ML em HEP II

Escola INCT 2024

A. Sznajder

UERJ
Instituto de Fisica

Novembro - 2024

Outline

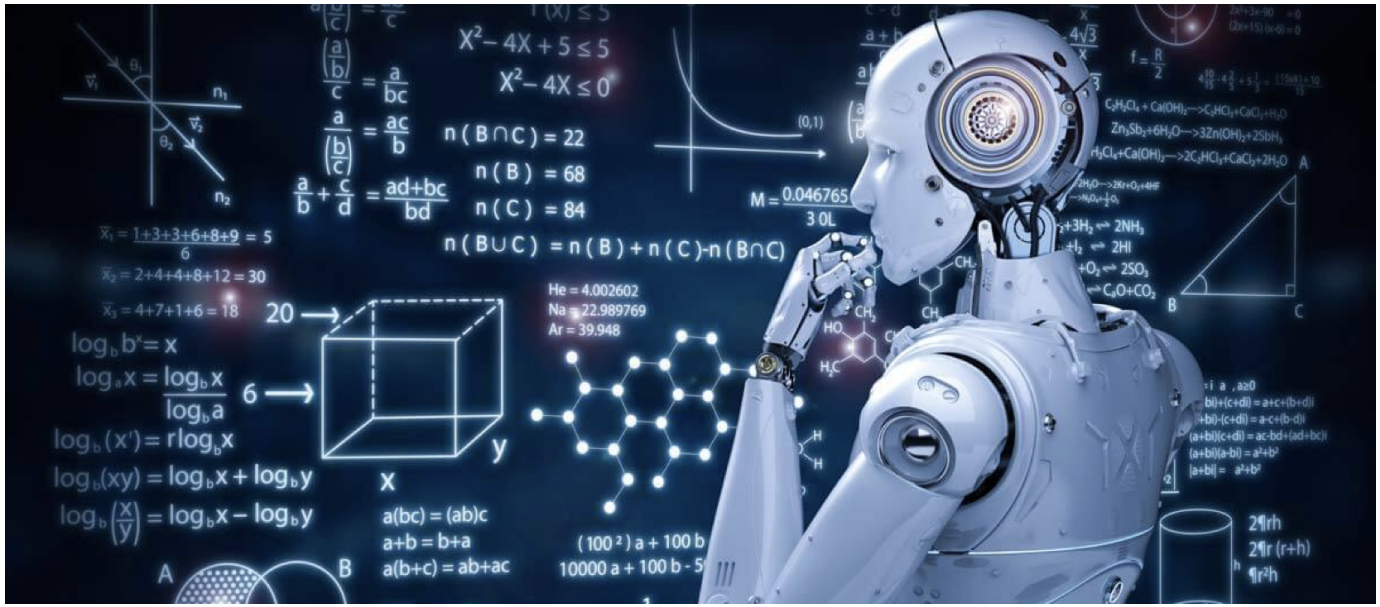
Support Material

- **Textbook:**
  Aurélien Géron - Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow
- **Jupyter Notebooks :**
  `https://github.com/INCT-CERN-Brasil/Escola2024`
- **Google Colab :**
  `https://colab.google`

What is Machine Learning ?

Machine Learning (ML)

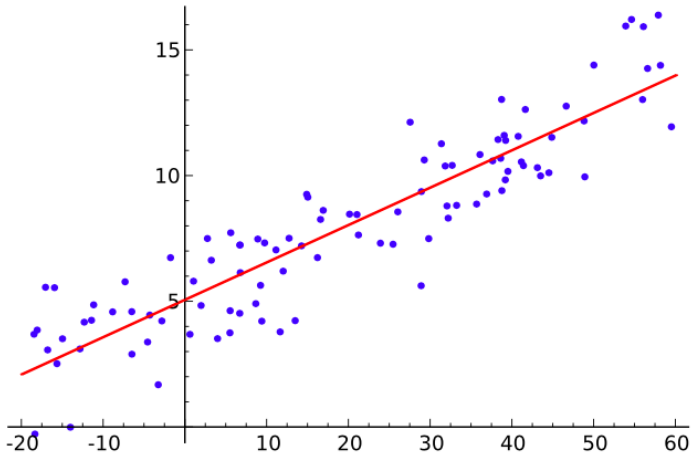
Machine learning (ML) is the study of computer algorithms capable of building a predictive model out of a data samples (learn from examples), without being explicitly programmed (sequence of instructions).



Centuries Old Machine Learning [1]

Take some points on a 2D graph, and fit a function to them. What you have just done is generalized from a few (x, y) pairs (examples) , to a general function that can map any input x to an output y

The Centuries Old Machine Learning Algorithm



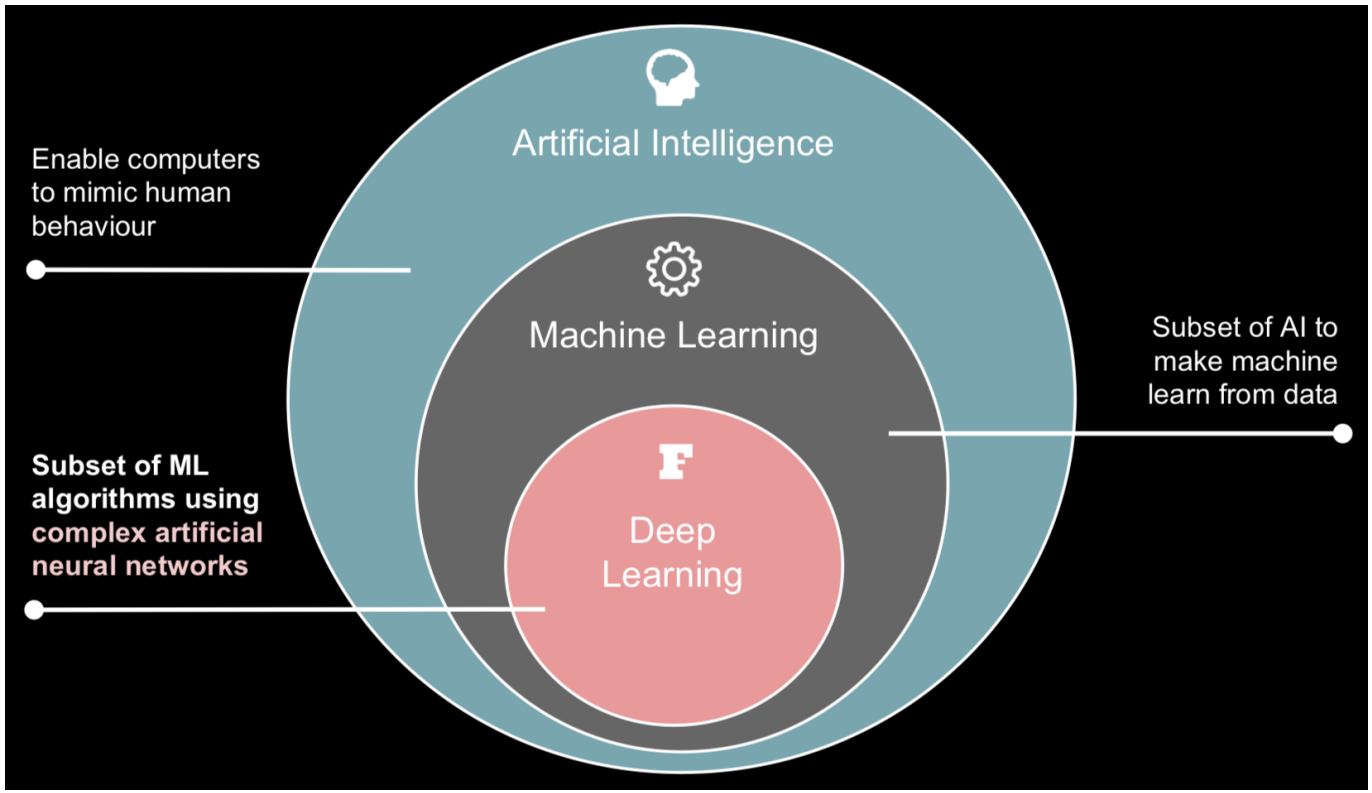
Linear regression (Source)

Linear regression is a bit too wimpy to solve the complex problems dealing with image, speech or text, but what it does is essentially what supervised ML is all about (glorified regression) !

[1] http://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learnin

Artificial Inteligence

"Inteligence can be understood as the ability to process current information to inform future decisions"



None of current ML systems we have are real AI and the brain learns so efficiently that no ML method can match it [2]

[2] Yann LeCun , Epistemology of Deep Learning : https://www.youtube.com/watch?v=gG5NCkMerHU&t=944s , https://www.youtube.com/watch?v=cWzi38-vDbE&t=768s

Artificial Inteligence

None of the systems we have nowadays are real AI. The brain learns so efficiently that no ML method can match it !

AI versus Brain

- Brain has 10^{14} parameters and we live only $10^9 s$ (lot more parameters than data)
- So, it must do lots of unsupervised and self-supervised learning and must predict what we observe !
- Supervised and reinforcement learning requires milions of examples(trials)
- Still missing a learning paradigm that builds predictive models through observation and action

Yann LeCun on the Epistemology of Deep Learning:
https://www.youtube.com/watch?v=gG5NCkMerHU&t=944s
https://www.youtube.com/watch?v=cWzi38-vDbE&t=768s

Introduction to Machine Learning

Machine Learning(ML) can be approached from many different angles:

1) Tasks:

- Classification
- Regression
- Data Generation or Simulation

2) Data Structure

- Tabular
- Image
- Sequence
- Graph
- Sets

3) Learning Paradigms

- Supervised Learning (labeled)
- Un(Self)supervised Learning (unlabeled)
- Reinforcement Learning (reward)

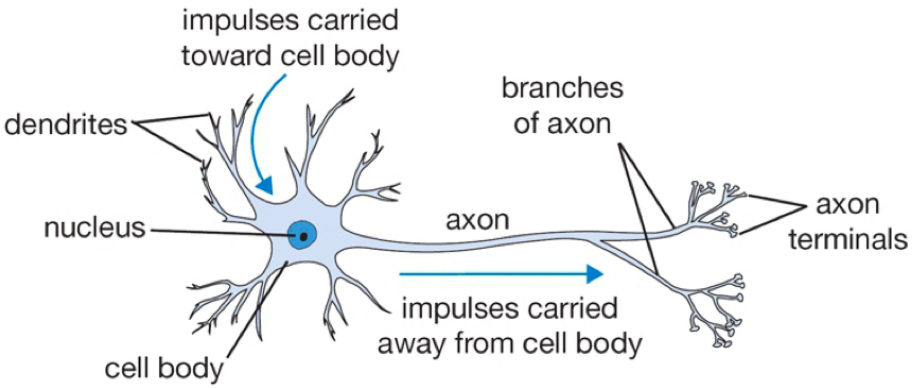
3) Neural Networks Architectures

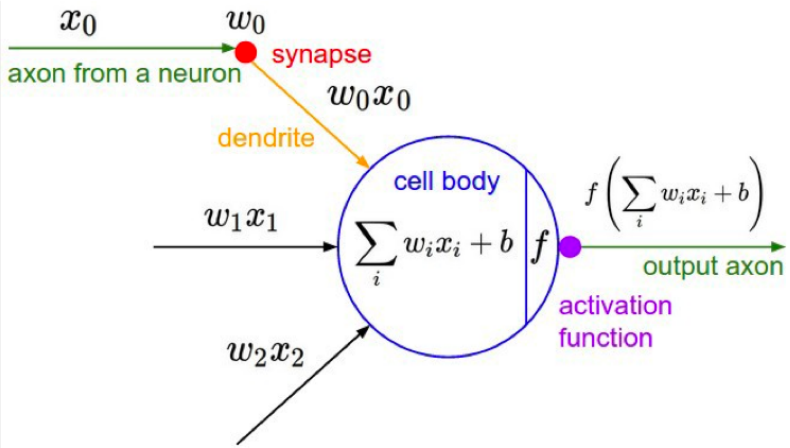
- Multilayer Perceptron (MLP)
- Recurrent Networks (RNN,LSTM,GRU)
- Transformer Networks (TN)
- Convolutional Networks (CNN)
- Graph Networks (GCN,GAT,DGCN,IN)
- Point Nets and Deep Sets (PN,DS)
- Autoencoders (AE,DAE,VAE)
- Generative Adversarial Network (GAN)
- Normalizing Flows (NF)
- Diffusion Models (DM,SDM)

Obs: ML can be implemented by different algorithms (ex: SVM, BDT, PCA ...) but we will discuss only Neural Networks

Neural Networks

Artificial Neural Networks (NN) are computational models vaguely inspired [3] by biological neural networks. A NN is formed by a network of basic elements called neurons, which receive an input, change their state according to the input and produce an output





Original goal of NN approach was to solve problems like a human brain. However, focus moved to performing specific tasks, deviating from biology. Nowadays NN are used on a variety of tasks: image and speech recognition, translation, filtering, playing games, medical diagnosis, autonomous vehicles, ...

[3] Design of airplanes was inspired by birds, but airplanes don't flap wings to fly !

Artificial Neuron

Artificial Neuron Model

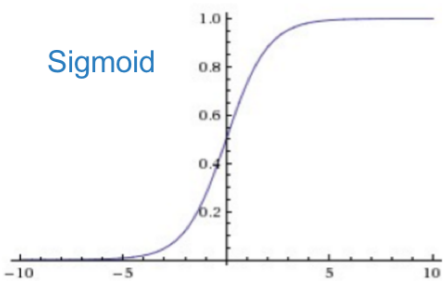
Each node of a NN receives inputs $\vec{x} = \{x_1, \ldots, x_n\}$ from other nodes or an external source and computes an output y according to the expression

$$y = F\left(\sum_{i=1}^{n} W_i x_i + B\right) = F\left(\vec{W}.\vec{x} + B\right) \quad (1)$$

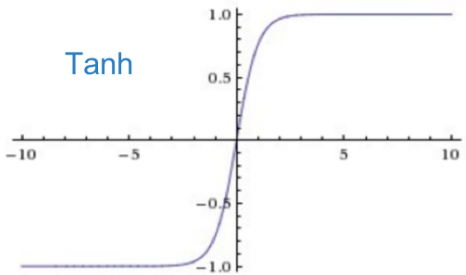
, where W_i are connection weights, B is the treshold and F the activation function [4]

There are a variety of possible activation function and the most common ones are

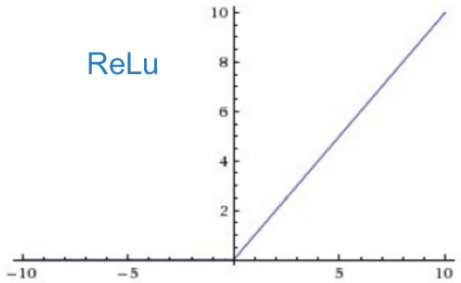
Sigmoid



Tanh



ReLu



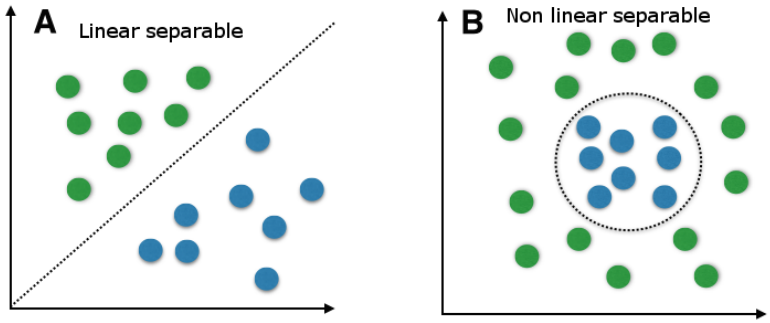
[4] Nonlinear activation is fundamental for nonlinear decision boundaries

The Perceptron

The perceptron [6] algorithm is a binary linear classifier invented in 1957 by F.Rosenblatt. It's formed by a single neuron that takes input $\vec{x} = (x_1, \ldots, x_n)$ and outputs $y = 0, 1$ according to

Perceptron Model [7]

$$y = \begin{cases} 1, \textit{ if } (\vec{W}.\vec{x} + B) > 0 \\ 0, \textit{ otherwise} \end{cases}$$



To simplify notation define $W_0 = B$, $\vec{x} = (1, x_1, \ldots, x_n)$ and call θ the Heaviside step function

Perceptron Learning Algorithm

1. Calculate the ouput error: $y_j = \theta(\vec{W}.\vec{x}_j)$ and $Error = 1/m \sum_{j=1}^{m} |y_j - t_j|$
2. Modify(update) the weights to minimize the error: $\delta W_i = r.(y_j - t_j).X_i$, where r is the learning rate
3. Return to step 1 until output error is acceptable

[6] https://medium.com/towards-data-science/perceptrons-the-first-neural-network-model-8b3ee
[7] Equation of a plane in $\mathbb{R}^n$ is $\vec{W}.\vec{x} + B = 0$

The Perceptron as a Universal Function Approximator

Universal Approximation Theorem

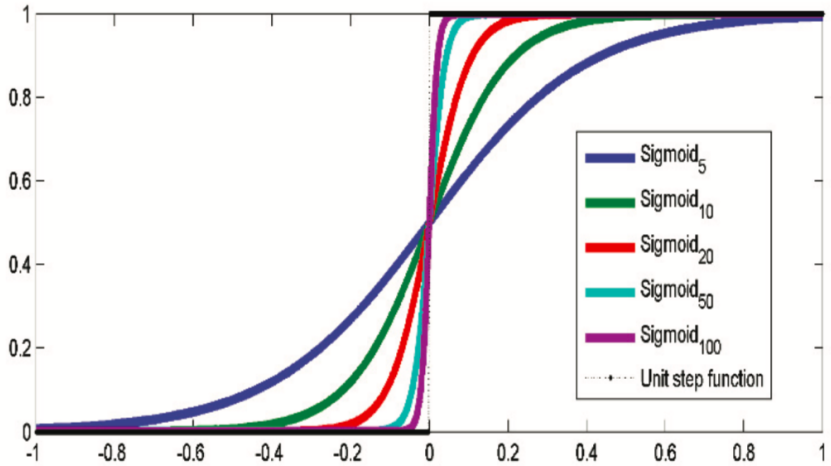
A single hidden layer feed forward neural network with a linear output unit can approximate any continuous function arbitrarily well, given enough hidden neurons [9]

The theorem doesn't tell us how many neurons or how much data is needed !

