Machine learning introduction

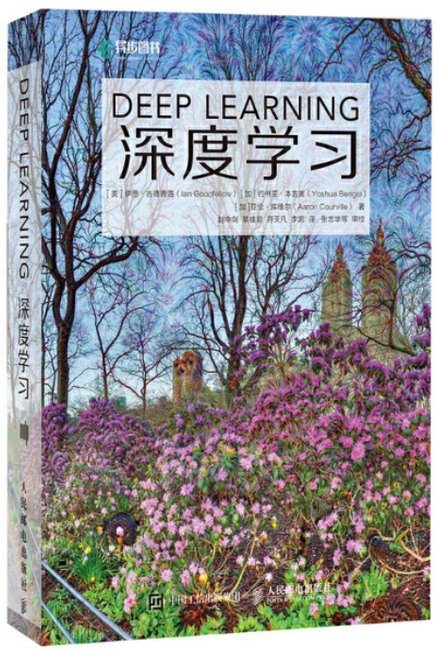
Rui Zhang (张 瑞)

rui.zhang@cern.ch

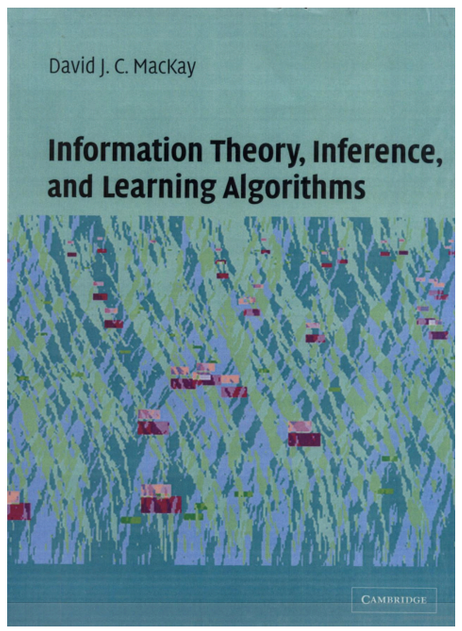
University of Wisconsin-Madison, Wisconsin

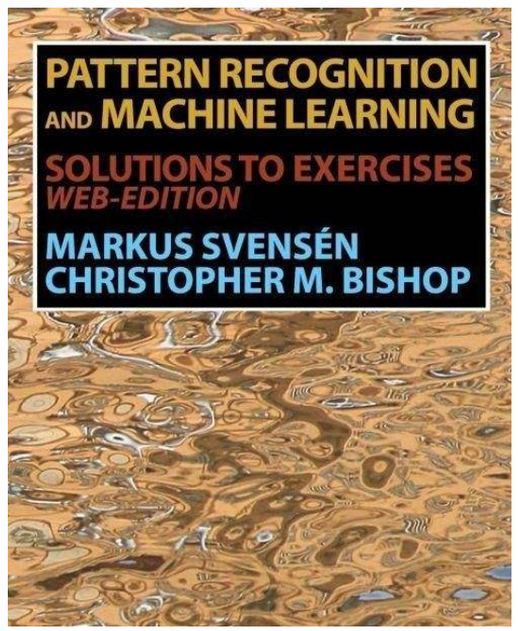
03 Aug, 2024

Books and references



电子版





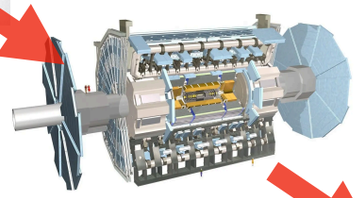
电子版

- 台湾大学李宏毅视频课程(https://github.com/Fafa-DL/Lhy_Machine_Learning)
- 深度学习论文精读(https://github.com/mli/paper-reading)

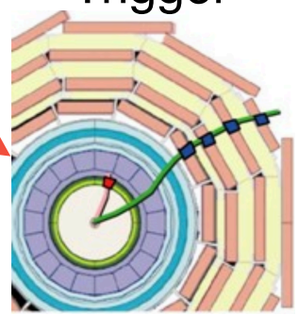
A typical LHC Physics Analysis Workflow

Nature

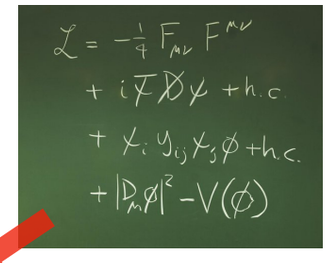
Experiment



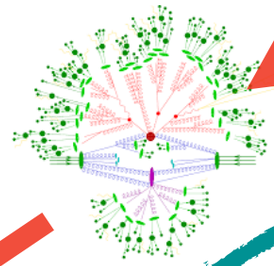
Trigger



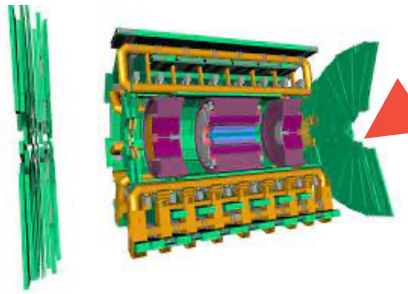
Theory of physics



Event generation



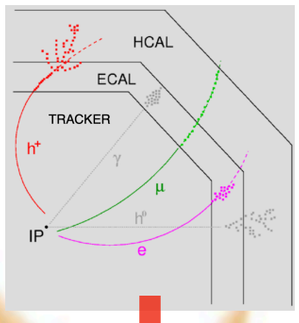
Detector response simulation



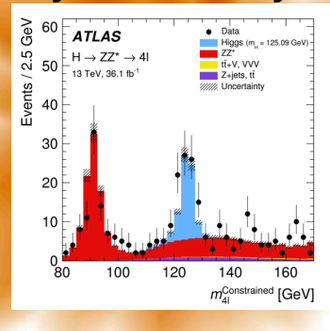
Data from nature

Simulation from knowledge

Physics object reconstruction



Physics analysis

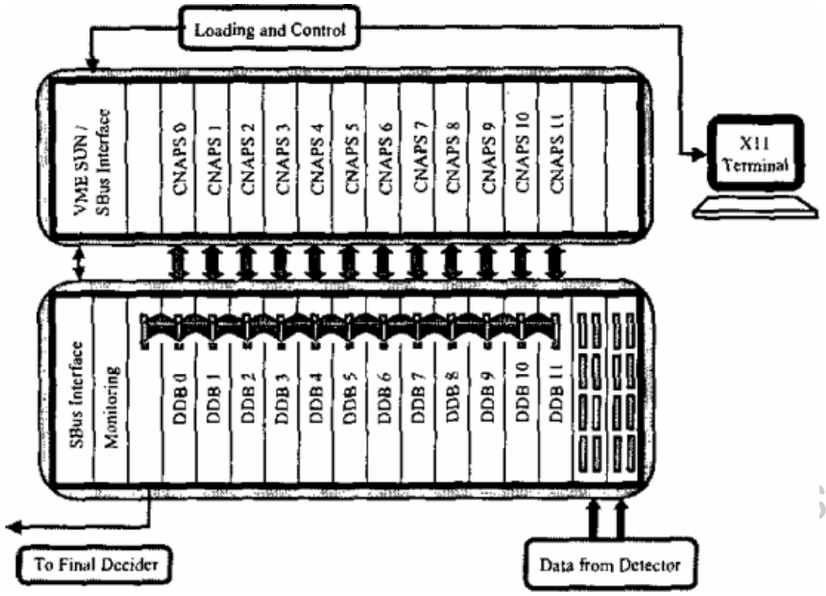


ML is an old friend of HEP

Nature

Experiment

Trigger



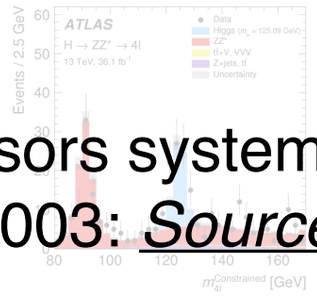
Theory of physics

Event simulation

Interaction with simulated detector

Reconstruction

Sig vs bkg separation



Connected network of adapted processors system (CNAPS) used for H1 trigger system, 2003: *Source*

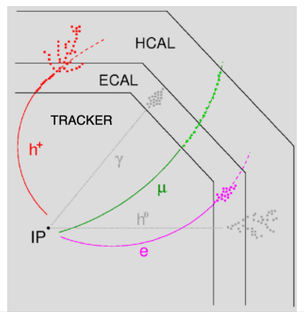
ML is an old friend of HEP

Nature

Experiment

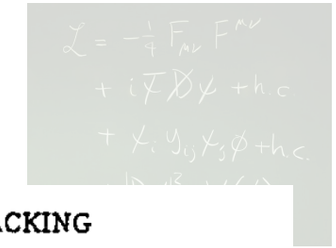
Trigger

Reconstruction

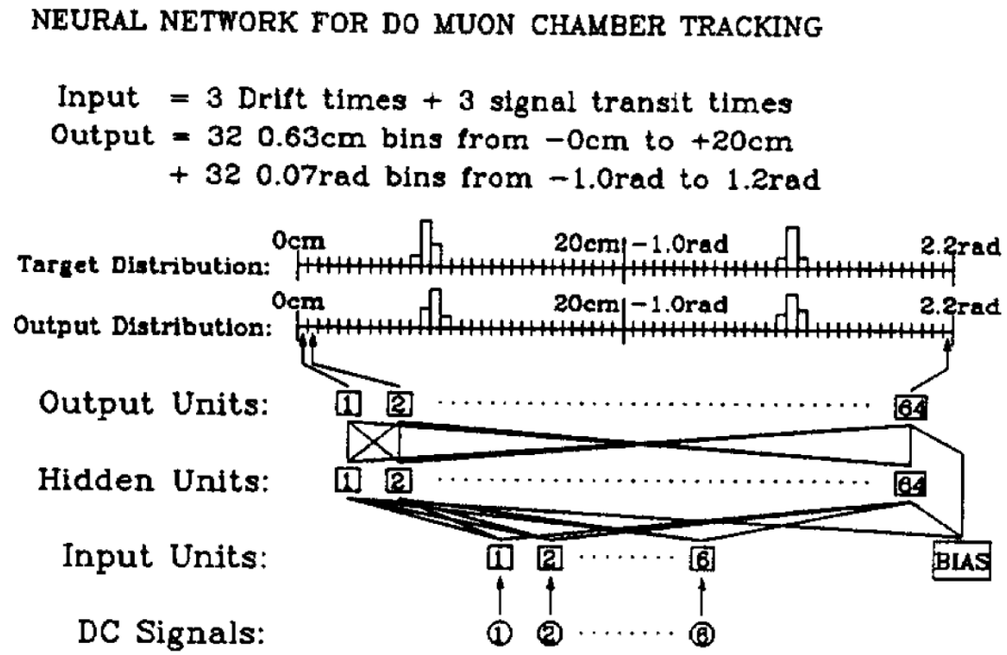


Sig vs bkg separation

Theory of physics



Event simulation



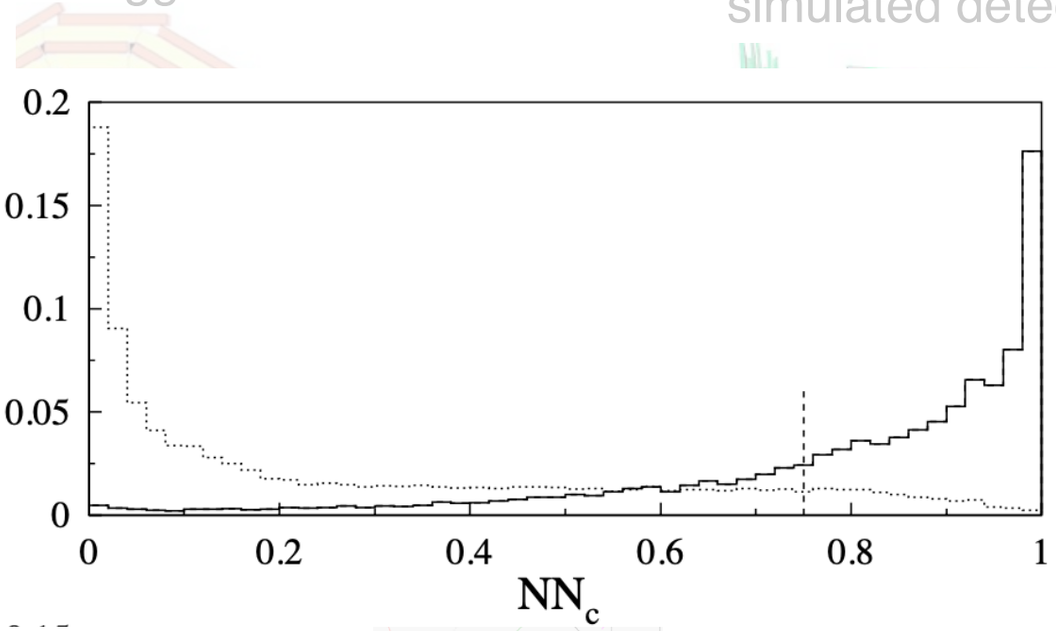
Primary vertexing based on the fired wires at E735, Fermilab, 1991: *Source*

ML is an old friend of HEP

Nature

Experiment

Trigger



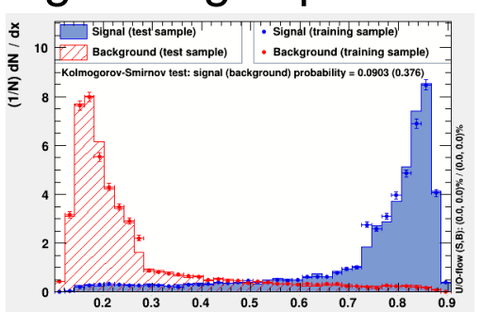
Theory of physics

Event simulation

Interaction with simulated detector

Selection of b hadrons at ALEPH, 1999:
Source

Sig vs bkg separation

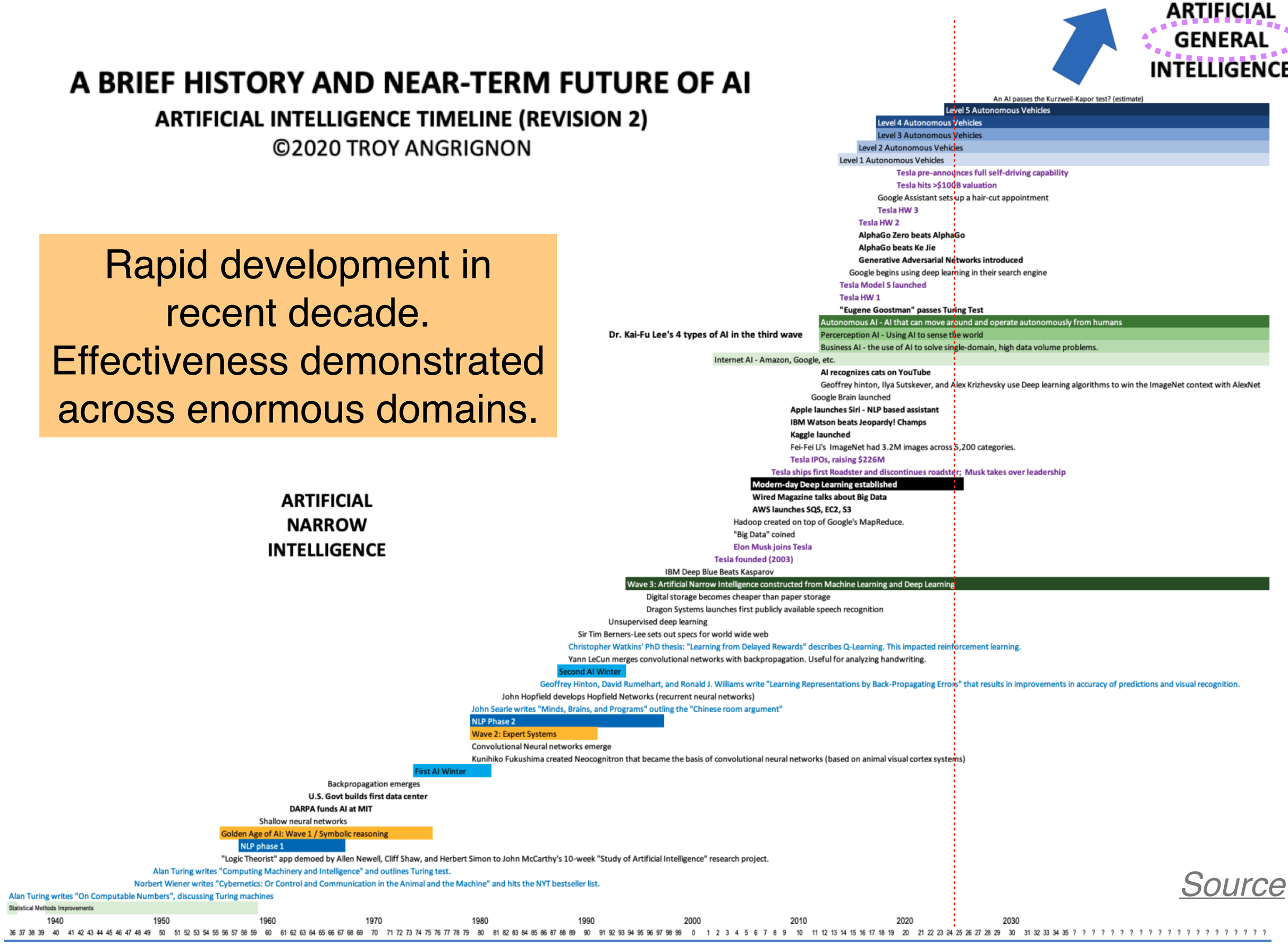


A lot more, see reviews in 1993 and in 1999

A BRIEF HISTORY AND NEAR-TERM FUTURE OF AI

ARTIFICIAL INTELLIGENCE TIMELINE (REVISION 2)

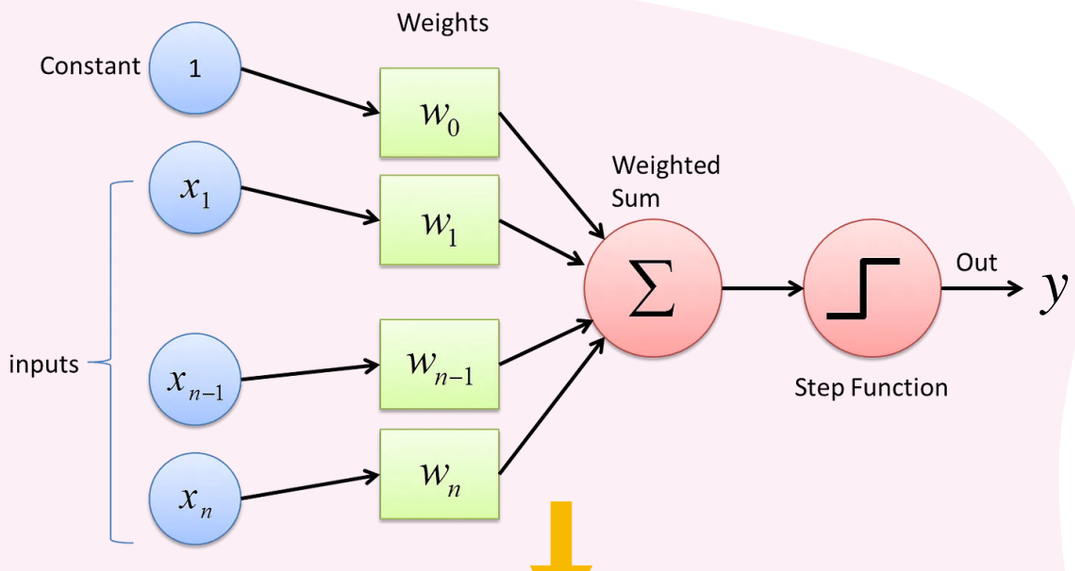
Rapid development in recent decade.
Effectiveness demonstrated across enormous domains.



Source

ML is not a magic

It's built upon linear algebra and information theory



$$\sigma\left(\Sigma\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}\right) = y$$

$$y = f_W(x)$$

Neural network is a function that maps input to output; "universal approximation theorem"

Learning procedure is to compress the input to output.

$$\begin{aligned} y_1 &= f_1(x) \\ y_2 &= f_2(x) \\ &\vdots \\ y_n &= f_n(x) \end{aligned}$$

Which function is close to truth?

Need to quantify "similarity" between y_i and y_{truth}.

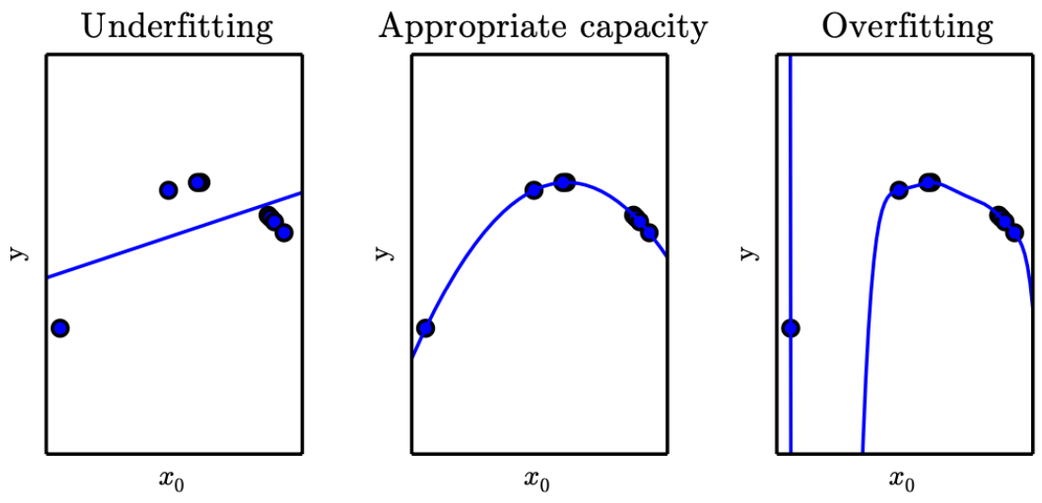
- Both are distributions (PDF)
- Also known as "loss" => min(Loss(y_i, y_{truth}))

Information theory offers measures for quantifying similarity

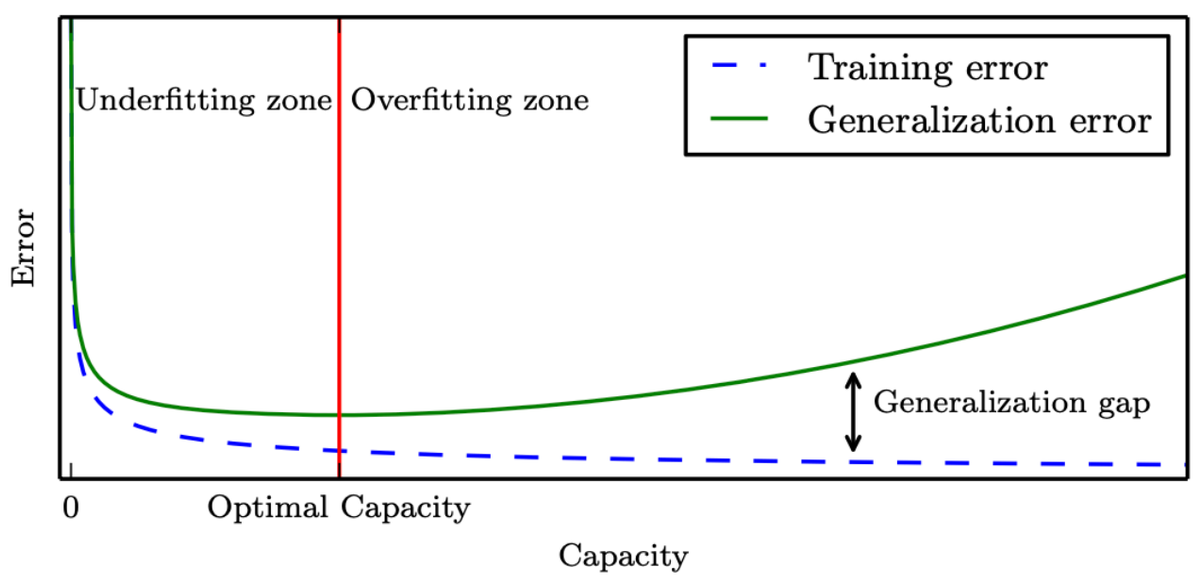
- Entropy: disorder of 1 PDF
- Divergence: disorder between 2 PDFs

ML is not a magic: capacity and regularisation

$$y = f_W(x)$$



- f_W has too few free parameters ➜ underfit
- f_W has too many free parameters ➜ overfit
- Avoid underfit/overfit? ➜ check on the test sample (generalisation error)



- Model's effective capacity can be affected by some factors:
  - Optimisation is very difficult and may not find the global optimum
  - Adding regularisation can limit the capacity

- An example regularisation: L2-norm

$$J(w) = \text{MeanSquaredError} + \lambda ||w||_2^2$$

$$||w||_2 = \sqrt{\sum_{k=1}^{n} |w_k|^2}$$

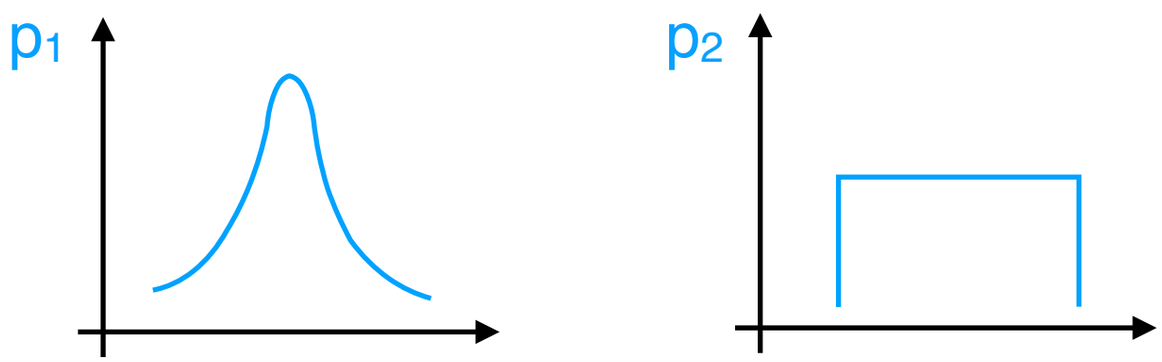
- Large weight are suppressed

ML is not a magic: distance between distributions

- In information theory, learning that an unlikely event has occurred is more informative than learning that a likely event has occurred.
  - $p(x_1) < p(x_2) \rightarrow I(x_1) > I(x_2)$
  - $I(x) >= 0$
  - If x_1, x_2 are independent, ie, $p(x_1) \cdot p(x_2) = p(x_1 + x_2)$, $I(x_1 + x_2) = I(x_1) + I(x_2)$

} $I = -\log p$

- Shannon entropy: $H(p) = E_{x \sim p}[I(x)] = -E_{x \sim p}[\log p\,(x)]$
  - Shannon entropy of a distribution is the expected amount of information in an event drawn from that distribution; when p is continuous, it is differential entropy
  - Distributions that are nearly deterministic (where the outcome is nearly certain) have low entropy; distributions that are closer to uniform have high entropy.



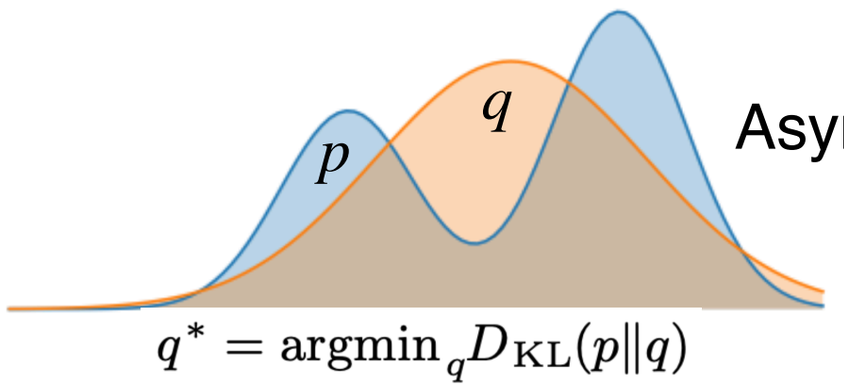
Who has larger Shannon entropy?

ML is not a magic: distance between distributions

Divergence is a measure of statistical distance between two distributions.

