Université Paris Cité

Advanced Methods of Data Analysis

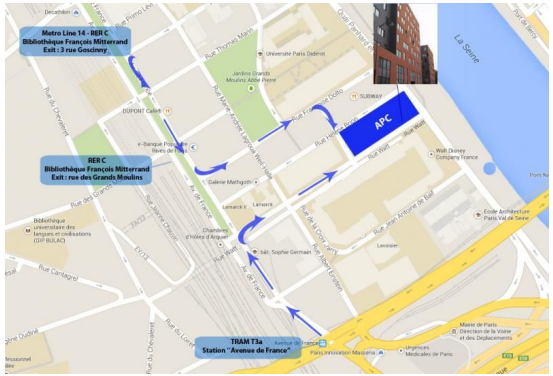
2024

–
Yvonne Becherini
Université de Paris Cité
Laboratoire Astroparticule et Cosmologie
Data Intelligence Institute of Paris

Welcome! Who am I

- My name is Yvonne Becherini
- I am an Astroparticle Physicist
- I work at University of Paris Cité, Laboratoire Astroparticule et Cosmologie & Data Intelligence Institute of Paris
- My principal interests are
  - Astroparticle Physics
  - Gamma-Ray & Neutrino Astronomy
  - Extra-galactic sources
- I develop Data Analysis methods with Machine Learning

Campus des Grands Moulins



Applied Data Analytics Today

Today's Data

- Fine-granularity of observations/measurements in real-world phenomena
- Complex (high-dimensional, heterogeneous, ...)
- Large scale (up to Terabytes for every single experiment)

That need to be analysed

- Data-driven scientific discovery
- Refine existing/build new models to explain real-world phenomena
- Analysis becomes complex because of data size/complexity

- New data analysis/intelligence techniques/tools are needed

Machine Learning

- Machine Learning helps you to set up analysis strategies in a fast way with already-existing algorithms
  ➔ provided that you have control on what is being done
- So, we will try to use Machine Learning from the point of view of applied research

Data Science

- Is a very fast developing topic,
  helped also through the scientific needs proposed by use cases (Biology, Medicine etc)
- An algorithm developed in Biology, might at some point be useful for another domain

Astrophysical data is characterized by large volumes and high complexity, often involving multi-dimensional datasets that challenge traditional analysis methods.

AI plays an important role in efficiently processing and analyzing this data, enabling researchers to identify patterns and anomalies that may not be apparent through conventional techniques.

AI has the ability to handle uncertainty and variability which are essential tools for extracting meaningful insights from astrophysical phenomena.

Time Series Data

- Light curves from variable stars or exoplanet transits
- Radio signals (e.g., pulsar timing)

Catalogues, Sky Surveys

Image Data

- High-resolution images from telescopes (Hubble, VLT, etc.)
- Multi-wavelength observations (X-ray, UV, infrared)

High-Energy Physics Data (raw, calibrated, high-level)

- Gamma-ray signatures in data (HESS)
- Neutrino signatures in data (KM3Net)

Machine Learning versus Programming. And Unsupervised Learning

Programming

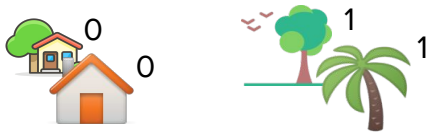
- You collect a bunch of data, you apply some known rules, and you turn that set of data and rules into the results

Supervised Machine Learning

- We have the data and the results (the labels) and we input these into an ML model that produces the rules that we want for the programming

Unsupervised Learning

- We do not have rules nor labels in input, so here we only have the unlabelled data
- We want to output something about the structure of the data (how data cluster, how dense are the structures, or, we just want to reduce the dimensionality of data)



Supervised Learning

A type of machine learning where models are trained on labeled data. Plenty of models to perform classification or regression: Random Forests, Boosted Decision Trees, Neural Networks, etc

Addressing uncertainty in Data

AI models can quantify uncertainty

Probabilistic models incorporate uncertainty into predictions effectively.

Bayesian methods provide a framework for updating beliefs with new data.

Unsupervised Learning

This branch deals with unlabeled data, aiming to identify patterns or groupings. Typical algorithms include k-Means clustering and Hierarchical Clustering. Often used for

- Visualization purposes, for clustering and for subsequent labelling of items
- Generative modelling
- Dimensionality Reduction methods simplify data without losing critical information.

Examples of AI in Astrophysics

- **Supervised Learning**
  - Classification of galaxy morphologies using labeled image datasets
  - Convolutional Neural Networks (CNNs) to identify features like spiral arms or elliptical shapes
  - Pulsar timing and light curve prediction
  - Identifying periodic phenomena in massive time series datasets
  - RNNs and LSTMs are ideal for understanding sequences and temporal dependencies in time series data.

- **Unsupervised Learning**
  - Clustering of star clusters or galaxies in large datasets
  - K-Means or DBSCAN for detecting hidden structures in cosmic maps
  - Example: Identification of gravitational lensing phenomena

Semi-supervised learning (SSL)

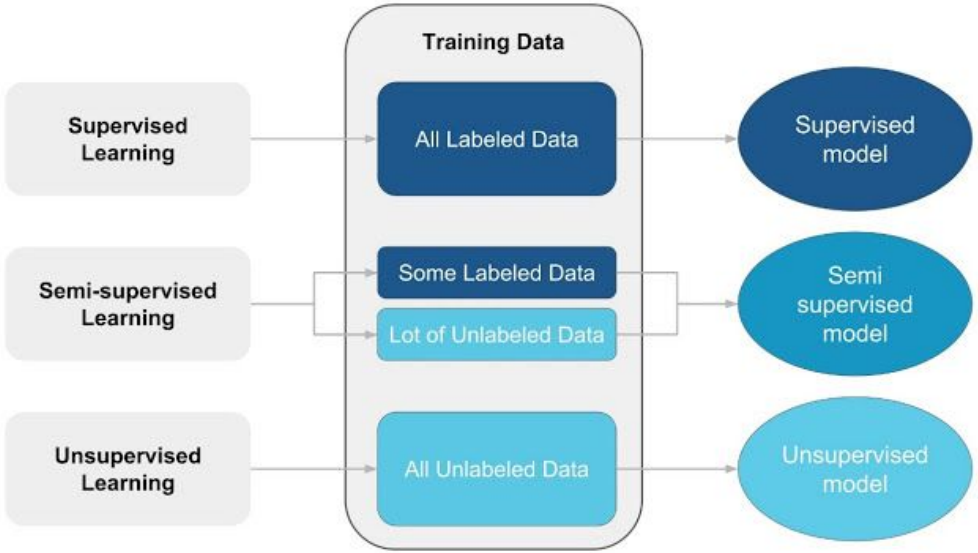
Supervised learning requires a lot of labelled data that in many cases we just don't have.

Semi-supervised learning (SSL) is a different method that combines unsupervised and supervised learning to take advantage of the strengths of each.

SSL uses a combination of supervised and unsupervised learning techniques, where we have a combination of both labelled and unlabelled data

Suppose you have a large dataset with unlabelled data, and labelling one by one the items would not be practical:

- We could just label a small portion of the data and then use this sample to train our model
- Use this model to pseudo-label the unlabelled data
- Then re-train the model on all labelled +pseudo-labelled data



In Supervised Machine Learning, we need:

1) Input data points

   Tables, catalogues, time series, images, sound files, videos

2) Examples of the expected output

   What the data represent, i.e. “the labels”

3) A way to measure whether the algorithm is doing a good job

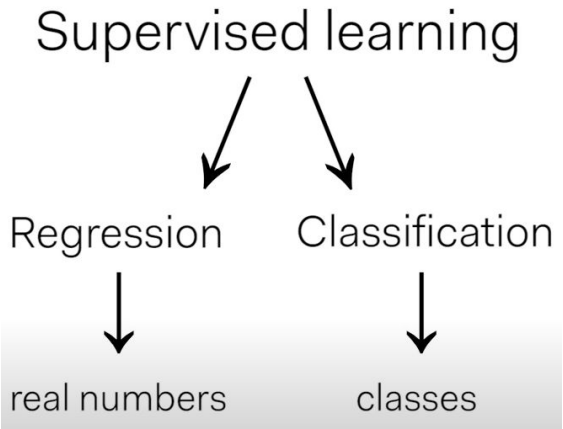
   This is necessary in order to determine the distance between the algorithm’s current output and its expected output. The measurement is used as a feedback signal to adjust the way the algorithm works. This adjustment step is what we call learning.

The central problem in machine learning and deep learning is to meaningfully transform data: in other words, to learn useful representations of the input data that get us closer to the expected output.

Deep Learning focuses on learning successive layers of increasingly meaningful representations

Different possible goals:

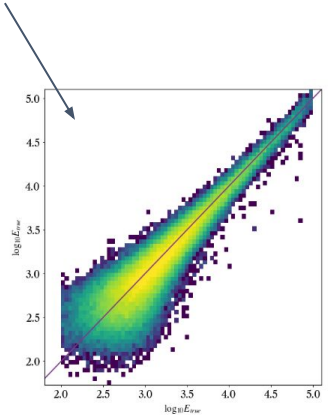
- Classification of data samples for:
  Signal extraction (Binary classification)
  Multi-class separation
- Reconstruction of properties on independent datasets (Scalar regression)
- Generate new data, learn representations



Supervised Learning has two main branches:

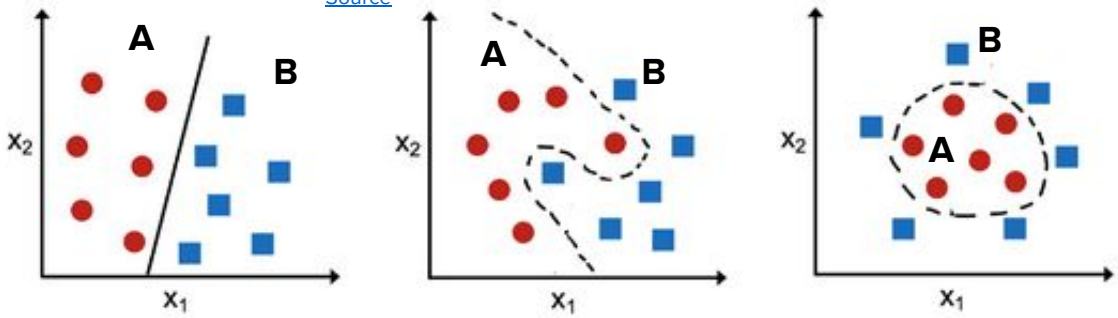
We can solve Classification or Regression problems.

