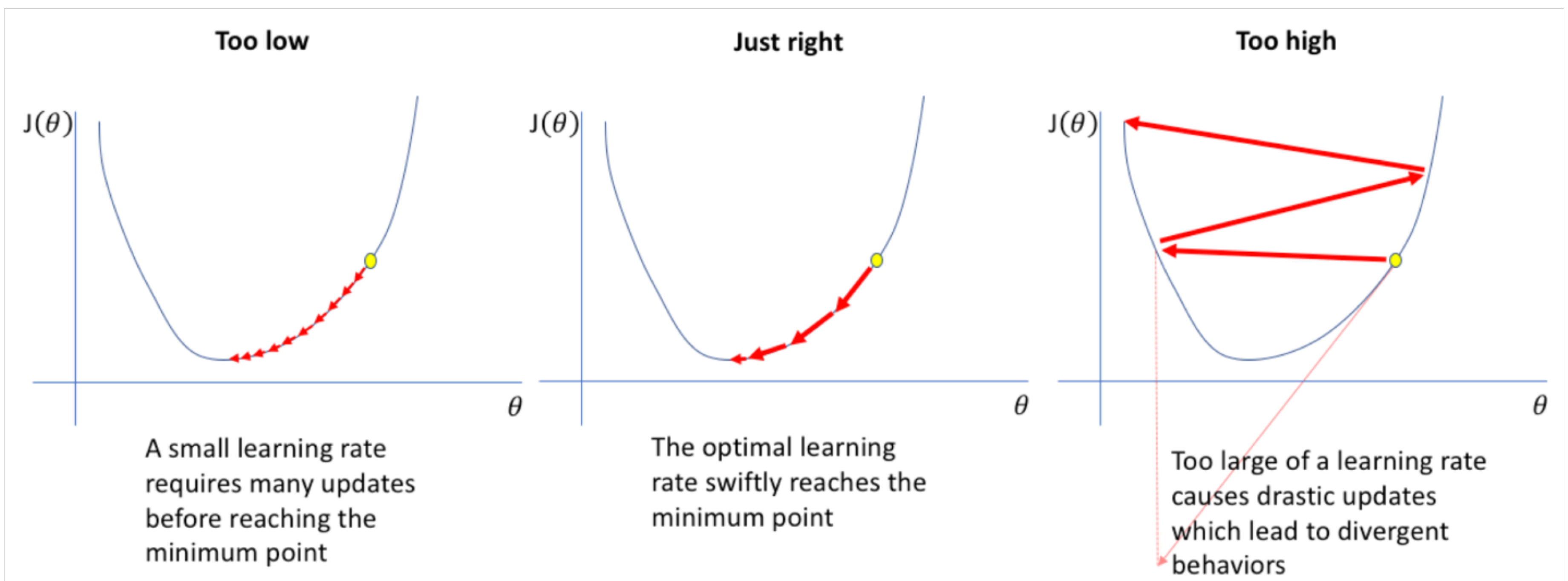




Learning Rate

- Many popular minimization algorithms have a tunable parameter called Learning rate α
- Should be chosen such that the training smoothly converges

$$\theta \rightarrow \theta - \alpha \frac{\partial L(\theta)}{\partial \theta}$$



https://www.jeremyjordan.me/nn-learning-rate/

Adam

$$m_w^{(t+1)} \leftarrow \beta_1 m_w^{(t)} + (1-\beta_1)\,\nabla_w L^{(t)}$$

$$v_w^{(t+1)} \leftarrow \beta_2 v_w^{(t)} + (1-\beta_2)\left(\nabla_w L^{(t)}\right)^2$$

Maintain history of previous changes: momentum

Exponential moving average of gradient and gradient squares

$$\hat{m}_w = \frac{m_w^{(t+1)}}{1-\beta_1^t}$$

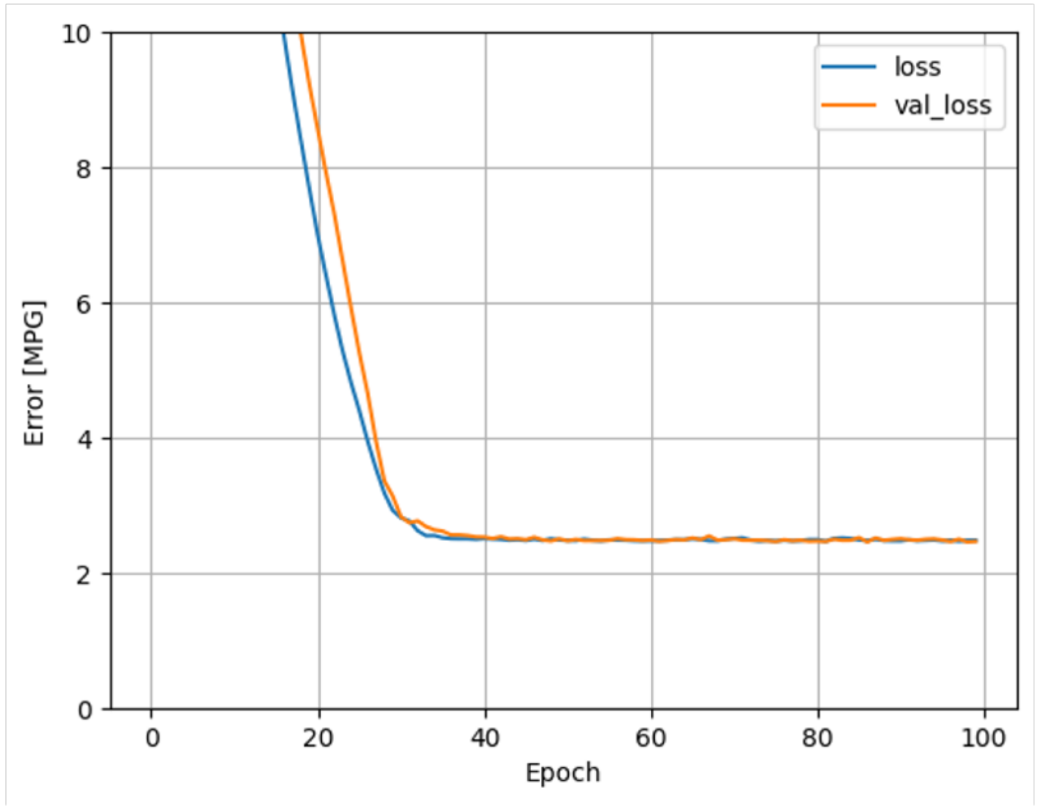
$$\hat{v}_w = \frac{v_w^{(t+1)}}{1-\beta_2^t}$$

$$w^{(t+1)} \leftarrow w^{(t)} - \eta \frac{\hat{m}_w}{\sqrt{\hat{v}_w} + \epsilon}$$

https://arxiv.org/abs/1412.6980

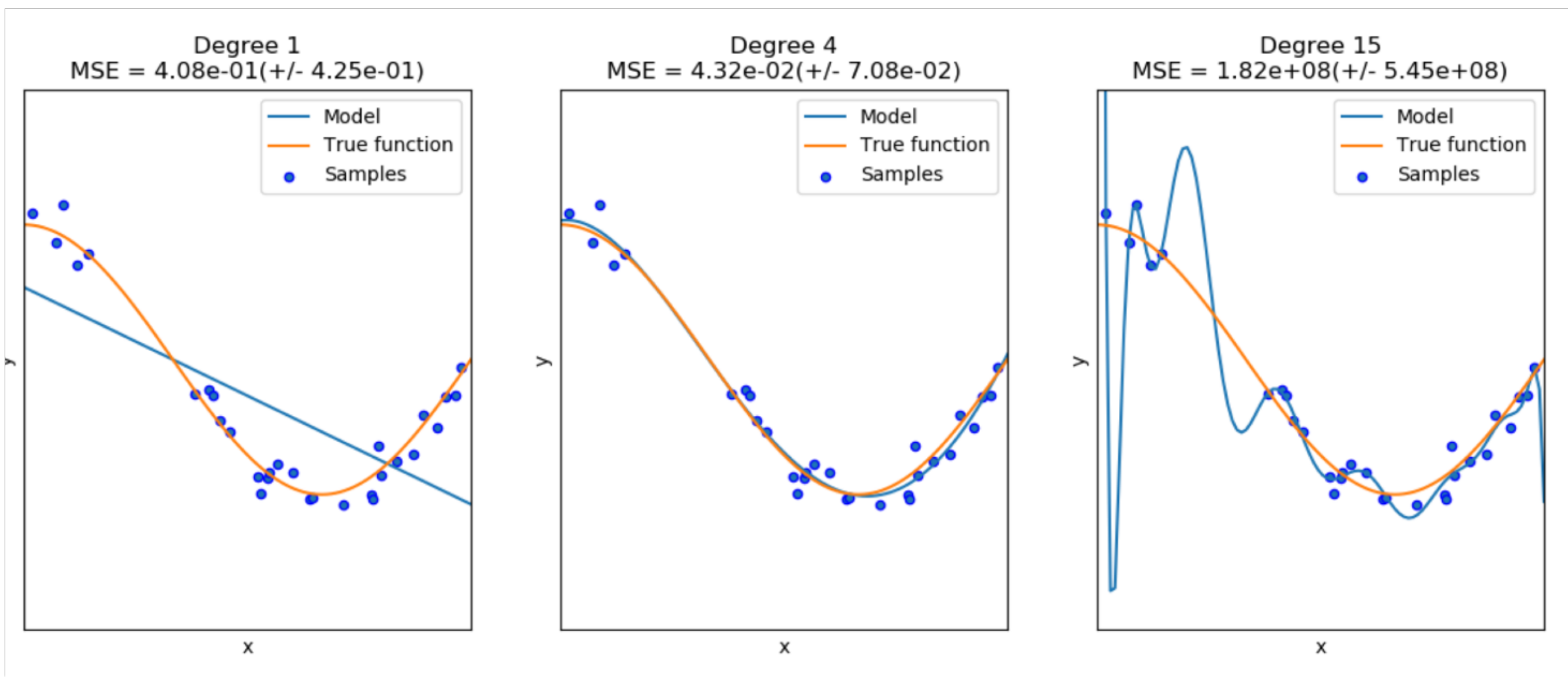
The loss curve

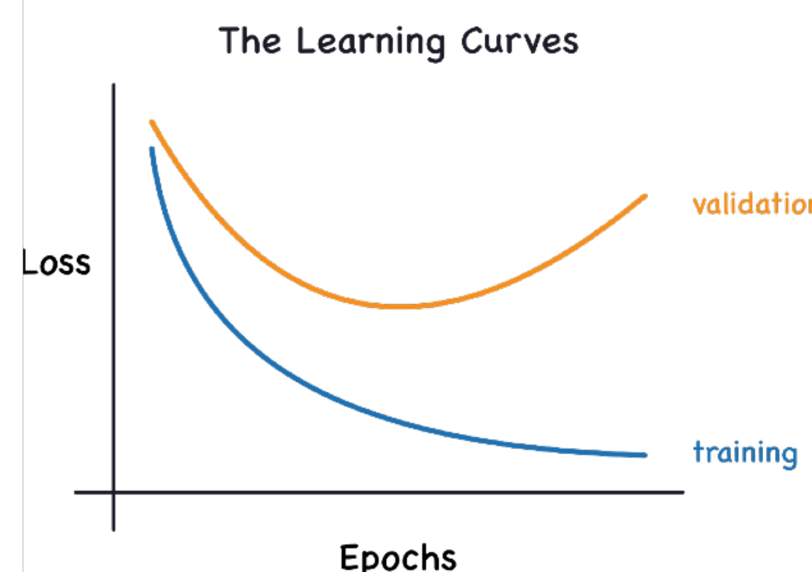
- Once the training is done it is time to analyze the training process and the performance of the model
- For the training process it is a good idea to look at the loss as a function of the number of epochs – for the training data and validation data
- Ideally the training converges with similar loss values for the validation and training data



Overfitting

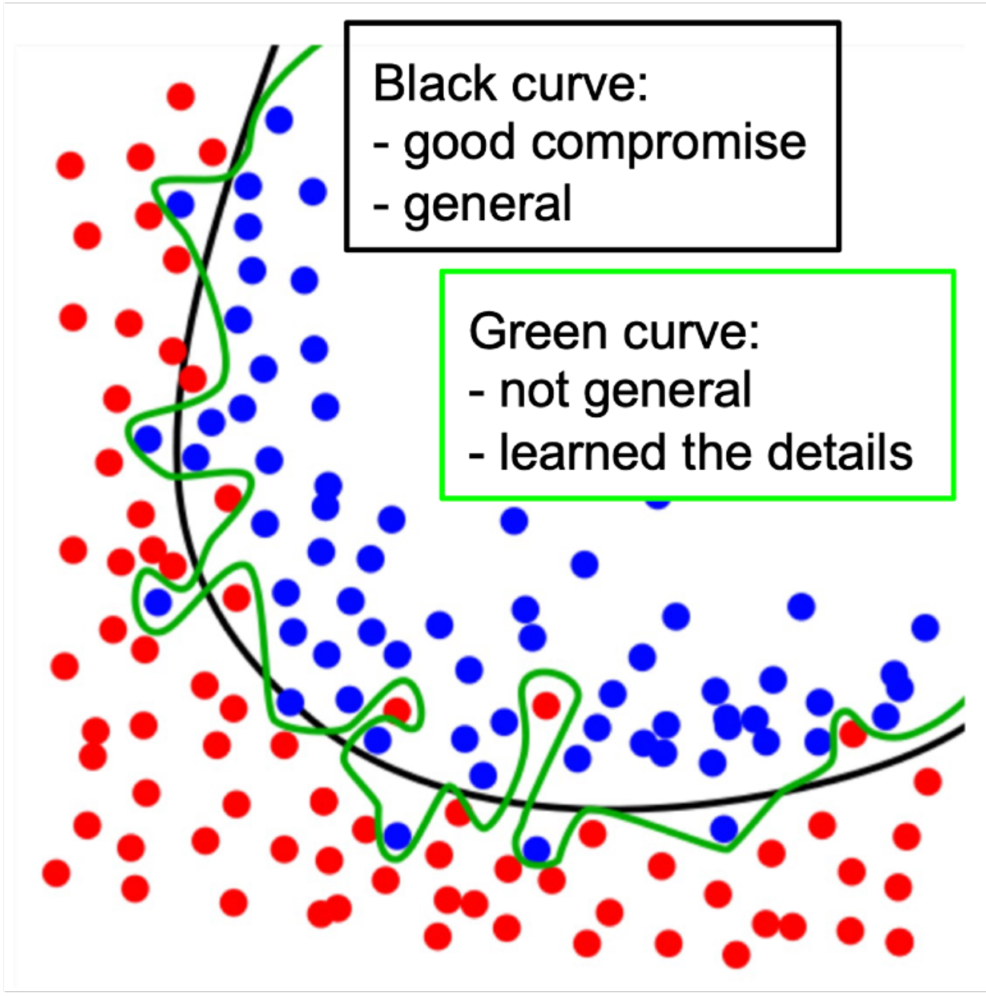
- If the model has too many parameters it can overfit the data
- This leads to a worse performance on data with different statistical fluctuations than the training data
- -> the training and validation losses diverge





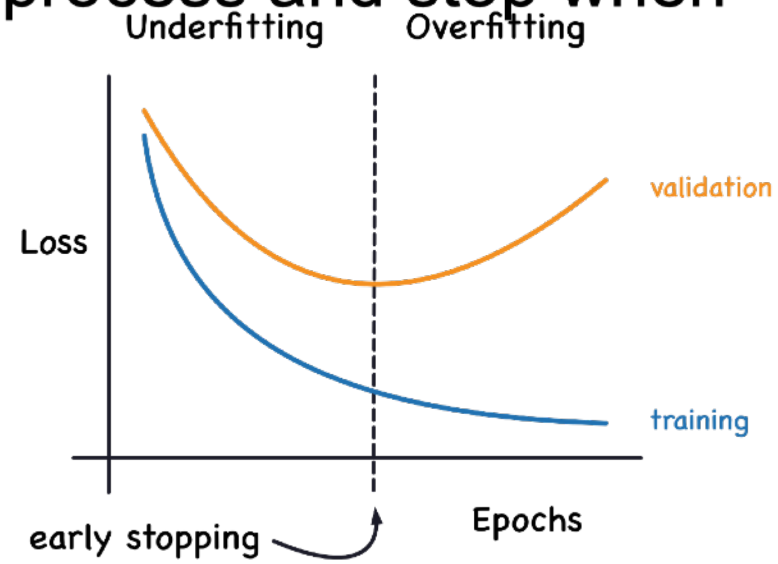
Overfitting

- Also an issue in classification tasks (which are after all a form of regression of discrete values)
- Overfitting can be interpreted as remembering the training samples
- We can see that the performance would change if the dots move around even a bit



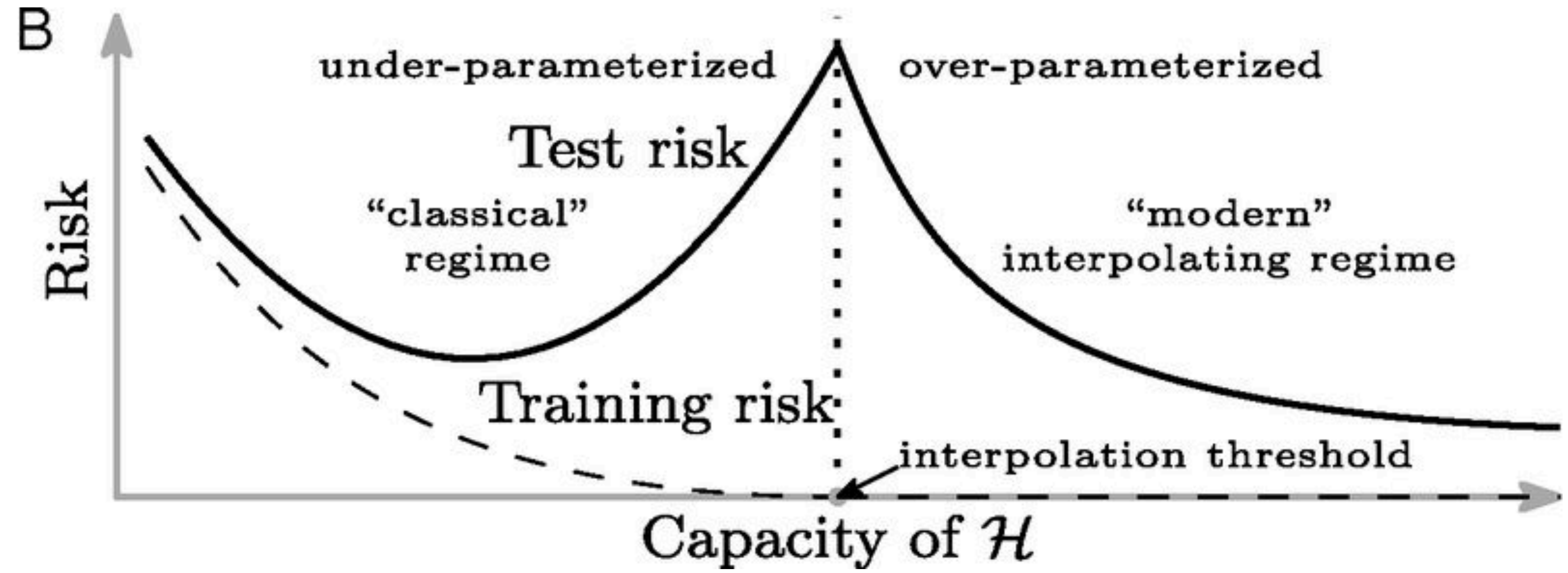
Overfitting - Mitigation

- There are multiple ways to mitigate overfitting such as dropout, regularization or early stopping
- Dropout:
  - Add layers that randomly discard a fraction of inputs in training phase
- Early stopping:
  - Monitor training and validation loss during the training process and stop when they diverge
- Regularisation:
  - Add penalty for large weight values in loss function



Side note: Overfitting

Reconciling modern machine-learning practice and the classical bias–variance trade-off (PNAS, Belkin et al)



- Recent observation in computer science
  - Extremely high capacity does not guarantee overfitting
- Even more recent:
  - Grokking: Sudden generalisation of a trained model
- Active research into foundations, does not stop us from using it

General Strategy

Loss function $\mathcal{L}$ ✓

Neural network f_θ✓

Parameters θ✓

Opt. Parameters θ^*✓

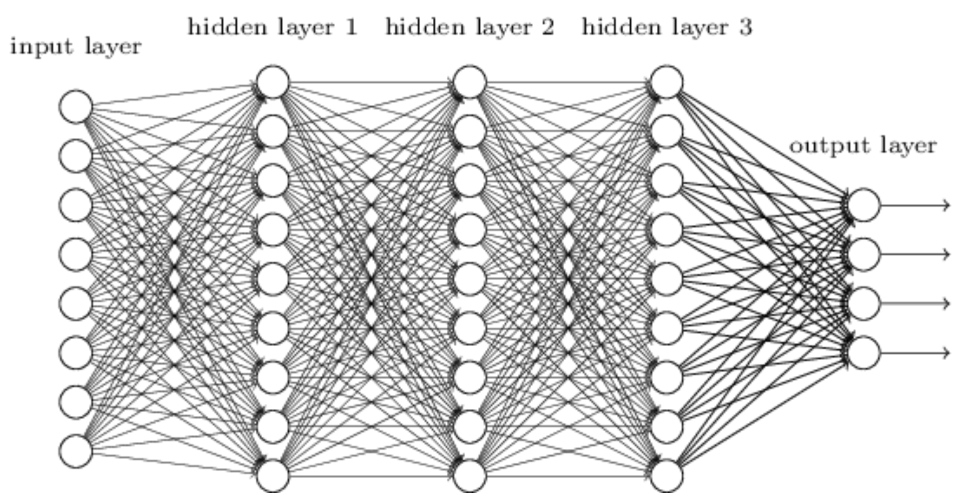
Data $\mathbf{x}$

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

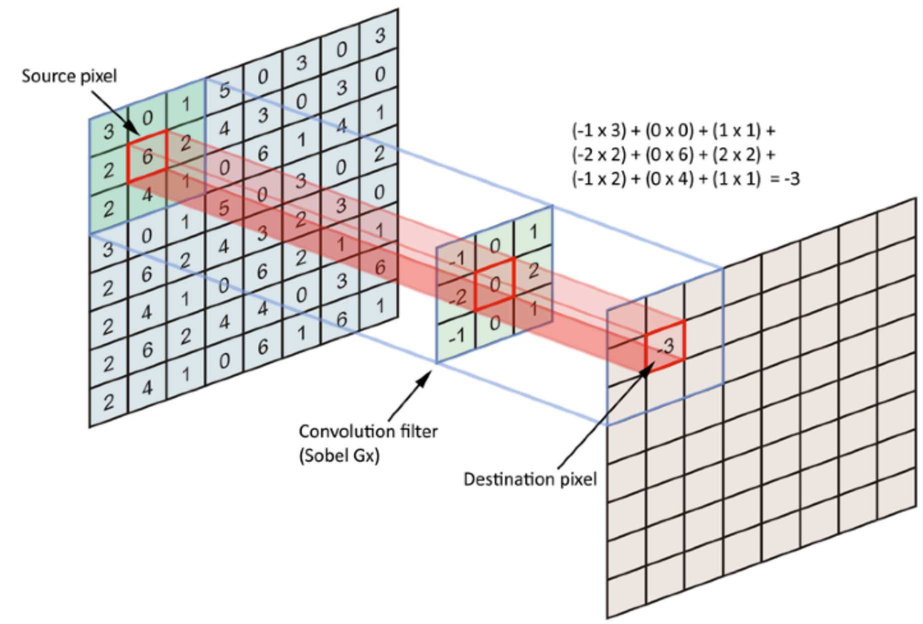
$$\theta^* = \text{argmin}_\theta \, \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \left[\mathcal{L}(f_\theta(\mathbf{x}), \mathbf{x}) \right]$$

Data Representation

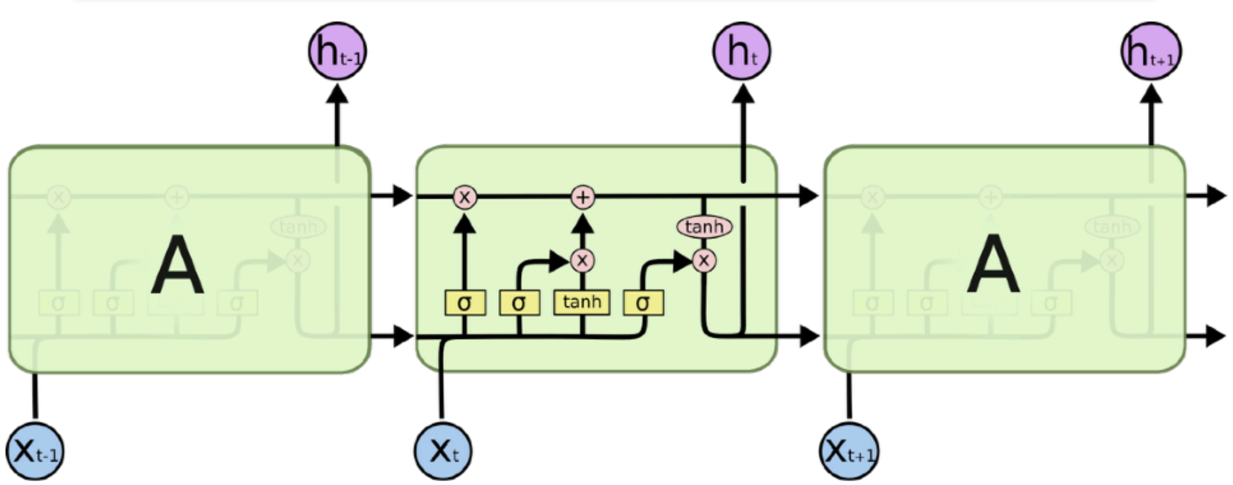
High-level/no structure: Fully Connected



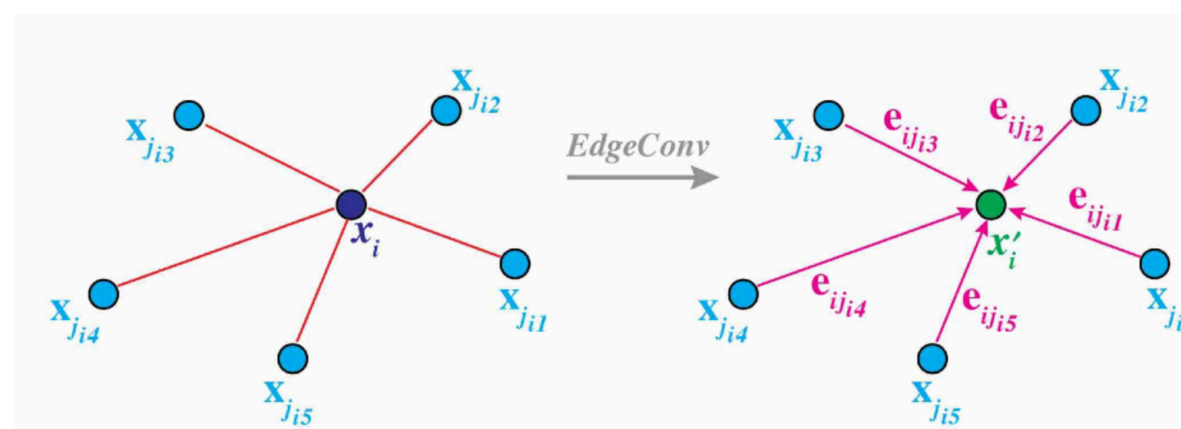
Regular grid: Convolution



Sequence/Time Series: Recurrent



Point cloud: Sets & Graphs



General Strategy

Loss function $\mathcal{L}$ ✓

Neural network f_θ ✓

Parameters θ ✓

Opt. Parameters θ^* ✓

Data $\mathbf{x}$ ✓

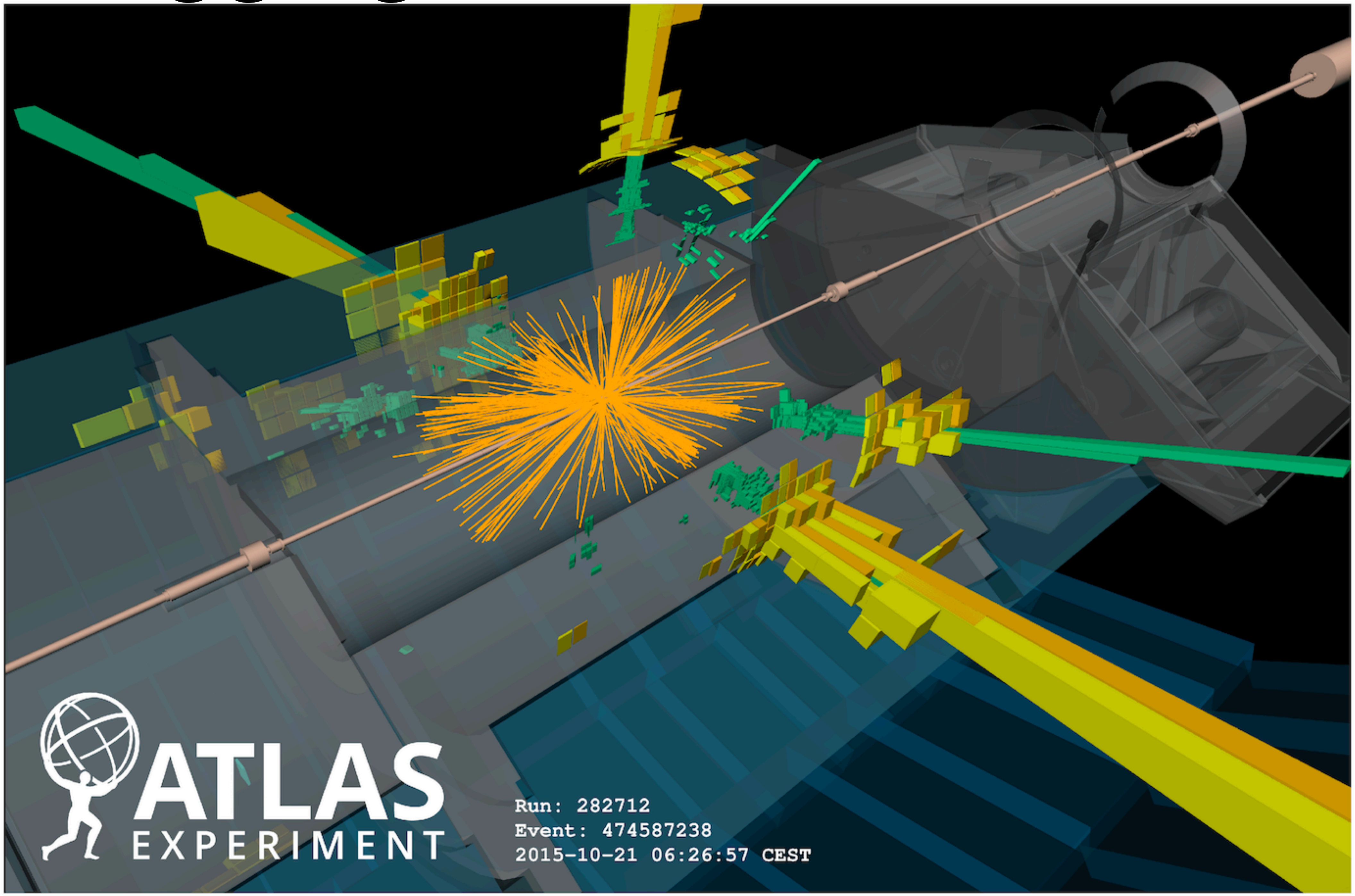
Data distribution $p(\mathbf{x})$ ✓

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

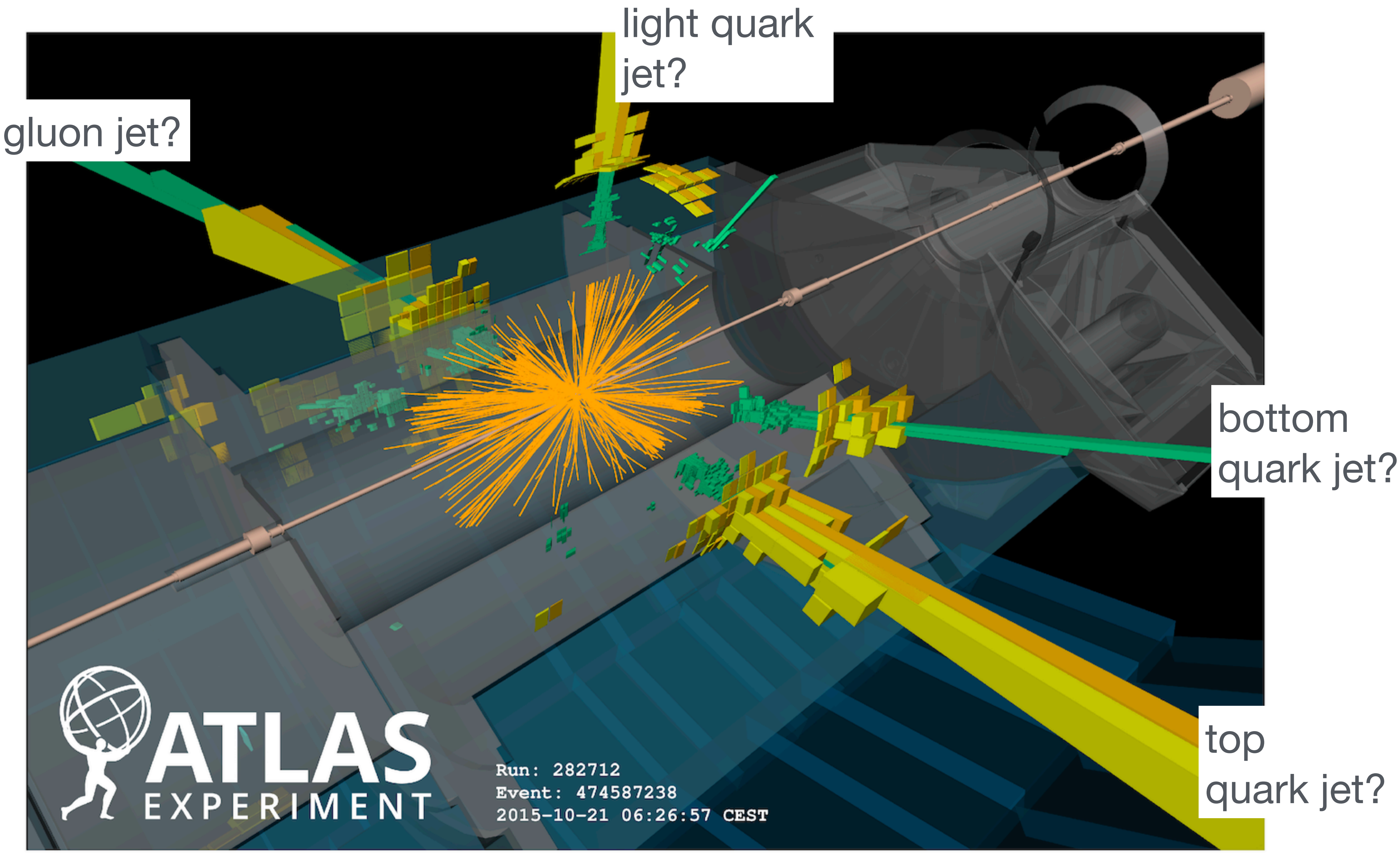
Summary so far

- Key idea of deep learning:
  - Define a optimisation target and use an expressive function (neural network) to minimise it using gradient descent
- Let's look at a typical problem in particle physics…

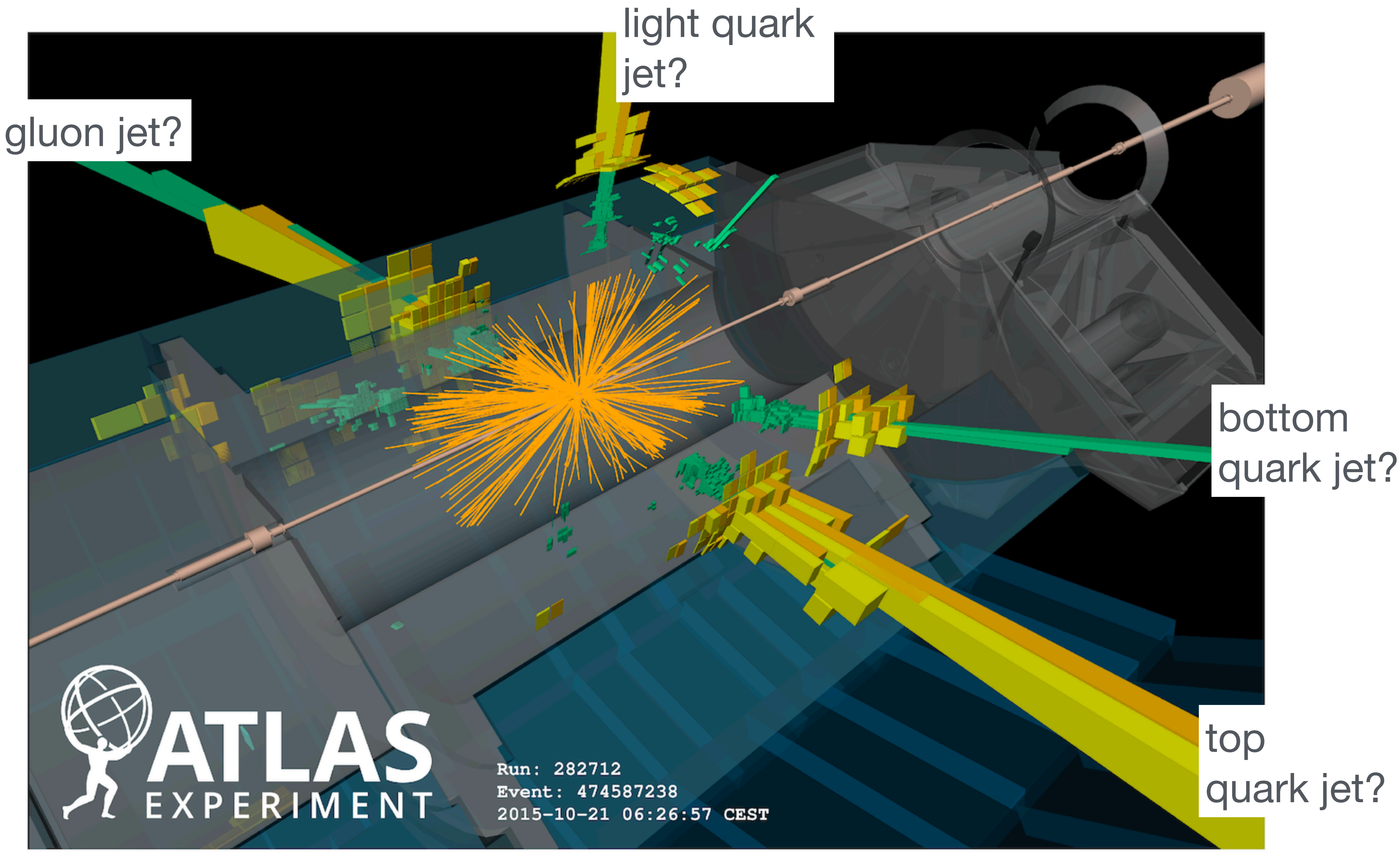
Jet Tagging



A jet is a
collimated shower of particles in the detector



We want to know
which particle produced a jet



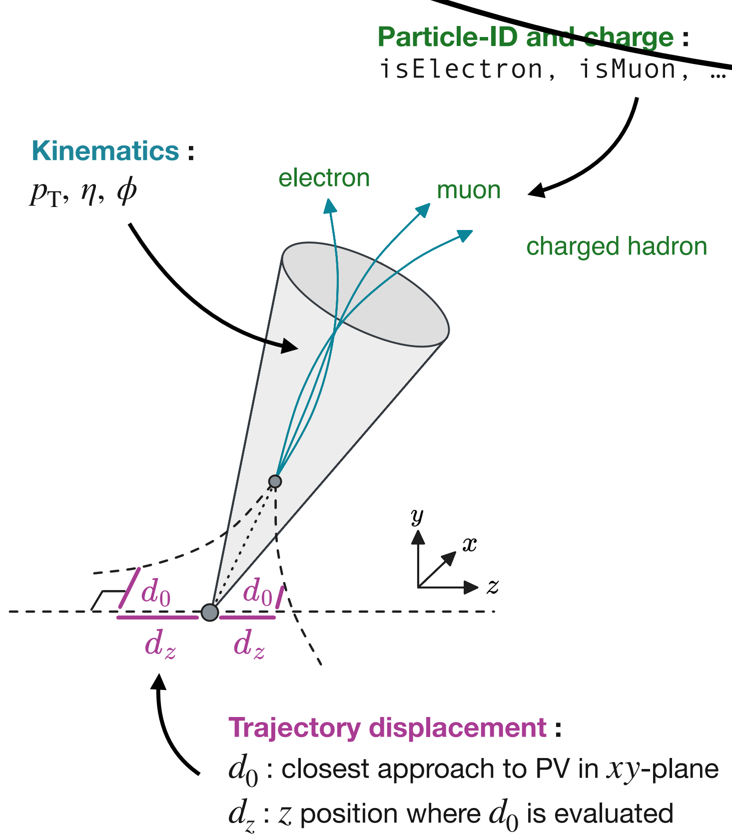
How to build ML algorithms for complex, heterogenous data?

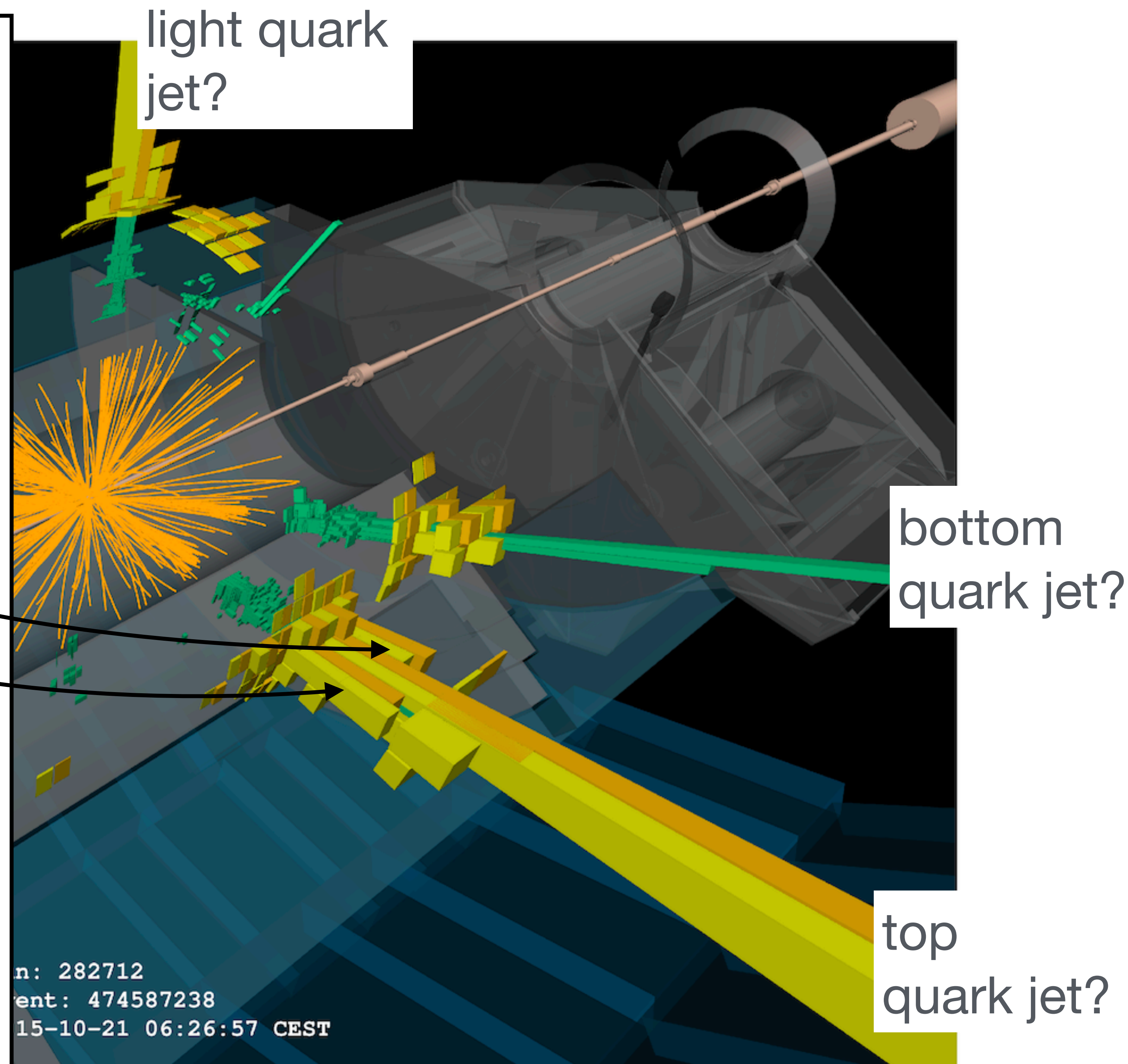
Data most naturally viewed as point cloud:

Each input (e.g. jet, event, ..) is a set of k-dimensional vectors (individual particles, hits, ..)

$$J_i = \{\vec{p_1}, \ldots, \vec{p_n}\}$$

Example per-particle features

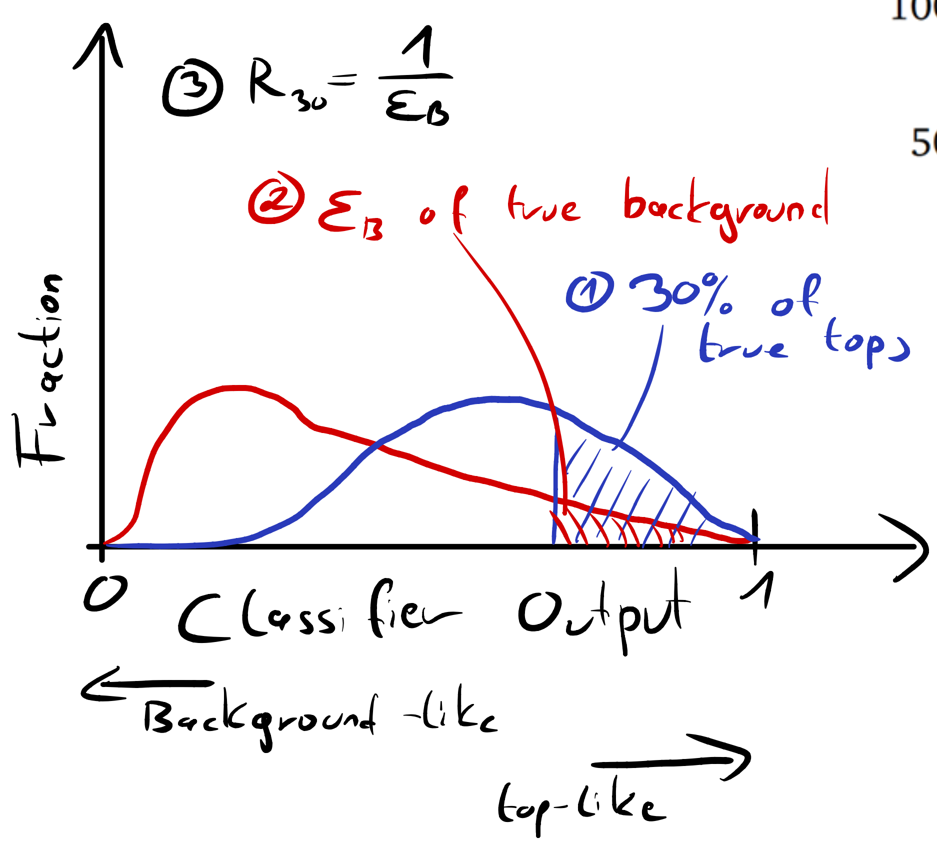


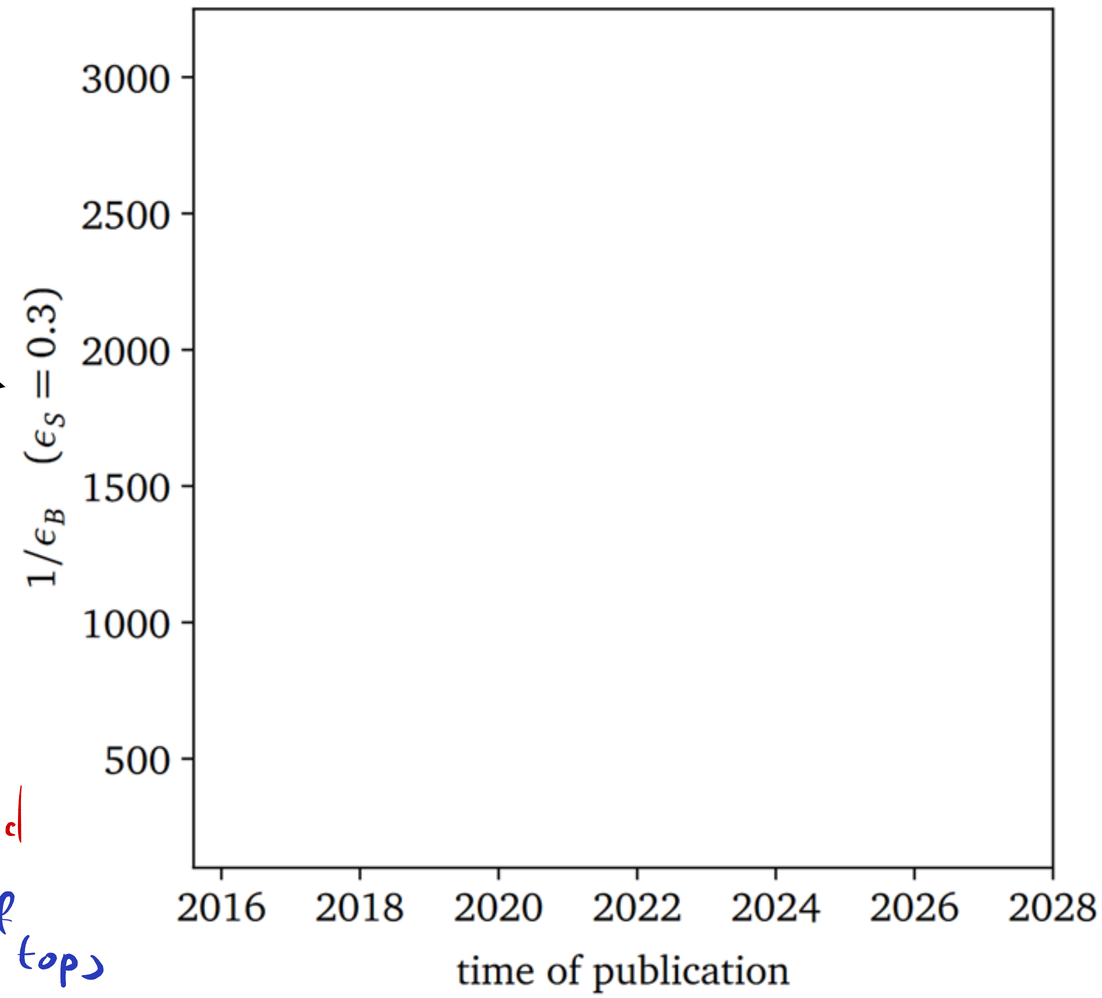


1810.05165; **GK** et al 2312.00123

Top Quark Tagging

Public simulated* benchmark dataset





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.