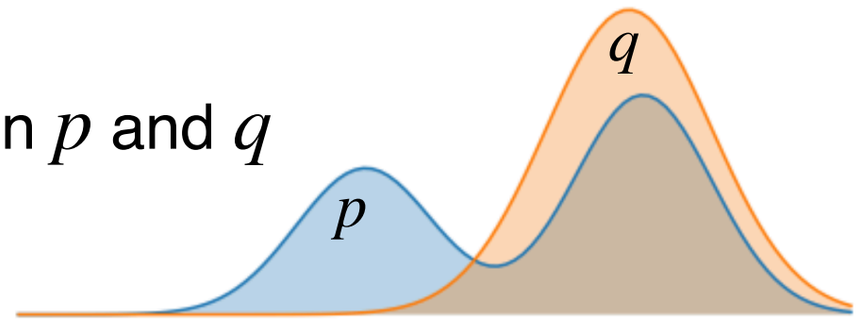
Most popular one: Kullback-Leibler (KL) divergence: $D_{\mathrm{KL}}(P\|Q) = \mathbb{E}_{\mathrm{x}\sim P}\left[\log \frac{P(x)}{Q(x)}\right]$

Information needed to send a message containing symbols drawn from P, when we use a code that was designed to minimise the length of messages drawn from probability distribution Q.



$q^* = \mathrm{argmin}_q D_{\mathrm{KL}}(p\|q)$

Asymmetric between p and q



$q^* = \mathrm{argmin}_q D_{\mathrm{KL}}(q\|p)$

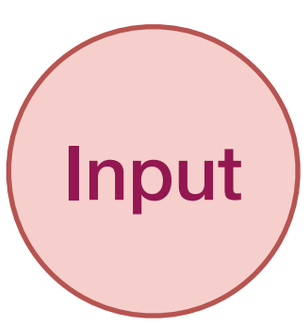
Wish all events in p will be found in q

Wish all events in q will be found in p

- Jensen-Shannon Divergence (JSD)
- Wasserstein Distance (Earth Mover's Distance)
- Total Variation Distance (TV Distance)
- Bhattacharyya Distance
- Hellinger Distance
- f-Devergence
- Rényi divergence
- …

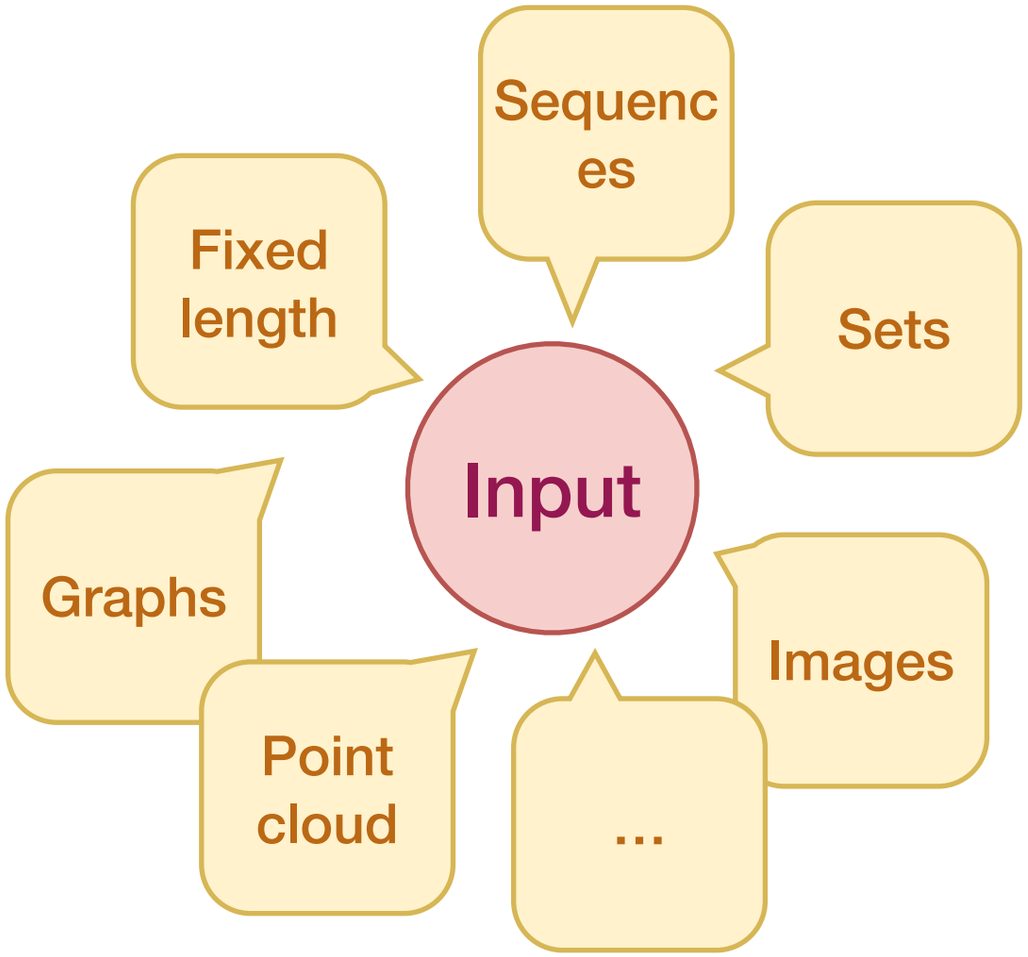
ML is a Tool for HEP

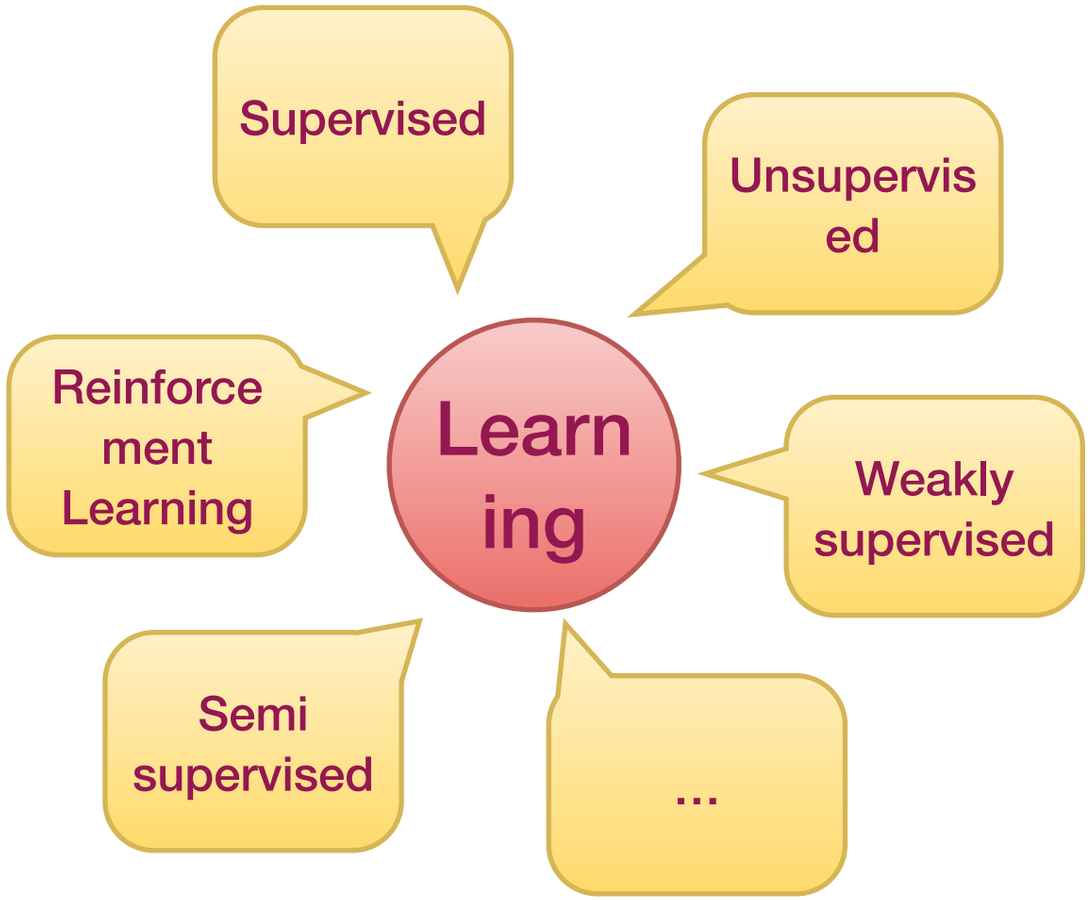
- Step 1: how to represent data
- Step 2: how to train the model



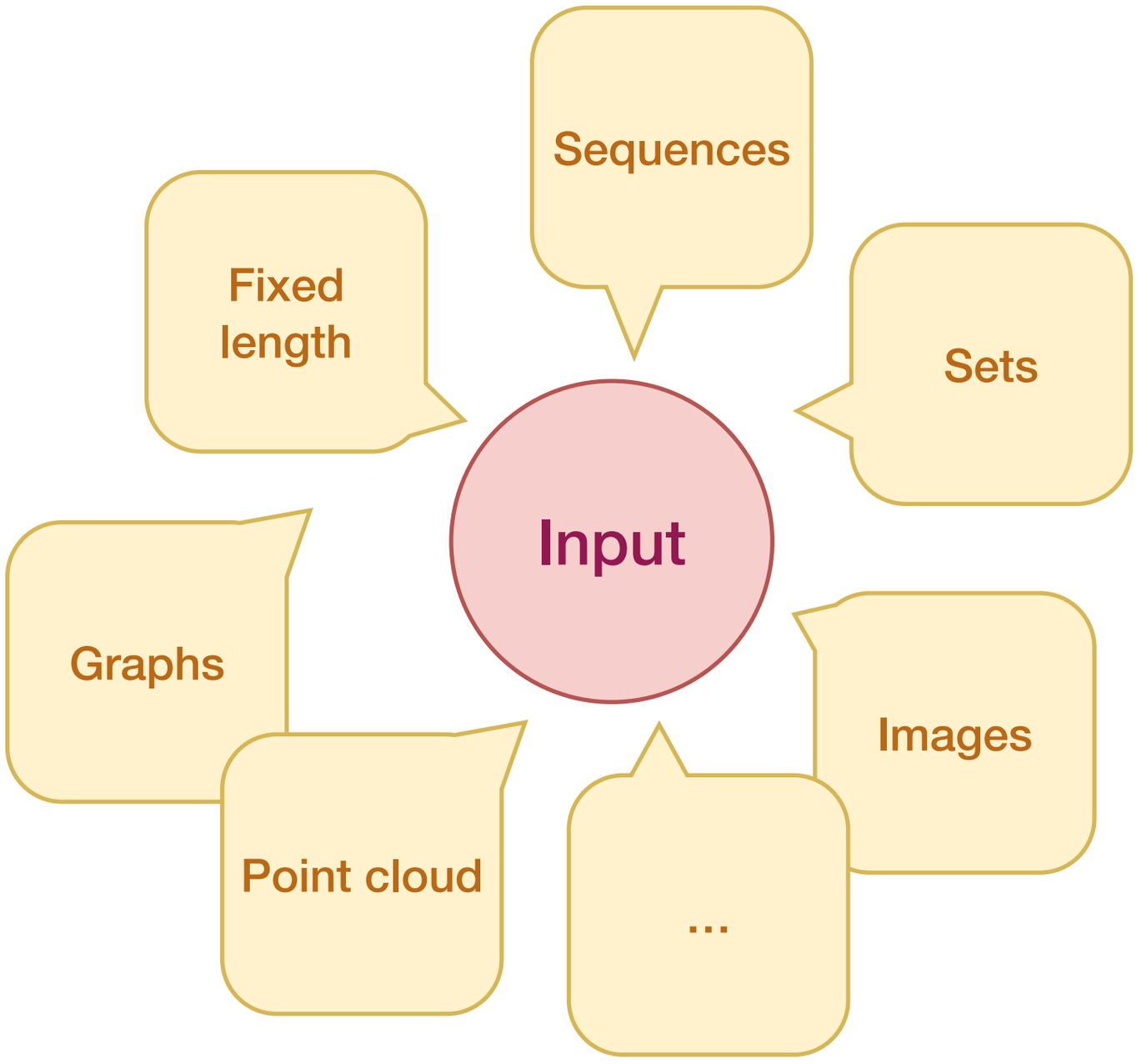
ML is a Tool for HEP

- Step 1: how to represent data
- Step 2: how to train the model





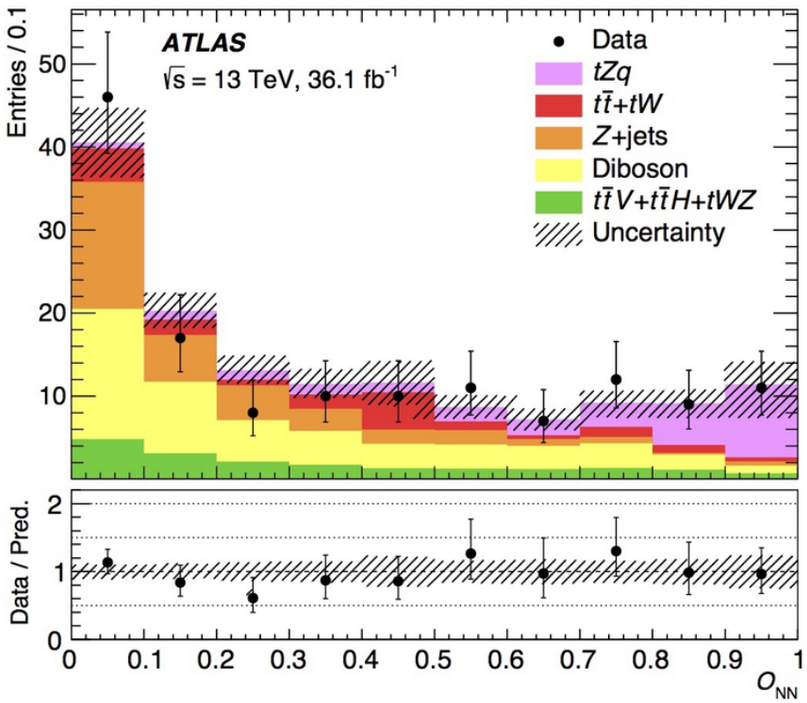
Step 1: how to represent data?



1.1 Input has fixed length

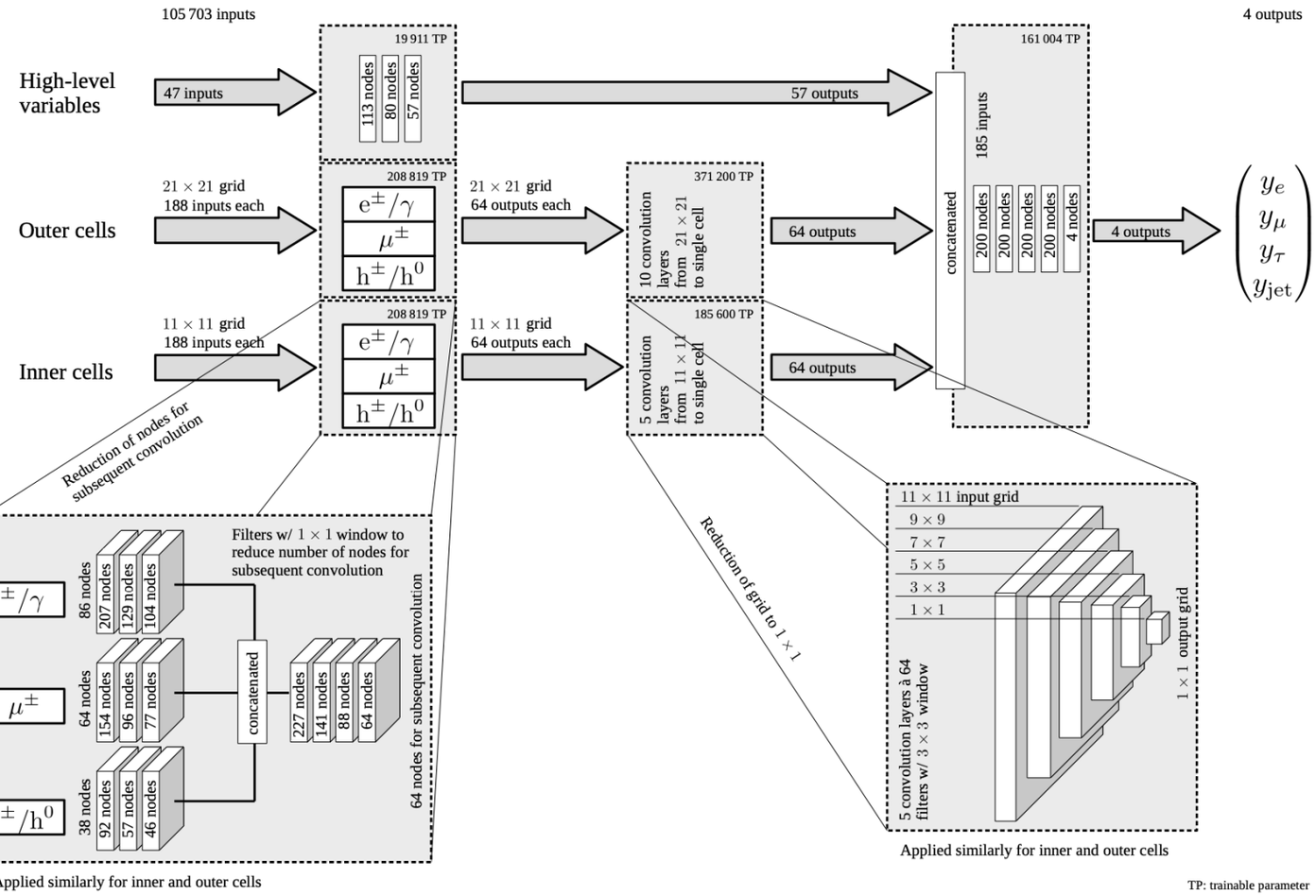
- Decide in advance variable list for training, then train a deep neural network / BDT

A typical signal extraction using NN



Phys. Lett. B 780 (2018) 557

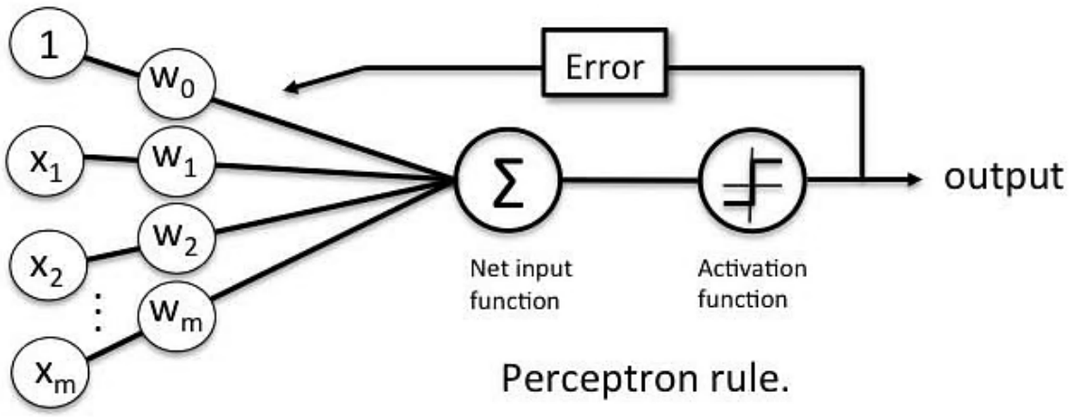
CMS tau ID deep network



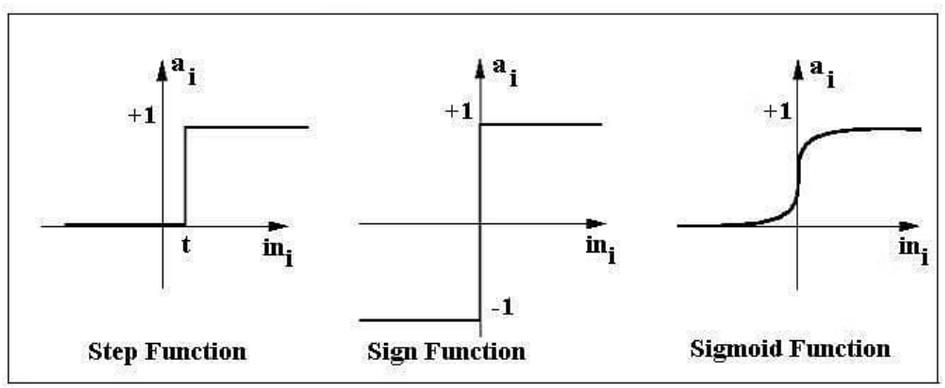
JINST 17 (2022) P07023

1.1 Perceptron / Deep neural network

- Perceptron was invented in 1943 by Warren McCulloch and Walter Pitts, sharing the very similar concepts with the modern neural network



- Step 1: multiply input with weights and sum over
- Step 2: apply activation function on the output



Non-linearity

reference

1.1 Why deep (MLP)? Example of XOR

XOR truth table

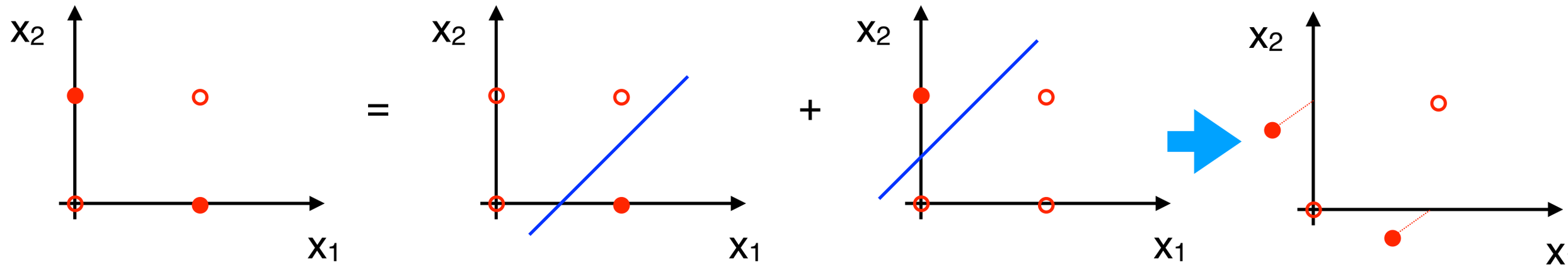
x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

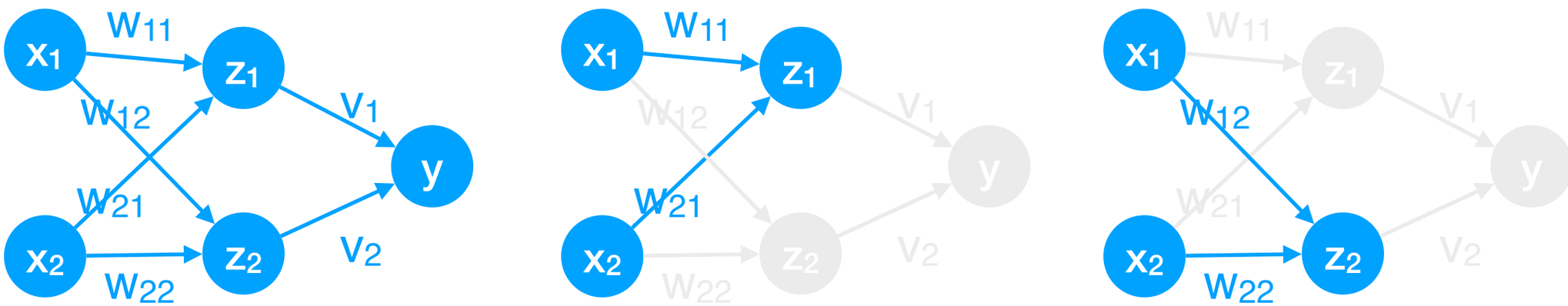
=

x_1	$\bar{x}_2$	$x_1\bar{x}_2$
0	1	0
0	0	0
1	1	1
1	0	0

+

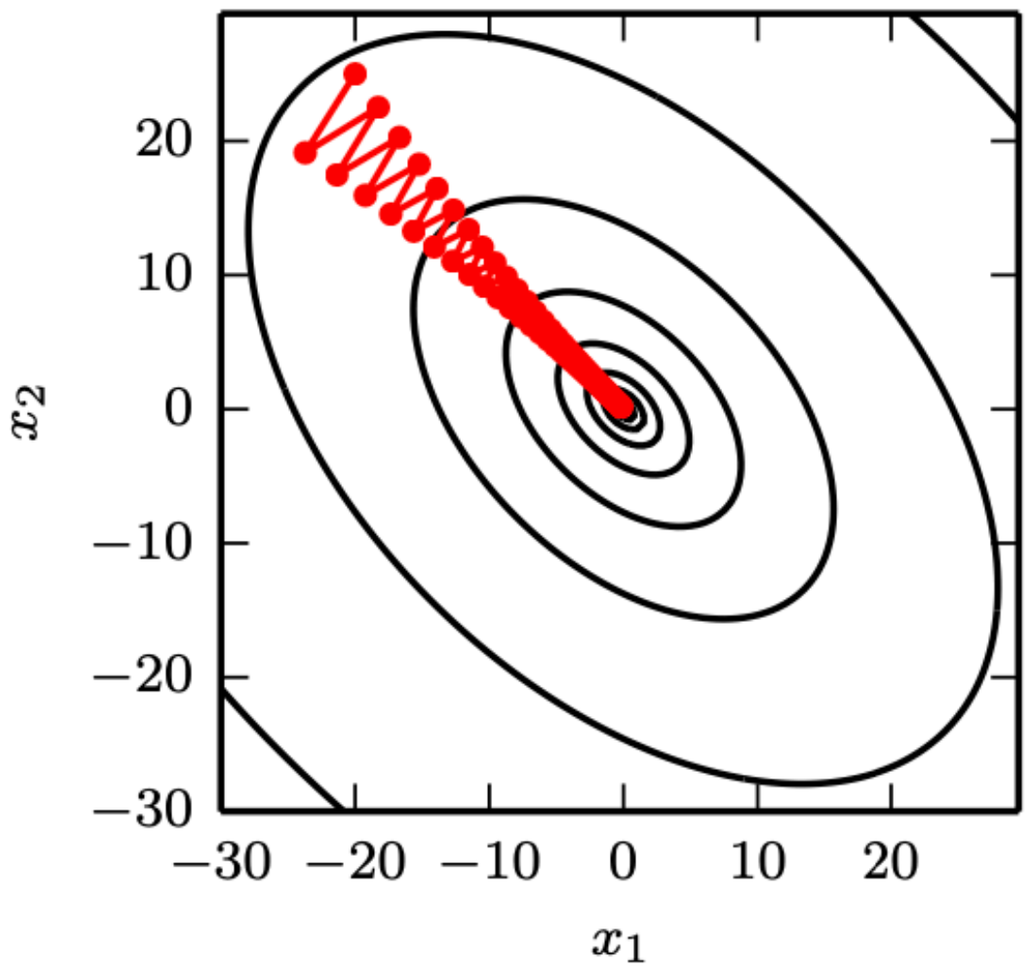
$\bar{x}_1$	x_2	$\bar{x}_1x_2$
1	0	0
1	1	1
0	0	0
0	1	0