Sigmoid $\rightarrow$ Step Function

For large weight W the sigmoid turns into a step function, while B gives its offset

$$y = \frac{1}{1 + e^{-(w.x+b)}} \quad (2)$$



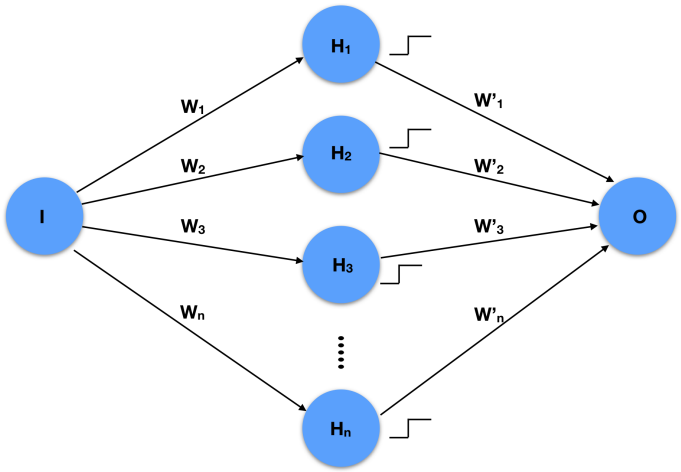
[9] Cybenko,G.(1989) Approximations by superpositions of sigmoidal functions, Math.ofCtrl.,Sig.,andSyst.,2(4),303
Hornik,K.(1991) Approximation Capabilities of Multilayer Feedforward Networks, Neural Networks,4(2), 251

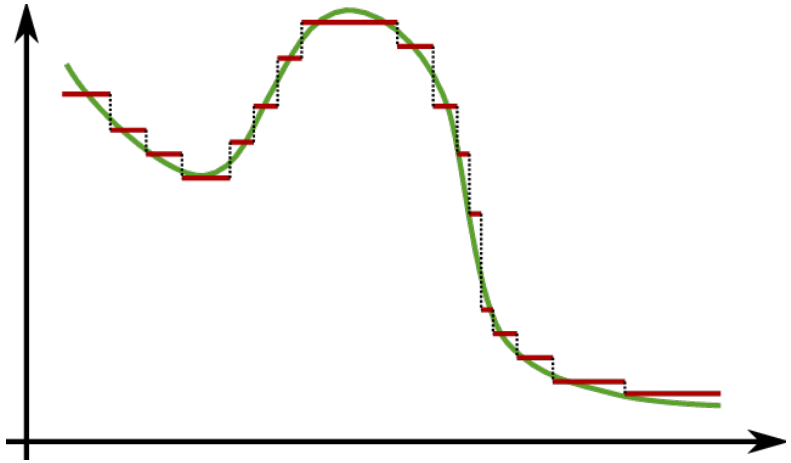
The Perceptron as a Universal Function Approximator

Approximating $F(x)$ by Sum of Steps

A continuous function can be aproximated by a finite sum of step functions. The larger the number of steps(nodes), the better the approximation. [10]

Consider a NN composed of a single input , *n* hidden nodes and a single output. Tune the weights such that the activations approximate steps functions with appropriate treshold and add them together !





One can always tune weights such that any activation behaves aproximately as a step function !

[10] M.Nielsen , http://neuralnetworksanddeeplearning.com/chap4.html
M.Cranmer, https://www.youtube.com/watch?v=fk2r8y5TfNY
S.Prince, Understanding Deep Learning 2023

Multilayer Perceptron (MLP)

The Multilayer Perceptron(MLP) is a fully connected NN with at least 1 hidden layer and nonlinear activation function F [11] . It is the simplest deep NN with a compositional "inductive bias". [12]

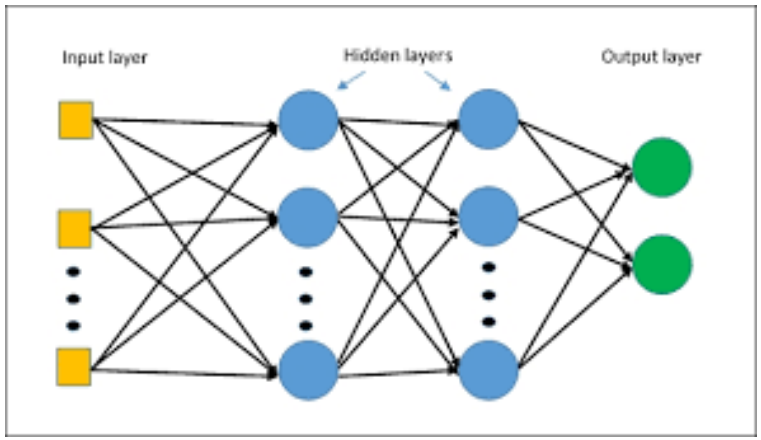
Multilayer Perceptron Model

For a MLP with inputs nodes $\vec{x}^{(0)}$, one hidden layer of nodes $\vec{x}^{(1)}$ and output layer of nodes $\vec{x}^{(2)}$, we have

$$\begin{cases} \vec{x}^{(1)} = \vec{F}^{(1)}\left(\vec{W}^{(1)}.\vec{x}^{(0)}\right) \\ \vec{x}^{(2)} = \vec{F}^{(2)}\left(\vec{W}^{(2)}.\vec{x}^{(1)}\right) \end{cases}$$

Eliminating the hidden layer variables $\vec{H}$ we get

$$\Rightarrow \vec{x}^{(2)} = \vec{F}^{(2)}\left(\vec{W}^{(2)}.\vec{F}^{(1)}\left(\vec{W}^{(1)}.\vec{x}^{(0)}\right)\right) \quad (3)$$



A MLP can be seen as a parametrized composite mapping $F_{w,b} : \mathbb{R}^n \rightarrow \mathbb{R}^m$

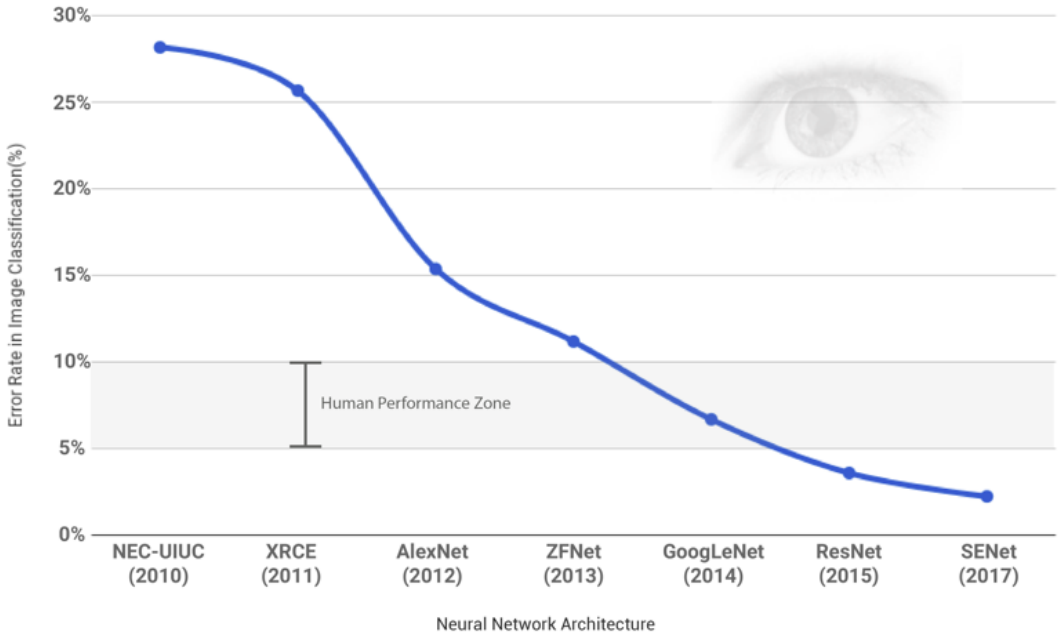
[11] A MLP with m layers using linear activation functions can be reduced to a single layer !

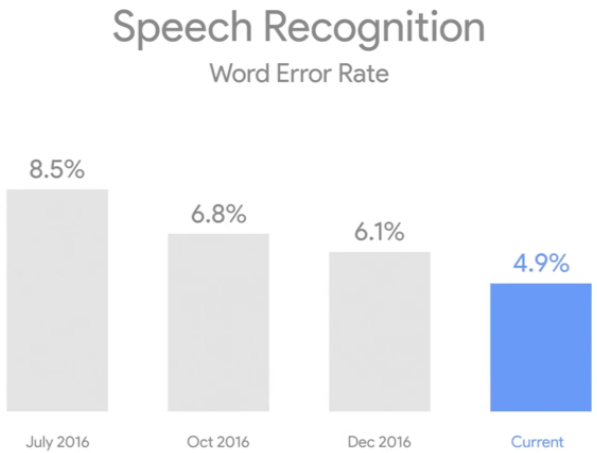
[12] The tresholds $\vec{B}$ are represented as weights by redefining $\vec{W} = (B, W_1, \ldots, W_n)$ and $\vec{x} = (1, x_1, \ldots, x_n)$ (bias is equivalent to a weight on an extra input of activation=1)

Why Deep Learning ?
and
Why Now ?

Why Deep Learning ?

Image and Speech Recognition perfomance (DNN versus Humans)

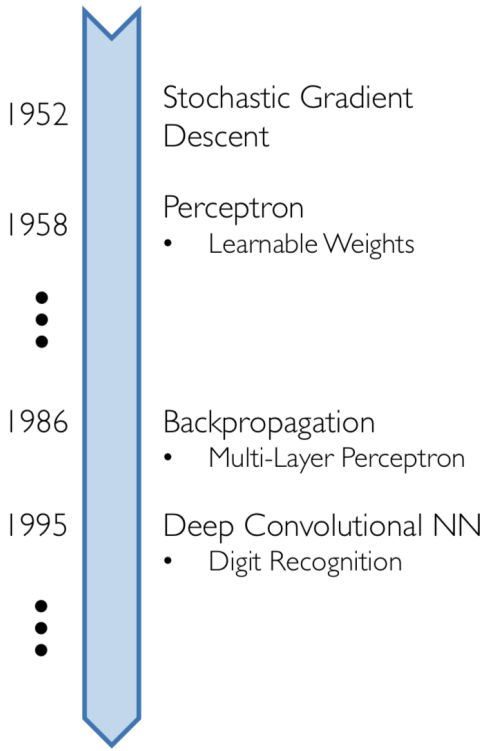




https://arxiv.org/pdf/1409.0575.pdf

Why Now ?

Neural networks date back decades , so why the current resurgence ?

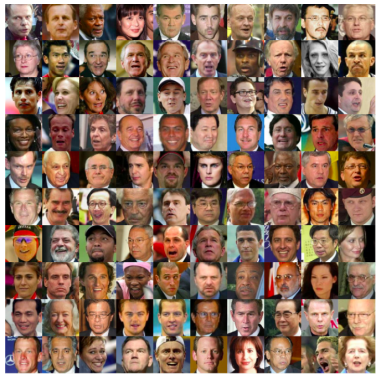


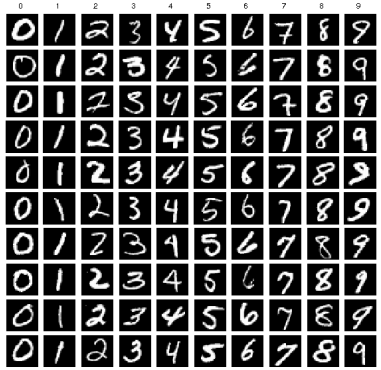
The main catalysts for the current Deep Learning revolution have been:

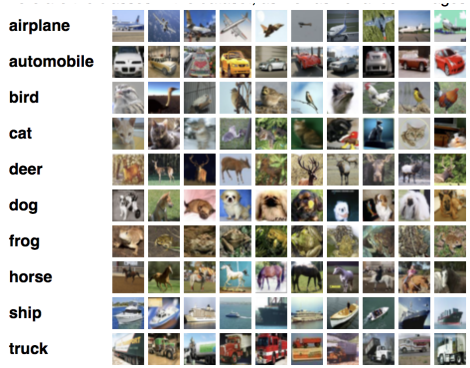
- **Software:**
  TensorFlow, PyTorch, Keras and Scikit-Learn
- **Hardware:**
  GPU, TPU and FPGA
- **Large Datasets:**
  MNIST

Training Datasets

Large and new open source datasets for machine learning research [26]





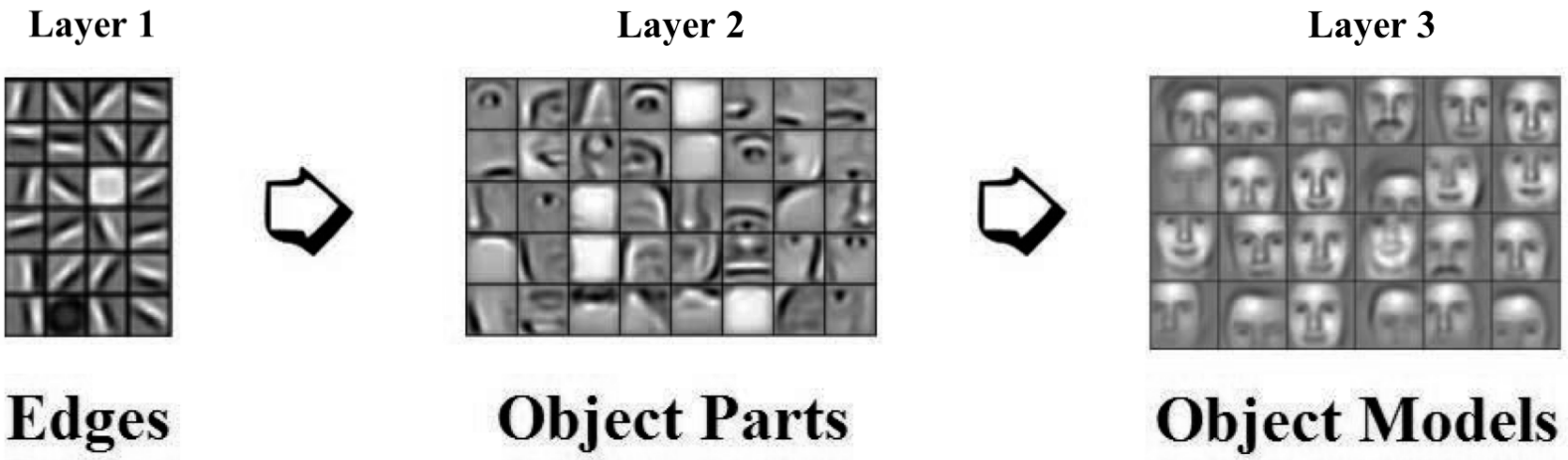


[26] https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research

Deep Learning - Need for Depth

Deep Neural Networks(DNN)

Depth allows the NN to factorize the data features, distributing its representation hierarchically across the layers



$\Rightarrow$ DNN explores the compositional character of nature as an inductive bias !!! [27]

[27] Deep Learning , Y.LeCunn, J.Bengio, G.Hinton , Nature , vol. 521, pg. 436 , May 2015

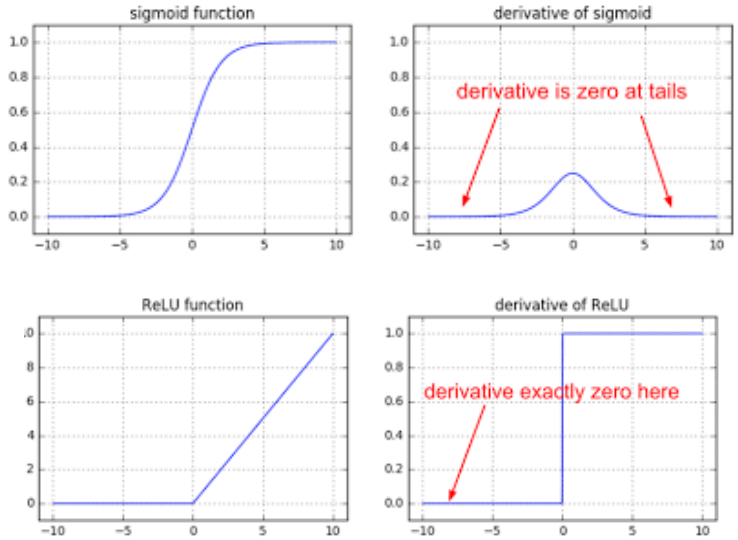
Deep Learning - Vanishing Gradient Problem

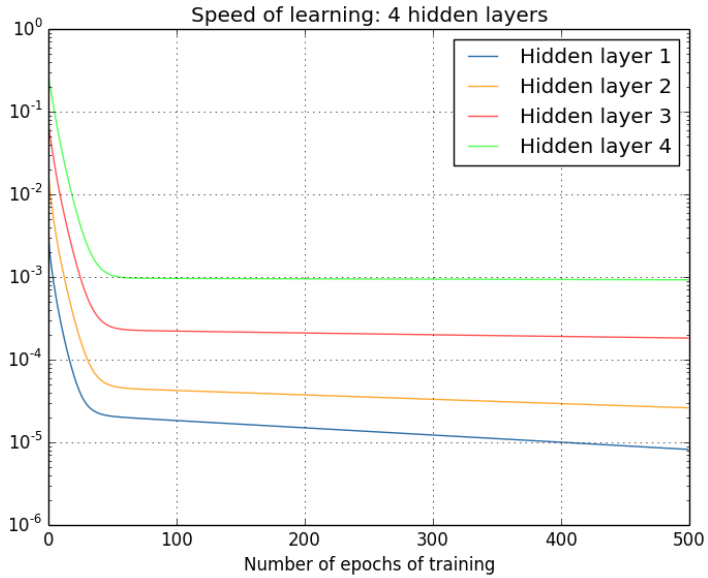
Vanishing Gradient Problem [28]

Backpropagation computes gradients iteratively by multiplying the activation function derivate F' through n layers.

$$\delta_k^{(l)} = \left(\sum_m \delta_m^{(l+1)} W_{mk}^{(l+1)} \right) F'(z_k^{(l)})$$

For $\sigma(x)$ and $Tanh(x)$ the derivate F' is asymptoticaly zero, so weights updates gets vanishing small when backpropagated $\Rightarrow$ Then, earlier layers learns much slower than later layers !!!