* techniques are also applied to collider data

GK, Plehn, et al 1902.09914

Landscape Dataset

- Open dataset for the development of better tagging algorithms for particle physics
- 2 million simulated examples
- Input: momentum sorted list of 200 particles/jet with 3 features/particle (p_X, p_Y, p_Z)
- Perfect class labels:
  top jet or light quark/gluon jet
- Supervised learning problem

SciPost Physics | Submission

arXiv:1902.09914v3 [hep-ph] 23 Jul 2019

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

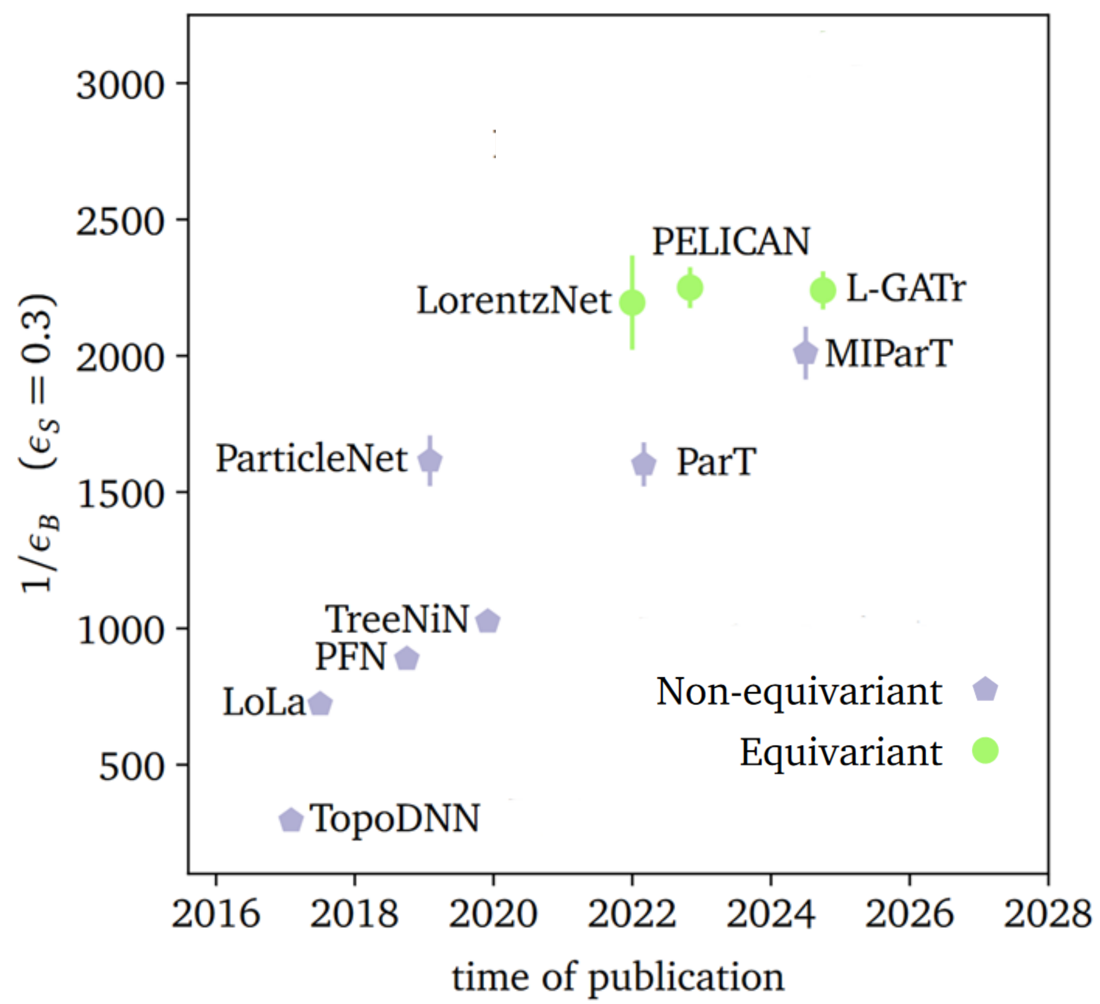
July 24, 2019

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.

GK, Plehn, et al 1902.09914

Top Quark Tagging

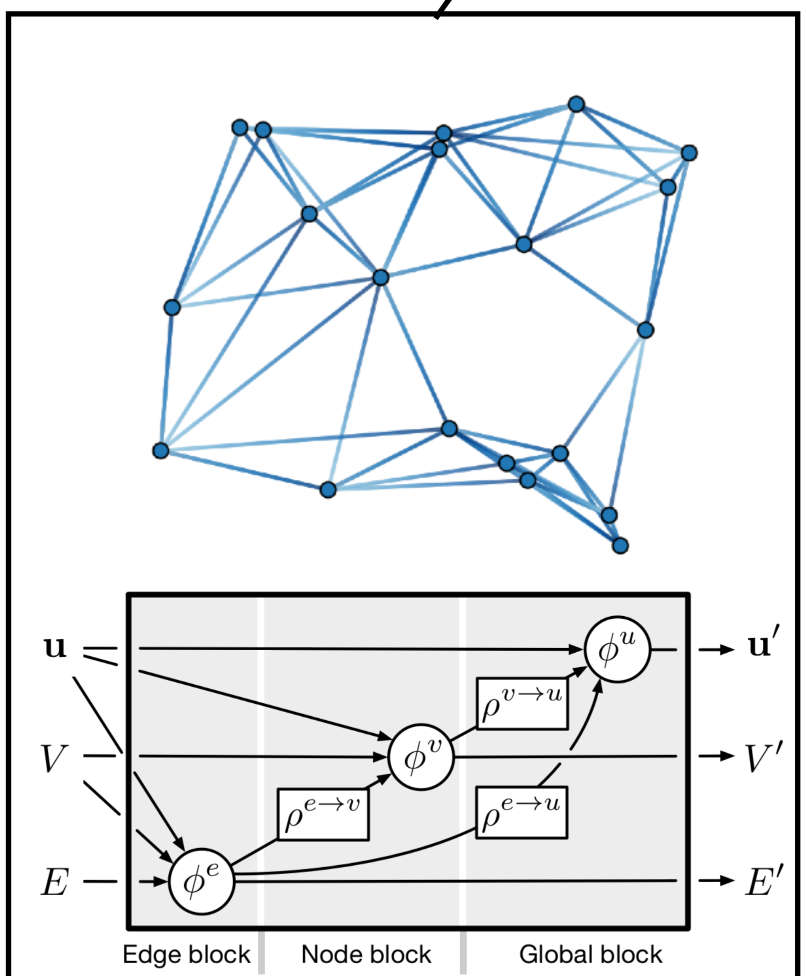


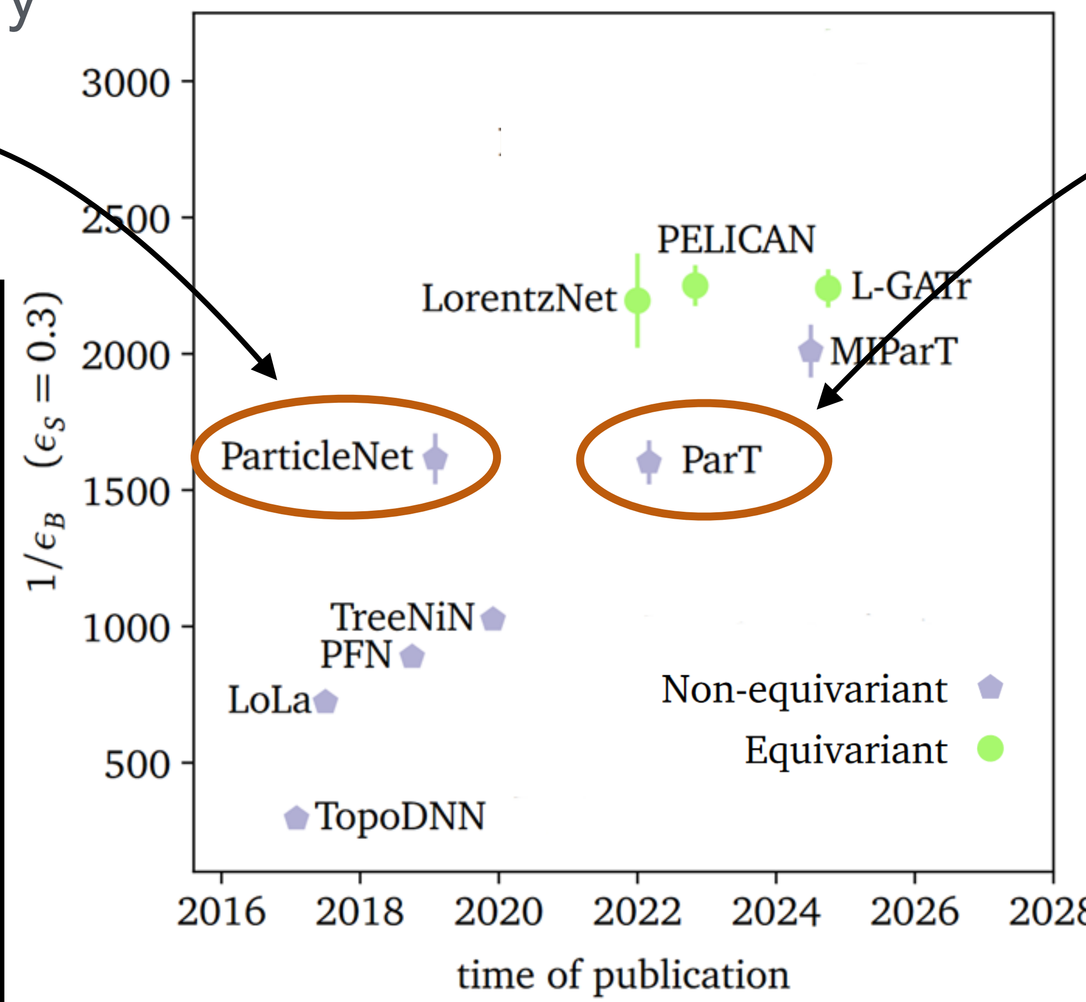
(*Subset of methods, graph from
V Breso at ML4Jets)

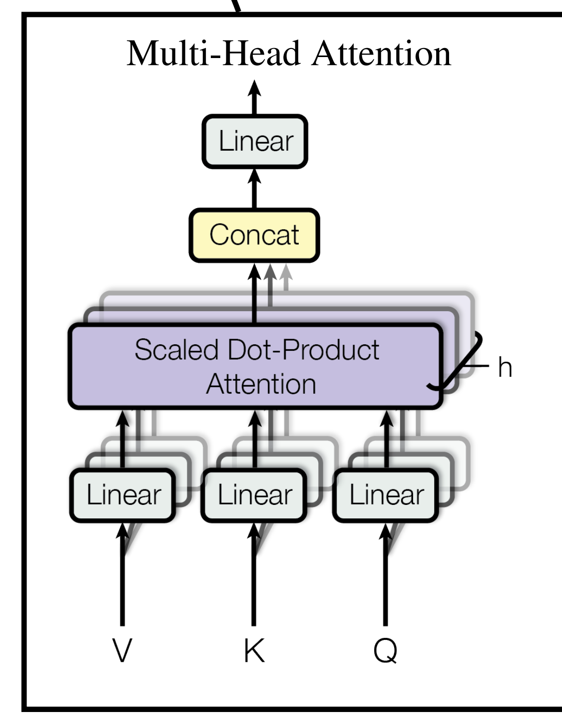
Top Quark Tagging

View data as
graph based on geometry
& use message passing

Learn relations from data
using attention



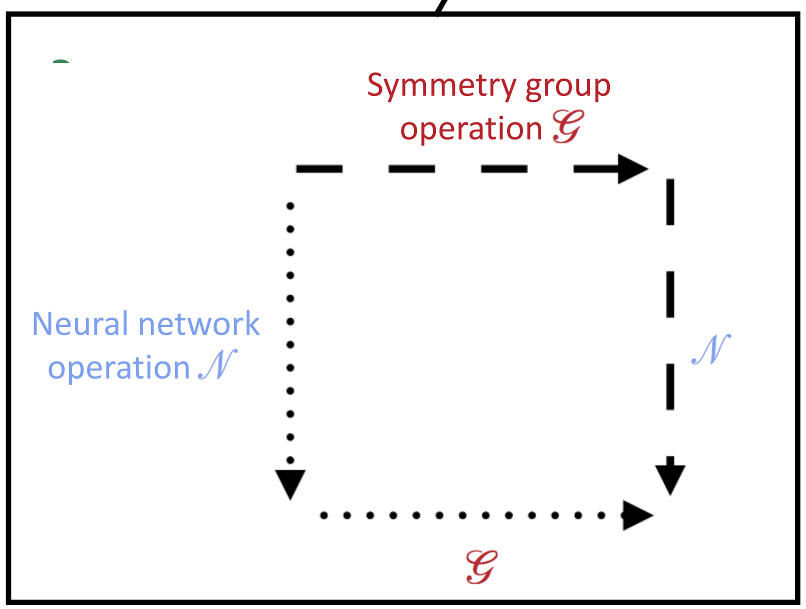


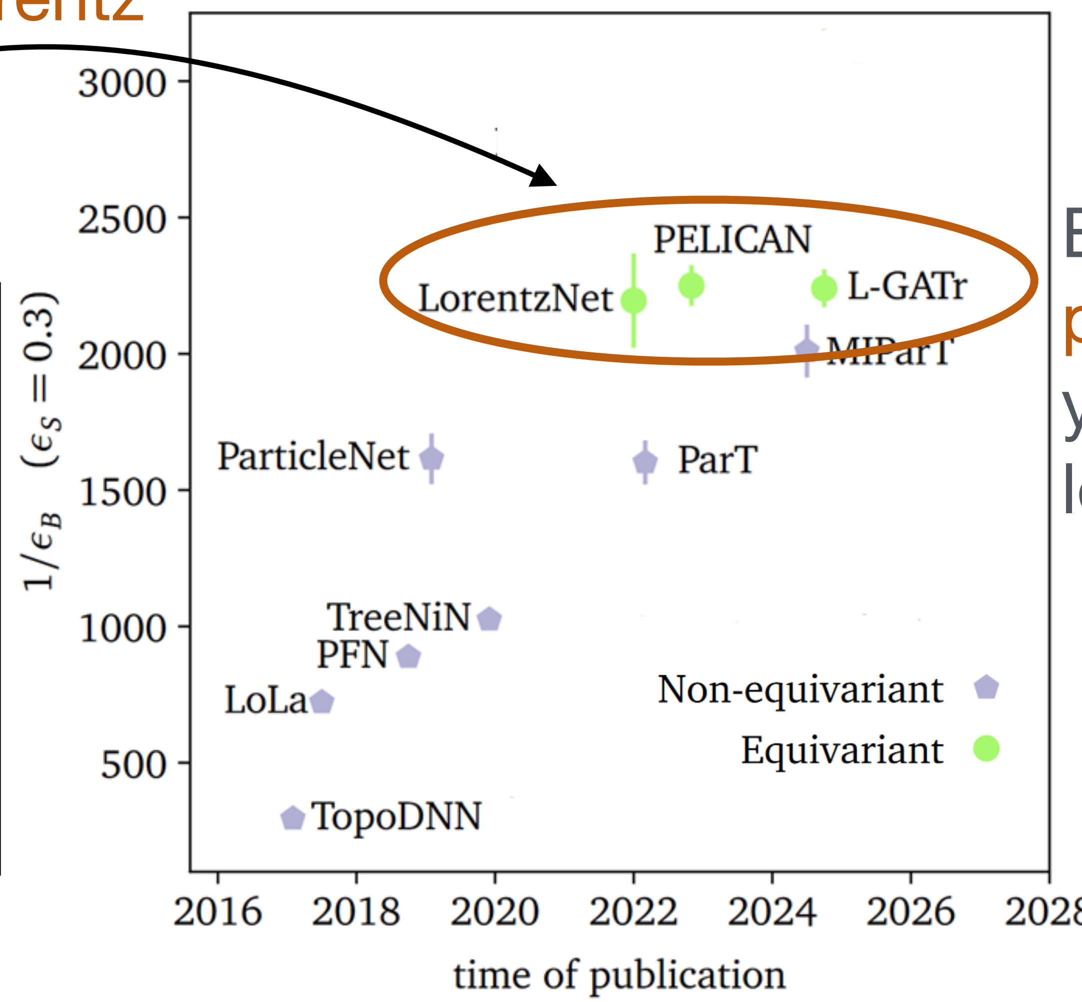


Qu, Gouskos 1902.08570; Qu et al 2202.03772; Gong et al 2201.08187; Shlomi et al 2007.13681;
Bogatskiy et al 2211.00454;

Top Quark Tagging

Constrain functions to be equivariant under the Lorentz group

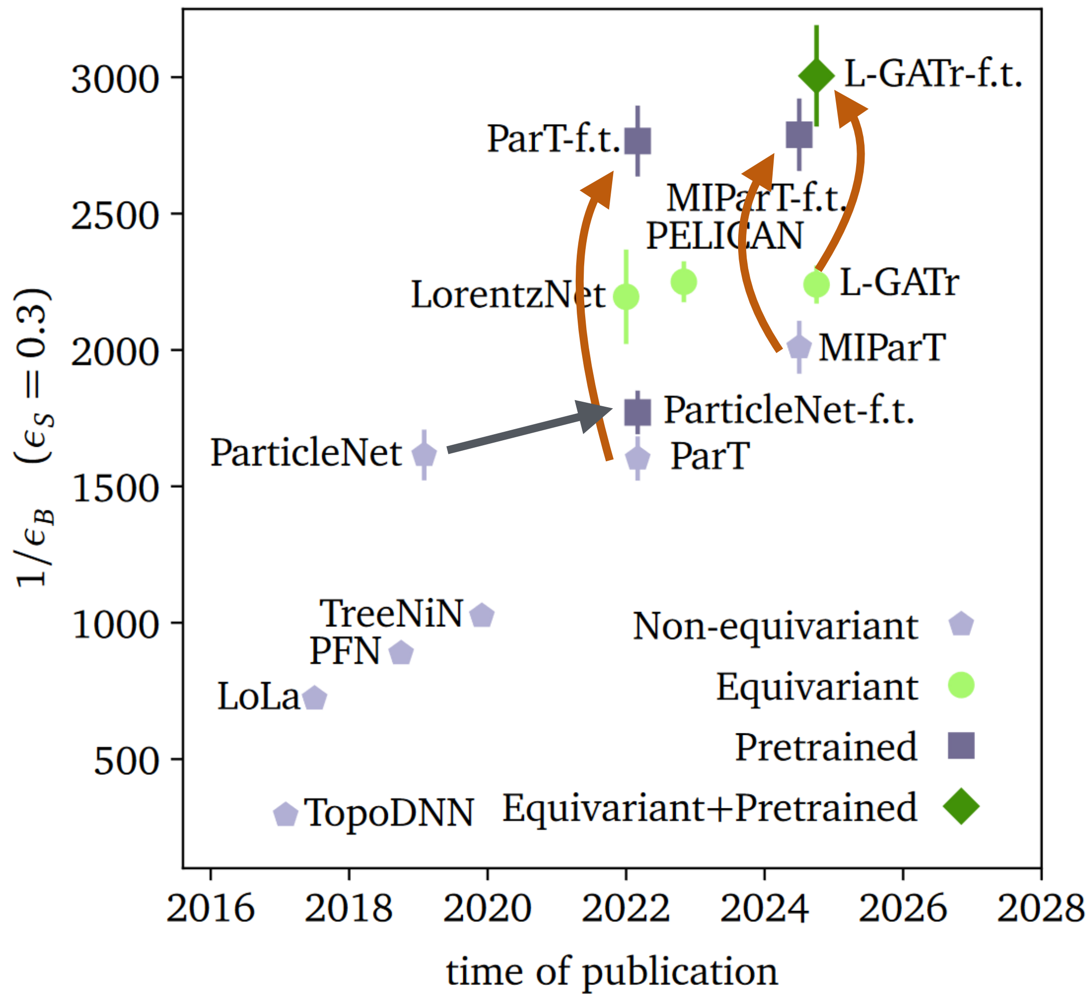




Explicitly injecting physics knowledge yields more efficient learning

Qu, Gouskos 1902.08570; Qu et al 2202.03772; Gong et al 2201.08187; Shlomi et al 2007.13681; Bogatskiy et al 2211.00454; Brehmer et al 2411.00446

Top Quark Tagging



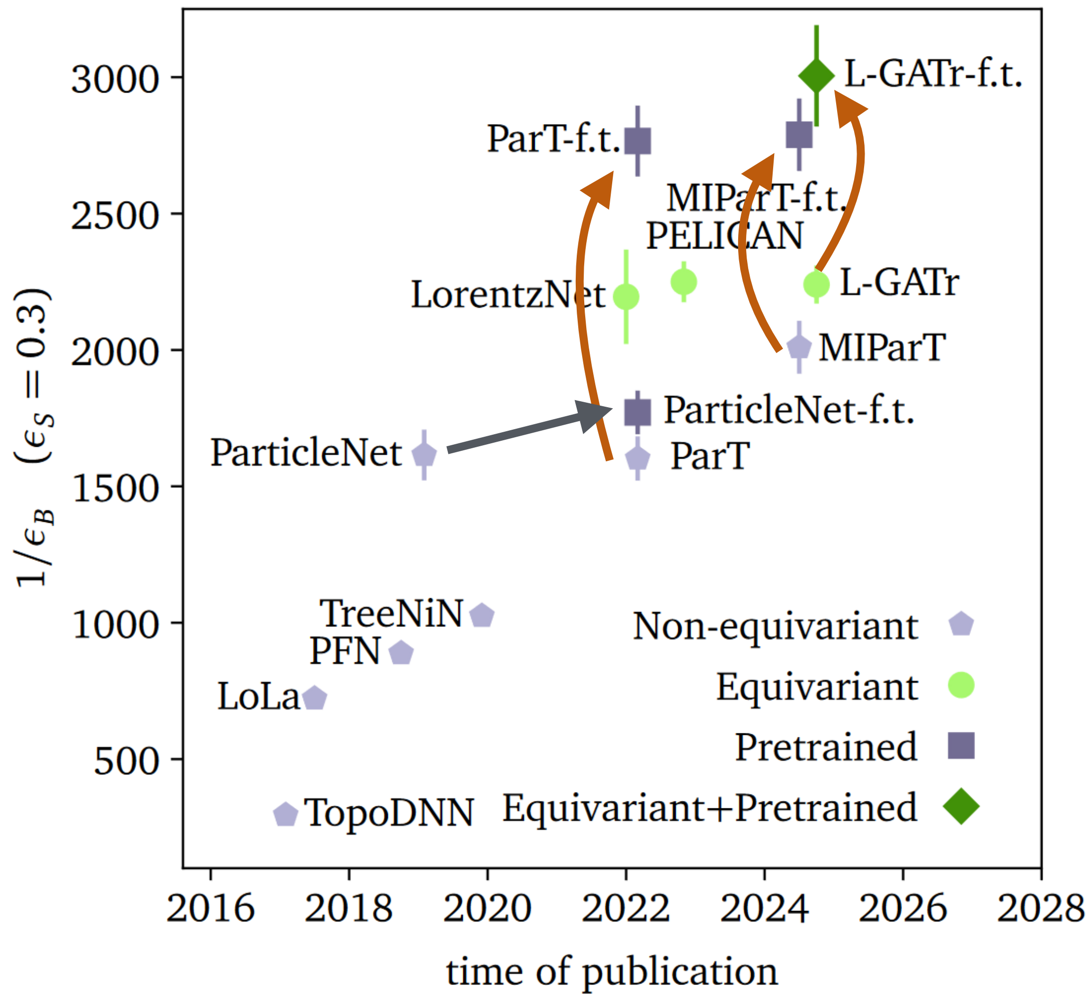
Fine tuning:
Pre-train on one dataset and recycle weights for training on new data

Yields substantial boost in performance (for transformer-based models)

Can combine with equivariance

Top Quark Tagging

- Usefulness of open benchmark data
- Good representation of data pays off
- Transformers + physics rules
- Boost by fine-tuning across datsets



Important issues on application side

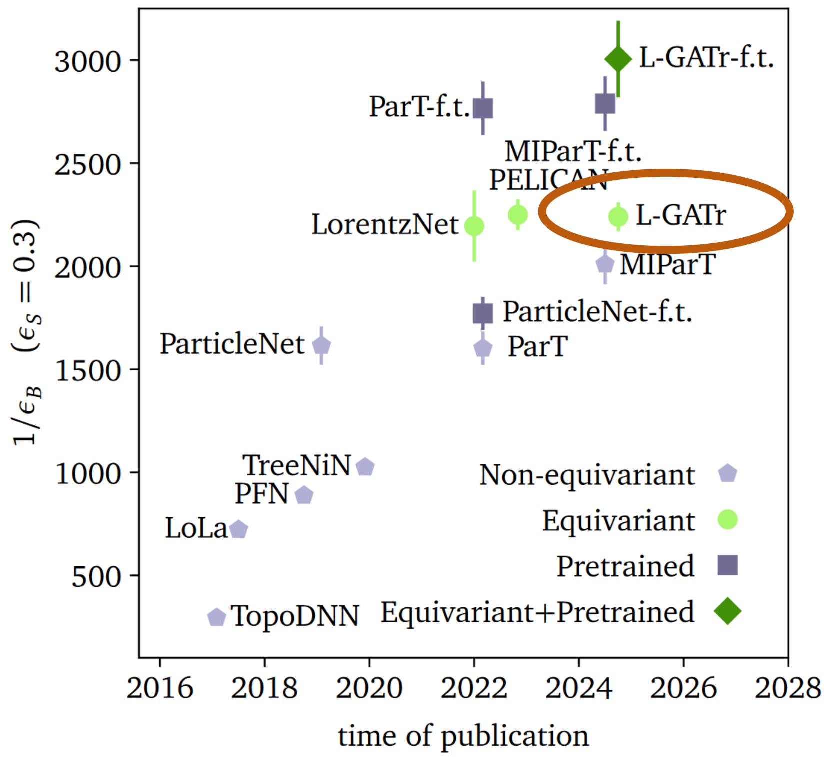
- Domain shift
- Calibrateable
- Compute cost

Modern tagging algorithms are widely used in searches for new particles

End of Part I.

Bonus Material

Symmetries



Reminder: Space-time described by Lorentz group (rotations, boosts)

$$(t, x, y, z) \mapsto t^2 - x^2 - y^2 - z^2$$

Equivariance: $\text{L-GATr}\big(\Lambda(x)\big) = \Lambda\big(\text{L-GATr}(x)\big)$

Multivector: $x = x^S\, 1 + x^V_\mu\, \gamma^\mu + x^B_{\mu\nu}\, \sigma^{\mu\nu} + x^A_\mu\, \gamma^\mu \gamma^5 + x^P\, \gamma^5$

Scalar

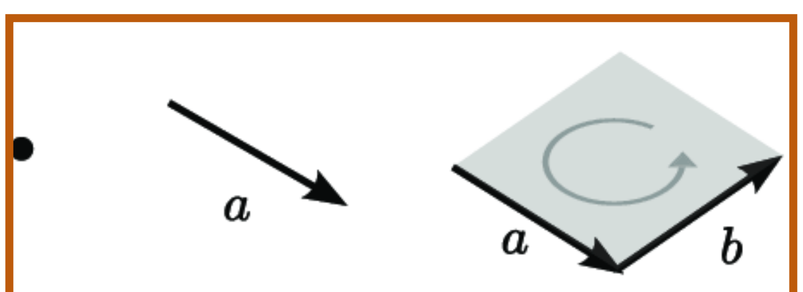
Vector

Bi-Vector

Axial-Vector

Pseudo-Scalar

"Grades"
with well defined behaviour under Lorentz transformations



$$\gamma^0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}, \quad \gamma^1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix},$$

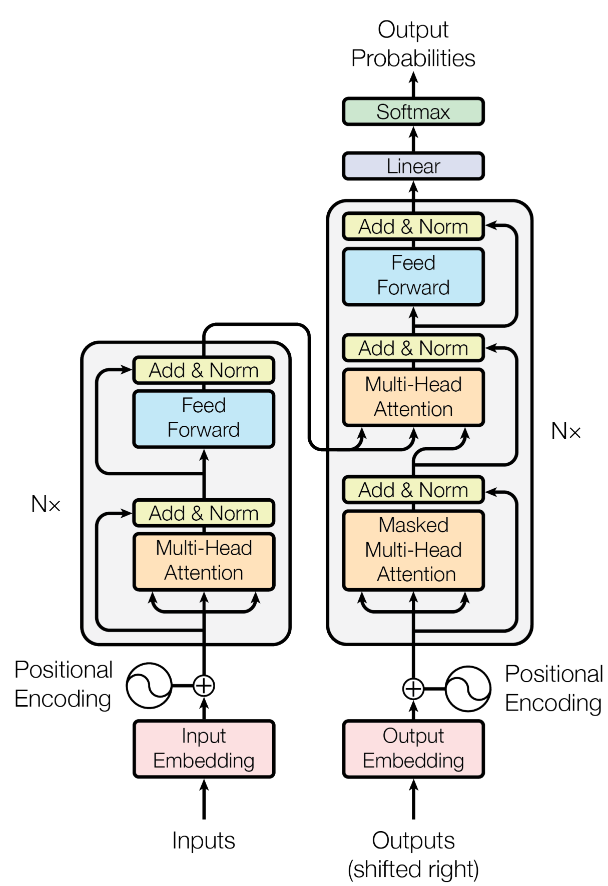
$$\gamma^2 = \begin{pmatrix} 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \\ 0 & i & 0 & 0 \\ -i & 0 & 0 & 0 \end{pmatrix}, \quad \gamma^3 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

2411.00446

Symmetries

Sum over grades

Layer type	Transformer	L-GATr
Linear(x)	$vx + w$	$\sum_{k=0}^{4} v_k \langle x \rangle_k$
Attention$(q,k,v)_{ic}$	$\sum_{j=1}^{n_t} \mathrm{Softmax}_j \left(\sum_{c'=1}^{n_c} \frac{q_{ic'} k_{jc'}}{\sqrt{n_c}} \right) v_{jc}$	$\sum_{j=1}^{n_t} \mathrm{Softmax}_j \left(\sum_{c'=1}^{n_c} \frac{\langle q_{ic'}, k_{jc'} \rangle}{\sqrt{16 n_c}} \right) v_{jc}$
LayerNorm(x)	$x \left[\frac{1}{n_c} \sum_{c=1}^{n_c} x_c^2 + \epsilon \right]^{-1/2}$	$x \left[\frac{1}{n_c} \sum_{c=1}^{n_c} \sum_{k=0}^{4} \left\lvert \left\langle \langle x_c \rangle_k, \langle x_c \rangle_k \right\rangle \right\rvert + \epsilon \right]^{-1/2}$
Activation(x)	GELU(x)	GELU$(\langle x \rangle_0)x$
GP(x,y)	–	xy



Norm of inner product by grade

Inner product (requires some work but can compute Algebra-rules in advances)

$$x_\mu x^\mu = \eta_{\mu\nu} x^\mu x^\nu = (ct)^2 - \mathbf{x} \cdot \mathbf{x} \stackrel{\text{def}}{=} (c\tau)^2$$

$$\eta^{\mu\nu} = \eta_{\mu\nu} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}.$$

2411.00446

Classification

Cross Entropy?

$$\boxed{-\frac{1}{N}\log(L) = -\frac{1}{N}\sum_{k}^{N}[y_k \cdot \log(\hat{y}(x_k)) \ + (1-y_k)\cdot\log(1-\hat{y}(x_k))]}$$

- Note:

$$y_k \cdot \log\big(\hat{y}(x_k)\big) + (1-y_k)\cdot\log\big(1-\hat{y}(x_k)\big) = \sum_i q_i \log(p_i)$$

- Now remember entropy

$$\text{entropy} = \int p(x)\log\big(p(x)\big)dx$$

$$\text{cross-entropy} = \int q(x)\log\big(p(x)\big)dx$$

Gibbs inequality:
Cross entropy is minimal if
p=q

For multiple classes: Categorical cross entropy

Multiple classes

- For more than 2 classes it is usually a good idea to use **One-Hot-Encoding**

  -> For m classes the output is a m-dimensional vector with components

  $\hat{y}_c(x_k) \in [0,1]$

- One neuron per class in output layer
- All outputs should sum to one
- Loss function: Extension of cross entropy to multiple classes: **Categorical cross entropy**

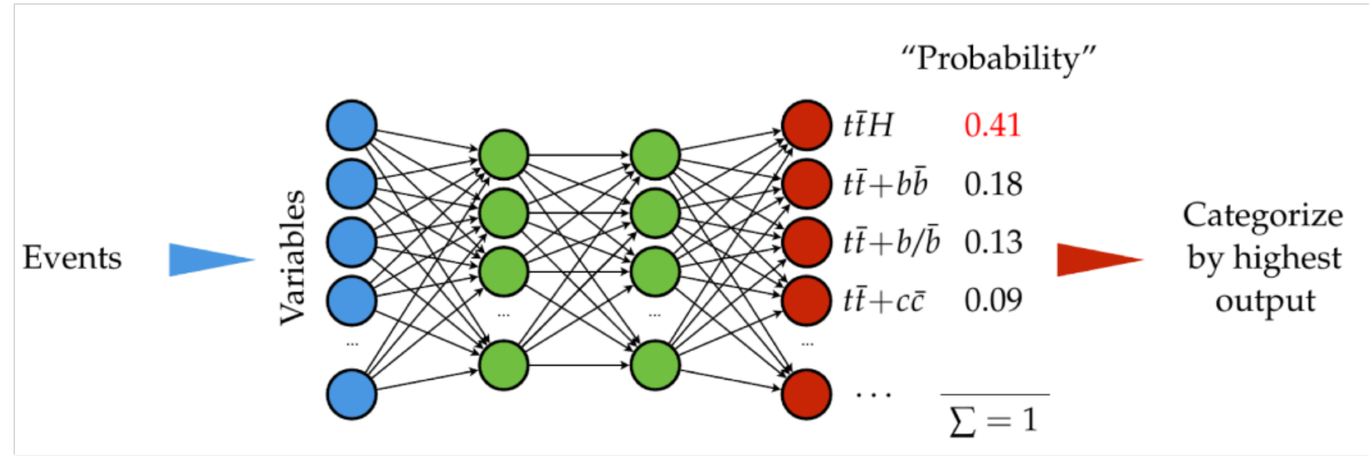
Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

$\rightarrow$

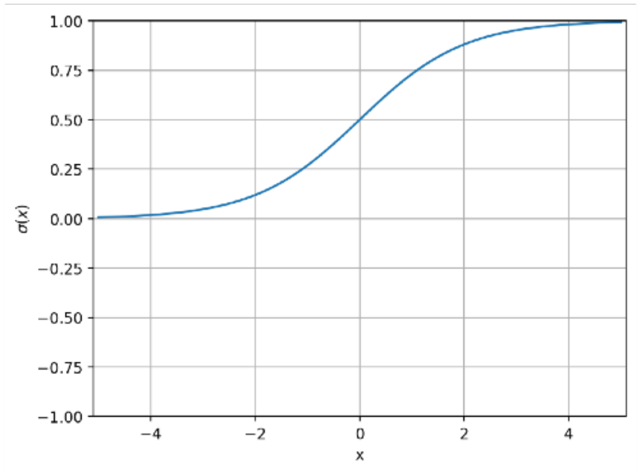
One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50



https://medium.com/@michaeldelsole/

Activation functions for Classification



- Need activations in the output layer that allow probability interpretation:
  - $\hat{y}(x_k) \in [0,1]$
  - $\sum_{classes} \hat{y}(x_k) = 1$

- Binary classification:
  - Only 1 output node needed
  - Activation: Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

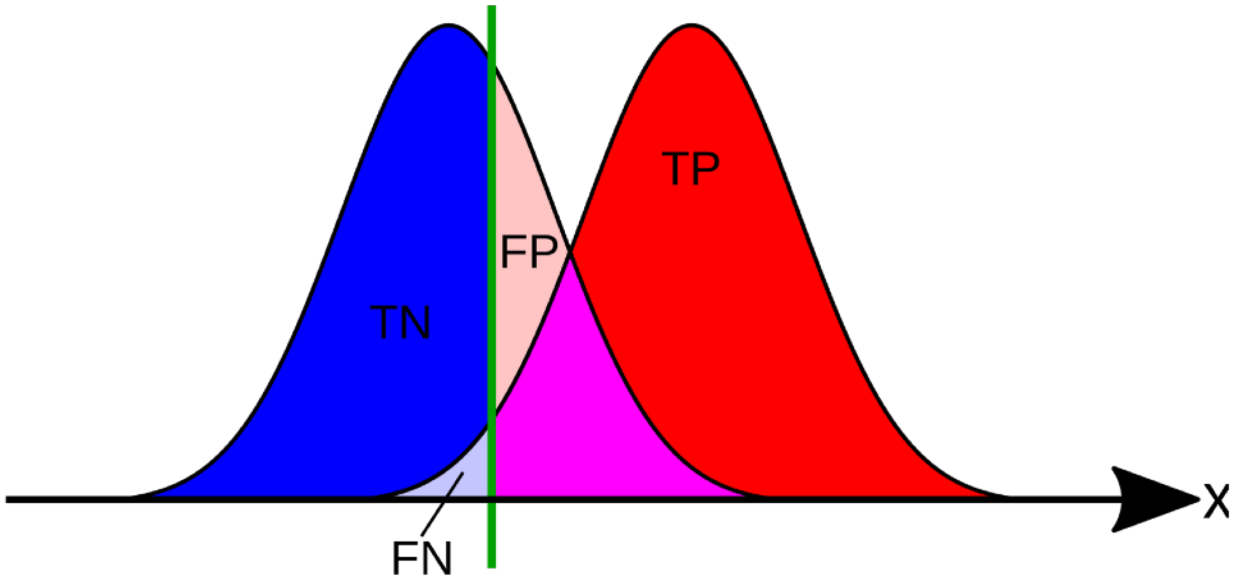
- Multiclassification:
  - One output node per class i
  - Softmax activation:

$$\sigma(x) = \frac{e^{x_i}}{\sum_j^{classes} e^{x_j}}$$

Binary Classification Outputs

- The NN outputs a score for belonging to class „signal“ (high value) or class „background“ (low value)

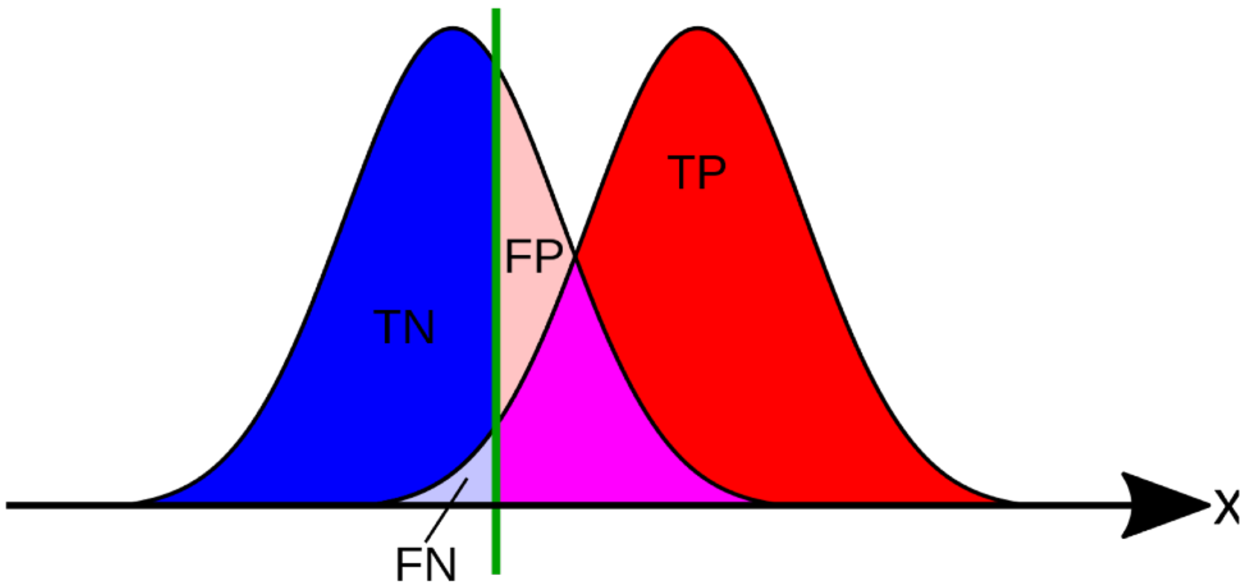
- Often you need to make a decision based on this score -> Choose a threshold



- Key quantities:
  - True Positives (TP)
  - False Positives (FP)
  - True Negatives (TN)
  - False Negatives (FN)

- Purity/Precision = TP/(TP+FP): How pure is the sample predicted as signal?
- True Positive Rate/Sensitivity/Efficiency/Recall = TP/(TP+FN): What fraction of the signal is classified correctly?
- False Positive Rate = FP/(FP +TN)
- Accuracy = (TP+TN)/(TP+FP+TN+FN): What fraction of all data is classified correctly?

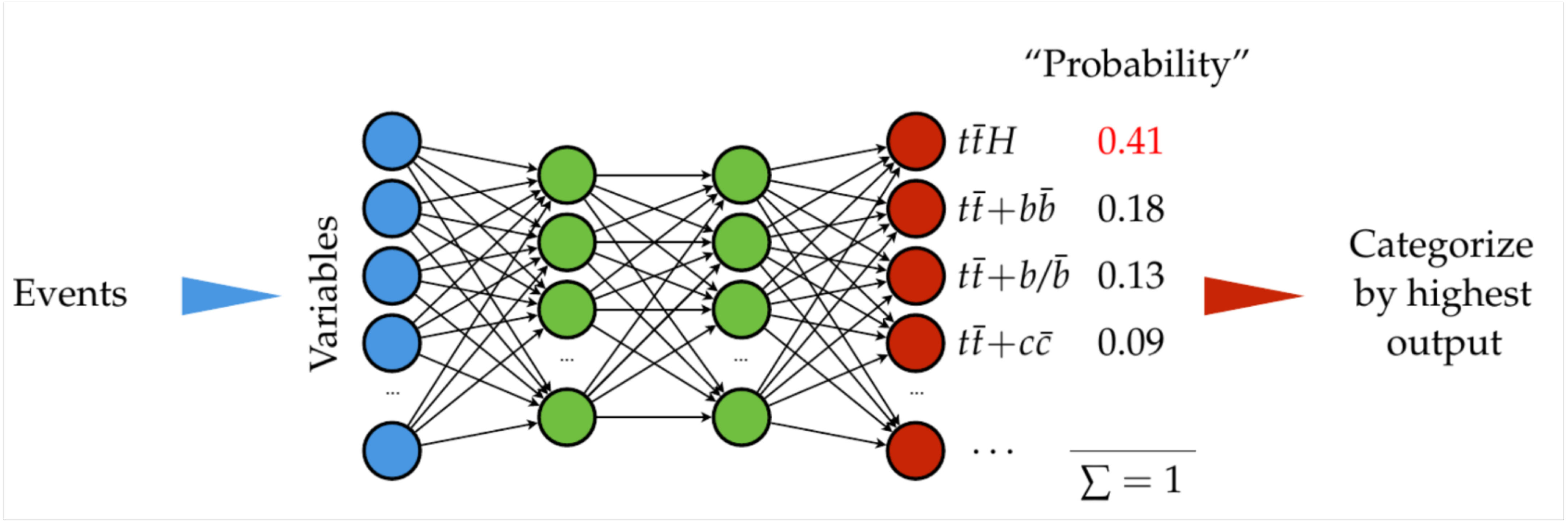
Choosing a threshold



- Which threshold to choose?
- Depends on your problem:
  - Think about what questions you want to answer
  - The cost of being wrong
- Do you need a high purity? ("We need to be sure that all animals we tag as harmless are actually harmless!")
- Do you need a high Efficiency? („We need to detect as many positive Covid cases as possible")

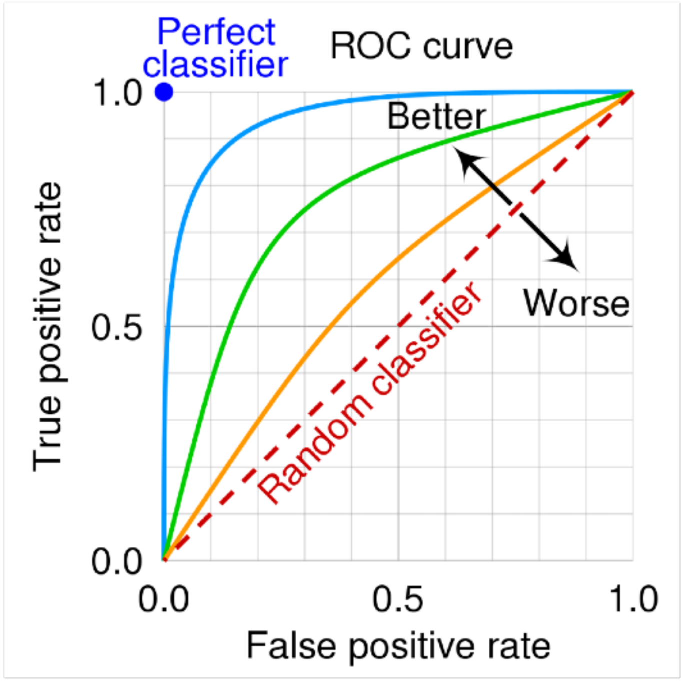
Multiclassification

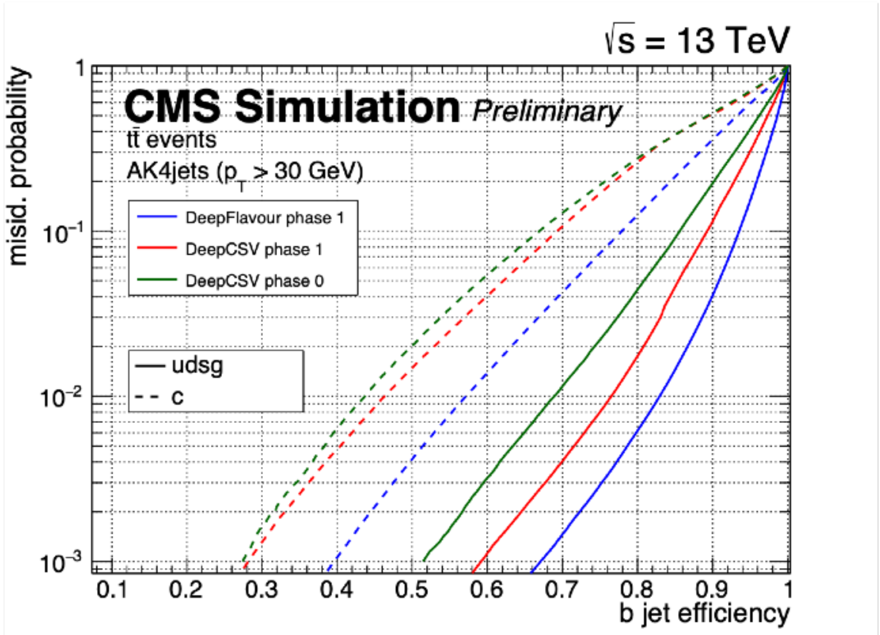
- In multiclassification we have one output neuron for each class
- If we use softmax as final activation function, the values of each output neuron sum to 1 and can be interpreted as probability
- Common approach: Classify a sample according to the neuron with the highest output value



Receiver Operator Characteristic (ROC)

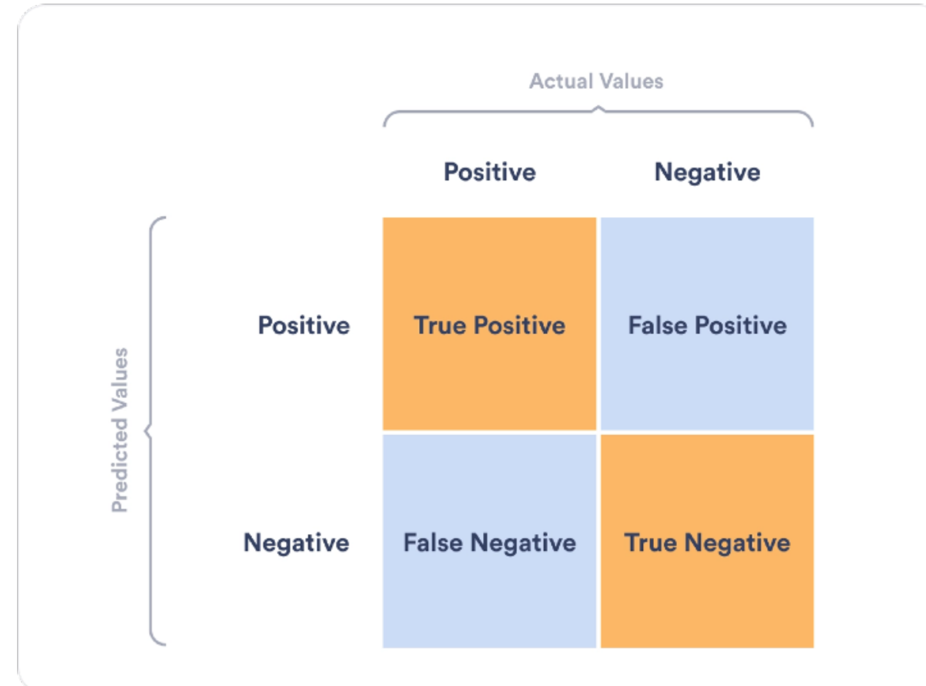
- Overall performance of a classifier is usually visualized using the ROC curve:
  - Scan all possible thresholds and evaluate True Positive Rate and False Positive Rate
  - Plot one as a function of the other
- Area under the curve (AUC) often quoted as performance metric
- ROCs can aide in choosing a classifier and threshold
- Note: Sometimes ROCs of different classifiers cross each other -> they are better at different TPR regions

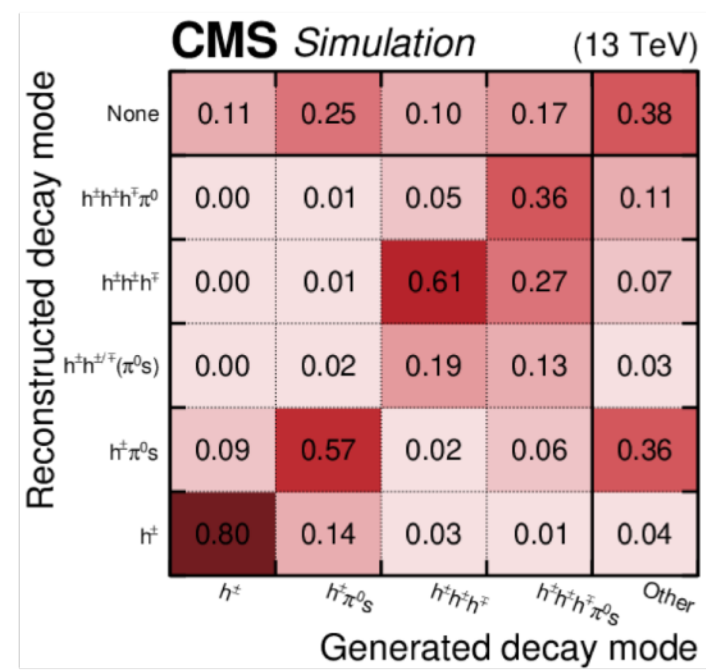


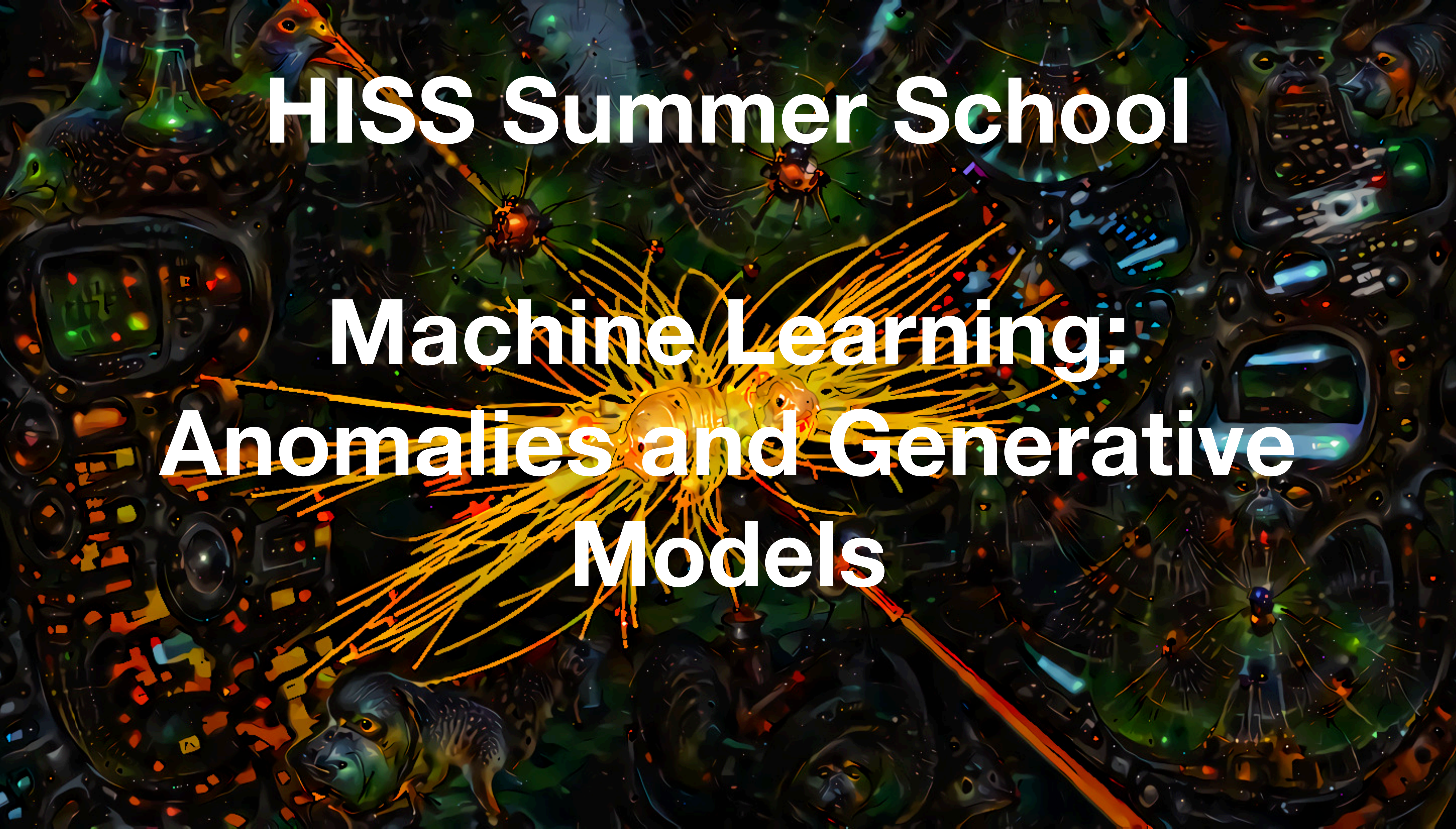


Confusion matrices

- With a chosen threshold we can plot confusion (migration) matrices
- Shows migration from one class to another
- Very usefull for multiclassification:
  - Which background class looks most signal like?
  - Which classes are difficult to separate?
- Can be normalized to:
  - True classes (efficiency matrix)
  - Predicted class (purity matrix)