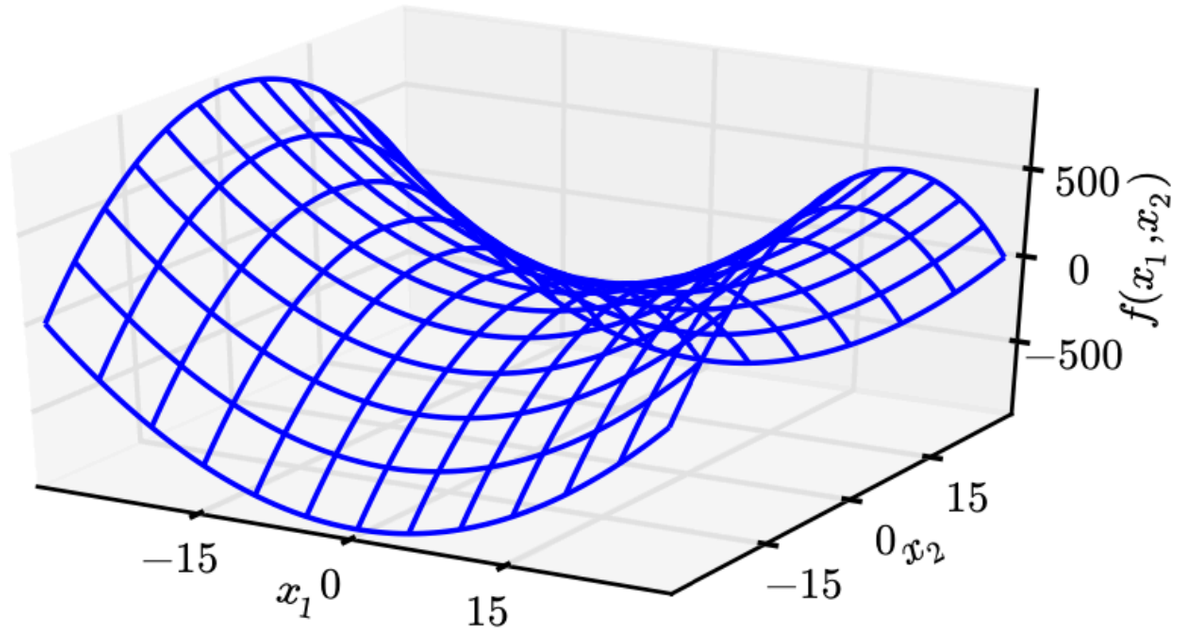


Multiple layers increase non-linearity of the function represented by the NN

1.1 Gradient decent



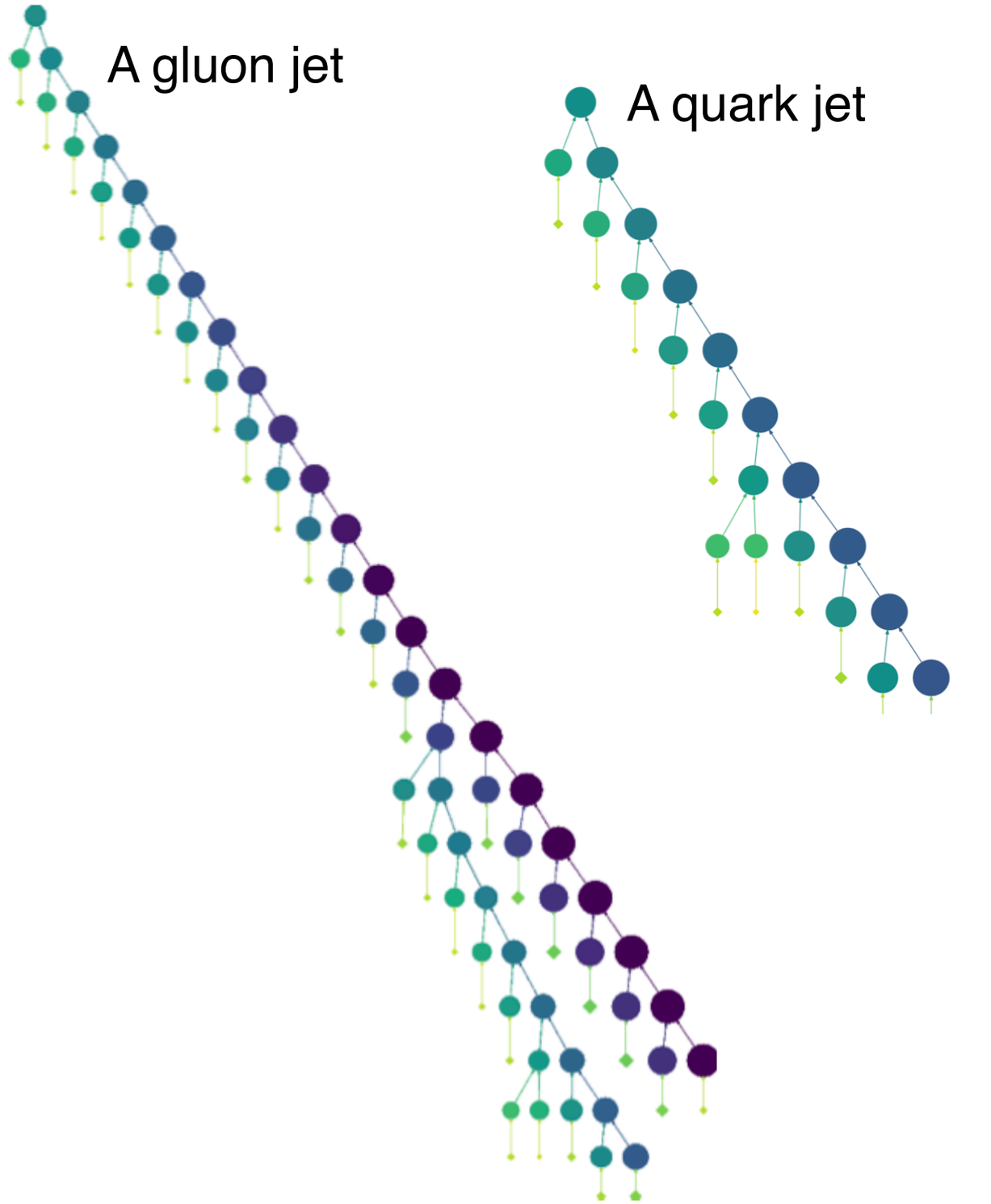
细长的二次函数类似一个长峡谷。梯度下降把时间浪费于在峡谷壁反复下 降，因为它们是最陡峭的特征



x 是 f 某个横截面 的局部极大点，却是另一个横截面的局部极小点

1.2 Input as sequences

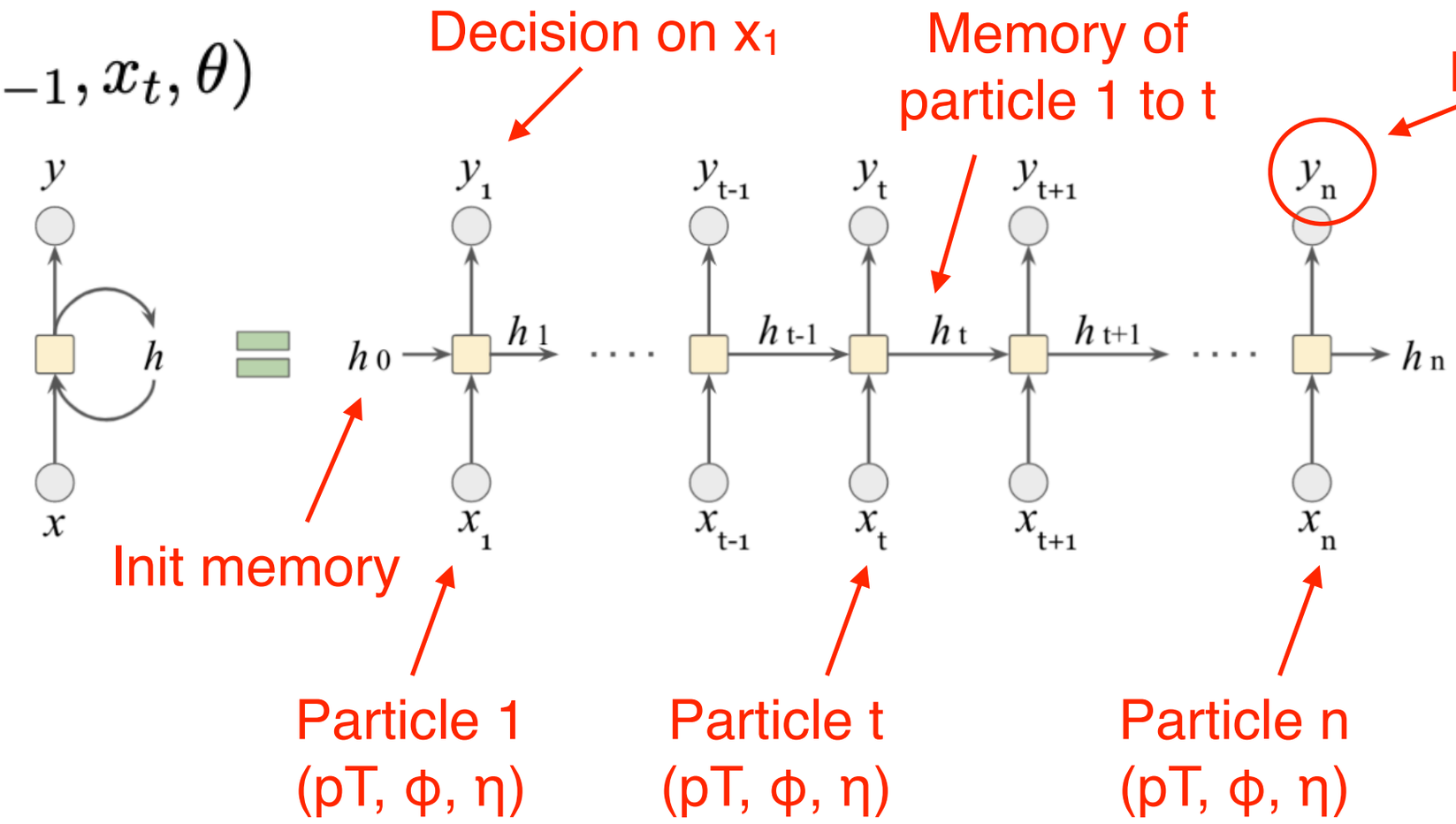
- In some situations, fixed length is not suitable
  - e.g. Jets contain a variable number of particles
  - Recurrent Neural Networks shows great performance for Natural Language Processing tasks
  - Information across the entire sequence can be accumulated and used



Recursive Neural Networks in Quark/Gluon Tagging

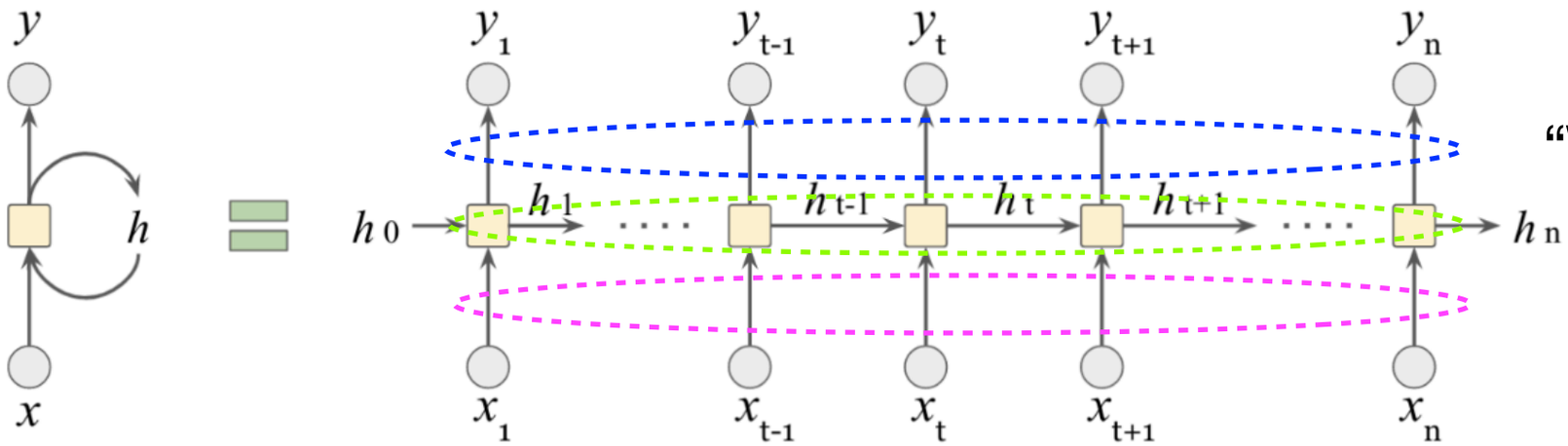
1.2 Recurrent Neural Networks

$$h_t = g_h(h_{t-1}, x_t, \theta)$$



1.2 Recurrent Neural Networks

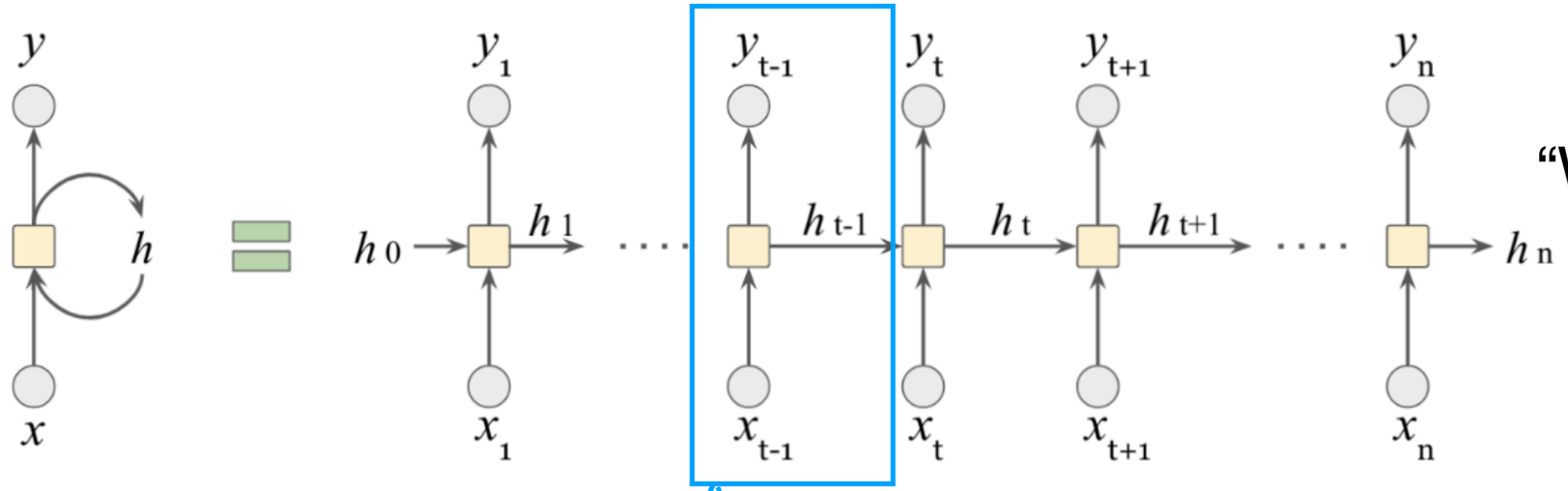
$$h_t = g_h(h_{t-1}, x_t, \theta)$$



"Weight sharing"

1.2 Recurrent Neural Networks

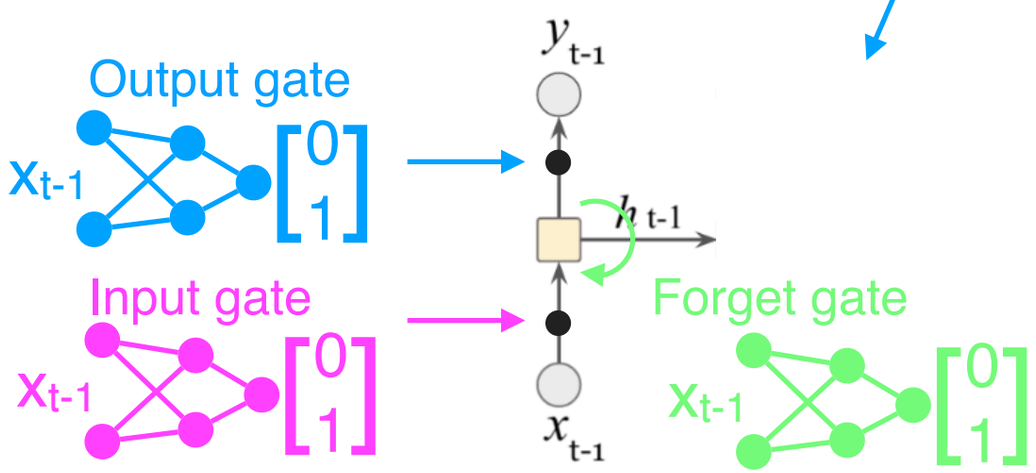
$$h_t = g_h(h_{t-1}, x_t, \theta)$$



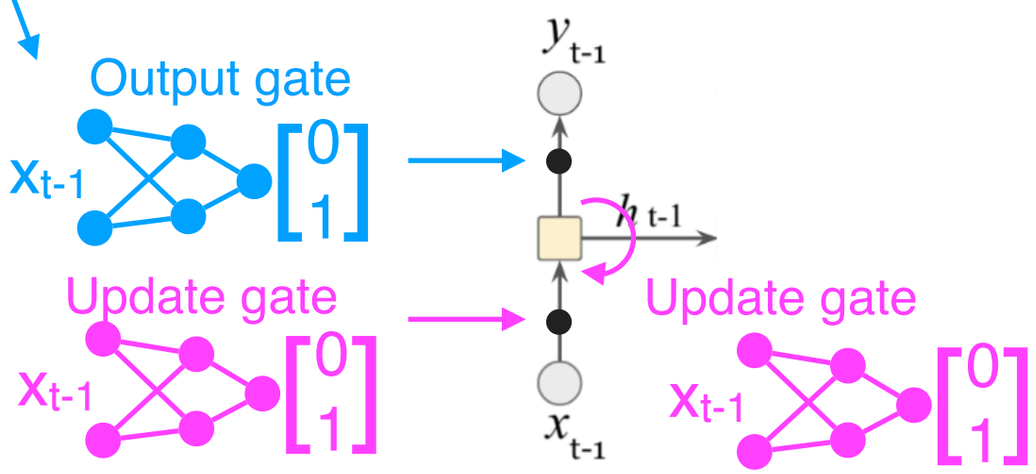
"Weight sharing"

However, in practise, RNN is not capable to deal with complex tasks since its output (y_t) is affected by the previous memory (h_{t-1}) too much.

Long short-term memory (LSTM)

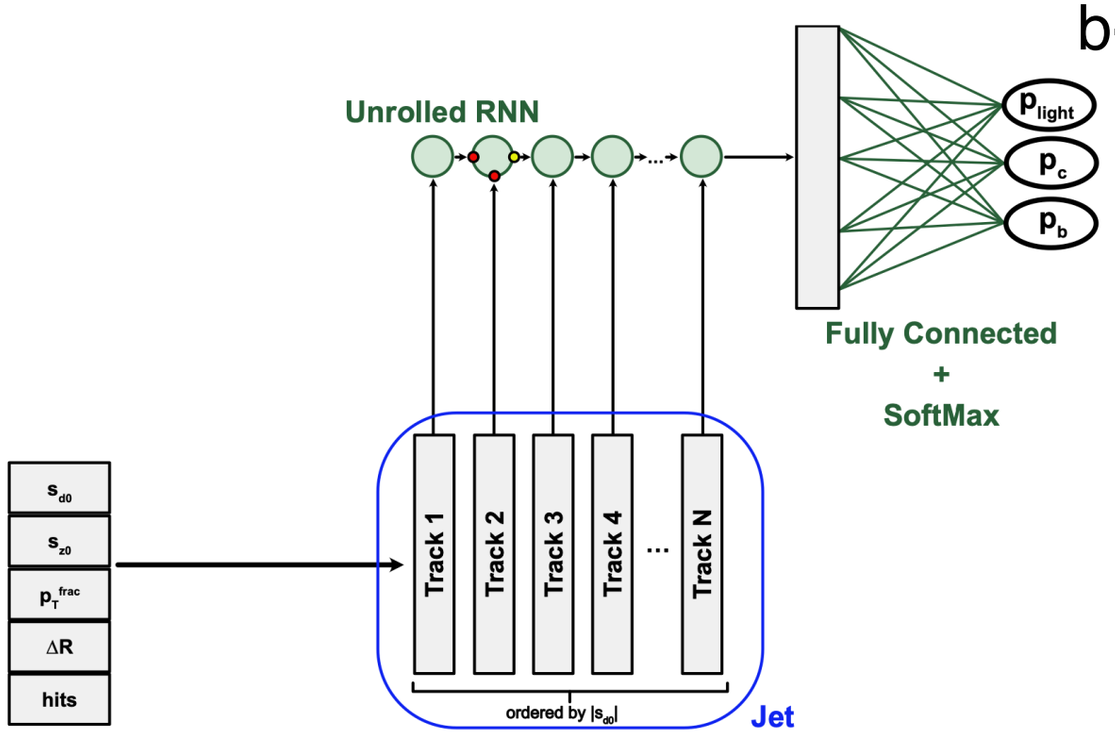


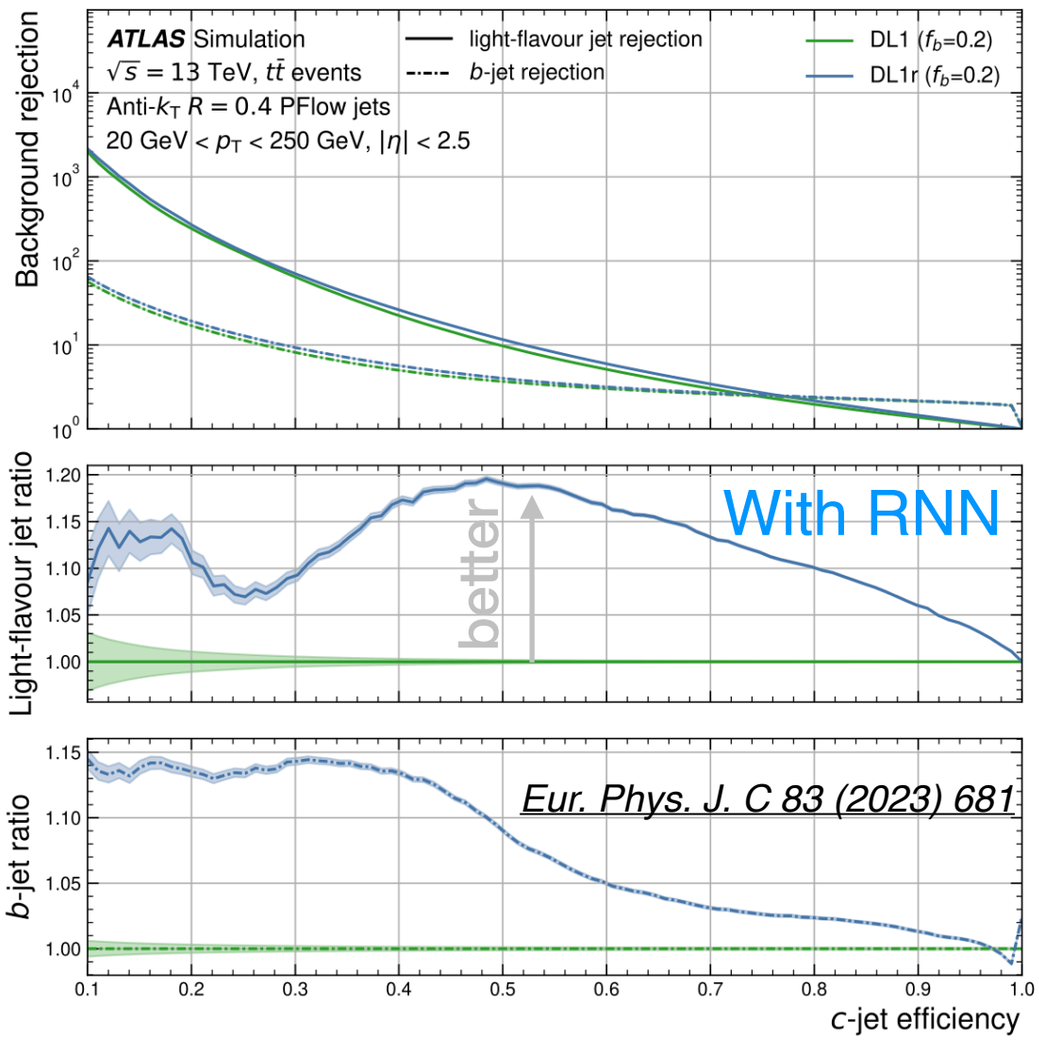
Gated Recurrent Unit (GRU)



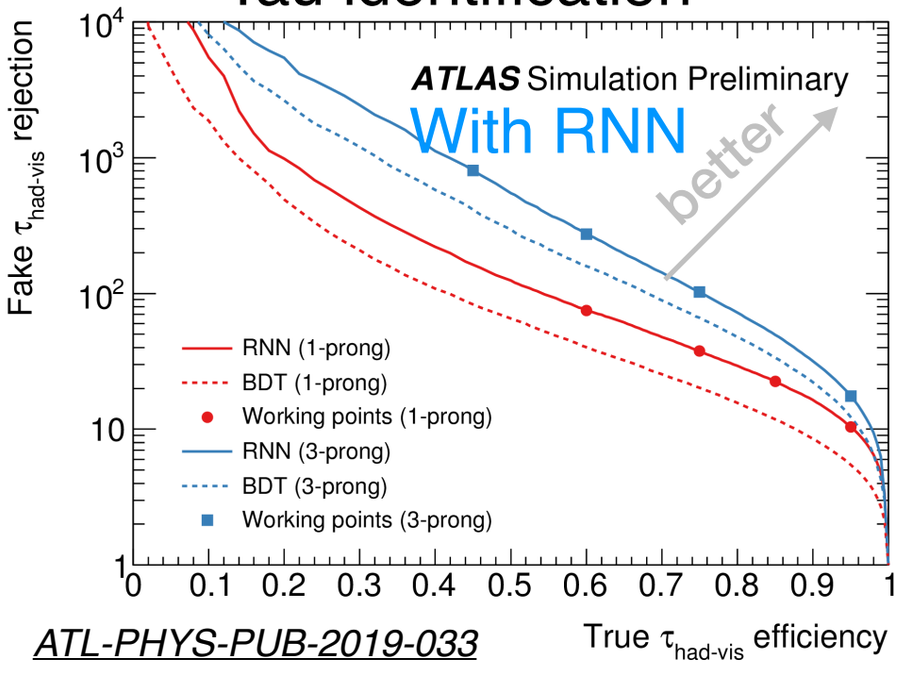
1.2 RNN application

b-tagging





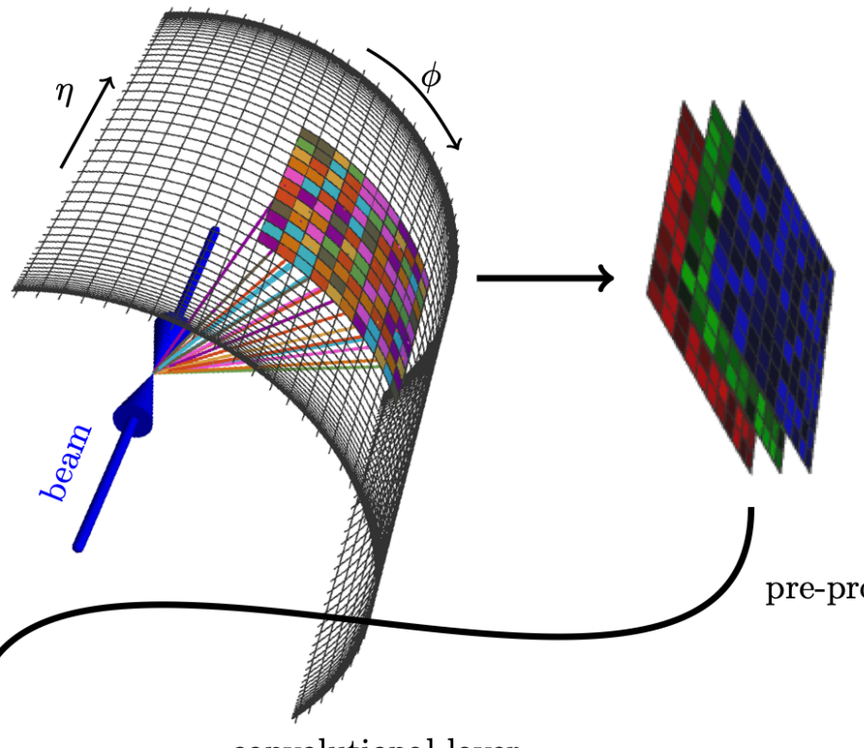
Tau identification

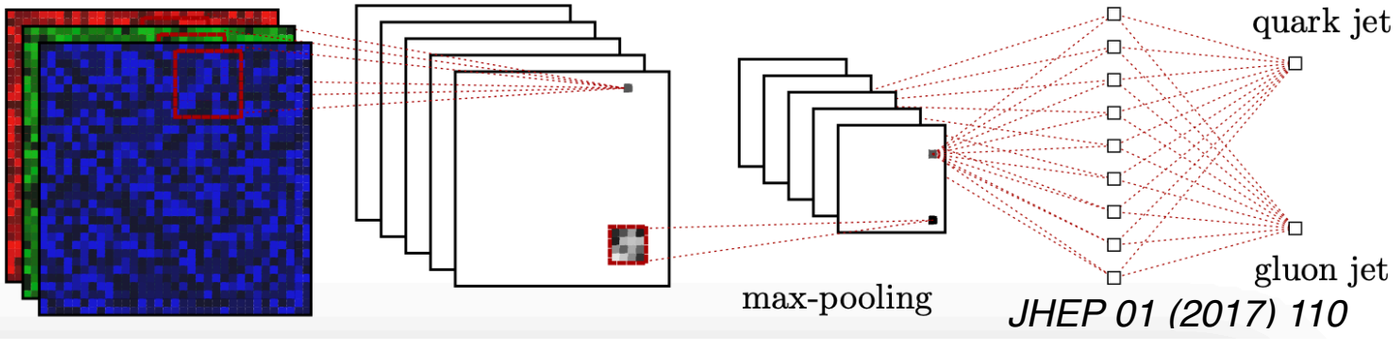


ATL-PHYS-PUB-2019-033

1.3 Input as images

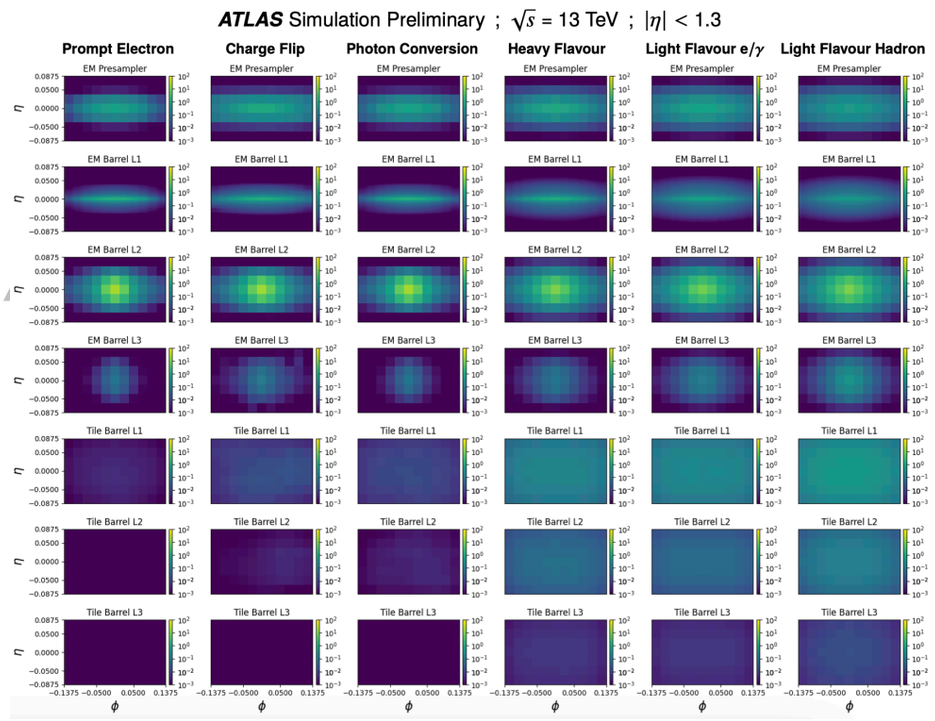
- Jets can be viewed as images
  - So as electrons