- In Classification, the output of the model is a label denoting a Class or a Category.
- In Regression, the output of the model is a real number



Supervised Learning: Binary Classification

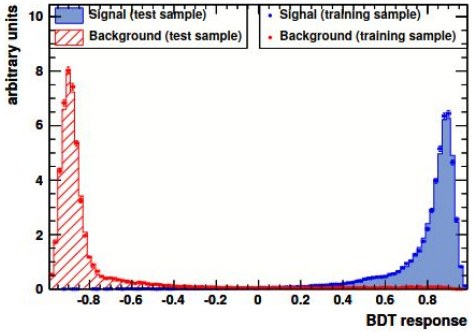
Source



- You wish to classify your data in subsamples (A, B), aka Binary classification
- Machine Learning is able to learn non-linear relations in data
- Classes are mutually exclusive
  ➜ Sum of all predicted probabilities is 1

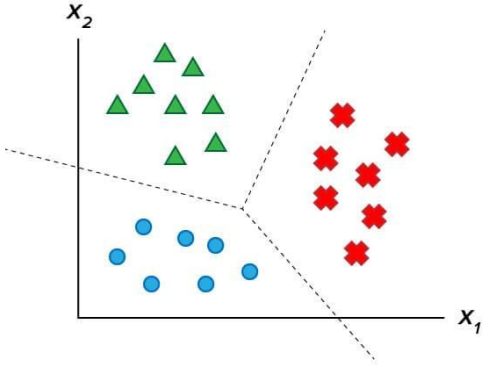
Examples:

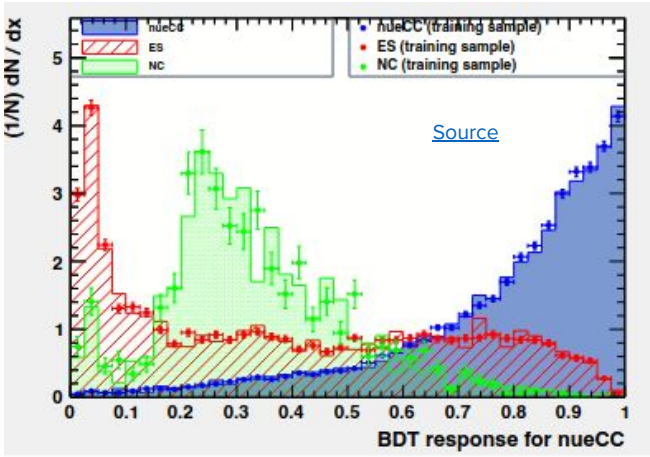
- Healthy food versus junk food
- Spam emails versus non-spam emails
- Gamma-rays versus protons
- Credit card fraud detection
- Twitter sentiment analysis (Happy versus Sad Tweets)
- Spiral versus elliptical galaxies



Y. Becherini *et al.*, 2011
(Astrop. Phys., Vol 34, 12, 2011, 858-870)

Supervised Learning: Multiclass Classification





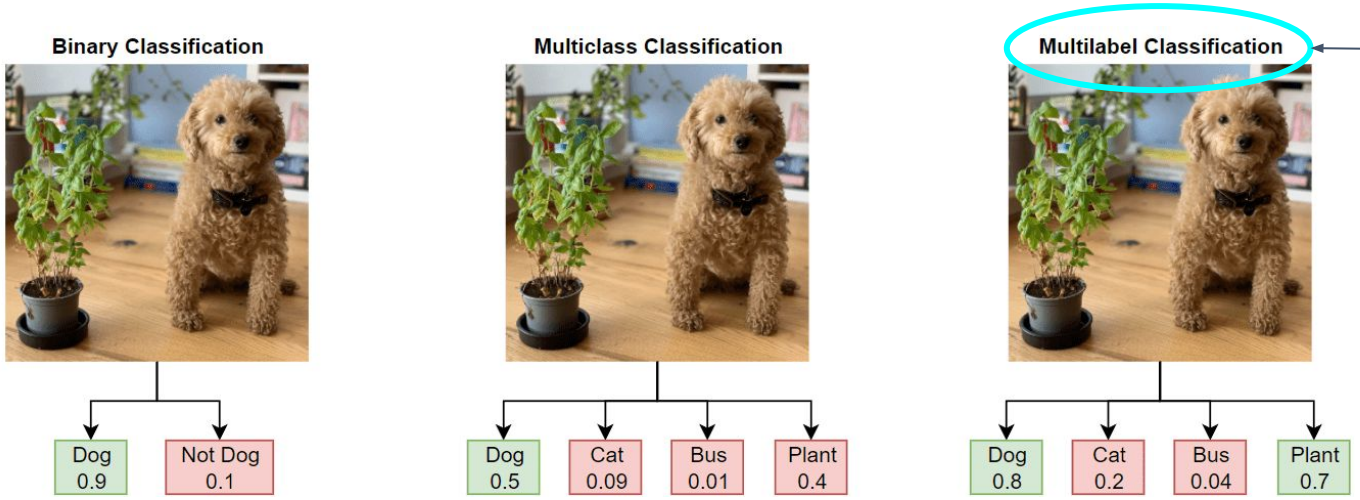
- You wish to classify your data in subsamples (A, B, C) or more, aka Multiclass classification

- Classes are mutually exclusive
  ➔ Sum of all predicted probabilities must be 1

Examples:

- Recognize digits (0, ..., 9)
- Recognize characters (A, ..., Z)
- Classify images of animals, fruits, buildings etc

Supervised Learning: Multi-label Classification



Mathworks

- Multiple labels assigned to each instance of data
- Treats each label as its own unique binary classification
- For every label, the predicted probability value must be between 0 and 1
  ➜ if the predicted probability is greater than 0.5 the label is selected.
- The probability assigned to class A has no impact on the probability of any other classes.

Examples of multi-label classification:

- For movies: horror, romance, adventure, action or horror+action, romance+adventure, romance+adventure+action, etc.
- For images (see figure)
- For research articles with multiple tags: computing+social sciences, computing+humanities, physics+mathematics, machine-learning+biology

Supervised Learning: Regression

A regression problem is when the output variable is a real continuous value.

Many different models can be used, the simplest is the linear regression, where the model tries to fit data with the best hyper-plane which goes through the points.

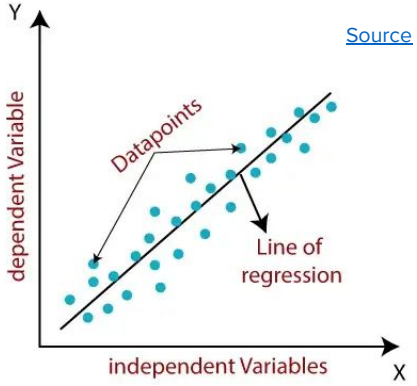
You can regress one single output, or multiple outputs at the same time

Examples of single output regression:

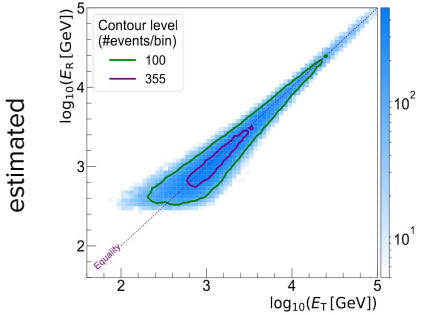
- Predicting the age of a person
- Predicting house prices
- Predicting the Energy of a particle
- Predicting the degree of severity of a disease

Example of a multiple output regression:

- Predicting degree of insulin resistance and age of onset of Type-II diabetes in adults



Source



Y. Becherini

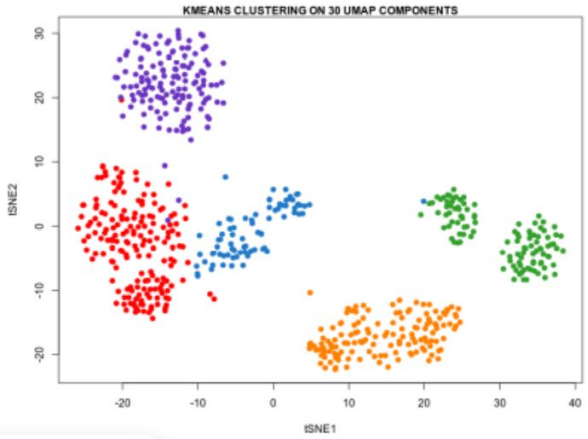
M. Senniappan, Y. Becherini et al, JINST (2021)

Unsupervised Learning

The system tries to learn with no supervision (unlabelled data)

Used for:

- Clustering
- Visualization and dimensionality reduction
- Dimensionality reduction & Clustering
- Anomaly Detection and novelty detection



Generative Models

The system learns dense representations of the input data, called latent representation

Autoencoders

- Learns to efficiently construct dense representations of the input data, called latent representation, useful for Dimensionality Reduction and Visualization purposes
- Acts as a feature detector
- Can generate new data that looks very similar to the input data

Generative Adversarial Networks (GANs)

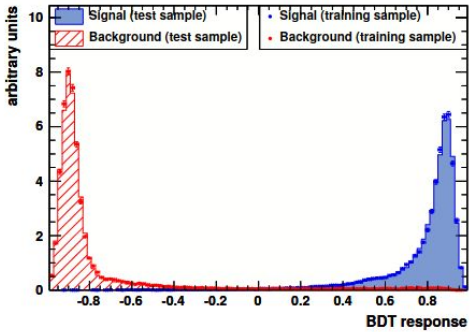
- Can very efficiently generate new data by using two neural networks
  - A generator which tries to generate data that looks similar to the training data and
  - A discriminator that tries to tell real data from fake data

Science: Astrophysics

- Machine Learning is widely used in Astrophysics and Particle/Astroparticle Physics
- Several regression, classification and generative applications have been implemented successfully
- In Astroparticle Physics:
  - We use it to reconstruct the properties of the incoming “events” caused by particles coming from the Universe
  - To classify different phenomena
  - To generate more synthetic data samples

In Gamma-Ray Astronomy:

- Big success of classification procedures which led and lead to better detection sensitivities
- More astrophysical sources discovered and studied
- Our knowledge of the energetic Universe increased
- We can reach classification procedures of 95% signal & 1% of remaining background