HISS Summer School

Machine Learning: Anomalies and Generative Models

Prof. Gregor Kasieczka
gregor.kasieczka@uni-hamburg.de

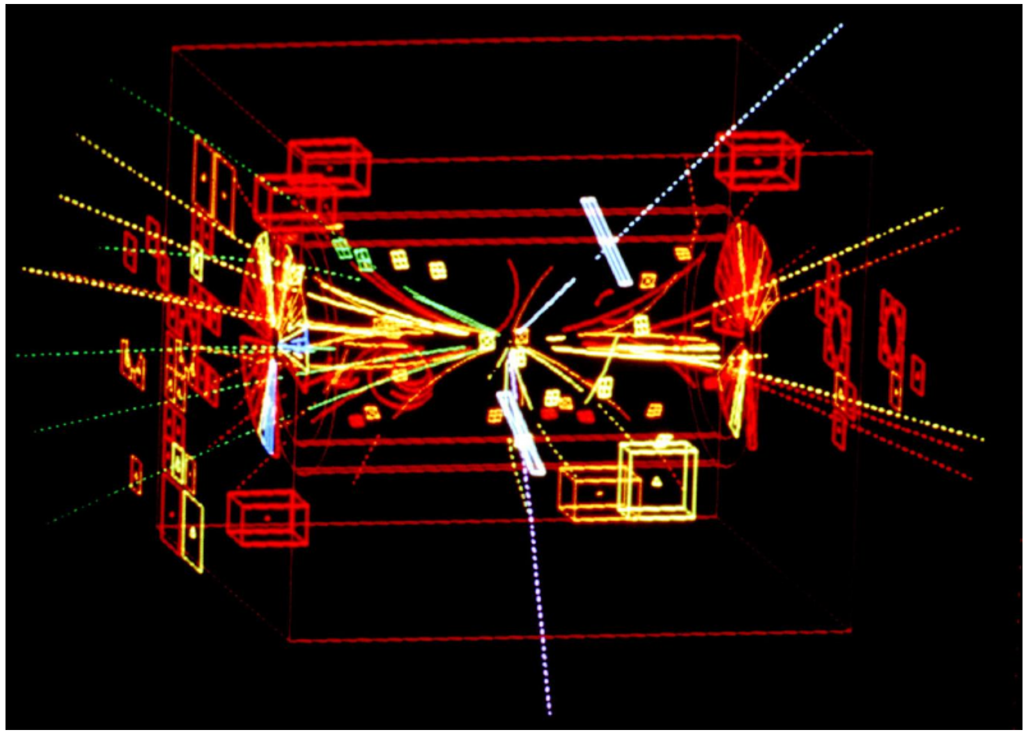
Plan

- Lecture 1:
    - Basics of Neural Networks
    - Classification example: Top Tagging
- Lecture 2:
    - Anomaly Detection
    - Generative Models

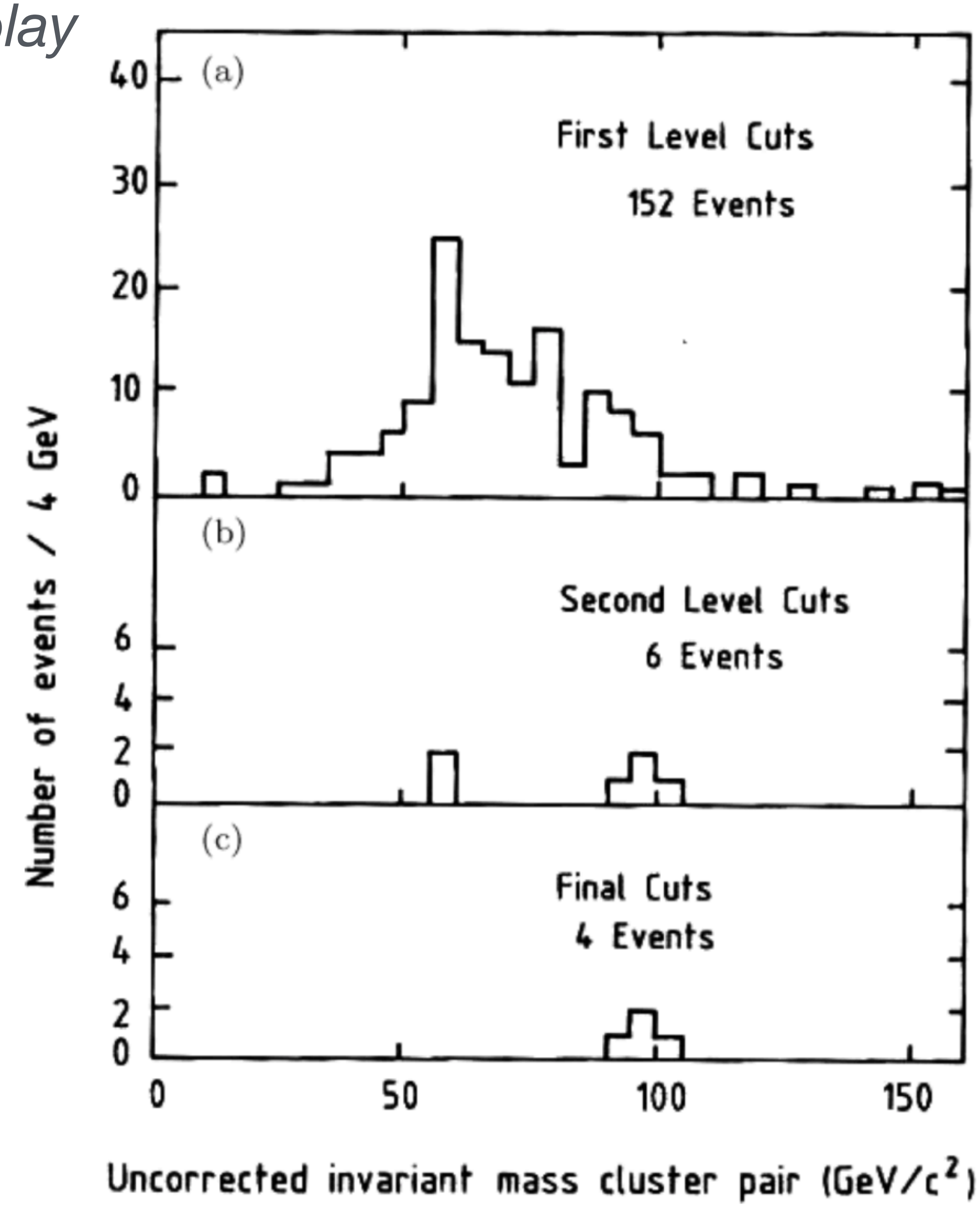
Plan

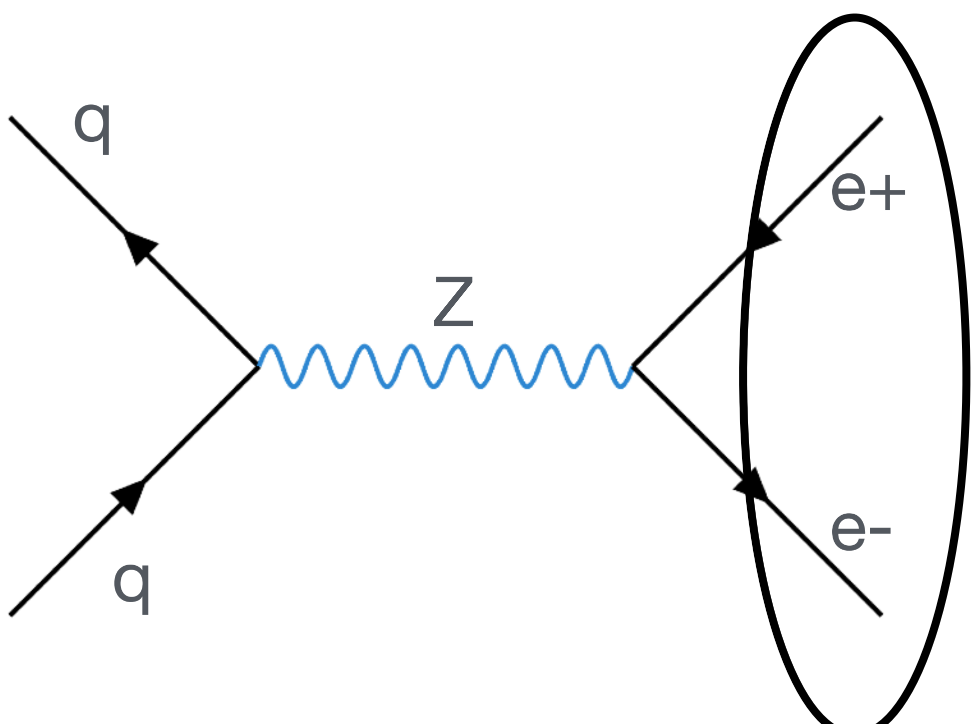
- Lecture 1:
  - Basics of Neural Networks
  - Classification example: Top Tagging
- Lecture 2:
  - **Anomaly Detection**
  - Generative Models

How to discover a new particle?



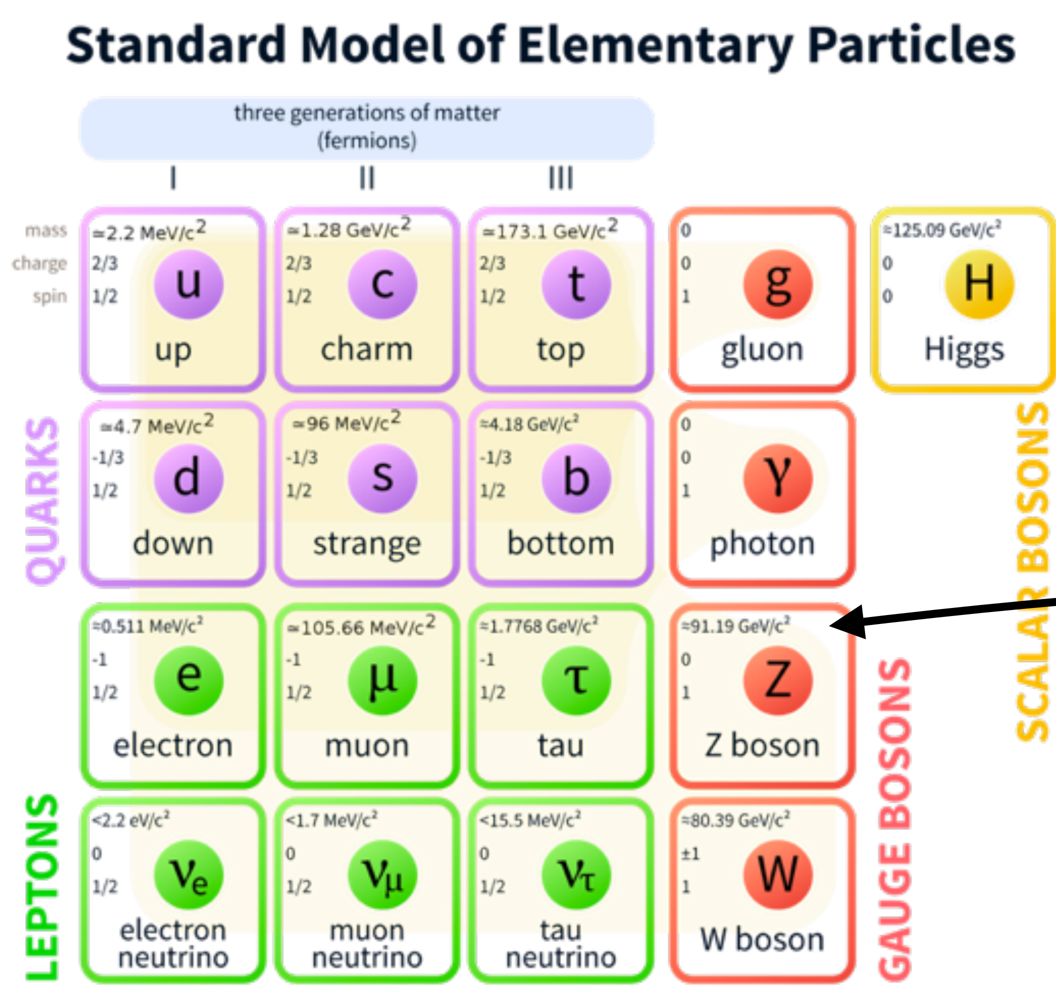
Event display from UA1 from 1983

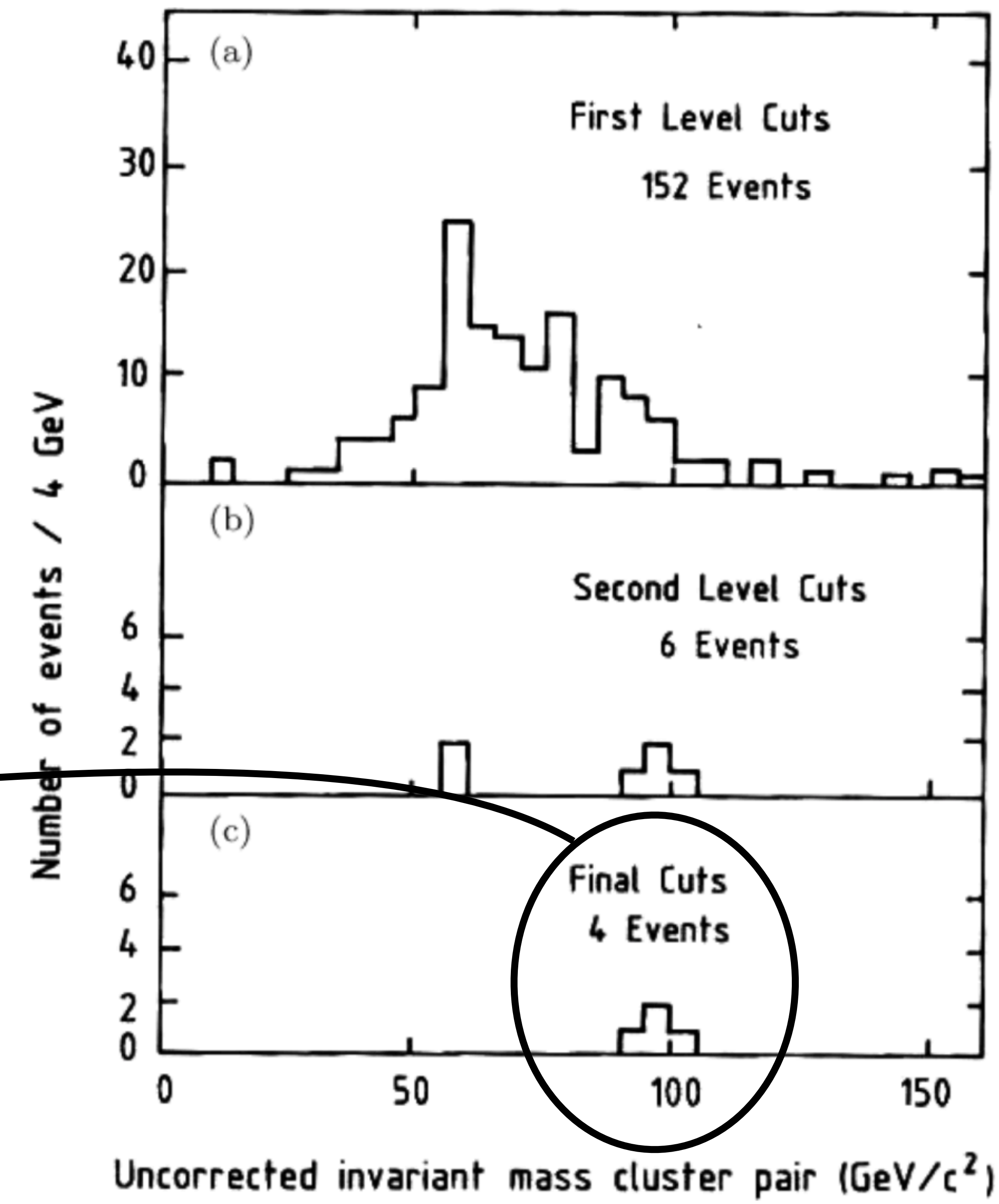




Calculate invariant mass of two particles to discover resonance

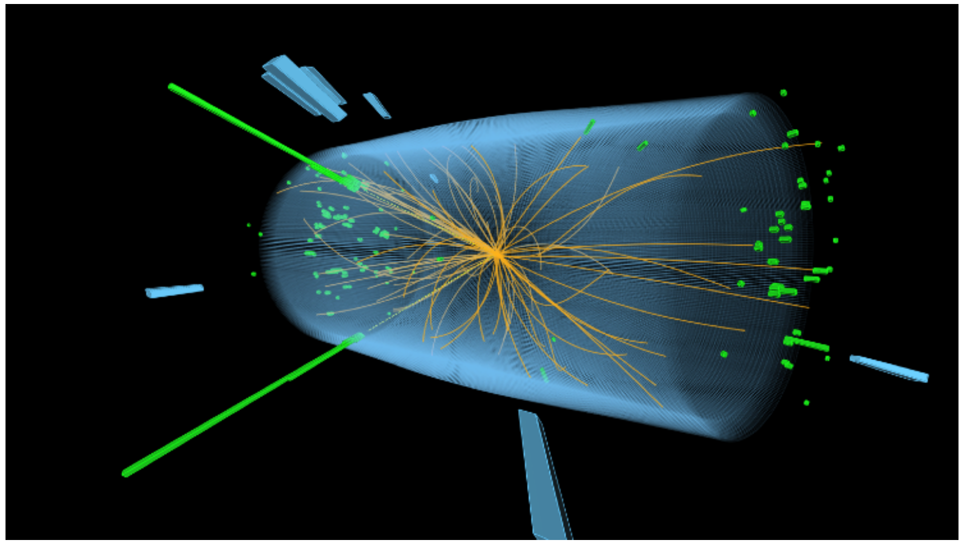
How to discover a new particle?



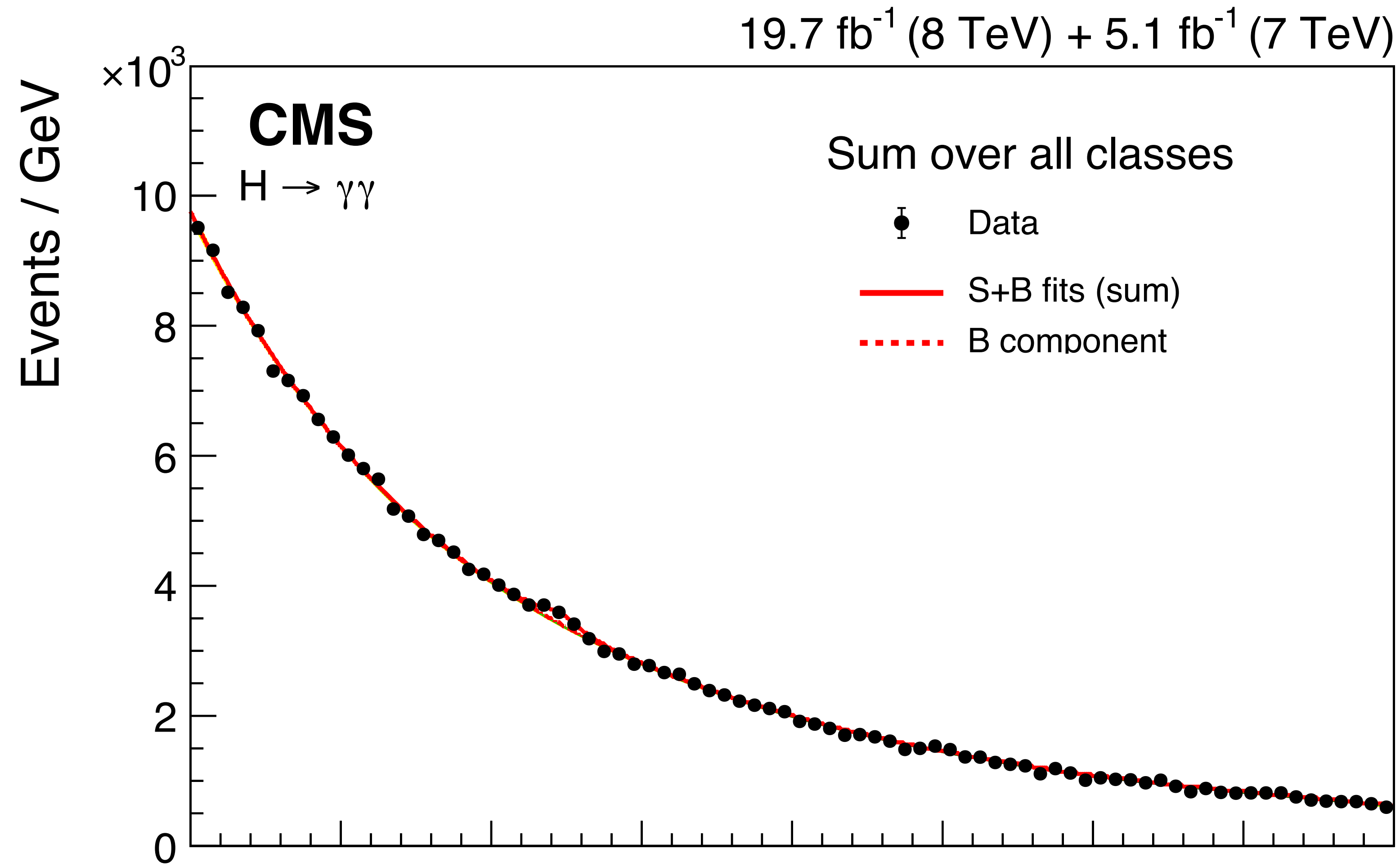


Nobel price for discovery of W and Z boson in 1984 (Rubbia and van der Meer)

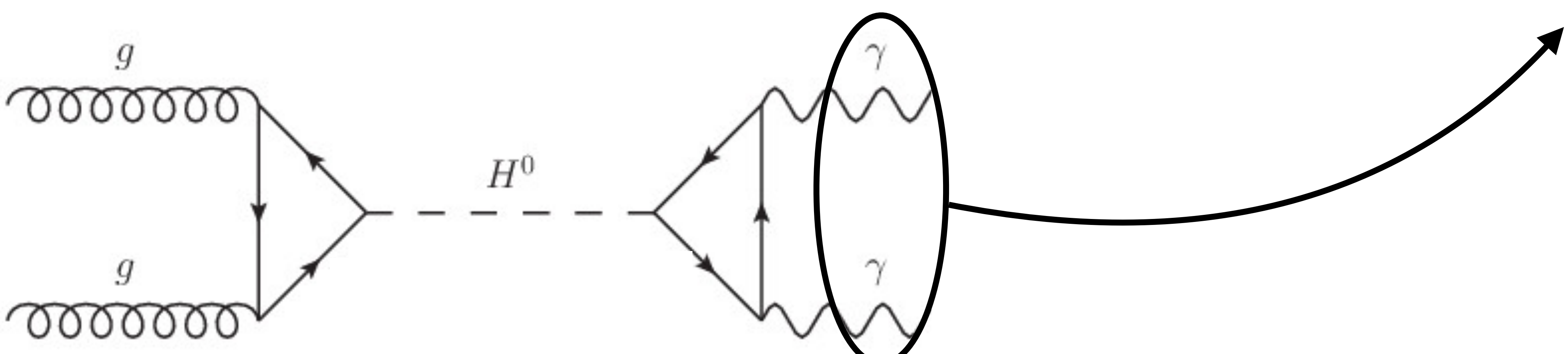
How to discover a new particle?



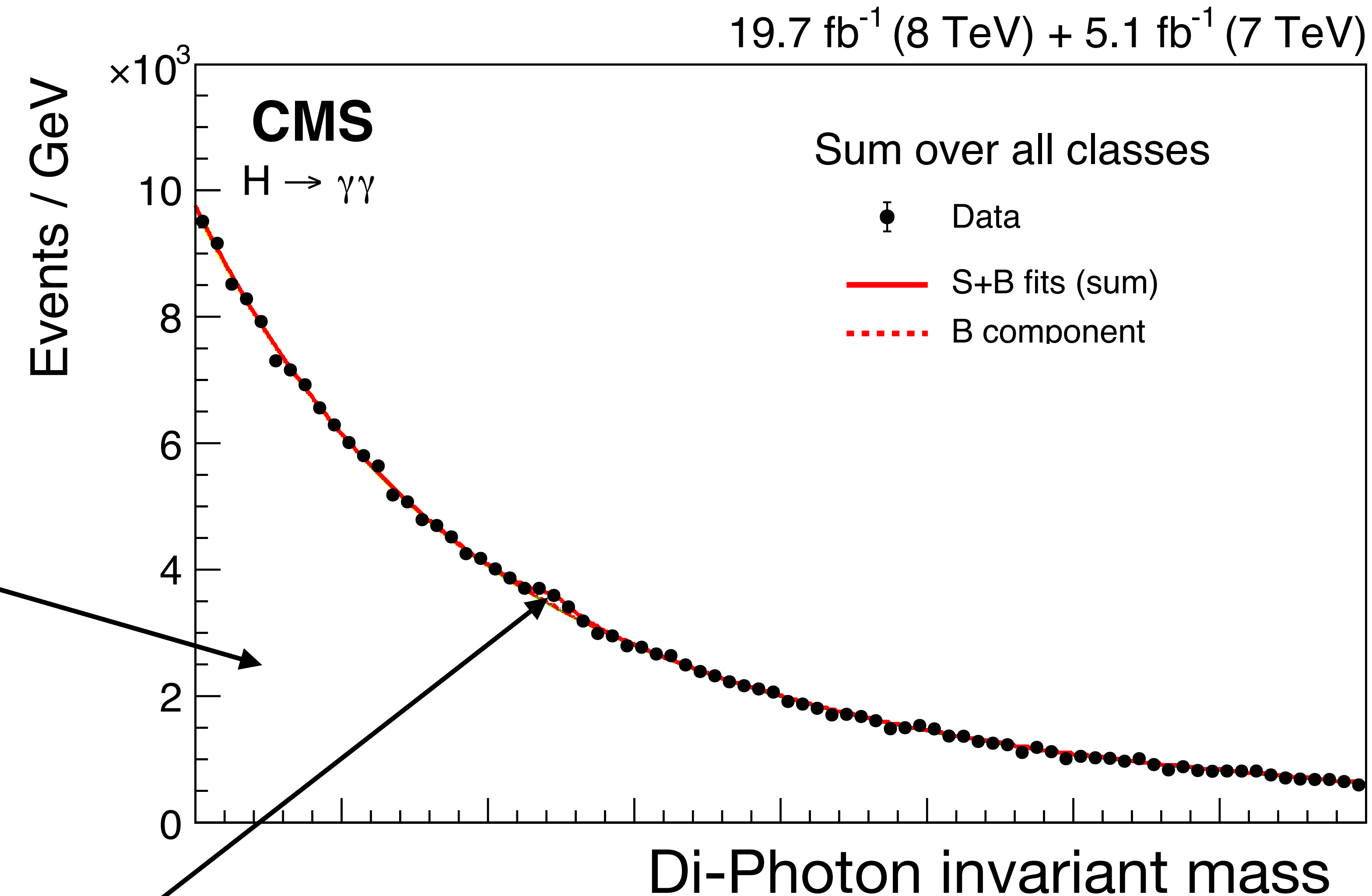
Event display from CMS from 2012



Di-Photon invariant mass



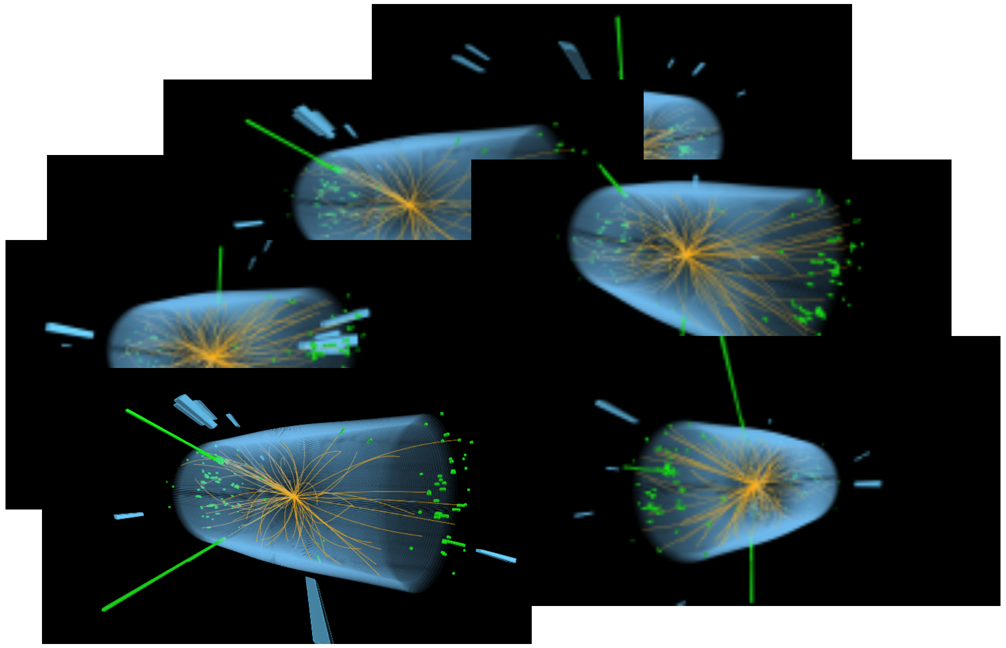
How to discover a new particle?



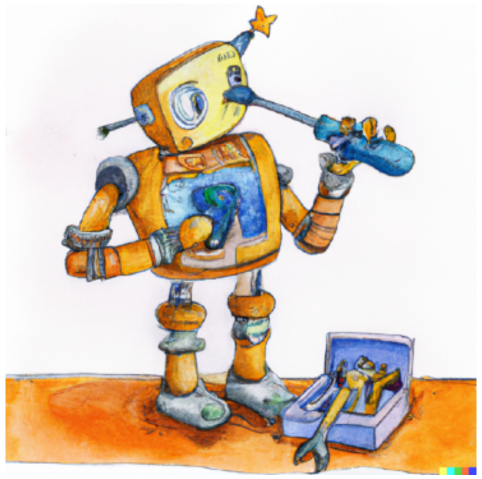
Most events are *background*

With a tiny signal on top

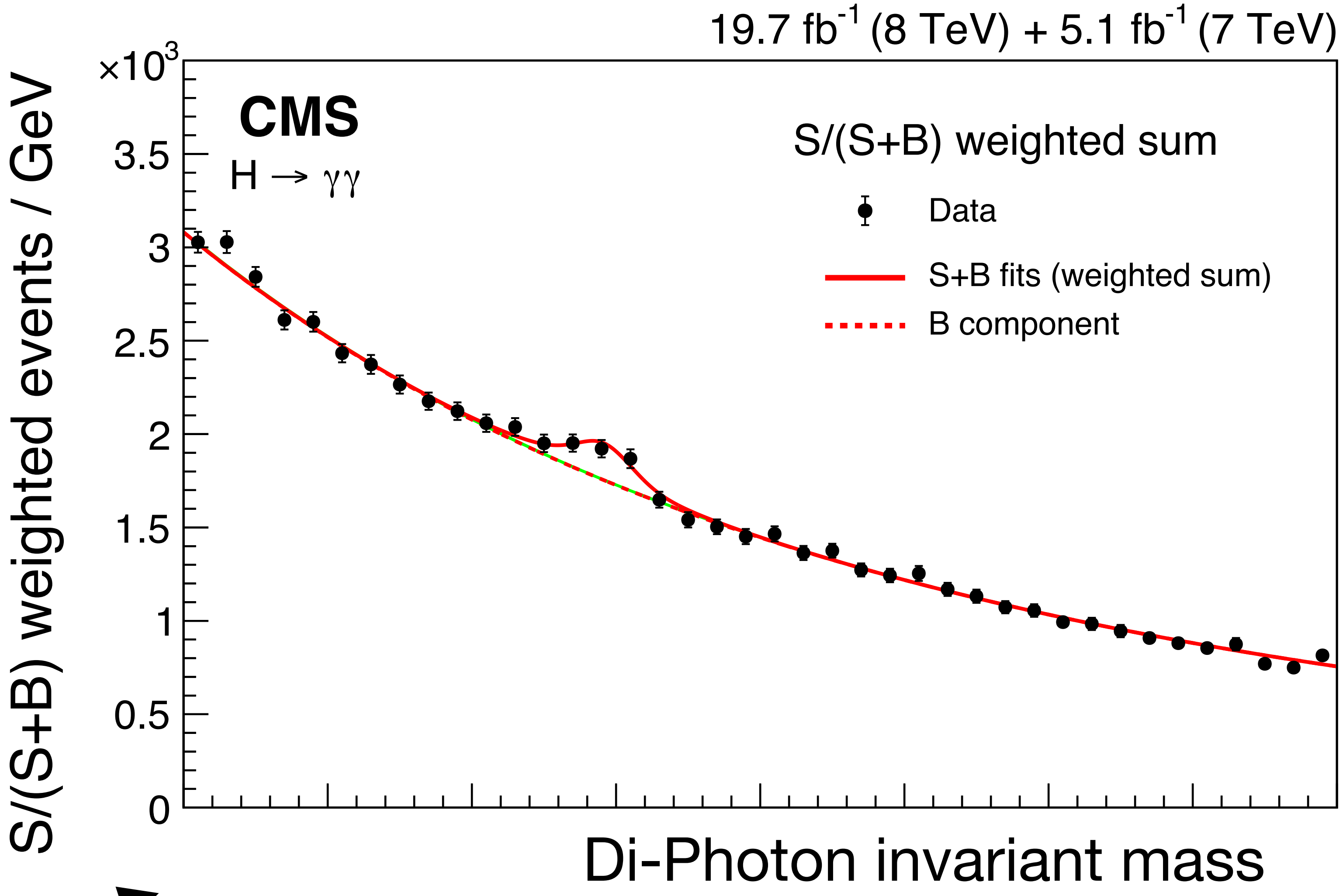
How to discover a new particle?



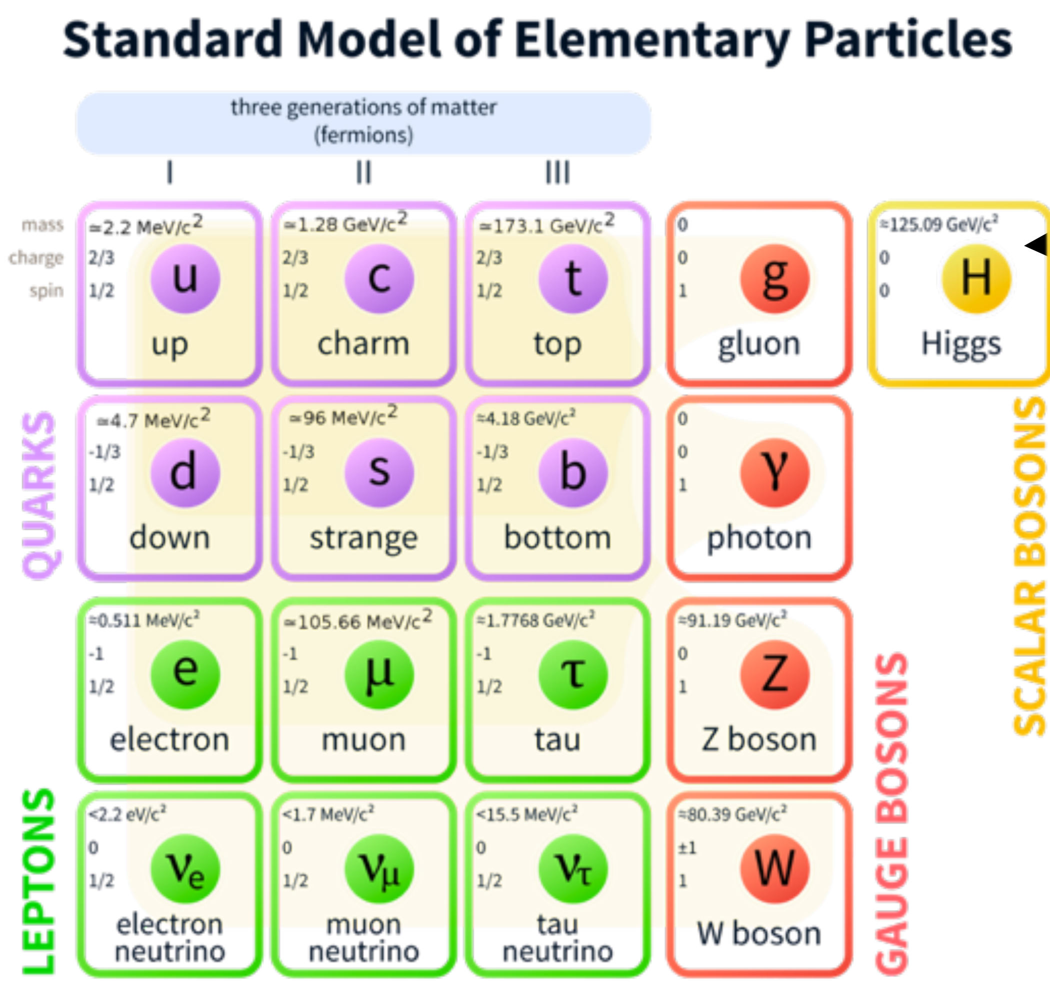
Divide into 25 categories according to purity, relative amount of signal using simple ML

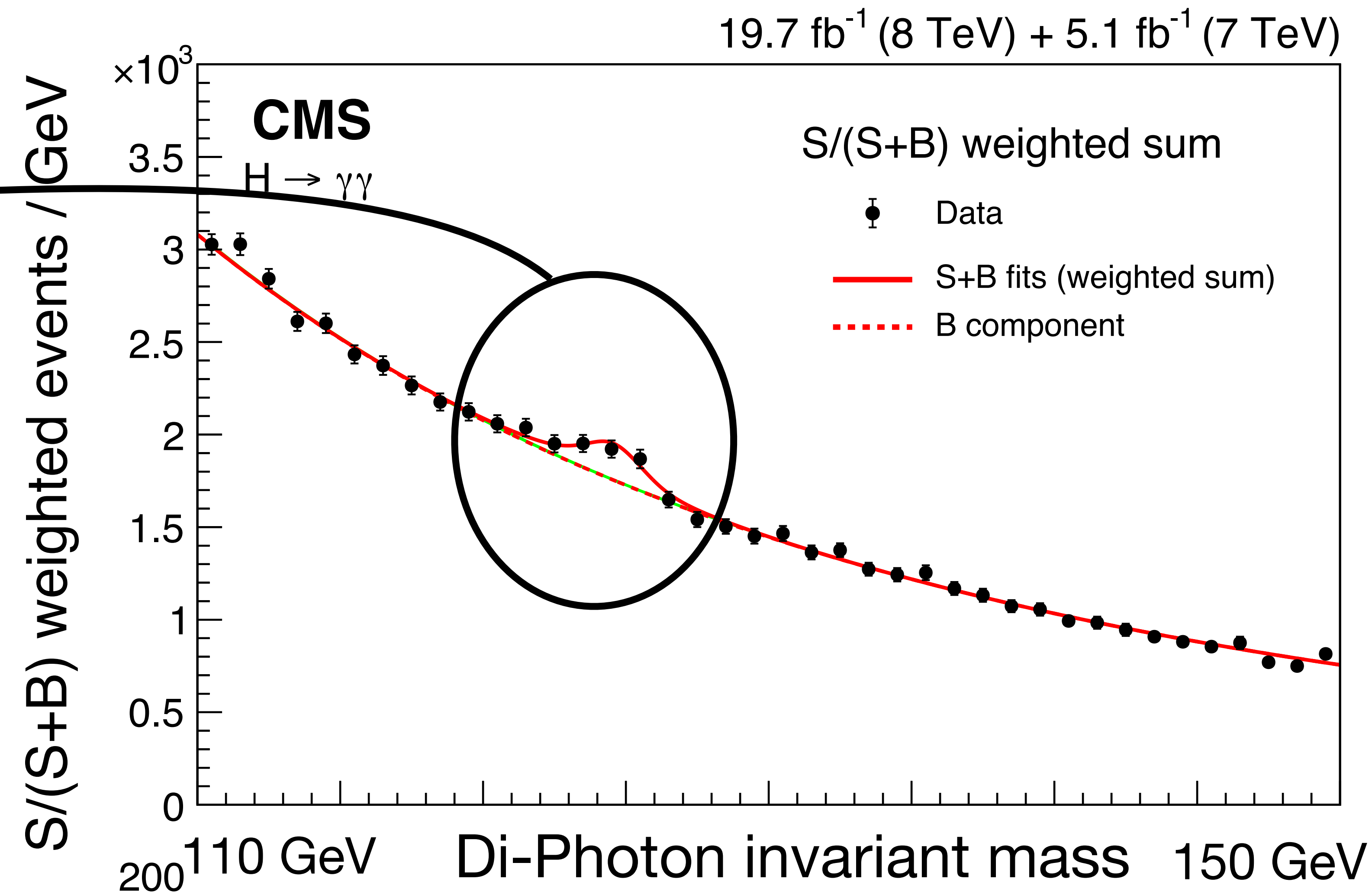


Weight events according to excepted purity of category



How to discover a new particle?



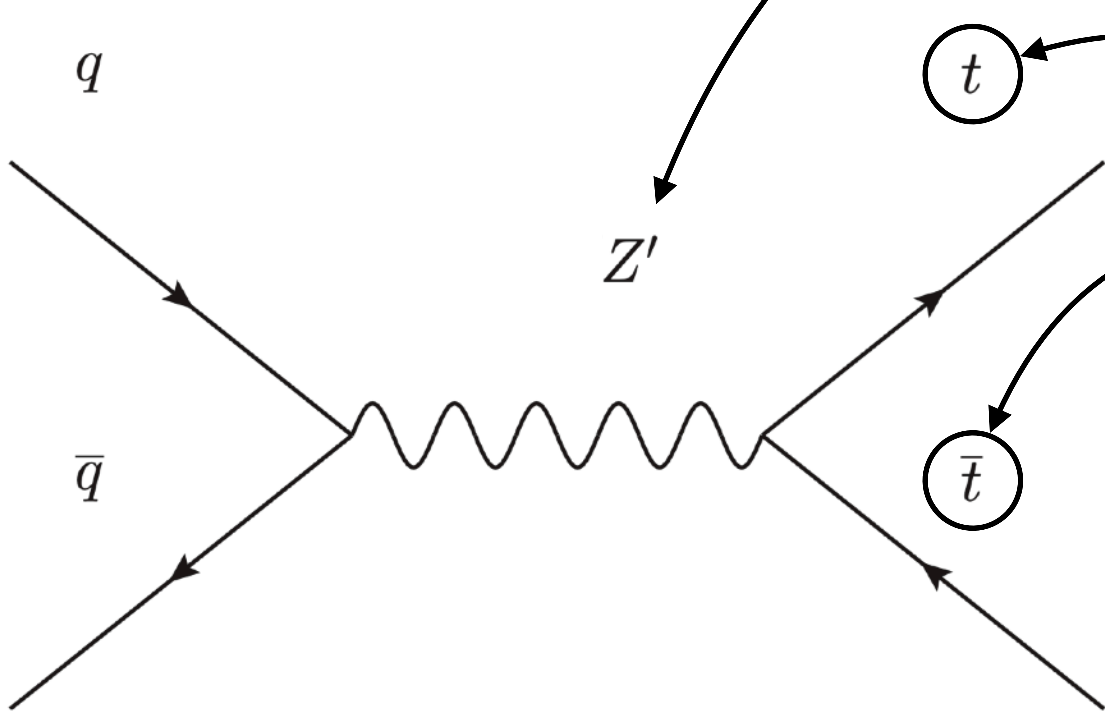


Nobel price for Higgs mechanism 2013 (Higgs and Englert)

How to discover a new particle?

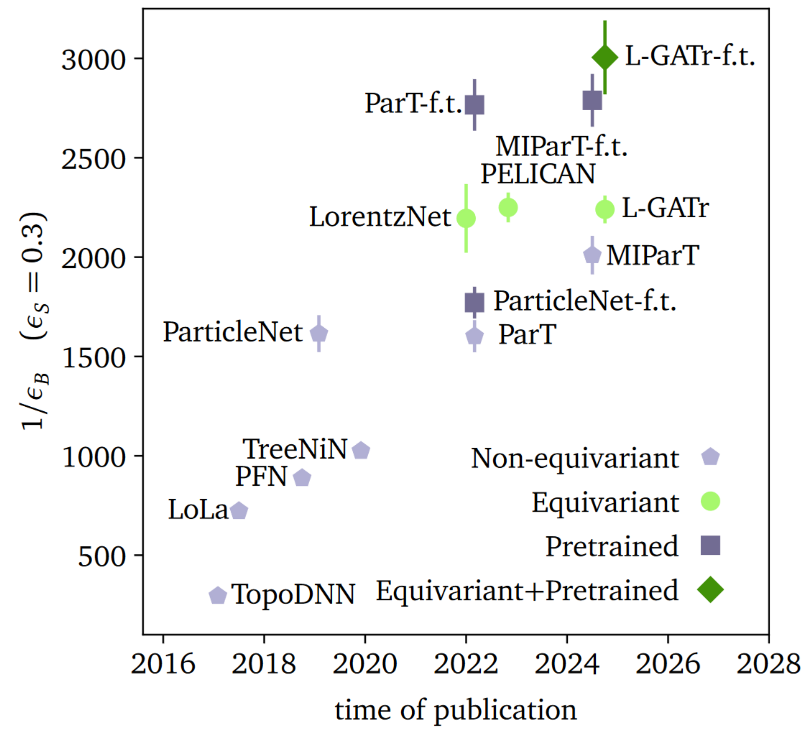
Assume we know what we are looking for: e.g. a heavier version of the Z boson

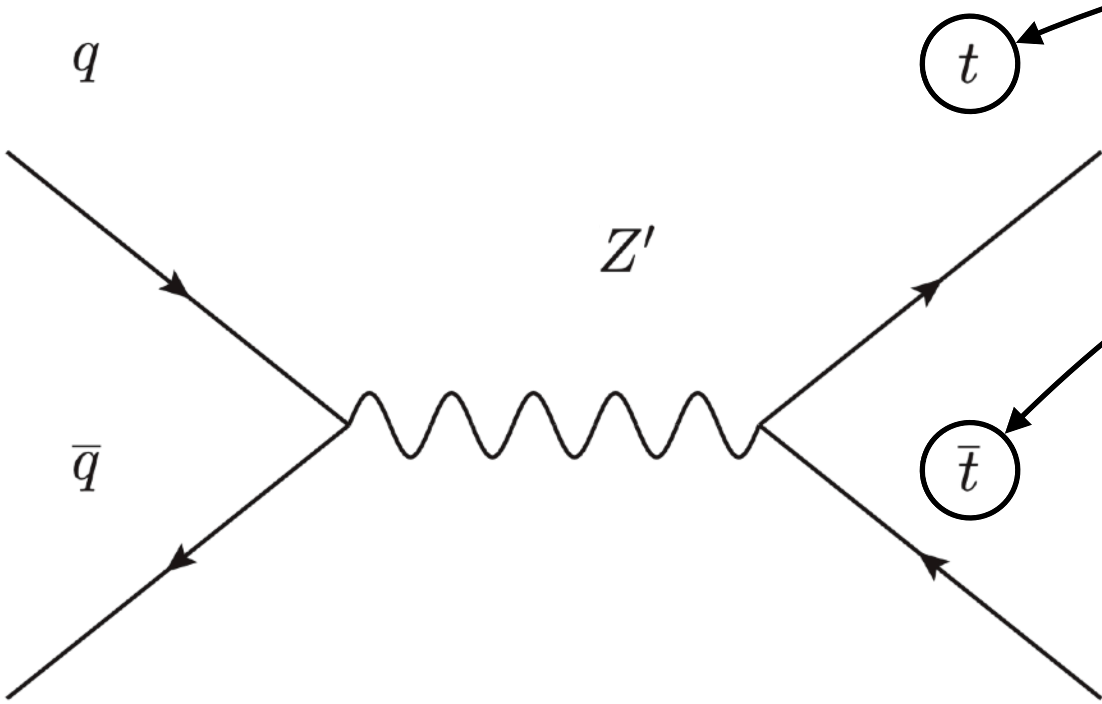
We don't know the mass of the resonant particle

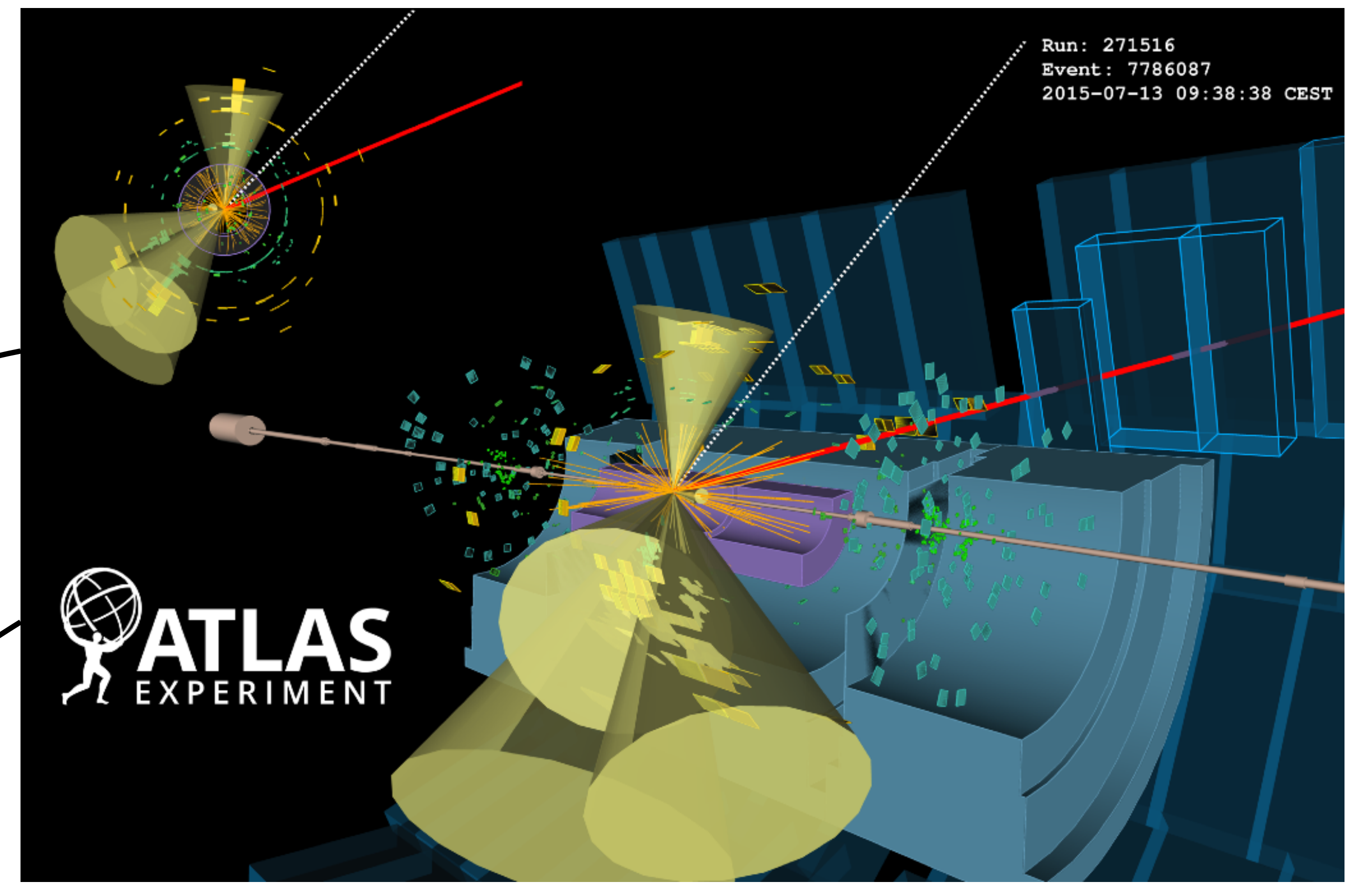


But we assume it decays to e.g. a pairs of top quarks

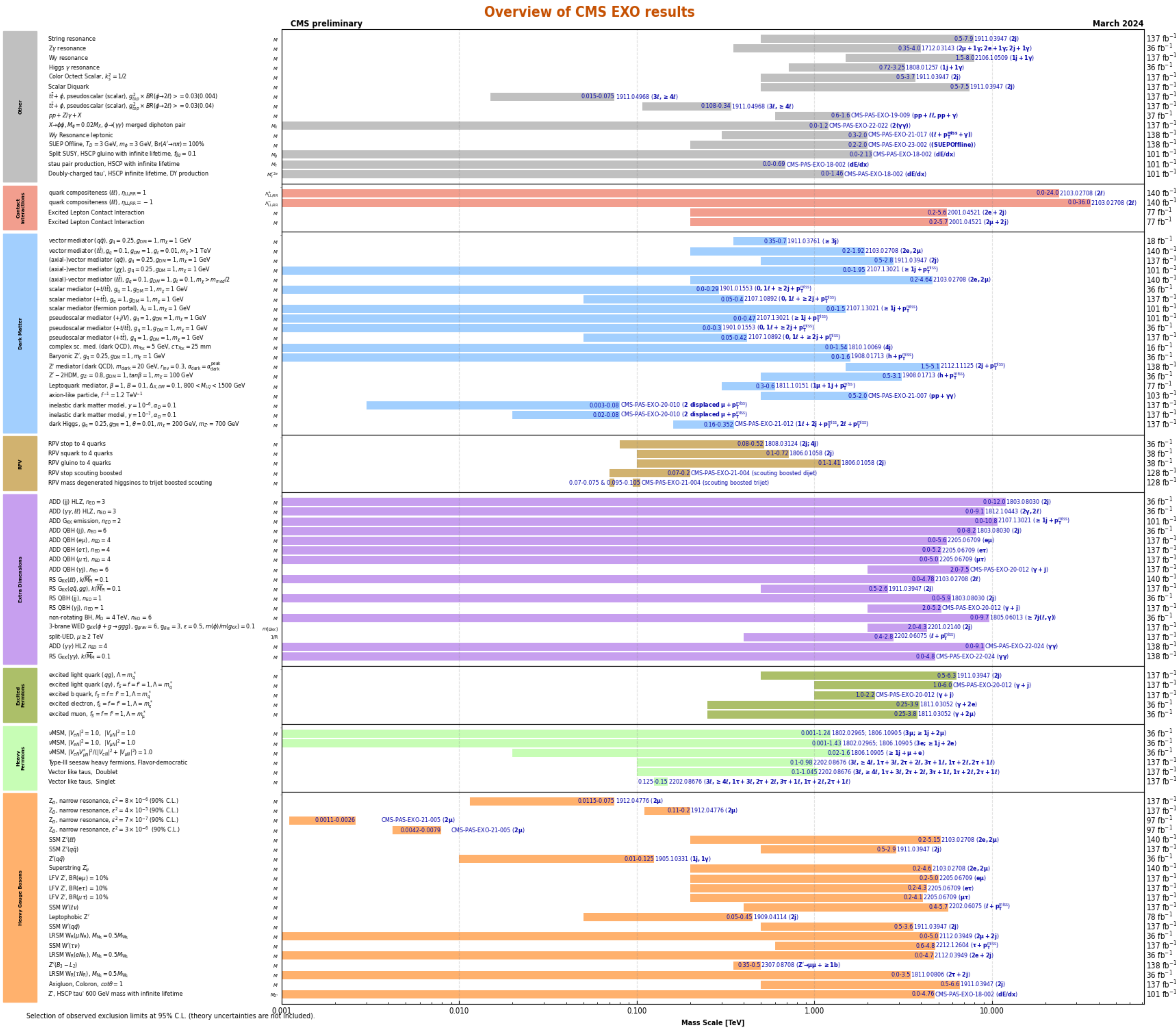
How to discover a new particle?