JHEP 01 (2017) 110

Electron classes

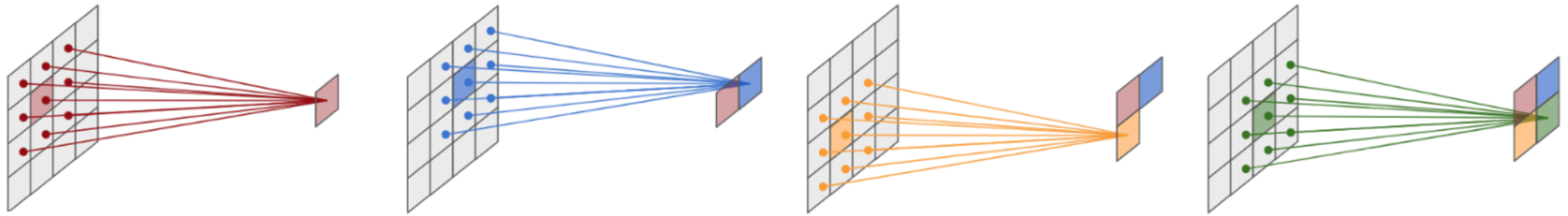


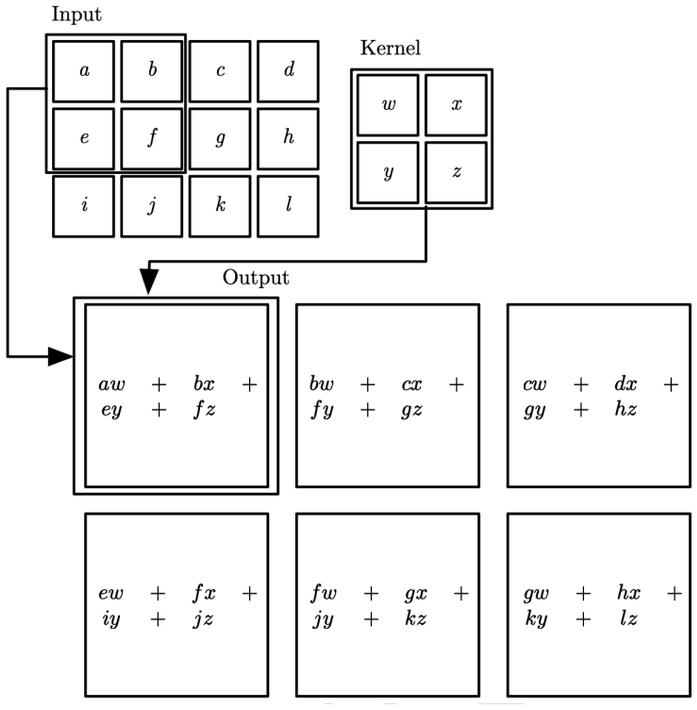
ATL-PHYS-PUB-2023-001

1.3 Convolutional neural network

- Convolutional neural network shows great performance for computer vision tasks
  - Nice features: “sparse interactions”: kernel’s dim is much smaller than image’s dim
  - “parameter sharing”: kernel is shared by different patches of the image
  - “equivariance” if f(g(x)) = g(f(x)), then f and g are equivariance

Convolution operation:





PDG Machine Learning

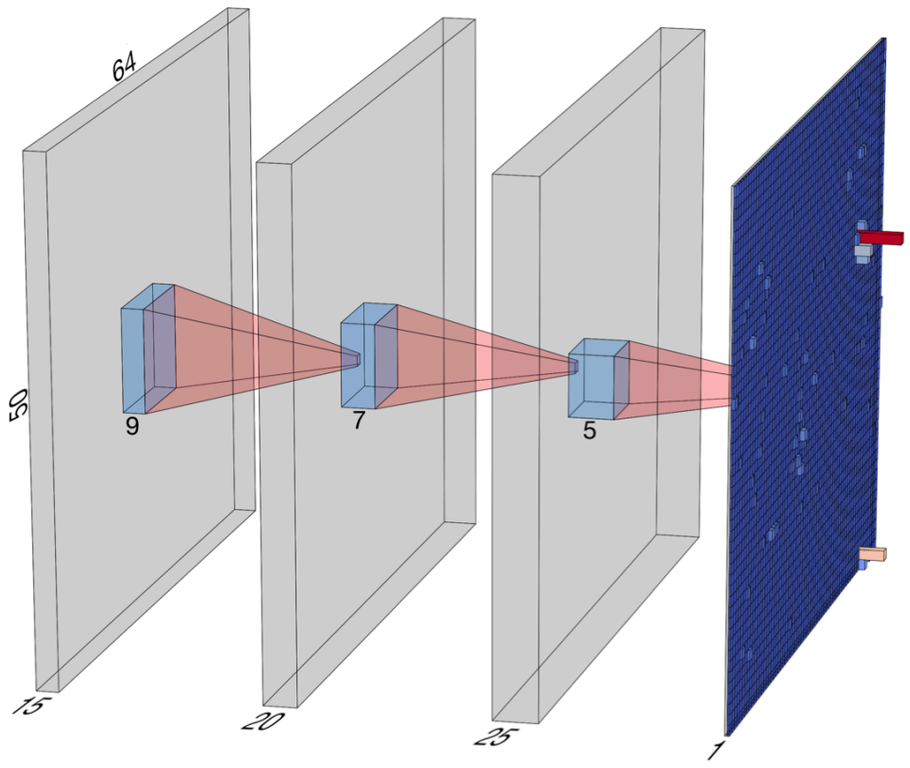
Goodfellow et al. Deep learning. MIT press, 2016.

1.3 CNN applications

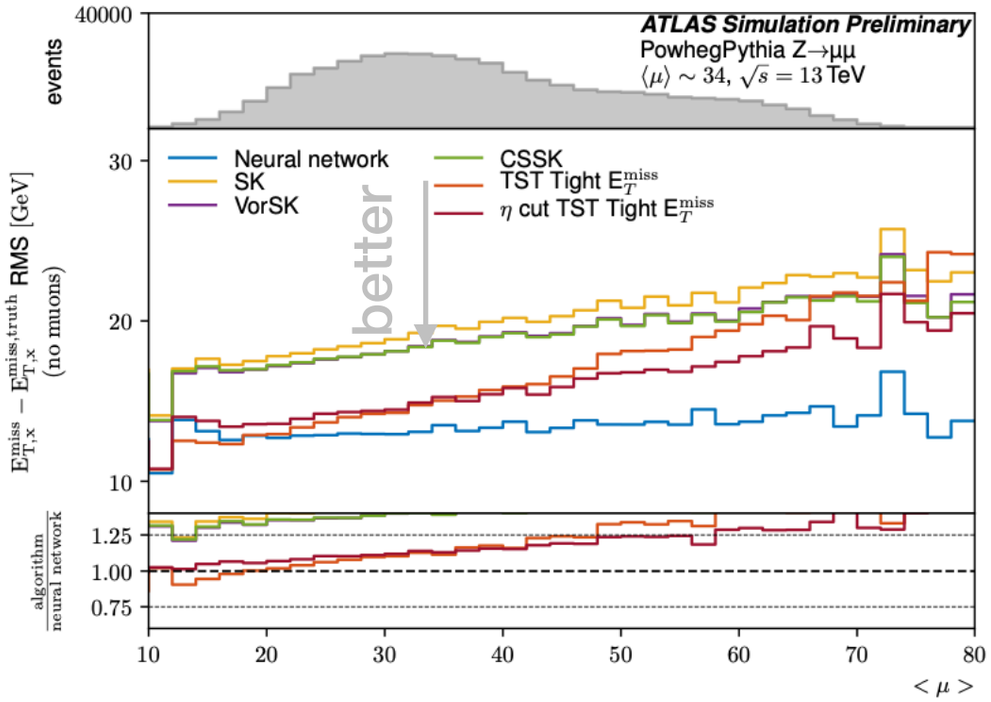
E_T^{miss} reconstruction

ATL-PHYS-PUB-2019-028

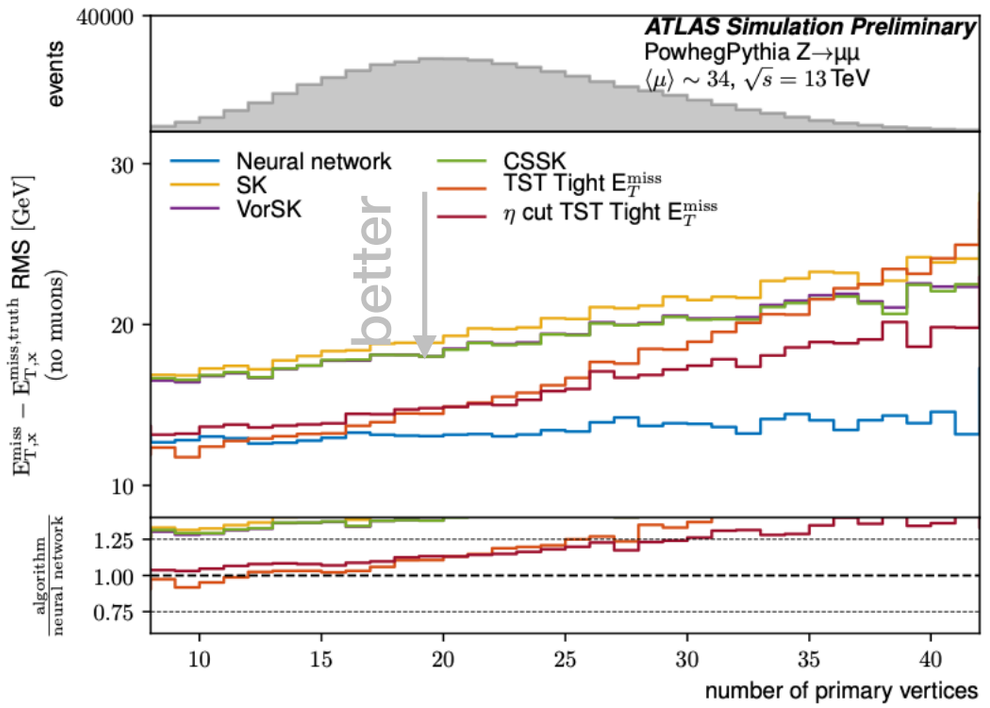
An event



Convolution

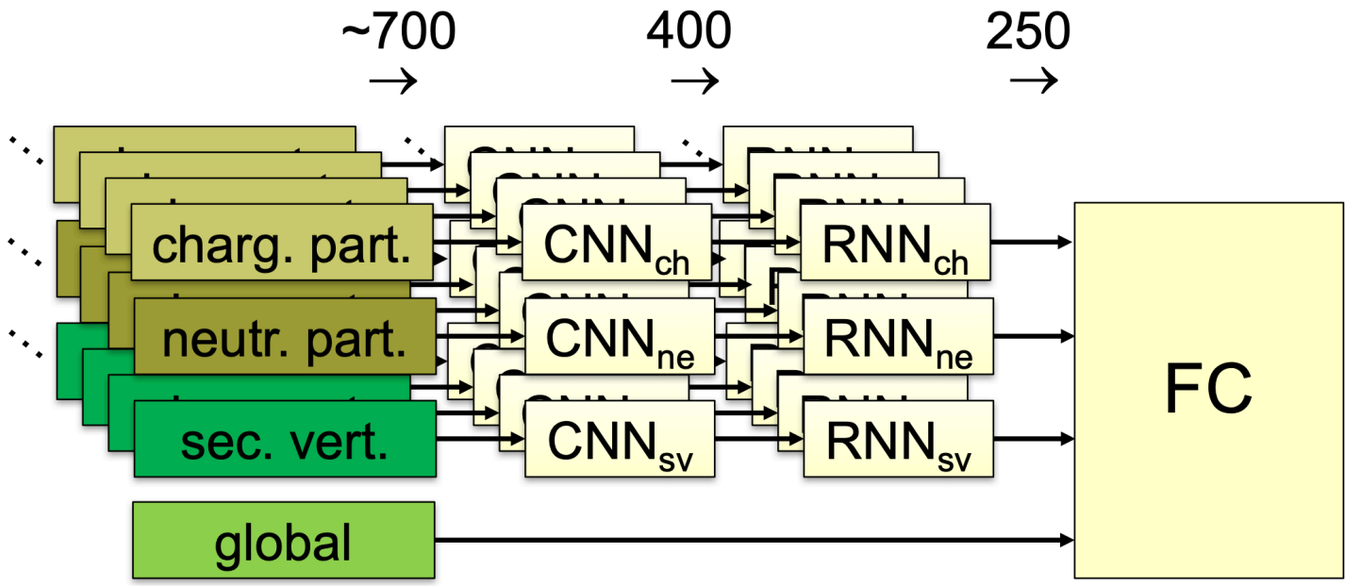


Robust against μ and primary vertices



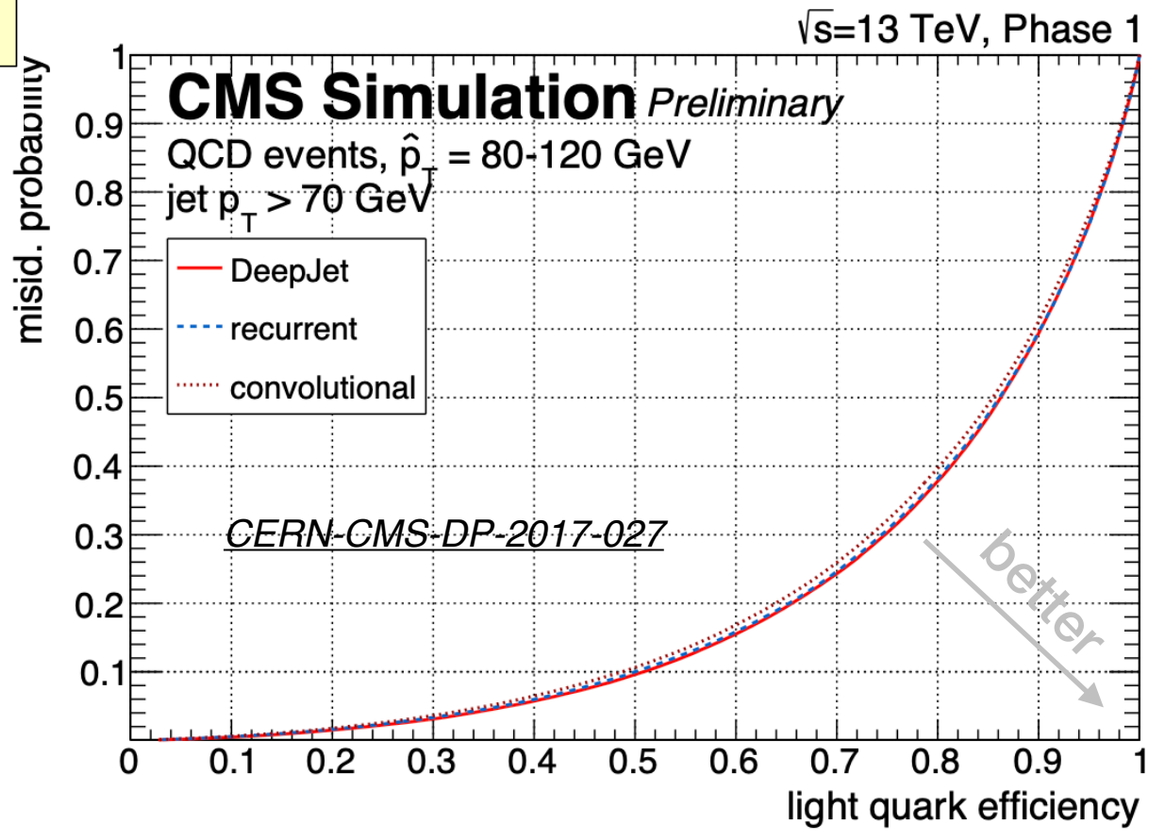
Hybrid: DNN + RNN + CNN application

Particle and vertex based DNN: DeepJet



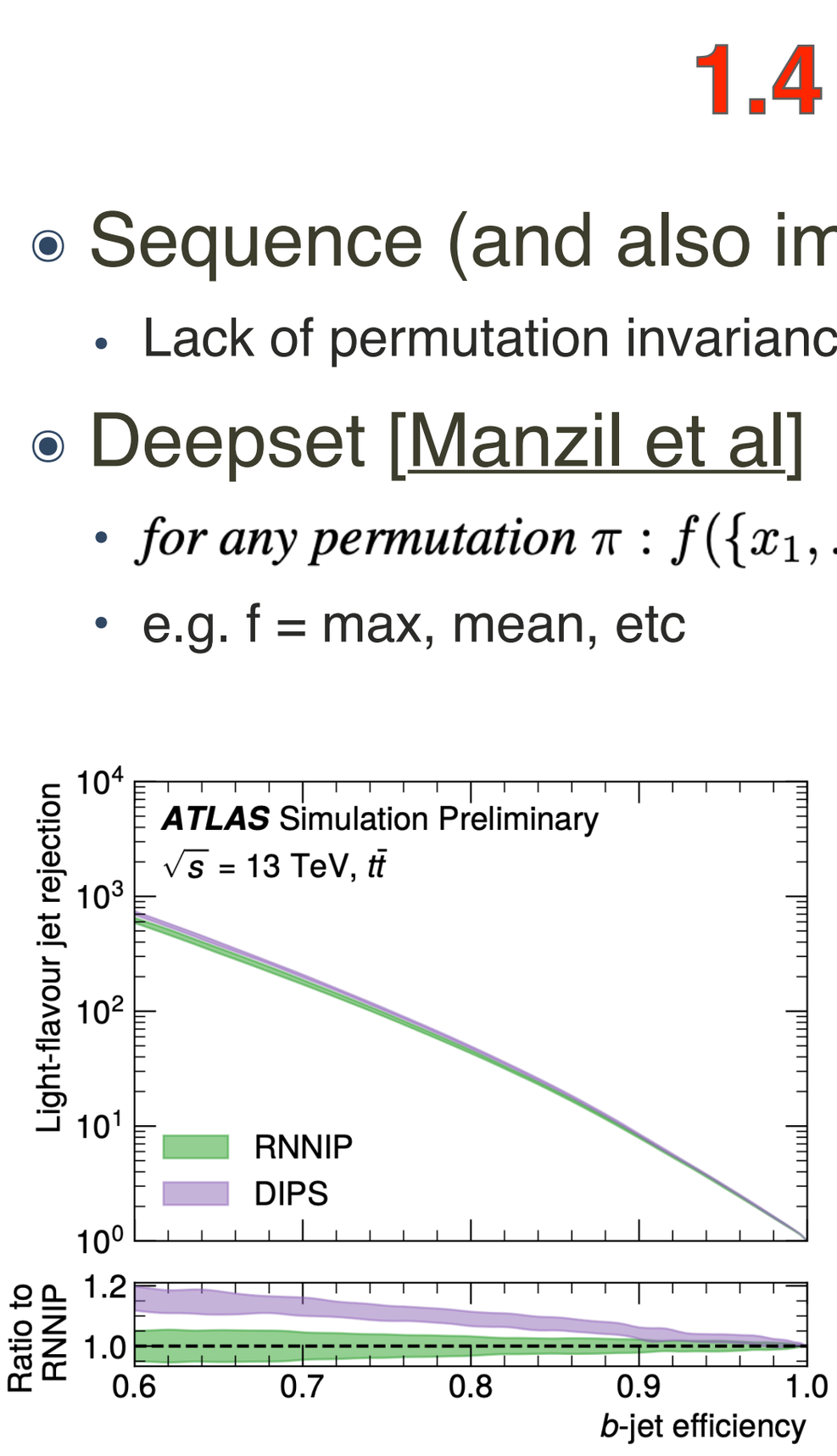
~ 700 inputs and 250.000 model parameters

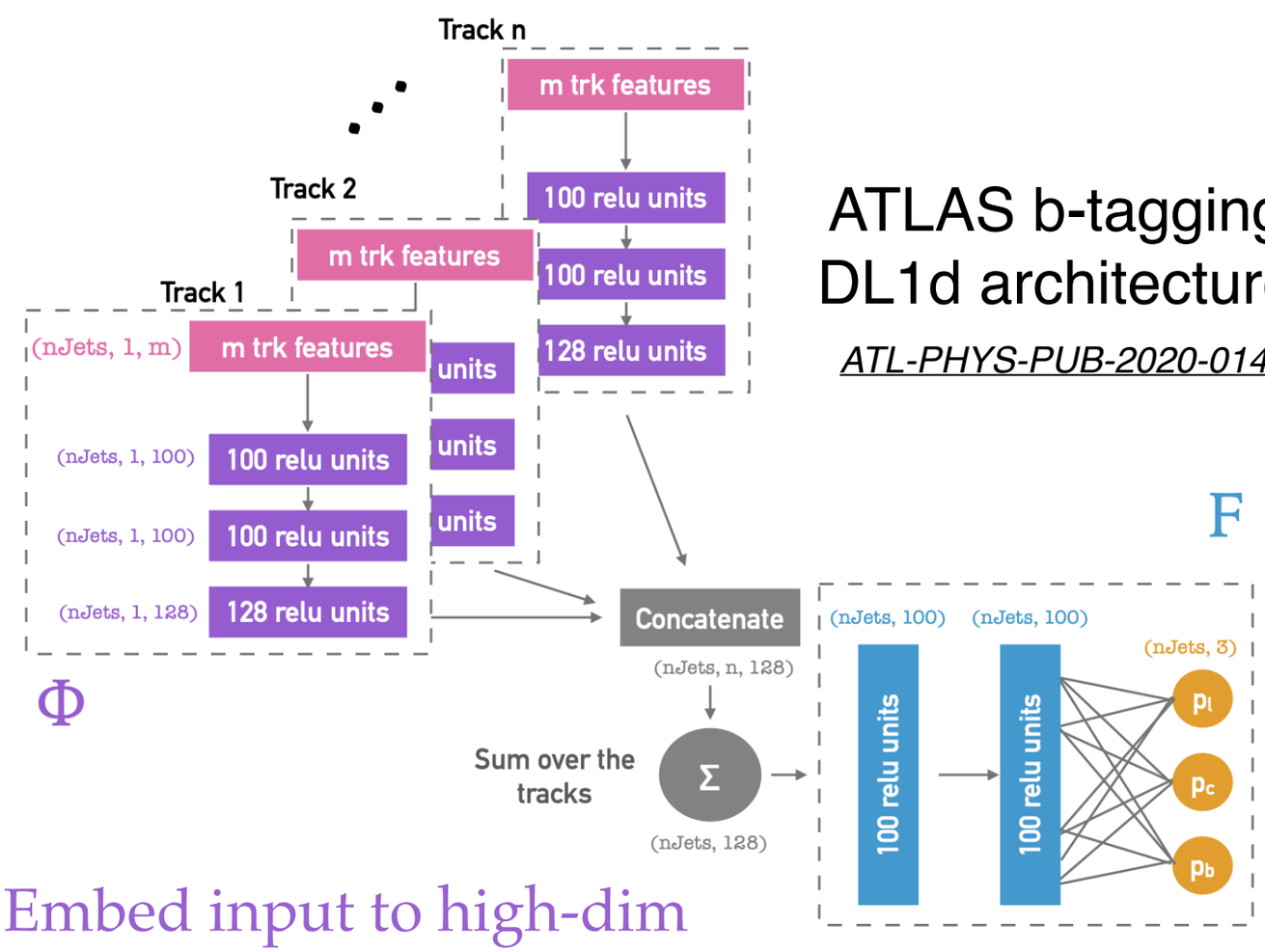
- CMS **DeepJet** algorithm used CNN, RNN and fully connected DNN at the same time



1.4 Input as sets

- Sequence (and also image) implies certain ordering
  - Lack of permutation invariance $f(x_1, x_2) \neq f(x_2, x_1)$
- Deepset [Manzil et al]
  - *for any permutation* π : $f(\{x_1, \ldots, x_M\}) = f(\{x_{\pi(1)}, \ldots, x_{\pi(M)}\})$
  - e.g. f = max, mean, etc





ATLAS b-tagging DL1d architecture

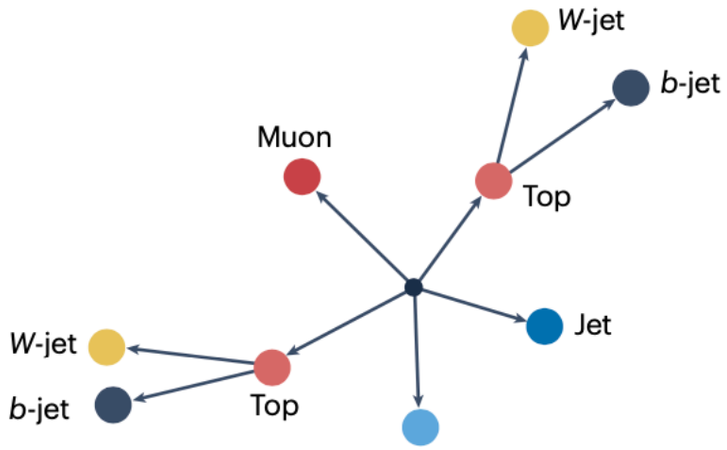
ATL-PHYS-PUB-2020-014

Φ: Embed input to high-dim space to preserve properties

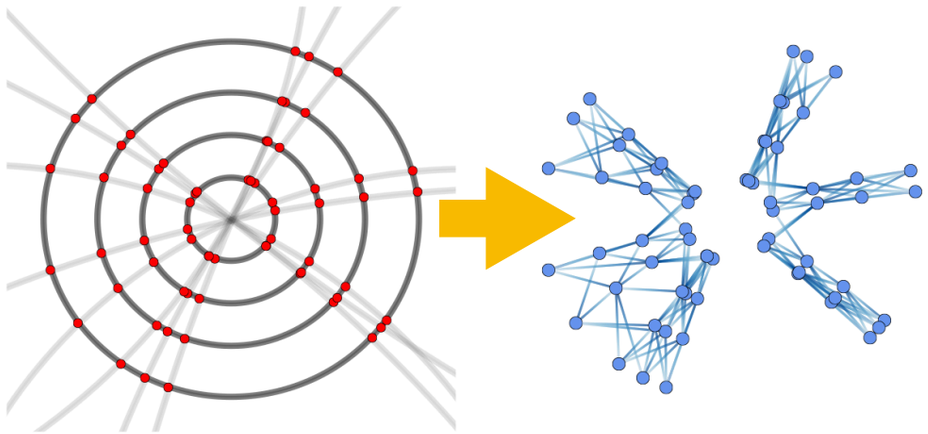
1.5 Input as graphs (including point cloud)