Y. Becherini

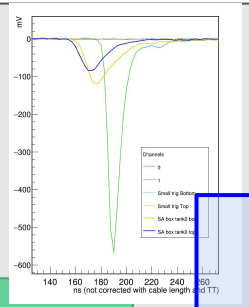
Preparing an analysis strategy in Astroparticle Physics (with ML)

Signal Monte Carlo

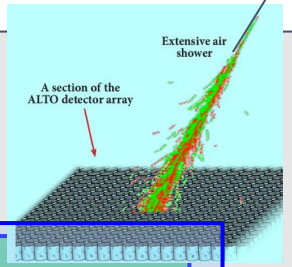
Unsupervised Learning

Background Monte Carlo

Background Real Data



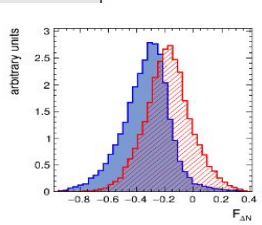
Calibration



Regression

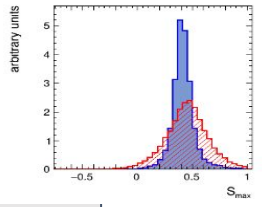
Reconstruction of arrival direction, Energy, etc for all events

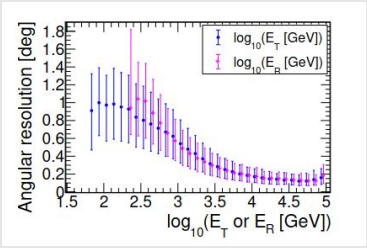
Supervised Learning



Classification

Signal over Background Discrimination
i.e. Fix the analysis cuts



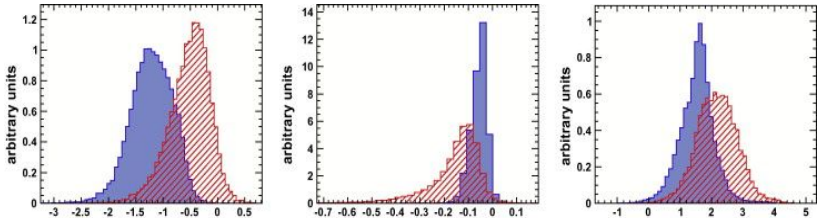


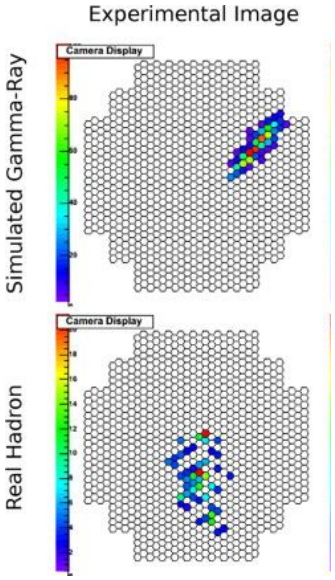
Instrument Response Functions,
Angular & Energy resolutions,
Effective areas

Preparation

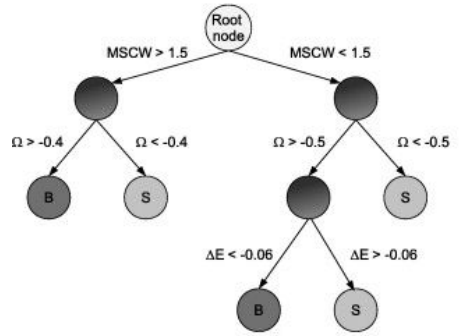
Gamma/Proton classification in H.E.S.S. data

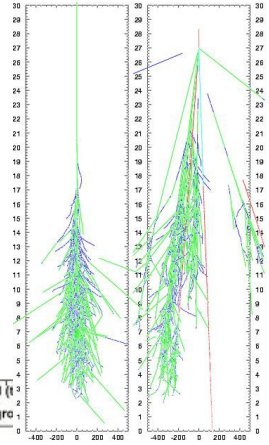
- Supervised feature-based ML
- Classification of gamma-rays and protons using a set of user-defined input variables
- The algorithm performing the separation is the Boosted Decision Trees method.

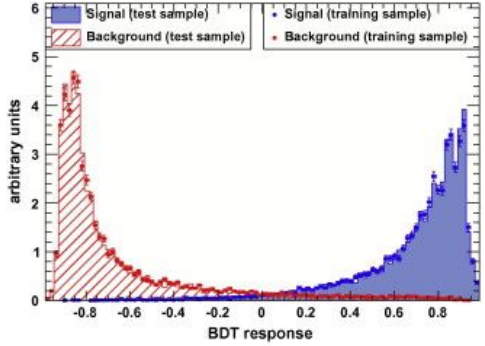


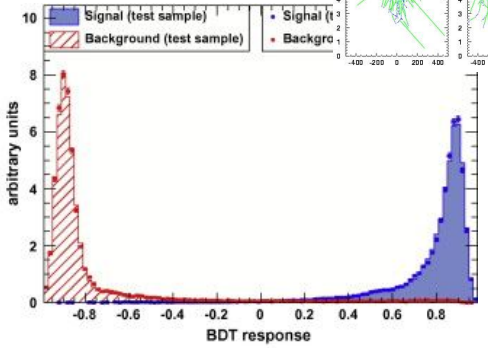


[“A new analysis strategy for detection of faint gamma-ray sources with Imaging Atmospheric Cherenkov Telescopes”, Astroparticle Physics, (2011)]



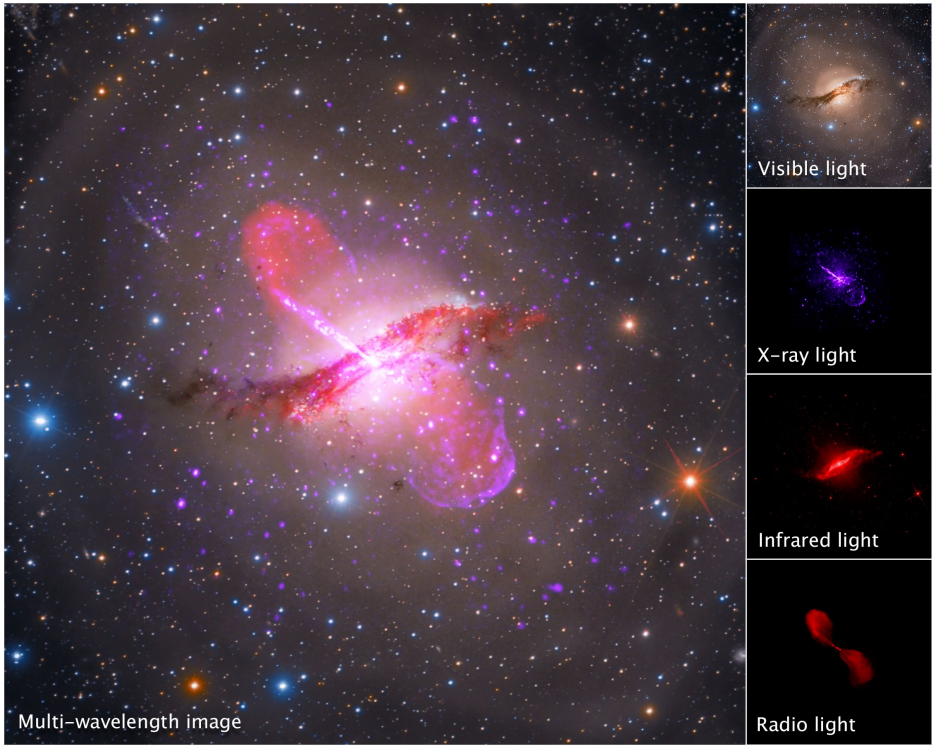




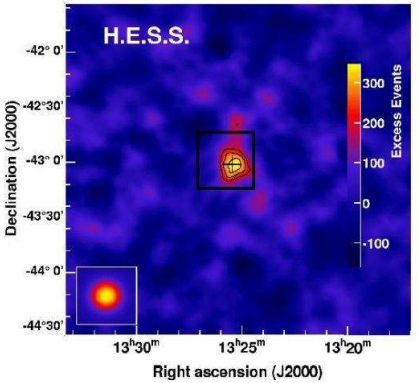


Implications of having a better event classification

The Active Galactic Nucleus
Centaurus A - A very weak source (110 hours of obs)

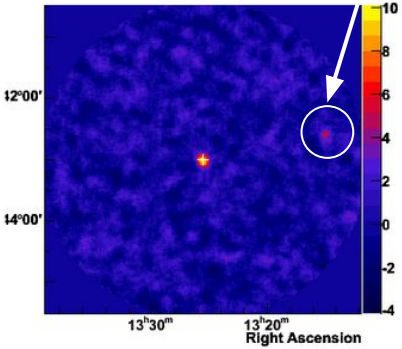


2008: discovery with a detection significance of 5σ with standard analyses (no ML)



2016: Re-analysis of data using supervised ML 9.8σ !

Appearance of a second source in the field of view!