More complicated than electrons:
Use AI for top tagging
Discussed yesterday



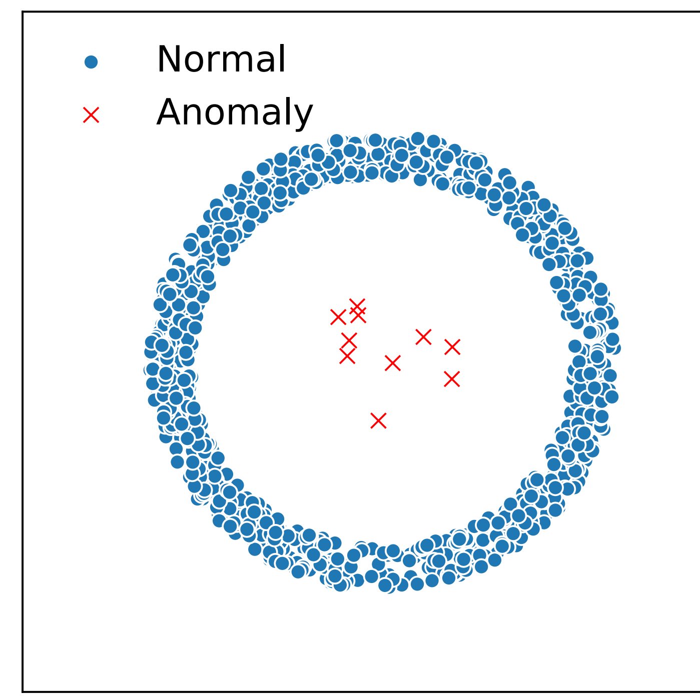
So far, no new physics in model-driven searches

Anomaly Detection

What is an anomaly?

Point anomaly

- **Outliers**: Datapoints far away from regular distribution
- Examples:
  - Detector malfunctions
  - Background-free search



And now?





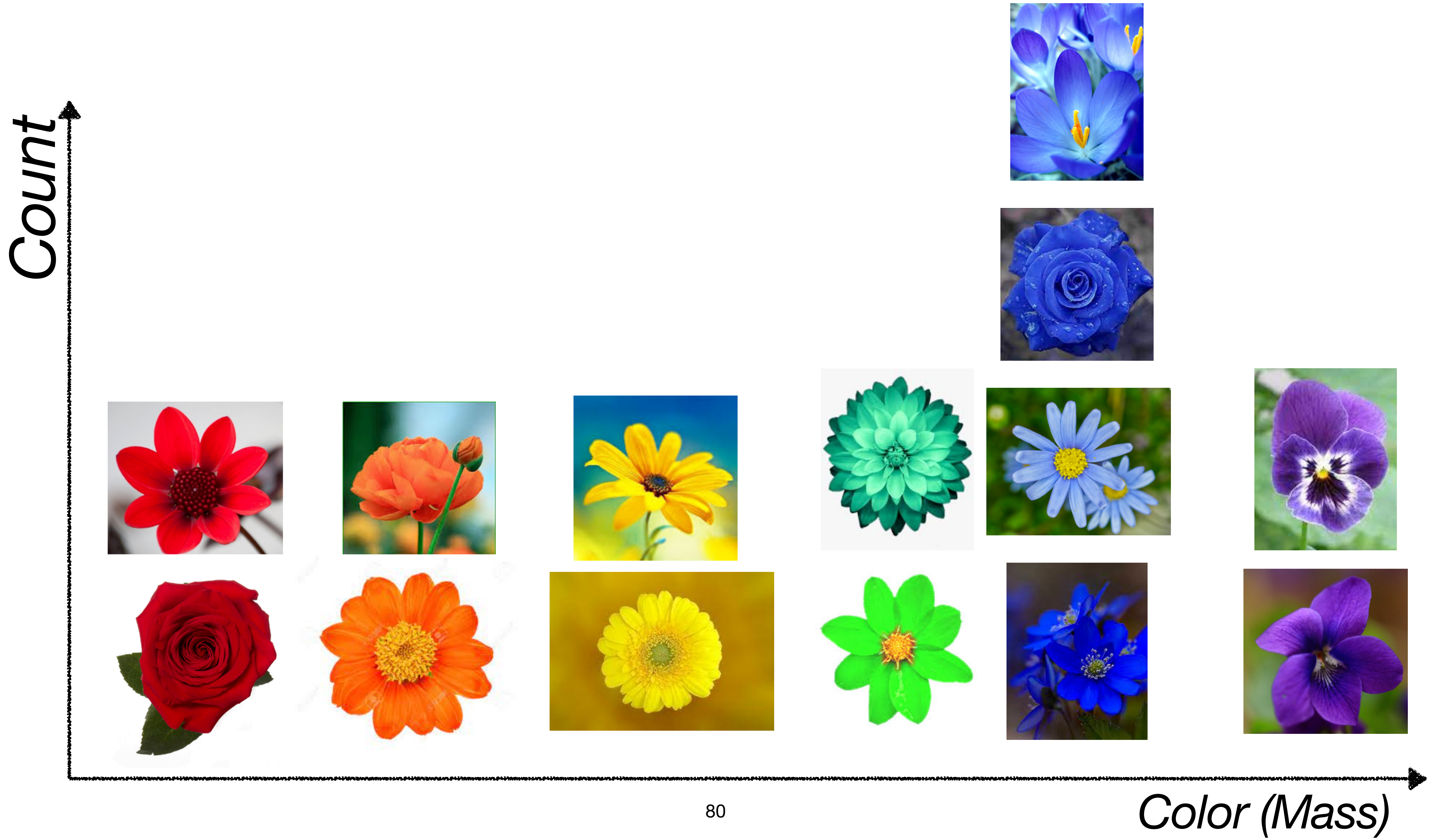






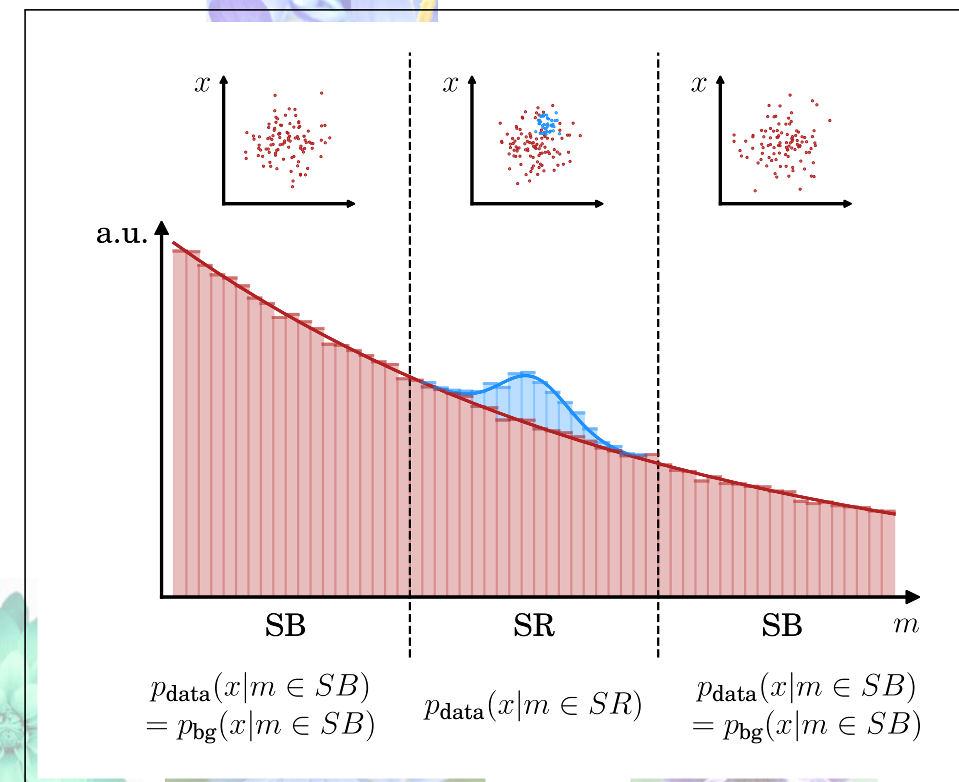



Group anomaly



Group anomaly

- Individual examples not anomalous, but interesting collective behaviour
- Examples:
    - New physics searches, e.g. resonances
    - Excess in time series

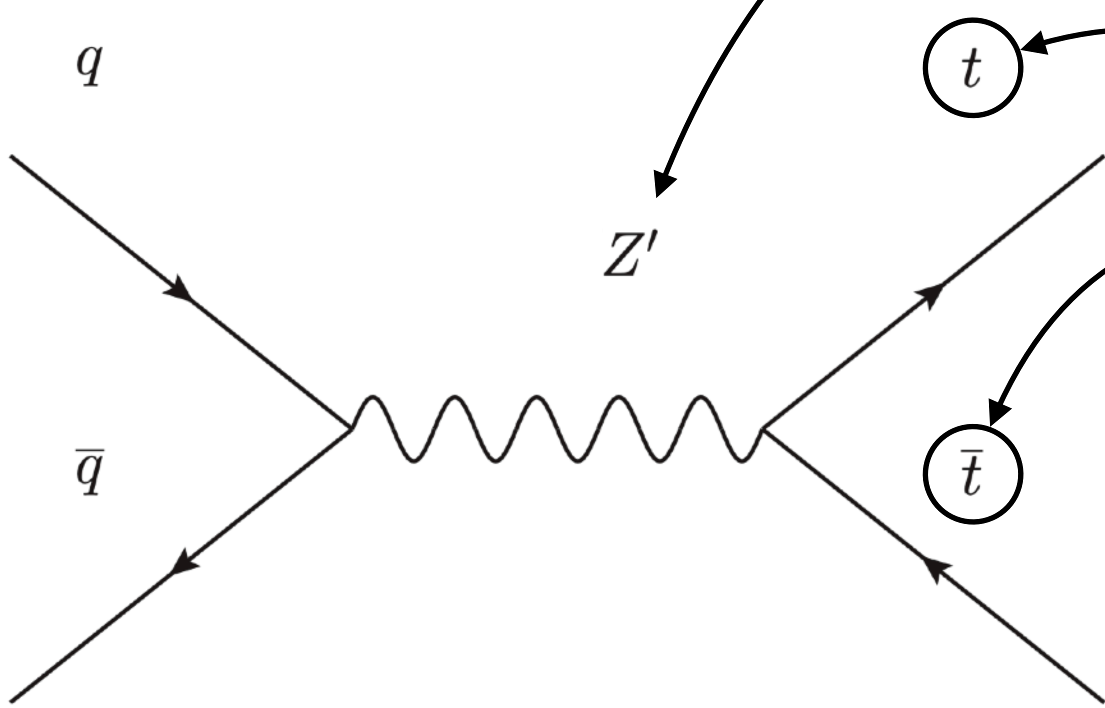


Group anomalies

Revisit assumptions

Assume we know what we are looking for: e.g. a heavier version of the Z boson

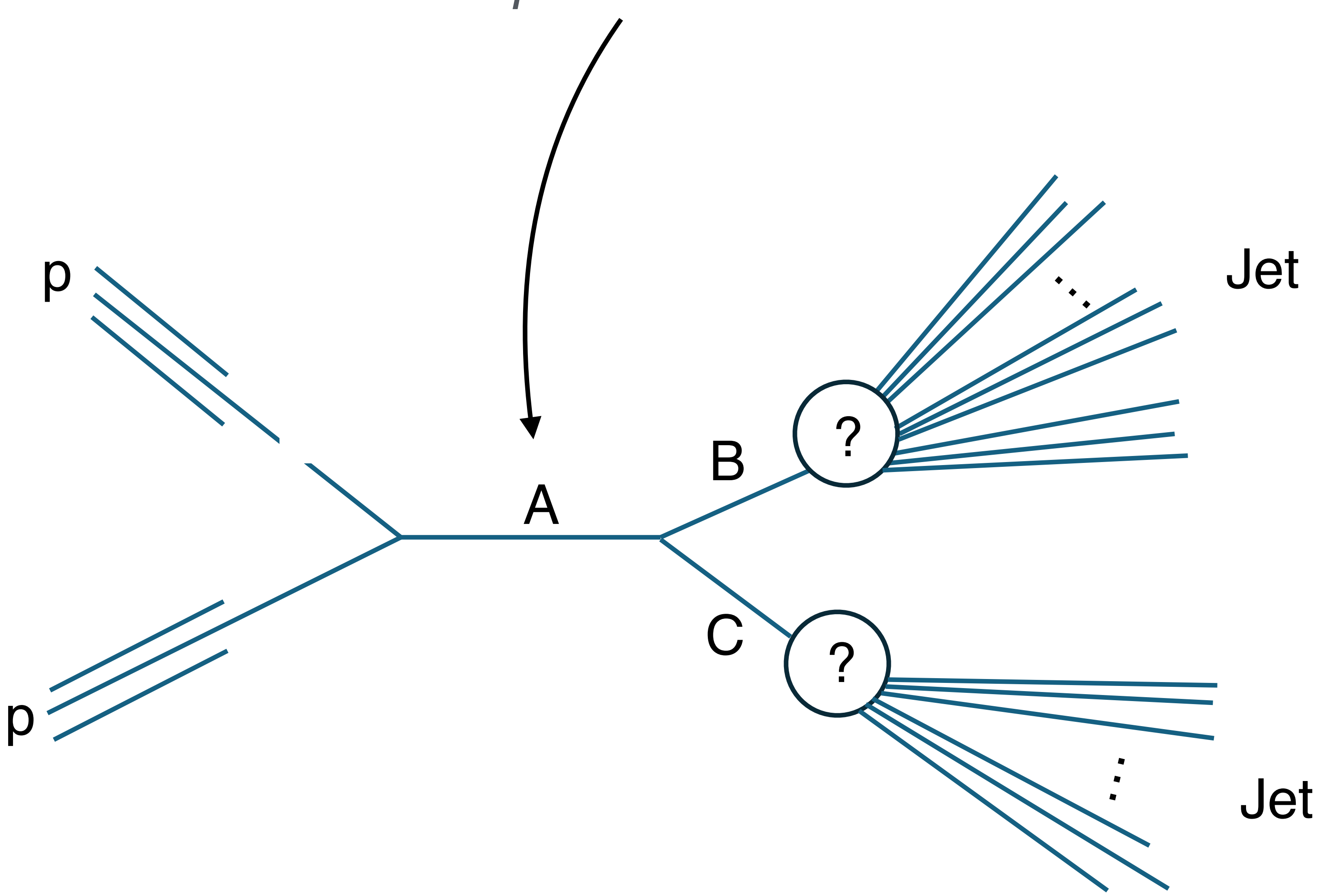
We don't know the mass of the resonant particle



But we assume it decays to a pairs of top quarks

Assumptions, revisited

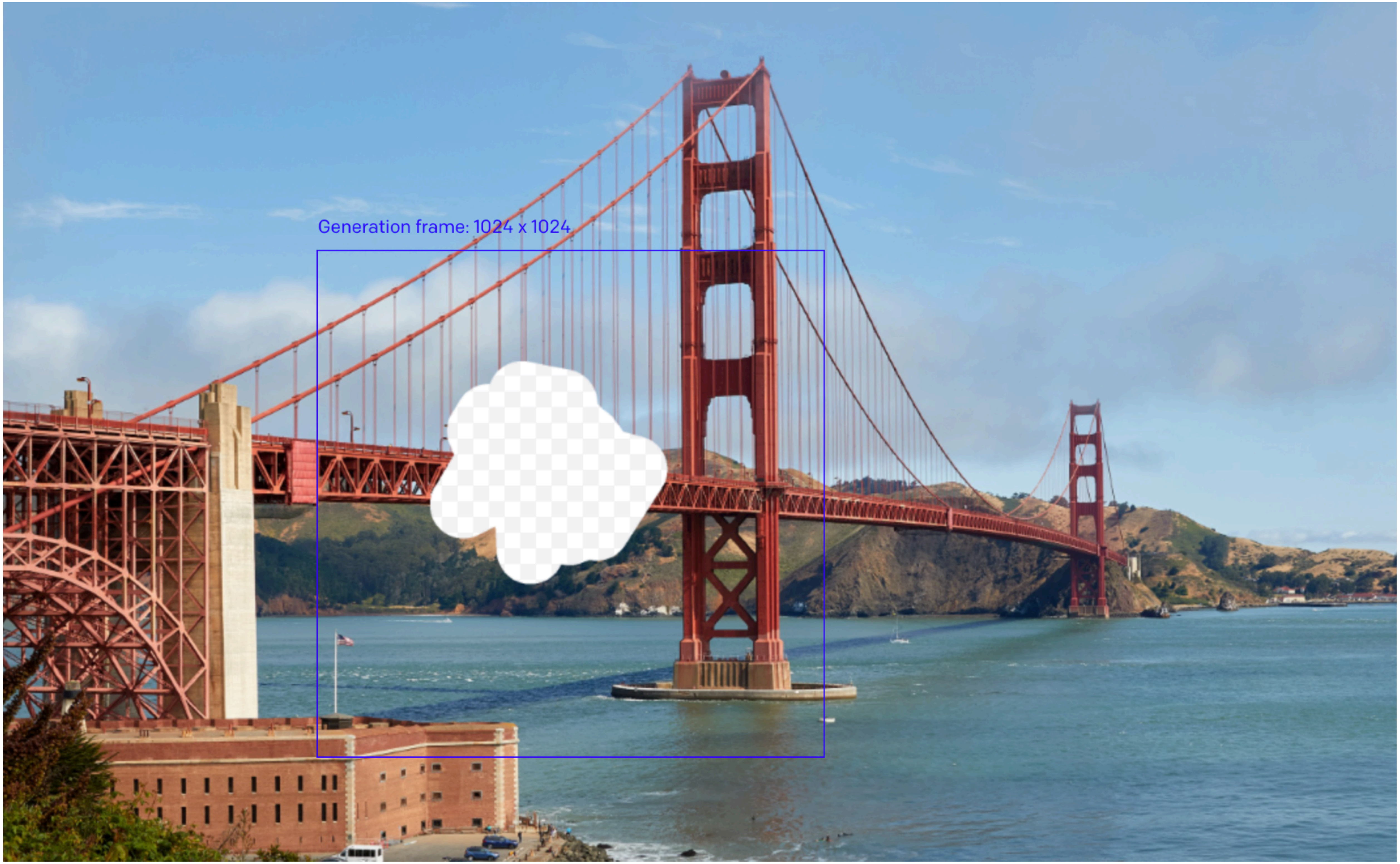
We don't know the mass and type of the resonant particle



And we don't know what particles it decays to

(for now assume it decays into two Jets)

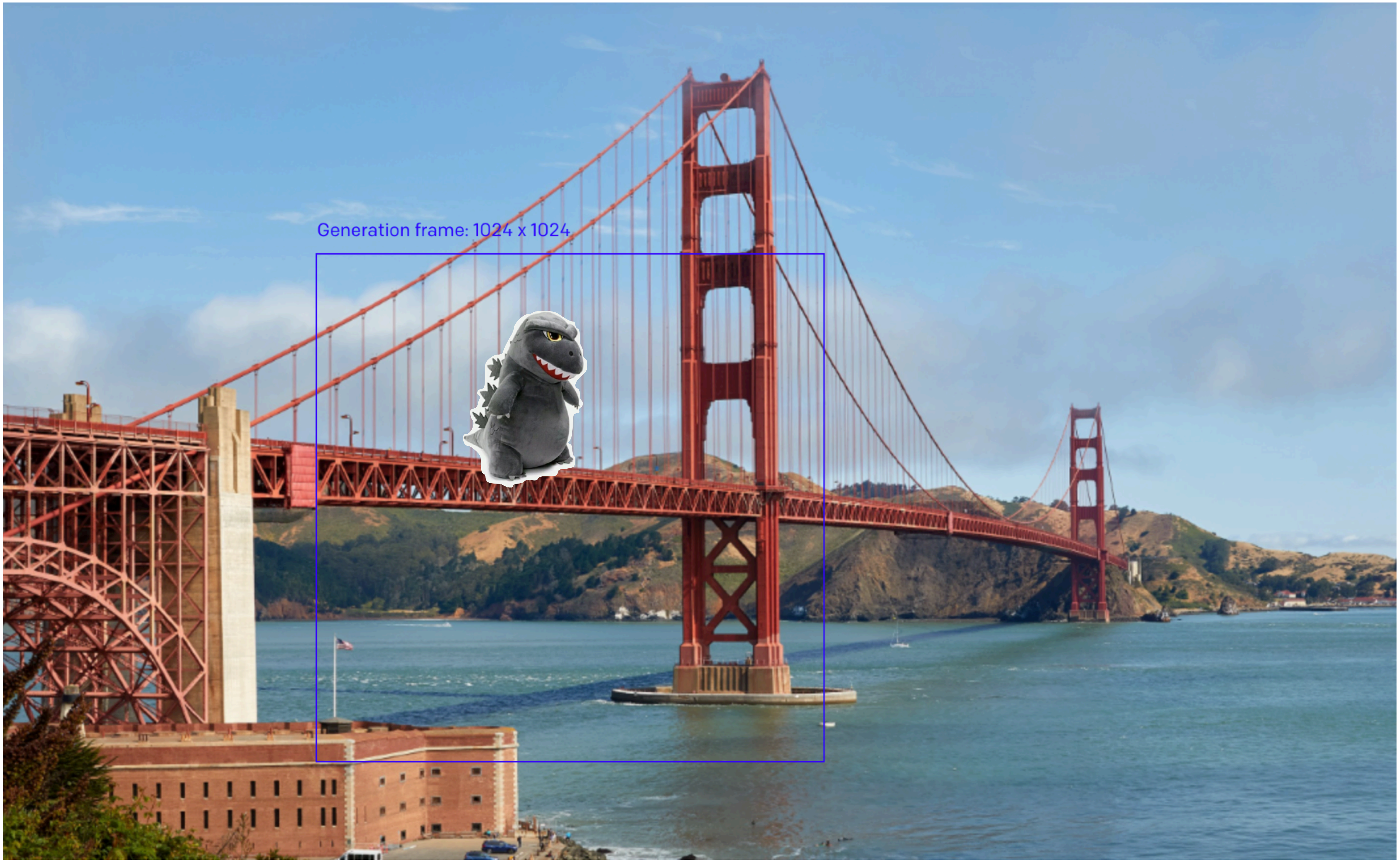

Generation frame: 1024 x 1024



Define a signal cut-out

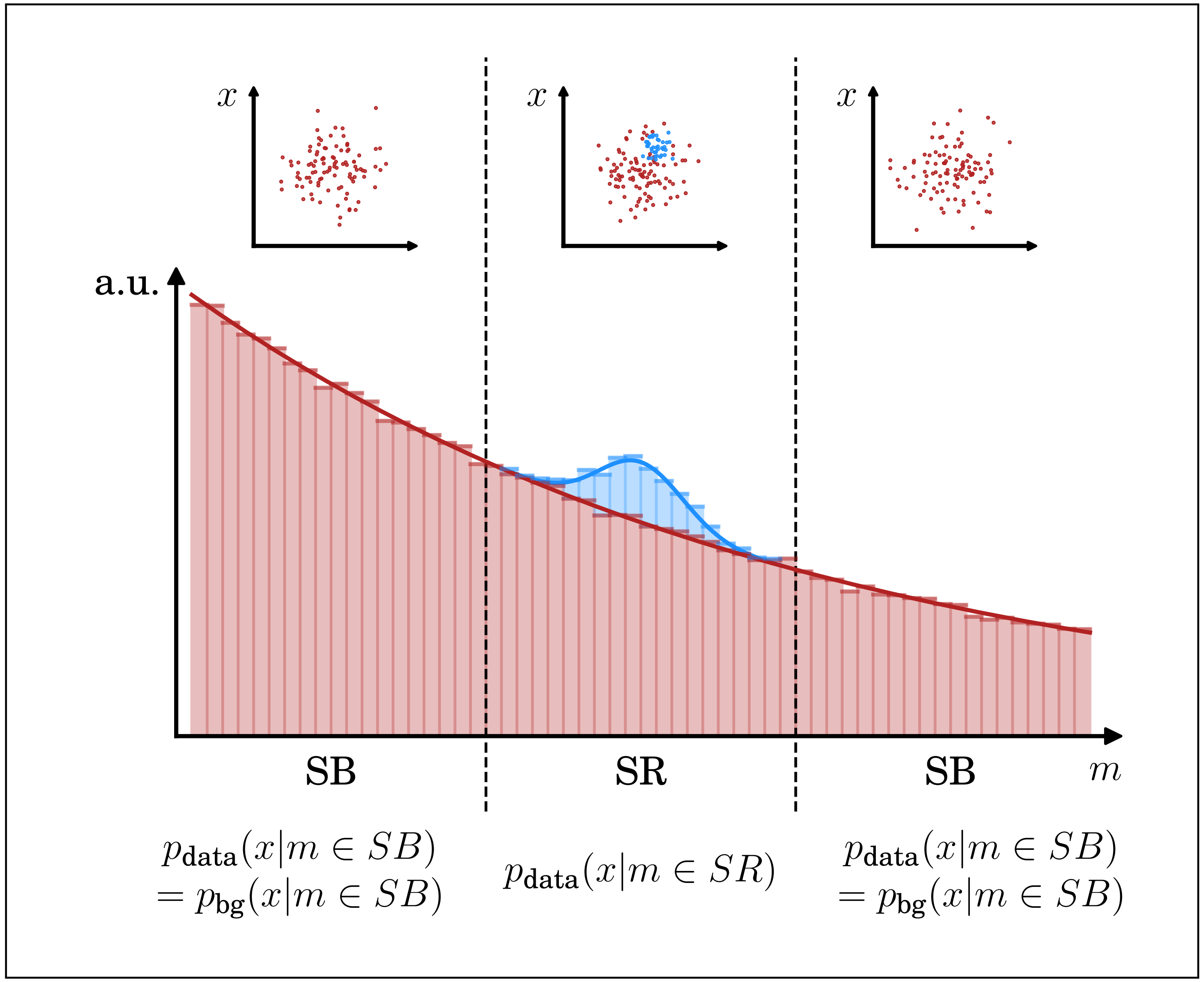
And interpolate the background

Original Interpolated


Generation frame: 1024 x 1024


Original
Interpolated
Generation frame: 1024 x 1024
Generation frame: 1024 x 1024
Corporate needs you to find the differences
between this picture and this picture.
Machine learning
classifier
ANOMALY FOUND!

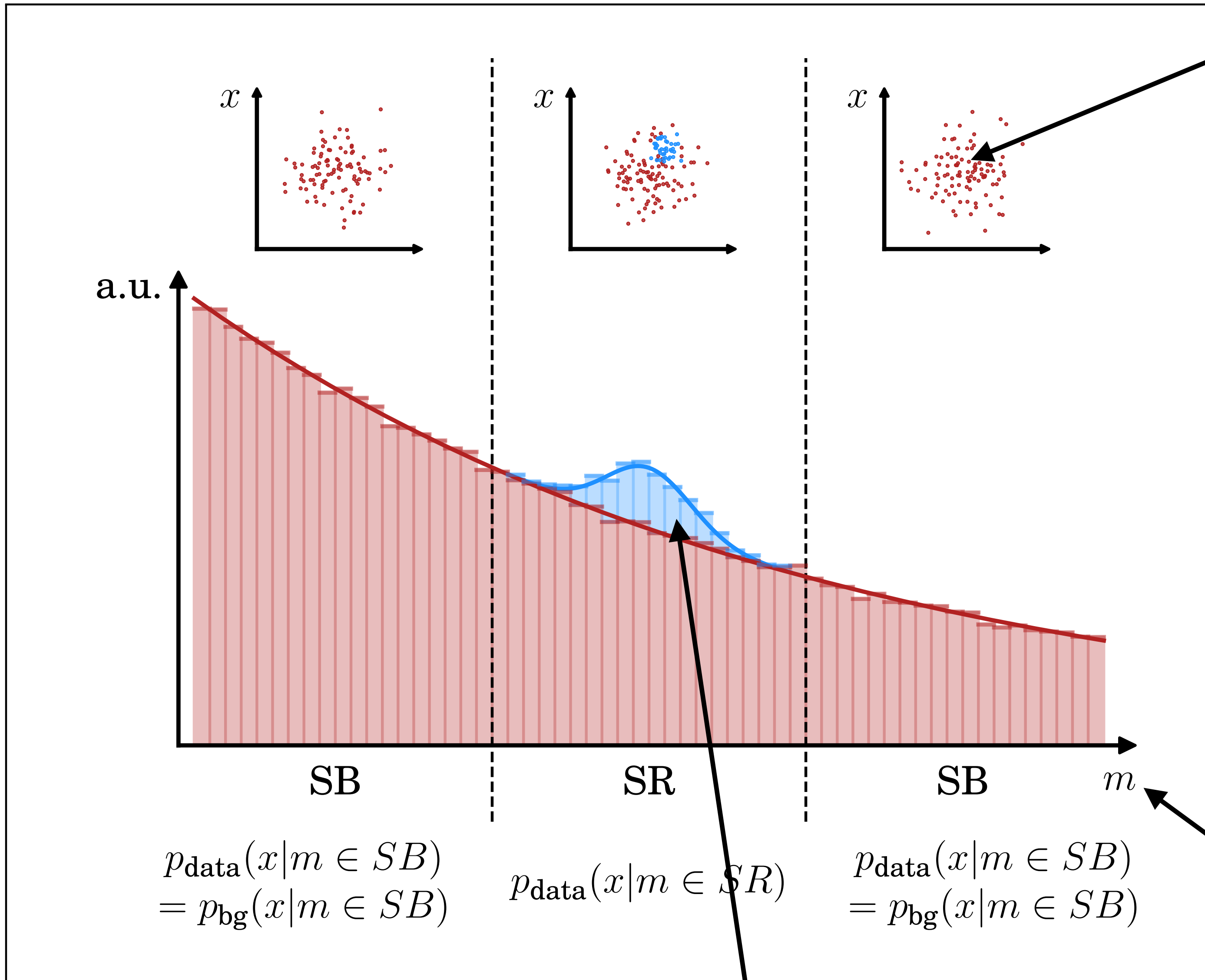
Resonant Anomalies



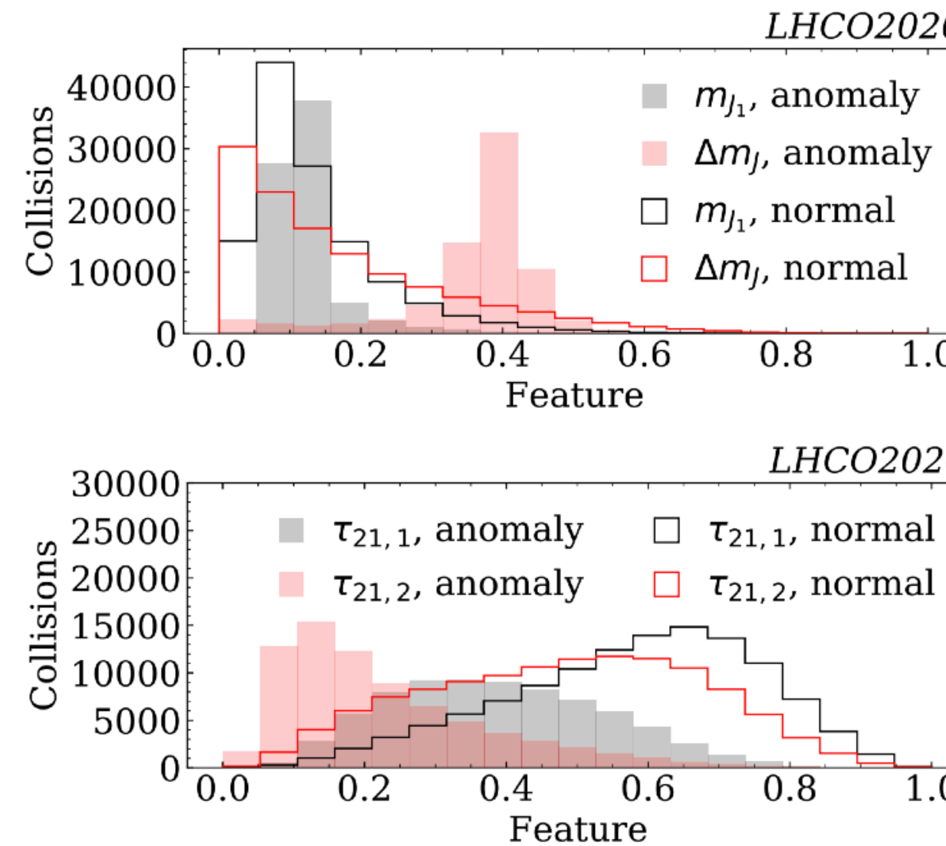
Consider resonant anomalies: fully data-based construction of anomaly detection score

GK, Nachmann, Shih et al 2101.08320;
Hallin, .., **GK** et al 2109.00546;

Resonant Anomalies



4 additional features

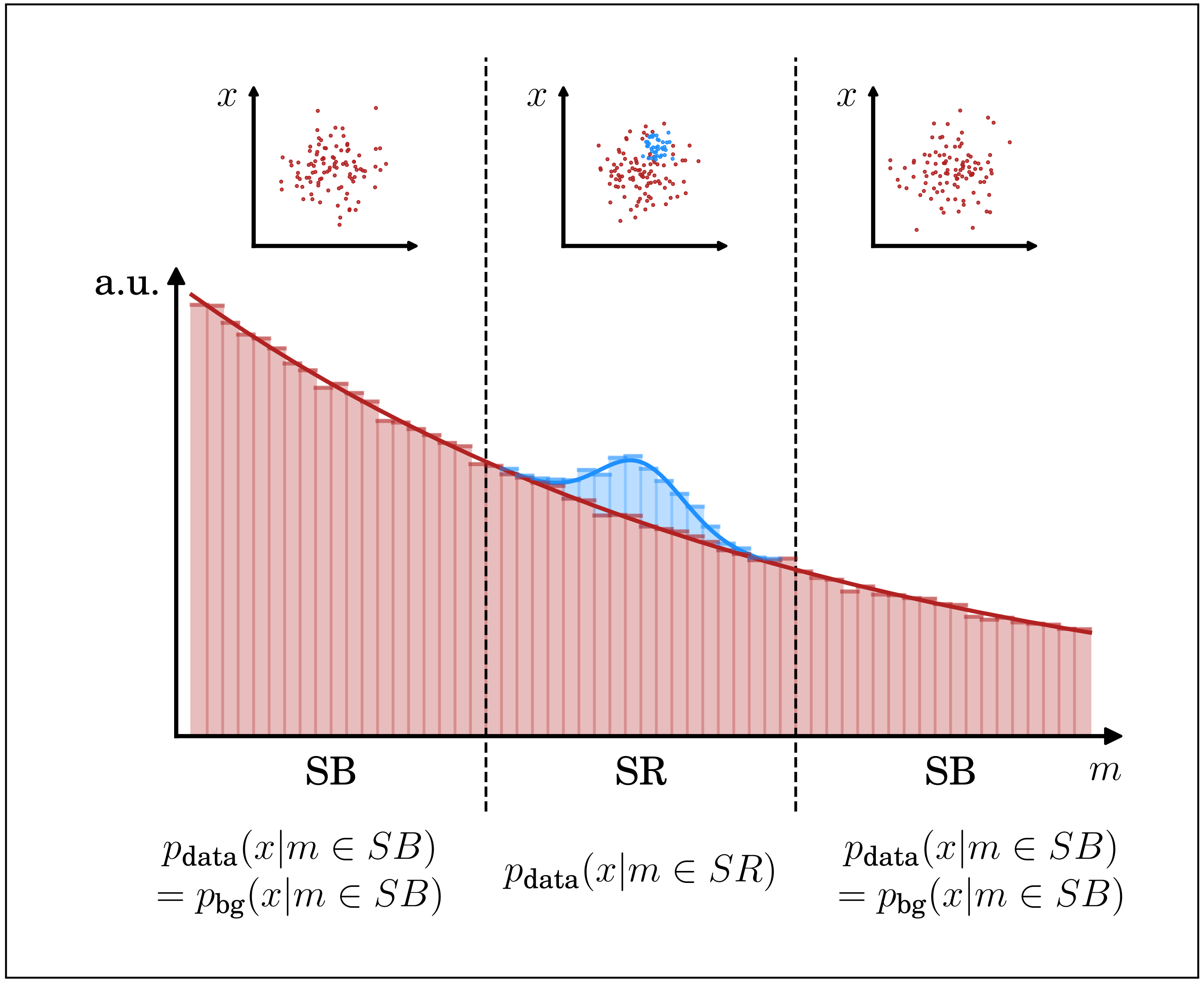


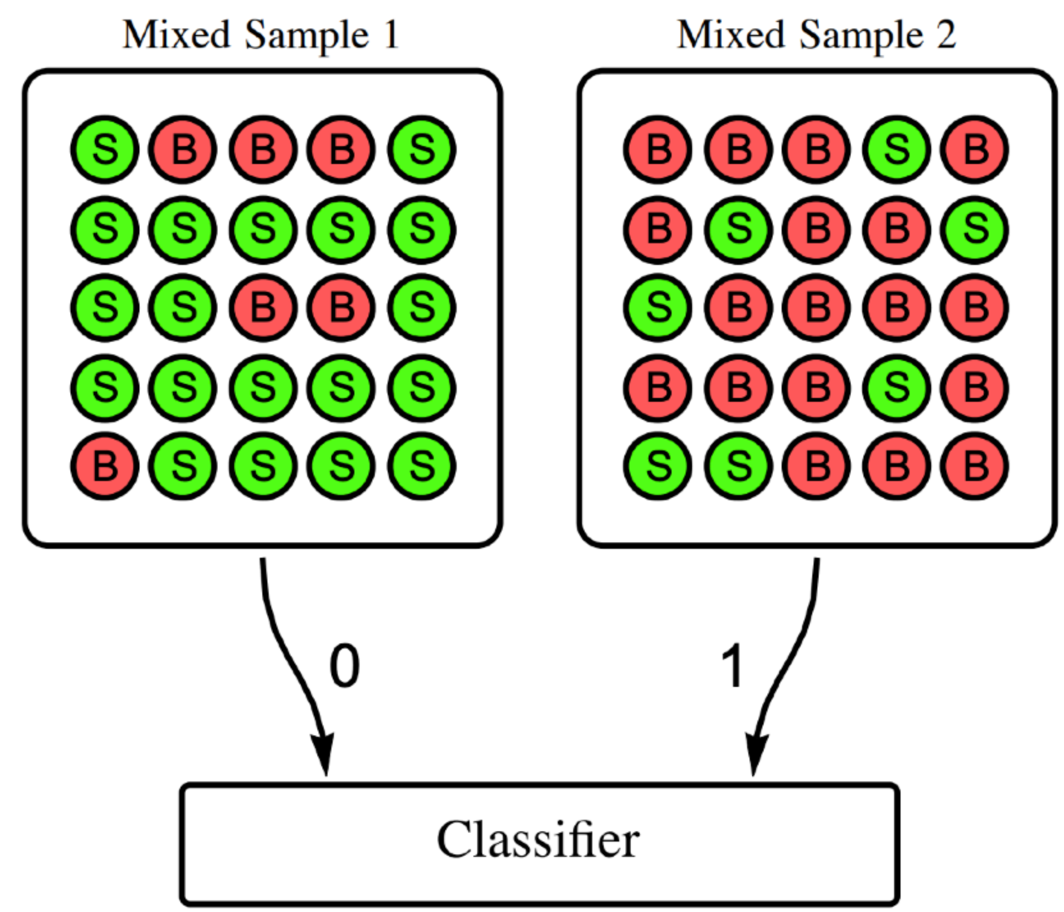
one resonant feature

Signal too small to be visible in inclusive distribution

GK, Nachmann, Shih et al 2101.08320;
Hallin, .., **GK** et al 2109.00546;

CWola





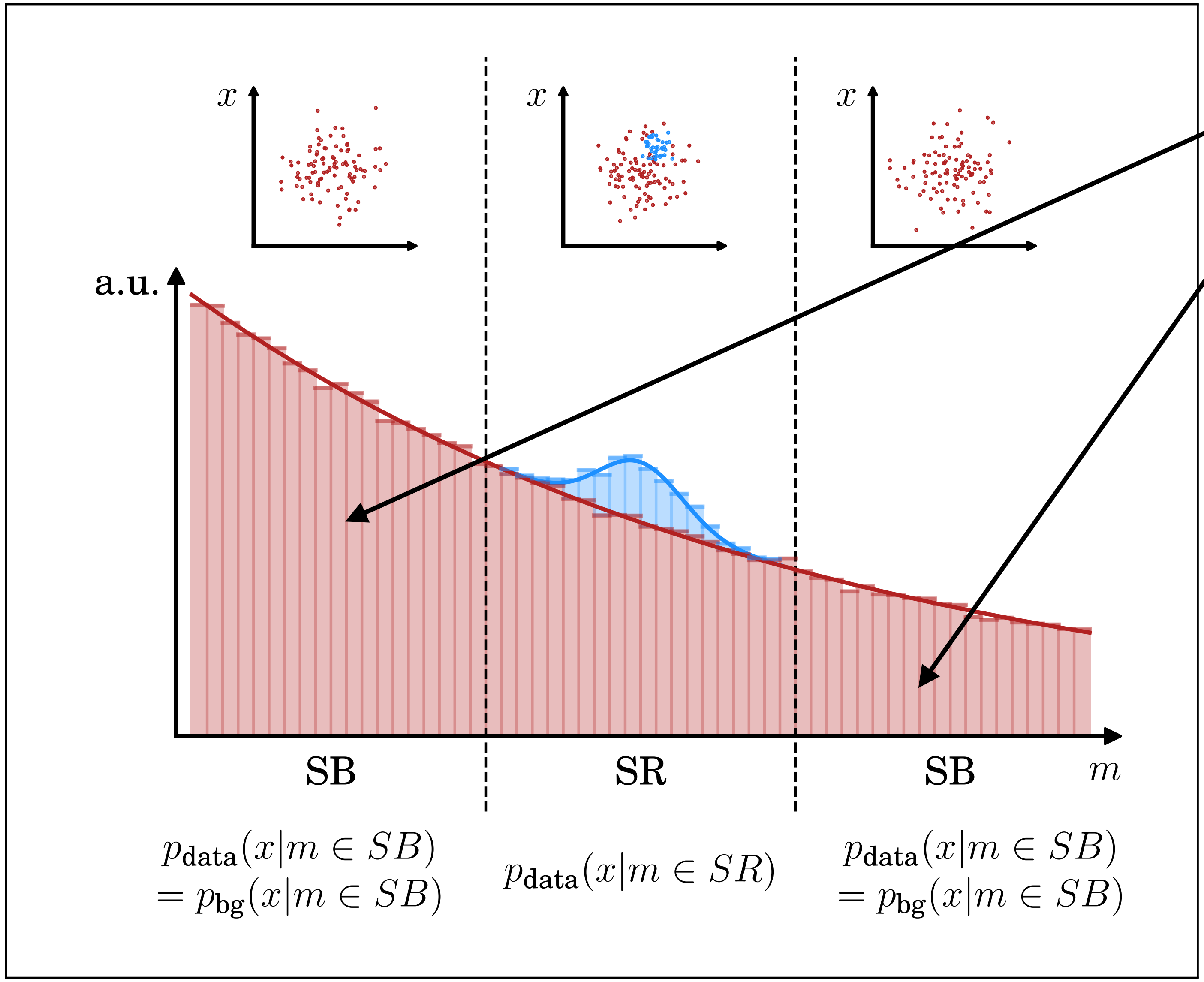
Use weak classifier.

$$L_{M_1/M_2} = \frac{p_{M_1}}{p_{M_2}} = \frac{f_1\, p_S + (1-f_1)\, p_B}{f_2\, p_S + (1-f_2)\, p_B} = \frac{f_1\, L_{S/B} + (1-f_1)}{f_2\, L_{S/B} + (1-f_2)},$$

Problems?