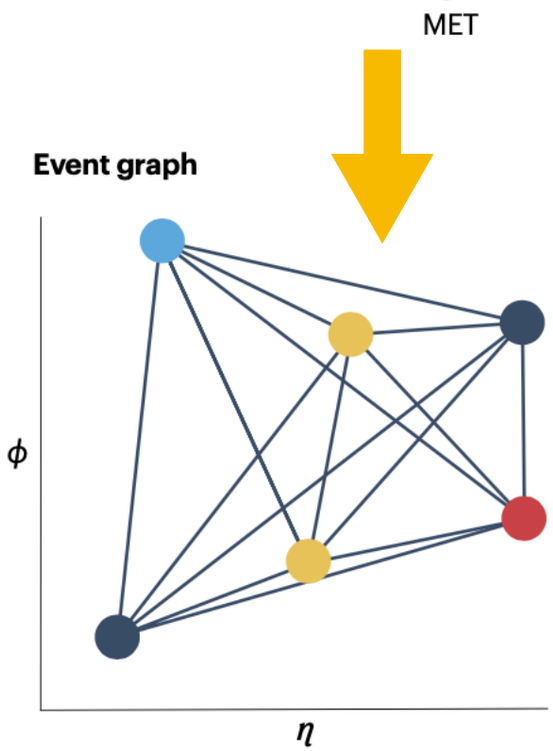
- Graph is also a natural way to represent LHC data

A $t\bar{t}$ event

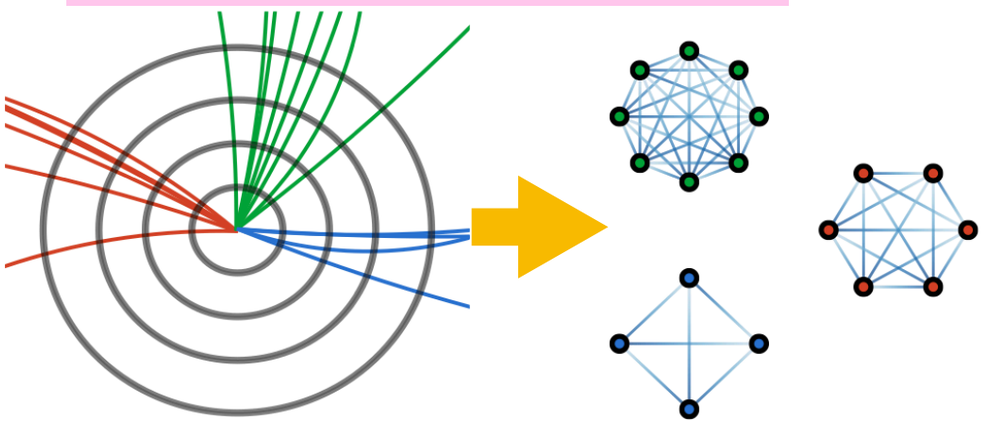


Hits in tracker





Jet is a graph of particles



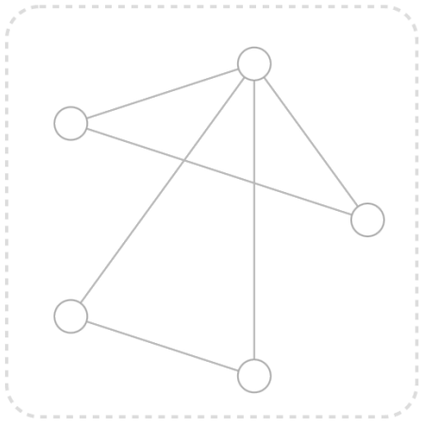
[Graph neural networks in particle physics]

[Graph Neural Networks for Particle Tracking and Reconstruction]

[Graph neural networks at the Large Hadron Collider]

[Graph Neural Networks in Particle Physics: Implementations, Innovations, and Challenges]

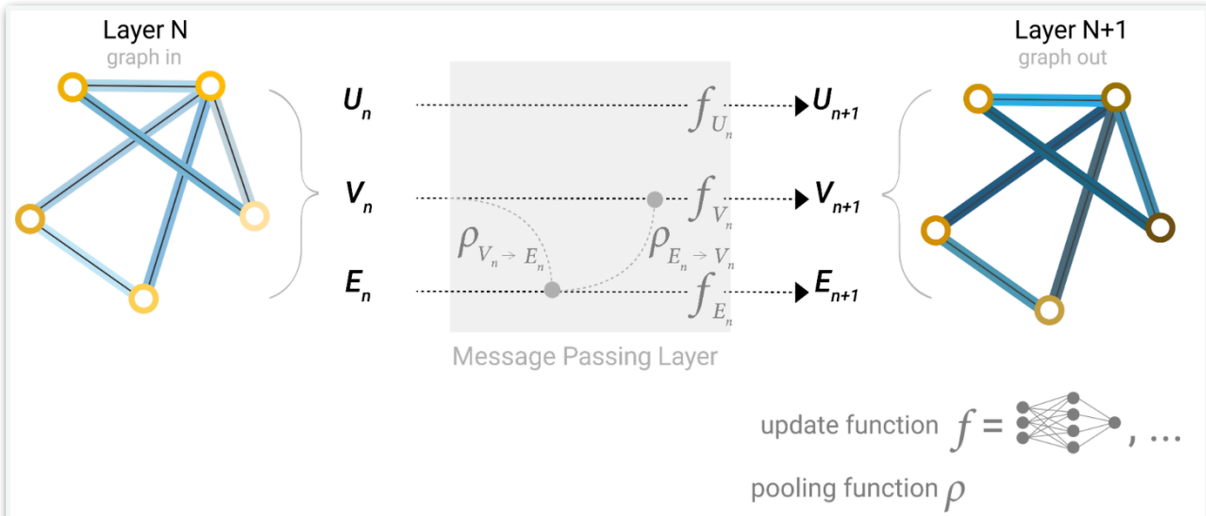
1.5 Graphs neural networks



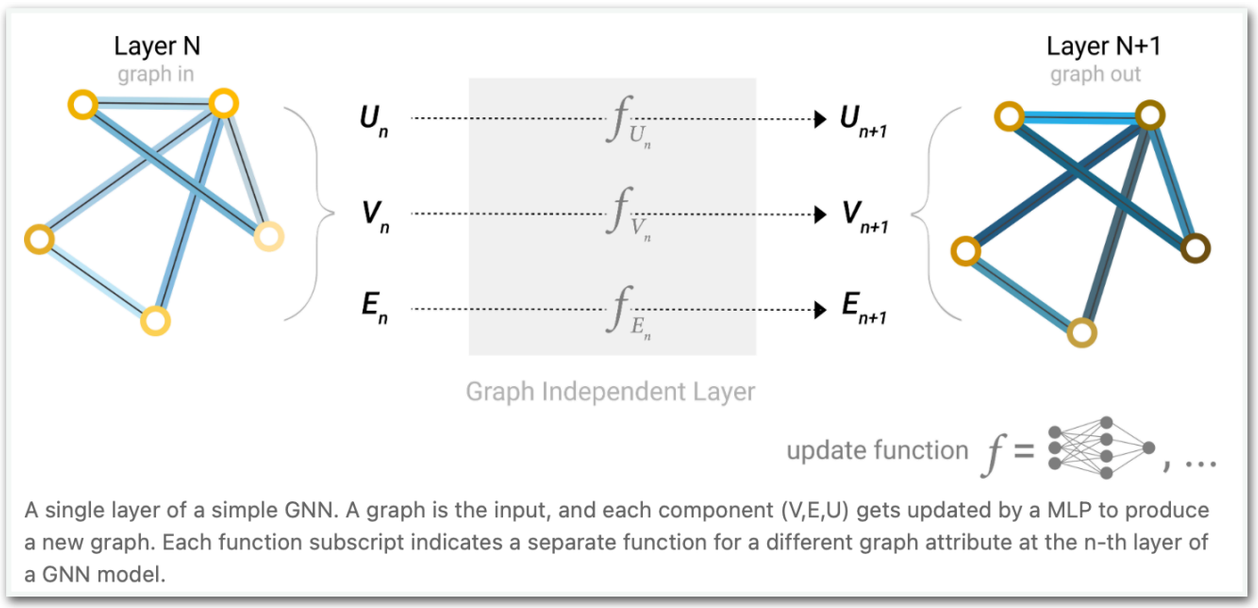
V Vertex (or node) attributes
e.g., node identity, number of neighbors

E Edge (or link) attributes and directions
e.g., edge identity, edge weight

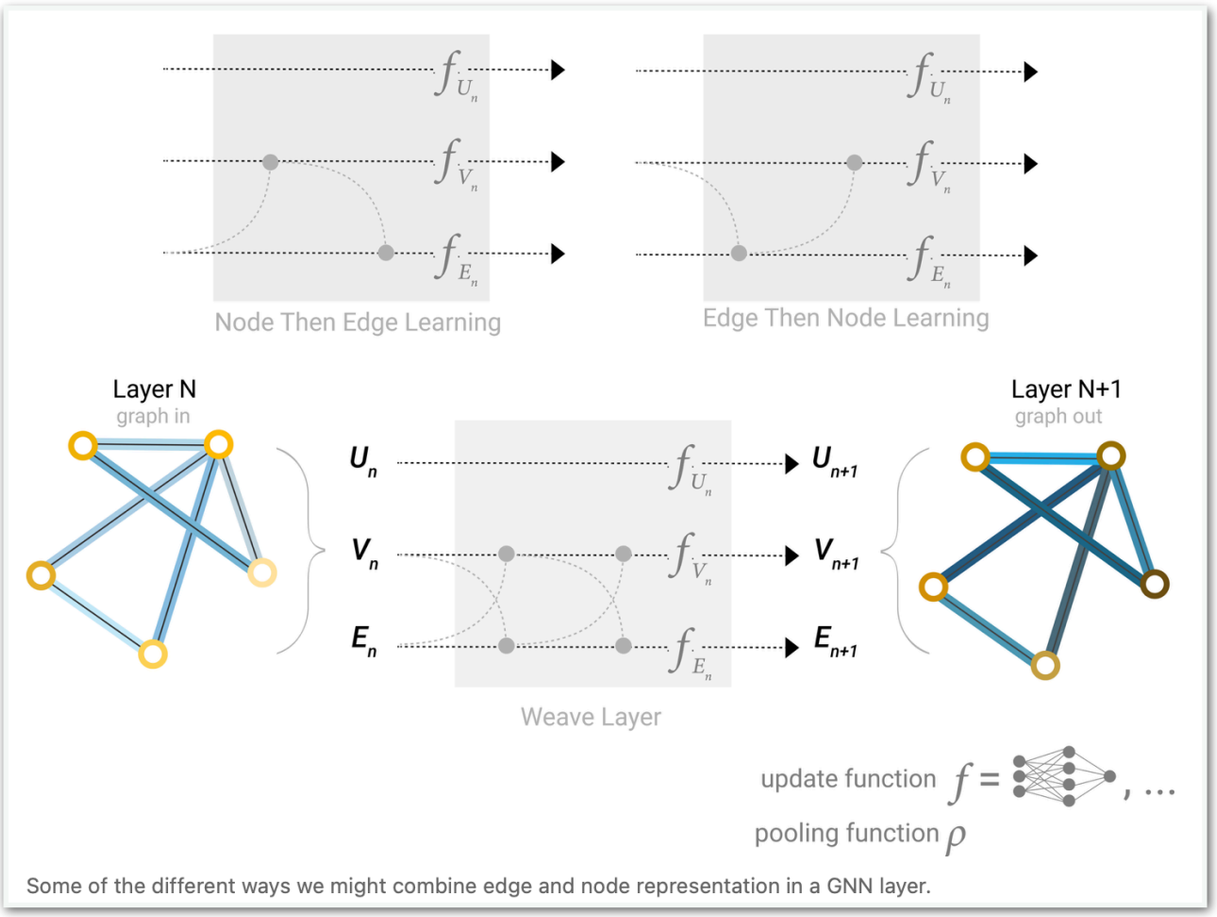
U Global (or master node) attributes
e.g., number of nodes, longest path



Architecture schematic for Message Passing layer. The first step "prepares" a message composed of information from an edge and it's connected nodes and then "passes" the message to the node.



A single layer of a simple GNN. A graph is the input, and each component (V,E,U) gets updated by a MLP to produce a new graph. Each function subscript indicates a separate function for a different graph attribute at the n-th layer of a GNN model.

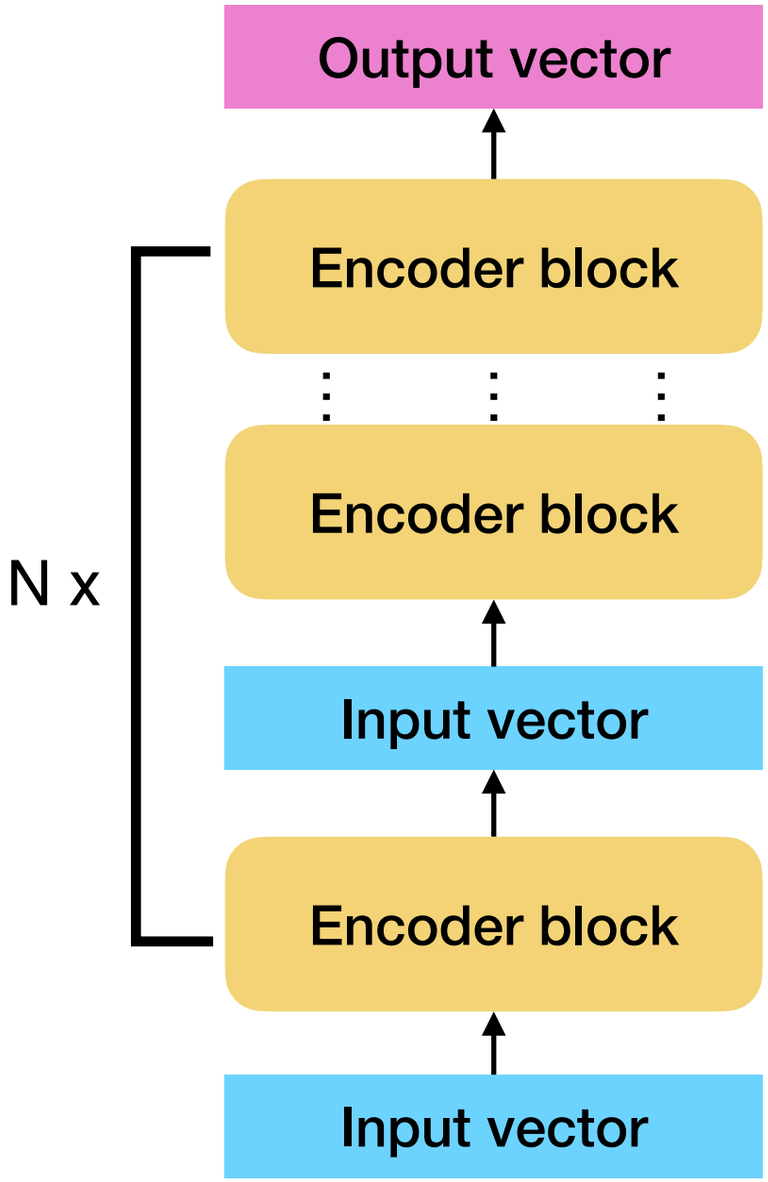


Some of the different ways we might combine edge and node representation in a GNN layer.

[A Gentle Introduction to Graph Neural Networks]

1.5 Self-attention and transformer

- First proposed for natural language processing (NLP)
- Later find good performance in computer visual



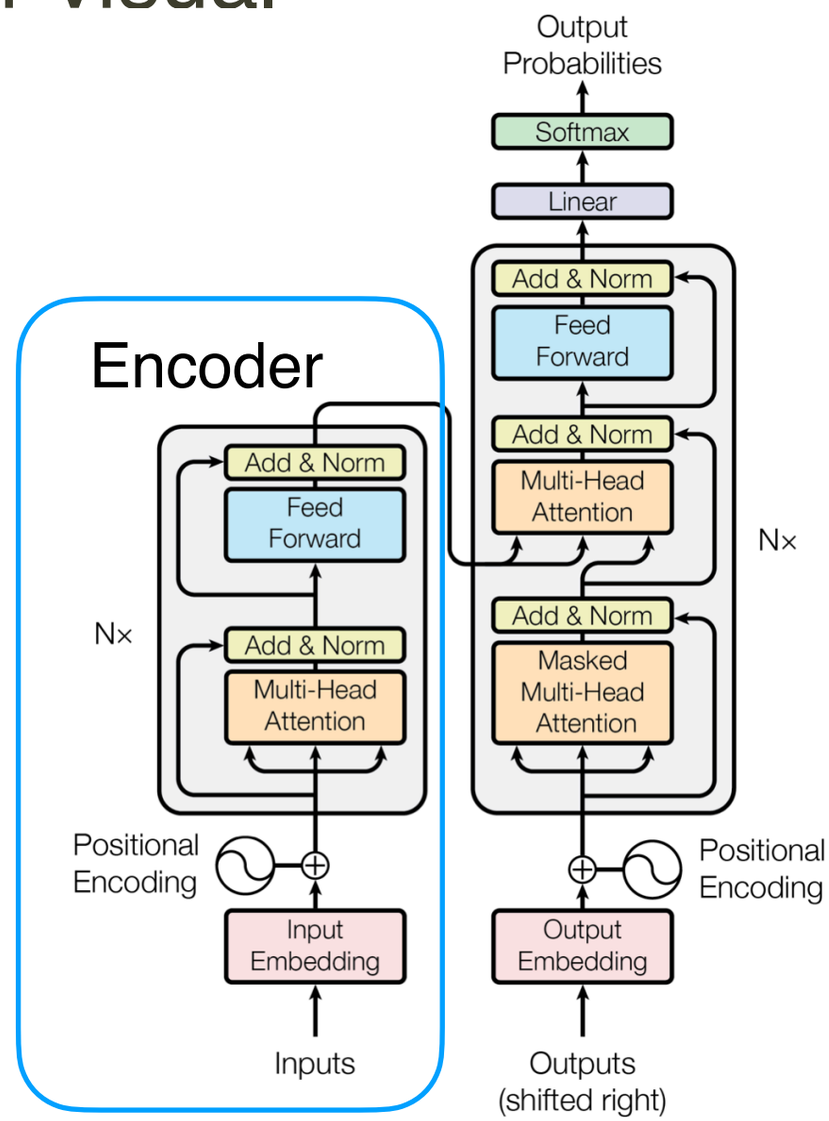
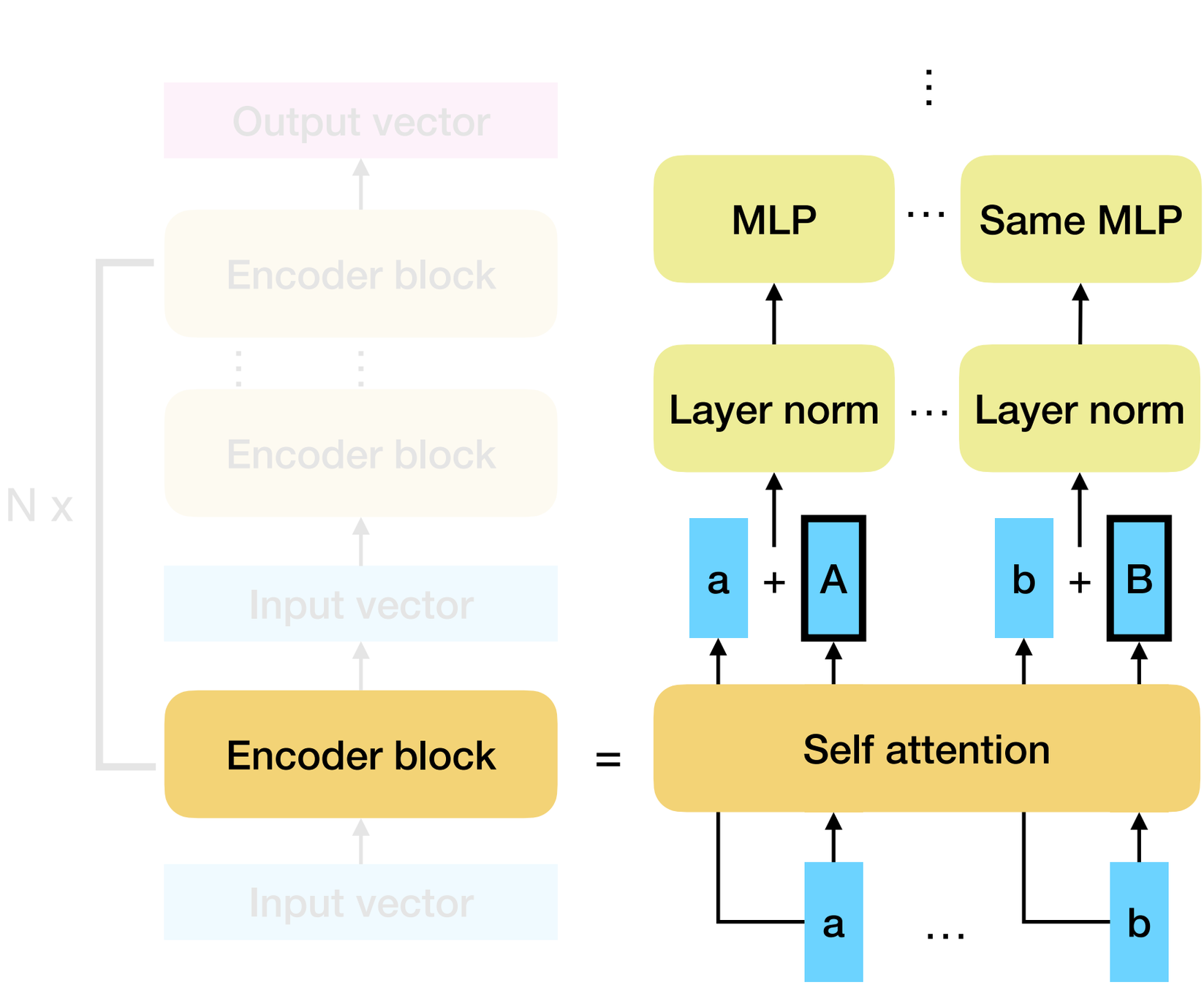


Figure 1: The Transformer - model architecture.

[Attention Is All You Need]

[An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale]

1.5 Self-attention and transformer



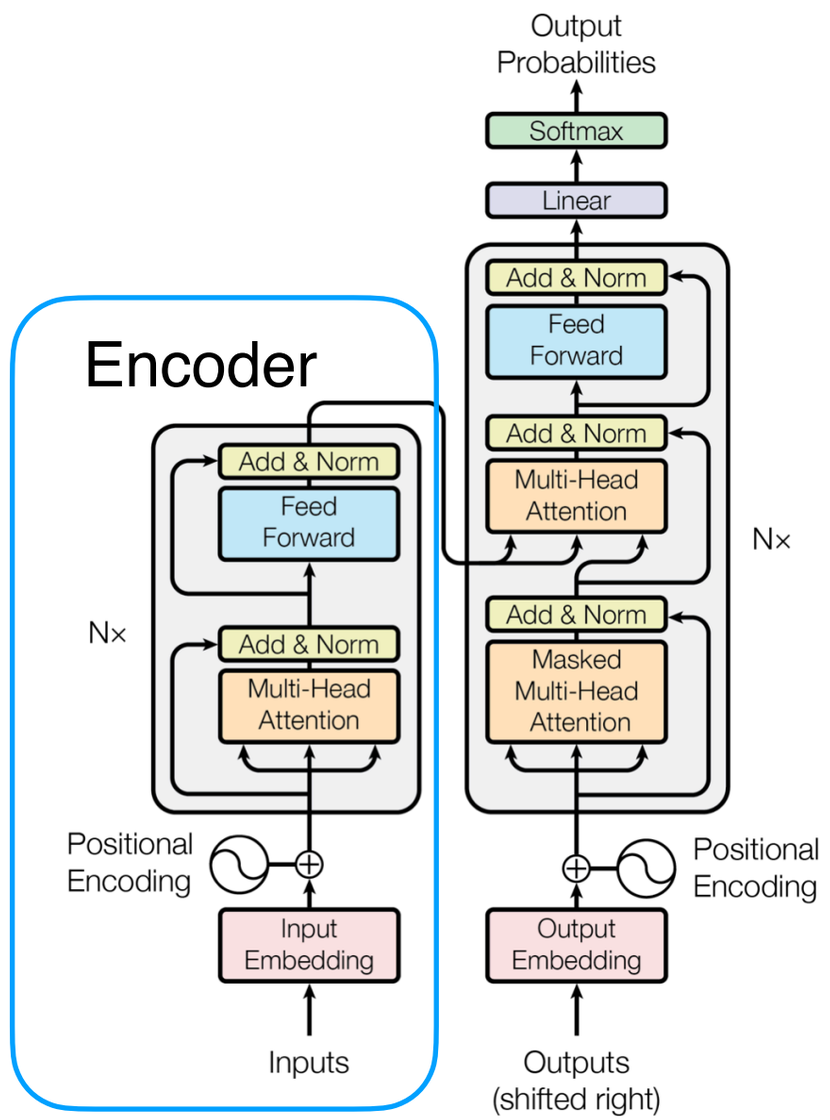
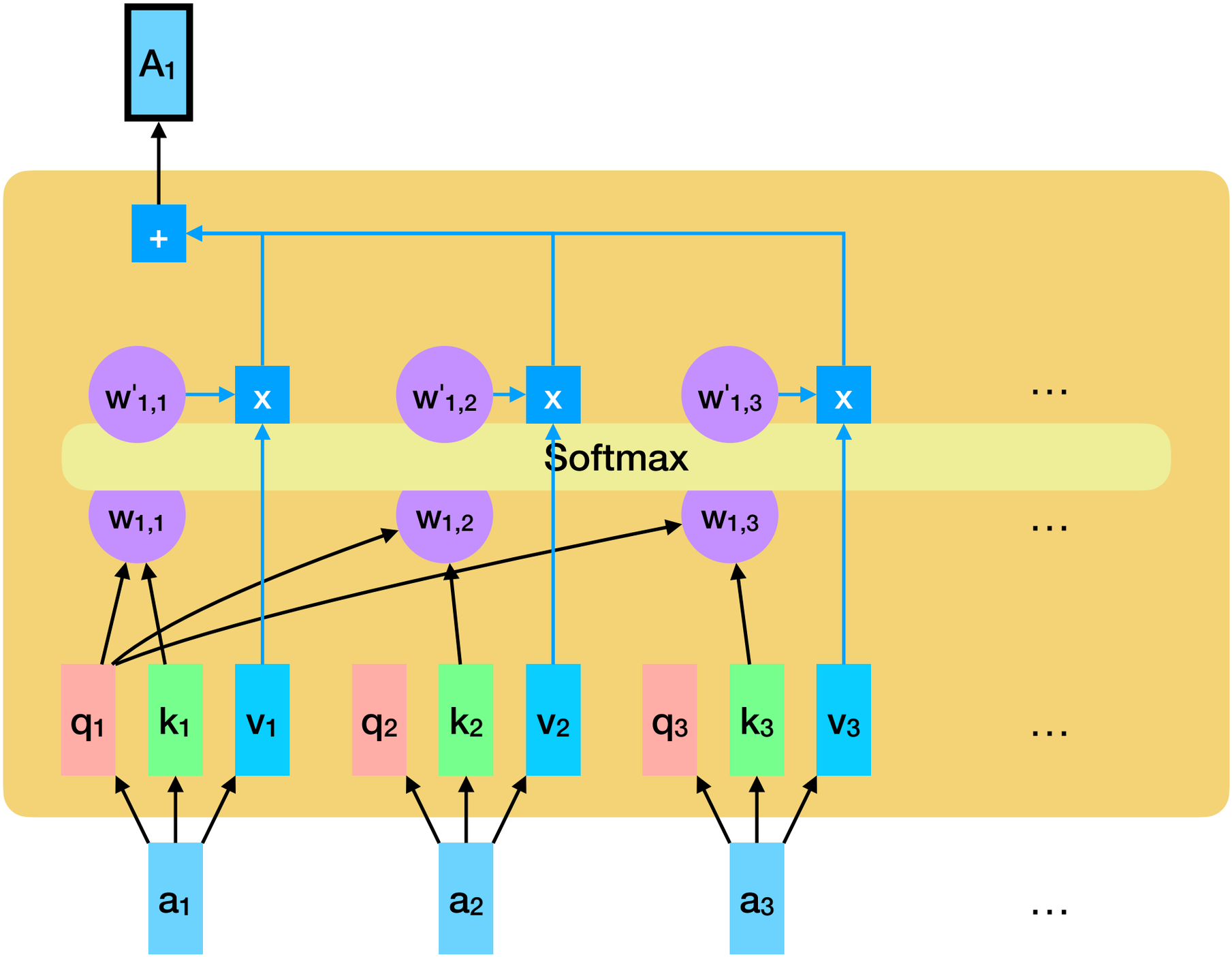


Figure 1: The Transformer - model architecture.