A clear gain in the detectability of **weak** gamma-ray emitting sources

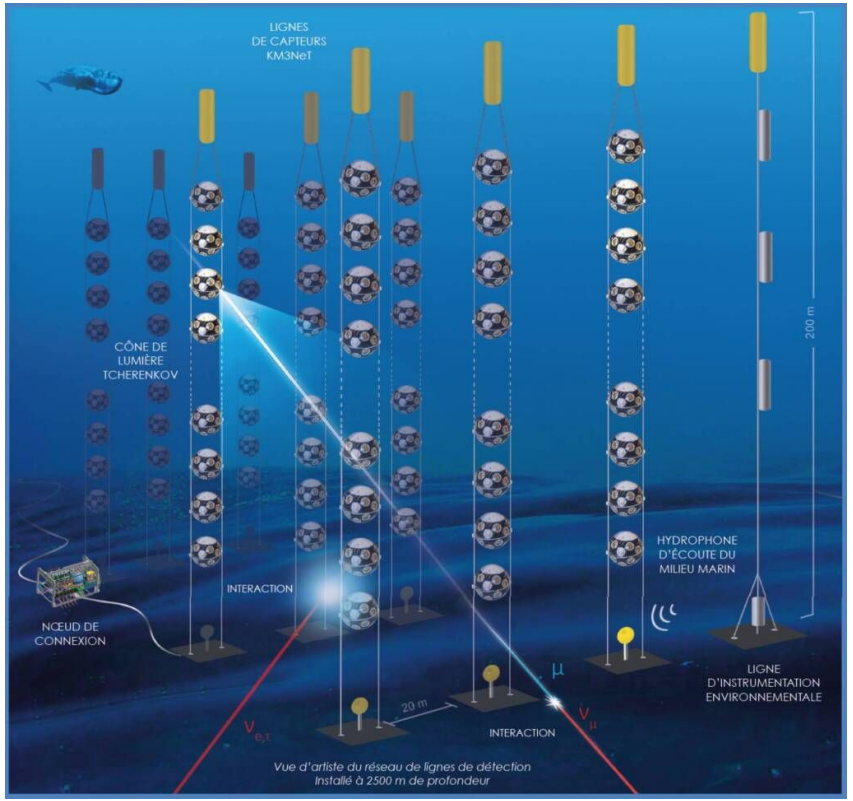
Programming versus Machine Learning versus Deep Learning for Regression

- A clear gain in performance in energy reconstruction seen with Machine Learning and Deep Learning

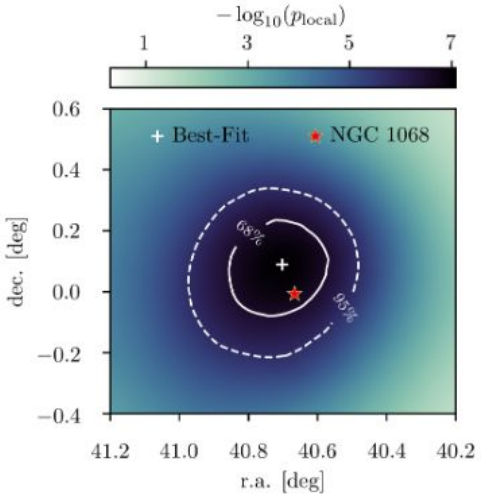
- The improvement in the incoming direction of gamma rays is still uncertain, as standard algorithm-based reconstruction methods are performing well

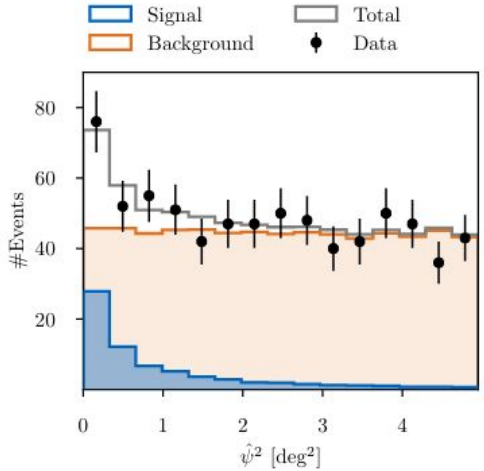
- Deep Learning And especially Graph Neural Networks (GNNs) for direction reconstruction in Neutrino Telescopes (IceCube, KM3NeT) seems much more promising, especially at the lowest neutrino energies

    - The uncertainty on the ice and water absorption and scattering
    - These are not known very well, so Deep Learning helps in modelling them



Low energy neutrinos from NGC 1068

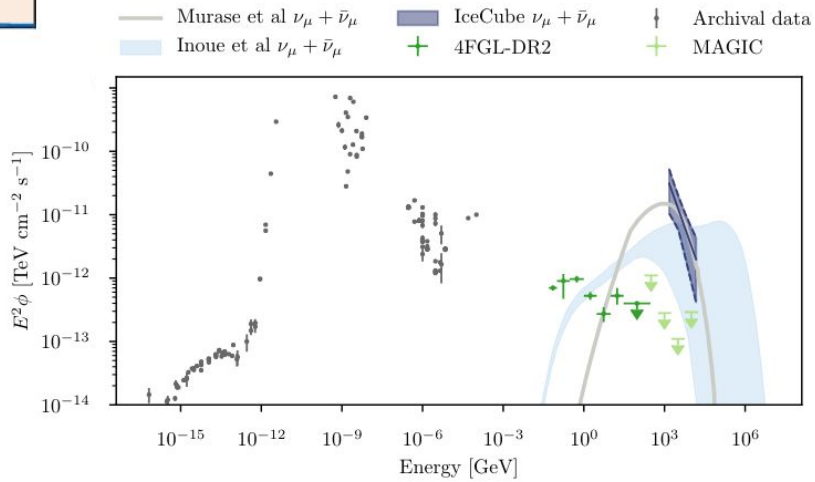




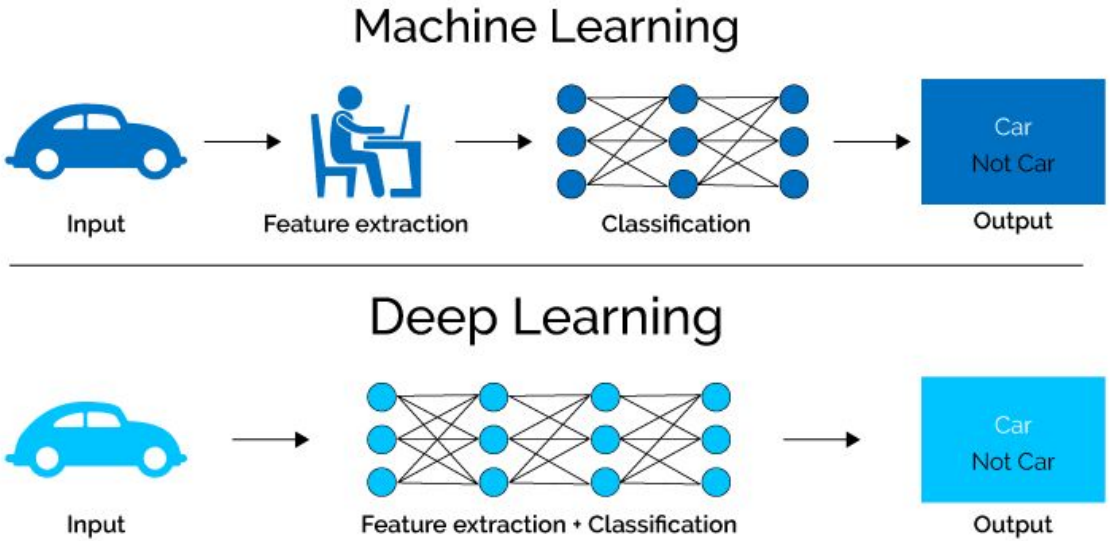
"Evidence for neutrino emission from the nearby active galaxy NGC 1068"
Science, 2022

- Results from November 2022
- Signal significance 4.4 sigma
- The highest up to now

No Graph Neural Networks but
- Gradient boosted decision tree for angular errors (so standard ML)
- CNN for energy reconstruction (DL)

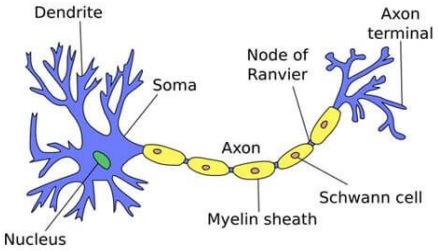


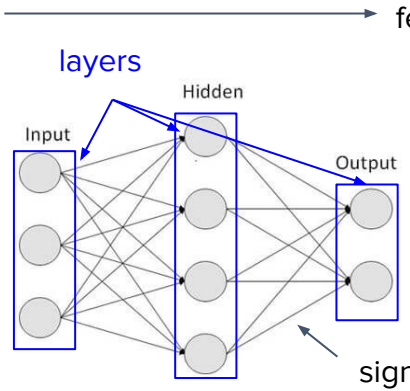
Machine Learning versus Deep Learning



XenonStack

Feedforward Artificial Neural Networks: The simplest Neural Network

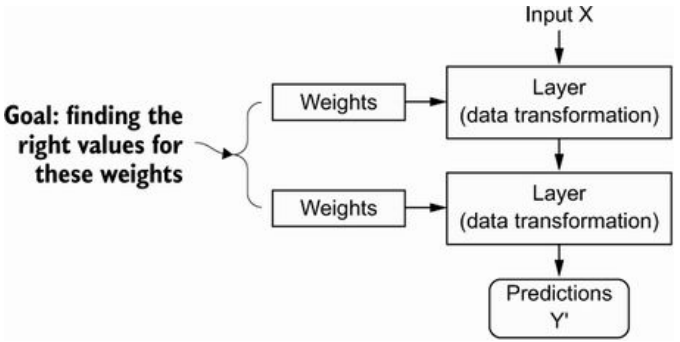




Adapted from
https://clevertap.com/blog/neural-networks/

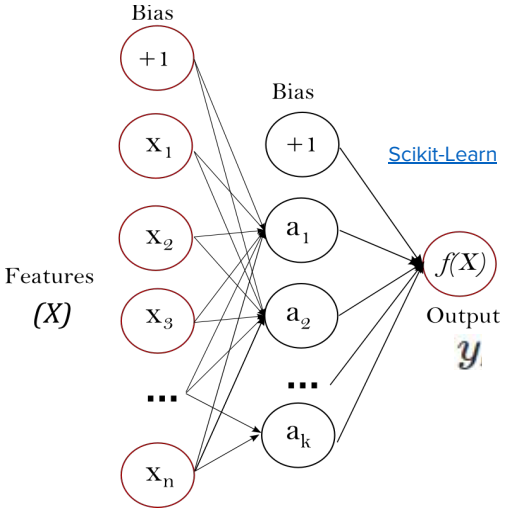
What is a Neural Network?

- Computing system that can learn a non-linear function approximation for either Classification and Regression
- Based on a collection of connected units called Neurons
- Each connection between neurons can transmit a signal from one neuron to another via a weight
- The receiving neuron then processes the signal and then signals downstream neurons connected to it
- Neurons are organized in layers, performing different kinds of transformations on their inputs
- The transformation implemented by a layer is parameterized by its weights
- Signals travel from the first layer (“Input”) to the last layer (“Output”)
- In between the Input and the Output layers there are Hidden layers
- No communication in between the neurons of the layer
- Deep NN ➔ many hidden layers



What does learning mean in this context?