Nachman et al 1708.02949

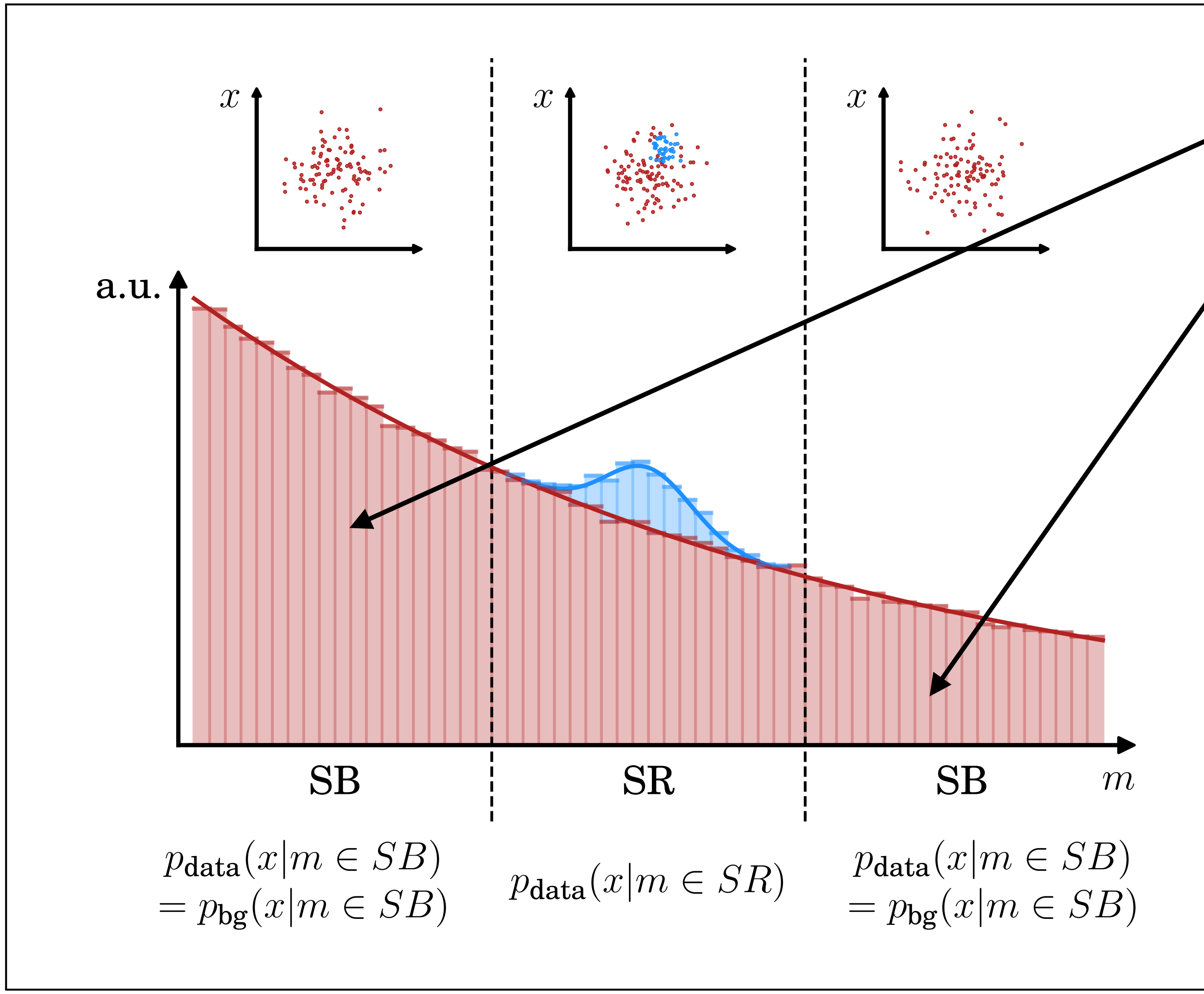
CATHODE



Train generative model

GK, Nachmann, Shih et al 2101.08320;
Hallin, .., **GK** et al 2109.00546;

Aside: Generative Models



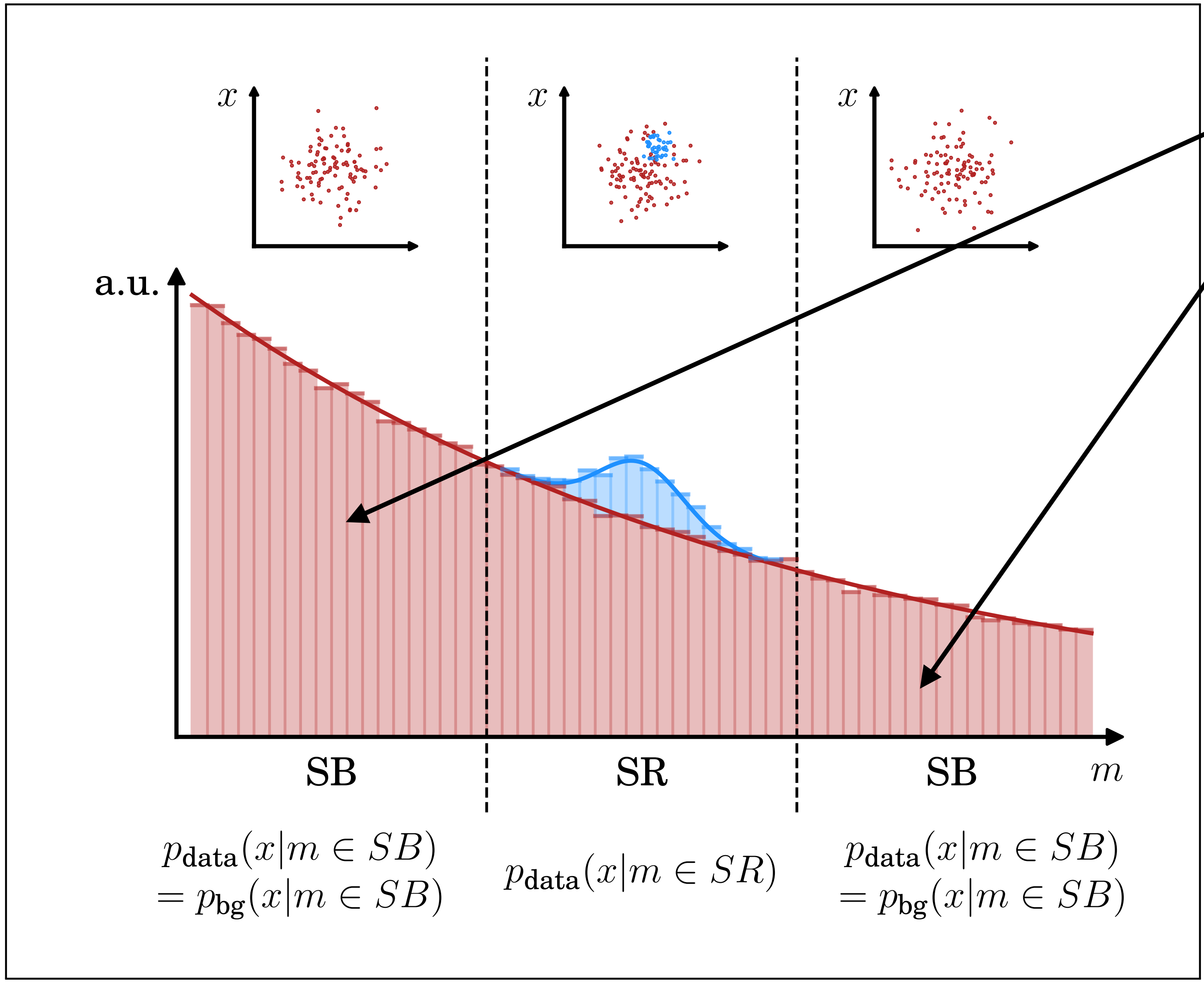
Train generative model

Given a set of examples $\{x_i\}$ drawn from a distribution $p(\mathbf{x})$, learn a mapping f_Θ so that $f_\Theta(q(\mathbf{z})) = p(\mathbf{x})$ for a random distribution $q(\mathbf{z})$

More later

GK, Nachmann, Shih et al 2101.08320;
Hallin, .., **GK** et al 2109.00546;

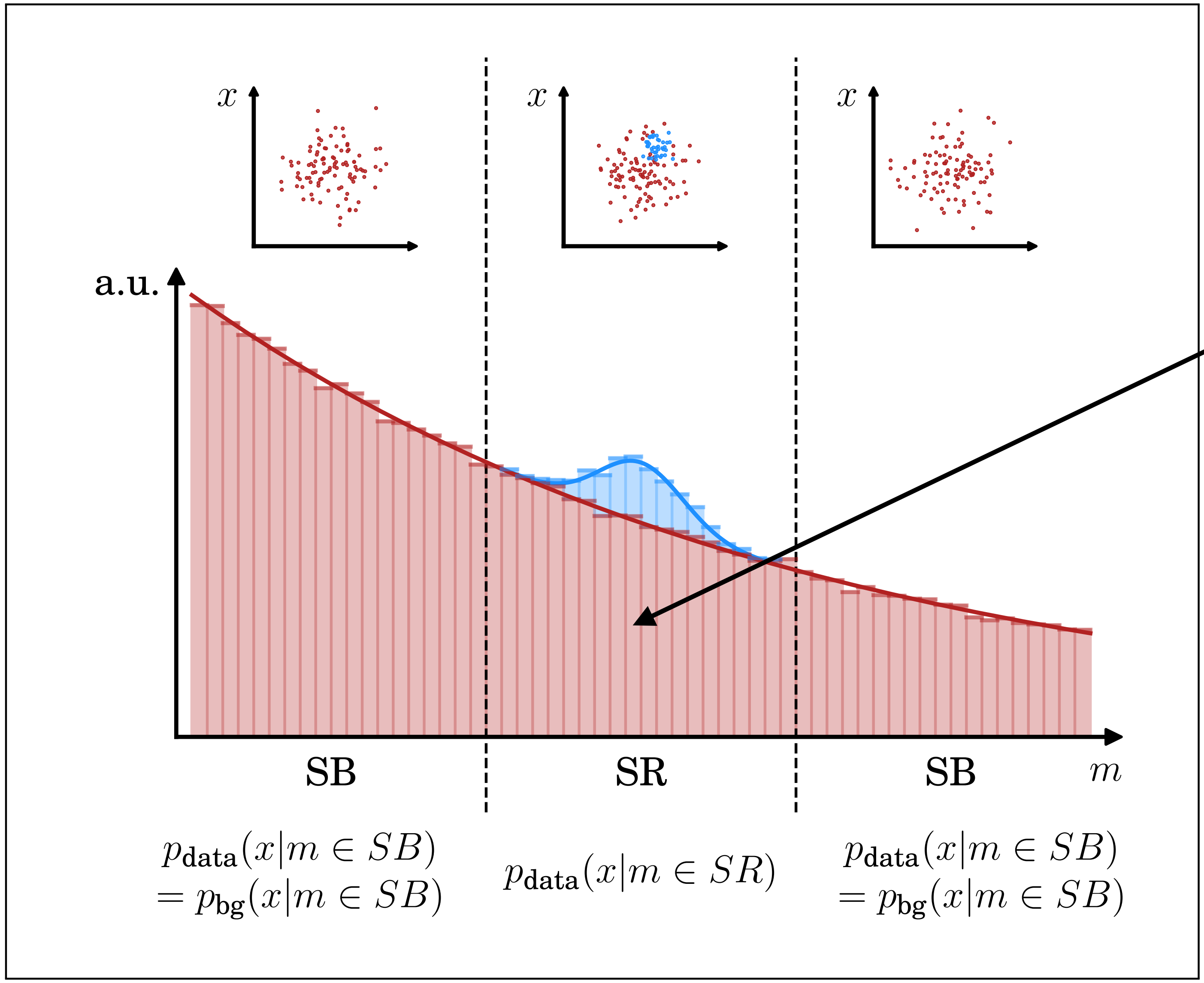
CATHODE



Train generative model conditional on m

GK, Nachmann, Shih et al 2101.08320;
Hallin, .., **GK** et al 2109.00546;

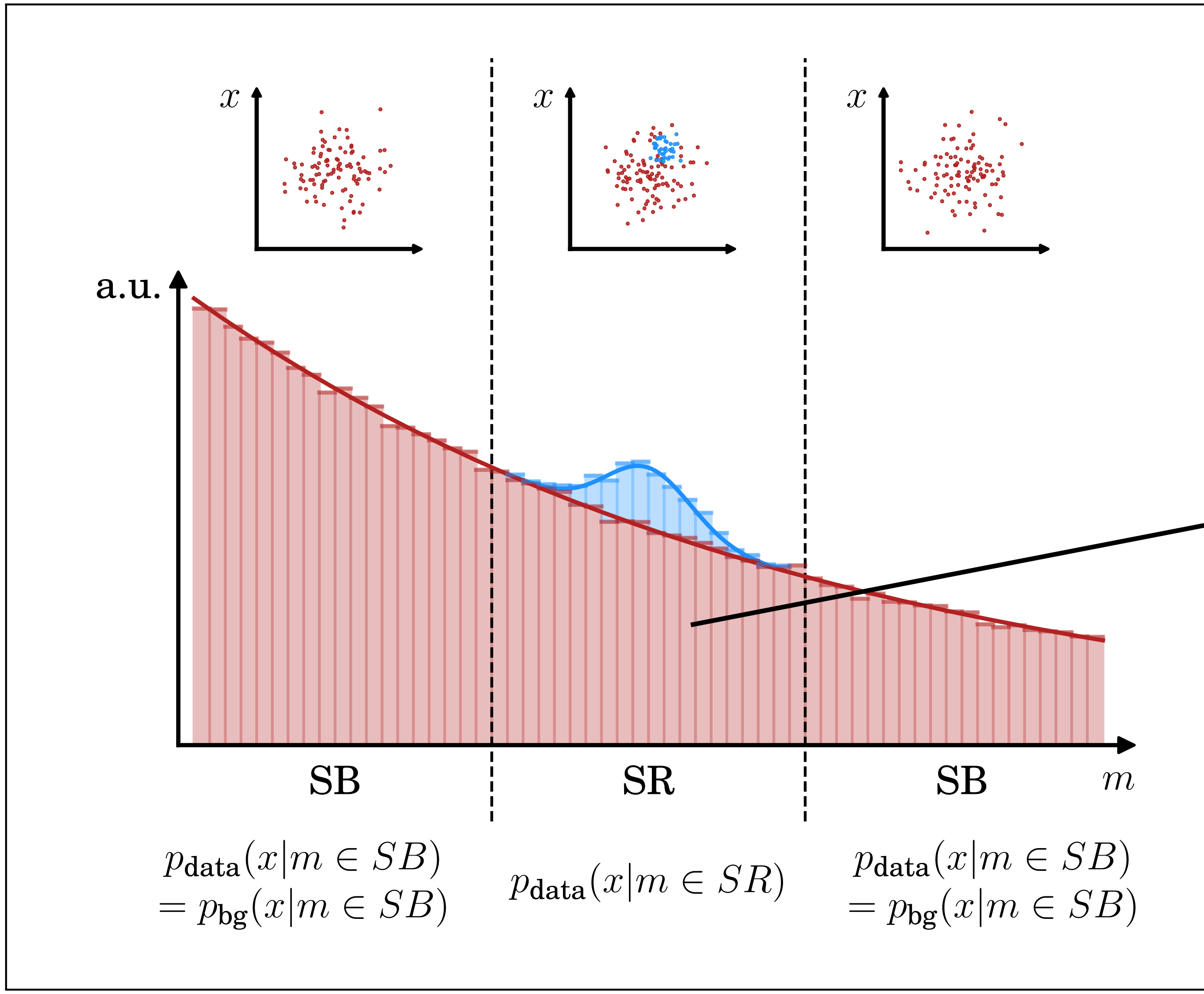
CATHODE



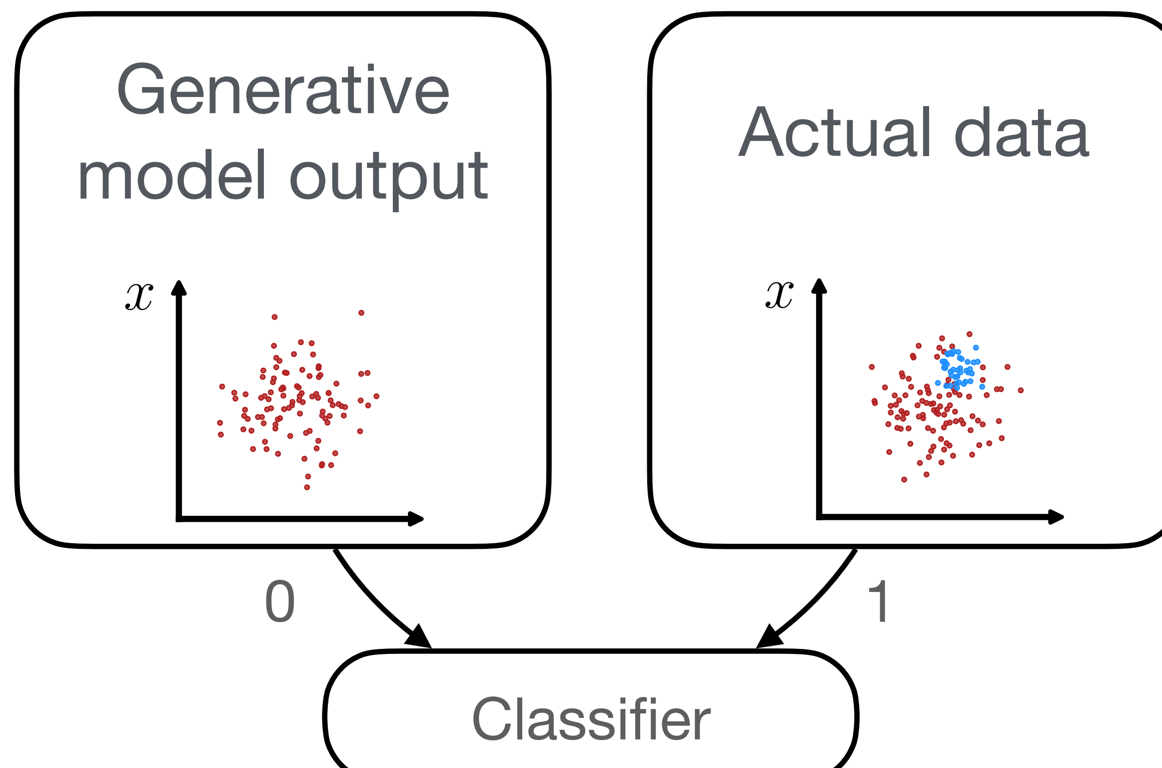
Interpolate &
and sample here

GK, Nachmann, Shih et al 2101.08320;
Hallin, .., **GK** et al 2109.00546;

CATHODE

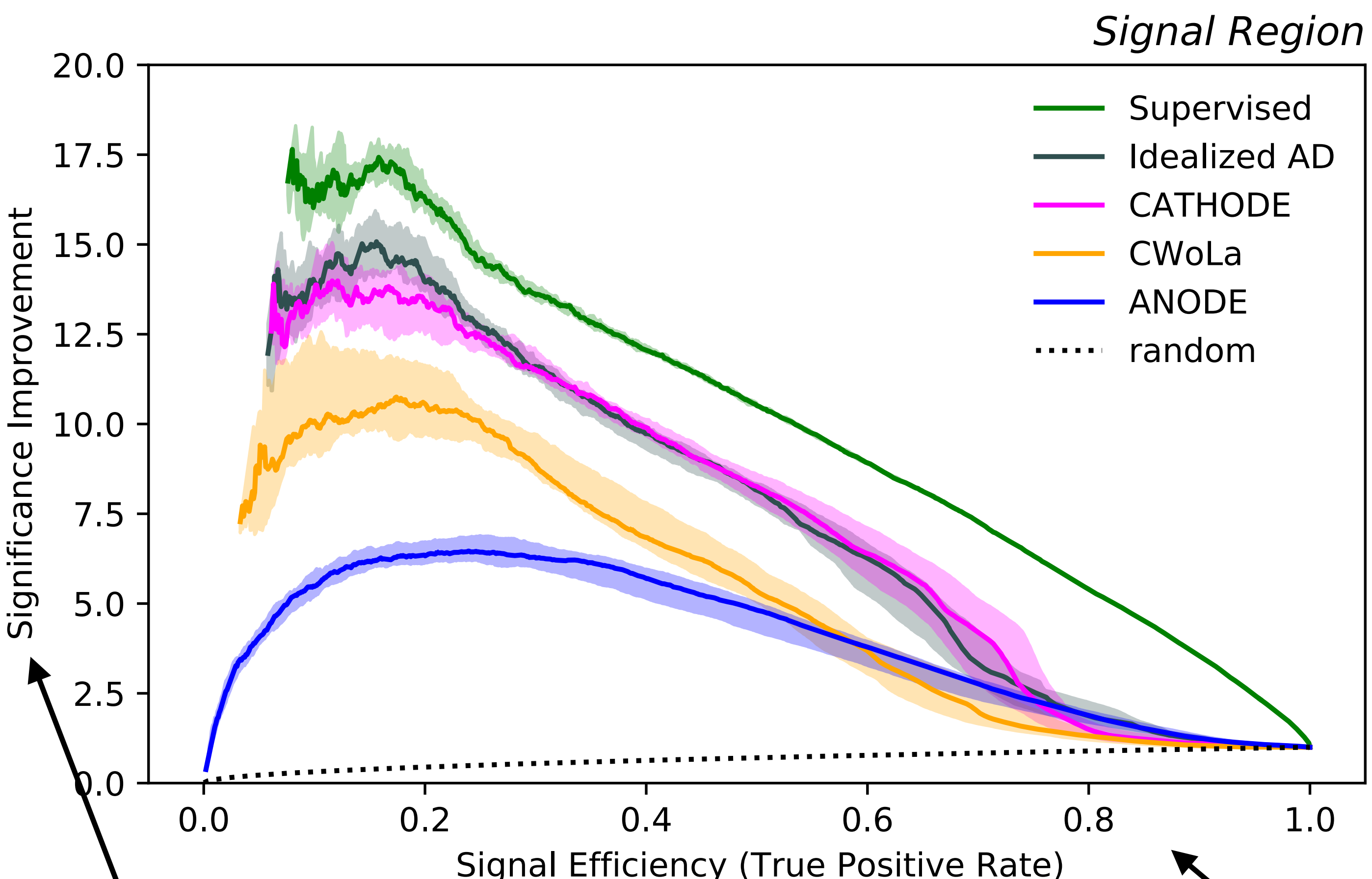


Train a classifier between
prediction vs data



GK, Nachmann, Shih et al 2101.08320;
Hallin, .., **GK** et al 2109.00546;

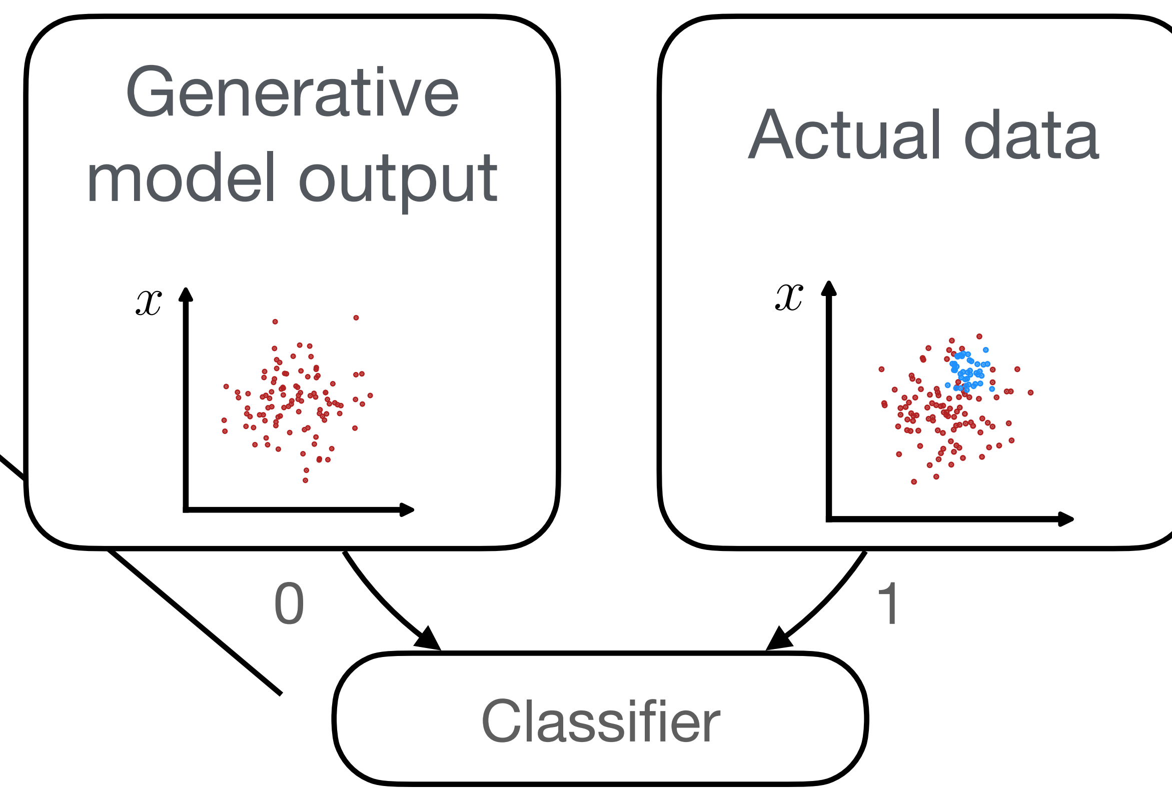
CATHODE



$$\mathrm{SIC} = \frac{\epsilon_S}{\sqrt{\epsilon_B}}$$

Use classifier to identify anomalies

Promising, but does this work on data?



Aside: CASE

Available on the CERN CDS information server **CMS PAS EXO-22-026**

CMS Physics Analysis Summary

Contact: cms-pag-conveners-exotica@cern.ch 2024/03/20

Model-agnostic search for dijet resonances with anomalous jet substructure in proton-proton collisions at $\sqrt{s} = 13$ TeV

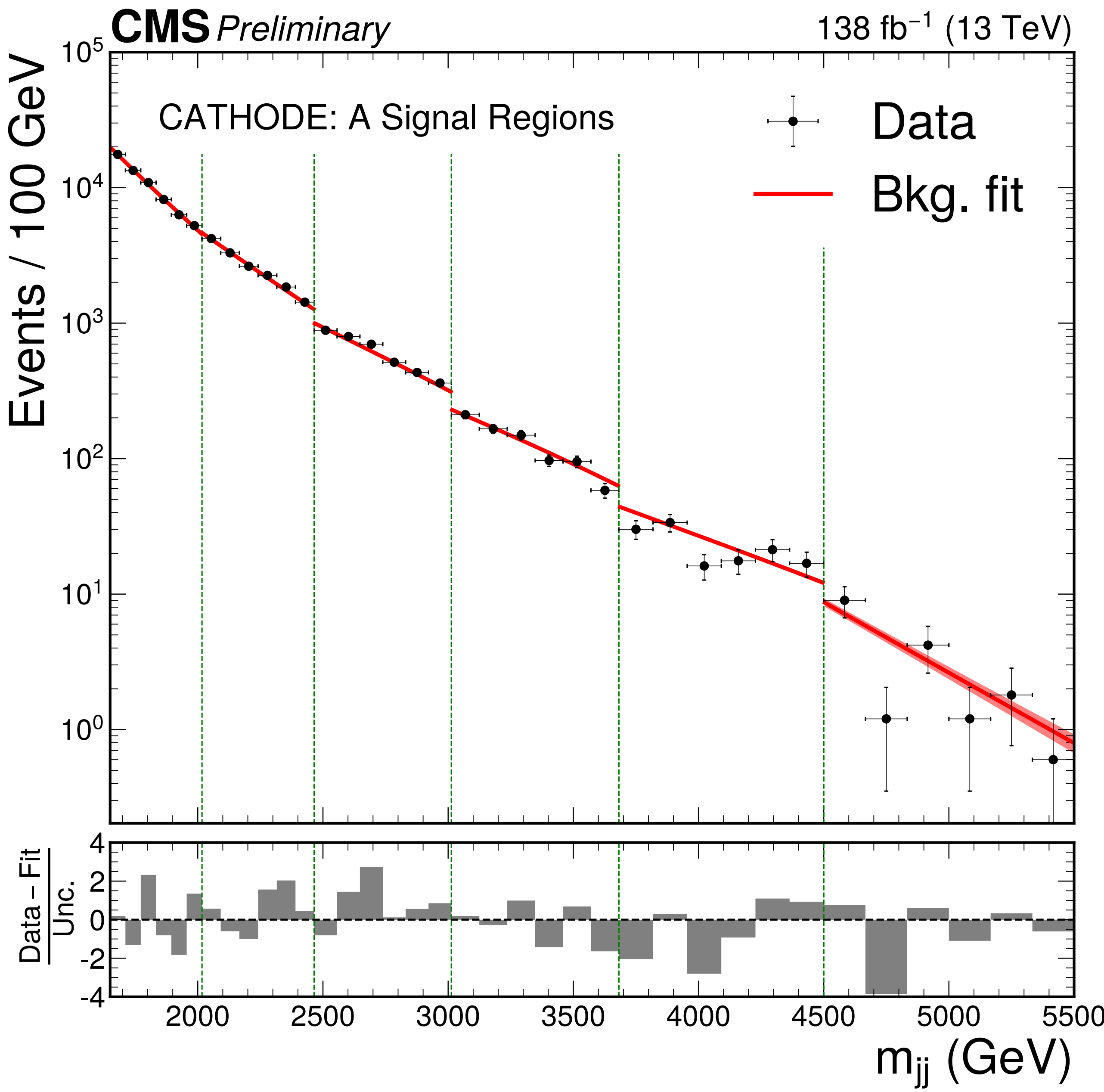
The CMS Collaboration

Abstract

This note introduces a model-agnostic search for new physics in the dijet final state. Other than the requirement of a narrow dijet resonance with a mass in the range of 1800-6000 GeV, minimal additional assumptions are placed on the signal hypothesis. Search regions are obtained by utilizing multivariate machine learning methods to select jets with anomalous substructure. A collection of complementary anomaly detection methods – based on unsupervised, weakly-supervised and semi-supervised algorithms – are used in order to maximize the sensitivity to unknown new physics signatures. These algorithms are applied to data corresponding to an integrated luminosity of 138 fb^{-1}, recorded in the years 2016 to 2018 by the CMS experiment at the LHC, at a centre-of-mass energy of 13 TeV. No significant excesses above background expectation are seen, and exclusion limits are derived on the production cross section of benchmark signal models varying in resonance mass, jet mass and jet substructure. Many of these signatures have not previously been searched for at the LHC, making the limits reported on the corresponding benchmark models the first ever and the most stringent to date.

- New result by the CMS collaboration
- Full Run 2 dataset
- 6 anomaly detectors in parallel

Aside: CASE



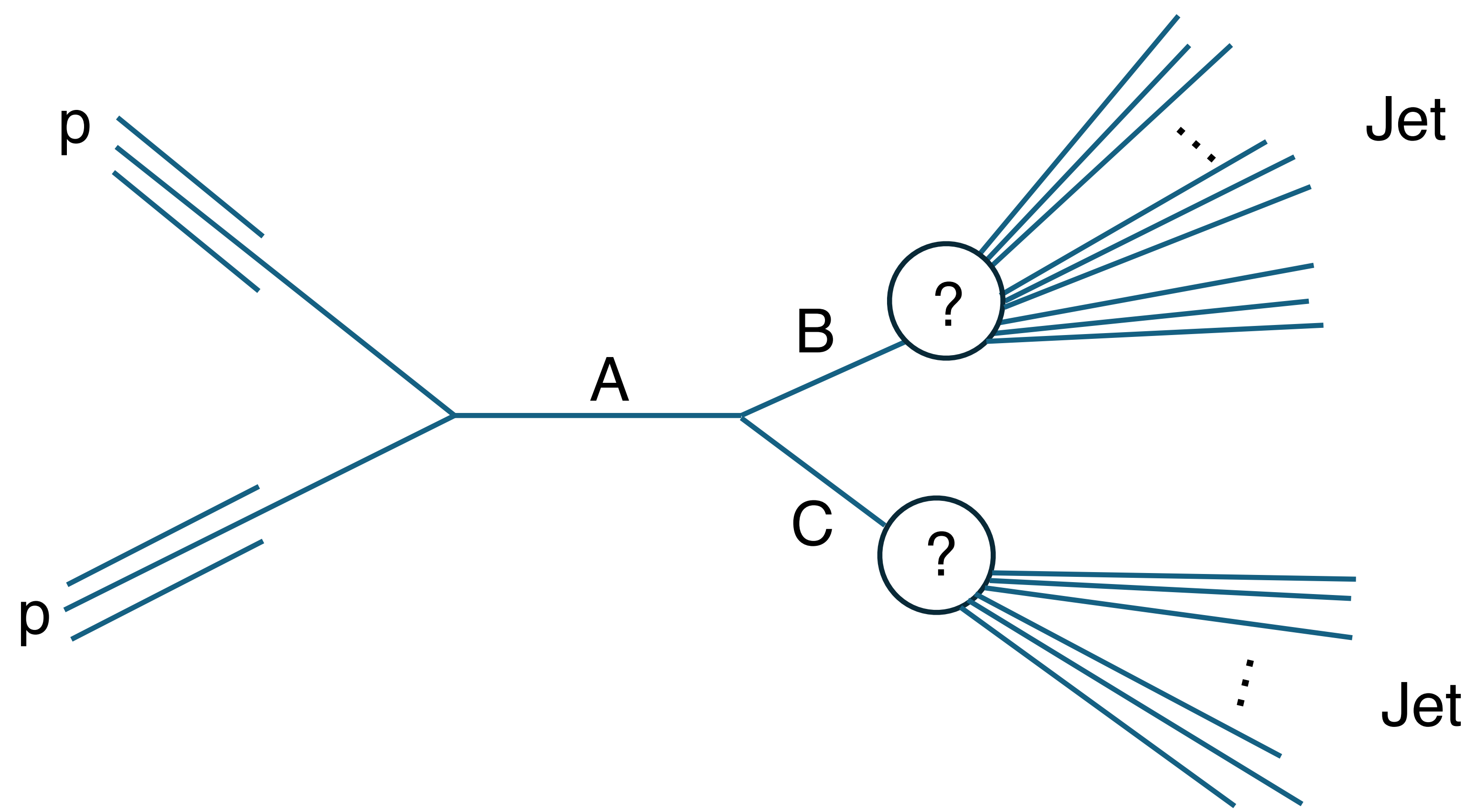
- New result by the CMS collaboration
- Full Run 2 dataset
- 6 anomaly detectors in parallel

Fully train CATHODE on data

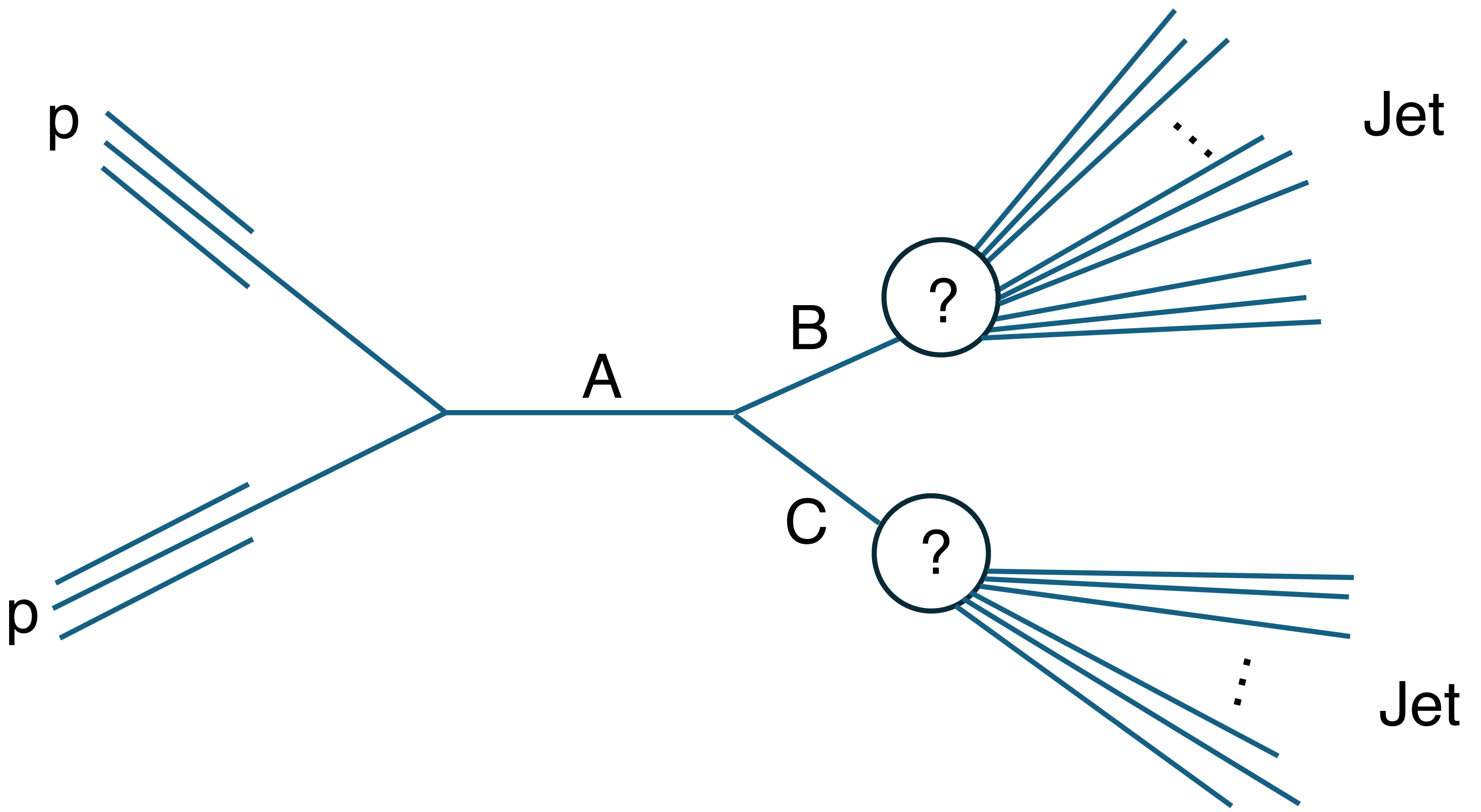
Select top 1% most anomalous events, perform bump-hunt

No signal-like outlier: set limits

What’s Beyond?



What's Beyond?



More features

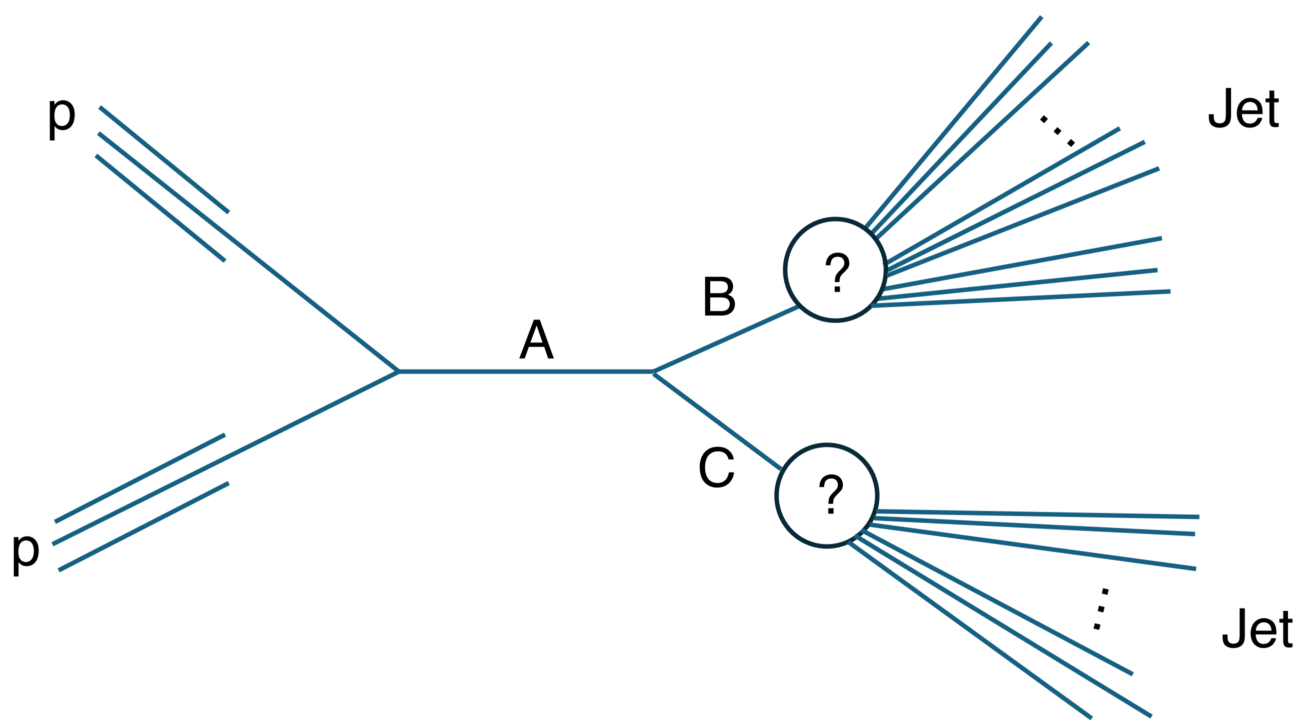
Better generative models / classifiers

Other topologies

Non-resonant features

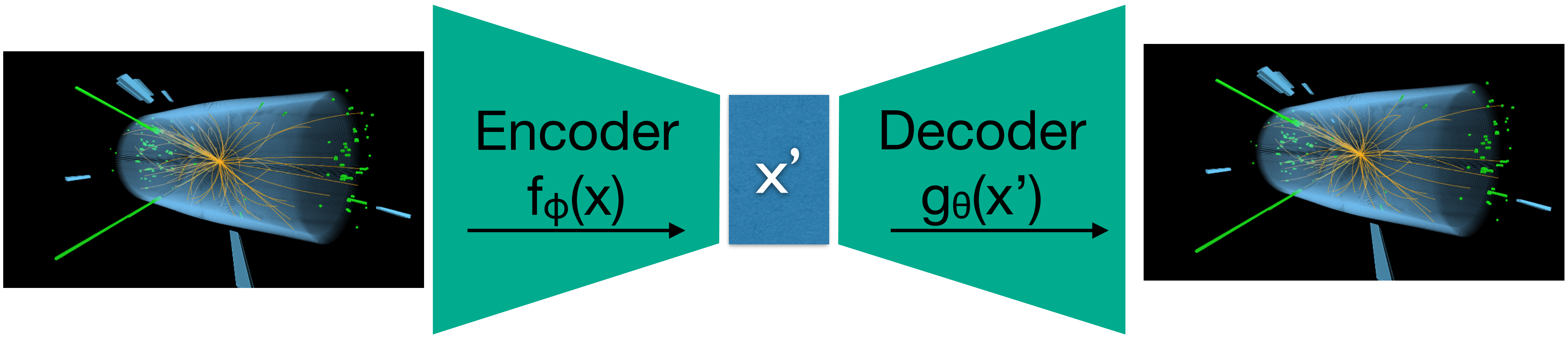
Outliers

Outliers



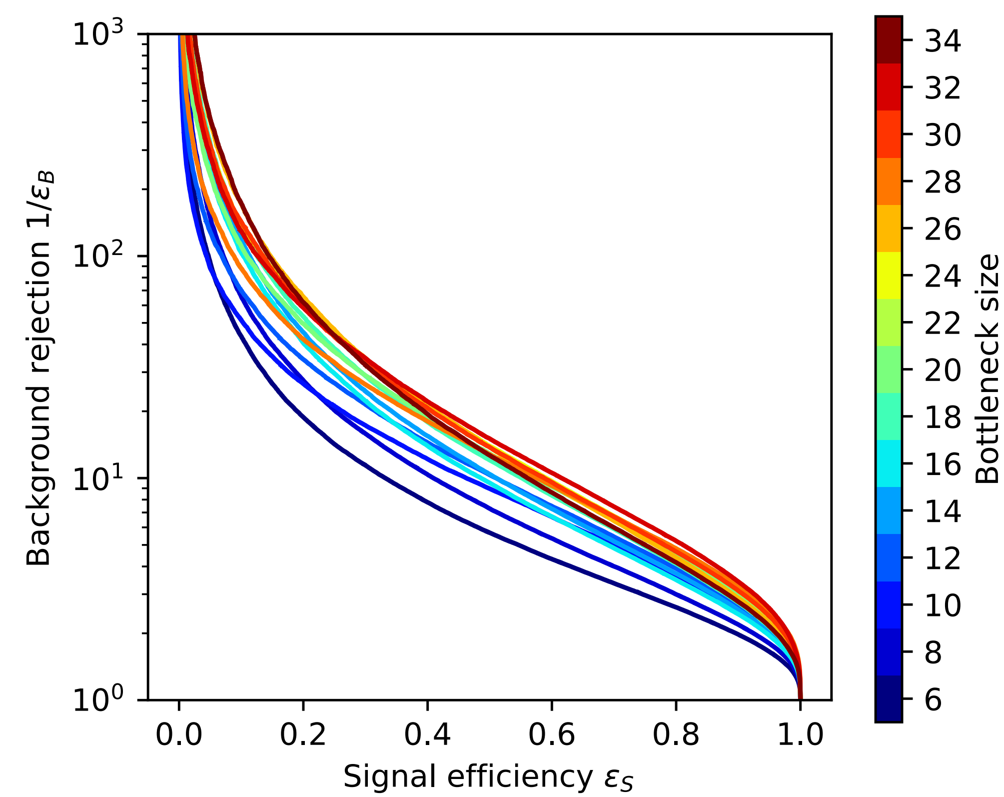
What if the events are even weirder?

Autoencoders



Learn-compression/decompression on signal free sample and use as anomaly score

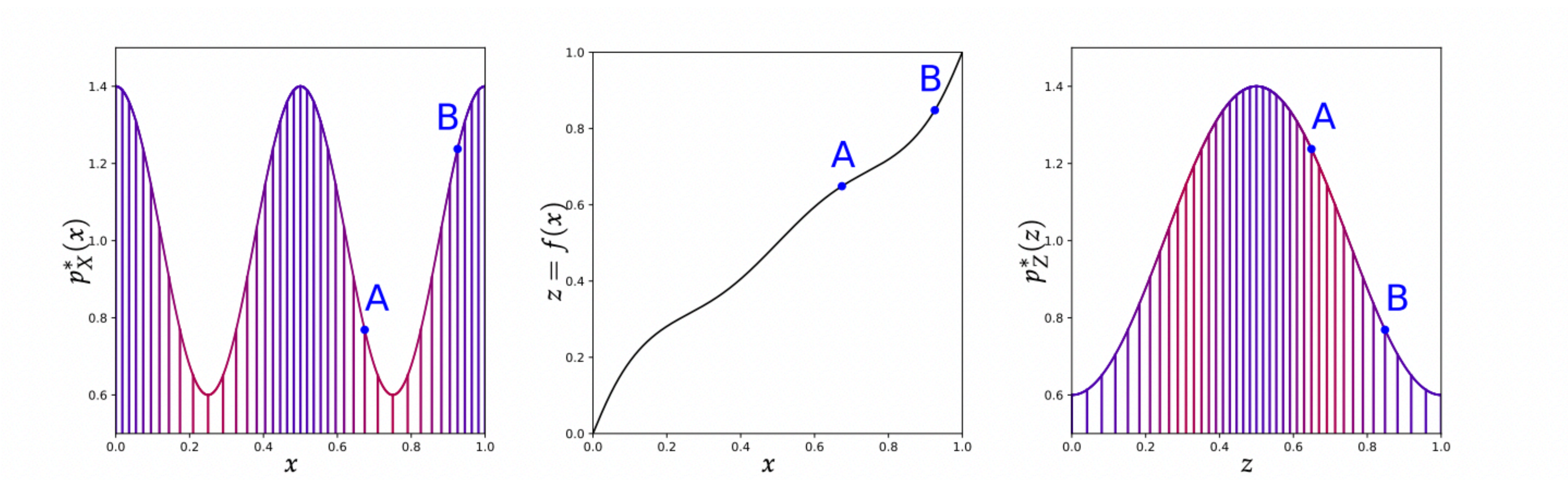
$$\mathcal{L}(x) = ||x - g_\theta(f_\phi(x))||$$



Heimel, **GK**, et al 1808.08979; Farina et al 1808.08992;

Autoencoders

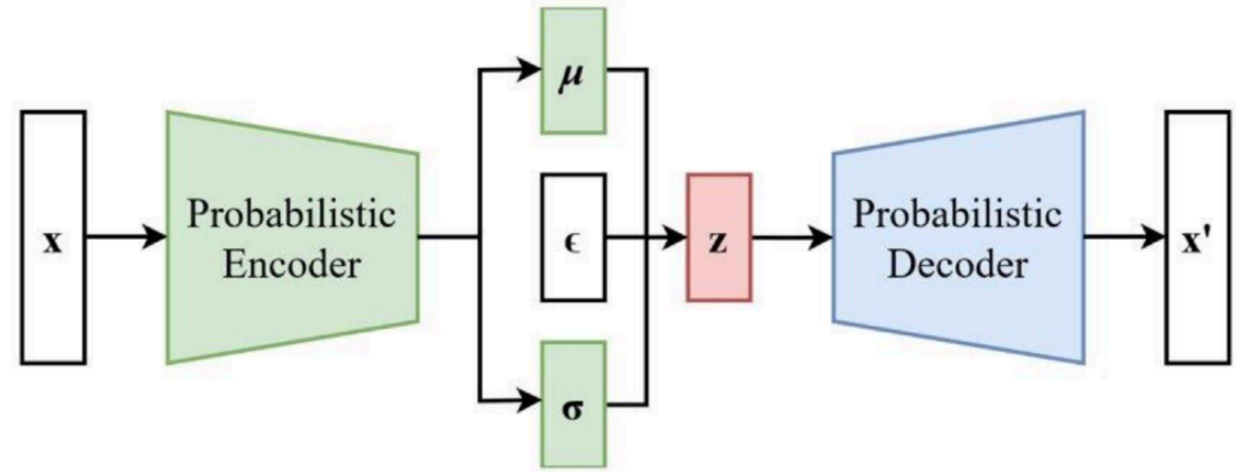
- Potential issue of models that only use $p_{\text{Background}}$ (e.g. autoencoders)

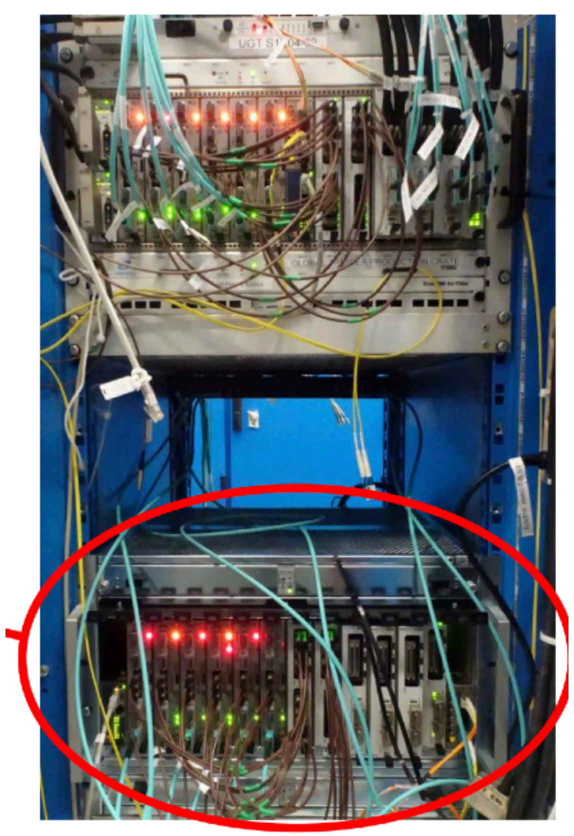


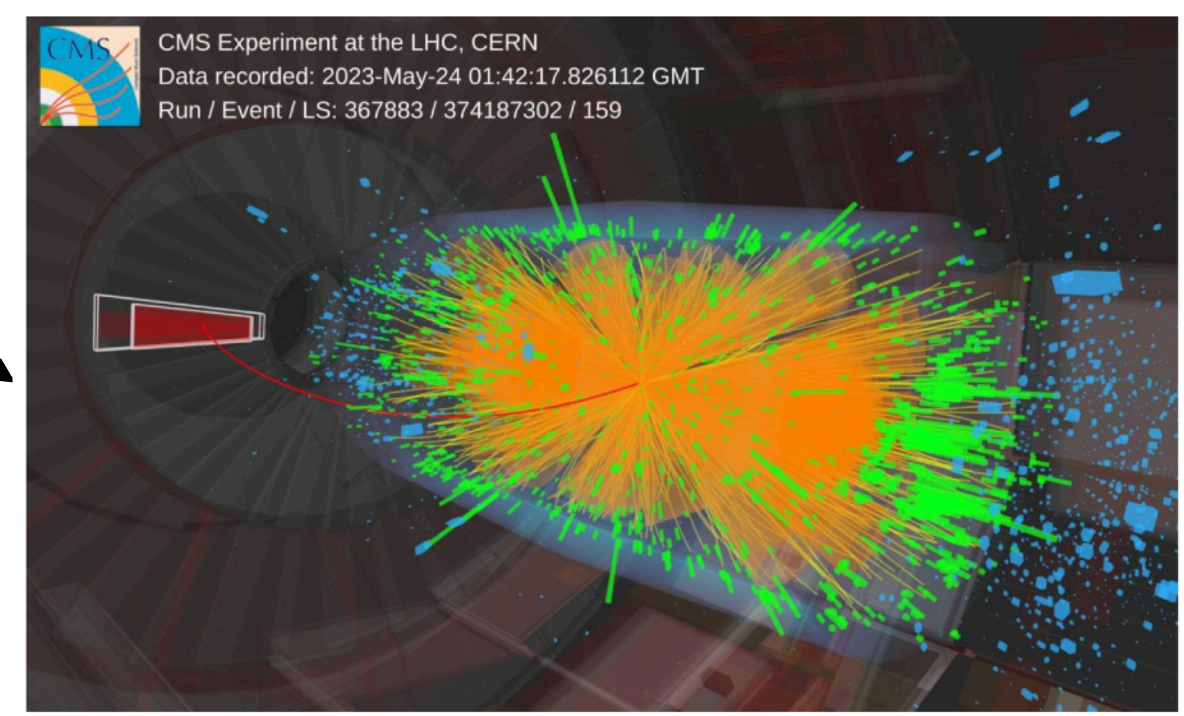
Example: Triggering Outliers

Learn-compression/decompression on signal free sample and use as anomaly score

Now testing in CMS Level 1 trigger







https://indico.cern.ch/event/1283970/contributions/5554350/attachments/2720710/4727877/axol1tl_fastml.pdf

Comments on Anomalies

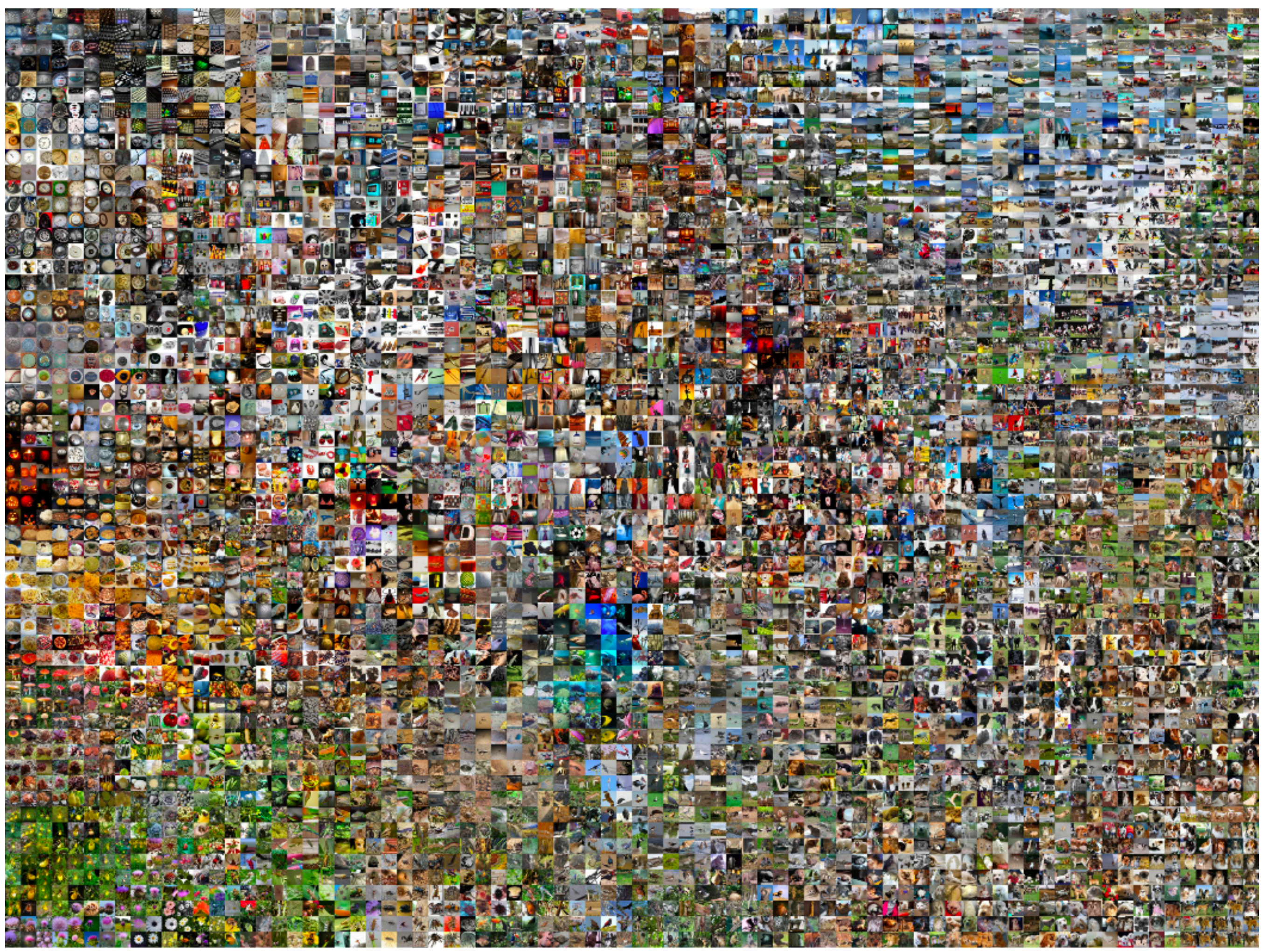
- Systematic anomaly searches as fail-safe strategy to signature based searches
- Trade off between sensitivity and breadth
- Weak supervision as useful technique beyond anomalies

Plan

- Lecture 1:
  - Basics of Neural Networks
  - Classification example: Top Tagging
- Lecture 2:
  - Anomaly Detection
  - **Generative Models**

Motivation

Have: input examples (collision events, detector readouts, …)



Want: more data

Specifically: new data similar to the input, but not exact copies

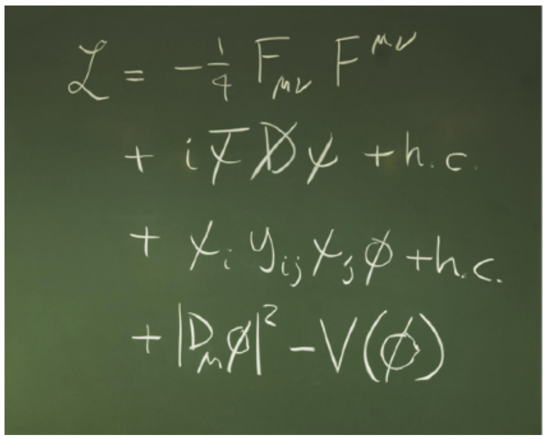
How to encode in neural net?