1.5 Self-attention



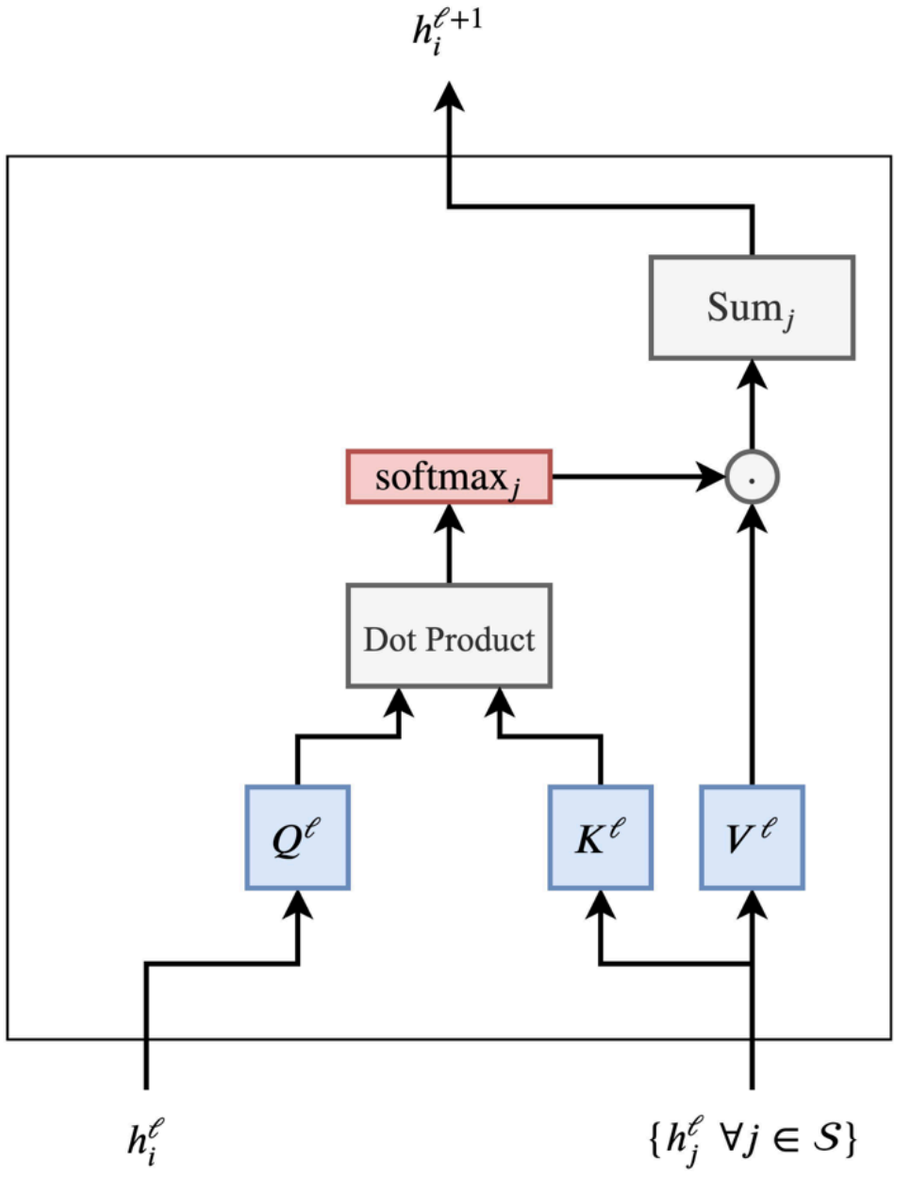
$q_1 = MLP_q(a_1)$ $q_2 = MLP_q(a_2)$

$k_1 = MLP_k(a_1)$ $k_2 = MLP_k(a_2)$

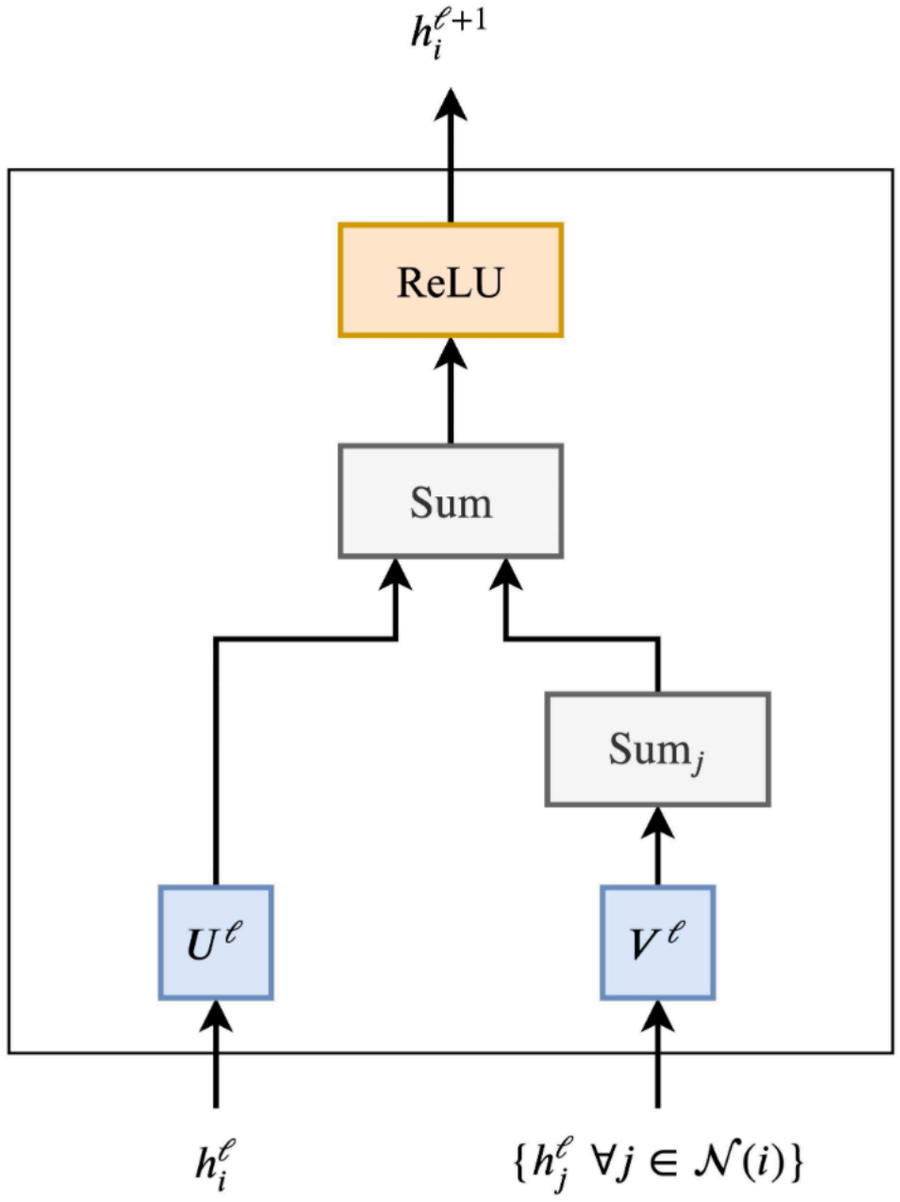
$v_1 = MLP_v(a_1)$ $v_2 = MLP_v(a_2)$

1.5 Transformer vs GNN

Transformer

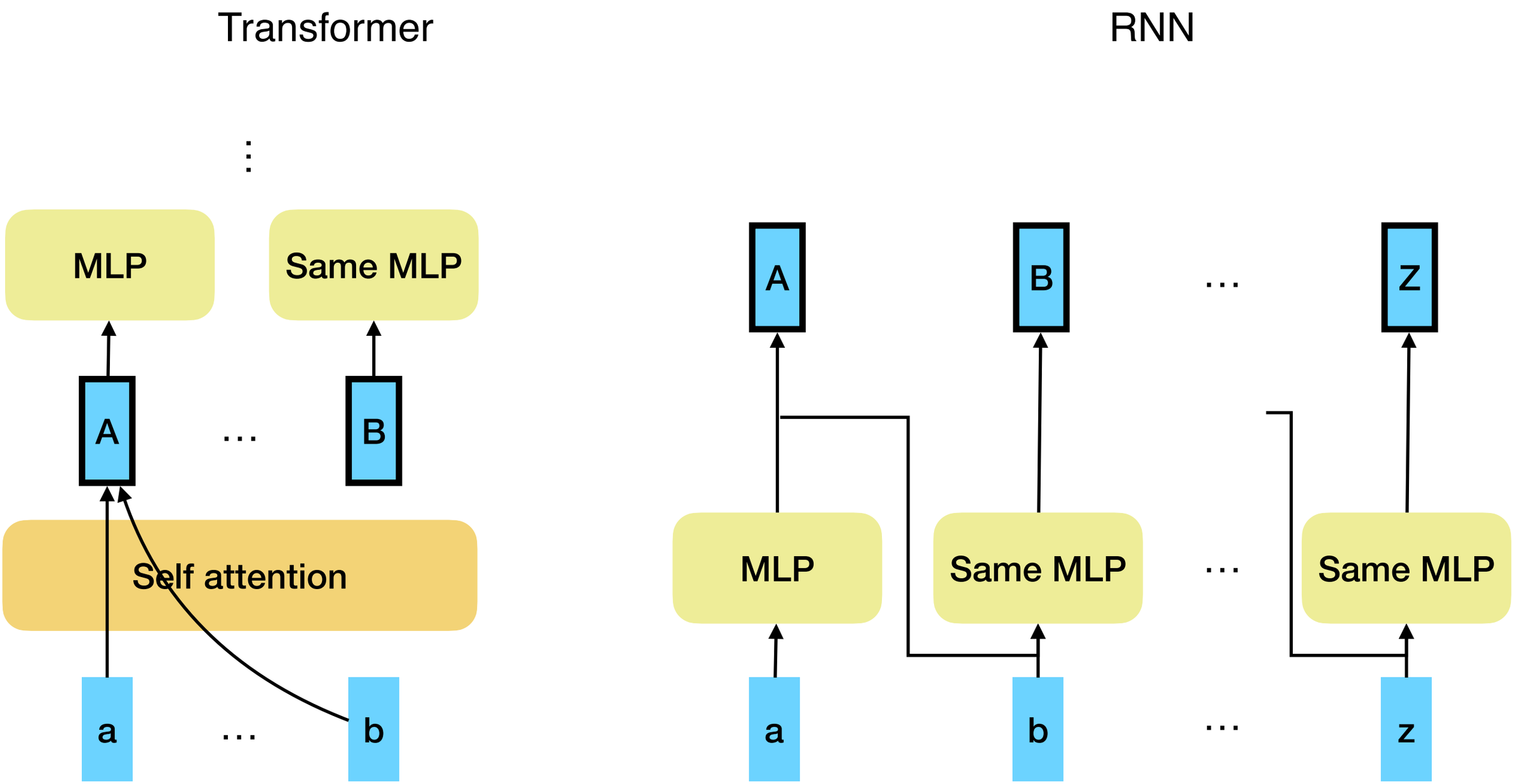


GNN



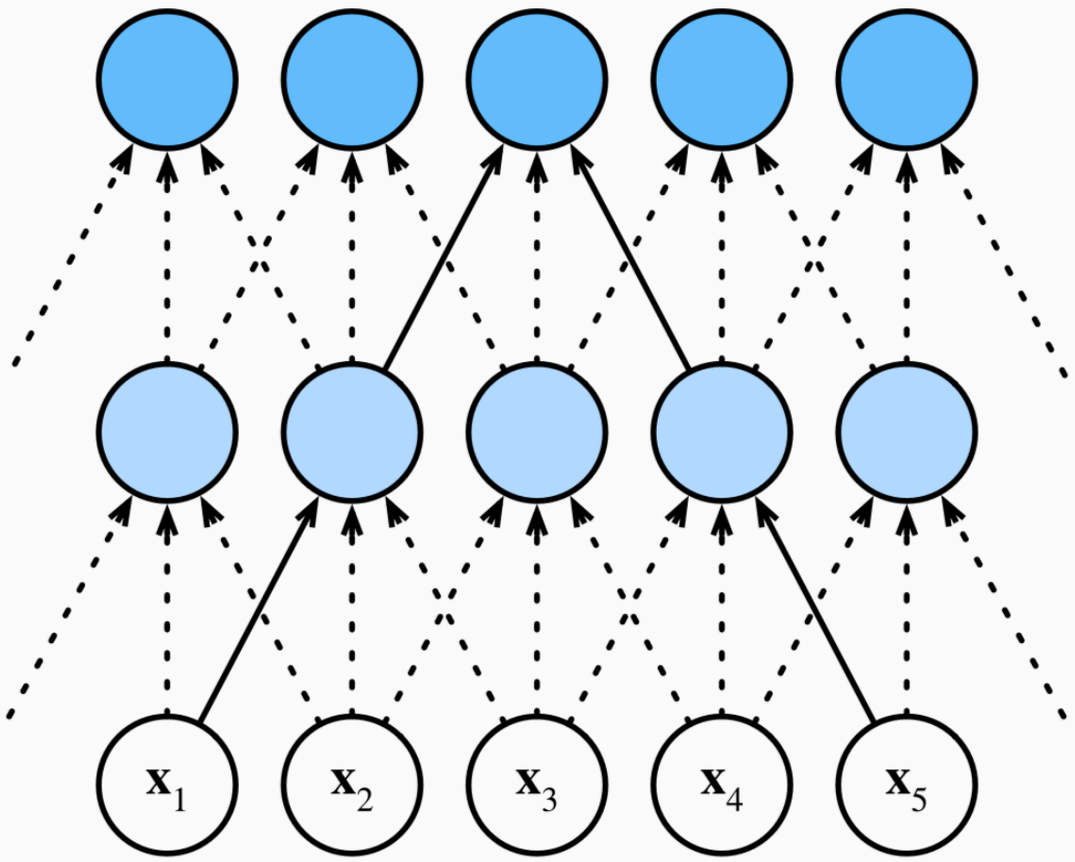
Transformers are Graph Neural Networks

1.5 Transformer vs RNN

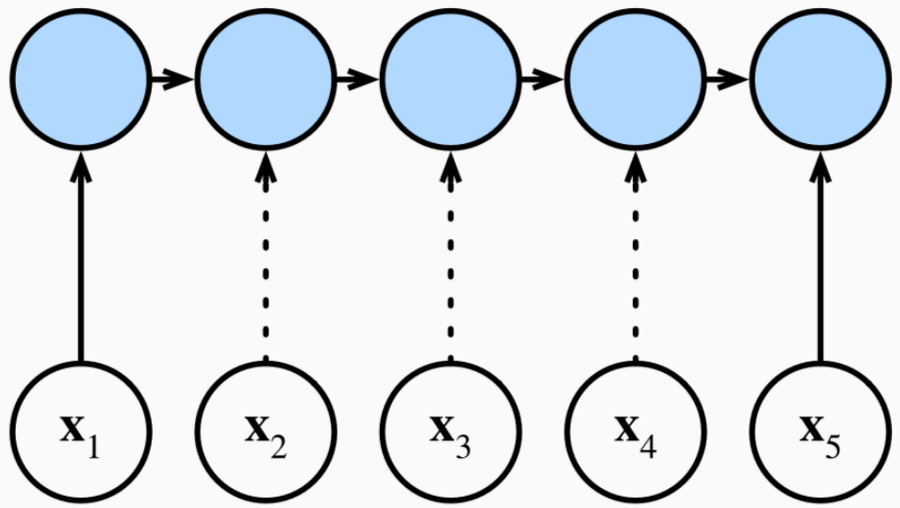


1.5 Transformer vs CNN

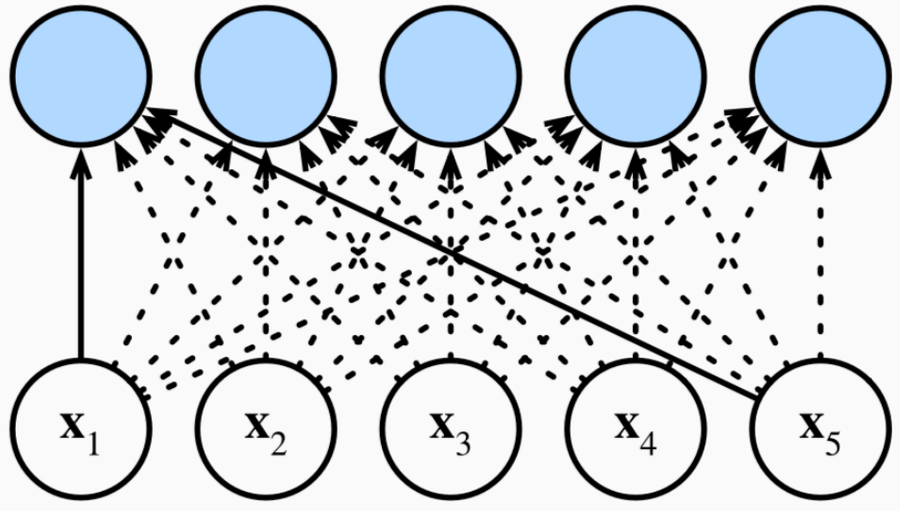
CNN



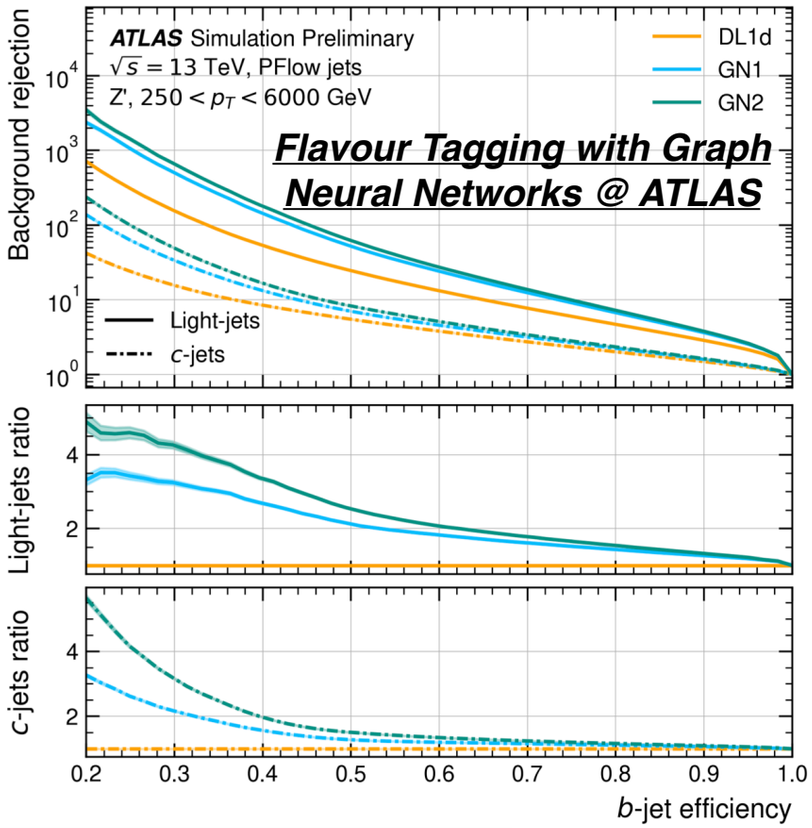
RNN



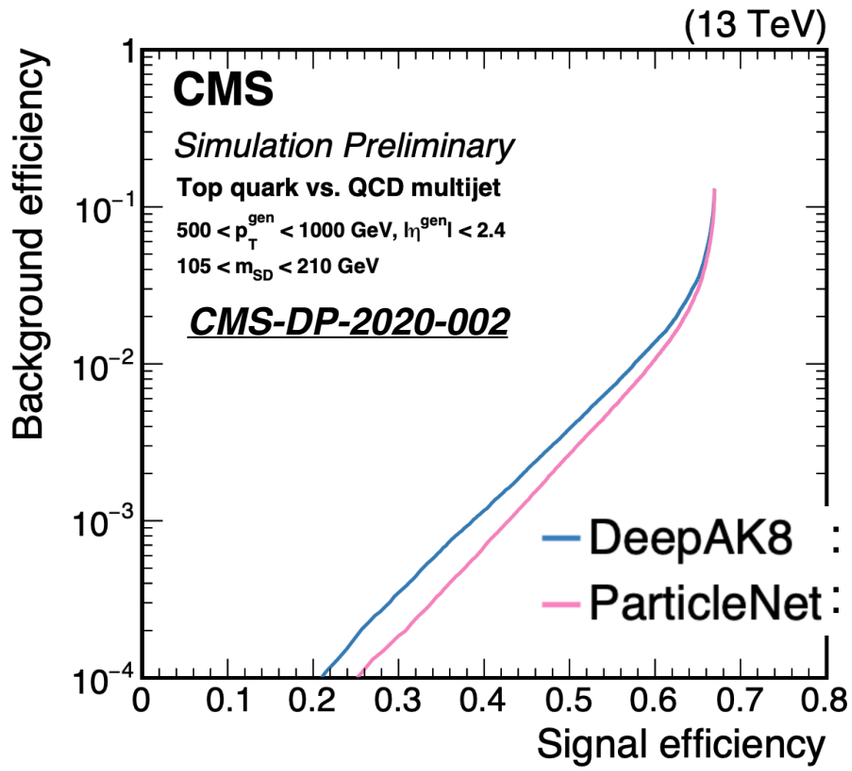
Self-attention



1.5 GNN applications

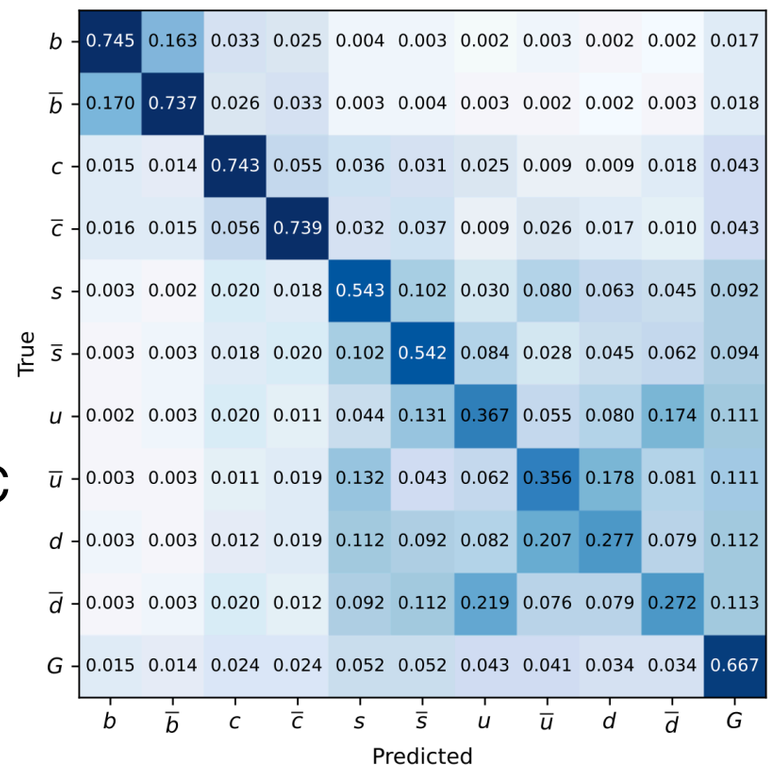


DL1d is DeepSet based b-tagger
GN1 is GNN based b-tagger
GN2 is optimised GN1 + attention mechanics [1706.03762]



: CNN based large R jet tagger

: GNN based large R jet tagger



Jet origin identification using ParticleNet at CEPC
Excellent performance in confusion matrix

arXiv:2310.03440

see Manqi's talk on 13th Saturday for more details

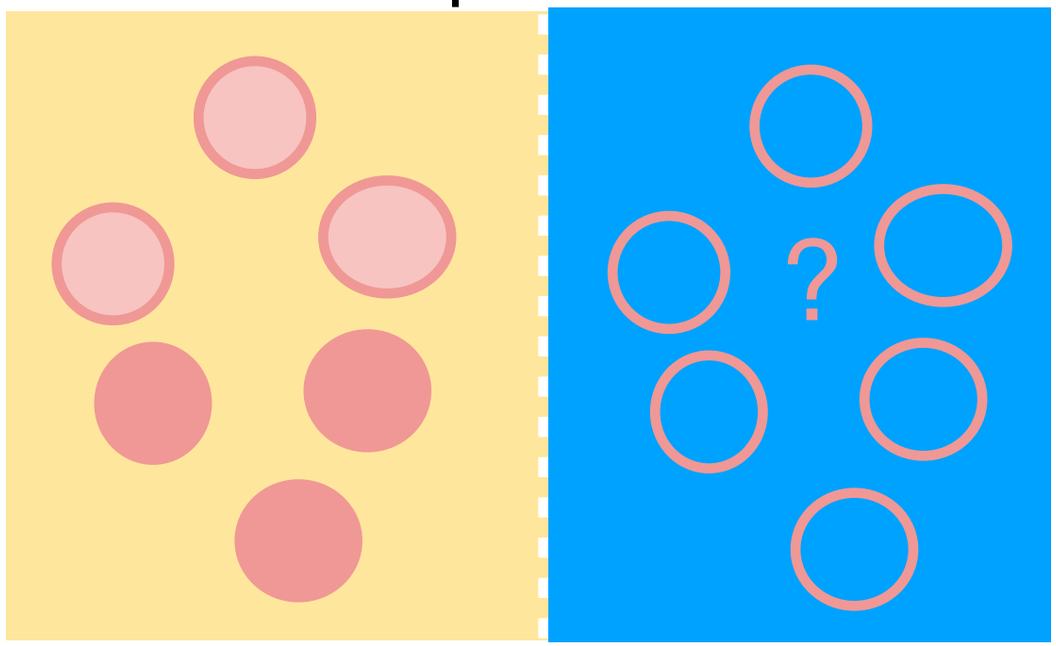
Step 2: set up the learning task

- A common feature of previous examples are **supervised** machine learning

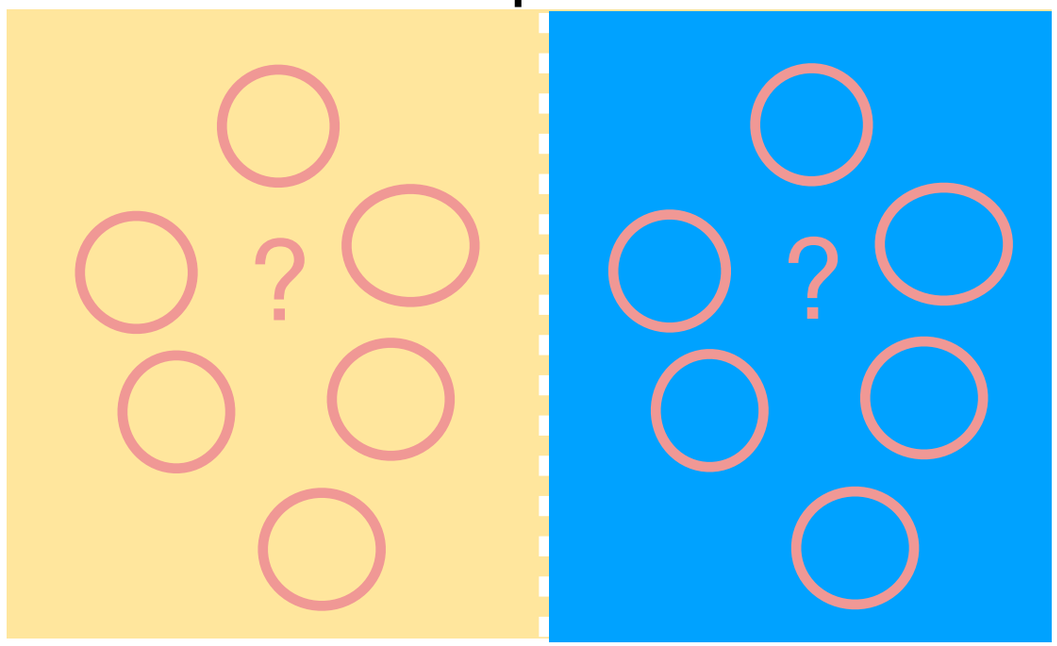
- Trained on data with known signal, known background → known labels