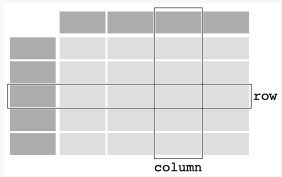
[28] http://neuralnetworksanddeeplearning.com/chap5.html

Deep Architectures and Applications

Data Structures

Data comes structured in a variety of formats.

Tabular Data



Sequential Data

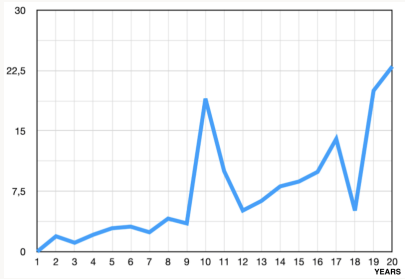
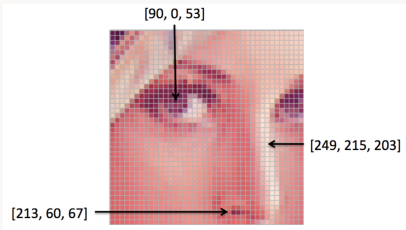


Image Data



Graph & Point Sets Data

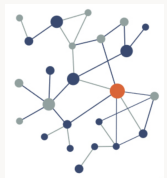
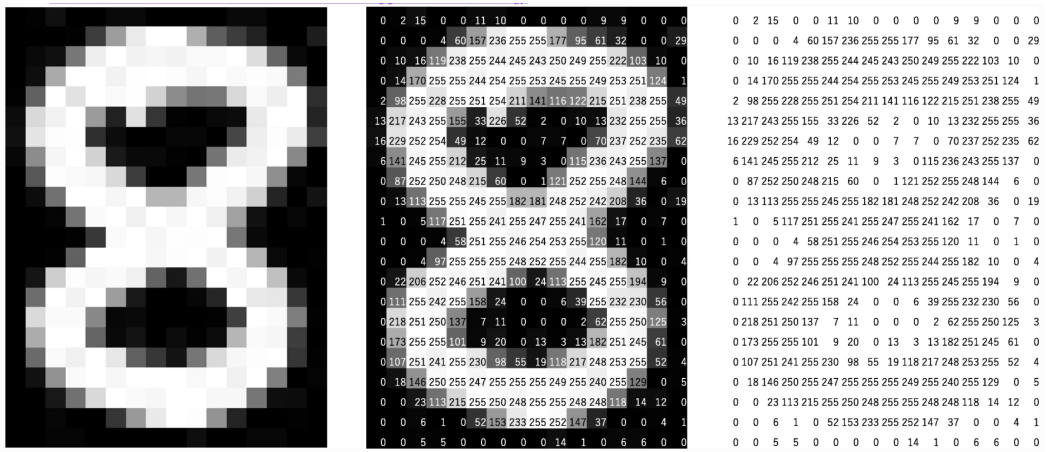


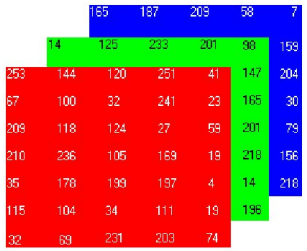
Image Structured Data

Image Representation

A computer sees an image as an array of numbers. The image is associated to a matrix containing numbers between 0 and 255, corresponding to each pixel brightness.



The image color RGB can be represented by expanding the depth of the representation, where each matrix represent a fundamental color intensity

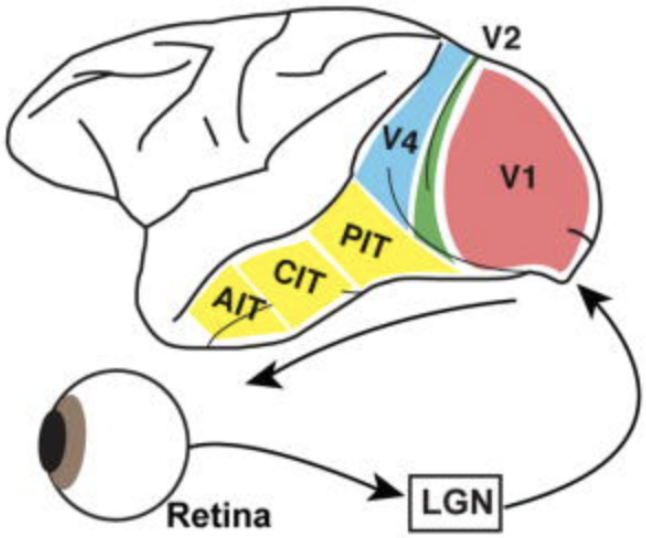


Convolutional Neural Network

Full connectivity between neurons in a MLP makes it computationally too expensive to deal with high resolution images. A $1000x1000$ pixels image leads to $O(10^6)$ weights per neuron !

Convolutional Neural Network (CNN)

Inspired by the visual cortex [30] , where neurons respond to stimuli only in a restricted region of the visual field, a CNN mitigates the challenges of high dimensional inputs by restricting the connections between the input and hidden neurons. It connects only a small contiguous region, exploiting local image features.



A CNN uses inductive biases (local connectivity) built into the network layers to reduce de number of learnable parameters

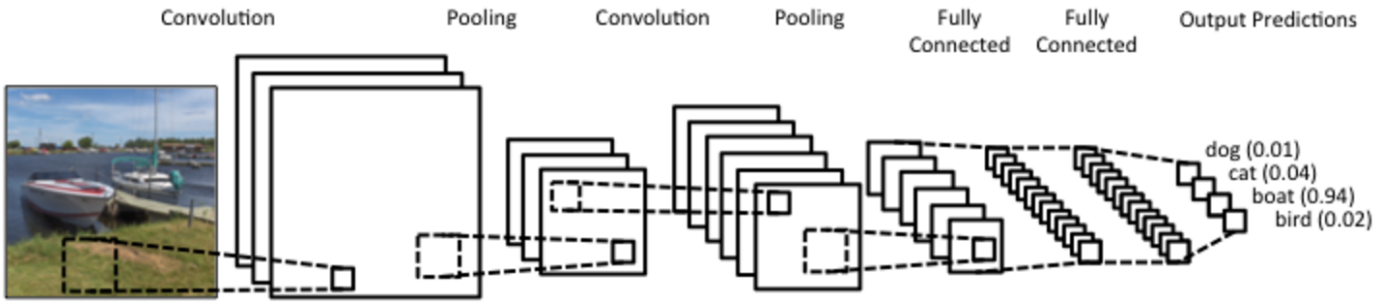
[30] How does the brain solve visual object recognition? , https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3306444

Convolutional Neural Network

Convolutional Neural Network (CNN)

A typical CNN architecture is composed by a stack of distinct and specialized layers:

1. Convolutional (extract image feature maps) [31]
2. Pooling (downsampling to reduce size)
3. Fully connected (MLP for image classification)



[31] http://ufldl.stanford.edu/tutorial/supervised/FeatureExtractionUsingConvolution

CNN - Convolutional Layer

Convolutional Layer

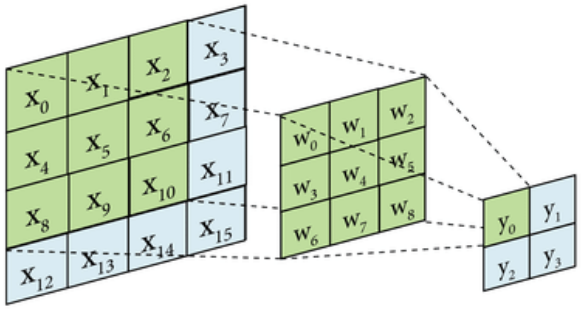
The convolutional layer [32] is the CNN core building block. A convolution can be seen as a sliding window transformation that applies a filter to extract local image features.

Discrete Convolution

The convolution has a set of learnable filters weights that are shared across the image

$$y_{ij}^{(l+1)} = \sum_{a=0}^{n-1} \sum_{b=0}^{m-1} w_{ab}^{(l)} x_{(i+a)(j+b)}^{(l)}$$

(Click on the figure)



The same set of weights of a given filter are applied all across the image, reducing the number of learnable parameters (shared weights)

[32] http://ufldl.stanford.edu/tutorial/supervised/FeatureExtractionUsingConvolution
https://machinelearningmastery.com/padding-and-stride-for-convolutional-neural-networks

Convolutional Filters

A convolution filter can apply an effect (sharpen, blurr), as well as extract features (edges, texture) [33]

0	0	0	0	0
0	0	-1	0	0
0	-1	5	-1	0
0	0	-1	0	0
0	0	0	0	0

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

	0	1	0	
	1	-4	1	
	0	1	0	

	-2	-1	0	
	-1	1	1	
	0	1	2	

[33] https://docs.gimp.org/2.6/en/plug-in-convmatrix.html

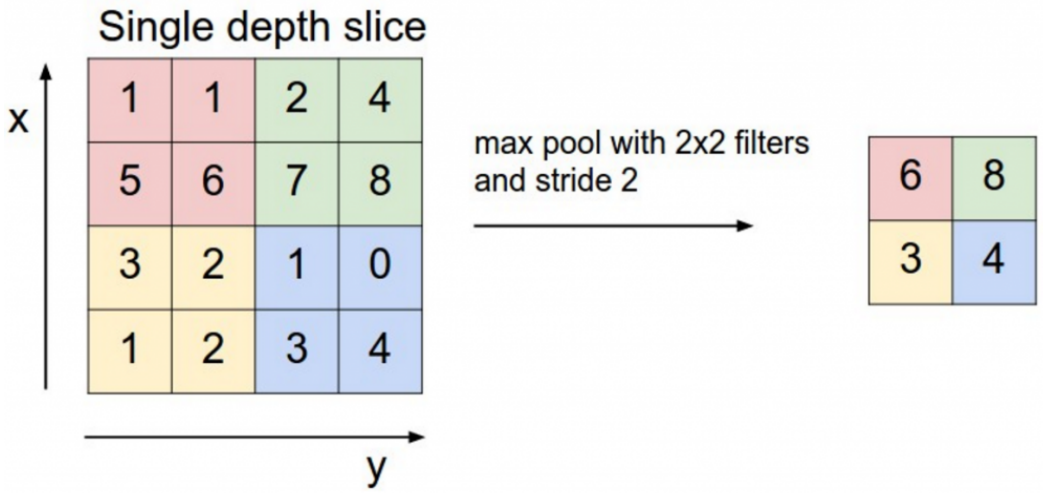
CNN - Pooling Layer

Pooling(Downsampling) Layer

The pooling (downsampling) layer [34] has no learning capabilities and serves a dual purpose:

- Decrease the representation size $\Rightarrow$ reduce computation
- Make the representation approximately invariant to small input translations and rotations

Pooling layer partitions the input in non-overlapping regions and, for each sub-region, it outputs a single value (ex: max pooling, mean pooling)



[34] http://ufldl.stanford.edu/tutorial/supervised/Pooling

Typical CNN Architecture

Typical Architecture

- CNN usually have pyramidal shape, decreasing in spatial extent and increasing in feature space
- Multiple layers of convolutional filters extract more-and-more abstract features

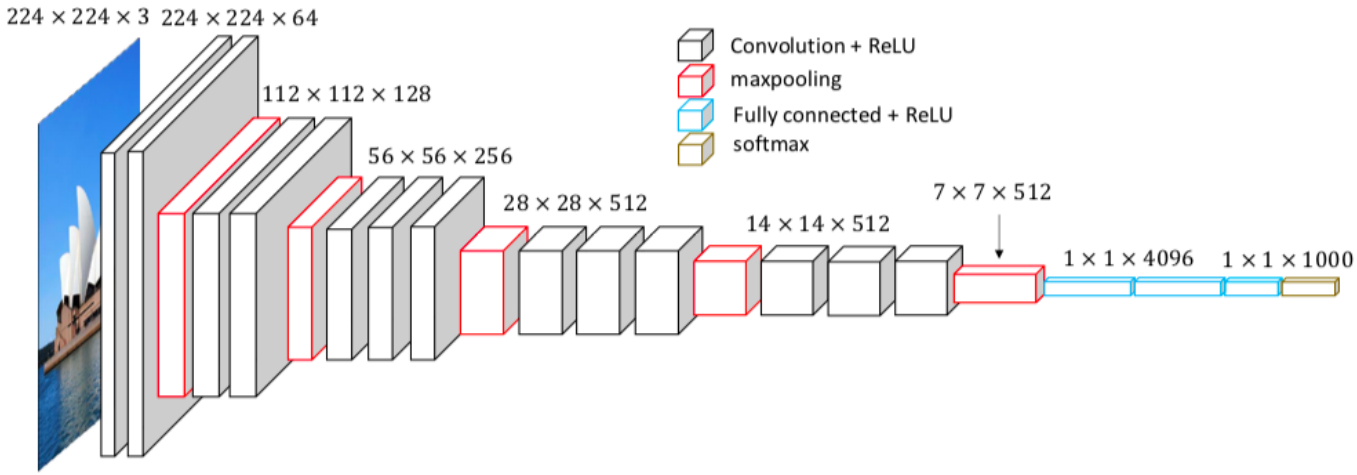
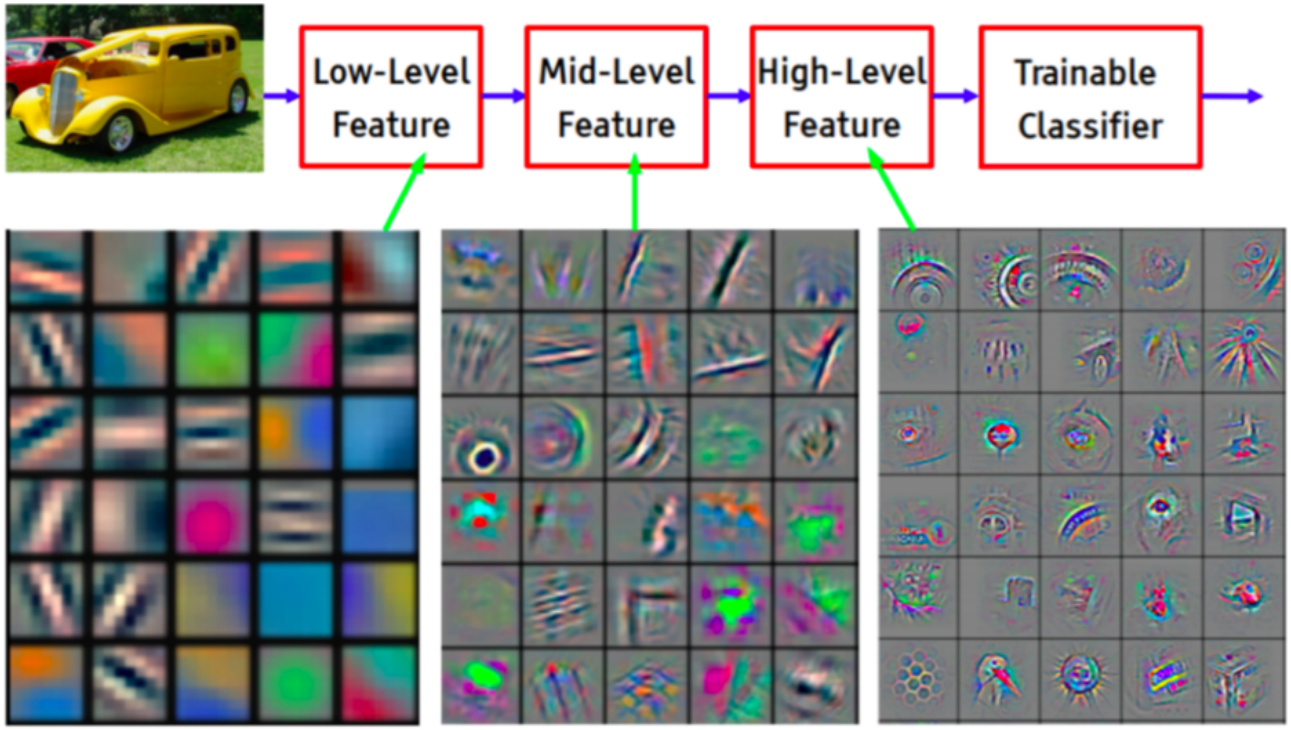


Figure 2: The architecture of VGG16 model .