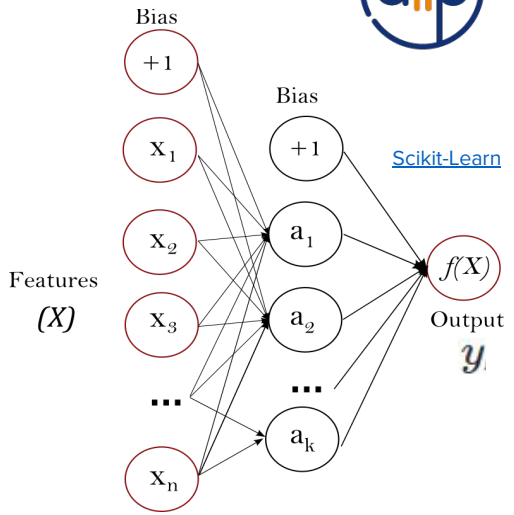
Learning means finding a set of values for the weights of all layers in a network, such that the network will correctly map example inputs to their associated targets

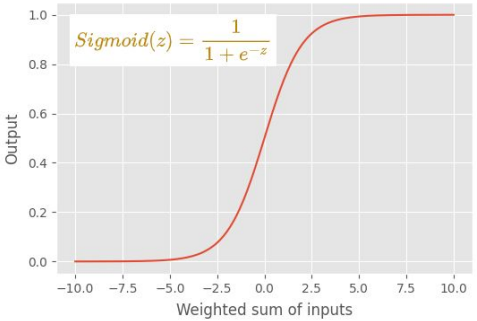


- Features (already covered) ➔ Input data
- Neurons
- Layers (Input, Output, Hidden) ➔ Filters that extract different representations of data
- Weights
- Weighted linear summation
- Activation function
- Loss function
- Gradient descent

Feedforward Artificial Neural Networks: Neural Network for Classification

- Given a set of input features $\{x_i | x_1, x_2, \ldots, x_m\}$
- And a target value y; where the target value can be the result of a classification task or a regression task
- Each connection from a neuron of one layer to another neuron of another layer will have its own assigned weight ***w*** which represents the strength of the connection between the neurons
- The value assigned to each hidden neuron will be given:
  - by the weighted linear summation $w_1x_1 + w_2x_2 + \ldots + w_mx_m$ + *Bias*
  - which is then passed to a non-linear activation function which transforms the output. Without an activation function, the network would behave like a linear model regardless of the depth, meaning it would not be able to model complex relationships in the data.
  - So the activation function of a neuron defines the output of that neuron, given a set of inputs. Example: Sigmoid



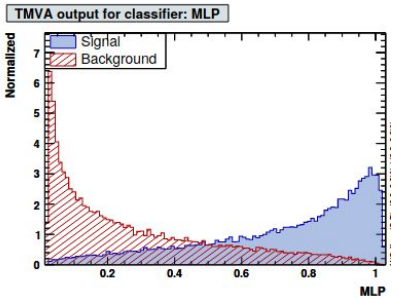


R. Pramoditha

The ***Sigmoid*** activation function for **the output layer** in classification

- If the input is $\ll 0$, then it will be transformed in a number $\simeq 0$
- If the output is $\gg 0$, it will be transformed in a number $\simeq 1$
- For values $\simeq 0$, it will transform to a number $\in [0, 1]$.
- So for Sigmoid, the output will be between [0, 1], in the end.
- For numbers close to 1: neurons are “activated”
- For numbers close to 0: neurons are “deactivated”
- A threshold defined in the [0, 1] output makes it possible to define the purity of the classification

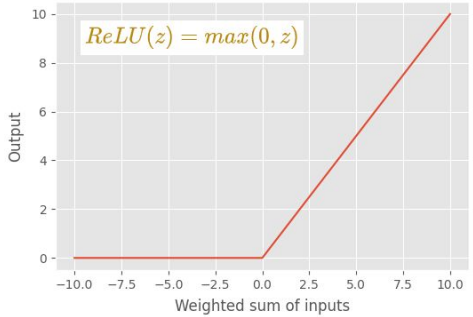
TMVA



Classification

Feedforward Artificial Neural Networks: Neural Network Activation Functions

R. Pramoditha



The ***Rectified Linear Unit (ReLU)*** activation function in the **hidden layers** for classification

- ReLU = max(0, z)
- If the input value ≥ 0, the ReLU function returns the input as it is.
- If the input < 0, the ReLU function returns 0.
- Used for hidden layers ("default") as no vanishing gradient problem

Classification

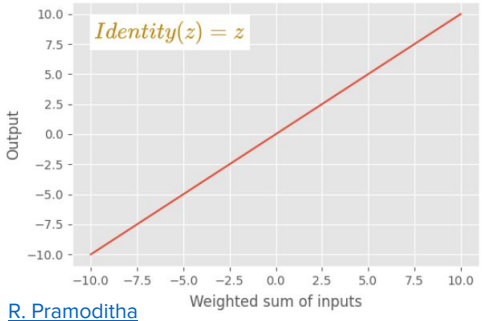
The ***Softmax*** activation function in the **output layer** of multi-class classification

$$Softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

R. Pramoditha

- The softmax function calculates the probability value of an event (class) over **K** different events (classes).
- It calculates the probability values for each class: the sum of all probabilities is 1 meaning that all events (classes) are mutually exclusive.
- Not for hidden layers, only output layer

Classification



R. Pramoditha

Linear activation function in the **output layer** of a regression problem

- This is the only function which is considered linear
- This function outputs the input value as it is. No changes are made to the input.
- This function is only used in the output layer of a neural network model that solves a regression problem.

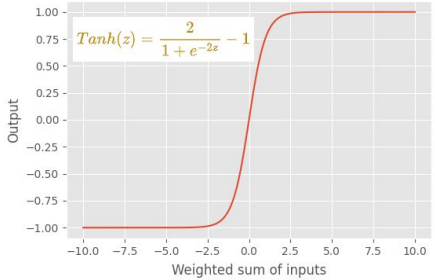
Regression

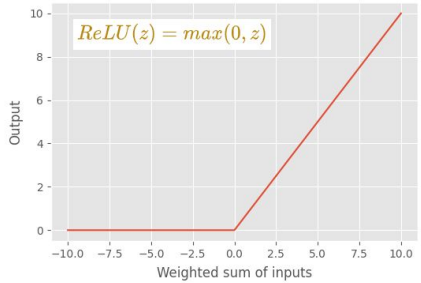
Feedforward Artificial Neural Networks: Activation Functions Summary

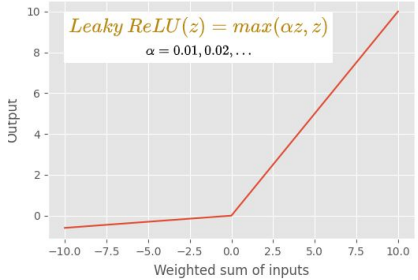
How to choose the correct activation function? It depends on the problem you wish to solve.

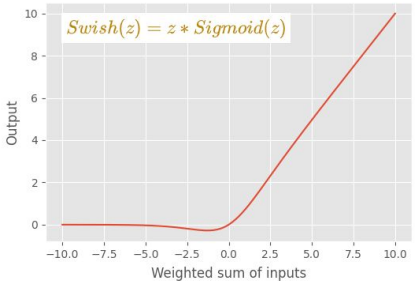
- No activation function is required in the input layer
- In the output layer:
  - ***Linear*** with one node for regression
  - ***Sigmoid*** for a binary classifier
  - ***Softmax*** with one node per class for *multiclass classification*
  - ***Softmax*** or ***Sigmoid*** for *multi-label classification*
- In hidden layers, non-linear activation functions:
  - First choice ***ReLU*** (always default for MLP and CNN), if NN not performing well, then "***Leaky ReLU***"
  - ***Sigmoid*** or ***Tanh*** in RNN
  - ***Swish*** and ***Hard Swish*** functions (new)

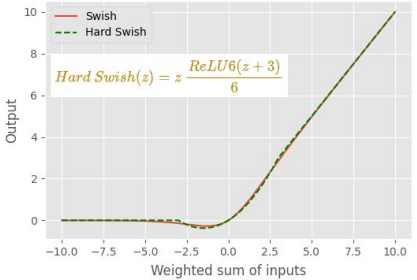
Problem Type	Output Type	Final Activation Function
Regression	Numerical value	Linear
Classification	Binary outcome	Sigmoid
Classification	Single label, multiple classes	Softmax
Classification	Multiple labels, multiple classes	Sigmoid











S. Ronaghan

R. Pramoditha

Feedforward Artificial Neural Networks: The role of the bias

What is the role of the bias? It gives flexibility in the activation of the neuron.

- Let's consider the ReLU activation function: if it receives an input value

$$a_1 = w_1x_1 + w_2x_2 + \cdots + w_nx_n < 0$$

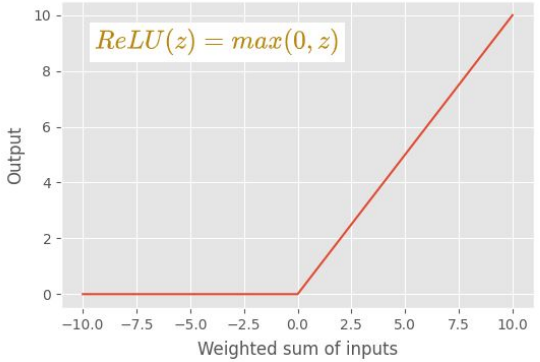
it will output $max(0, a_1) = 0$
so that neuron will not "fire", i.e. it will be deactivated.