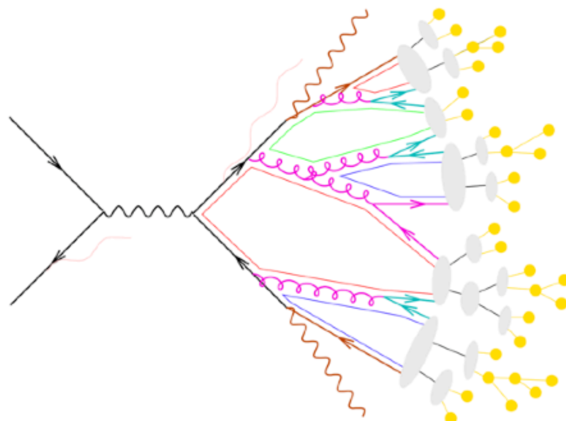
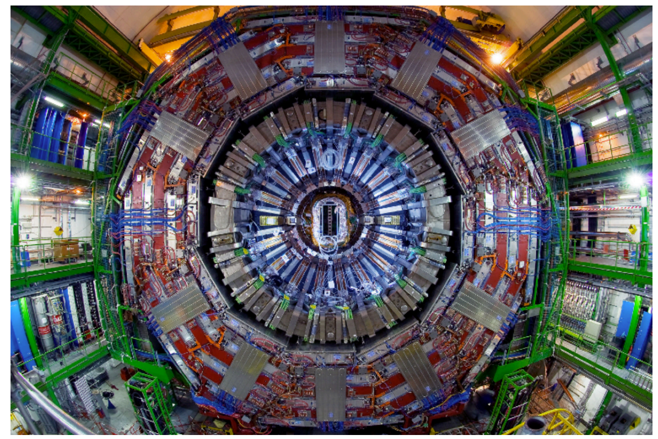
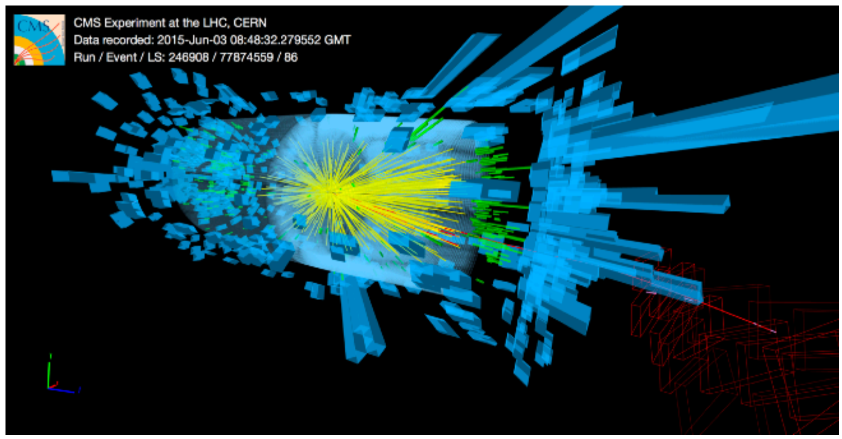
Uses:

- Simulations
- In-situ background estimation
- Surrogate models
- …

Generative Models

This happens in the experiment



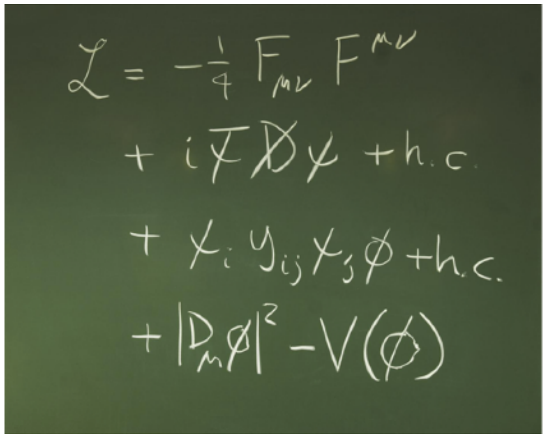




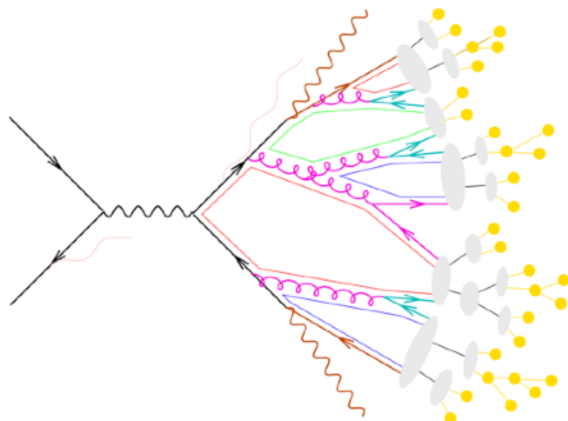
This is what we want to know

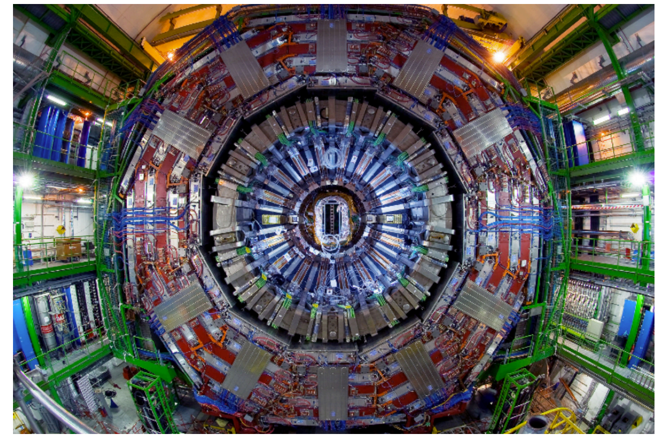
Simulation is crucial to connect experimental data with theory predictions

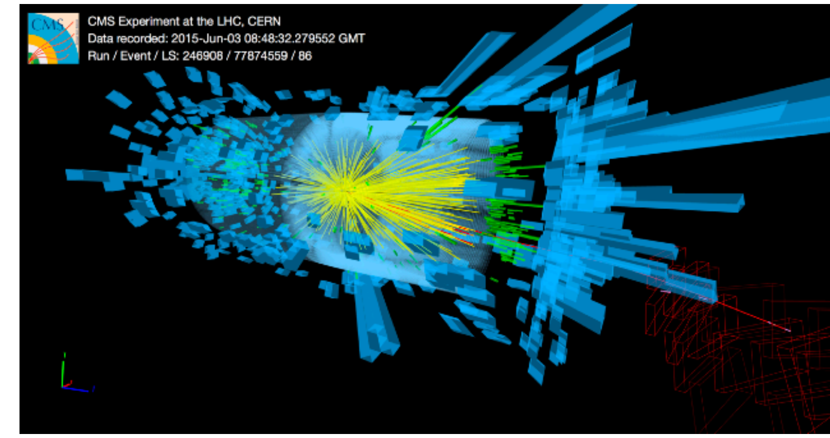
Generative Models

This happens in the experiment



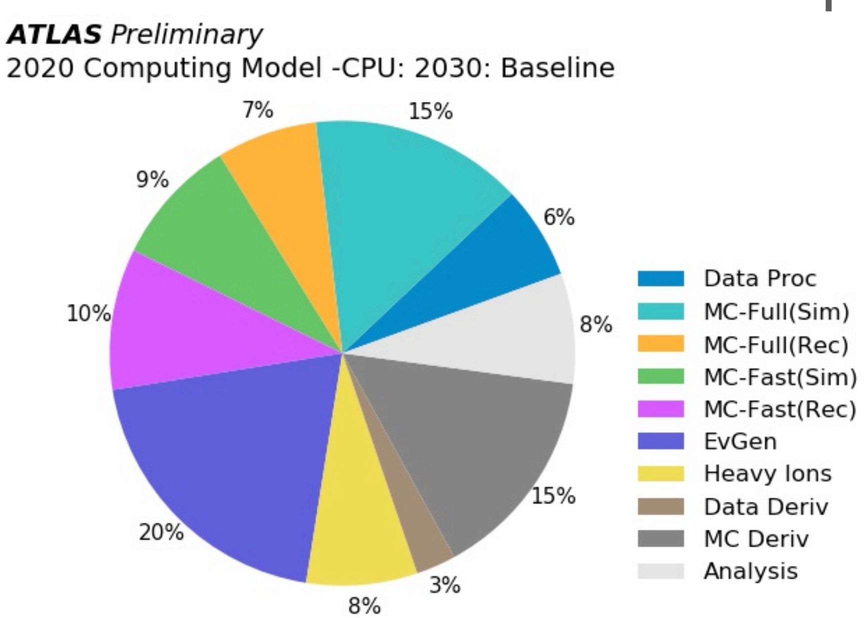


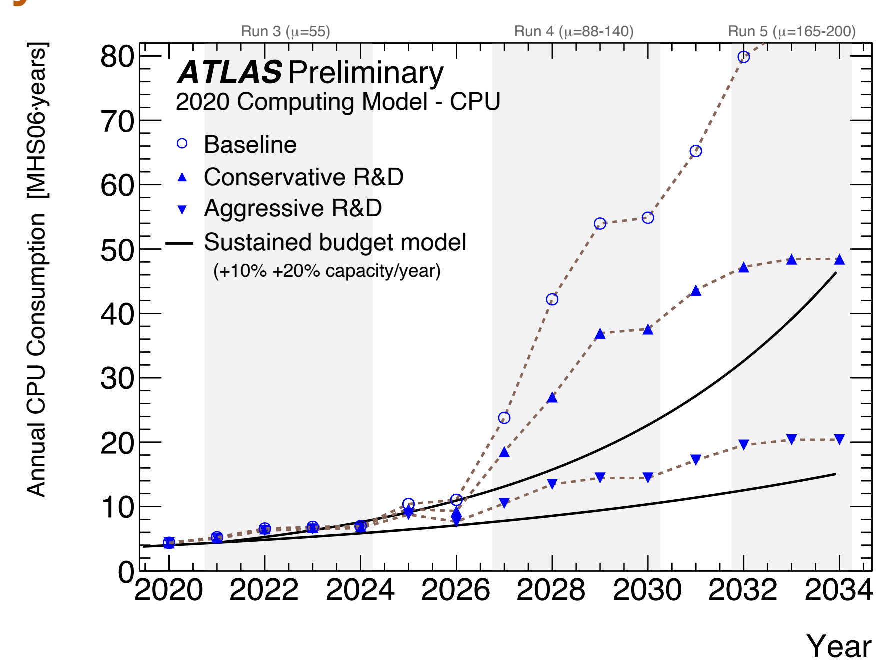




This is what we want to know

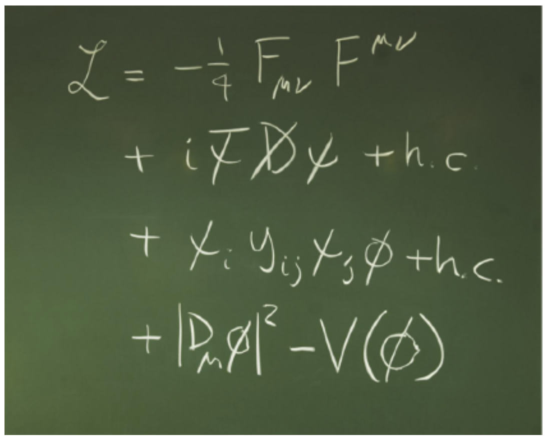
Simulation is crucial to connect experimental data with theory predictions, but computationally very costly

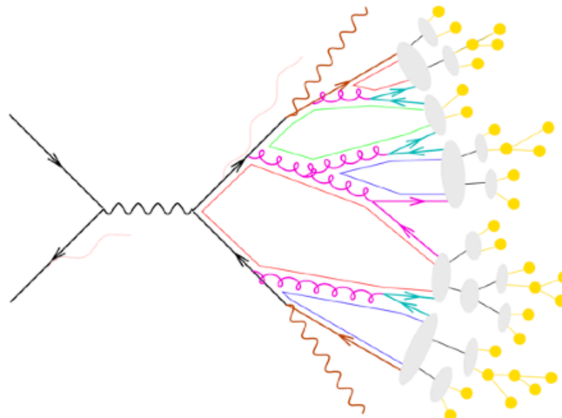


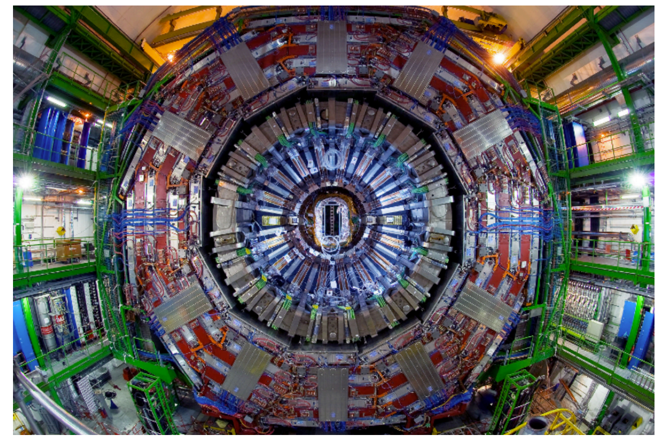


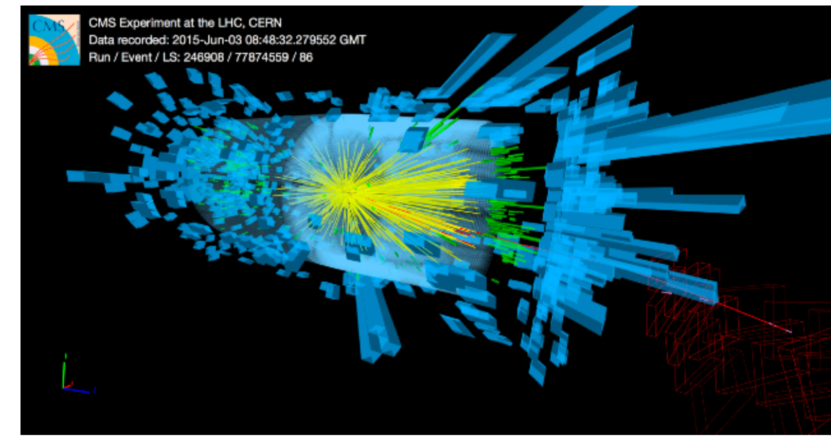
Generative Models

This happens in the experiment









This is what we want to know

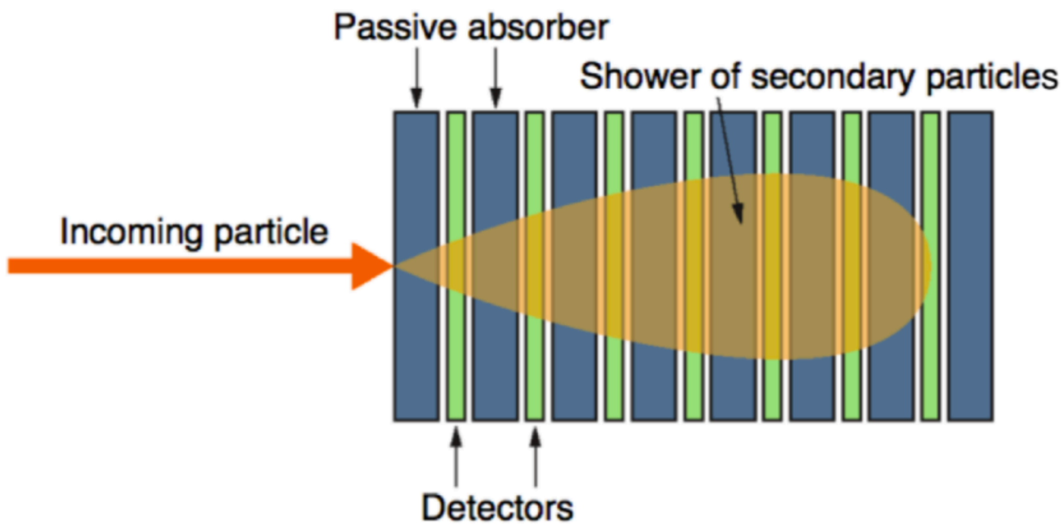
Simulation is crucial to connect experimental data with theory predictions, but computationally very costly

Use AI to improve efficiency of simulation codes or learn surrogates

Calorimeter Simulation

Interaction of particles with multi-layer detectors to determine their initial energy (and type)

Measurement of energy, position, (and time) of secondary particle hits

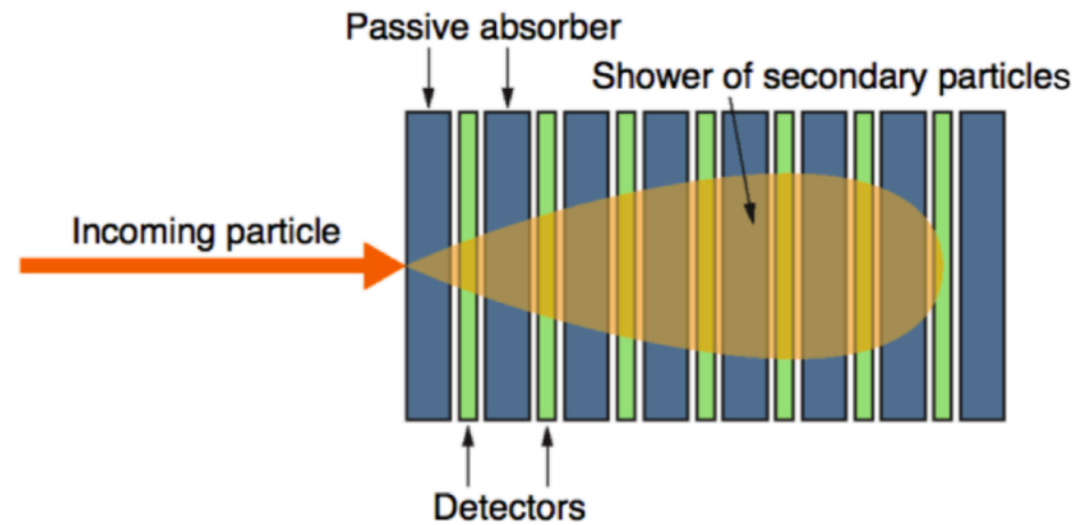


Calorimeter Simulation

Interaction of particles with multi-layer detectors to determine their initial energy (and type)

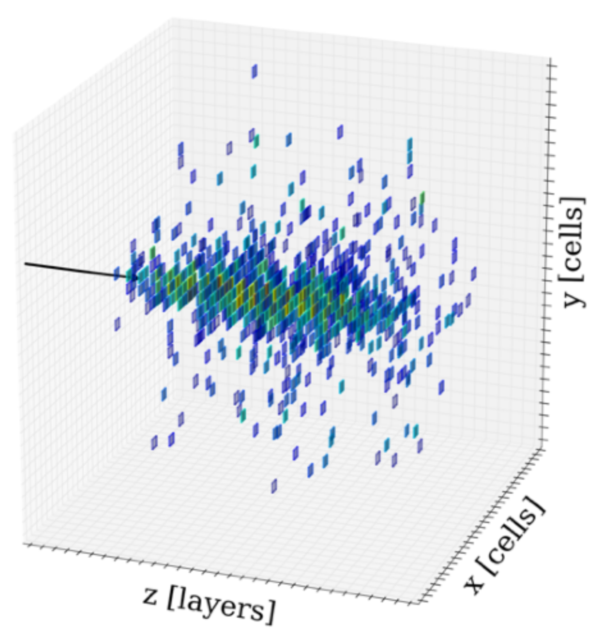
Measurement of energy, position, (and time) of secondary particle hits

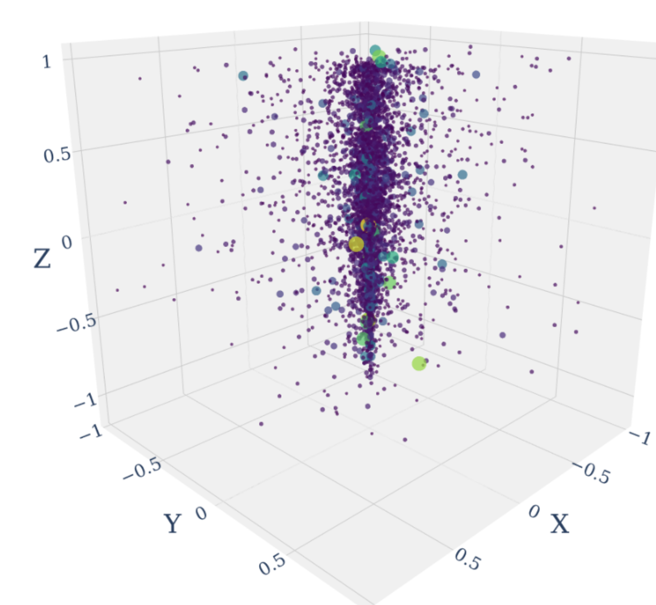
Represent data as
-fixed grid
(3d matrix of detector elements, value=energy)
-point cloud
(set of hits, each hit is a 3d vector with position+energy)



as fixed grid

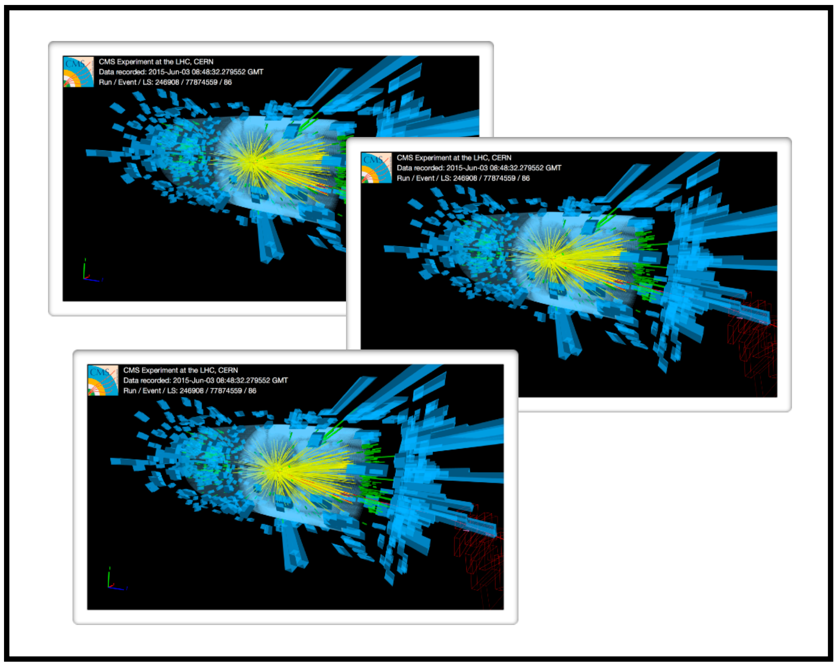
as point cloud





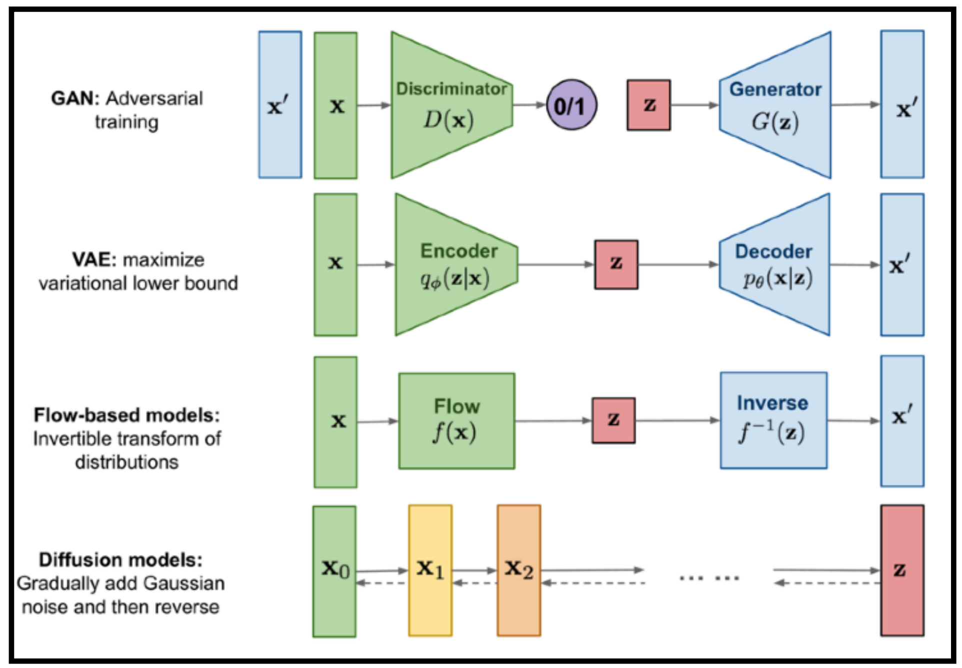
HEP Strategy

1. Use classical simulation or data as input

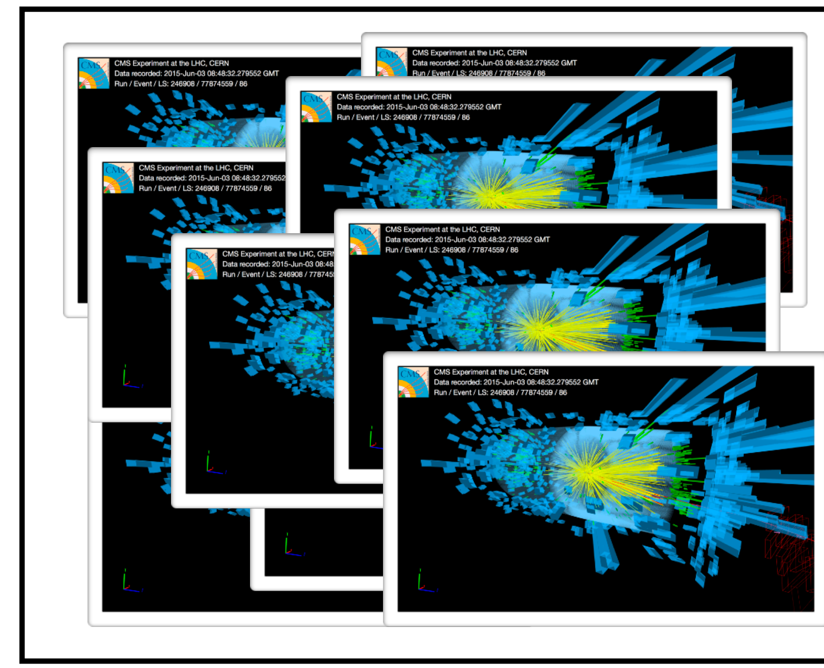


(slow)

2. Train generative surrogate



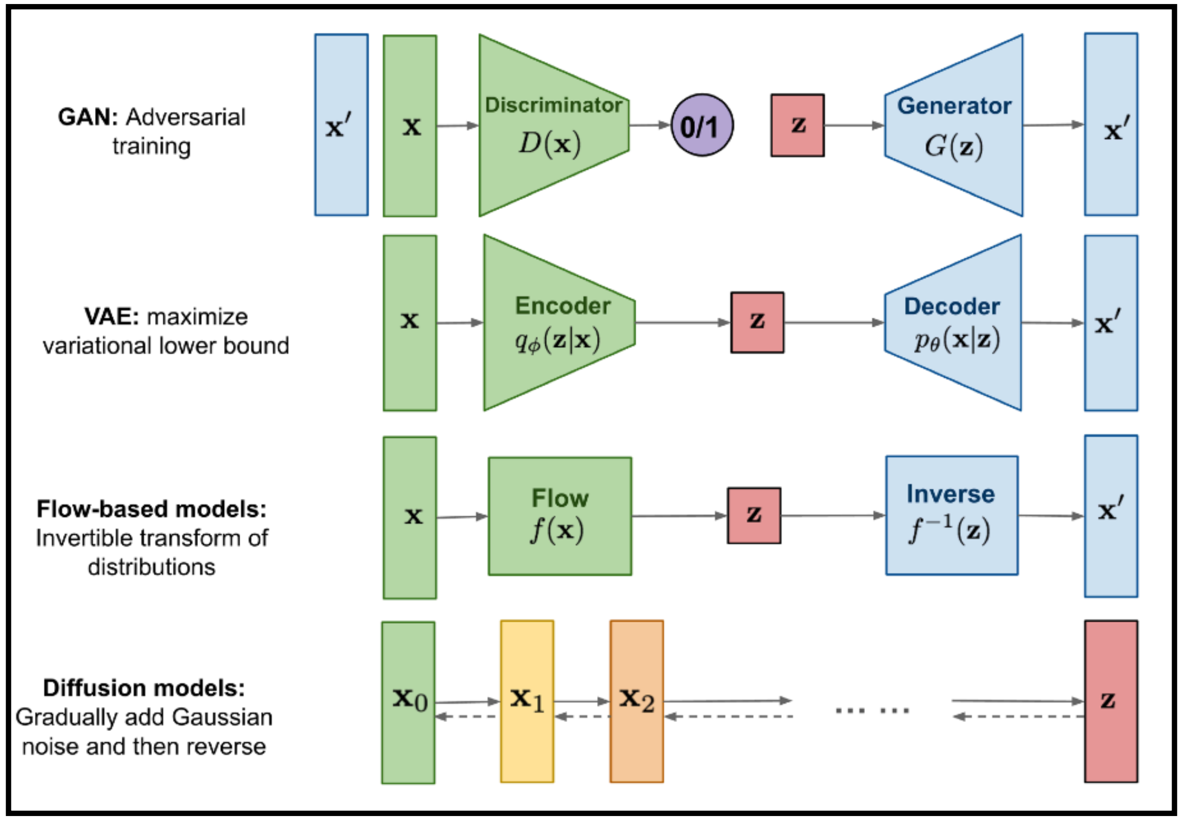
3. Oversample



(fast)

Paganini, Oliveira, Nachman 1705.02355; Butter, Diefenbacher, **GK**, et al 2008.06545;

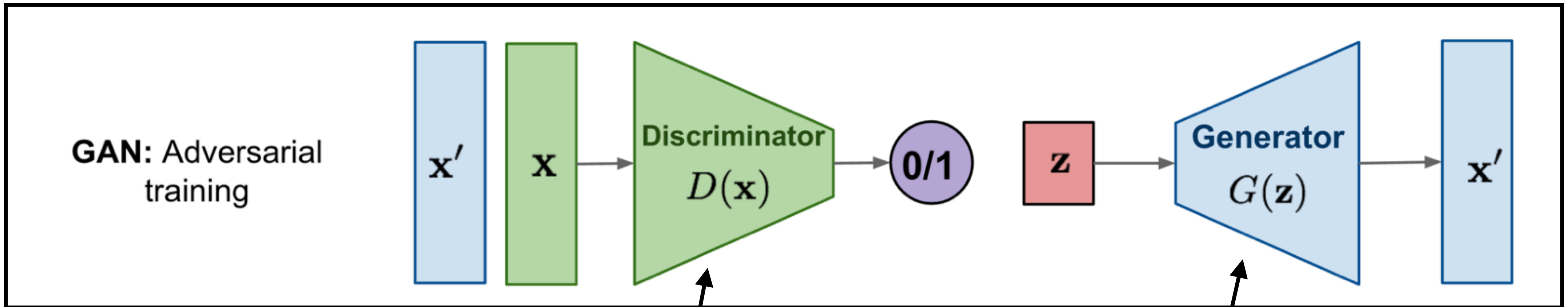
Overview



GAN: Adversarial training
$\mathbf{x}'$
$\mathbf{x}$
Discriminator $D(\mathbf{x})$
0/1
$\mathbf{z}$
Generator $G(\mathbf{z})$
$\mathbf{x}'$
VAE: maximize variational lower bound
$\mathbf{x}$
Encoder $q_\phi(\mathbf{z}|\mathbf{x})$
$\mathbf{z}$
Decoder $p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$
Flow-based models: Invertible transform of distributions
$\mathbf{x}$
Flow $f(\mathbf{x})$
$\mathbf{z}$
Inverse $f^{-1}(\mathbf{z})$
$\mathbf{x}'$
Diffusion models: Gradually add Gaussian noise and then reverse
$\mathbf{x}_0$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{z}$

Generative Adversarial Networks

Generative Adversarial Networks



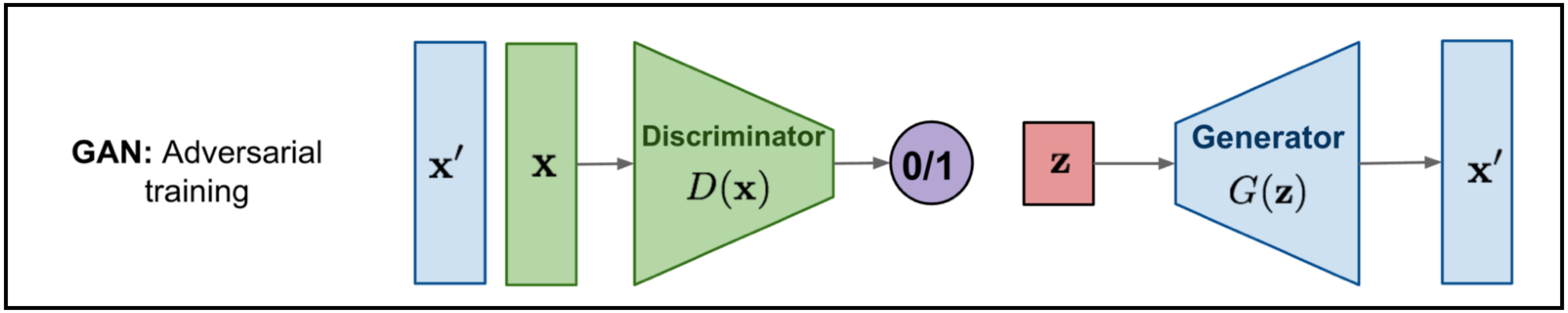
Generative Adversarial Networks (GANs) consist of 2 networks

Provides feedback on quality of examples

Maps random noise to realistic examples

1406.2661
lilianweng.github.io

Generative Adversarial Networks



Training objective:
Binary cross entropy

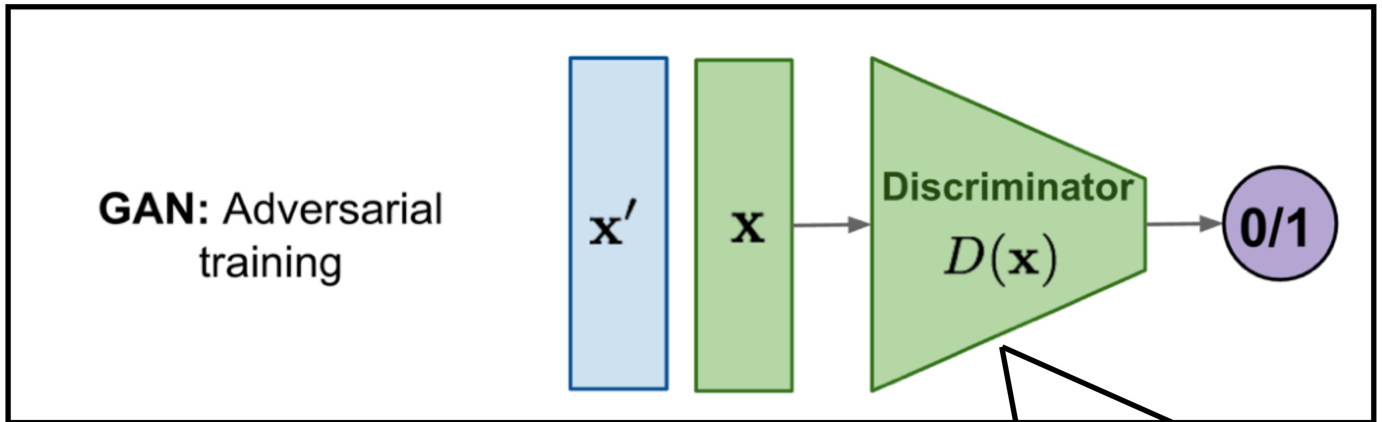
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

True examples

Fake examples

1406.2661

Generative Adversarial Networks



Training objective:

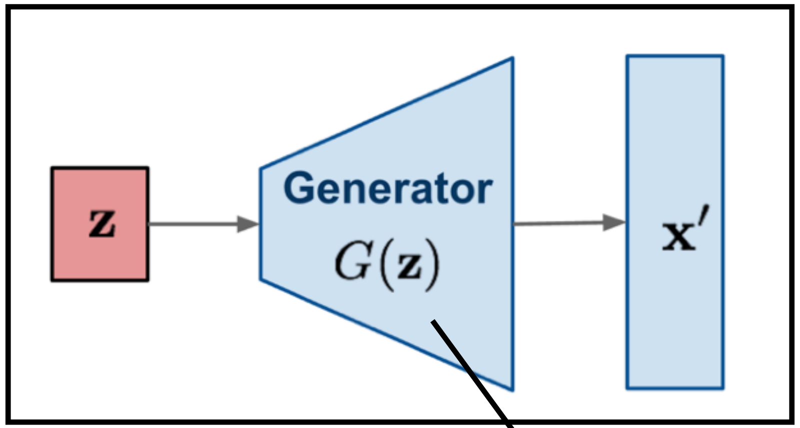
Binary cross entropy

Maximise for discriminator

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

1406.2661

Generative Adversarial Networks



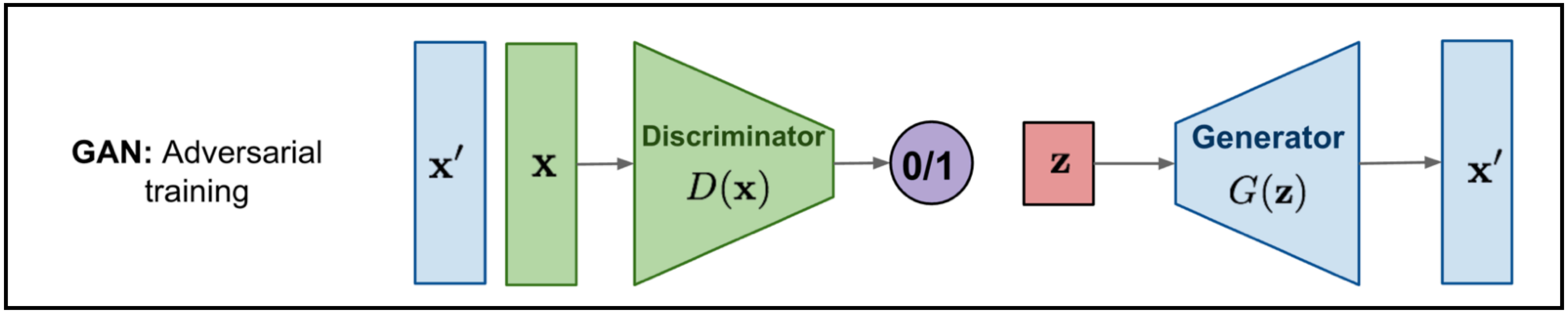
Training objective:
Binary cross entropy

Minimise for generator

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

1406.2661

Generative Adversarial Networks



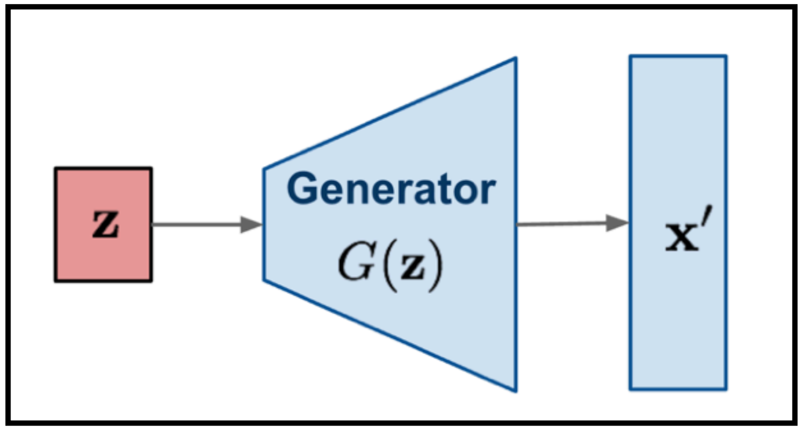
Training objective:
Binary cross entropy

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

At (Nash) equilibrium:
Generator produces realistic examples
Discriminator is maximally confused

1406.2661

Generative Adversarial Networks



Training objective:
Binary cross entropy

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

For generation:
Sample from Generator
Discard Discriminator

1406.2661

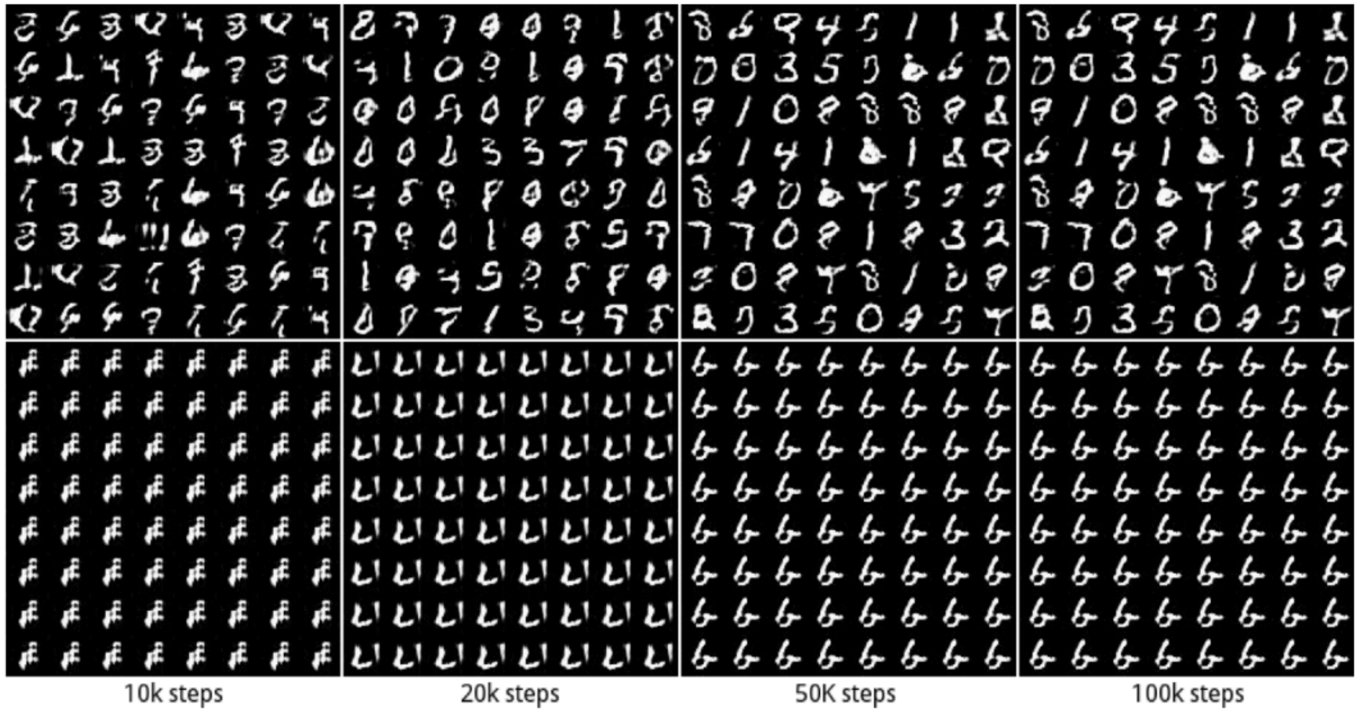
Comments on GANs

Architecture:

- Low complexity, fast and adaptable

Learning:

- Unstable training
- Matching of generator/discriminator (vanishing gradients)
- Mode collapse
- Loss function not interpretable



Mode collapse

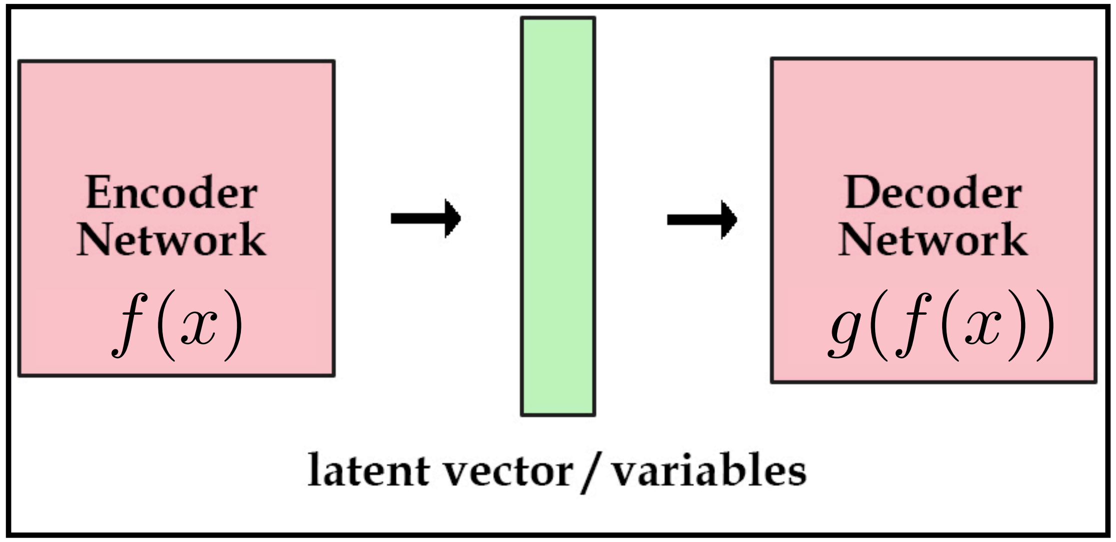
Maturity:

- Well established,
  many variants and extensions

1611.02163

Variational Autoencoders

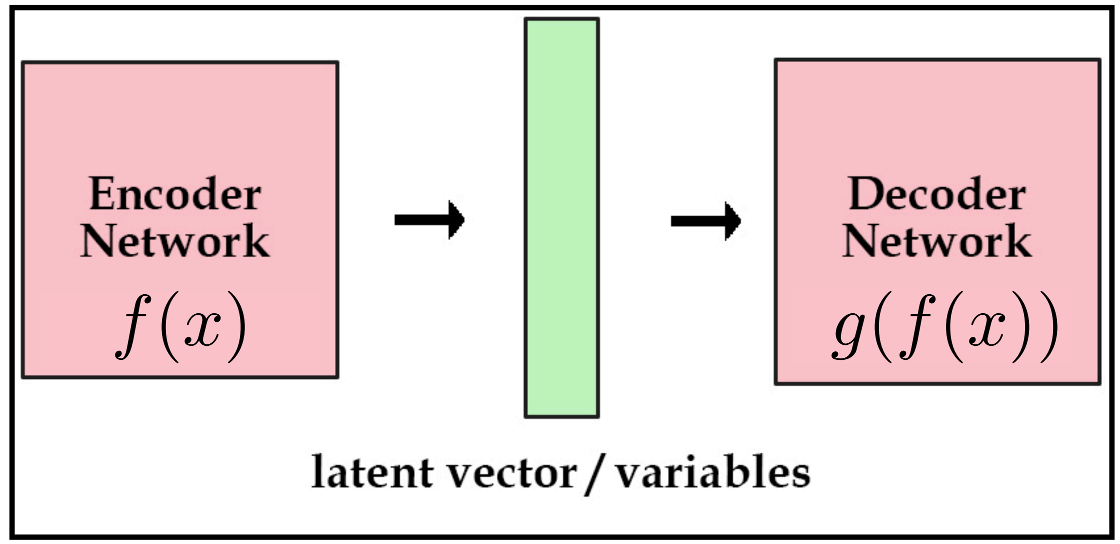
Autoencoder



Two networks
Encoder: data → latent space
Decoder: latent space → data

1312.6114
lilianweng.github.io

Autoencoder



Two networks
Encoder: data → latent space
Decoder: latent space → data

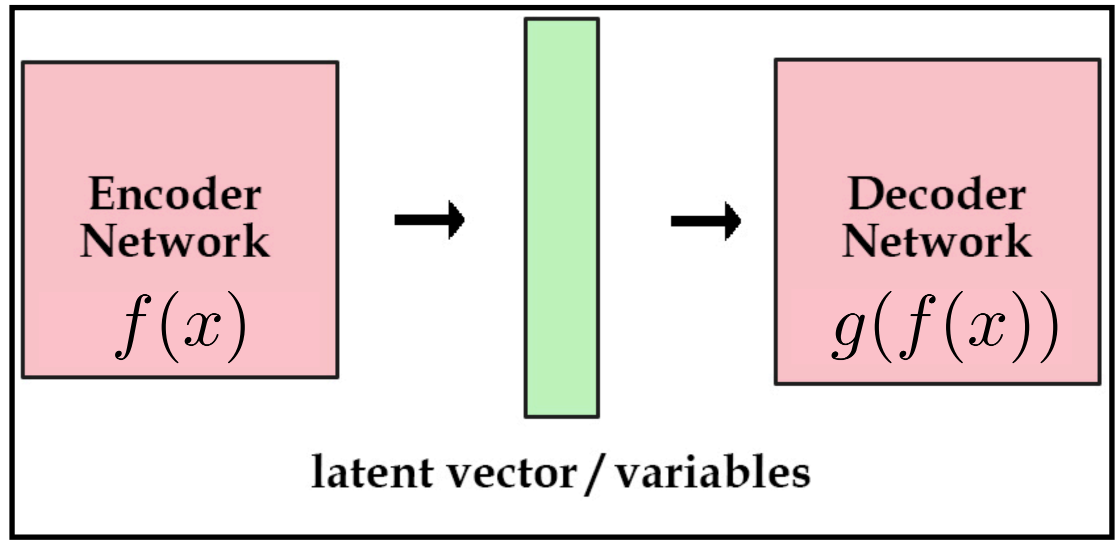
Training objective:
Minimise input/output difference

Decoder Encoder

$$L = (x - f(g(x)))^2$$

1312.6114

Autoencoder



Two networks
Encoder: data → latent space
Decoder: latent space → data

Training objective:
Minimise input/output difference

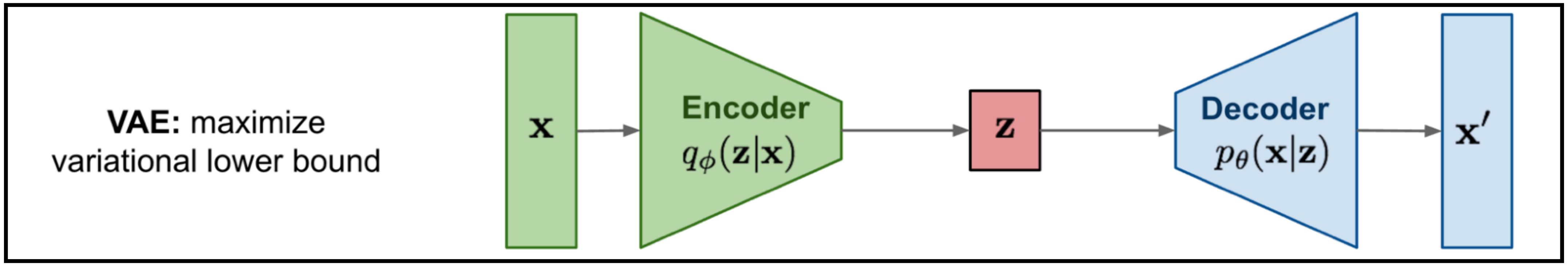
Decoder Encoder

$$L = (x - f(g(x)))^2$$

Uses:
Dimension reduction
Denoising
Anomaly detection
Generation?