CNN Feature Visualization

Each CNN layer is responsible for capturing a different level of features as can be seen from ImagiNet [35]

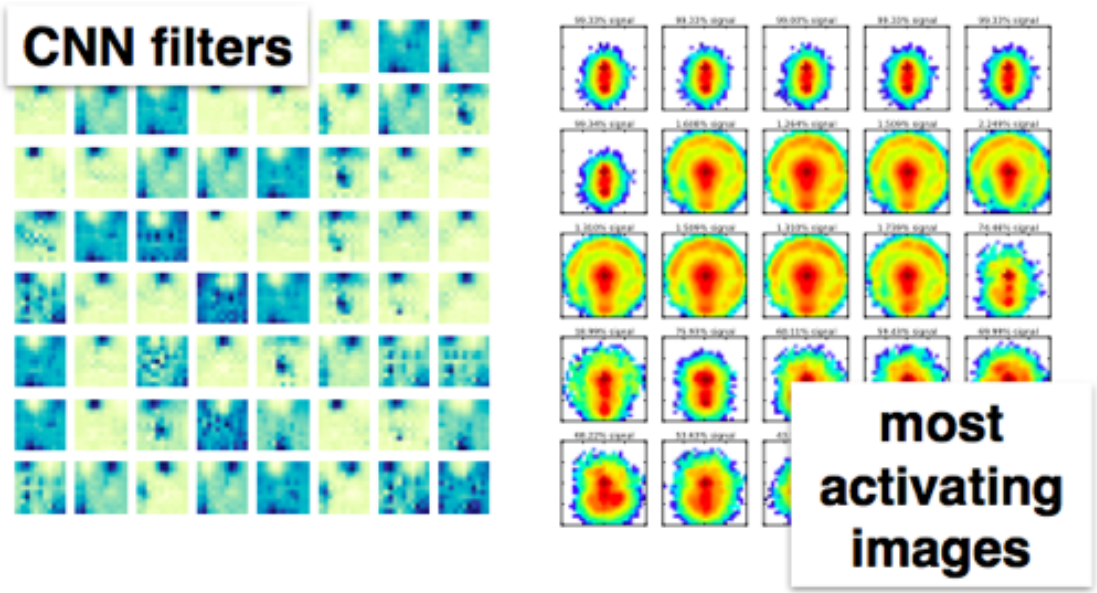


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

[35] https://arxiv.org/pdf/1311.2901

CNN Application in HEP: Jet ID

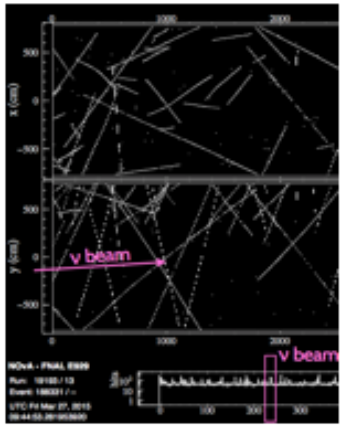
Jet images with convolutional nets

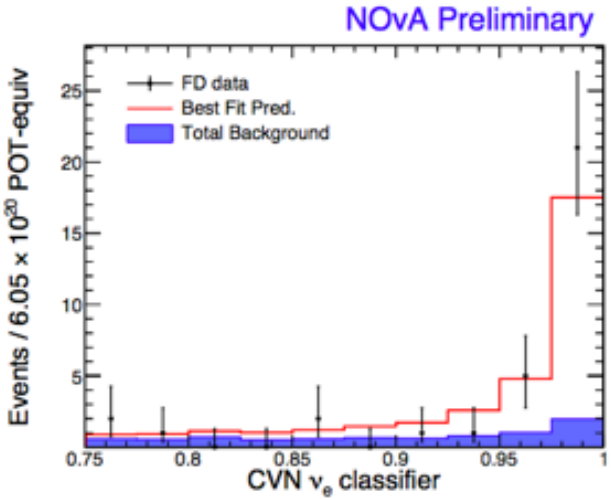


L. de Oliveira et al., 2015

CNN Application HEP: Neutrino ID

Neutrinos with convolutional nets

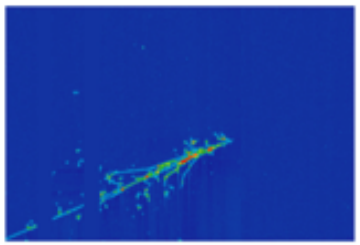


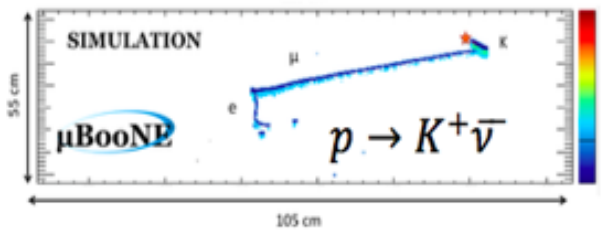


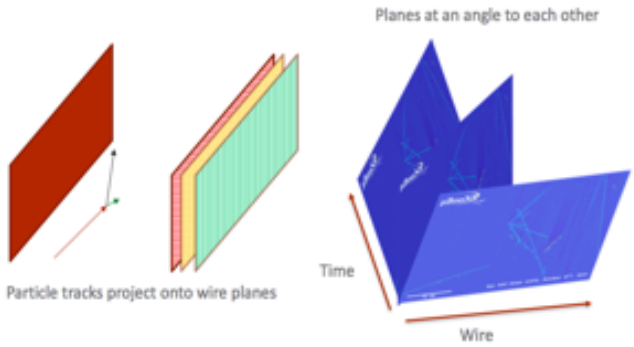
76% Purit
73% Effici

An equivalent increased exposure of 30%

Aurisiano et al. 2016







μBooNE

Sequential Data

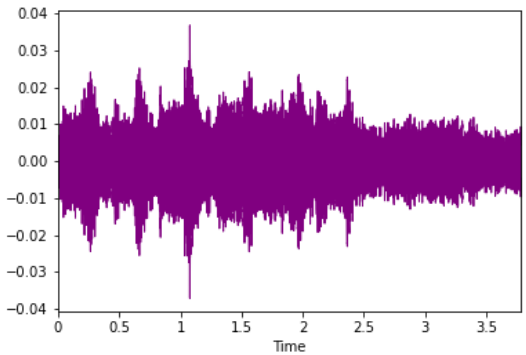
Sequential Data

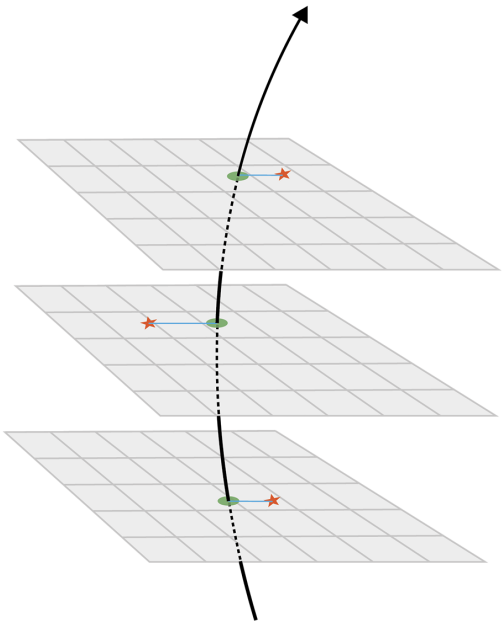
Sequential data is an interdependent data stream where data ordering contains relevant information

The food was good, not bad at all.

vs.

The food was bad, not good at all.





⇒ brain memorizes sequences for alphabet, words, phone numbers and not just symbols !

Recurrent Neural Network(RNN)

Feed forward networks can't learn correlation between previous and current input !

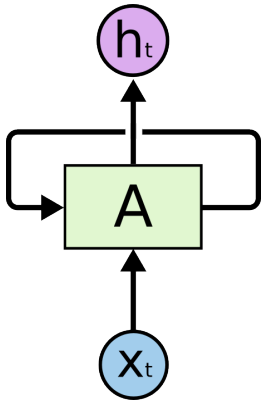
Recurrent Neural Networks (RNN)

RNNs are networks that use feedback loops to process sequential data [36]. These feedbacks allows information to persist, which is an effect often described as **memory**.

RNN Cell (Neuron [37])

The hidden state depends not only on the current input, but also on the entire history of past inputs

1. **Hidden State:** $h^{[t]} = F(W_{xh}x^{[t]} + W_{hh}h^{[t-1]})$
2. **Output:** $y^{[t]} = W_{hy}h^{[t]}$



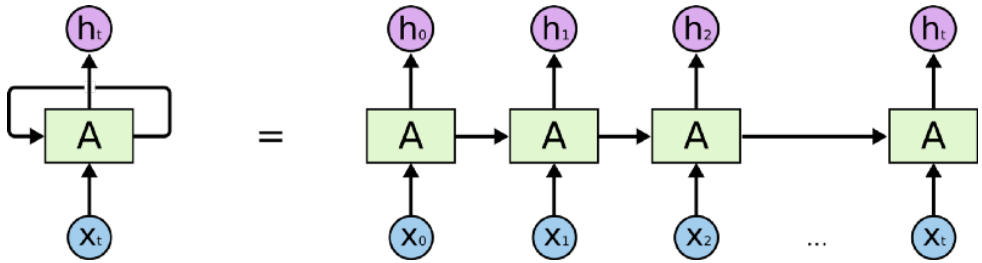
[36] https://eli.thegreenplace.net/2018/understanding-how-to-implement-a-character-based-rnn-
[37] https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks

Recurrent Neural Network(RNN)

RNNs process each input sequence element at a time, maintaining in their hidden units a 'state vector' that contains information about all the past elements history. [38]

RNN Unrolling

A RNN can be thought of as multiple copies of the same network, each passing a message to a successor. Unrolling is a visualization tool which views a RNN as a sequence of unit cells.



Backpropagation Through Time (BPTT)

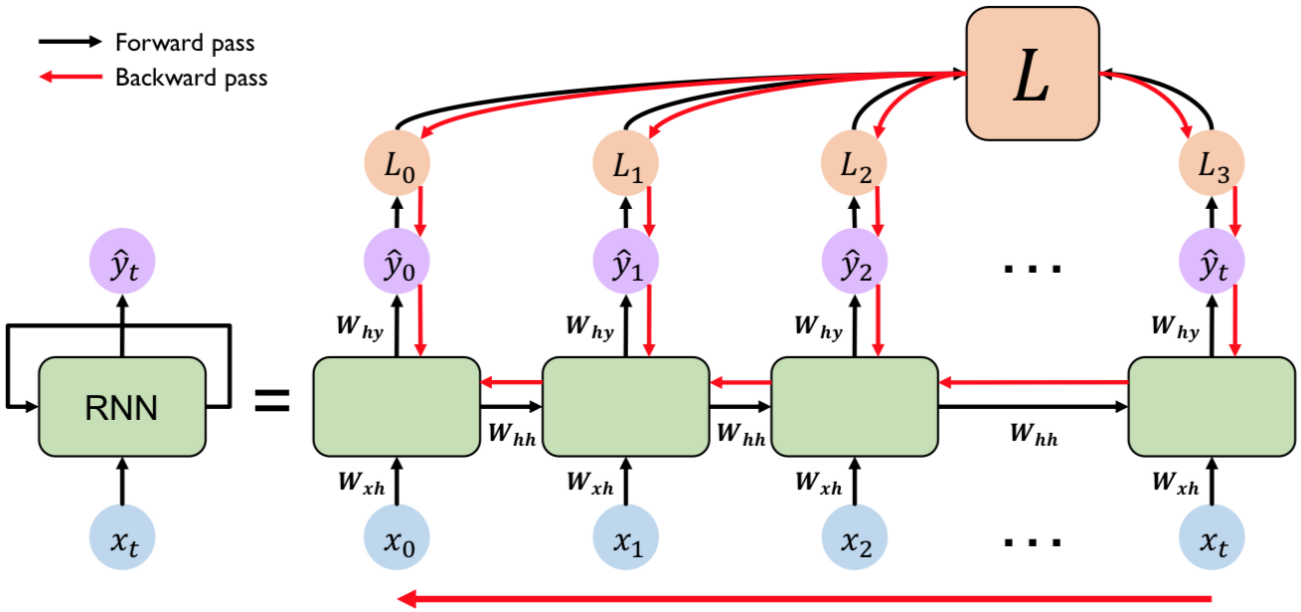
Backpropagation through time is just a fancy buzz word for backpropagation on an unrolled RNN

Unrolled RNN can lead to very deep networks $\Rightarrow$ bias to capture only short term dependencies !

[38] http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-
https://towardsdatascience.com/rnn-recurrent-neural-networks-how-to-successfully-model-seque

Recurrent Neural Network(RNN)

The RNN computational graph and information flow



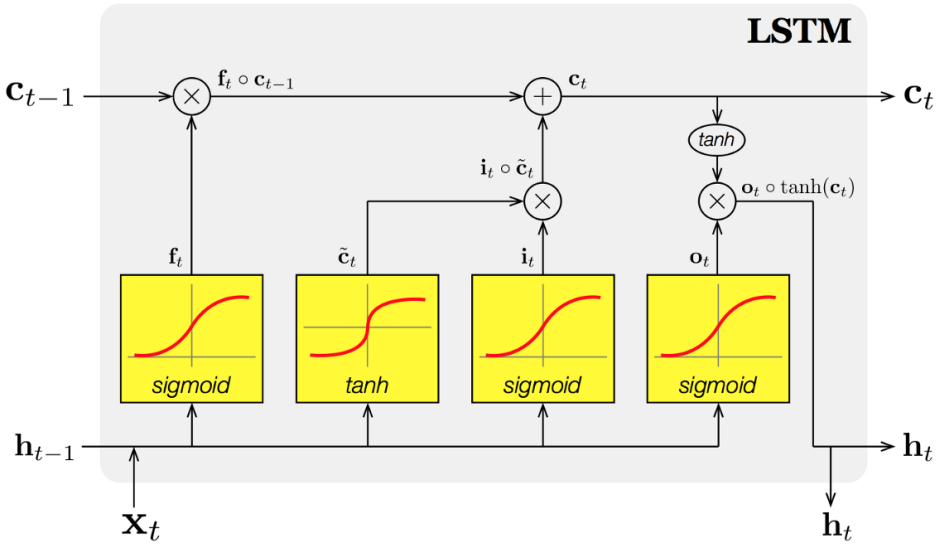
The same set of weights are used to compute the hidden state and output for all time steps (shared weights)

Long Short Term Memory(LSTM)

Long Short-Term Memory (LSTM) Network

LSTM [40] is a gated RNNs capable of learning long-term dependencies. It categorize data into **short** and **long** term, deciding its importance and what to remember or forget.

The LSTM unit cell has an **input** and a **forget** gate. The **input** defines how much of the newly computed state for the current input is accepted, while the **forget** defines how much of the previous state is accepted.



Gating variables

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_t\right)$$
$$\mathbf{i}_t = \sigma\left(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i\right)$$
$$\mathbf{o}_t = \sigma\left(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o\right)$$

Candidate (memory) cell state

$$\tilde{\mathbf{c}}_t = \tanh\left(\mathbf{W}_c[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c\right)$$

Cell & Hidden state

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tilde{\mathbf{c}}_t$$
$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t)$$

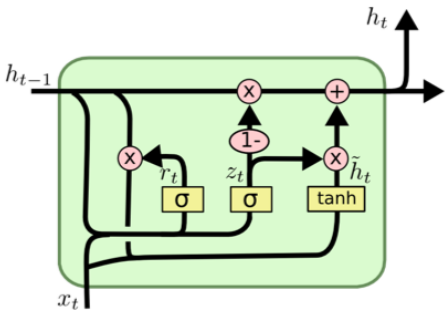
[40] http://colah.github.io/posts/2015-08-Understanding-LSTMs
https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-reme

Gated Recurrent Network (GRU)

Gated Recurrent Network (GRU)

GRU [41] networks have been proposed as a simplified version of LSTM, which also avoids the vanishing gradient problem and is even easier to train.