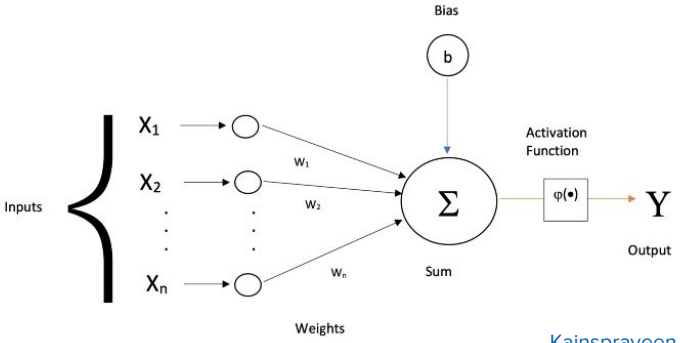
- By adding a constant value, I can accept more values and fire more neurons

$$a_1 = w_1x_1 + w_2x_2 + \cdots + w_nx_n + Bias > 0$$

- So, the Bias **allows you to shift the activation function by adding a constant (i.e. the given bias) to the input**

- With the Bias added in the weighted linear summation passed to the activation function, the model will have an increased flexibility in fitting the data, because it has a broader range in what values it considers to activated or not





Kainspraveen

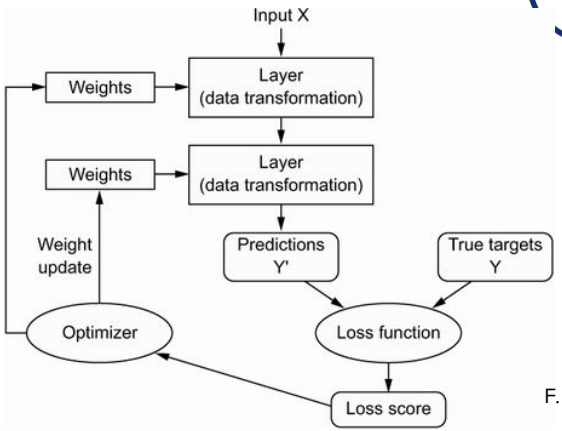
Feedforward Artificial Neural Networks: Training, Loss, Optimizer, Backpropagation

During training, the weights are constantly updated
in the attempt to reach their optimal values

How weights are optimized depends on the optimization algorithm or *optimizer* that you choose to use for your model

Examples of optimizers are:
Stochastic Gradient Descent (SGD) and Adam

Weights are optimized via the minimization of the loss function
➔ the goal is to make the loss function as close to zero as possible



F. Chollet

In a classification problem, we give the network a dataset with their true labels (ex: cats and dogs)

- If we give an image of a cat to the model, this image passes through the entire network, weights are calculated, and the model gives us an output probability at the end (ex: 75% cat, 25% dog)
- The loss is the distance (the error) between what the network is predicting for the image versus what the true label of the image, giving a score
- We then use this score as a feedback signal to adjust the value of the weights a little, in a direction that will lower the loss score.
  This adjustment is the job of the optimizer, which implements what's called the Backpropagation algorithm.
  So the optimizer tries to minimize this distance to make our model as accurate as possible in its predictions
- After a full pass of the data (an "epoch"), the network will continue to pass in the same data over and over again (many epochs) and will
  ➔ learn and update the weights of the model accordingly

Loss functions (examples)

Classification

Binary Cross-Entropy

$\hat{y}$ is prediction, y is ground truth

$$f(y, \hat{y}) = -\frac{1}{n}\sum_{i=1}^{n}[y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)]$$

Multi-Class Cross-Entropy Loss

$$f(y, \hat{y}) = -\sum_{c=1}^{M} y_c log(\hat{y}_c)$$

M: total number of class

Regression

$$\text{MAE} = \underbrace{\frac{1}{n}\sum_{i=1}^{n}}_{\text{test set}} \underbrace{|y_i}_{\text{predicted vaue}} - \underbrace{\hat{y}_i|}_{\text{actual value}}$$

Mean Absolute Error

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

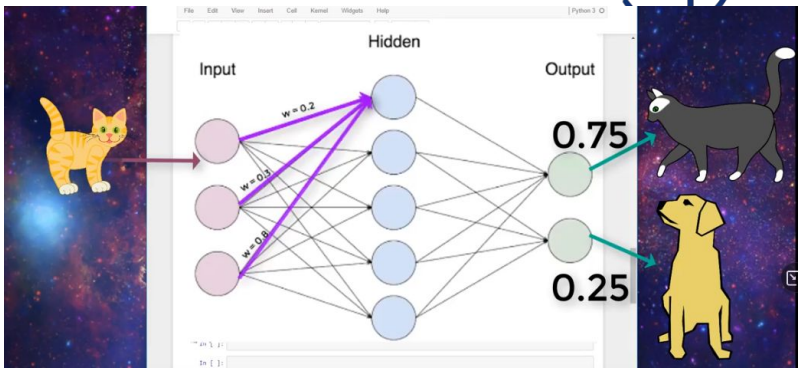
Mean Squared Error

Feedforward Artificial Neural Networks: The Loss function and Backpropagation

When the training starts, the model is set with arbitrary weights, which are then optimized during the training.

For each entry in the network, we get a prediction for each entry
➔ the model then computes the loss or the error of that output by comparing the truth label y with what the model predicted y^*
→ (y-y^*) through a Loss Function

Loss calculation: computation of the gradient of the loss function $\mathcal{L}$ wrt each weight that has been calculated $d\mathcal{L}/dw$



deeplizard

Backpropagation via chain rule:

$$a_1 = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + Bias$$

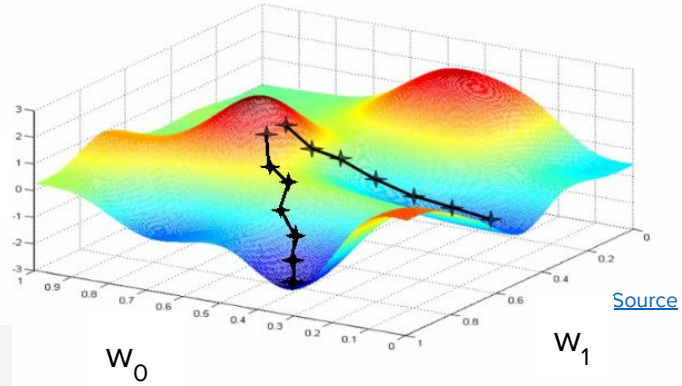
$$f(a_1) = \frac{1}{1 - e^{-a_1}} \qquad \frac{df}{da_1} = f(a_1)(1 - f(a_1))$$

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial f} \frac{\partial f}{\partial a} \frac{\partial a}{\partial w}$$

$\frac{\partial \mathcal{L}}{\partial f} =$ Usually simple, see previous slide

$\frac{\partial a}{\partial w} =$ constant

$\frac{\partial f}{\partial a} =$ Can be complex



Source

Feedforward Artificial Neural Networks: The Loss function and Backpropagation

When the training starts, the model is set with arbitrary weights, which are then optimized during the training.

For each entry in the network, we get a prediction for each entry
➔ the model then computes the loss or the error of that output by comparing the truth label y with what the model predicted y^*
→ (y-y^*) through a loss function

Loss calculation: computation of the gradient of the loss function $\mathcal{L}$ wrt each weight that has been calculated $d\mathcal{L}/dw$
➔ Backpropagation via chain rule $d\mathcal{L}/dw = d\mathcal{L}/da\ da/dw$

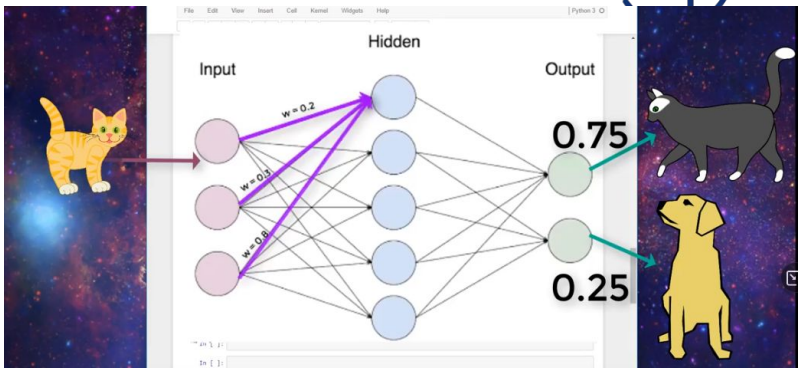
$d\mathcal{L}/dw$ is then multiplied by the learning rate (0.01-0.001)

The learning rate is a hyperparameter that determines the step size at each iteration while moving toward a minimum of a loss function $\mathcal{L}$

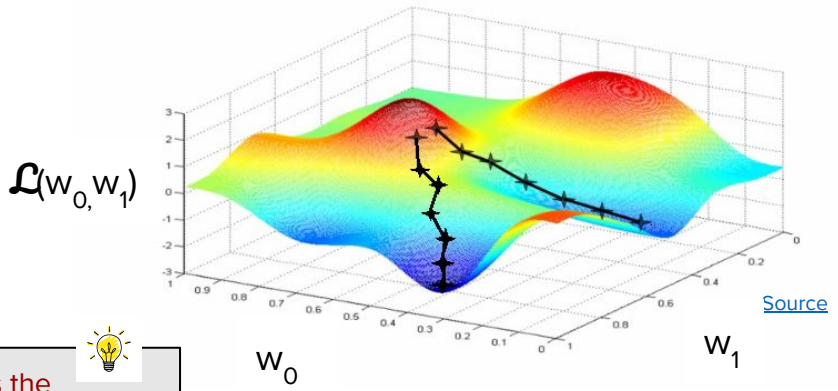
$d\mathcal{L}/dw$ x 0.001 becomes small, and it is used to update the value of the weights and biases:

$w = w - d\mathcal{L}/dw$ x 0.001
$b = b - d\mathcal{L}/db$ x 0.001

After each Forward Pass, BackPropagation updates the weights, then the next Forward Pass will begin its training with these new updated weights



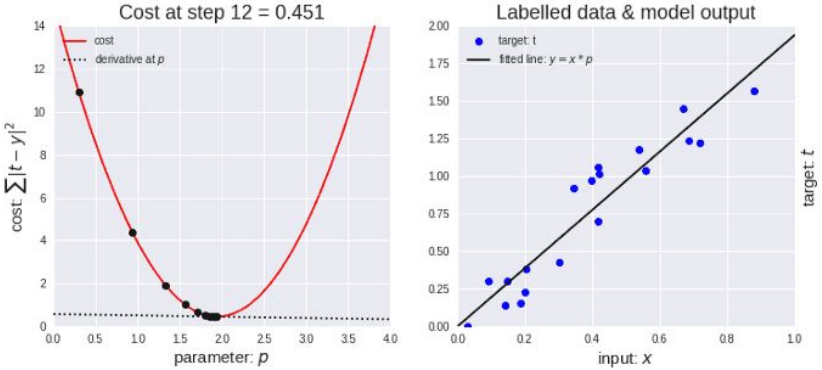
deeplizard



Source

Visualizing the Loss function with respect to the epoch

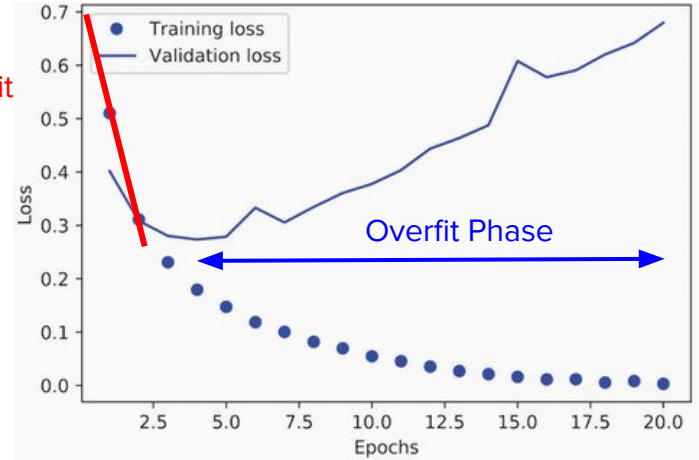
Source



During gradient descent optimization, the training loss decreases with every epoch (and with it the training accuracy)