Step 2: set up the learning task

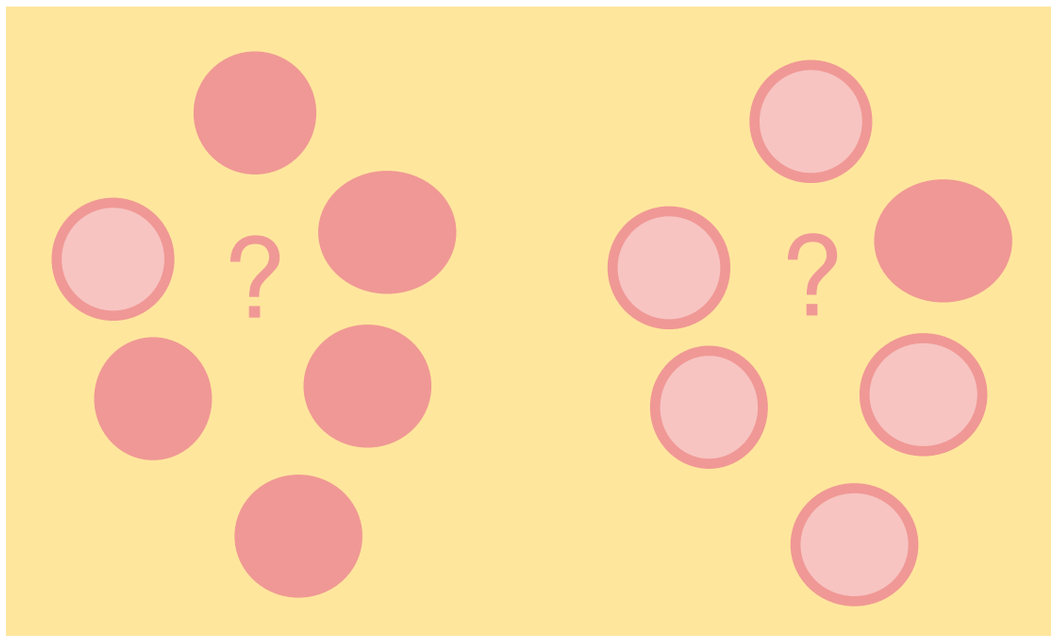
Supervised



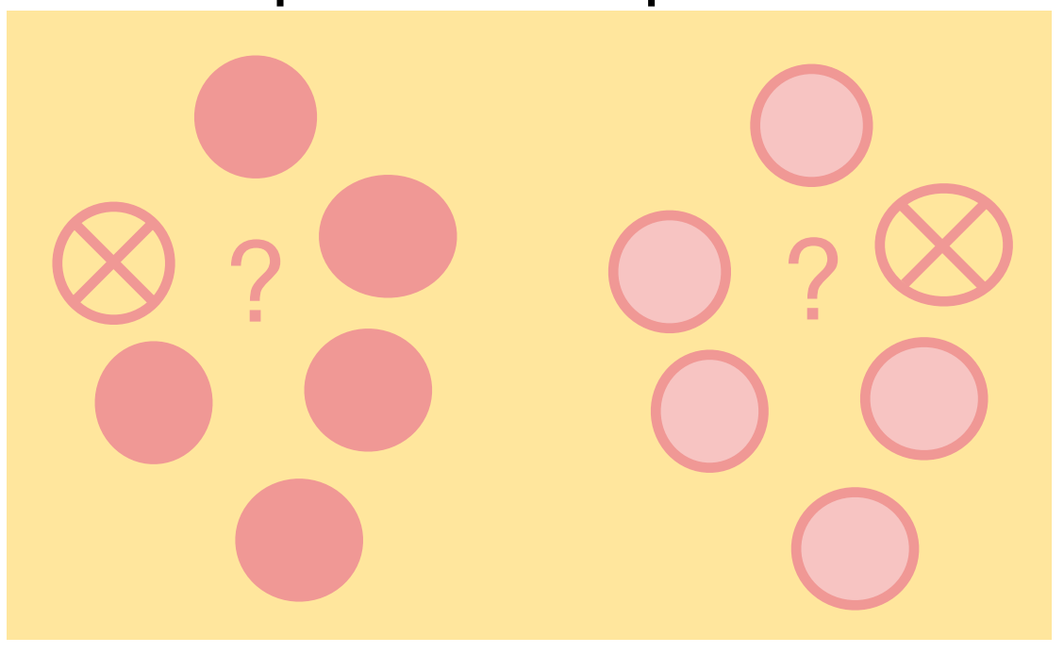
Unsupervised



Weakly-supervised = noisy labels



Semi-supervised = partial labels



Unsupervised—fast simulation (than Geant4)

Event Generation → Detector simulation → Digitisation → Reconstruction → Analysis

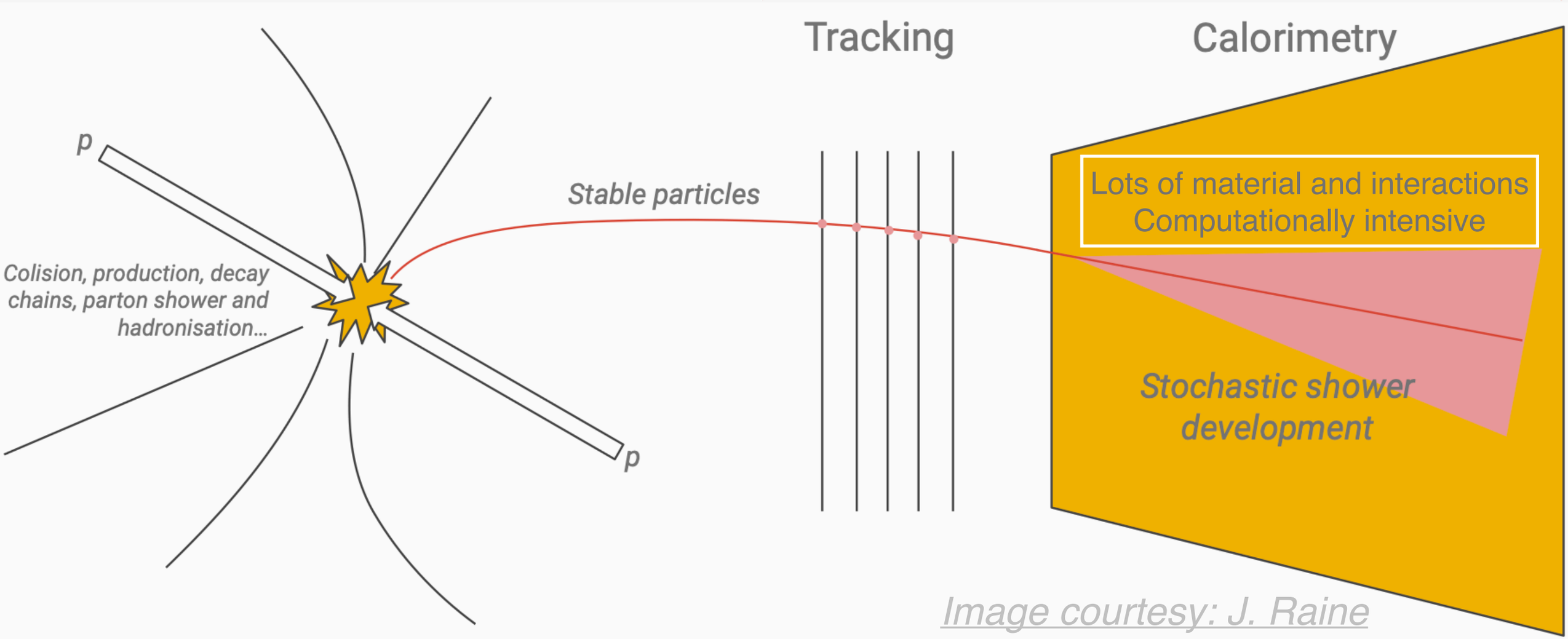
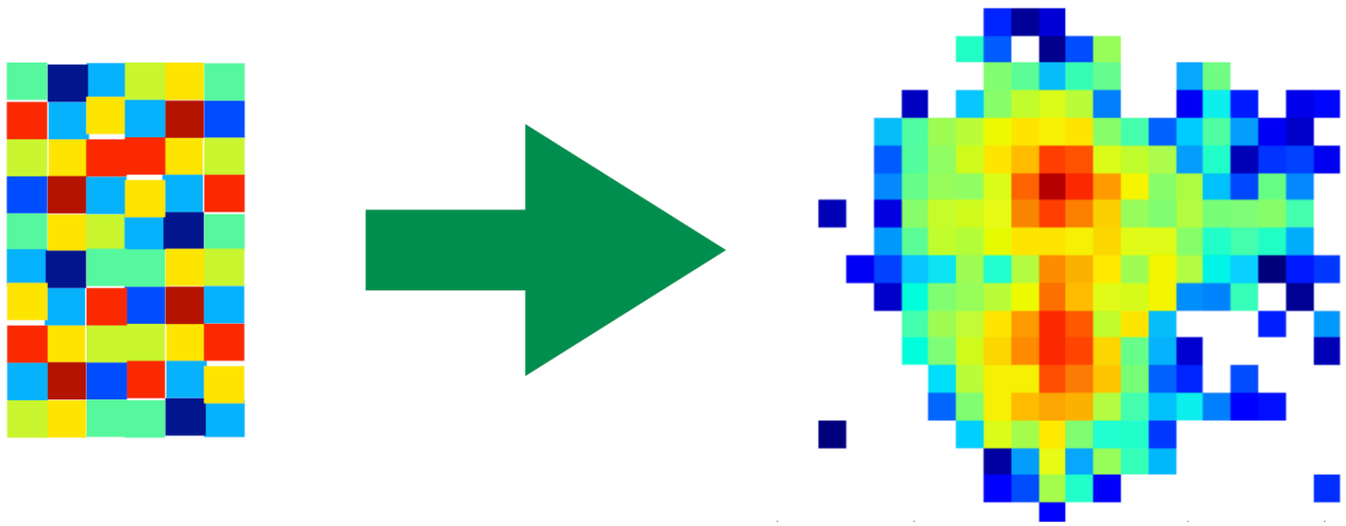


Image courtesy: J. Raine

Fast simulation: Generative models

A generator is a function that maps random numbers to structure.



Generative models are typically unsupervised.

GAN
Generative Adversarial Networks

PRD 97, 014021 (2018)
2309.06515
2207.04340
…

VAE
Variational Autoencoders

2211.15380
2203.00520
2210.07430
…

NF
Normalizing Flows

JINST 2023 18 P10017
2308.11700
PRD 107.113003
2305.11934
…

Diffusion
Diffusion model

PRD 108 (2023) 072014
2309.05704
2308.03847
…

Example—Integrated into real experiment

ATLAS fast simulation includes a GAN at intermediate energies for hadrons

	Inner Detector	Calorimeters	Muon Spectrometer
Electrons Photons	Geant4	FastCaloSimv2	
Hadrons	Geant4	Geant4 pions: $E_{kin} < 200$ MeV Other hadrons: $E_{kin} < 400$ MeV	FastCalo Sim V2 $E_{kin} < (8-16)$ GeV	**FastCalo GAN** $(8-16)$ GeV $< E_{kin} <$ $(256-512)$ GeV	FastCalo Sim V2 $E_{kin} > (256-512)$ GeV	Muon Punchthrough +Geant4
Muons	Geant4	Geant4	Geant4

COMPUT SOFTW BIG SCI 6, 7 (2022)

Example—Integrated into real experiment

FastCaloGAN has been expanded from Run 2 to Run 3

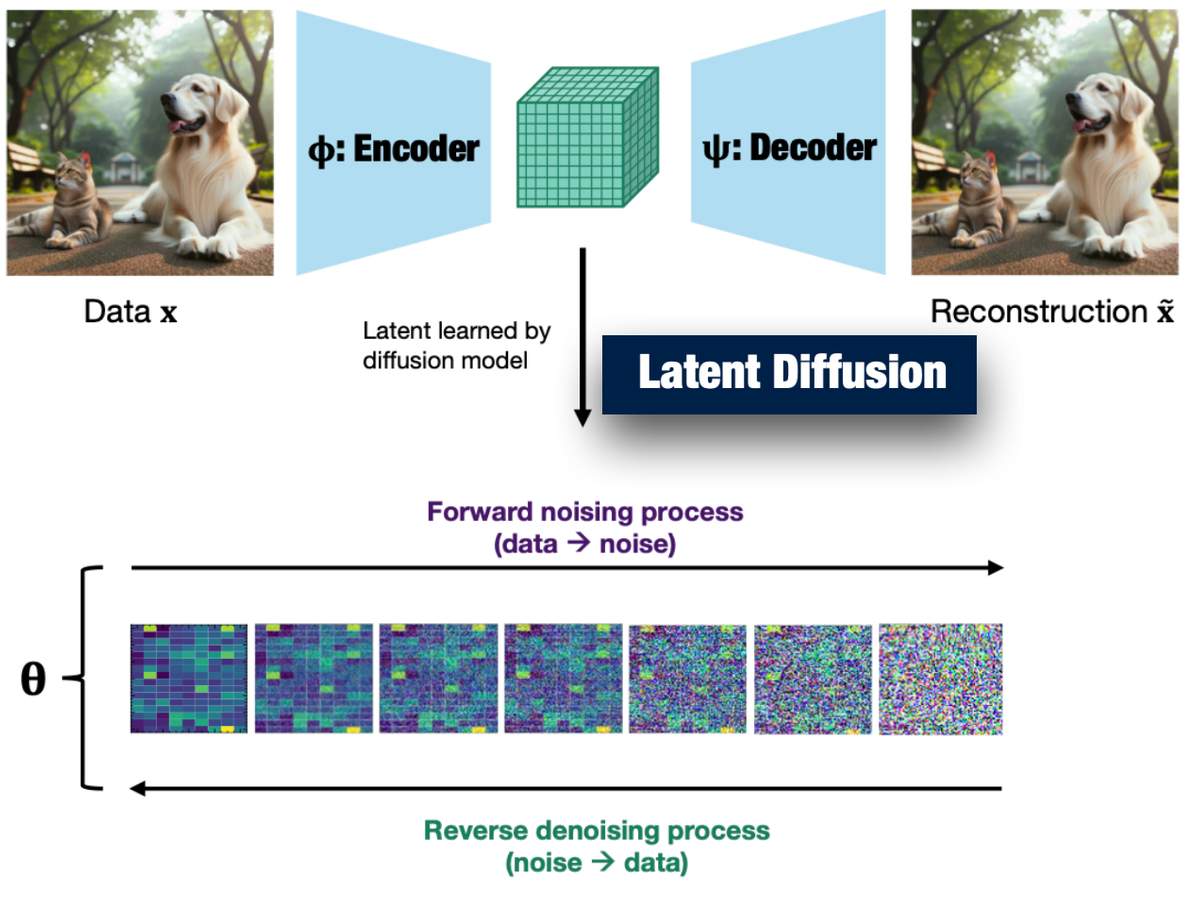
	Inner Detector	Calorimeters	Muon Spectrometer
Electrons Photons		FastCaloGAN V2: E_{kin} < 8 GeV && |η| < 2.4, Except [0.9<|η|<1.1, 1.35<|η|<1.5] FastCaloSim V2: E_{kin} > 16 GeV && |η| < 2.4, All E_{kin} && [0.9<|η|<1.1, 1.35<|η|<1.5, |η|>2.4]	
Charged Pions Kaons	Geant4	Geant4: Pions: E_{kin} < 200 MeV; Other hadrons: E_{kin} < 400 MeV FastCaloSim V2: E_{kin} < 4 GeV && |η| < 1.4, E_{kin} < 1 GeV && |η| < 3.15 FastCaloGAN V2: E_{kin} > 8 GeV && |η| < 1.4, E_{kin} > 2 GeV && 1.4 < |η| < 3.15, All E_{kin} && |η| > 3.15	Muon Punchthrough + Geant4
Baryons		FastCaloGAN V2	
Muons		Geant4	

arXiv:2404.06335

Marrying generative techniques (in R&D)

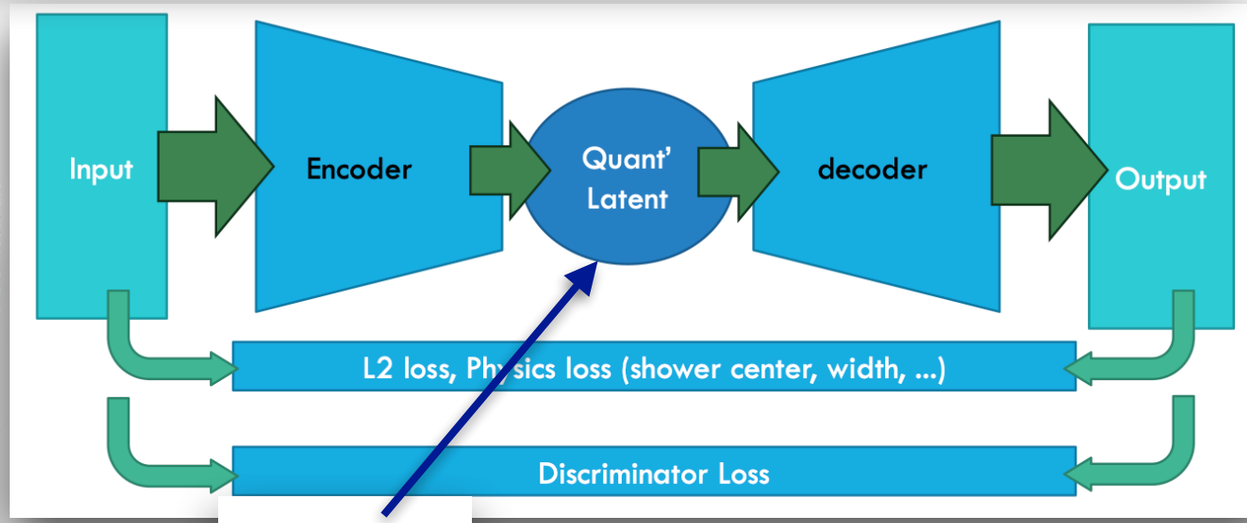
CaloLatent: Score-based Generative Modelling in the Latent Space for Calorimeter Shower Generation

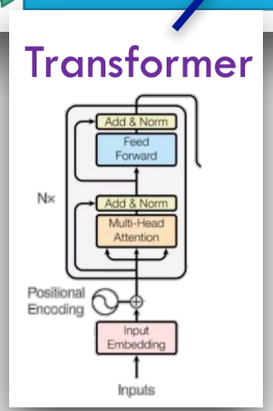
Thandikire Madula: UCL
Vinicius M. Mikuni: NERSC



Latent Generative Model for Calorimeter Fast Simulation

Qibin LIU, Chase Shimmin, Xiulong LIU, Eli Shlizerman, Shih-Chieh HSU, Shu LI





Transformer for sampling in latent space

"modern" concepts as a plugin put into traditional architectures

Fast Calorimeter Simulation Challenge 2022

View on GitHub

Welcome to the home of the first-ever Fast Calorimeter Simulation Challenge!

The purpose of this challenge is to spur the development and benchmarking of fast and high-fidelity calorimeter shower generation using deep learning methods. Currently, generating calorimeter showers of interacting particles (electrons, photons, pions, ...) using GEANT4 is a major computational bottleneck at the LHC, and it is forecast to overwhelm the computing budget of the LHC experiments in the near future. Therefore there is an urgent need to develop GEANT4 emulators that are both fast (computationally lightweight) and accurate. The LHC collaborations have been developing fast simulation methods for some time, and the hope of this challenge is to directly compare new deep learning approaches on common benchmarks. It is expected that participants will make use of cutting-edge techniques in generative modeling with deep learning, e.g. GANs, VAEs and normalizing flows.

This challenge is modeled after two previous, highly successful data challenges in HEP – the top tagging community challenge and the LHC Olympics 2020 anomaly detection challenge.

Datasets

The challenge offers three datasets, ranging in difficulty from “easy” to “medium” to “hard”. The difficulty is set by the dimensionality of the calorimeter showers (the number layers and the number of voxels in each layer).

Link

Un-/weakly/semi supervised—anomaly detection

Why anomaly detection?

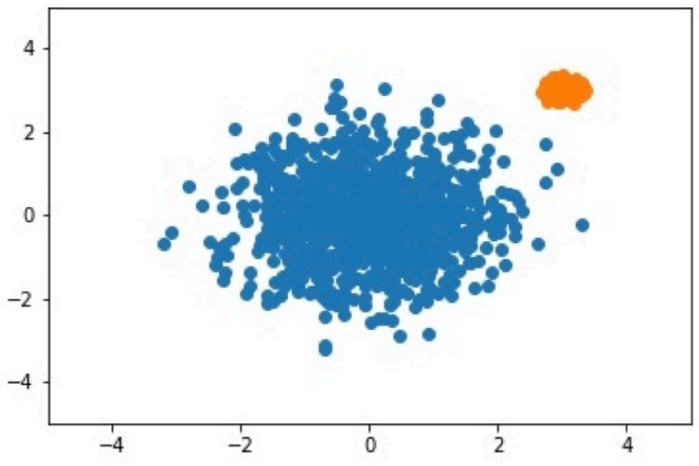
Typical Searches

- Looking for a specific, physics motivated signal
- Maximum sensitivity (using supervised learning e.g. BDT) for a specific model
- Not very useful for other signal models

Anomaly Detection

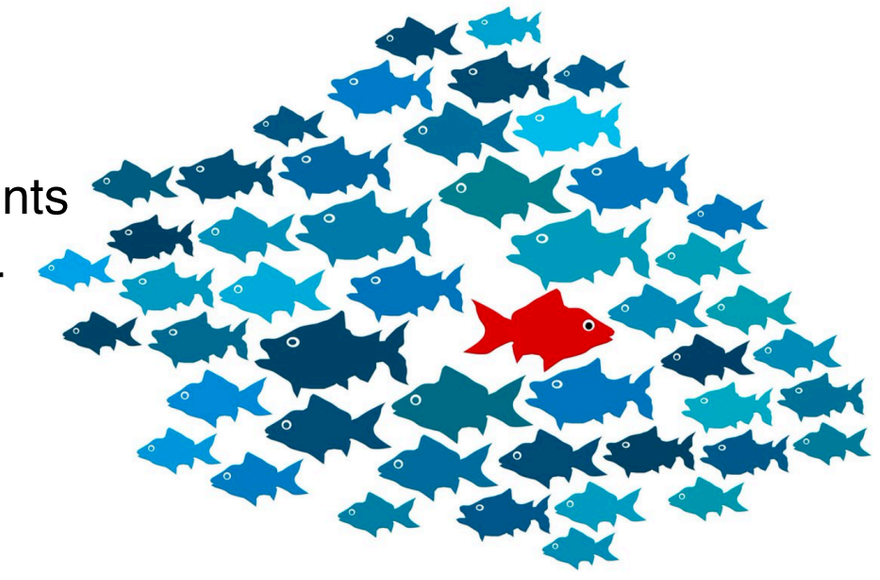
- Model agnostic/ independent search
- Looking for deviations from background only
- Less sensitive to any specific model, but can look for multiple different models

Two types of anomaly detection

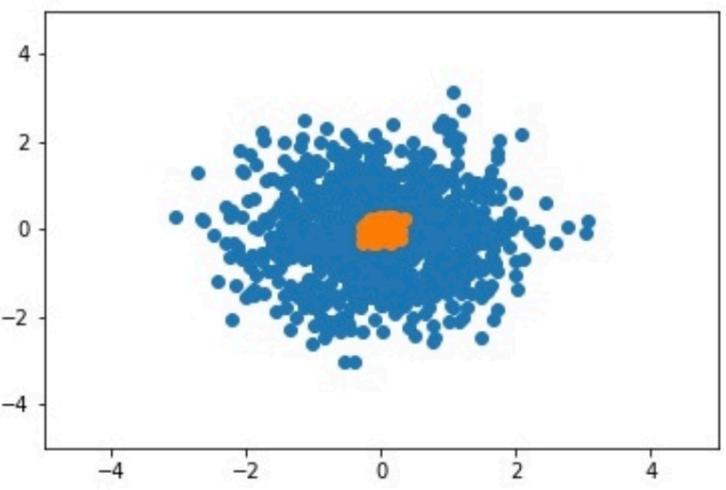


Outlier Detection

- Searching for unique or unexpected events
- In HEP, this is the tails of distributions or uncovered phase space

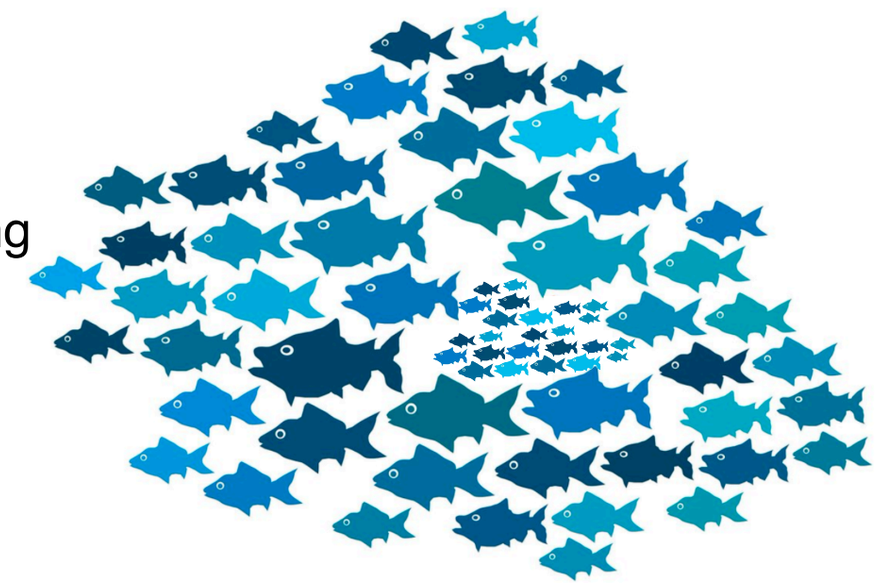


[1807.10261, 1808.08979, 1808.08992, 1811.10276, 1903.02032, 1912.10625, 2004.09360, 2006.05432, 2007.01850, 2007.15830, 2010.07940, 2102.08390, 2104.09051, 2105.07988, 2105.10427, 2105.09274, 2106.10164, 2108,03986, 2109.10919, 2110.06948, 2112.04958, 2203.01343,2206.14225, 2303.14134, 2304.03836, 2306.03637, 2308.02671, 2309.10157, 2309.13111, ...]



Overdensity detection

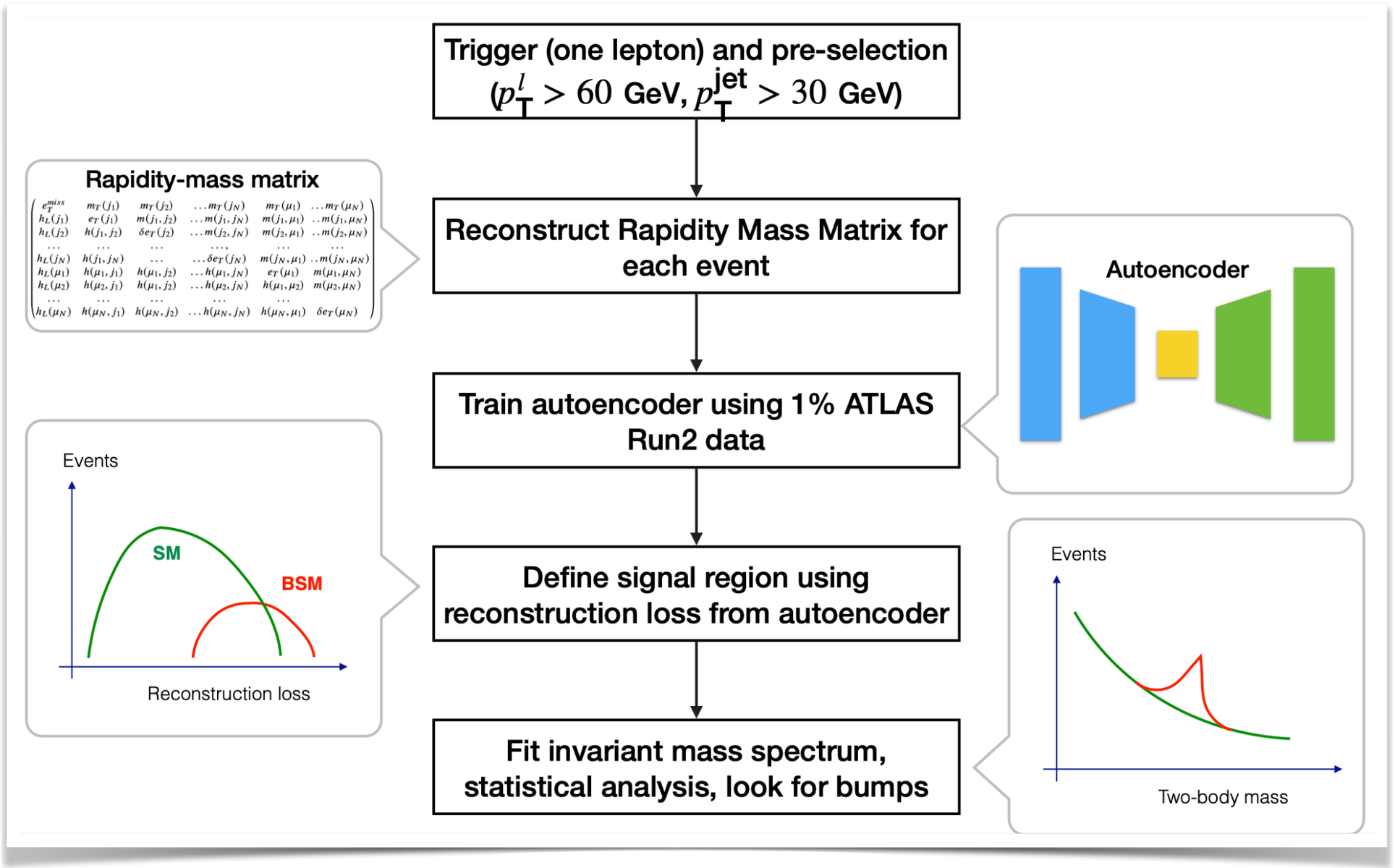
- Analogous to the traditional bump hunting



[1805.02664, 1806.02350, 1902.02634, 1912.12155, 2001.05001, 2001.04990, 2012.11638, 2106.10164, 2109.00546, 2202.00686, 2203.09470, 2208.05484, 2210.14924, 2212.11285, 2305.04646, 2305.15179, 2306.03933, 2307.11157, 2309.12918, 2310.06897, 2310.13057, ...]