In a GRU the **reset** gate determines how to combine the new input with the previous memory, and the **update** gate defines how much of the previous memory is keept

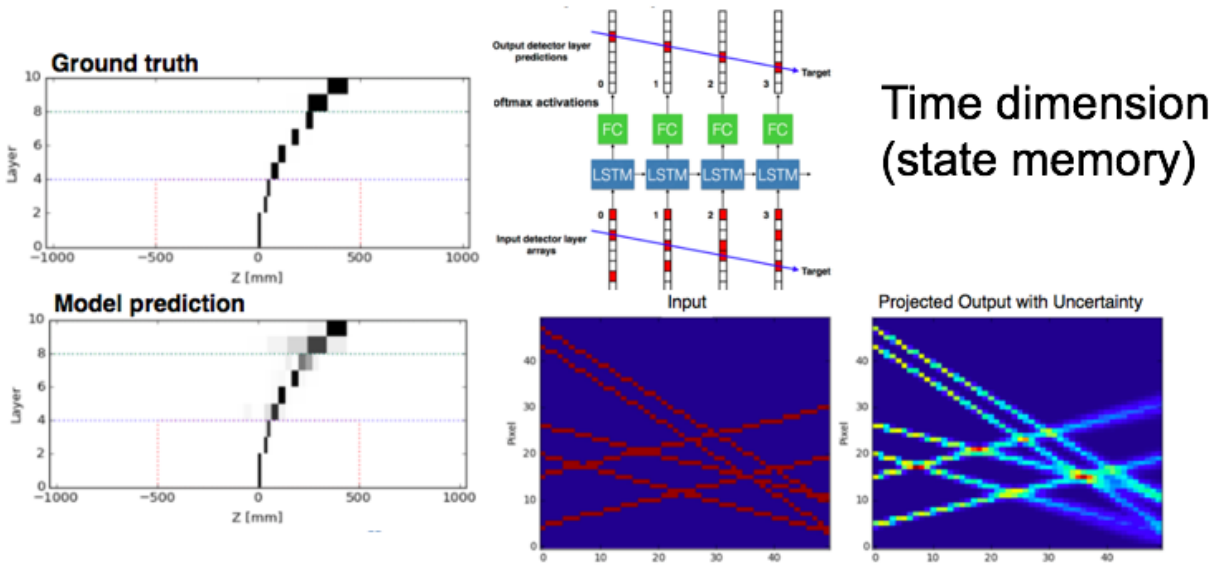


$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

[41] https://isaacchanghau.github.io/post/lstm-gru-formula

LSTM Application in HEP: Tracking

Tracking with recurrent neural networks (LSTM) [42]



[42] https://heptrkx.github.io

RNN Difficulties

RNN Difficulties

- Large inputs leads to very deep RNN networks:
  - vanishing or exploding gradient problem
  - memory limitations
  - hard to train
- Process data in a sequential way $\Rightarrow$ no parallel processing (GPU,TPU)

A solution to these problems is the attention mechanism , which allows to grab a whole data sequence in a parallel way and infer distant correlations. [43]

Attention

Attention is motivated by how we pay visual attention to different regions of an image, correlate words in setences or focus on a voice ignoring background noise. It also allow us to pursue one thought at a time and remember an individual event rather that all events.

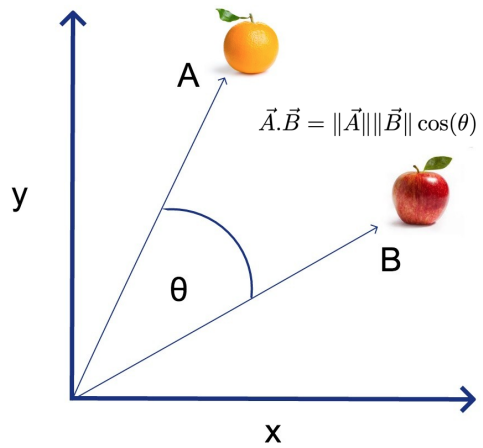
[43] https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

Attention Mechanism

Geometric similarity between vectors (points) can be estimated by the "dot product"

Attention

Attention mecanism generalizes geometric similarity by learning a metric to quantify similarity between arbitrary objects (words, images, ...) [44]



⇒ Attention can be interpreted as weights attributed to pixels in an image or words in a text, focusing on some part of the data while ignoring others. [45]

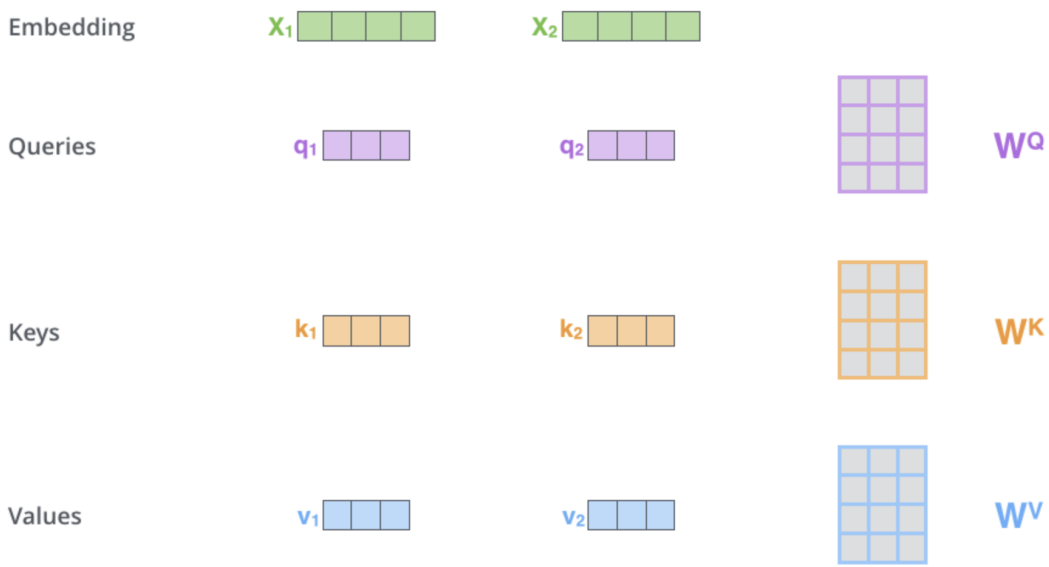
[44] A.Karpathy, Introduction to Transformers - https://www.youtube.com/watch?v=XfpMkf4rD6E
[45] C.Olah, Attention and Augmented RNN - https://distill.pub/2016/augmented-rnns/

Transformers : Attention is All you Need

The Transformer architecture is a sequential data model that uses attention based MLPs outperforming RNNs. [46]

Self-Attention

Self-attention is calculated from three vectors (Query, Key, Value) for each input vector (word embedding). They are defined by multiplying three weight matrices learned during training [47]



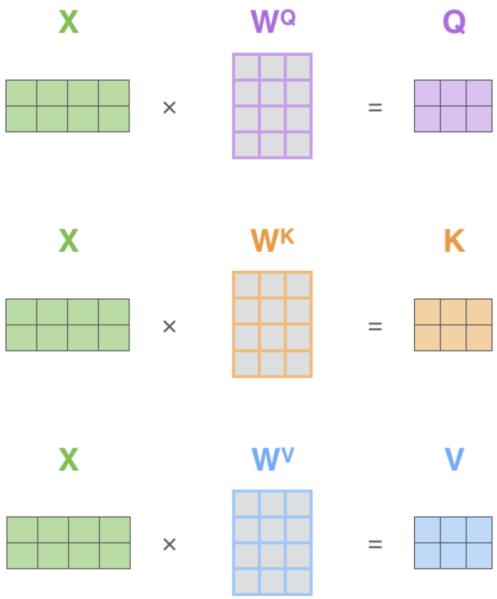
[46] A.Vaswani & all , Attention Is All you Need (2017) , https://arxiv.org/abs/1706.03762

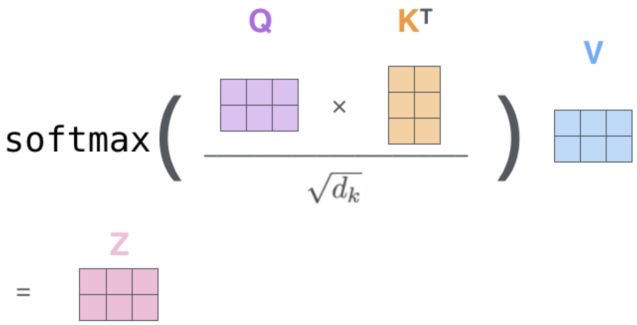
[47] https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-m

Transformer Networks

Matrix Calculation of Self-Attention

Query, Key, and Value matrices are defined by packing the embeddings into a matrix X and multiplying by weight matrices W^Q, W^K and W^V. The self-attention layer output is then softmaxed



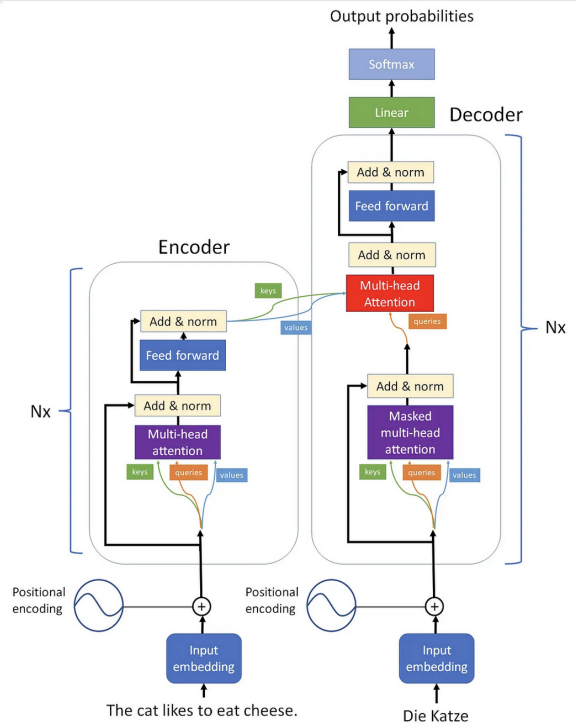


Transformers uses “multi-headed” attention where each ”head”has it’s own set of learned weigths (like CNN channels). Each head processes the inputs in parallel and independently, focusing on different aspects.

Transformer Networks

Transformer Architecture

The transformer architecture [48] has an encoder decoder struture. It uses attention and positional encoding to weight the importance of each part of the input data.



- The network has an encoder and decoder structure with independent weights.
- Self attention is used to weight the importance of each input token
- Cross attention combines keys and values from encoder (input sequence) with query from the decoder to predict the next output sequence token [49]
- Transformers can also deal with images by splitting it into patches (”image tokens”)

[48] https://towardsdatascience.com/transformers-explained-visually-part-1-overview-of-functi
[49] https://vaclavkosar.com/ml/cross-attention-in-transformer-architecture

Transformer Networks

ChatGPT

ChatGPT [50] is a generative pre-trained transformer (GPT) whose core function of a chatbot is to mimic a human conversationalist. ChatGPT is versatile and it can write and debug computer programs, compose music, fairy tales, poetry and student essays and it also answers test questions and play games.

`https://chat.openai.com/chat`

Vision Transformers (ViT)

Computing relationships for every pixel pair in a typical image is computationally prohibitive. Vit splits an image into fixed size patches and flattens it to create lower-dimensional embeddings. After including positional embeddings the sequence is input to a Transformer. ViT attains state of the art performance in image classification, segmentation object detection and etc. ViT outperform the current state-of-the-art CNNs in terms of computational efficiency and accuracy.

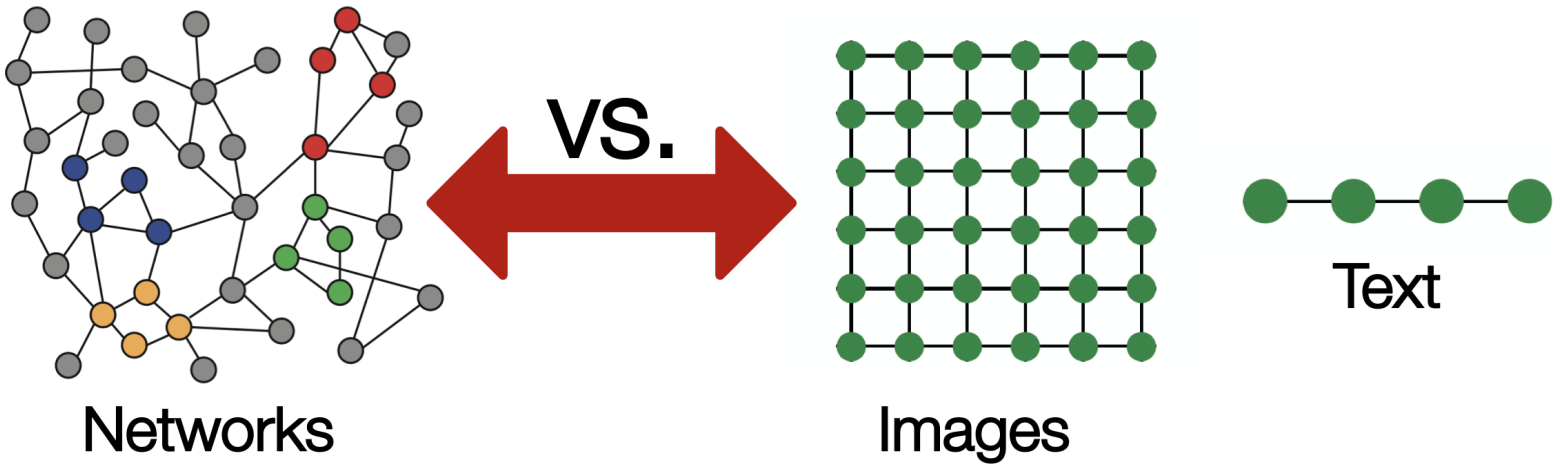
`https://viso.ai/deep-learning/vision-transformer-vit`

[50] A.Karpathy , Let's build GPT from scratch - `https://www.youtube.com/watch?v=kCc8FmEb1nY`

Graph Structured Data

Graphs

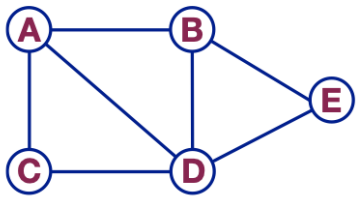
A graph is an abstract mathematical structure defined by its nodes(vertices) and edges(links): $G(N, E)$. The nodes list is unordered and can have variable length, while edges represent nodes relations. Each node and edge can have features (ex: node position, edge lengh, ...).



Sequences (linear graph) and images (regular grid graph) can be seen as special types of graph structured data !

Graph Representation

For a NN to operate on a given dataset it's necessary to encode it in a mathematical representation. A naive approach would be to represent graph structured data as an augmented adjacency matrix with node features information and use it as an MLP input. [52]



	A	B	C	D	E	Feat	
A	0	1	1	1	0	1	0
B	1	0	0	1	1	0	0
C	1	0	0	1	0	0	1
D	1	1	1	0	1	1	1
E	0	1	0	1	0	1	0

Problems:

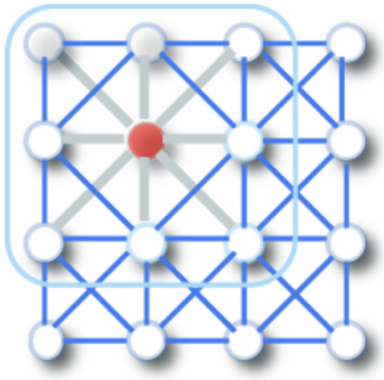
- Adjacency matrix depends on arbitrary node labeling (need permutation invariance)
- Adjacency matrix can became too large and sparse (need compact representattion)
- Matrix size depend on number of nodes (NN takes fixed input size)

[52] https://towardsdatascience.com/how-to-do-deep-learning-on-graphs-with-graph-convolutiona
https://tkipf.github.io/graph-convolutional-networks

Graph Neural Network (GNN)

From Convolution to Message Passing

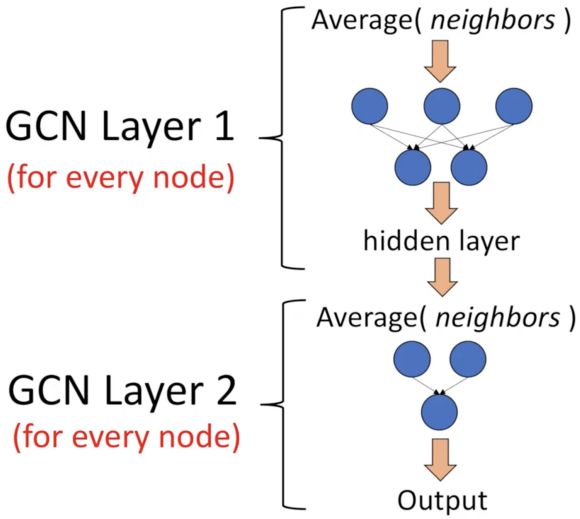
Inspired by convolution on images, which are grids of localized points, we can define a convolution on a graph where nodes have no spatial order. The node neighbourhood aggregation is often called *message passing* [53].



[53] J.Gilmer & all. , Neural Message Passing for Quantum Chemistry , https://arxiv.org/abs/1704.01212

Graph Convolutional Network (GCN)

Taking the node features as inputs the network propagates [55] it through the hidden layers getting a latent representation (embedding) of the node features. This latent representation can have an arbitrary number of dimensions, like the number of filters of a CNN.