1312.6114

Variational Autoencoder

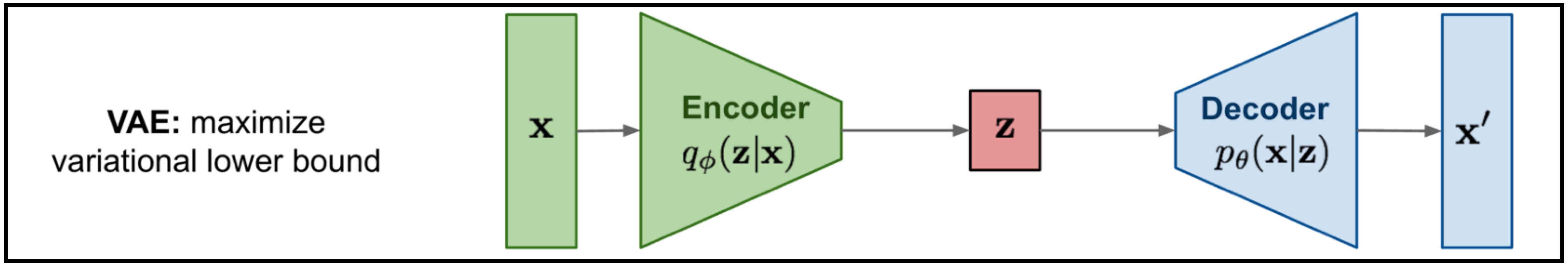


$$f(x) = (\mu, \sigma)$$

Variational Autoencoder (VAE):
Split latent space

1312.6114

Variational Autoencoder



Variational Autoencoder (VAE):
Split latent space
Sample before decoder

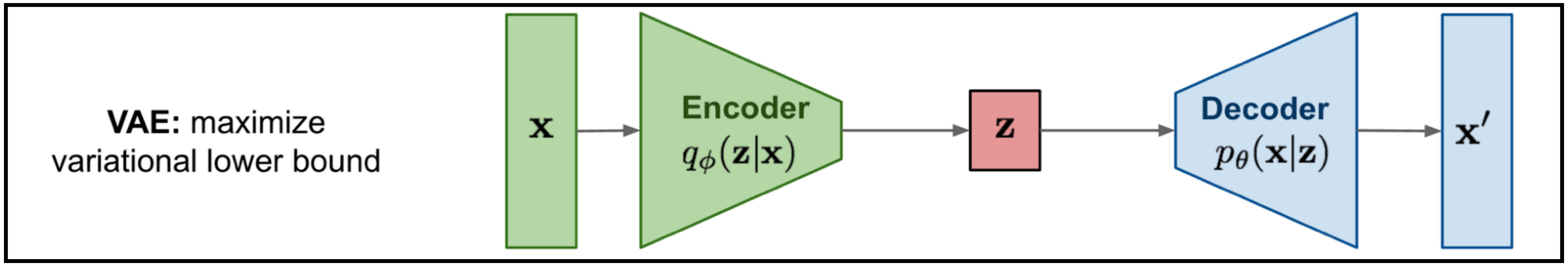
$$f(x) = (\mu, \sigma)$$

$$z = \text{Gaussian}(\mu, \sigma)$$

$$x' = g(z)$$

1312.6114

Variational Autoencoder



$$f(x) = (\mu, \sigma)$$

Variational Autoencoder (VAE):
Split latent space
Sample before decoder
Penalty so mean/std are close to unit Gaussian

$$z = \mathrm{Gaussian}(\mu, \sigma)$$

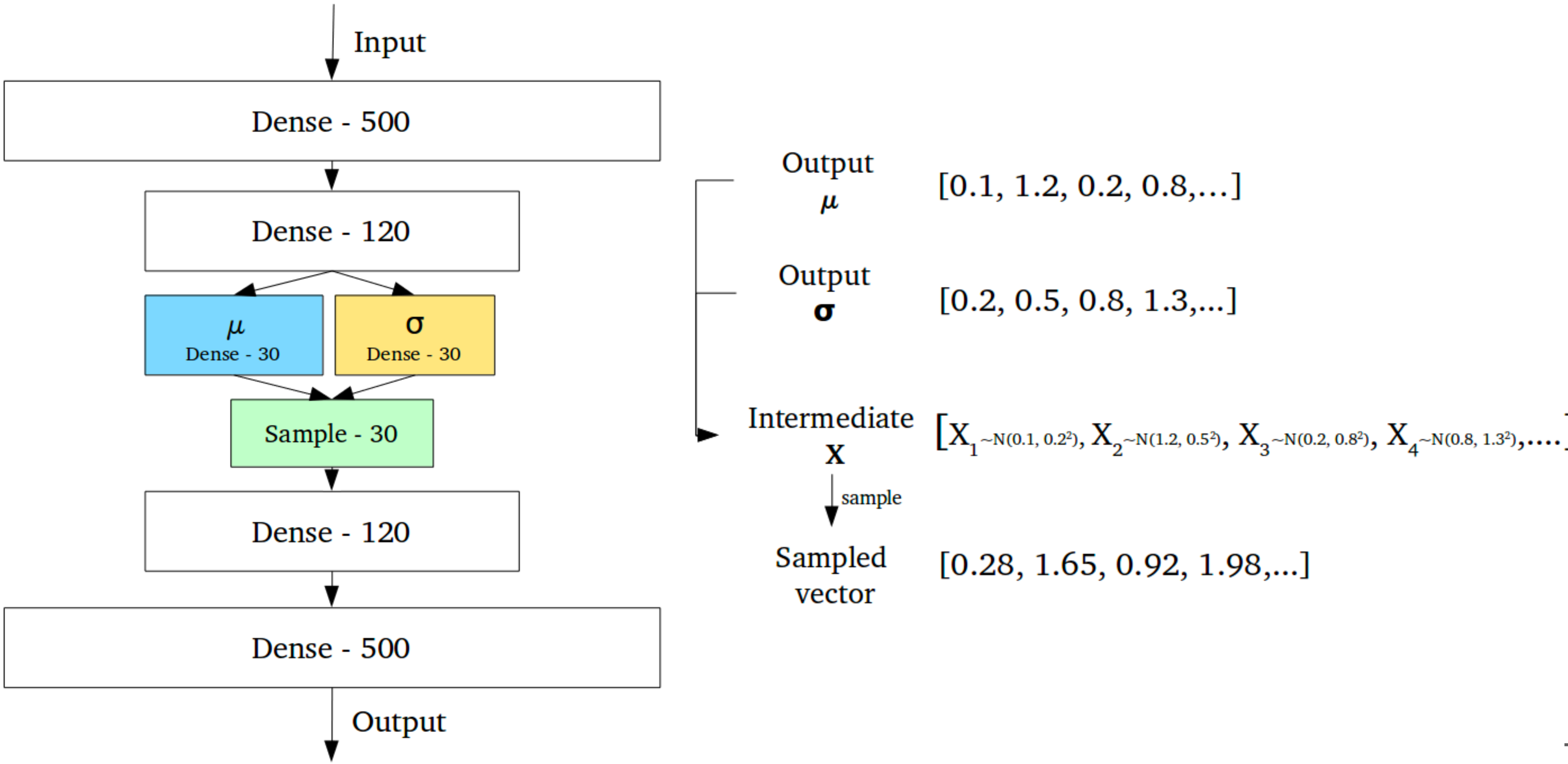
$$x' = g(z)$$

$$L = (x - g(z))^2 + \sigma^2 + \mu^2 - \log(\sigma) - 1$$

(Calculate KL-divergence
between Gaussians)

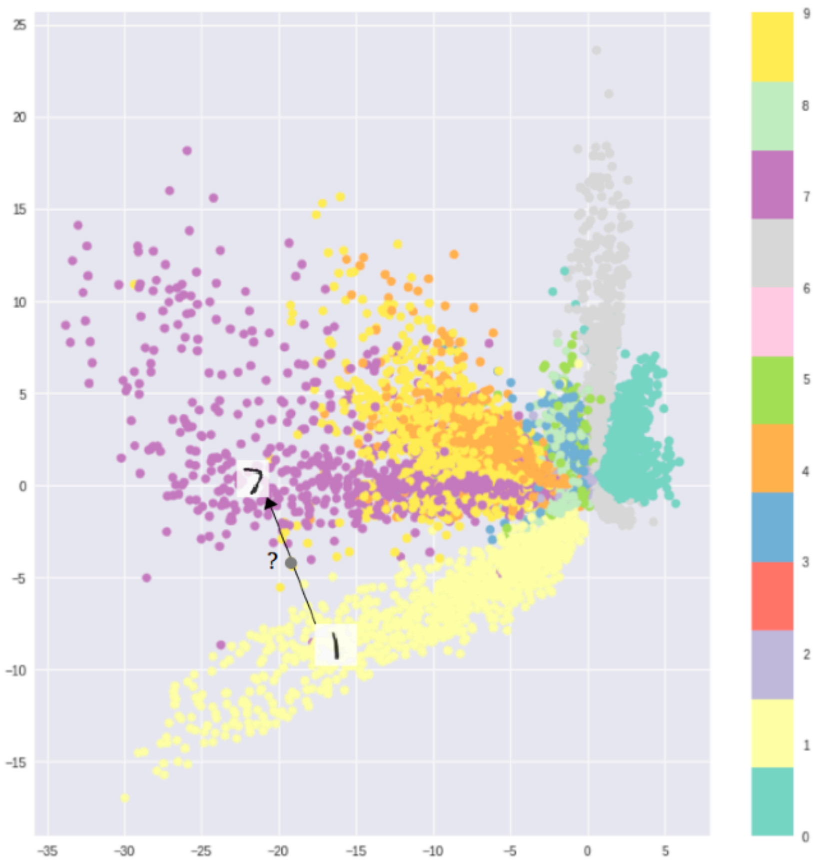
1312.6114

VAE Example


Input
Dense - 500
Dense - 120
μ
Dense - 30
σ
Dense - 30
Sample - 30
Dense - 120
Dense - 500
Output
Output
μ
[0.1, 1.2, 0.2, 0.8,...]
Output
σ
[0.2, 0.5, 0.8, 1.3,...]
Intermediate
X
$[X_{1 \sim N(0.1,\ 0.2^2)}, X_{2 \sim N(1.2,\ 0.5^2)}, X_{3 \sim N(0.2,\ 0.8^2)}, X_{4 \sim N(0.8,\ 1.3^2)},....]$
sample
Sampled
vector
[0.28, 1.65, 0.92, 1.98,...]

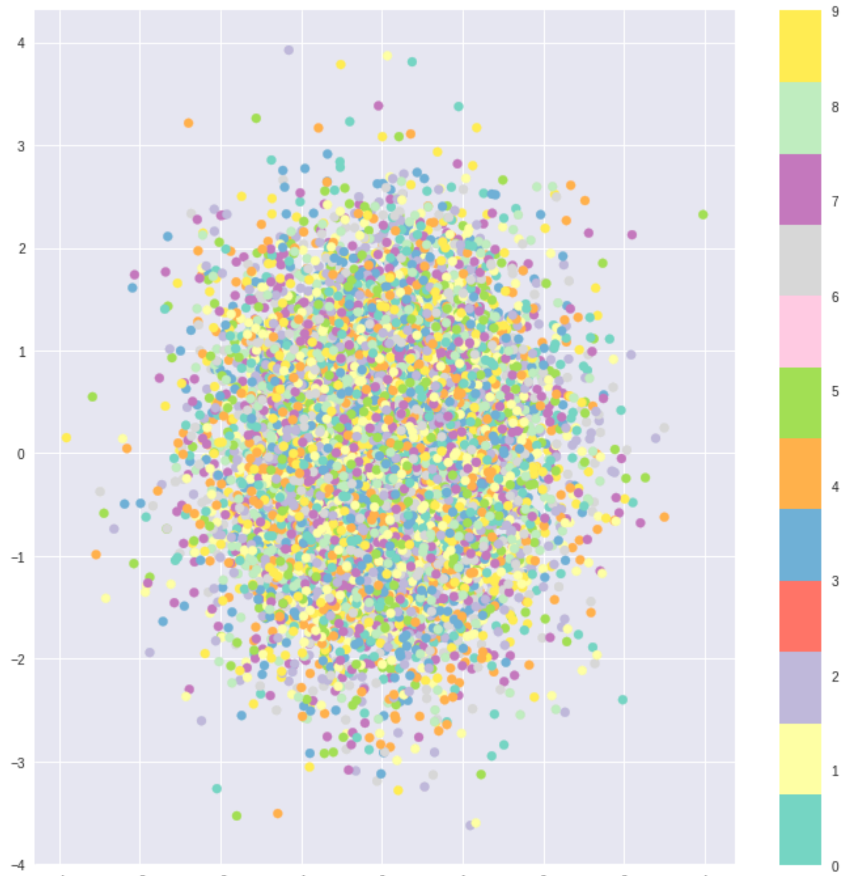
Loss terms

Latent space of MNIST VAE



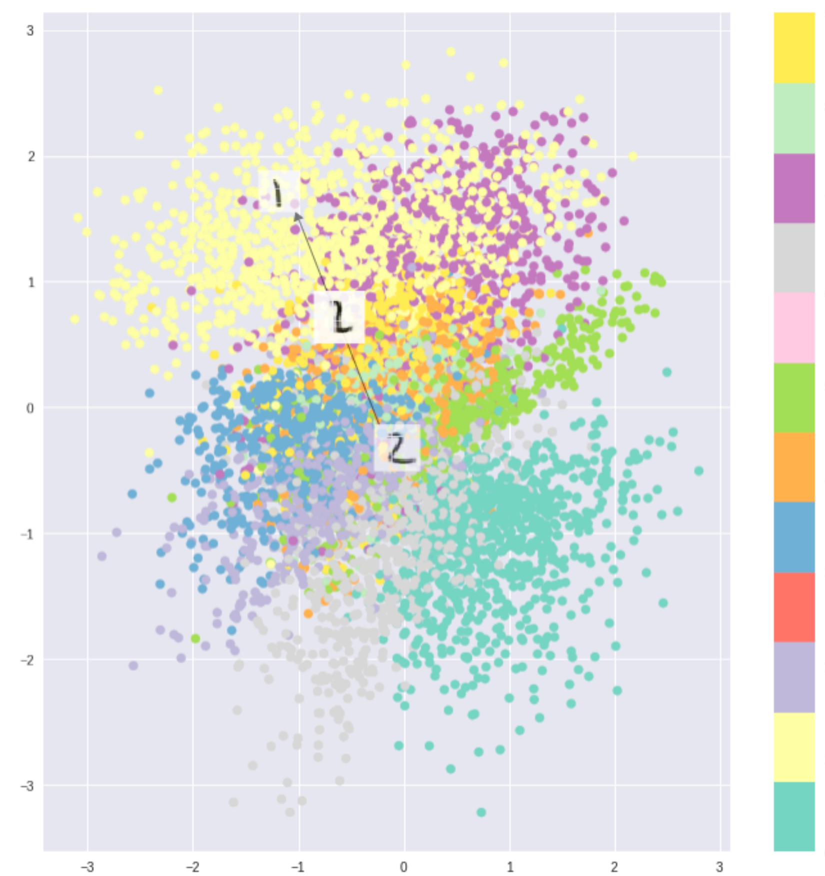
$$(x - g(z))^2$$

Reconstruction



$$\sigma^2 + \mu^2 - \log(\sigma) - 1$$

Regularisation



Both terms

towardsdatascience.com

Comments on VAEs

Architecture:

- Low complexity, fast and adaptable
- Target: Maximise lower bound on likelihood

Learning:

- Stable training
- Average prediction → blurrier output
- Interpretable latent space

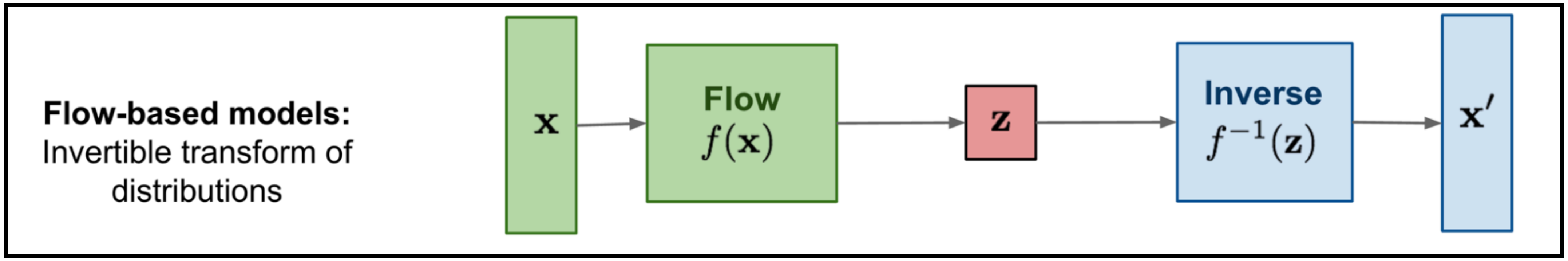
Maturity:

- Well established,
  many variants and extensions

1607.07539

Normalising Flows

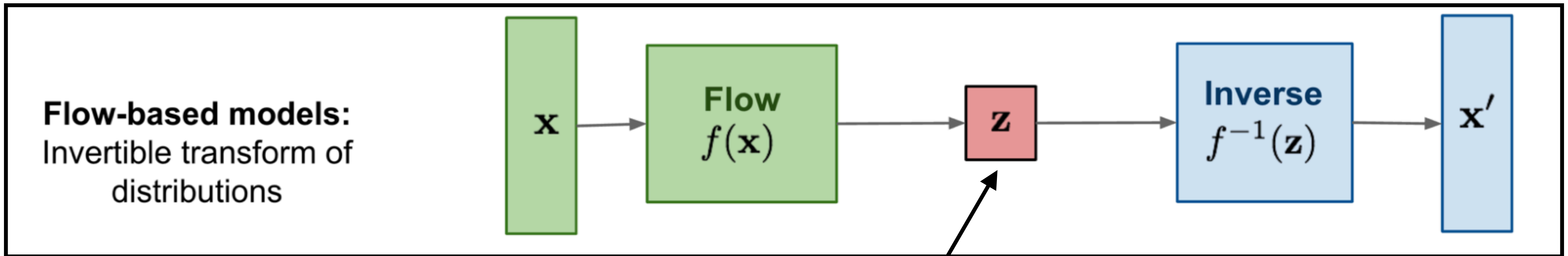
Generative models



In auto-encoders, the decoder learns to 'undo' the encoder

Can we make this exact?

Generative models

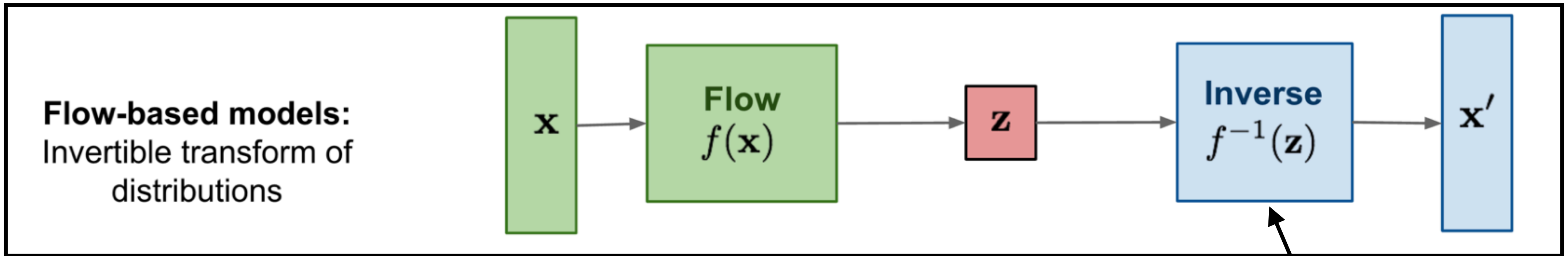


Choose latent space, e.g. standard normal distribution (normalising flow!)
Same dimension as data!

Learn a diffeomorphism between data and latent-space

1505.05770, 1908.09257

Generative models

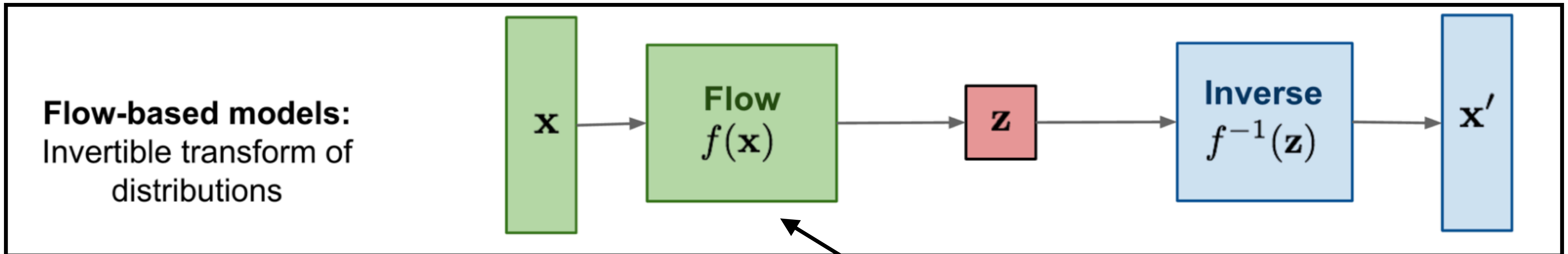


f^{-1} is not a learned inversion, but exact inverse by construction

Learn a diffeomorphism between data and latent-space

Bijective, invertable

Generative models



Take into account Jacobian determinant to evaluate probability density

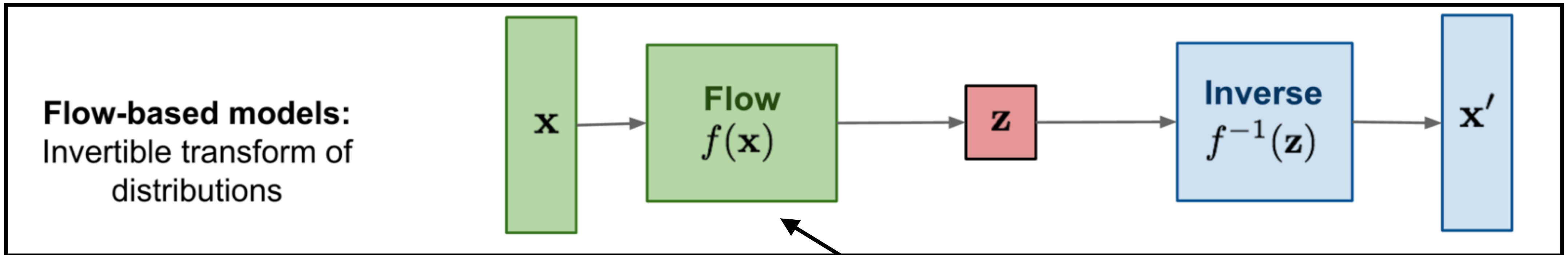
Learn a diffeomorphism between data and latent-space

Bijective, invertable

Learn likelihood of data

1505.05770, 1908.09257

Generative models



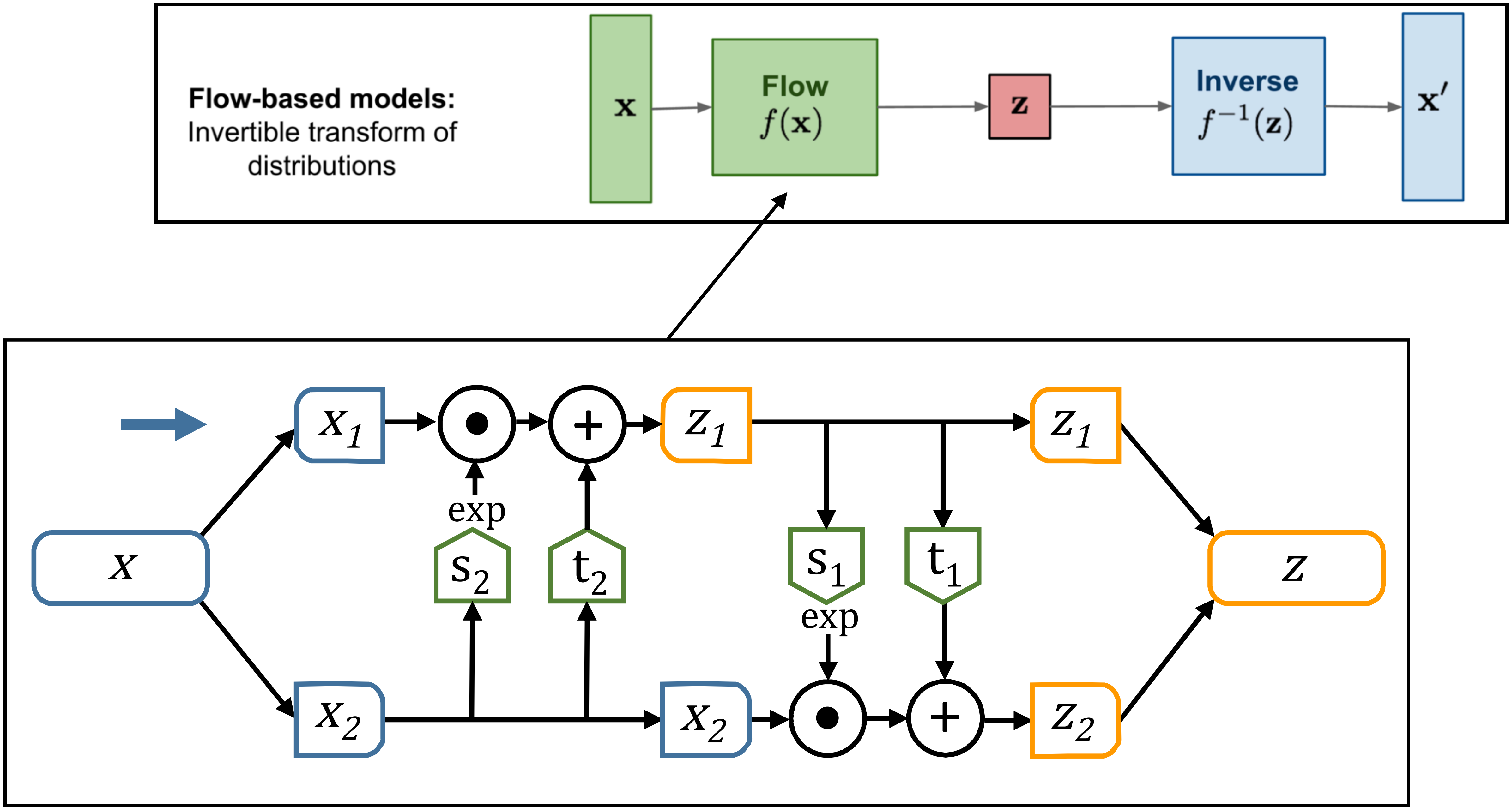
Take into account Jacobian determinant to evaluate probability density

2 challenges:

- Invertible
- Easy-to-calculate Jacobean

1505.05770, 1908.09257

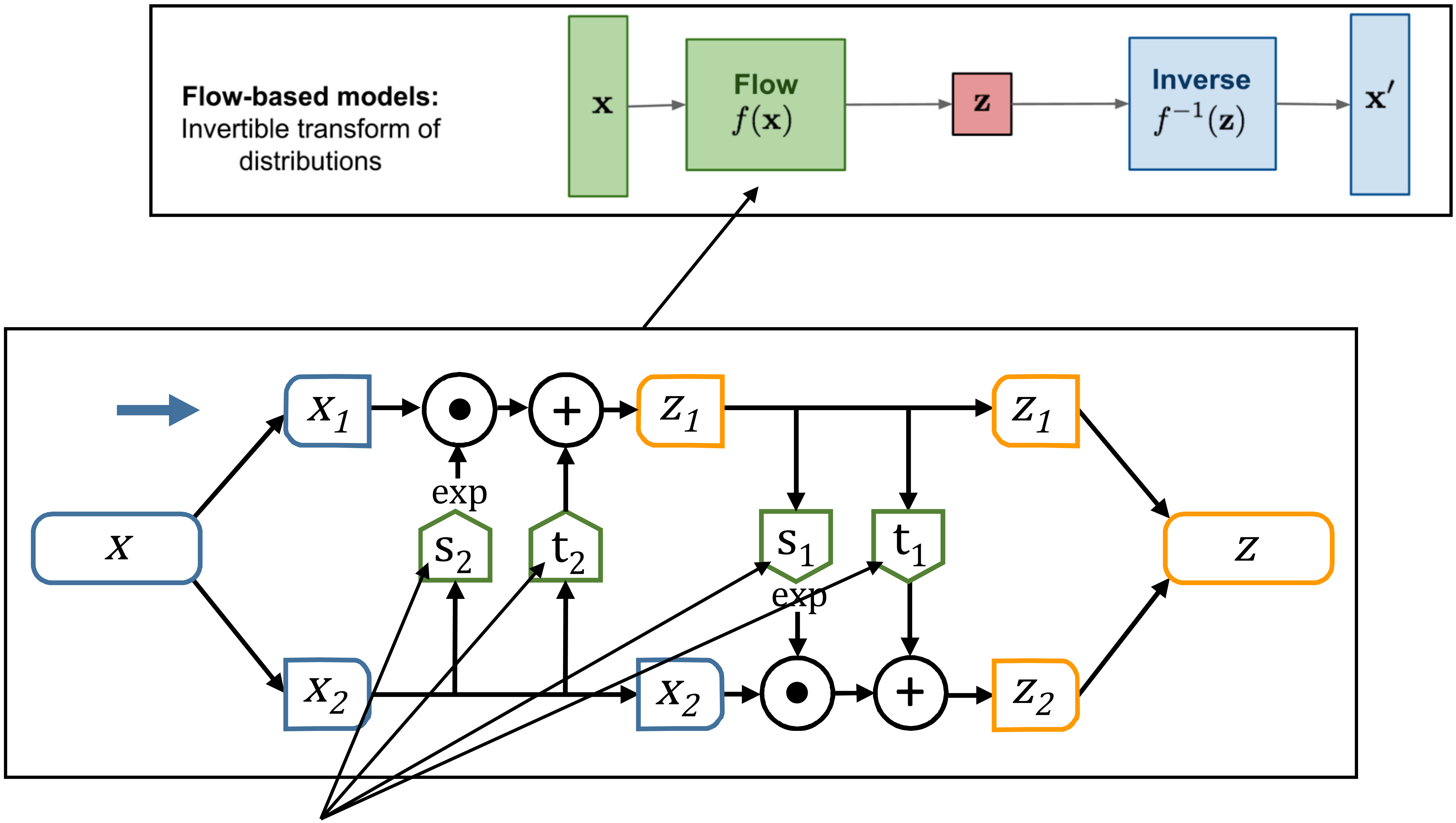
Coupling flows



Coupling layers: Not the most expressive, but useful for illustration/understanding

1505.05770, 1908.09257

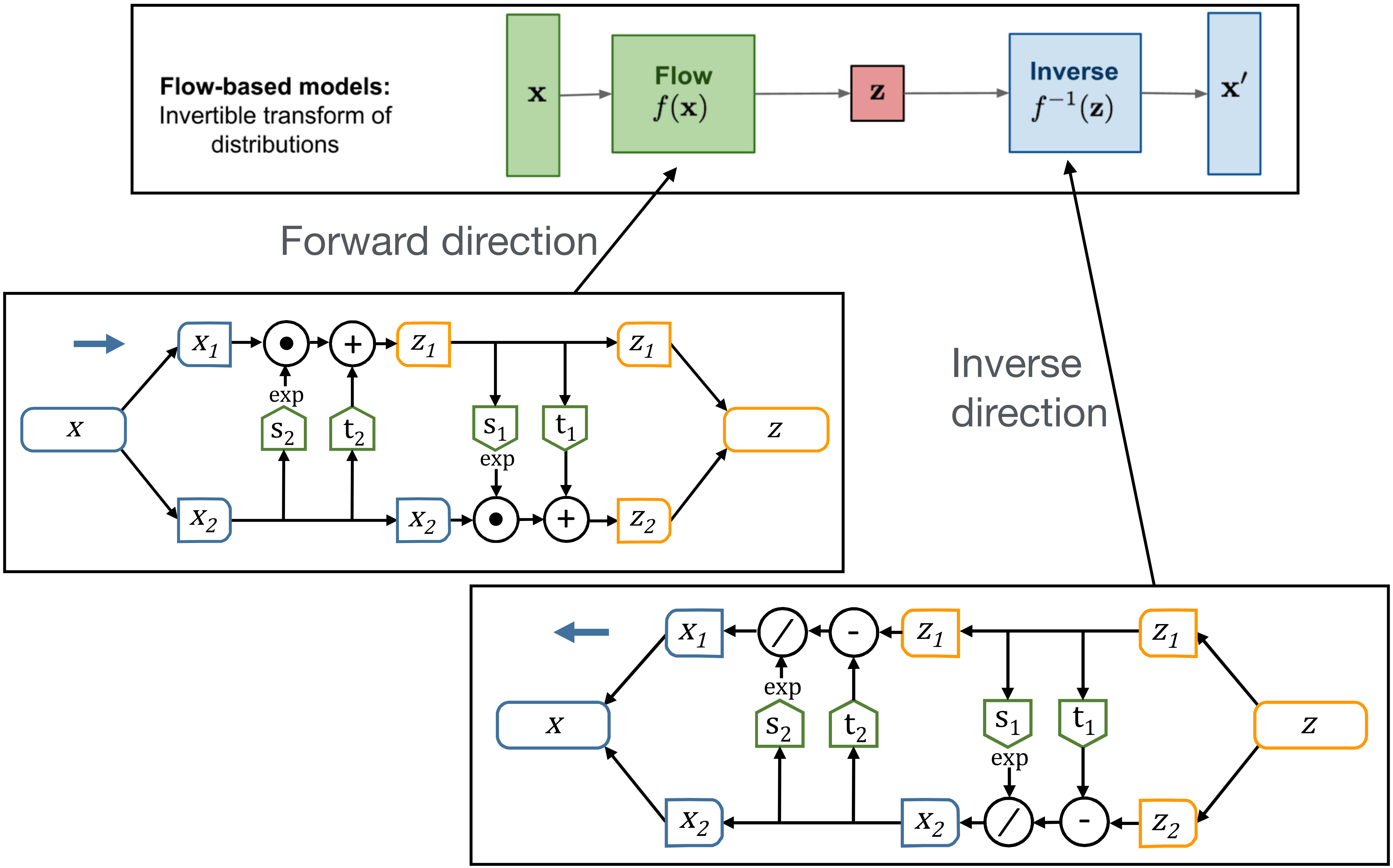
Coupling flows



Simple (e.g. dense) neural networks

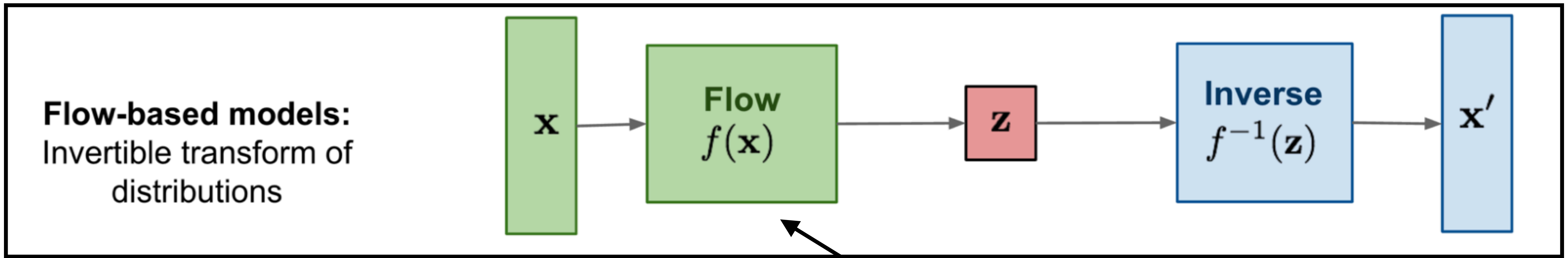
1505.05770, 1908.09257

Coupling flows



1505.05770, 1908.09257

Generative models

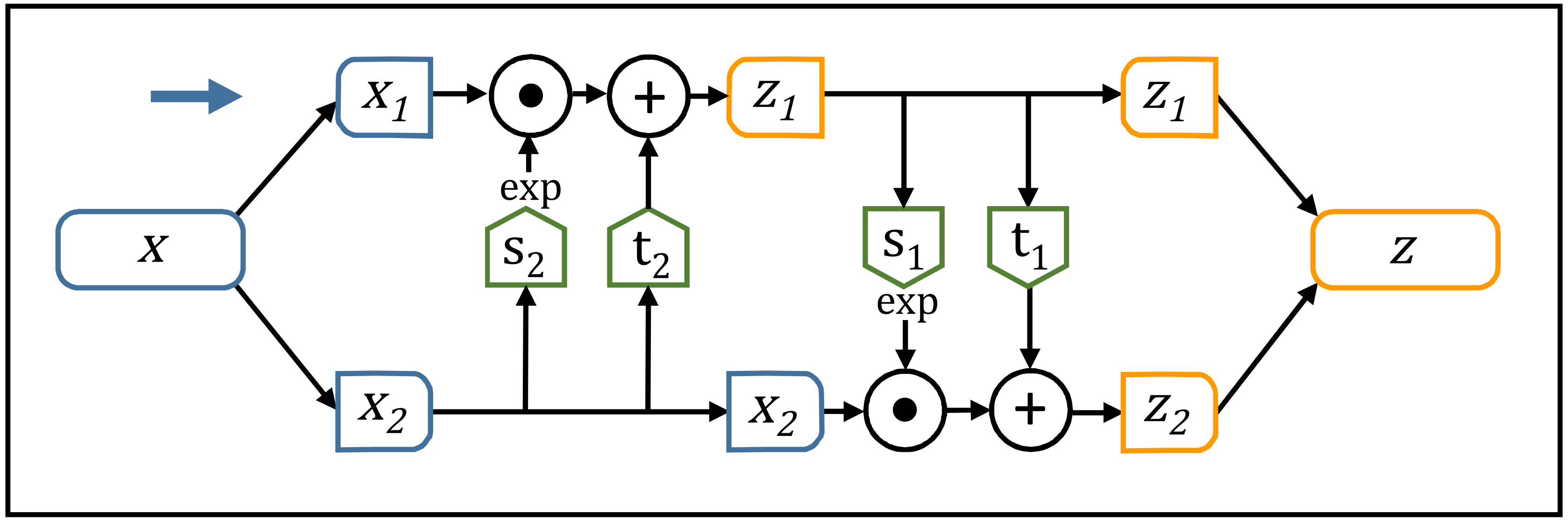


Take into account Jacobian determinant to evaluate probability density

2 challenges:
Invertible
Easy-to-calculate Jacobian

1505.05770, 1908.09257

Calculating Jacobian determinant



$$\begin{pmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{pmatrix} \xrightarrow{f_1} \begin{pmatrix}\mathbf{z}_1\\ \mathbf{x}_2\end{pmatrix} \xrightarrow{f_2} \begin{pmatrix}\mathbf{z}_1\\ \mathbf{z}_2\end{pmatrix}$$ with $$\mathbf{x}_1 \xrightarrow{f_1} \mathbf{z}_1 = \mathbf{x}_1 \odot \exp(s_2(\mathbf{x}_2)) + t_2(\mathbf{x}_2)$$ $$\mathbf{x}_2 \xrightarrow{f_1} \mathbf{x}_2.$$

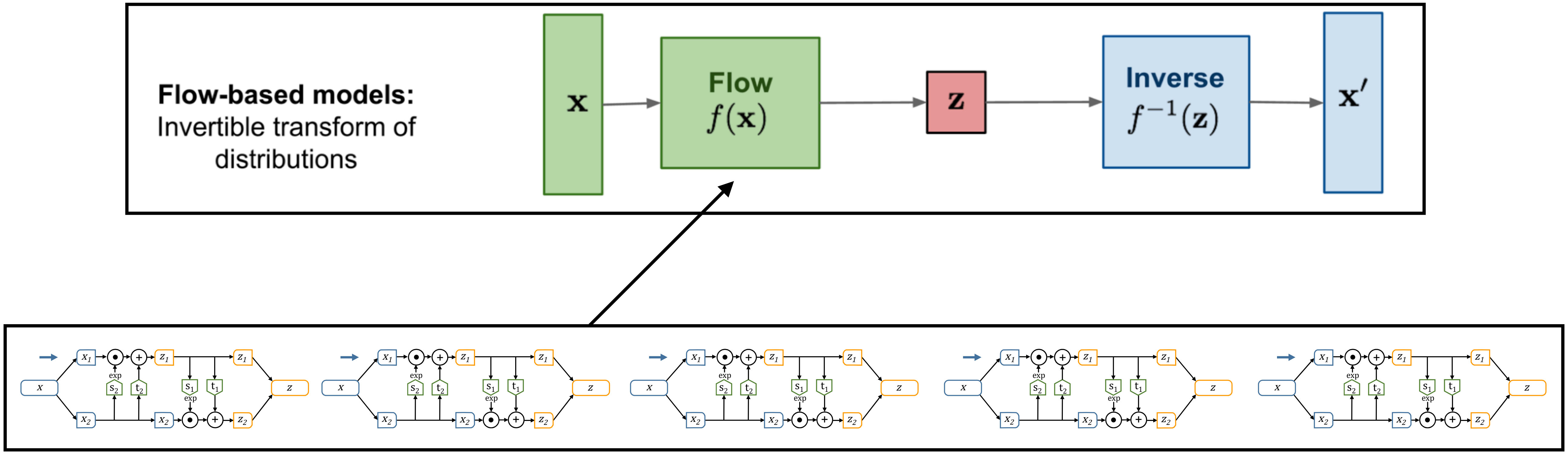
$$\mathbf{J_1} = \begin{pmatrix} \frac{\partial \mathbf{z}_1}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{z}_1}{\partial \mathbf{x}_2} \\ \frac{\partial \mathbf{x}_2}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{x}_2}{\partial \mathbf{x}_2} \end{pmatrix} = \begin{pmatrix} \mathrm{diag}(\exp(s_2(\mathbf{x}_2))) & \frac{\partial \mathbf{z}_1}{\partial \mathbf{x}_2} \\ 0 & \mathbb{1} \end{pmatrix}$$

Triangular by construction

$$\det \mathbf{J_1} = \prod \exp(s_2(\mathbf{x}_2)) = \exp\left(\sum s_2(\mathbf{x}_2)\right)$$

1505.05770, 1908.09257

Composition



Composition of bijective functions remains bijective

Chain rule: Jacobian determinant of composition is product of determinants

1505.05770, 1908.09257

How to train NF?

Training objective: Minimise negative log likelihood of data

Sample points from training data

$$\mathcal{L} = -\mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}}\left[-\frac{1}{2}||f(\mathbf{x}))||_2^2 + \sum s(\mathbf{x})\right]$$

1505.05770, 1908.09257

How to train NF?

Training objective: Minimise negative
log likelihood of data

$$\mathcal{L} = -\mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}}\left[-\frac{1}{2}||f(\mathbf{x}))||_2^2 + \sum s(\mathbf{x})\right]$$

Transform into latent space and
evaluate probability there

$$\varphi(\mathbf{z}) = \frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{1}{2}\mathbf{z}^2}$$

$$-\log(\varphi(f(\mathbf{x})) = -\log(\frac{1}{\sqrt{2\pi}}) - \log\left(\mathrm{e}^{-\frac{1}{2}f(\mathbf{x})^2}\right) = -\log(\frac{1}{\sqrt{2\pi}}) + \frac{1}{2}f(\mathbf{x})^2$$

How to train NF?

Training objective: Minimise negative log likelihood of data

$$\mathcal{L} = -\mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}}\left[-\frac{1}{2}||f(\mathbf{x}))||_2^2 + \sum s(\mathbf{x})\right]$$

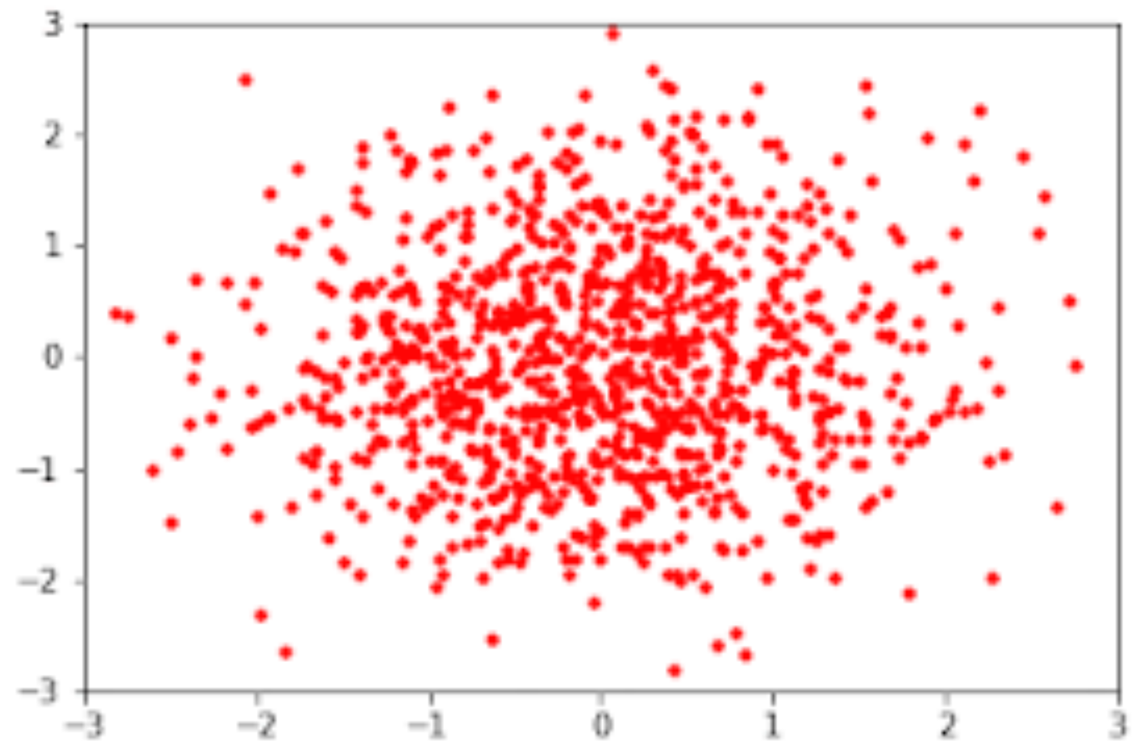
Contribution from Jacobian determinant

$$\det \mathbf{J} = \exp\left(\sum s(\mathbf{x})\right)$$

$$-\log(\det \mathbf{J}) = -\sum s(\mathbf{x})$$

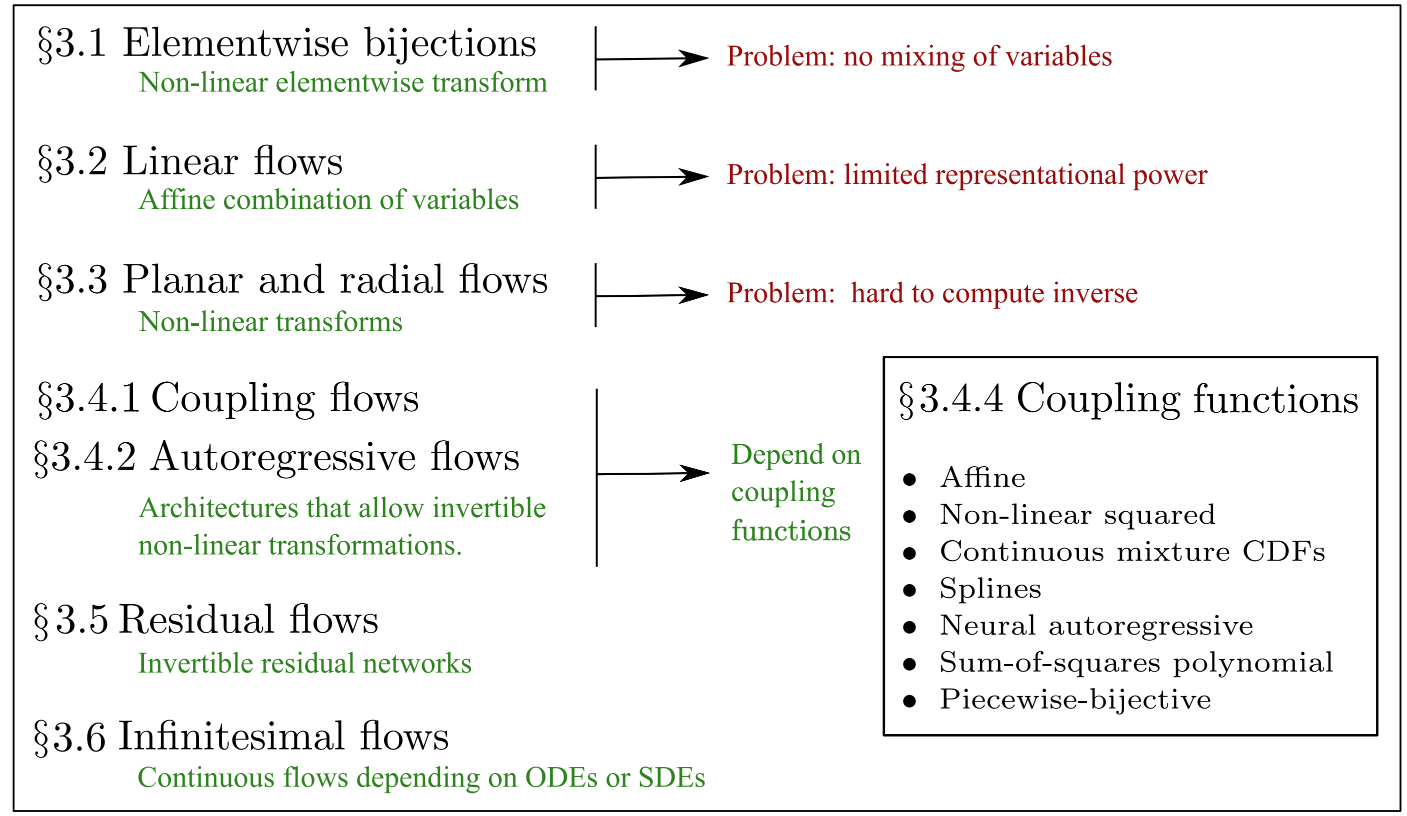
1505.05770, 1908.09257

Animation



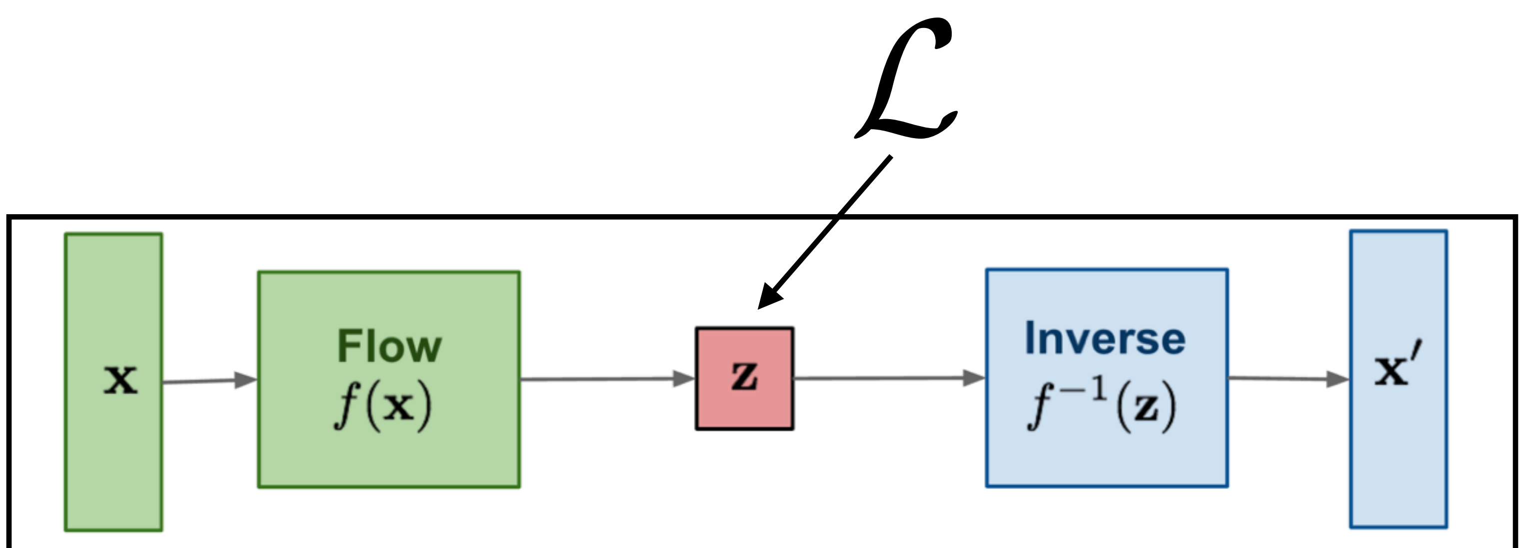
Comments on Flows

Only scratched the surface:
more constructions available



Exact learning of likelihood
→ Better generative fidelity
→ Can evaluate likelihood of data