Inspired by this presentation

Outlier Detection in experiments (ATLAS)

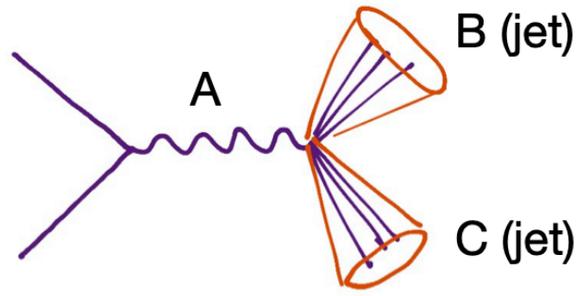


- Full event level anomaly detection
- Searched in 9 invariant masses including di-jet, di-b-jet, with three anomaly regions => demonstrating high efficiency in the search

arXiv: 2307.01612

Overdensity Detection in experiments (ATLAS)

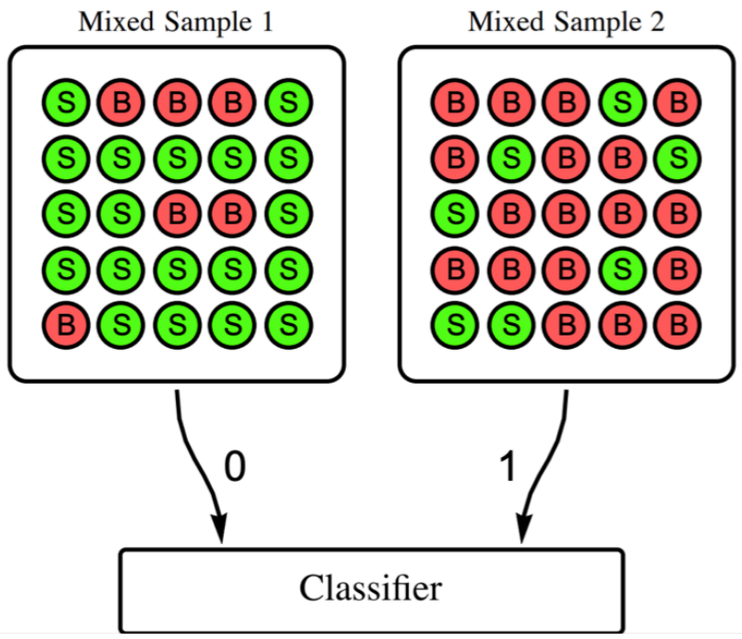
- Di-jet (large-R jets) resonance search
  - ➡ pp → A → BC → JJ
  - ➡ Training classifiers on **data**, with **no labels**

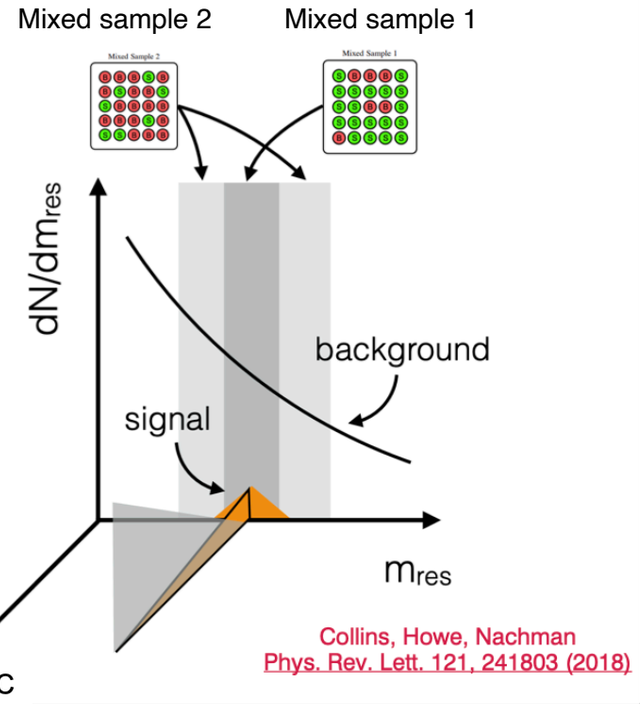


CWoLa

Classification Without Labels

Metodiev, Nachman, Thaler
JHEP 10 (2017) 174





Collins, Howe, Nachman
Phys. Rev. Lett. 121, 241803 (2018)

- Performed on di-fatjet resonant search
- Network is learning difference between Prob(b) and Prob(s+b)

Phys. Rev. Lett. 125 (2020) 131801

Summary

- ML and HEP: an enduring partnership
  - ML has been a longstanding companion in HEP in various stages of the data analysis pipeline
- ML as a Toolset for HEP
  - ML serves as a valuable assistant, maximising the exploration of costly collision data
  - Choosing ML architectures based on the data structures to optimise efficiency
  - Evolution towards unsupervised and semi-supervised learning on more generative tasks
- Future directions
  - Expanding training data to refine ML models
  - Delving into lower level features to uncover hidden patterns
  - Incorporating physics knowledge for a deeper contextual understanding

ML continues to unlock breakthroughs within the realm of HEP.

Backup

References

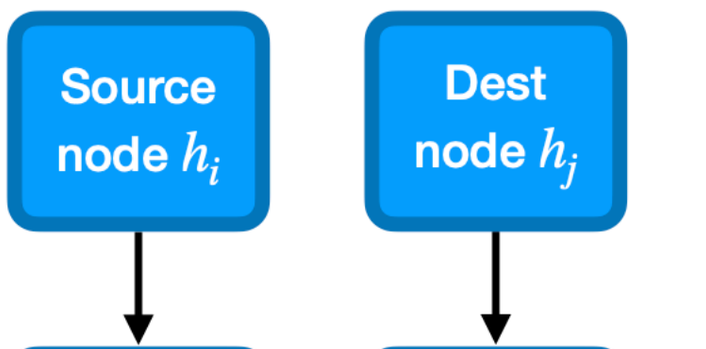
- A Living Review of Machine Learning for Particle Physics
  - https://iml-wg.github.io/HEPML-LivingReview/
  - Updated summary on arXiv available submission in machine learning in HEP
- Neural Networks, Types, and Functional Programming
  - http://colah.github.io/posts/2015-09-NN-Types-FP/
  - Deep learning introduction in 10 min
- (New) Machine learning chapter in the particle data group book:
  - https://pdg.lbl.gov/2023/reviews/rpp2022-rev-machine-learning.pdf

GN1 vs GN2

Updated Attention Mechanism

GN2 follows more closely the *transformer* architecture [1706.03762]

[2105.14491]

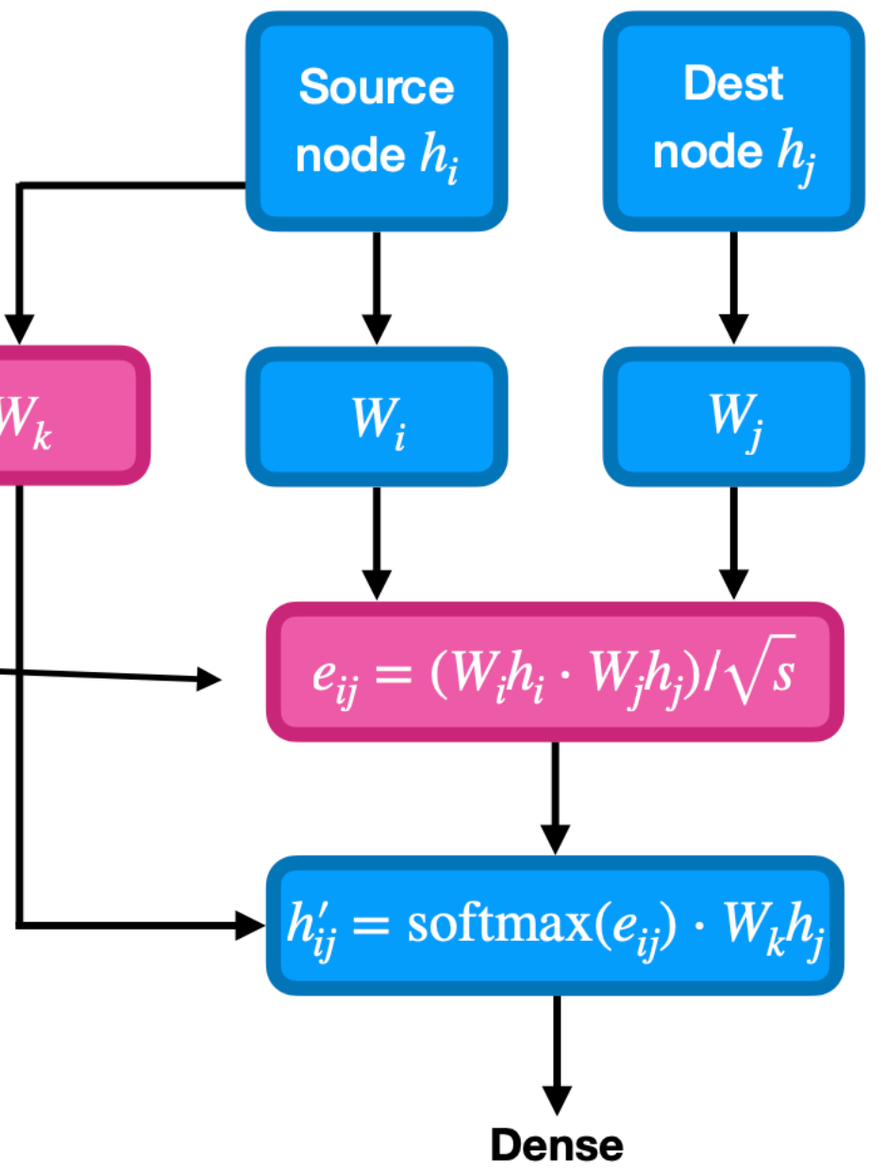


Linear projections

GN1 → GN2

Attention scores

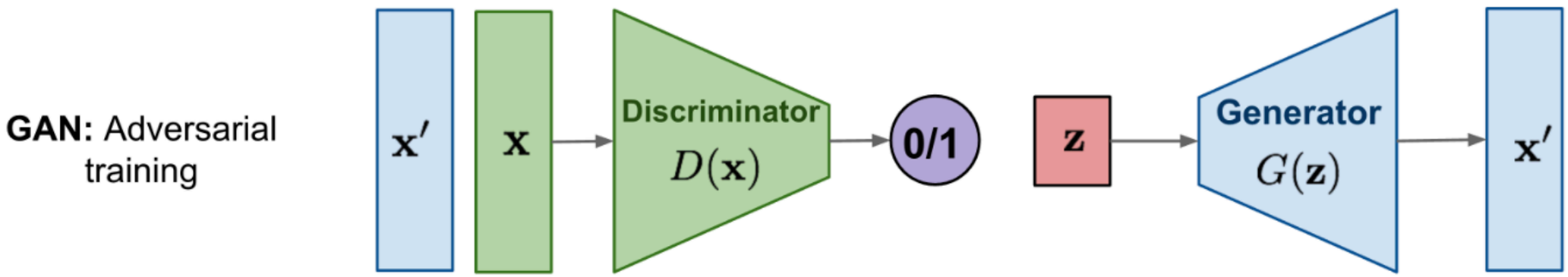
[1706.03762]



- Deeper & wider
- More attention heads
- ReLU → SiLU activation.

Generative model: GAN

A summary blog



Generative Adversarial Networks (GANs): A pair of networks where one produce realistic data and the other classifies it as fake or real.

🙂 High-quality output

☹️ Training instability

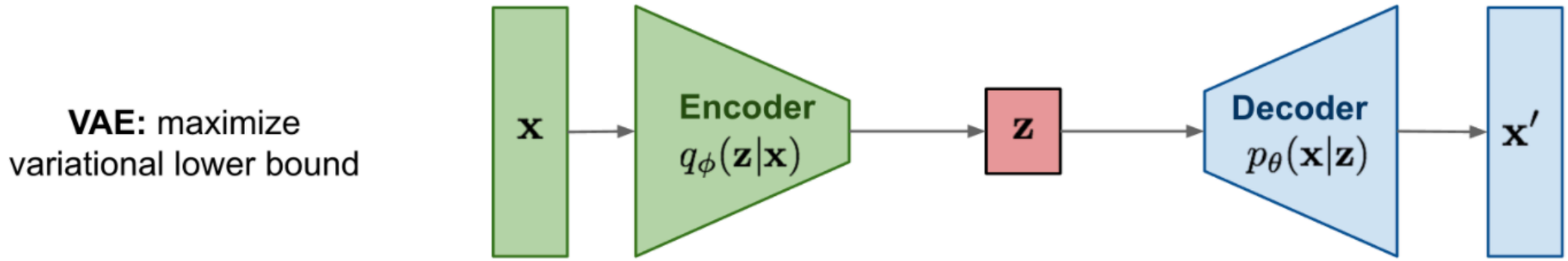
☹️ Mode collapse

Paganini, et al, Phys. Rev. D 97, 014021 (2018)
Fauuci et al, arXiv:2309.06515
Ratnikov et al, arXiv:2207.04340

Generative model: VAE

A summary blog

Variational Autoencoders: A pair of networks where one embed the data into a latent space with a given prior and the other decode back to the data space.



🙂 Structured latent representations

☹️ Less realistic outputs

Cresswell, et al, arXiv:2211.15380
Touranakou et al, arXiv:2203.00520
Abhishek et al, arXiv:2210.07430

Generative model: NF

A summary blog

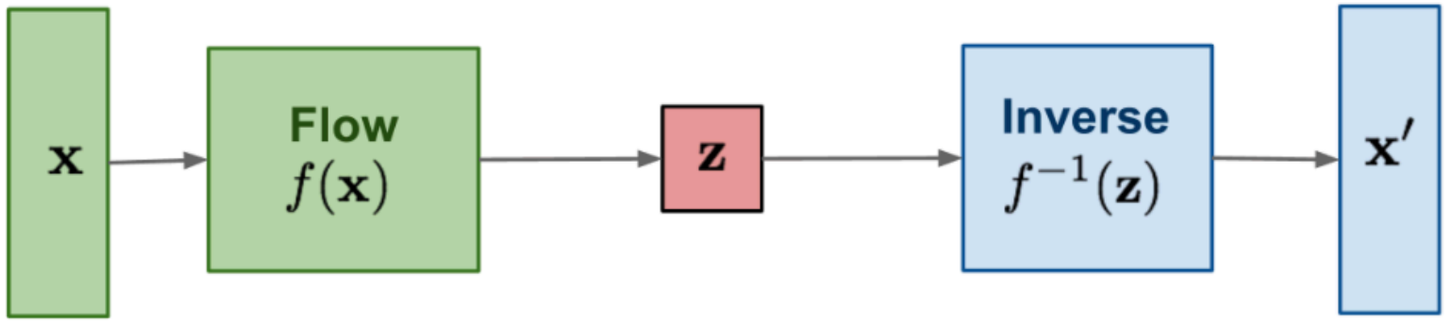
Normalising flow: invertible transformations to map a simple distribution to a complex one.

🙂 Exact likelihood computation

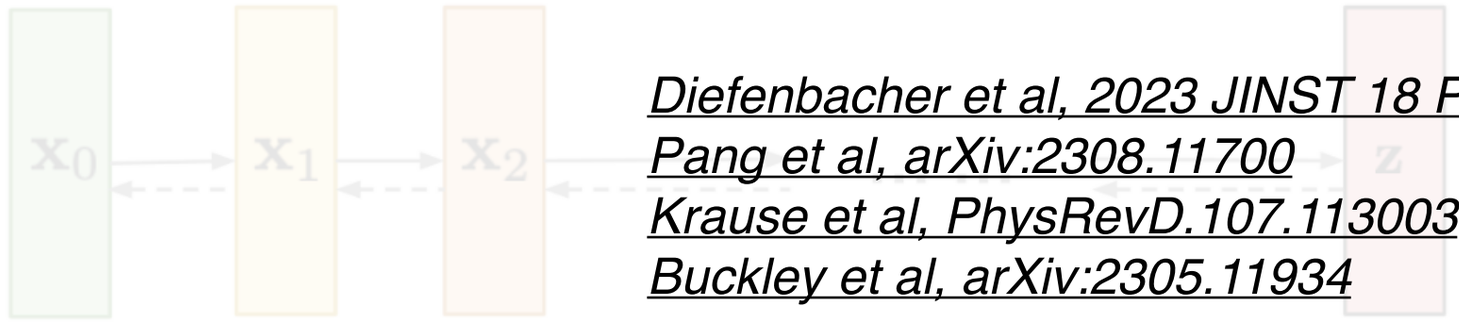
🙂 High generative capacity

☹ Slow sampling

Flow-based models:
Invertible transform of distributions



Diffusion models:
Gradually add Gaussian noise and then reverse



Diefenbacher et al, 2023 JINST 18 P10017
Pang et al, arXiv:2308.11700
Krause et al, PhysRevD.107.113003
Buckley et al, arXiv:2305.11934

Generative model: Diffusion model

A summary blog

Diffusion model: gradually add Gaussian noise to input and learn the added noise using NN.

🙂 Strong generative performance

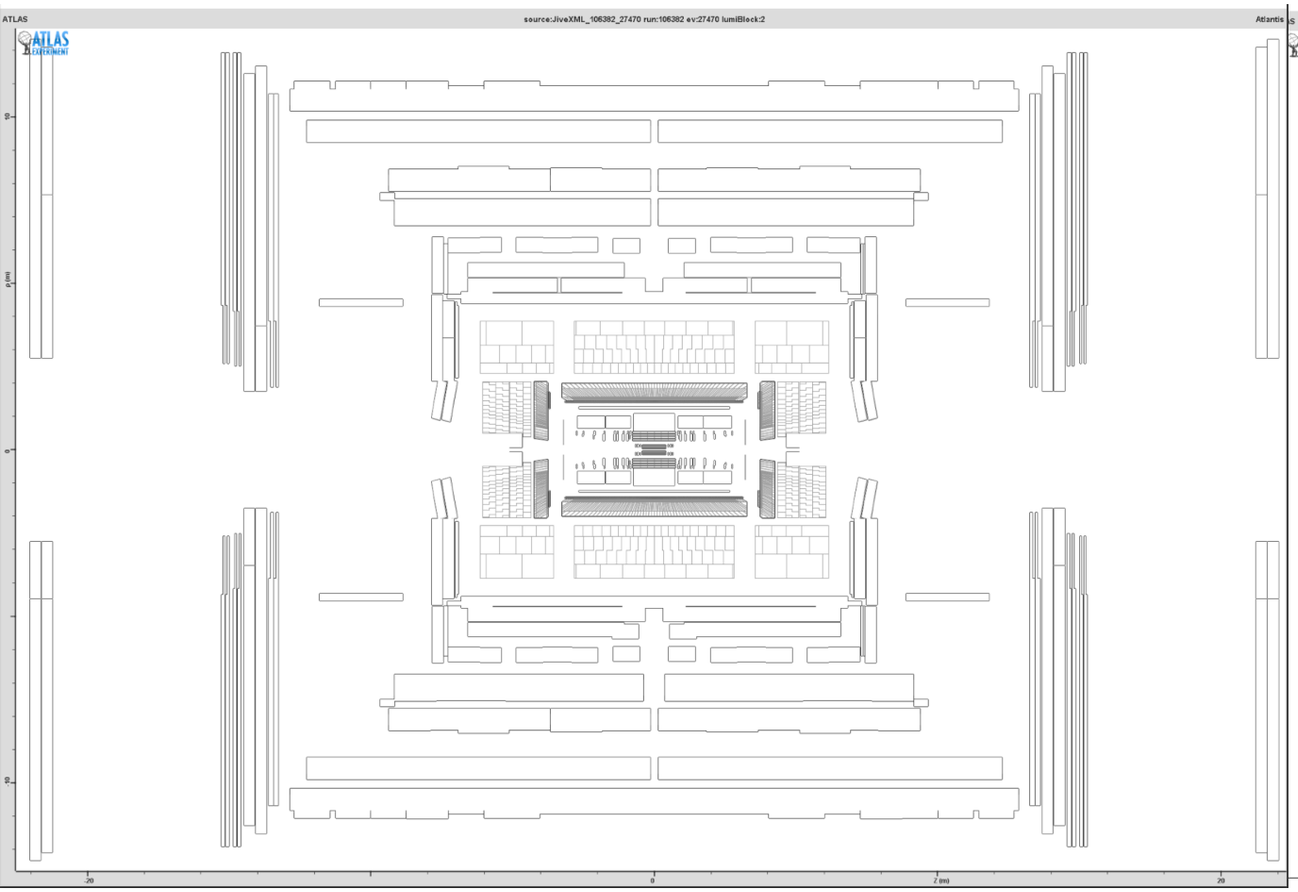
☹️ Slow sampling

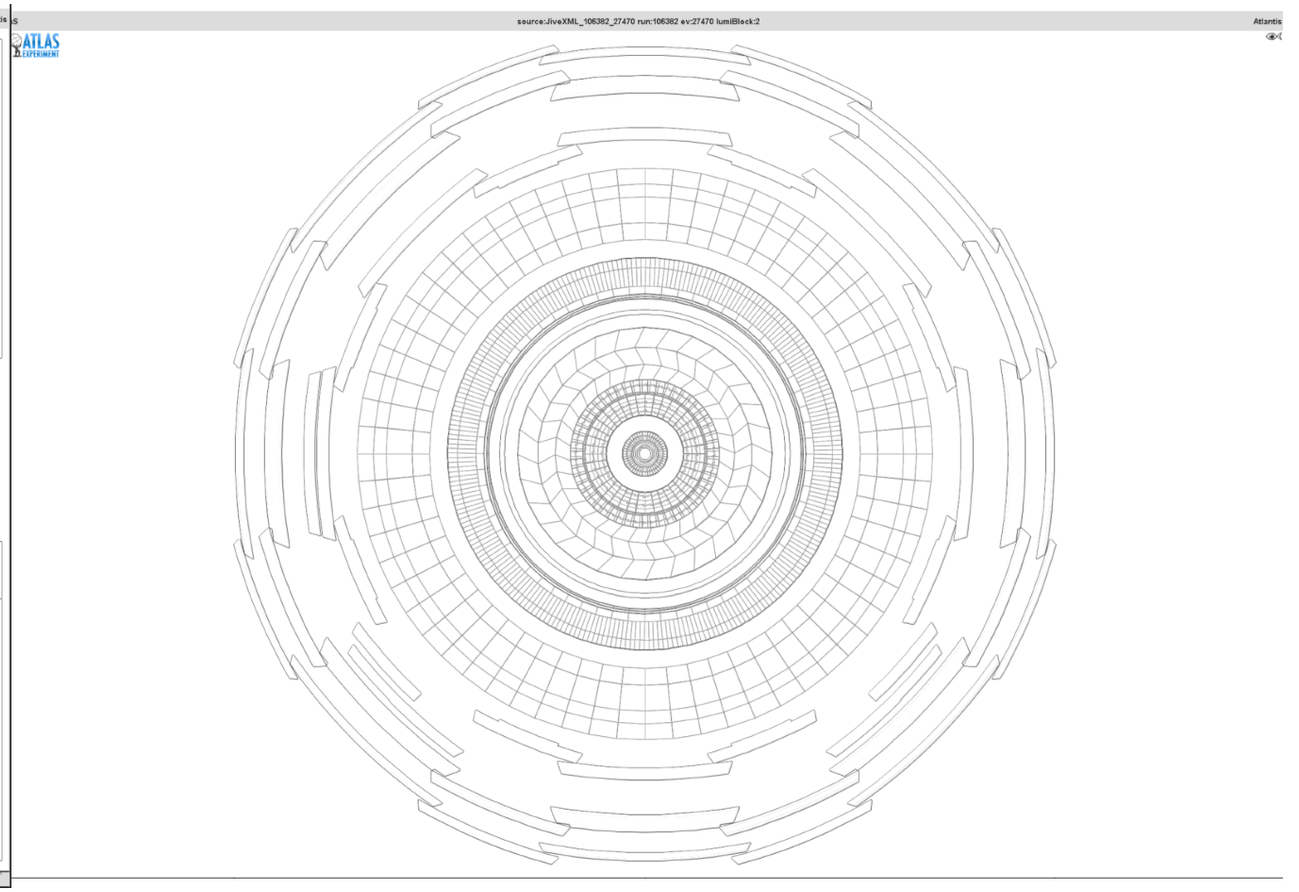
☹️ Difficult to train

Amram et al, Phys. Rev. D 108 (2023) 072014
Buhmann et al, arXiv:2309.05704
Mikuni et al, arXiv:2308.03847

Diffusion models:
Gradually add Gaussian noise and then reverse

Non-uniform ATLAS geometry



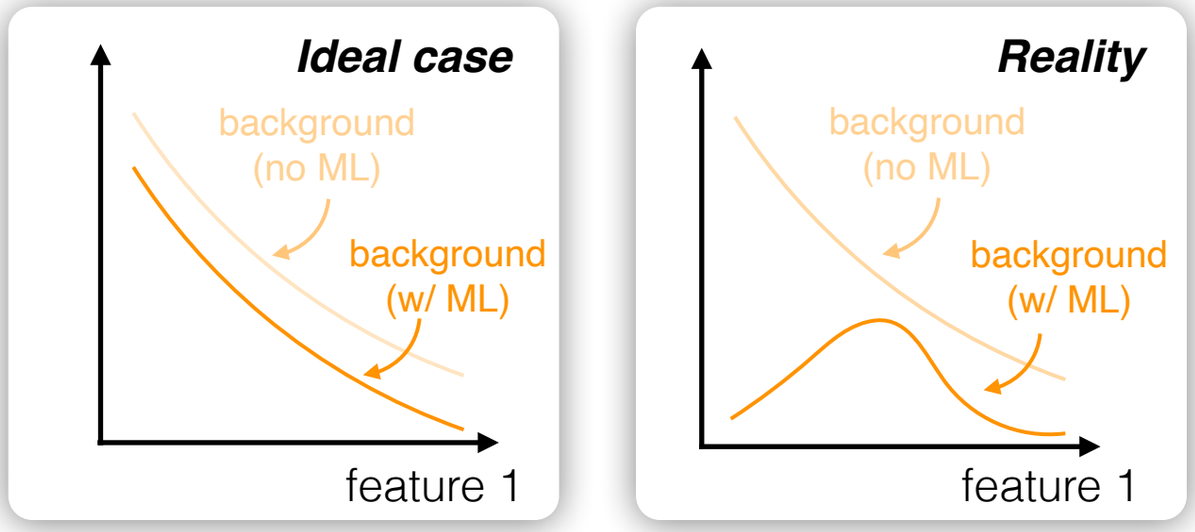


Decorrelation

Caution Part I

35

How can we learn a classifier that does not sculpt a bump in the background?



Enforcing Independence

39

Train e.g. a neural network with a **custom loss functional**

$$\mathcal{L}[f(x)] = \sum_{i\in s} L_{\text{classifier}}(f(x_i), 1) + \sum_{i\in b} L_{\text{classifier}}(f(x_i), 0) + \lambda \sum_{i\in b} L_{\text{decor}}(f(x_i), m_i)$$

Recent proposals:

Adversaries: L_{decor} is the loss of **a 2nd NN** (adversary) that tries to learn m from $f(x)$.

Distance Correlation: L_{decor} is **distance correlation** (generalizes Pearson correlation) between m and $f(x)$.

Mode Decorrelation: L_{decor} is small when the **CDF** of $f(x)$ is the same across different values of m.

Nachman, Overview of Machine Jet Image Learning for Particle Physics