Node Neural Network

$$h_v^0 = x_v \quad \text{(input features)}$$

$$h_v^k = \sigma\left(W_k' h_v^{k-1} + W_k \sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|} \right)$$

$$z_v = h_v^n \quad \text{(output)}$$

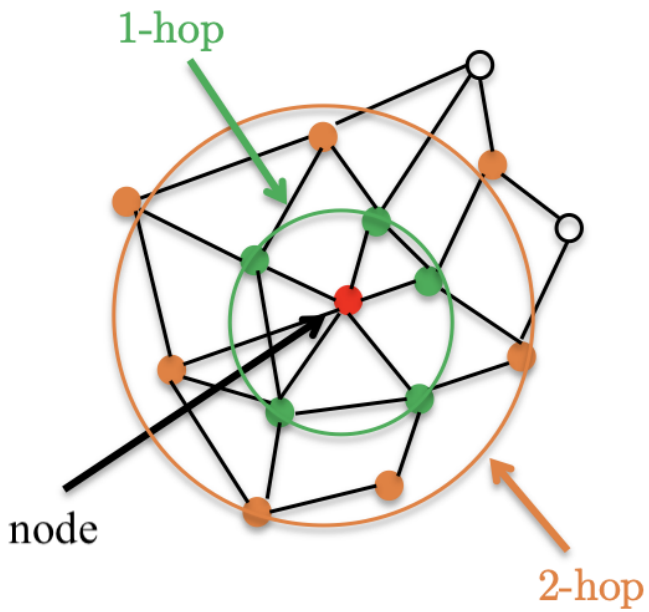
, where h_v^k is the node embedding in layer-k

[55] https://towardsdatascience.com/hands-on-graph-neural-networks-with-pytorch-pytorch-geome
http://web.stanford.edu/class/cs224w/slides/08-GNN.pdf

Graph Neural Network (GNN)

GNN Depth

GNNs with few hidden layers are enough to exhaust small graphs with a few hops .The larger the number of layers the farther away messages gets passed



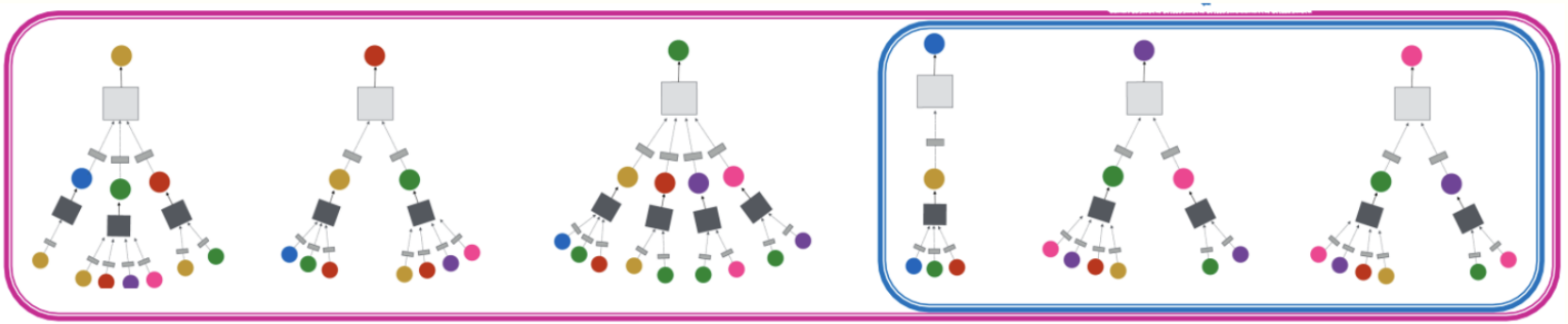
Obs: very deep GNNs tend to suffer from oversmoothing ⇒ nodes features becames identical

Graph Neural Network (GNN)

Inductive Capability

GNN can be applied to graphs with arbitrary number of nodes, since weights W_k are shared across all nodes of a given layer-k.

For an example , a model trained on graphs with nodes A,B,C , can also evaluate graphs with nodes D,E,F.

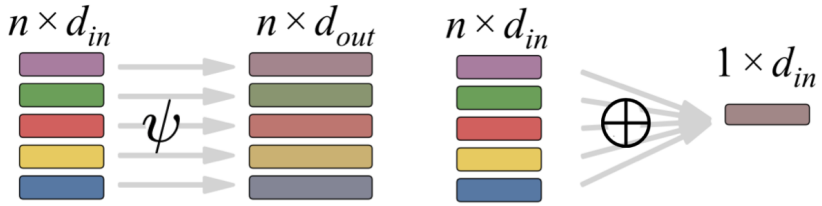


Graph Network (GN)

A general graph can have edge features as well as node features. [56]

Graph Network (GN)

The most general Graph Network(GN) can be modeled as a graph to graph mapping. An update (message passing) function ψ acts on nodes features x_i or edges features e_{ij}, followed by a permutation invariant aggregation $\bigoplus$ and an activation function ϕ.



$$h_i = \phi\left(x_i, \bigoplus_{j \in \mathcal{N}(i)} \psi(x_i, x_j, e_{ij})\right)$$

The update function takes in a *fixed size* d_{in} input and returns a *fixed size* d_{out} output, while the aggregation function takes in *variable sized* n inputs and returns a *fixed size* d_{in} output.

Obs: GN formalism is not a specific model architecture and doesn't determine what the update and aggregate functions are.

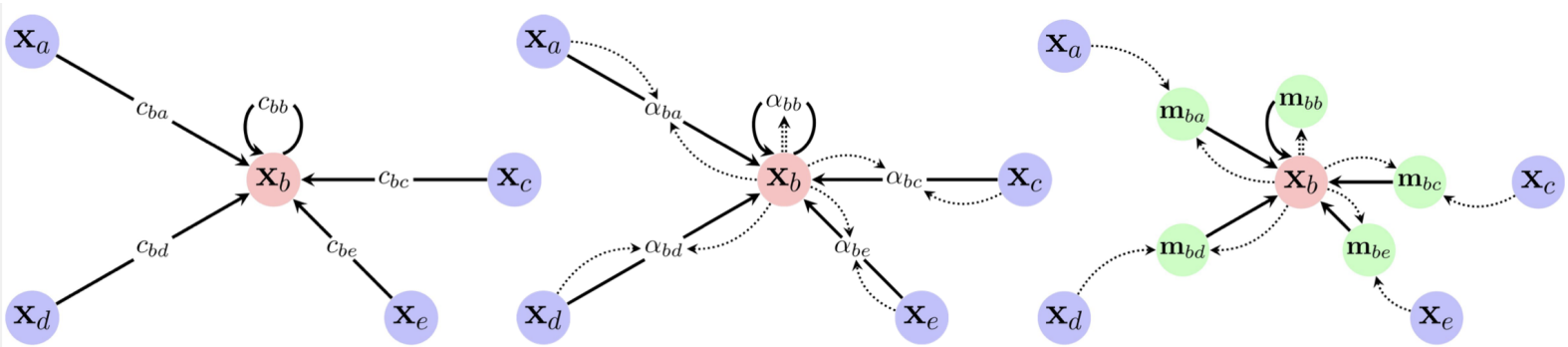
[56] https://github.com/deepmind/graph_nets
https://arxiv.org/pdf/2007.13681

Graph Neural Network

Three Particular Flavours of GNNs

GNNs can be further grouped [57] according to the following flavours:

- **Convolutional (GCN):** fixed weights coeficients c_{ij}
- **Attention Based (GAT):** learnable attention weights calculated as $\alpha_{ij} = a(x_i, x_j)$
- **Message Passing (GN):** general messages $m_{ij} = \psi(x_i, x_j)$ sent across edges



$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j\in\mathcal{N}_i} c_{ij}\psi(\mathbf{x}_j)\right) \qquad \mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j\in\mathcal{N}_i} a(\mathbf{x}_i,\mathbf{x}_j)\psi(\mathbf{x}_j)\right) \qquad \mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j\in\mathcal{N}_i} \psi(\mathbf{x}_i,\mathbf{x}_j)\right)$$

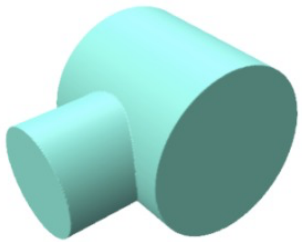
[57] https://petar-v.com/talks/GNN-Wednesday.pdf

From Graphs to Sets

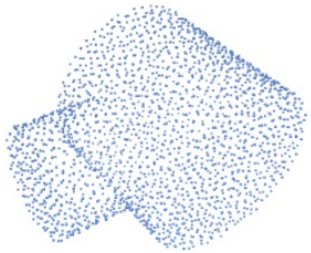
Sets

Abstracting even more on graphs we encounter sets , which are collections without intrinsic order or relations [58]

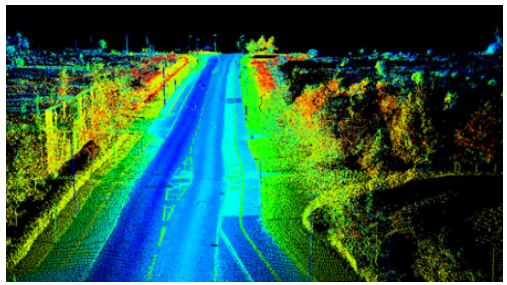
A MLP in principle can learn not to care about orderings, but training it for that would be a waste of computing resources. One can build a permutation invariant network by design, taking permutation symmetry as inductive bias in the architecture. DeepSets and PointNet are examples of networks capable of learning on point set structured data.



(a) Surface view.



(b) Point cloud.



Obs: data generated by 3D scanners and LIDAR sensors often come as point clouds with coordinates (x,y,z) as its features

[58] P.Velickovic, Graphs and Sets, https://www.youtube.com/watch?v=E_Wweuk5iqA&list=PLn2-dEmQeTfQ8YVuHBOvAhUlnIPYxkeu3&index=5&t=13s

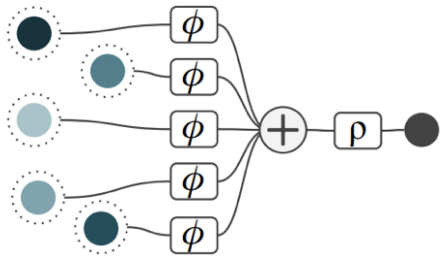
DeepSets / PointNet Networks

Consider an input vector $\vec{x}$ with N components. The set S_N of all permutations of its the input components. To construct a permutation invariant function we sum over all permutations $\pi(\vec{x})$ of $\vec{x}$. [59]

Janossy Pooling

$$f(\vec{x}) = \frac{1}{|S_n|} \sum_{\pi \in S_N} \phi\left(\pi(\vec{x})\right) \qquad \Rightarrow \text{computationally expensive and scales as N!}$$

Complexity can be reduced using permutation invariants sum of $k-tuples$, where $k < N$. For $k = 1$ we have the architectures known as DeepSets and PointNet

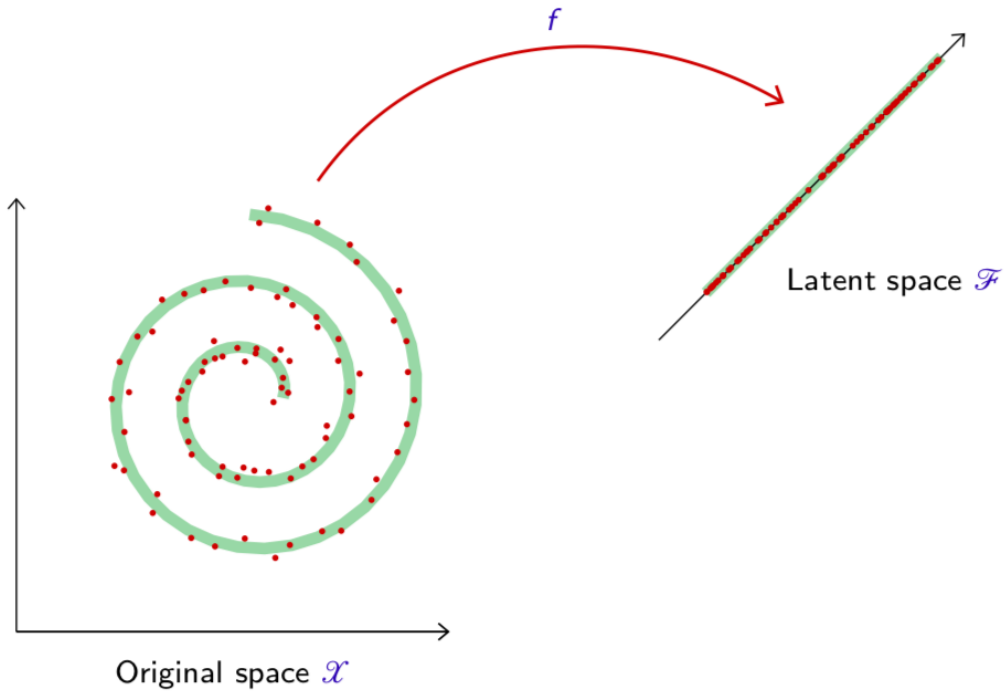


$\Rightarrow$ The sum aggregation of the ϕ layers gives permutation invariance over inputs

[59] https://fabianfuchsml.github.io/learningonsets

Autoencoder

Many applications such as data compression, denoising and data generation require to go beyond classification and regression problems. This modeling usually consists of finding “meaningful degrees of freedom”, that can describe high dimensional data in terms of a smaller dimensional representation



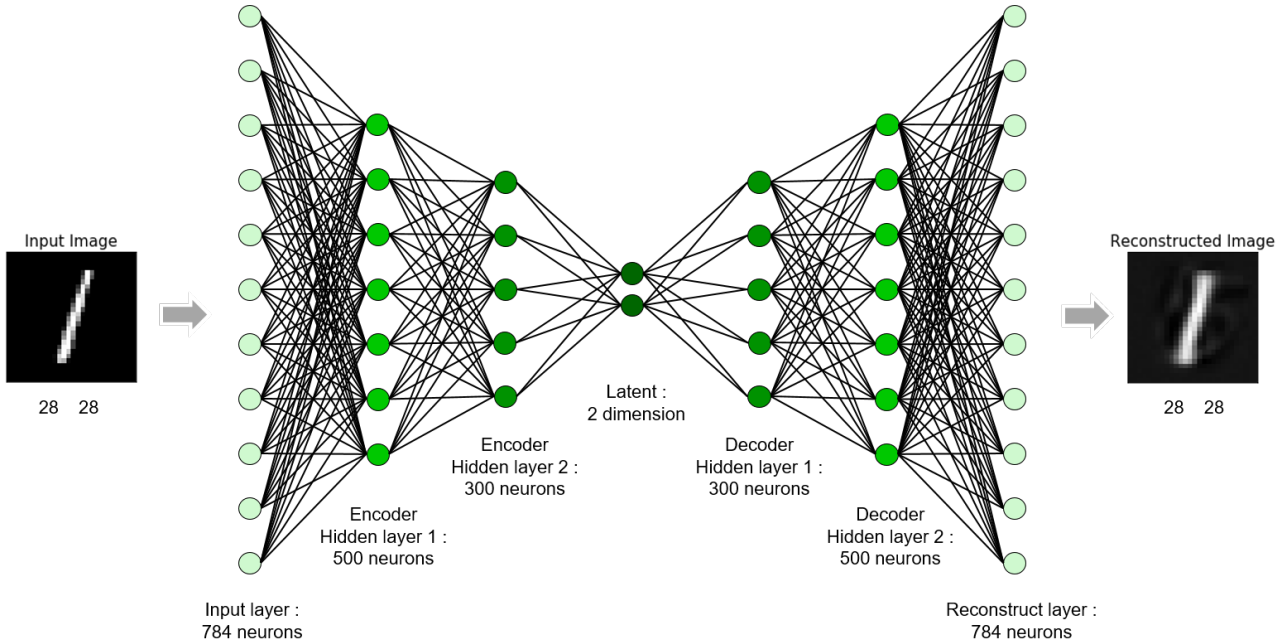
Traditionally, autoencoders were used for dimensionality reduction and denoising. Recently autoencoders are being used also in generative modeling

Autoencoder(AE)

Autoencoder(AE)

An AE is a neural network that is trained to attempt to copy its input to its output in an **self-supervised way**. In doing so, it learns a representation(encoding) of the data set features *l* in a low dimensional latent space.

It may be viewed as consisting of two parts: an encoder $l = f(x)$ and a decoder $y = g(l)$.



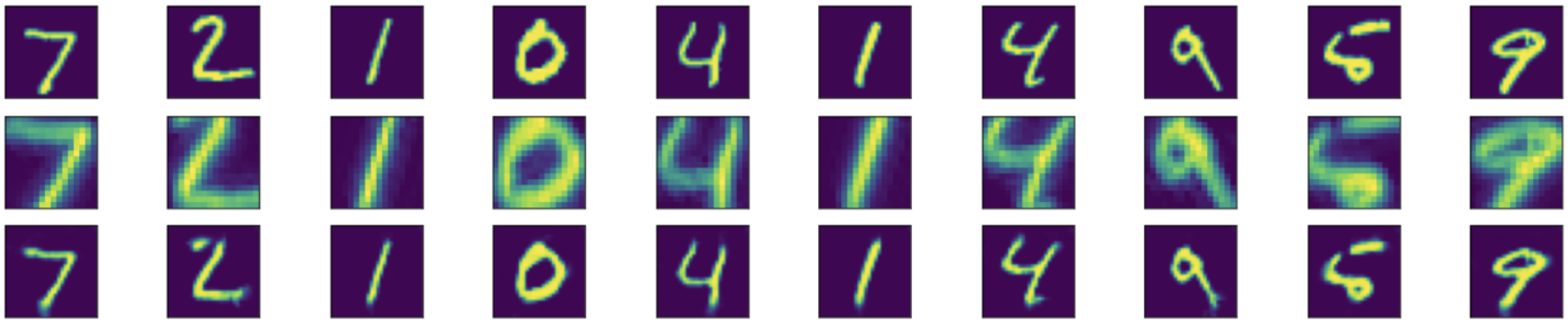
$\Rightarrow$ Understanding is data reduction

Convolutional Autoencoder(CAE)

Convolutional Autoencoder(CAE)

A CAE [62] is built out of convolutional layers assembled in a AE architecture and it's trained to attempt to reconstruct an output image from an input image after data compression. In doing so, it learns how to compress images.