The quantity that you are trying to minimize should be less at every iteration

Underfit Phase



Adapted from F. Chollet

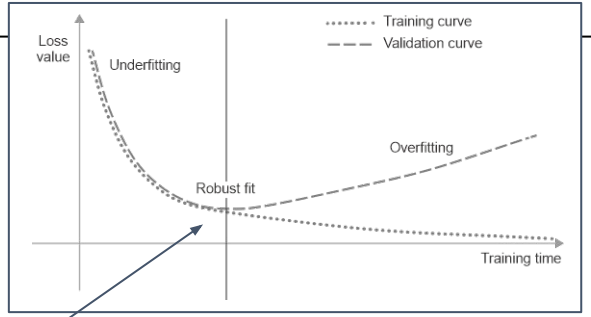
The Training Loss decreases as a function of the epoch ✔

The Validation Loss does not behave as the training Loss ➔ it reaches a minimum at 4 epochs

The model is performing well on the Training sample, but very badly on data he has never seen before ➔ Overfitting

Overfitting means that the model is beginning to learn patterns that are specific to the training data or that are misleading or irrelevant when it comes to new data.

How to avoid Overfitting



Possible solutions to overfitting:

- Get more training data
- Reduce the capacity of the network
- Let the network focus only on the most important patterns with Regularization
- Dropout
- Stop the training at the epoch where the model starts overfitting and re-train until that epoch only

Reduce Capacity

Reduce the size of the model, start with a few layers and feature, add new layers until when the Validation Loss continues to diminish

- A smaller network will start overfitting later
- A larger capacity network will model quicker, but it will be more prone to Overfitting

To play with Neural Networks: https://playground.tensorflow.org/

Regularization

Add a weight regularization term to the loss function to force the network weights to take only small values, making them more regular (L1 and L2)

$$J_{regularized} = \underbrace{-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log\left(a^{[L](i)}\right)+(1-y^{(i)})\log\left(1-a^{[L](i)}\right)\right)}_{\text{binary cross-entropy cost}} + \underbrace{\frac{1}{m}\frac{\lambda}{2}\sum_{l=1}^{L}\sum_{i=1}^{n^{l}}\sum_{j=1}^{n^{l-1}}W_{j,i}^{[l]2}}_{\text{L2 regularization cost}}$$

Dropout

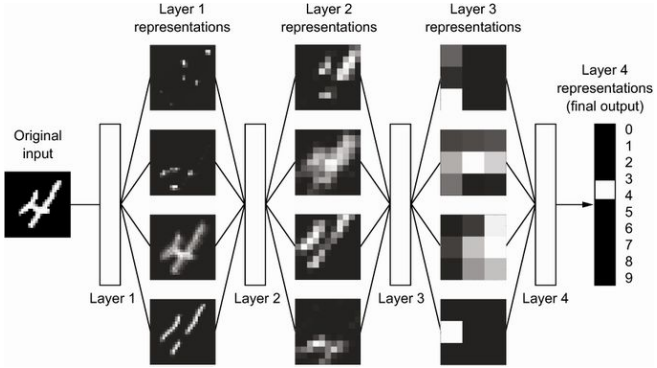
Very effective technique applied to a layer which consists in randomly dropping out (setting to zero) a number of output features of the layer during training

Deep Learning: what is it?

- Deep Learning is a group of techniques that are based on neural networks that have the capacity to learn complex patterns directly from the data
- Deep learning algorithms are able to capture patterns that are a little bit more abstract, so you can perform tasks with deep learning algorithms that you would not be able to with traditional machine learning algorithms
- In Deep Learning you can just feed your data into the algorithm, and then the important features will be extracted, also in the cases where humans would have difficulty to find them

- Deep Computer Vision
- **Deep Sequence Modelling**
- **Deep Generative Modelling**
- Deep Reinforcement Learning
- **Bayesian Deep Learning**
- Solving Partial Differential Equations

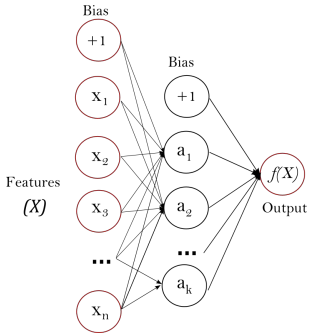
- CNNs are designed to work with images
- More generally, array data where there are local correlations in the features and translational invariance of the signal
- Detect object in natural images
- Each output of the CNN has a window of influence on the input



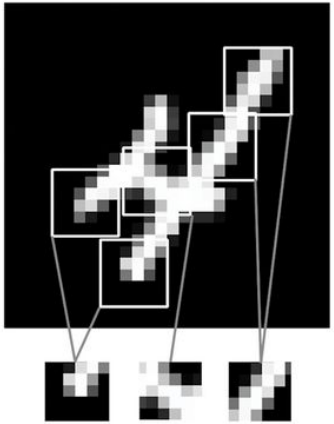

```
from tensorflow import keras
from tensorflow.keras import layers
inputs = keras.Input(shape=(28, 28, 1))
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(inputs)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
outputs = layers.Dense(10, activation="softmax")(x)
model = keras.Model(inputs=inputs, outputs=outputs)
```

Conv2D
MaxPooling

Fully connected



CNNs: Convolutional Layers



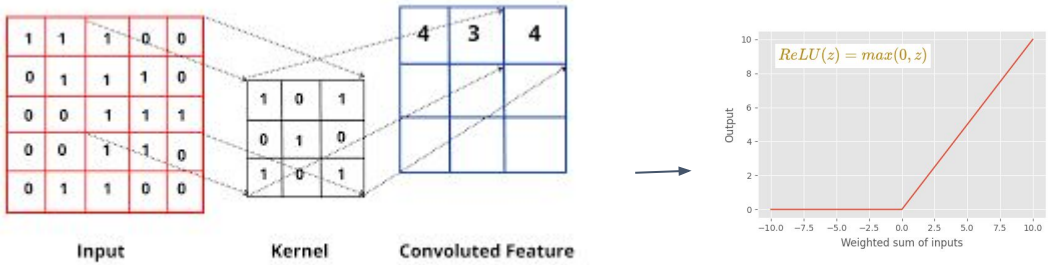
The main difference between a densely connected layer and a convolution layer:

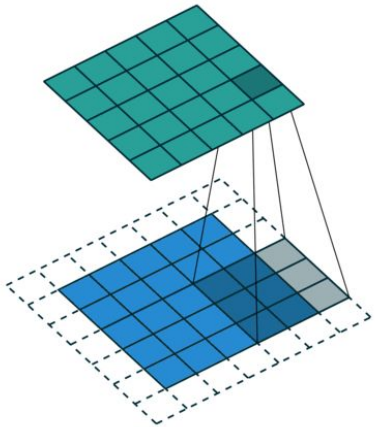
Dense layers learn global patterns in their input feature space (for example, for a MNIST digit, patterns involving all pixels), whereas convolution layers learn local patterns

A Convolution layer in the case of images learns patterns found in small 2D windows of the inputs.

```
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(inputs)
```

- Kernel type (3x3)
- Number of Kernels (32)
- Stride (Default = 1)