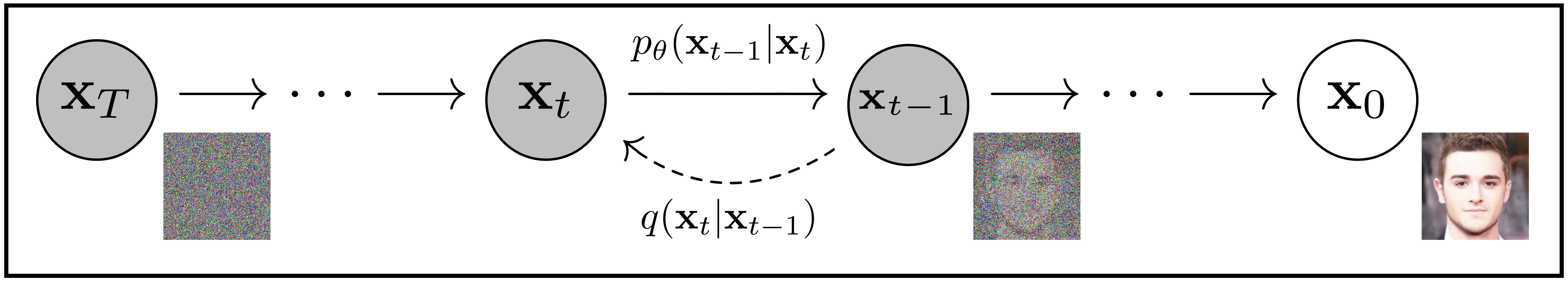
More complex
→ Slower, choice of fast direction



1908.09257

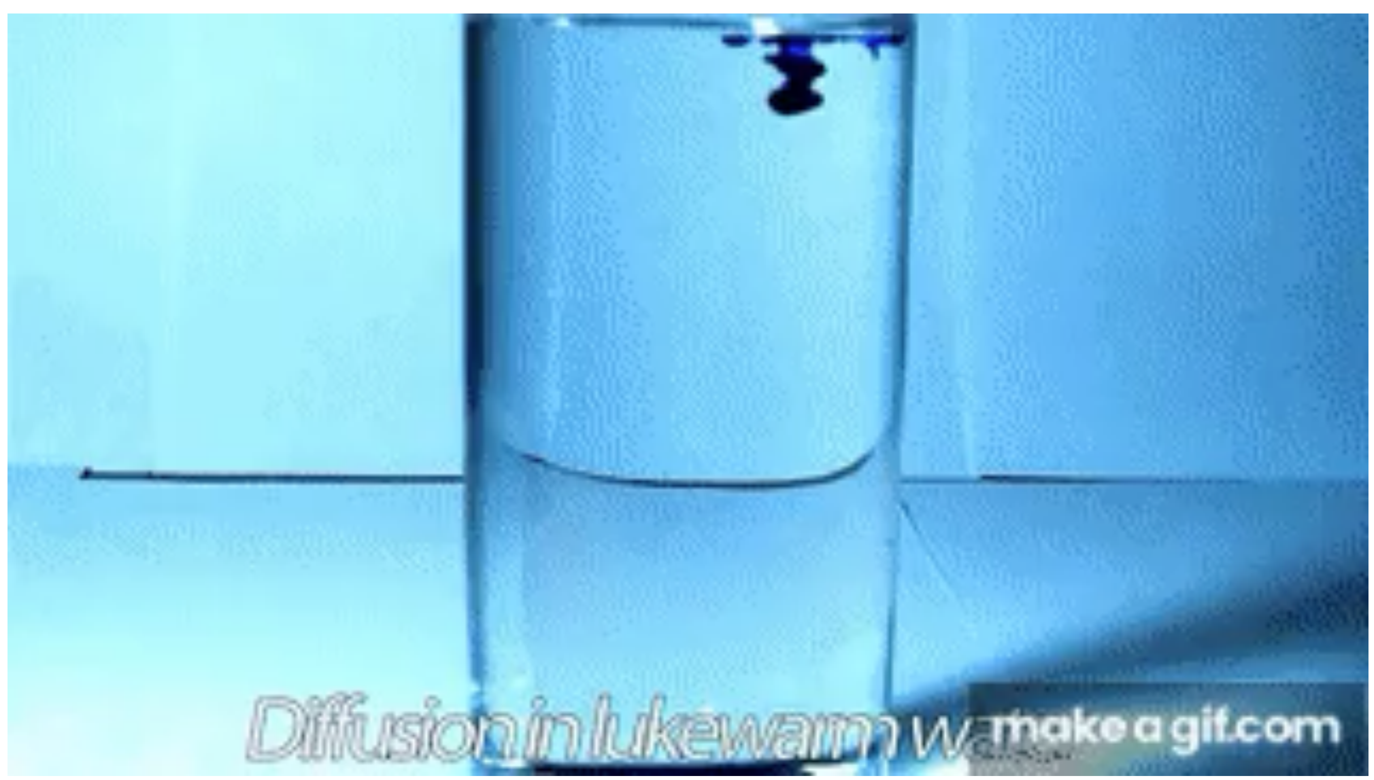
Diffusion Models

Diffusion Model

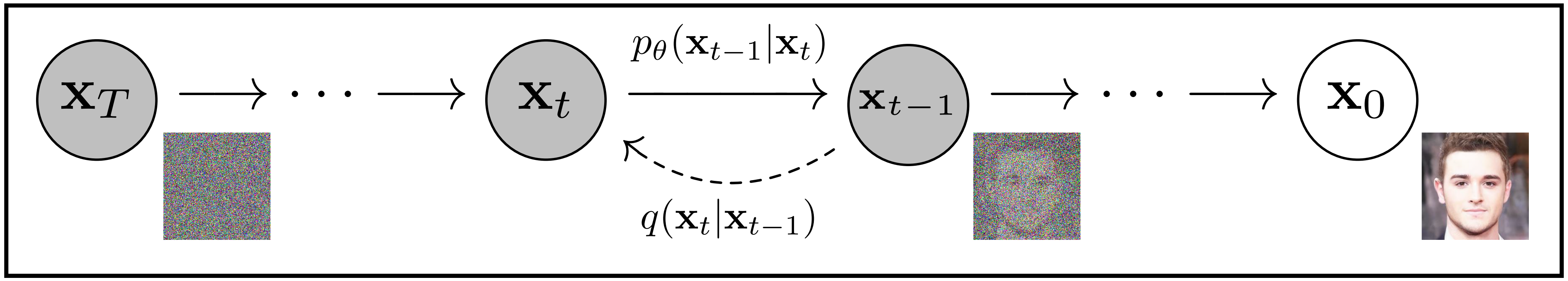


Core idea: Stepwise transition
from pure noise to data

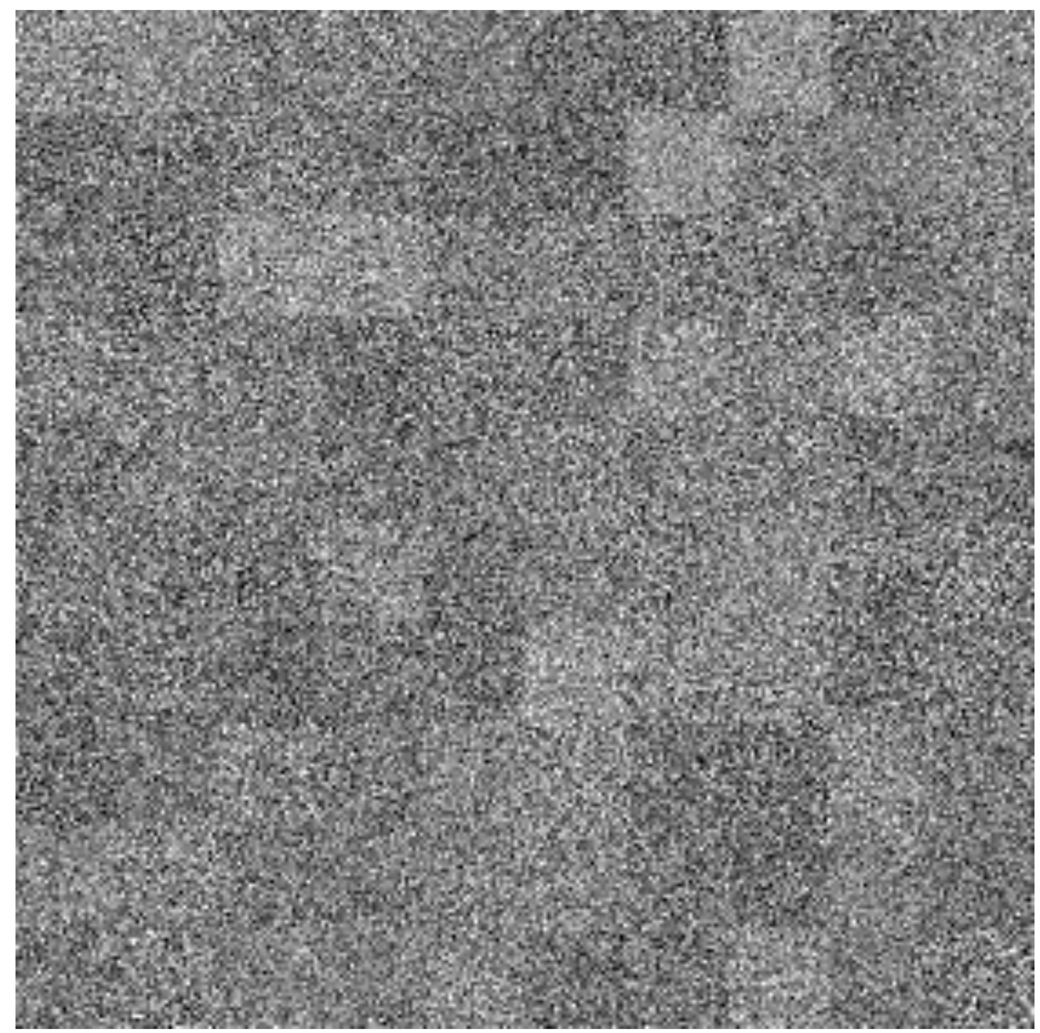
Markov chain



Diffusion Model

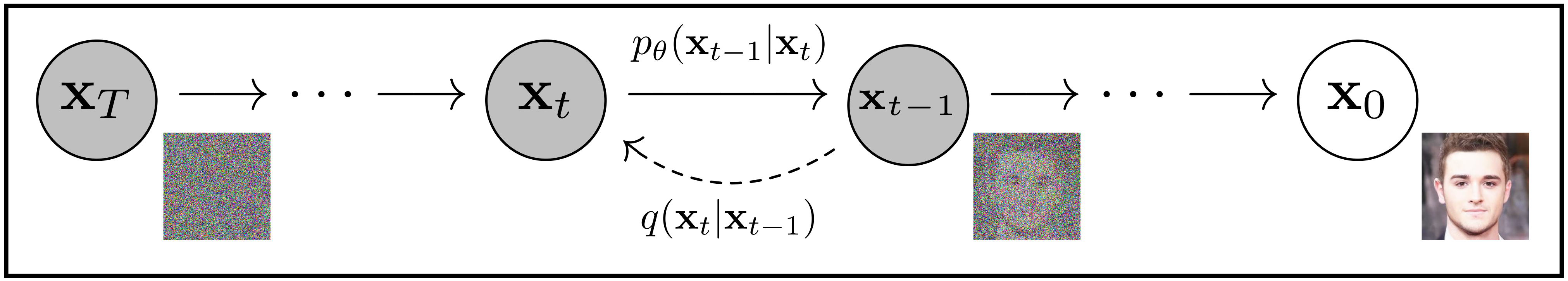


Core idea: Stepwise transition
from pure noise to data

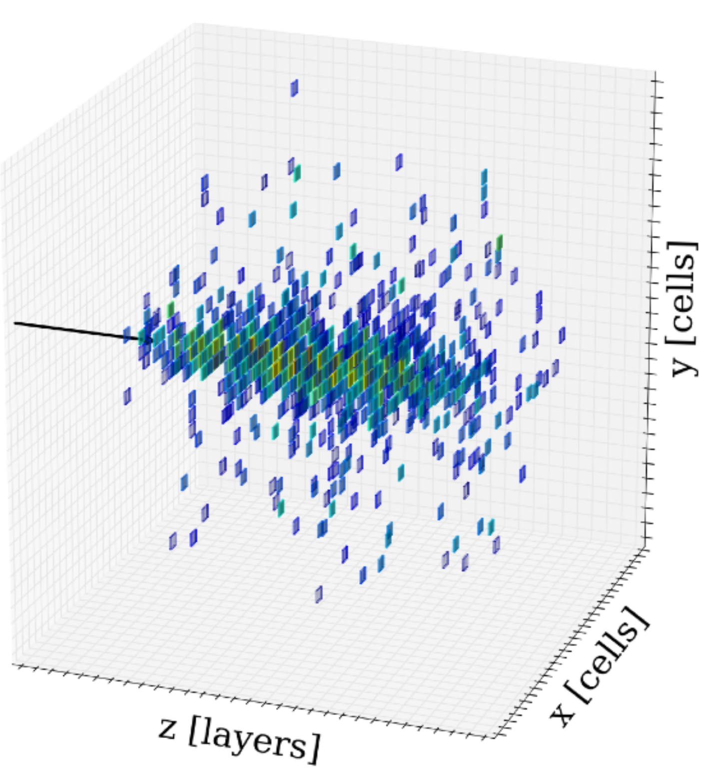


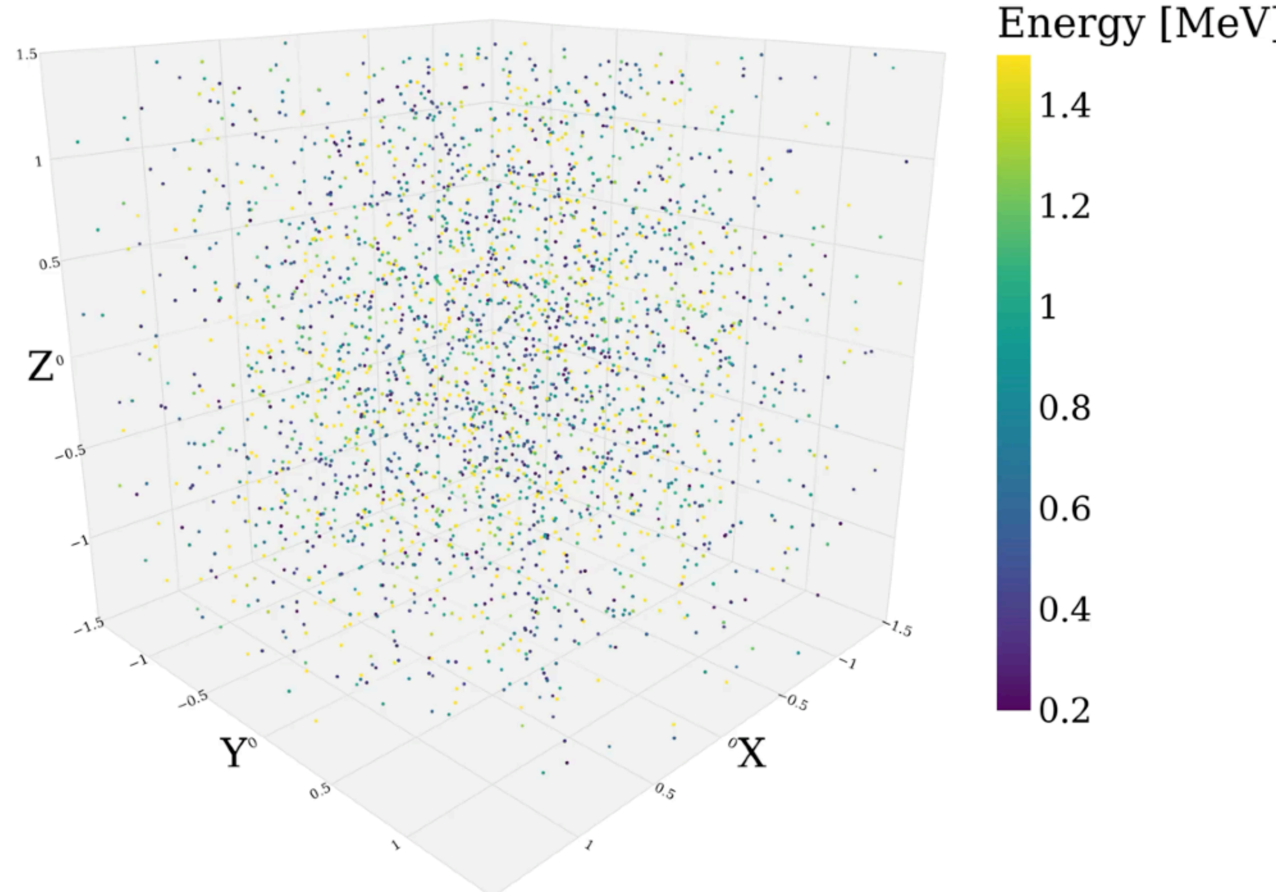
https://medium.com/mlearning-ai/enerating-images-with-
ddpms-a-pytorch-implementation-cef5a2ba8cb1

Diffusion Model



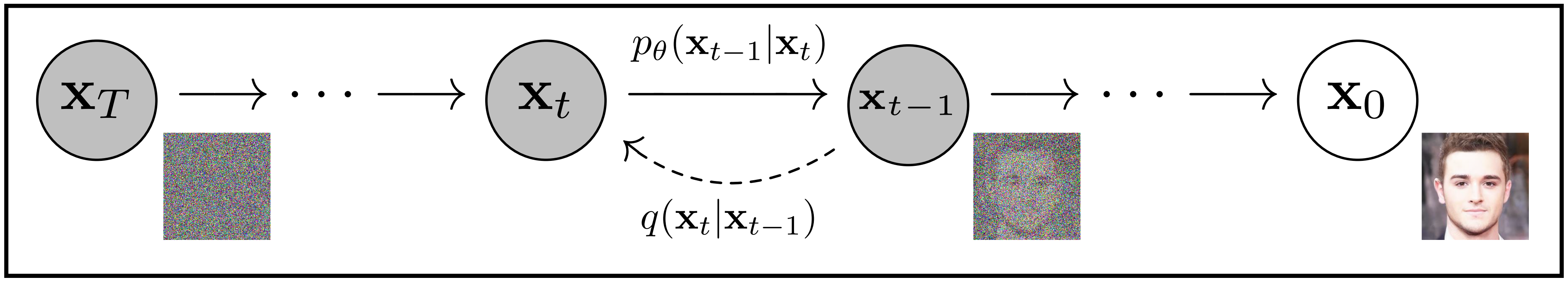
Core idea: Stepwise transition from pure noise to data



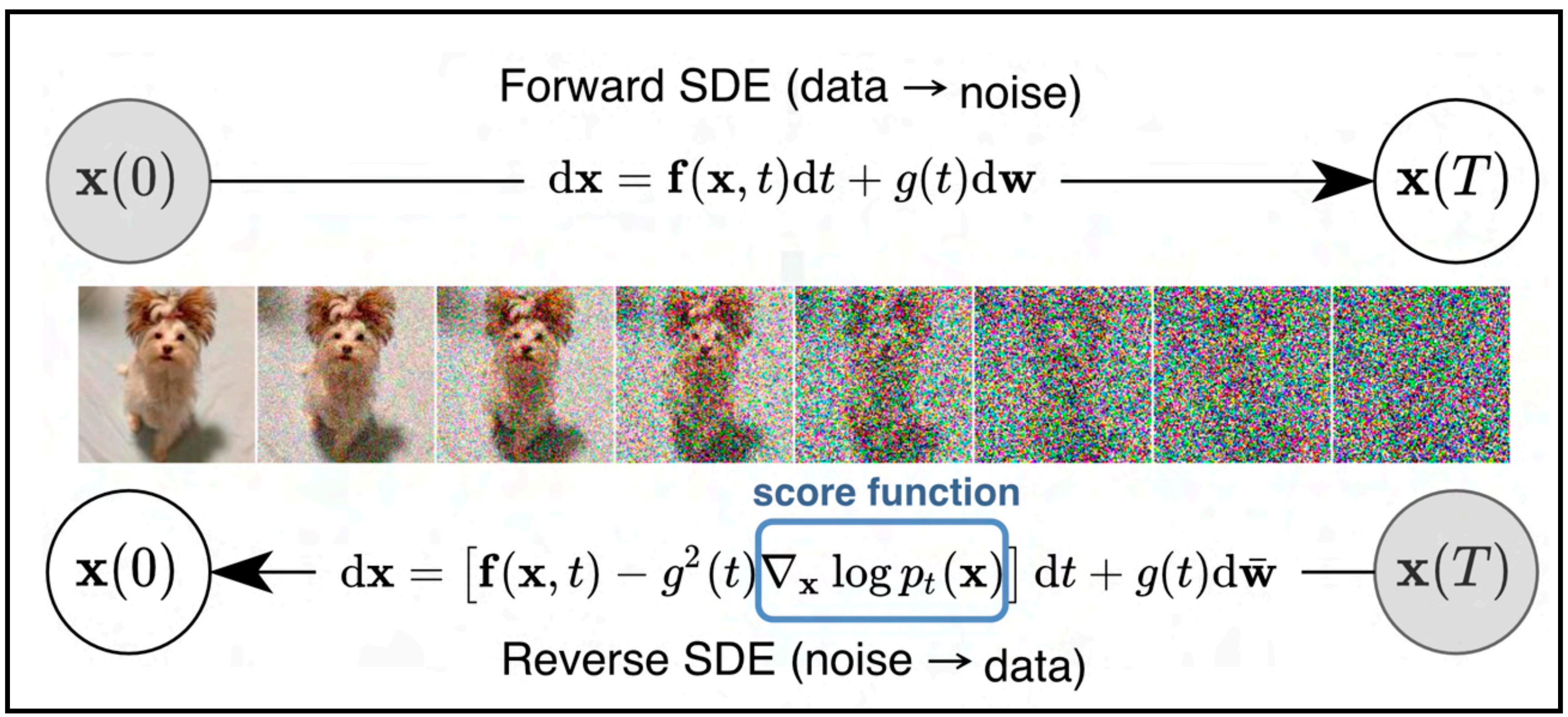


2305.04847

Continuos Diffusion Model



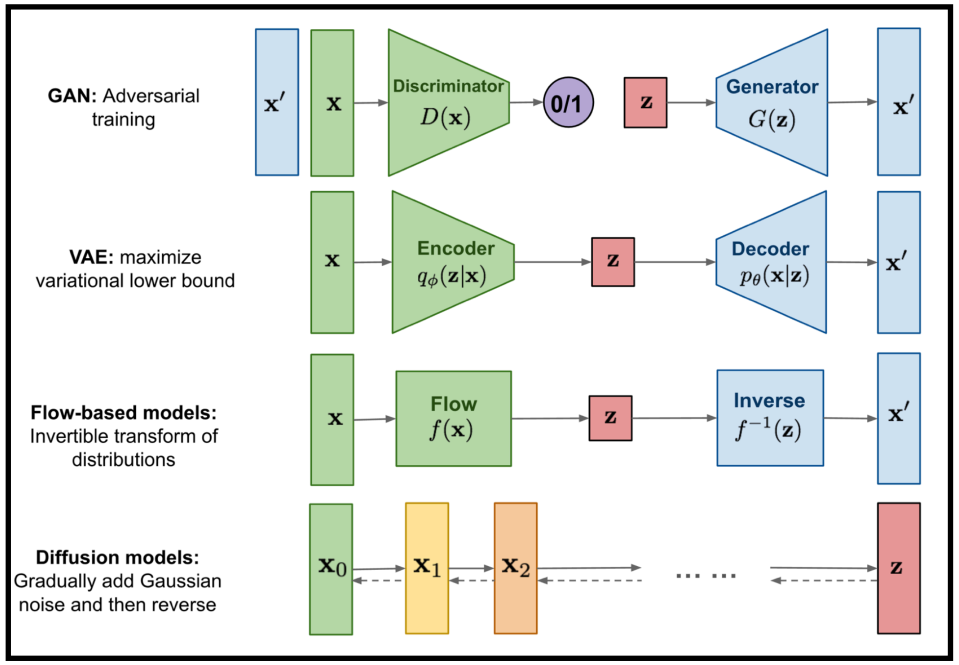
Replace discrete noise steps



with stochastic differential equations (SDEs)

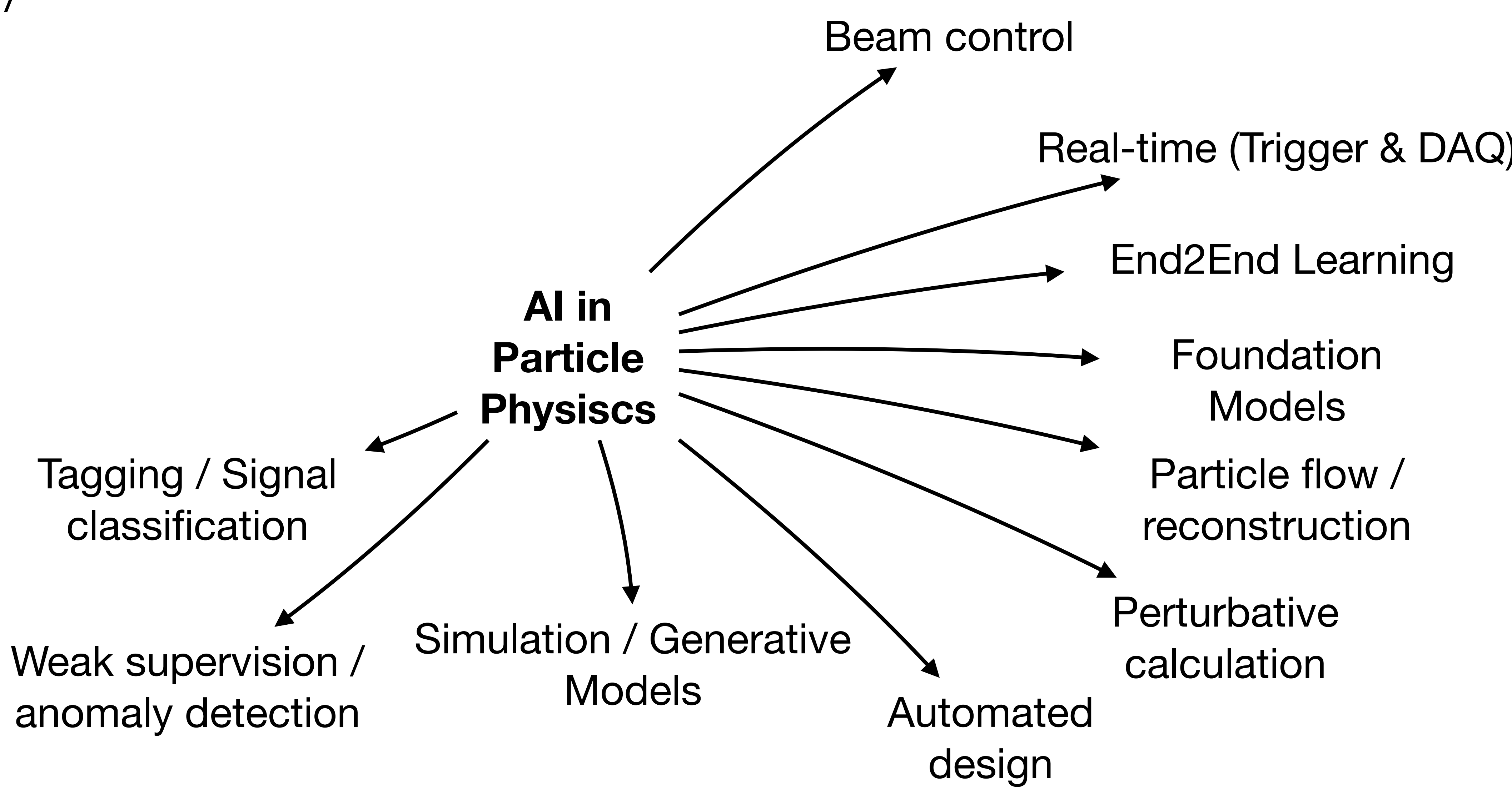
Comments on Generative Models

- Simulations are a core need of fundamental physics but computationally very costly
- Aided by developments in generative models
- Main application: Detector simulation, but also other developments (theory simulation, in-situ backgrounds, …)



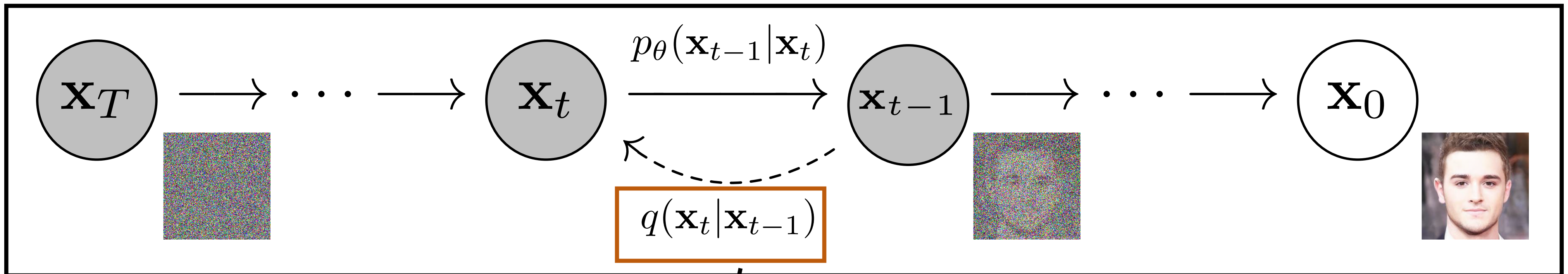
Closing

- Broad developments on ML in Particle Phsycis
- Covered some key topics, but wide range of other applications
- -> https://iml-wg.github.io/HEPML-LivingReview/



Bonus Material

Diffusion Model



Core idea: Stepwise transition from pure noise to data

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

Individual step

Noise schedule (hyper-parameter)

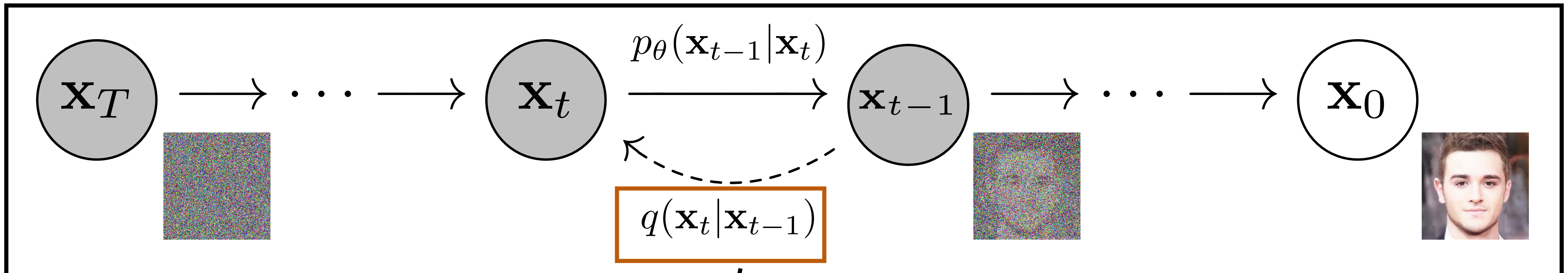
Forward (Data → Noise)

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

Full chain

This is added in the training process

Diffusion Model



Core idea: Stepwise transition from pure noise to data

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

Forward
(Data → Noise)

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

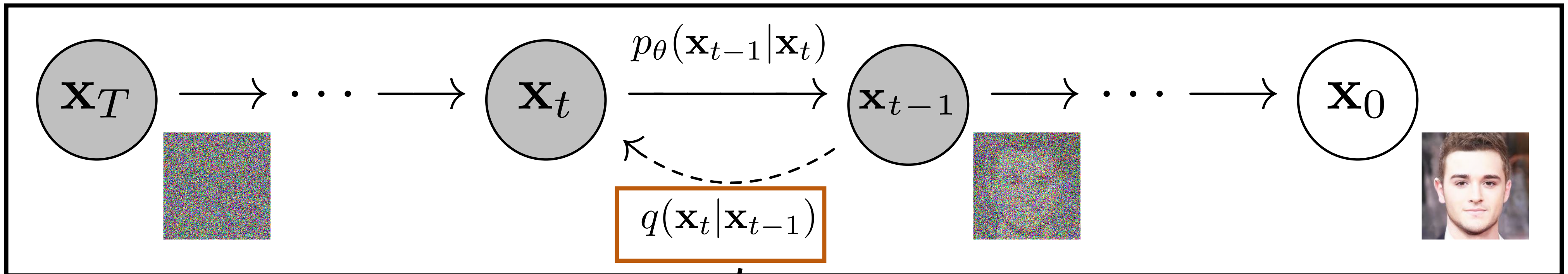
Shortcut:
transition to any time

$$\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$$

$$\alpha_t := 1 - \beta_t$$

This is added in the training process

Diffusion Model



Core idea: Stepwise transition from pure noise to data

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

Forward
(Data → Noise)

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

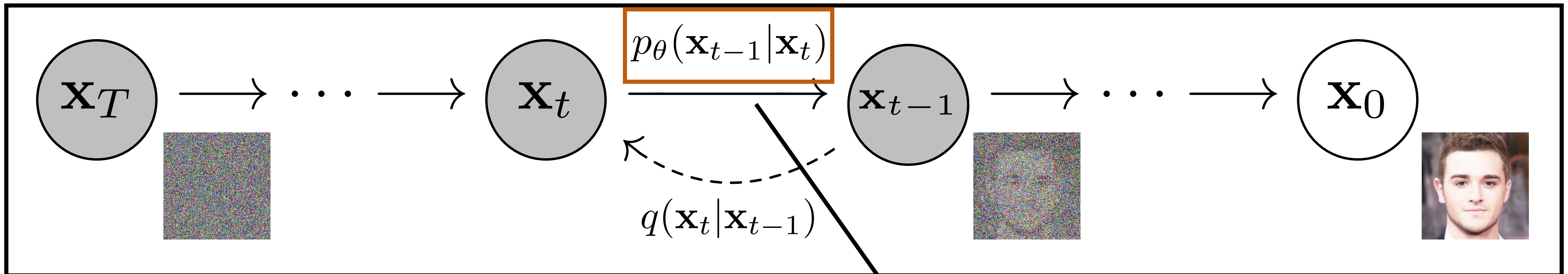
$$\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boxed{\boldsymbol{\epsilon}} \text{ for } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Rewrite: State at any time

Will try to predict

This is added in the training process

Diffusion Model



Core idea: Stepwise transition from pure noise to data

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

To be learned (in principle)

$$\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) = \sigma_t^2 \mathbf{I}$$

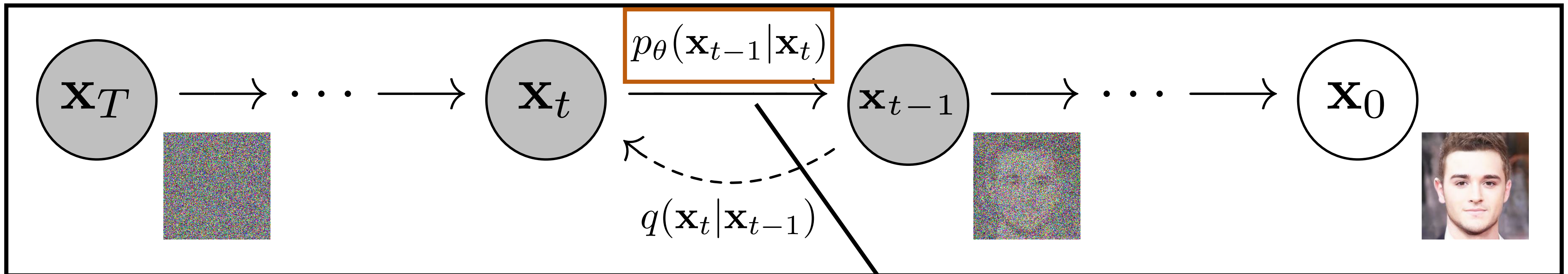
(hyper-parameter)

Reverse (Noise → data)

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

Diffusion Model



Core idea: Stepwise transition from pure noise to data

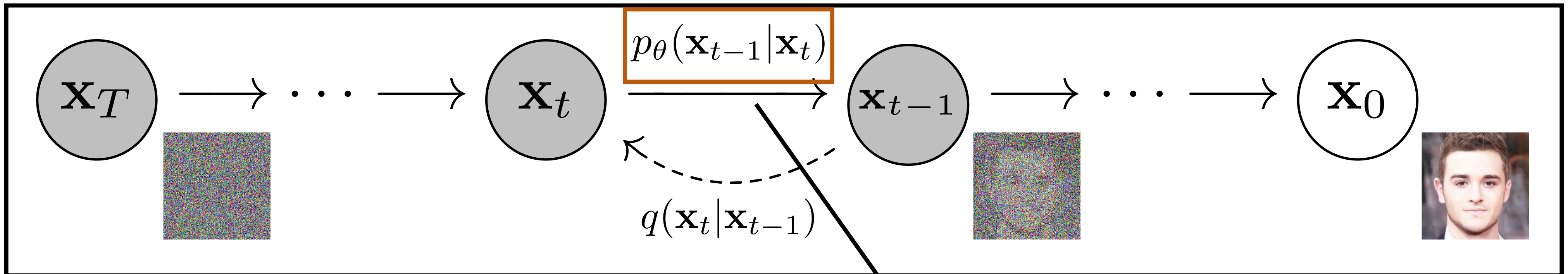
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

$$\frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)$$

Reverse
(Noise → data)

Optimal expected mean (take into account known noise schedule)

Diffusion Model



Core idea: Stepwise transition from pure noise to data

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

Reverse
(Noise → data)

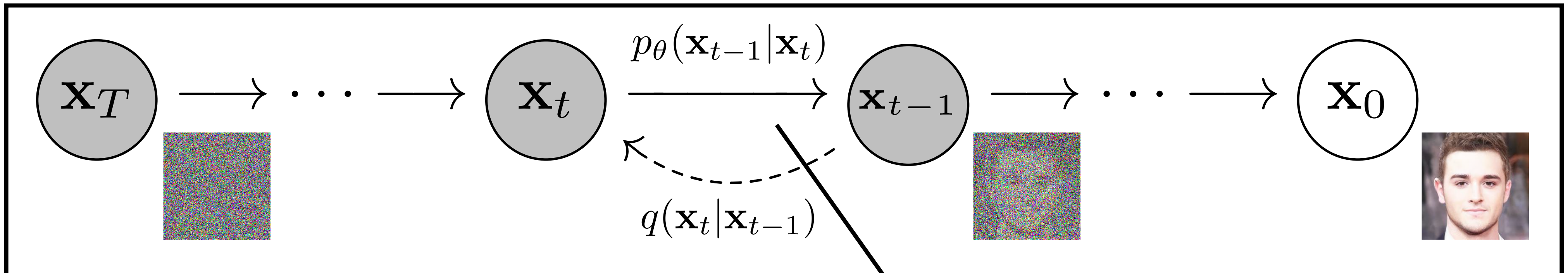
$$\frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon)$$

Use to parametrise function:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boxed{\epsilon_\theta(x_t, t)}))$$

and learn to predict noise

Diffusion Model



Core idea: Stepwise transition from pure noise to data

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

Reverse (Noise → data)

Resulting learning objective

$$L_{\mathrm{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boxed{\boldsymbol{\epsilon}_\theta}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right]$$

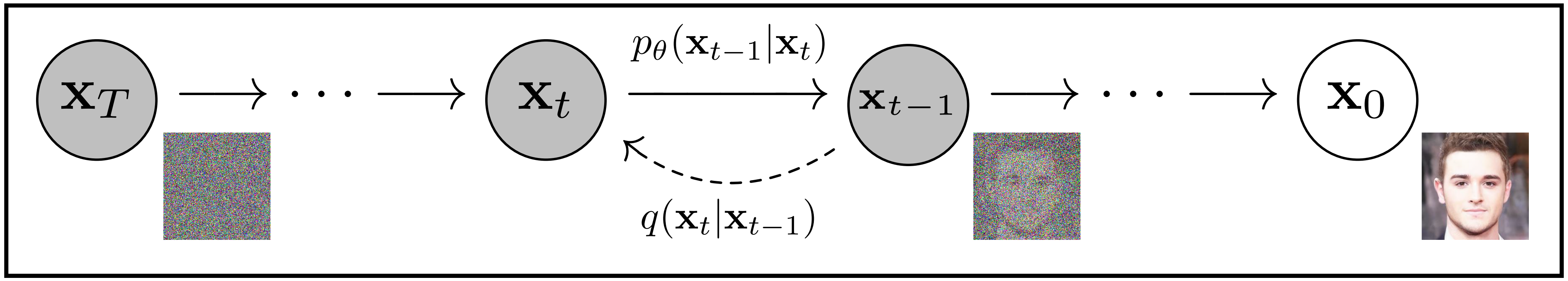
Noisy image

Reminder: Forward diffusion to time t

$$\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$$

Timestep

Diffusion Model



Core idea: Stepwise transition
from pure noise to data

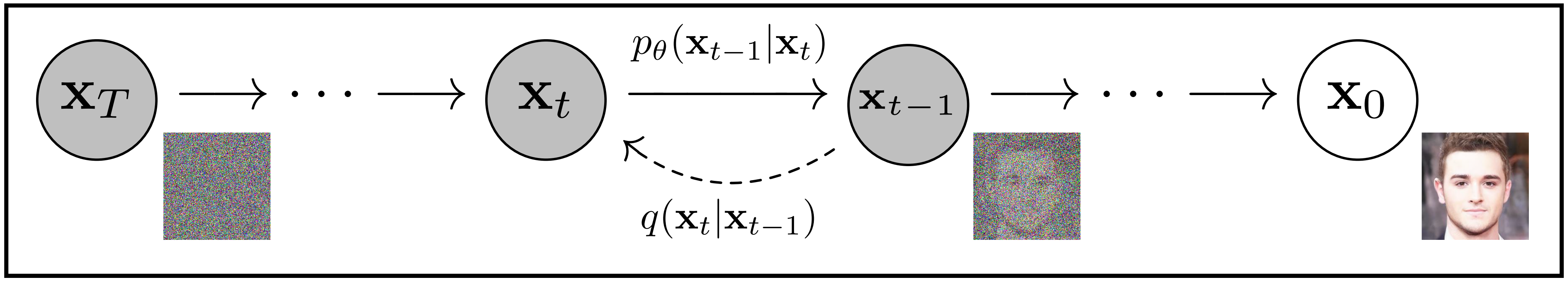
Algorithm 1 Training

1: **repeat**
2: $\quad \mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $\quad t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
4: $\quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: $\quad$ Take gradient descent step on

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

6: **until** converged

Diffusion Model



Core idea: Stepwise transition from pure noise to data

Algorithm 1 Training

```
repeat
  x_0 ~ q(x_0)
  t ~ Uniform({1, ..., T})
  ε ~ N(0, I)
  Take gradient descent step on
      ∇_θ ‖ε − ε_θ(√ᾱ_t x_0 + √(1 − ᾱ_t) ε, t)‖²
until converged
```

repeat

$\mathbf{x}_0 \sim q(\mathbf{x}_0)$

$t \sim \mathrm{Uniform}(\{1, \ldots, T\})$

$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

Take gradient descent step on

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

until converged

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

for $t = T, \ldots, 1$ **do**

$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$$

end for

return $\mathbf{x}_0$

Continuos Diffusion Model

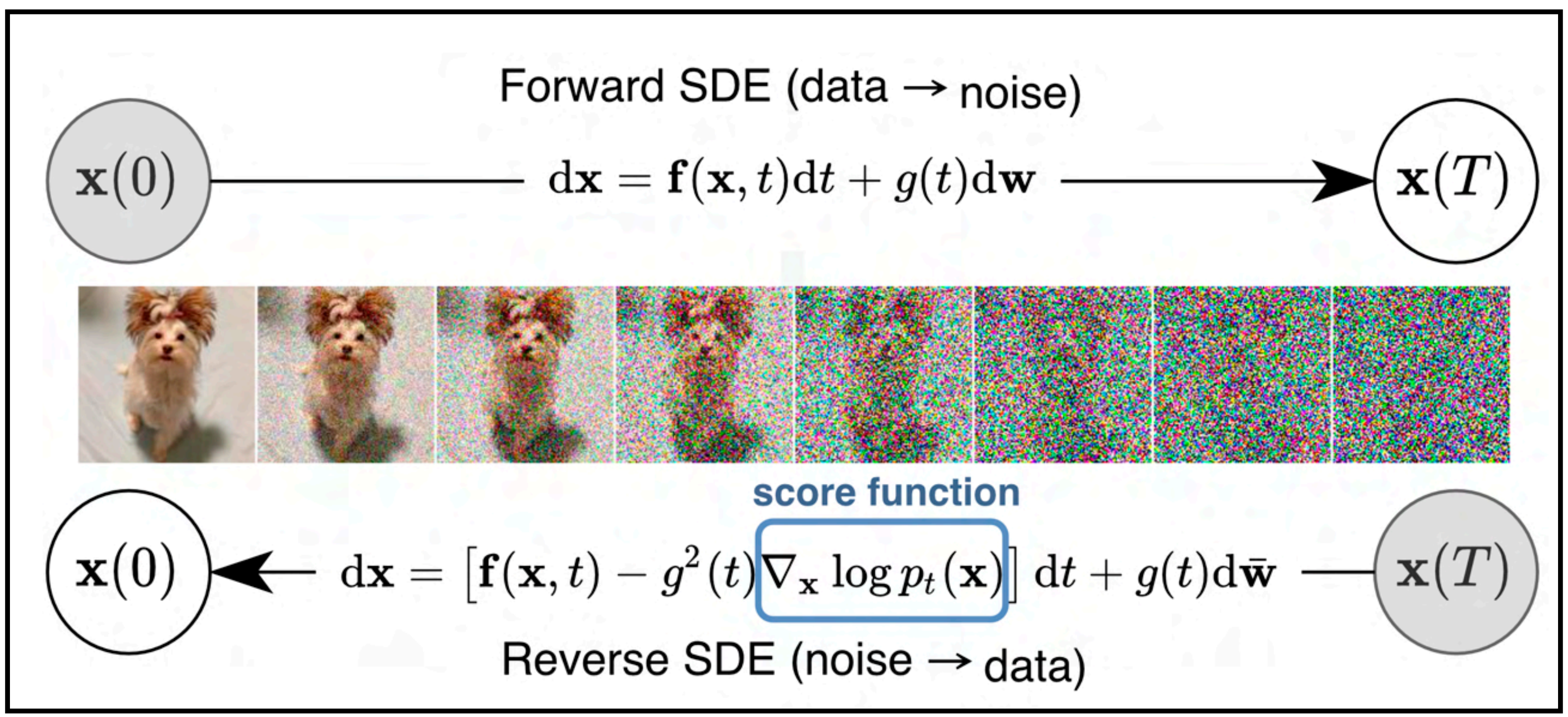
Forward SDE: $$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$$

Wiener process/ Brownian motion (independent Gaussian increments)

Drift term

Diffusion term

(Correspond to the noise schedule in discrete case)



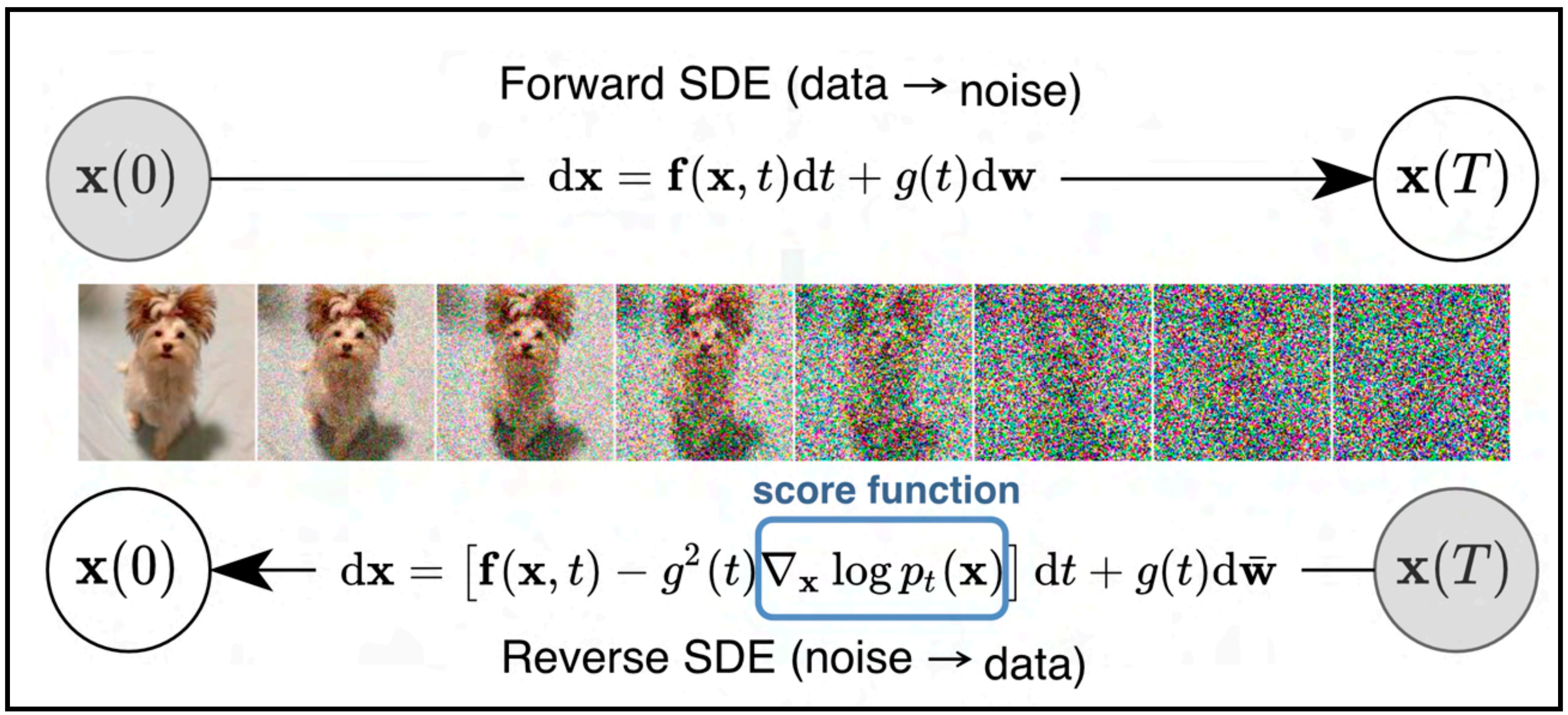
Continuos Diffusion Model

Probability density of x(t)

Reverse SDE: $$\mathrm{d}\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})]\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}$$

Score function

Reverse of a diffusion process is also a diffusion



Continuos Diffusion Model

Reverse SDE: $\mathrm{d}\mathbf{x} = [\mathbf{f}(\mathbf{x},t) - g(t)^2 \boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{x})}]\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}$

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_t \Big\{ \lambda(t) \mathbb{E}_{\mathbf{x}(0)} \mathbb{E}_{\mathbf{x}(t)\mid\mathbf{x}(0)} \Big[\big\| \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}(t),t) - \boxed{\nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))} \big\|_2^2 \Big] \Big\}$$

Learn to approximate score function with neural network