Bellow we have MNIST digits (28x28 pixels) used as input and the output images and the corresponding latent space compressed images (16x16 pixels)



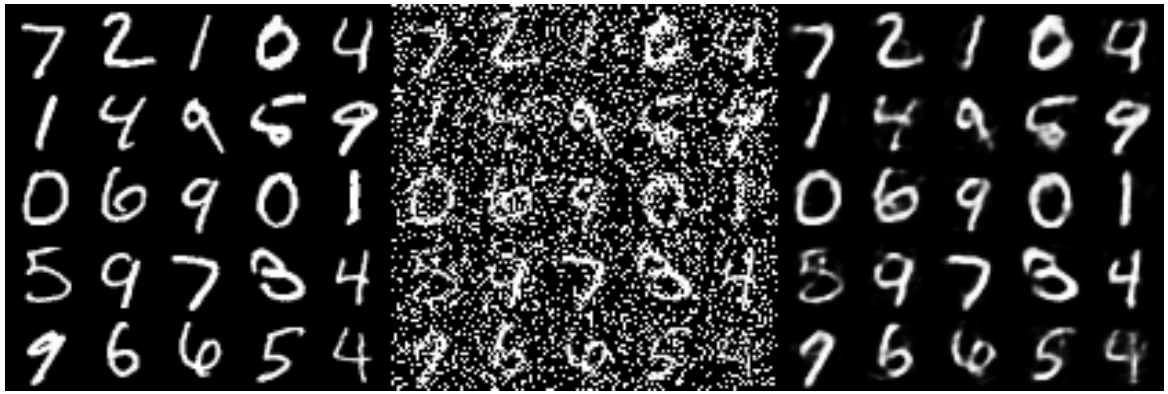
⇒ CAE learns that image borders have no information and chops the central part.

[62] https://towardsdatascience.com/introduction-to-autoencoders-b6fc3141f072

Denoising Autoencoder (DAE)

Denoising Autoencoder (DAE)

The DAE is an extension of a classical autoencoder where one corrupts the original image on purpose by adding random noise to it's input. The autoencoder is trained to reconstruct the input from a corrupted version of it and then used as a tool for noise extraction

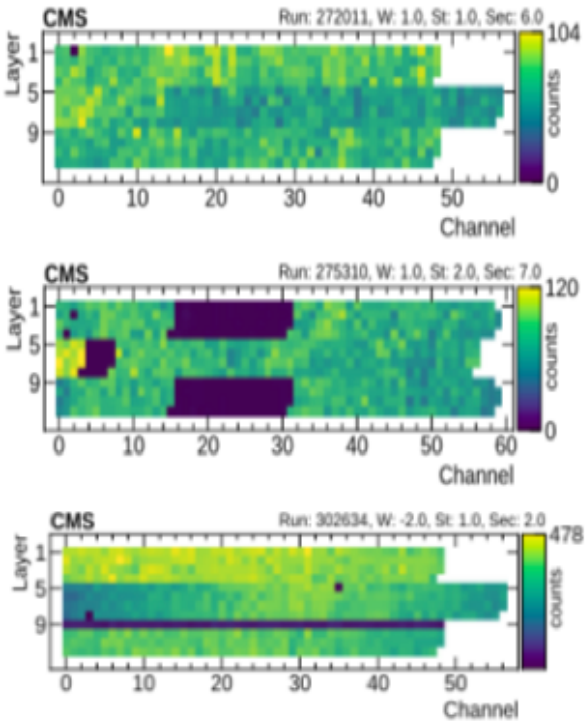


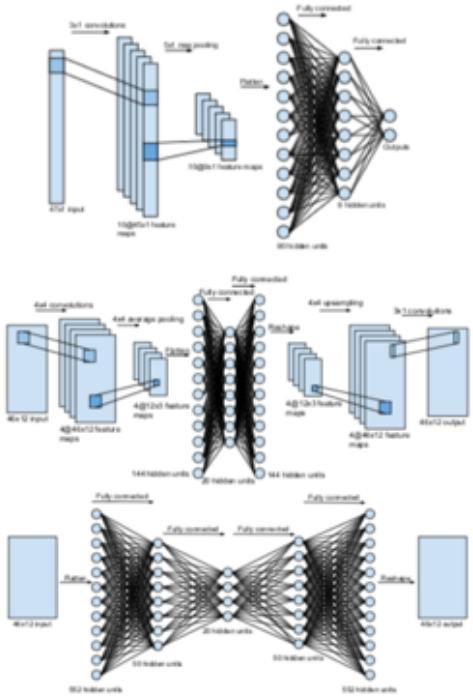
Autoencoder Application in HEP: DQM

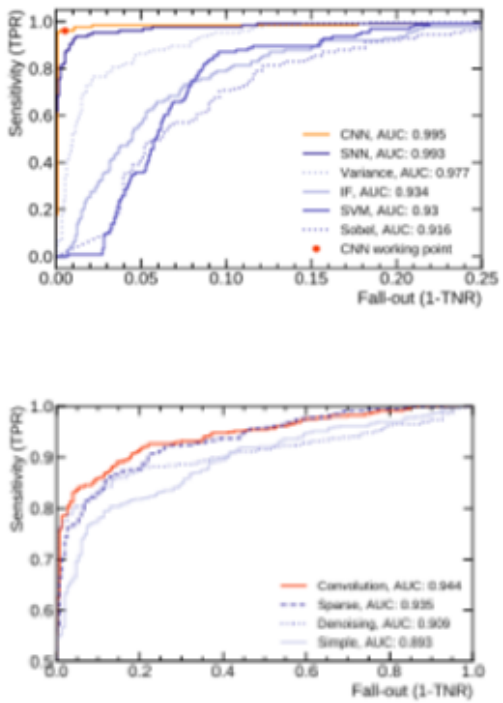
AE can be used for anomaly detection by training on a single class , so that every anomaly gives a large reconstruction error

Detector Quality Monitoring (DQM)

Monitoring the CMS data taking to spot failures (anomalies) in the detector systems





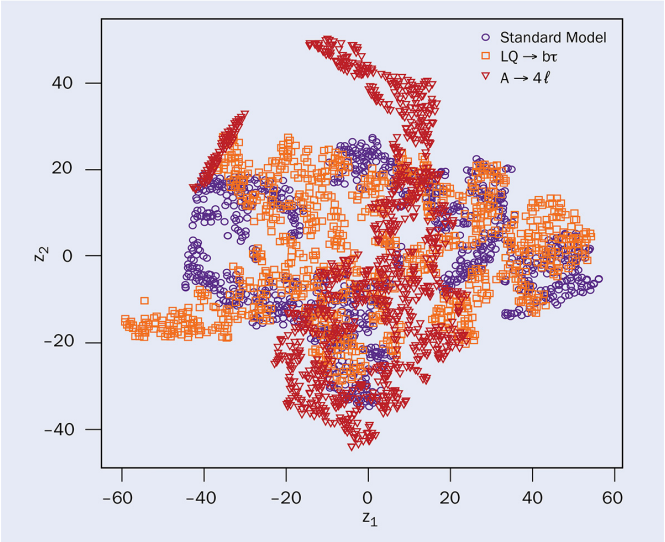


Autoencoder Application in HEP: Anomaly Detection at LHC (BSM)

AE can watch for signs of new physics at the LHC that have not yet been dreamt up by physicists [63]

Anomaly Detection (BSM Physics Search)

AE trained on a SM data sample , so that an anomalous event gives a large reconstruction error. The LHC collisions are compressed by an AE to a two-dimensional representation (z1,z2). The most anomalous events populate the outlying regions.



Outliers could go into a data stream of interesting events to be further scrutinised

[63] https://cerncourier.com/a/hunting-anomalies-with-an-ai-trigger/

Variational Autoencoder(VAE)

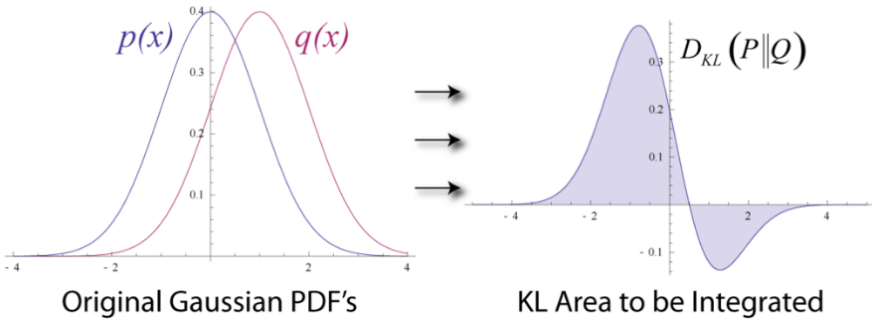
Variational Autoencoder(VAE)

A VAE [68] is an autoencoder with a loss function penalty Kullback–Leibler (KL)divergence [69] [70] that enforces a latent space representation that follows a normal distribution. To generate images with a VAE one just samples a latent vector from a unit gaussian and pass it through the decoder.

The KL divergence, or relative entropy, is a measures of how one probability distribution differs from a second, reference distribution ($\Rightarrow$ information loss)

Kullback–Leibler Divergence (Relative Entropy)

$$D_{KL}(p|q) = \int_{-\infty}^{+\infty} dx\ p(x)\ log\left(\frac{p(x)}{q(x)}\right)$$



Original Gaussian PDF's KL Area to be Integrated

[68] http://kvfrans.com/variational-autoencoders-explained
[69] https://en.wikipedia.org/wiki/Evidence_lower_bound
[70] https://www.youtube.com/watch?v=HxQ94L8n0vU

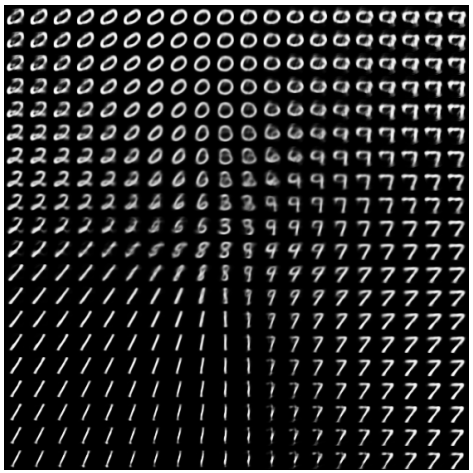
Variational Autoencoder(VAE)

VAE Loss

The VAE loss [71] is composed of a mean squared error term that measures the reconstruction accuracy and a KL divergence term that measures how close the latent variables match a gaussian.

$$L = \| x - f \circ g(x) \|^2 + D_{KL}(p(z|x)|q(z|x))$$

VAE generated numbers obtained by gaussian sampling a 2D latent space



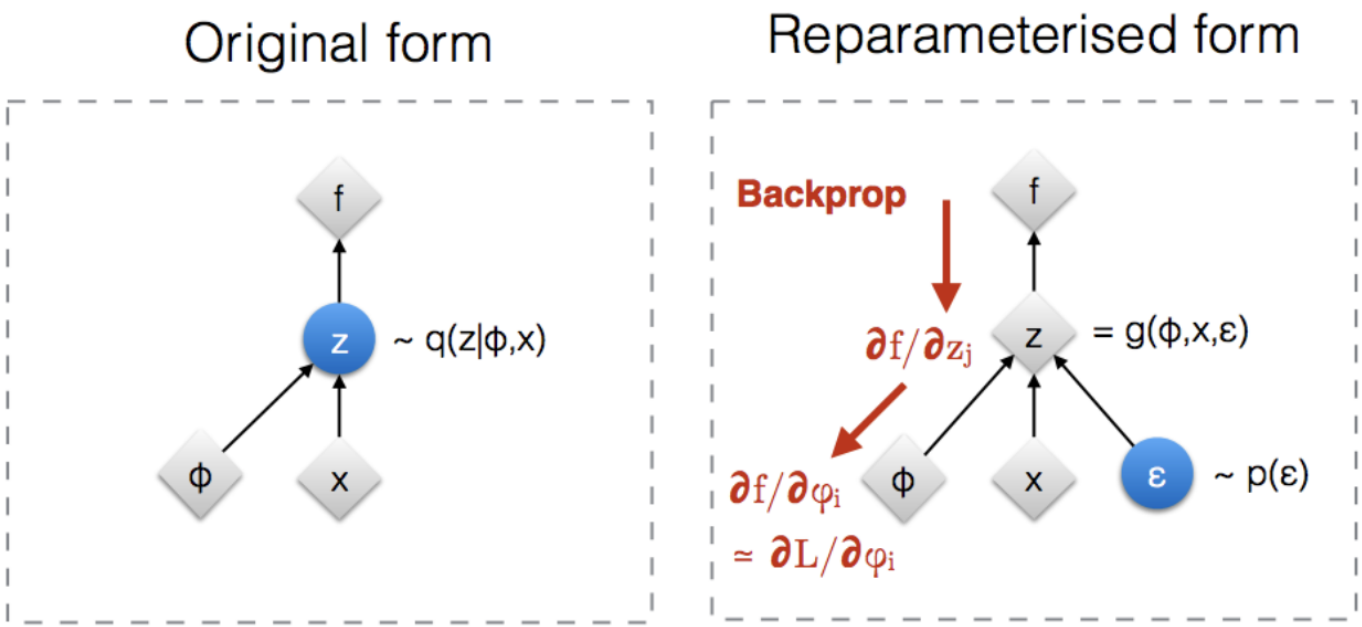
$\Rightarrow$ KL loss is a regularization term that helps learning "well organized"latent space

[71] https://tiao.io/post/tutorial-on-variational-autoencoders-with-a-concise-keras-implement

Variational Autoencoder(VAE)

VAE Reparametrization Trick

$$z = \mu + \sigma * z_c = \mu + \sigma * (\mu_c + \sigma_c * \epsilon)$$
$$= \underbrace{(\mu + \sigma * \mu_c)}_{VAE\ mean} + \underbrace{(\sigma * \sigma_c)}_{VAE\ std} * \epsilon$$



Variational Autoencoder(VAE)

Feature Disentanglement(Factorization) in VAE

Feature disentanglement is isolating the source of variation in data [72]

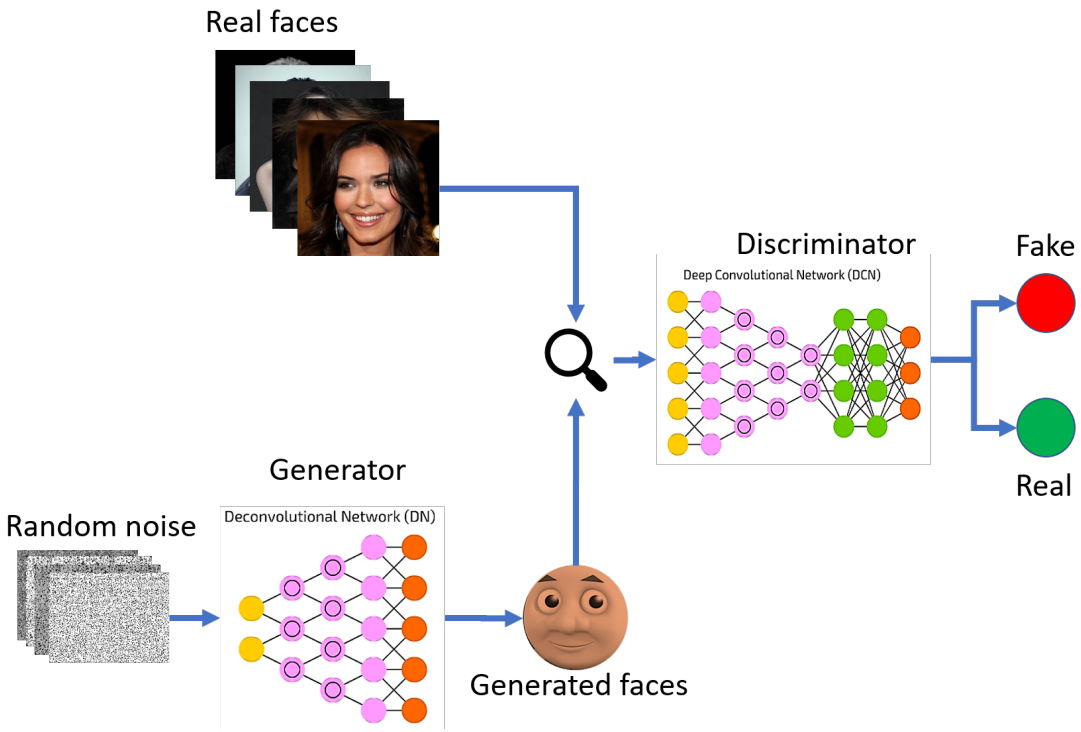
(a) Emotion (smile) (b) Gender (c) Azimuth

[72] http://proceedings.mlr.press/v97/mathieu19a/mathieu19a.pdf

Generative Adversarial Network(GAN)

Generative Adversarial Networks (GAN)

GANs are composed by two NN, where one generates candidates and the other classifies them. The generator learns a map from a latent space to data, while the classifier discriminates generated data from real data.

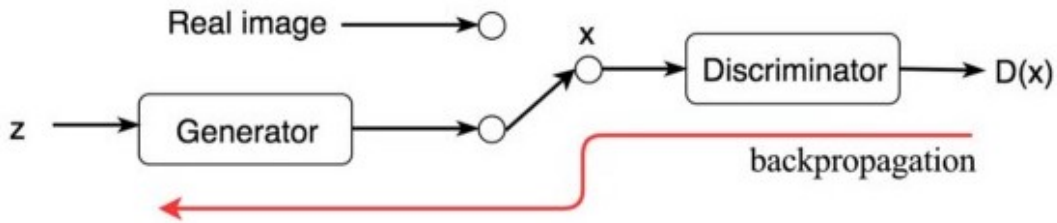


Generative Adversarial Network(GAN)

GAN adversarial training [73] works by the two neural networks competing and training each other. The generator tries to ”fool” the discriminator, while the discriminator tries to uncover the fake data.