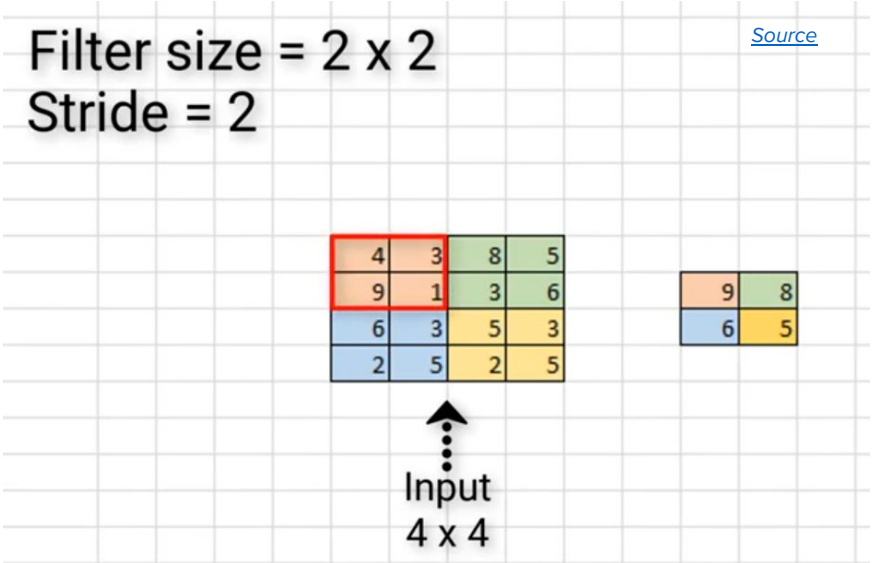


CNNs: Max Pooling 2D

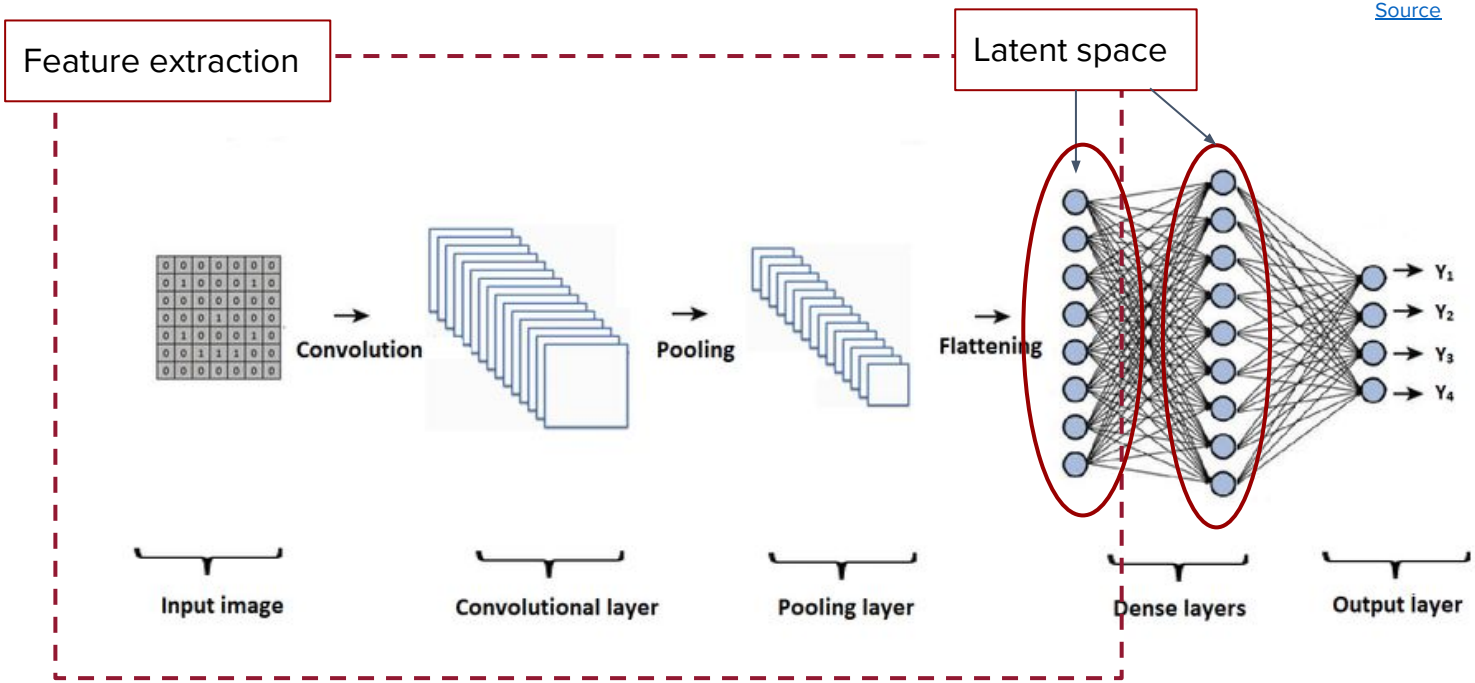
- Reduces the resolution of the output of the convolutional layer
- The NN will look at larger areas of the image

  ➜ Reduces the number of parameters and computational load

  ➜ Helps to reduce overfitting
- A particular NN will extract particular features from the image: edges, circles, lines, etc
- We can think of the higher valued output pixels of the conv layer as being the most activated
- So with Max Pooling we are pushing forward in the NN the most activated pixels



Source



The latent space is an abstract multidimensional space that encodes a meaningful internal representation of externally observed events

Samples that are similar “in the external world” are positioned close to each other in the latent space.

- Introduction to Transformers
- Key Features of Transformers
- Understanding the Attention Mechanism
- Handling Sequences with Transformers
- Understanding Encoder-Decoder Architecture
- Advantages of Transformers
- Introduction to Applications of Transformers
- BERT, GPT, and T5: Transformer-Based Models
- Transformers in Computer Vision
- Transformers in Speech Processing
- Transformers in Music Generation
- Transformers in Healthcare and Biotechnology
- Transformers in Reinforcement Learning
- Transformers in Time Series Analysis
- Transformers in Video and Graph Data Processing

The origin of Transformers is from the paper 'Attention Is All You Need' by Vaswani et al. in 2017

Transformers are the foundation for many state-of-the-art models in natural language processing (NLP)

Key features of Transformers

The key innovation of the Attention Mechanism allows the model to focus on different parts of the input sequence when predicting a part of the output sequence.

Transformers can process entire sequences of data simultaneously and in parallel, unlike RNNs or LSTMs that process data sequentially.

This parallelization significantly speeds up training and makes the model highly scalable with increasing data and compute power.

Many Transformer models use an Encoder-Decoder structure to process and generate sequential data.

Traditional sequence models (RNNs and LSTMs)

- Process data in sequential order
- Unable to focus on specific parts of the sequence

Transformer's Attention Mechanism

- Can focus on different parts of the input sequence
- Allows the model to capture complex dependencies and relationships in the data

Traditional Models

- Process data in order
- Sequential processing of data
- Limited by the sequence length and order

Transformer Models

- Process entire sequences simultaneously
- Parallel processing of data
- Not limited by sequence length or order

Data Generation

Encoder Process

- The encoder processes the input data, such as a sentence in English.
- It compresses the information into a context.

Decoder Process

- The decoder uses the context generated by the encoder to produce the output data.
- This output data could be a translated sentence in French, in the case of language translation tasks.

BERT, GPT, and T5: Transformer-Based Models

Computational Linguistics

Natural Language Processing (NLP)

BERT

Bidirectional Encoder Representations from Transformers.
Understands the context of a word in a sentence.

GPT

Generative Pretrained Transformer.
Known for its text generation capabilities.

T5

Text-To-Text Transfer Transformer.
Converts all NLP tasks into a text-to-text format.

Focus on: Syntax (sentence structure), semantics (meaning), and context of the language

Speech processing

Involves analysing the sound waves generated by human speech, detecting phonemes (basic units of sound), and understanding accents, intonation, and rhythm

Use in Speech Recognition

- Transformers are employed in models to convert spoken language into text.
- They can effectively handle the sequential nature of speech for accurate transcription.

Role in Speech Synthesis

- In text-to-speech systems, transformers help in generating natural and human-like speech from text.
- They enable converting spoken language into text.

Other applications of Transformers: Computer Vision

Impact on Computer Vision

Transformers, traditionally used in NLP, are now applied to process visual data.

The self-attention mechanism allows efficient processing of image patches for tasks like classification and object detection.

Example Applications

Vision Transformer (ViT) utilizes transformers for image classification, replacing traditional CNNs.

DEtection TRansformer (DETR) is a transformer-based model for object detection, eliminating the need for specialized components like region proposers or anchor boxes.

Unlike traditional convolutional neural networks (CNNs) that process images through a series of local operations (convolutions), Visual Transformers treat an image as a sequence of patches and applies self-attention mechanisms across these patches.

Other applications of Transformers

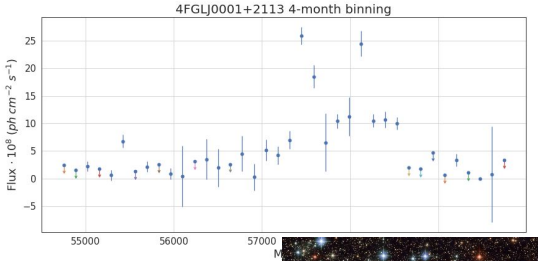
Transformers excel in capturing long-range dependencies, a key challenge in time series analysis and forecasting.

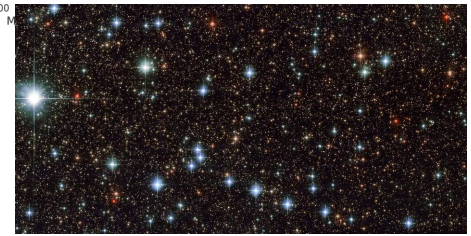
They are employed in video understanding tasks, like action recognition and video classification.

Graph Neural Networks (GNNs) with transformer architectures are used for tasks like social network analysis, molecule structure analysis, and recommendation systems.

Transformers can be used to generate music by learning the structure and patterns of musical pieces, and they can create new compositions by predicting sequences of notes or even entire musical pieces.

Deep Learning & Transfer Learning





Deep Learning

You use the data in your possession to train a model: light curves, images, etc

- But the data you have might not be enough to train a good model, or you do not possess enough labelled data. Or it will require exorbitant computing times that you do not possess
- Possible solutions: improve labelled data statistics, use GANs, take more data
- OR: Transfer Learning, especially if you use images or Time Series

Transfer Learning

Transfer Learning involves taking a **pre-trained model**, originally trained on a large dataset, and adapting it for a different but related task.

Fine-Tuning Techniques

Fine-tuning can include modifying the architecture, adjusting hyperparameters, and training with a smaller, domain-specific dataset to improve performance.

Benefits for Astrophysics

This approach significantly reduces training time and resource requirements, enabling researchers to use existing models for analyzing astrophysical data.

DINOv2: a pre-trained model by META AI

DINOv2 was trained on **1.2 billion images**

Self-Supervised Learning

DINOv2 learns from **unlabeled data**, reducing the need for expensive manual labeling and enabling use in fields with scarce labeled data.

Transfer Learning Capabilities

Pre-trained on large datasets, DINOv2 is ideal for tasks unrelated to its original training, offering versatile applications across various domains.

Vision Transformer Architecture

Utilizing Vision Transformers, DINOv2 processes images as sequences of patches, enabling complex pattern recognition and scalability.

Applications in Scientific Research

Applicable in medical imaging, astronomy, and biology, DINOv2 excels in fields requiring detailed image analysis without abundant labeled data.

Unsupervised Clustering

DINOv2 discovers meaningful data clusters without labels, aiding researchers in exploring latent structures within diverse datasets.

Scalability Across Domains

Efficiently handles high-resolution images, allowing quick adaptation for specialized projects in areas like satellite imagery and microscopy.

DINOv2: a pre-trained model by META AI

DINOv2 was trained on **1.2 billion images**

Cross-Domain Applications

DINOv2's adaptability allows usage in fields like medical imaging, astronomy, art, and ecology, offering a unifying tool for diverse research areas.

No Labeled Data Required

Researchers can leverage DINOv2 for problems where obtaining labeled data is challenging, making it ideal for cutting-edge studies.

Transfer Learning Capabilities

With pre-training on diverse datasets, DINOv2 offers features that can be adapted to new tasks, reducing training time and resource needs.

Unsupervised Clustering

DINOv2's latent space naturally aligns with data structure, facilitating new discoveries and supporting hypothesis testing in research.

Scalability

Designed to handle large-scale data, DINOv2 scales efficiently, enabling researchers to work with extensive datasets without performance loss.

Enhanced Model Accuracy

DINOv2 improves accuracy in predictive models, providing reliable results across various scientific fields, enhancing credibility of research findings.

Link to the exercises:

Binary Classification with Scikit-Learn

Using a pre-trained model to classify galaxy images

Thank you for your attention!

You are welcome to join my Zoom courses:

https://u-paris.fr/doctorat/applied-data-analytics/

https://u-paris.fr/doctorat/advanced-applied-data-analytics/