GAN Training

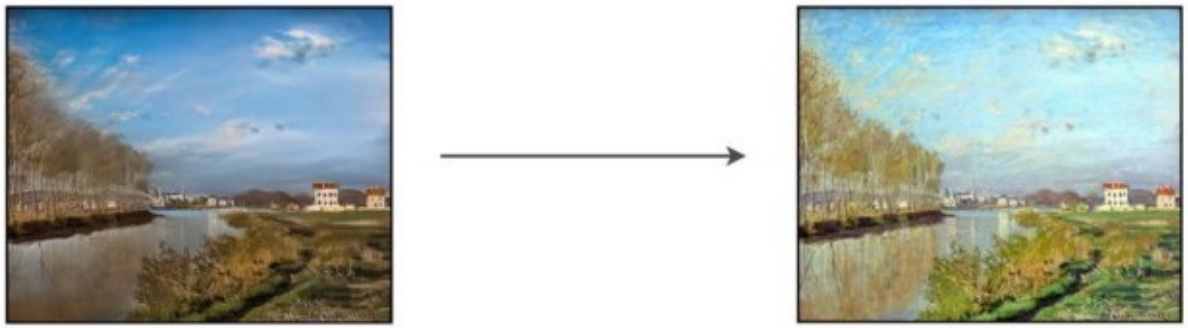
1. The discriminator receives as input samples synthesized by the generator and real data . It is trained just like a classifier, so if the input is real, we want output=1 and if it's generated, output=0
2. The generator is seeded with a randomized input that is sampled from a predefined latent space (ex: multivariate normal distribution)
3. We train the generator by backpropagating this target value all the way back to the generator
4. Both networks are trained in alternating steps and in competition



[73] https://medium.com/@jonathan_hui/gan-whats-generative-adversarial-networks-and-its-appli

Generative Adversarial Network(GAN)

GANs are quite good on faking celebrities images [75] or Monet style painitings [76] !

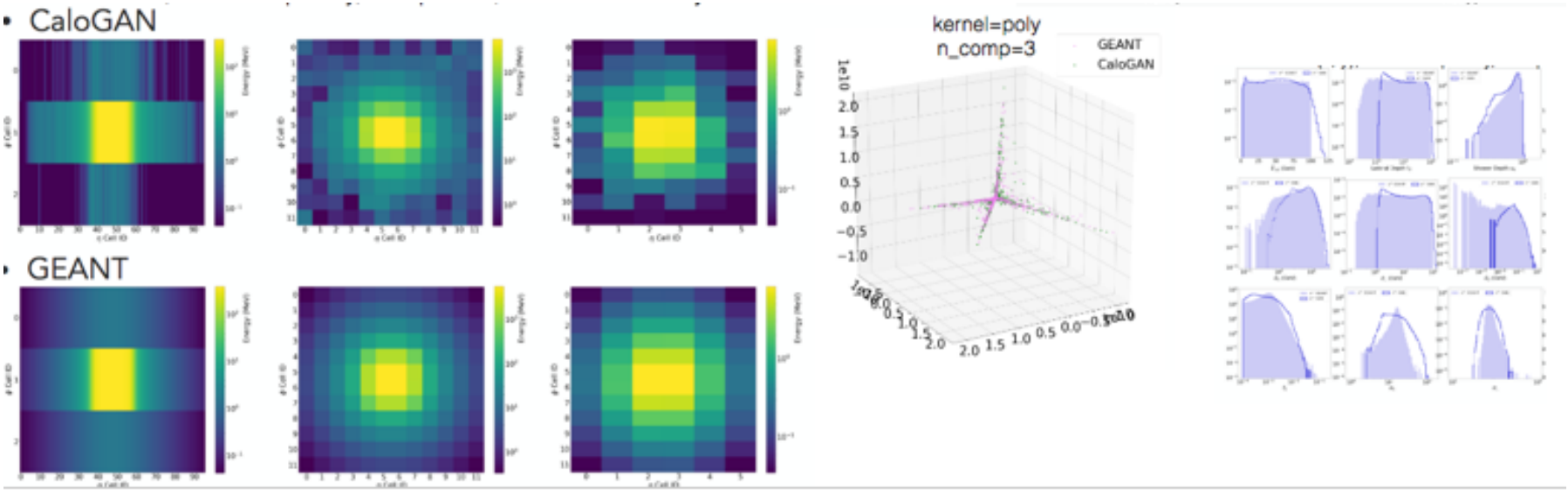


[75] https://research.nvidia.com/publication/2017-10_Progressive-Growing-of
[76] https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

GAN Application in HEP: MC Simulation

CaloGAN

Simulating 3D high energy particle showers in multi-layer electromagnetic calorimeters with a GAN [77]



[77] https://github.com/hep-lbdl/CaloGAN

Diffusion Model (DM)

A diffusion model works as a Denoising Autoencoder(DAE) trained to denoise images blurred with Gaussian noise.

Introducing Sora — OpenAI's text-to-video model

Diffusion Model (DM)

Diffusion Model

Diffusion models define a Markov chain of diffusion steps by slowly adding random noise to data and then learn to reverse the diffusion process in order to reconstruct desired data samples from the noise. [82]

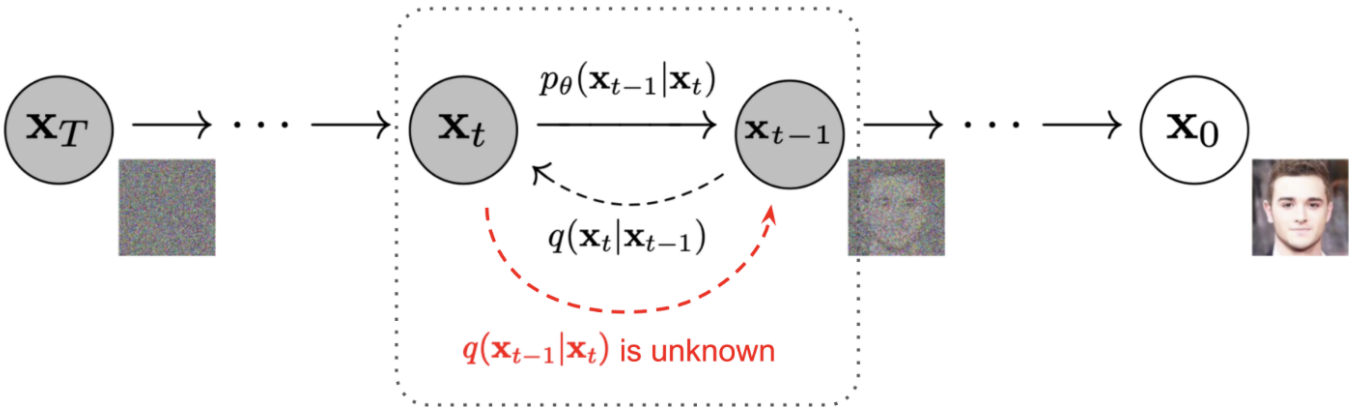


Fig. 2. The Markov chain of forward (reverse) diffusion process of generating a sample by slowly adding (removing) noise. (Image source: Ho et al. 2020 with a few additional annotations)

[82] https://lilianweng.github.io/posts/2021-07-11-diffusion-models

Diffusion Model (DM)

Forward Process

Given a data point x_0, the forward diffusion process adds small amount of Gaussian noise in steps (Markov Chain) , producing a sequence of noisy samples. The step sizes are controlled by β_t. At the end we are left with a noisy image represented by an isotropic Gaussian distribution (same variance along all dimensions).

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

$\Rightarrow$ The model latent space has the same dimensionality as the original data (high dimensionality)

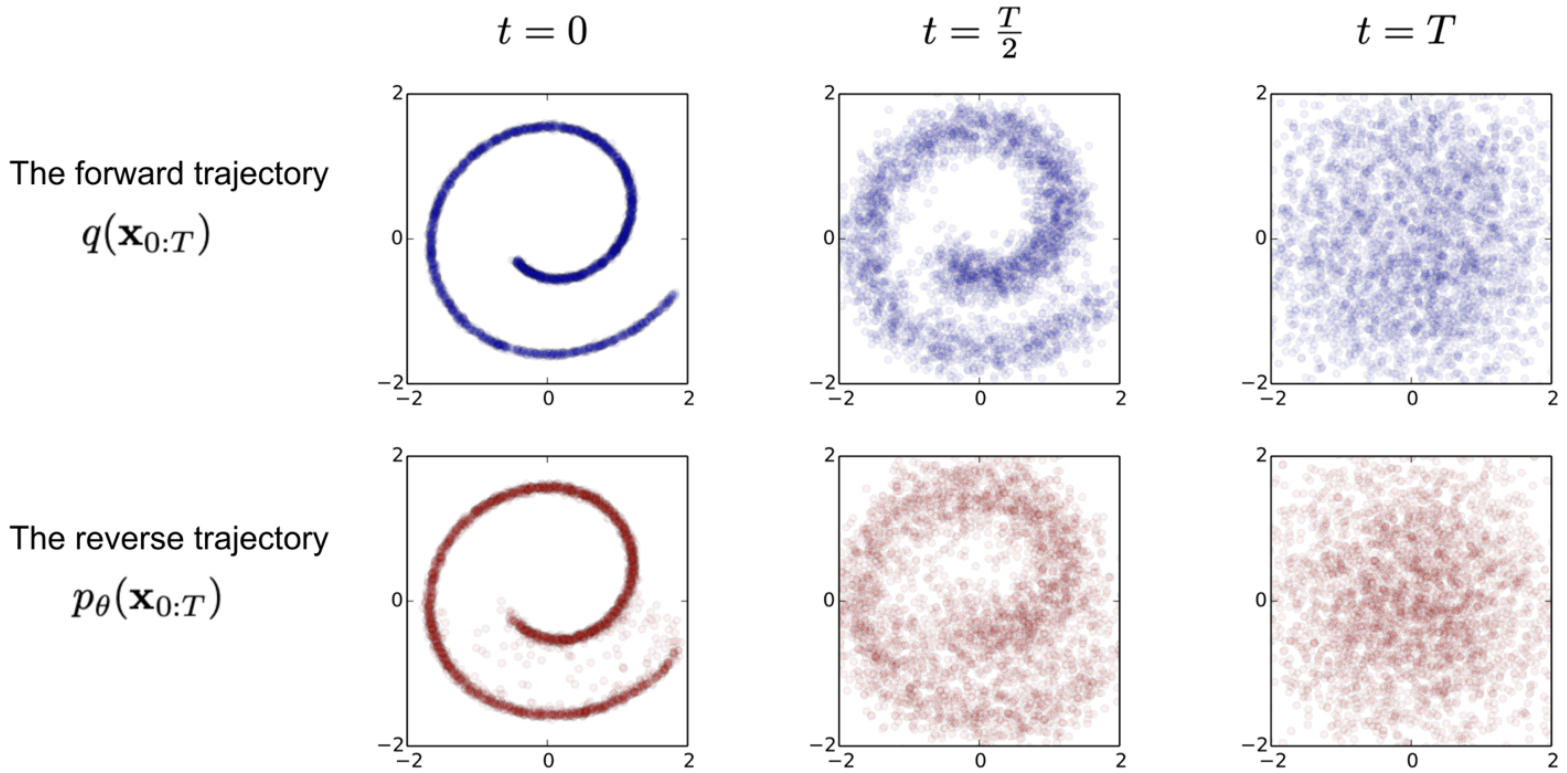
Reverse/Backward Process

The task of removing the added noise in the forward process, again in an iterative fashion, is done using a neural network. If we can reverse the above process and sample from $q(x_{t-1}|x_t)$, we can recreate the original data sample from a Gaussian noise input $X_T \simeq \mathcal{N}(0,1)$.

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

Diffusion Model (DM)

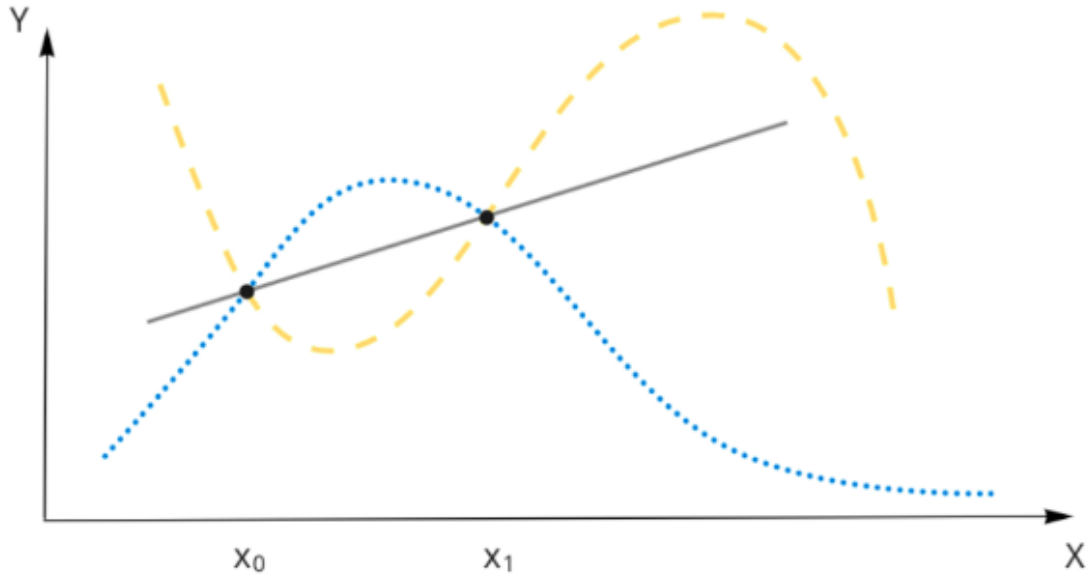
Example of training a diffusion model for modeling a 2D Swiss roll data [83]



[83] https://arxiv.org/abs/1503.03585

Domain Knowledge and Inductive Bias

As an example of how a set of observations can lead to different hypothesis depending on the inductive biases consider a fit given a a set of data points



There's an infinite number of functions that could pass through the samples data points [86]

[86] https://towardsdatascience.com/the-inductive-bias-of-ml-models-and-why-you-should-care-a
U. von Luxburg , Machine learning and inductive bias (Tubingen 2020) , https://www.youtube.com/watch?v=WB8eYZSZyaE&list=PL05umP7R6ij2XCvrRzLokX6EoHWaGA2cC&index=2&t=2372s

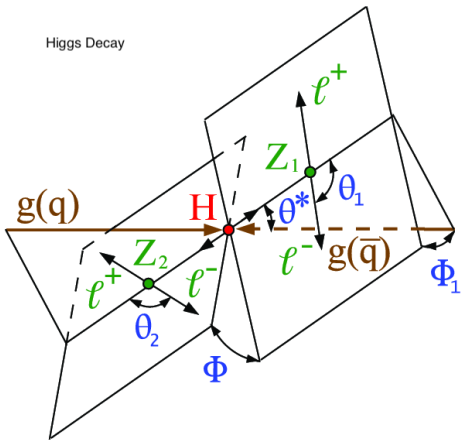
Domain Knowledge and Inductive Bias

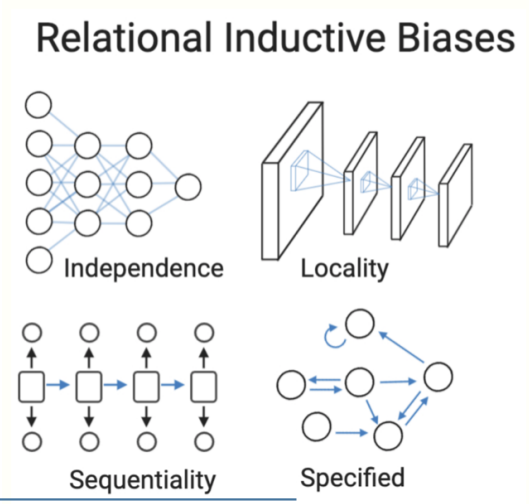
Domain knowledge can be infused into a model in different ways [87]

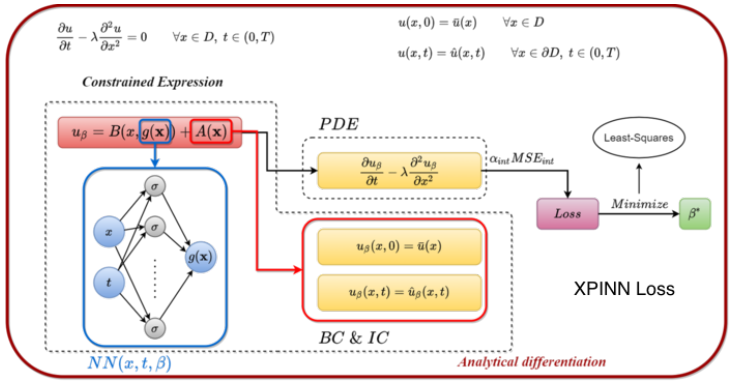
Domain Knowldge

- Using engineered variables variables as input data
- Creating specialized architecture layers: CNN, RNN, GNN, DS
- Adding a penalty term to the loss function: PINN

$\Rightarrow$ It's also possible to use simultaneous combinations of the above methods







[87] https://sgfin.github.io/2020/06/22/Induction-Intro/
https://www.nature.com/articles/s41598-021-04590-0

The End !!!