

# **Machine learning**

## **part 2**

# Recap

# General

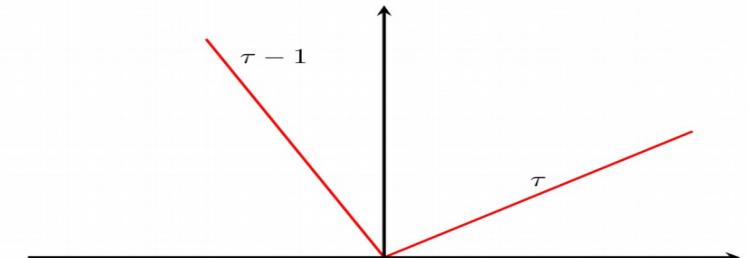
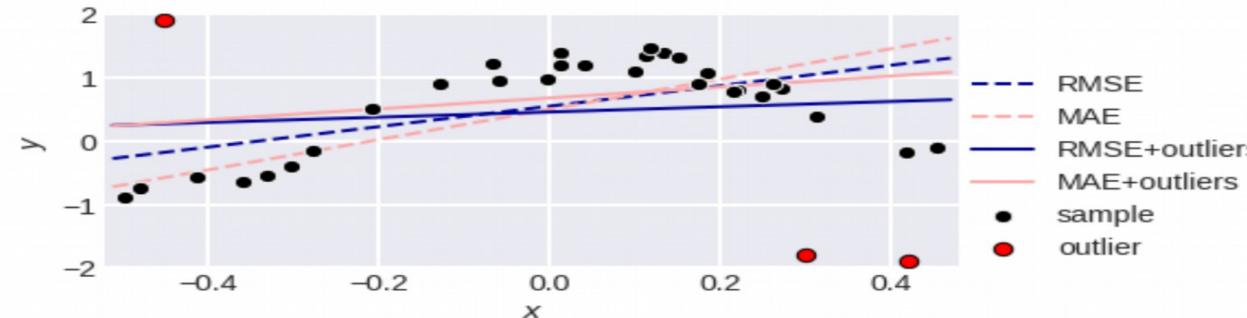
# Metrics

- A metric is a function that is used to judge the performance of your model.
- A metric function is similar to a loss function, except that the results from evaluating a metric are not used when training the model. You may use any of the loss functions as a metric function.

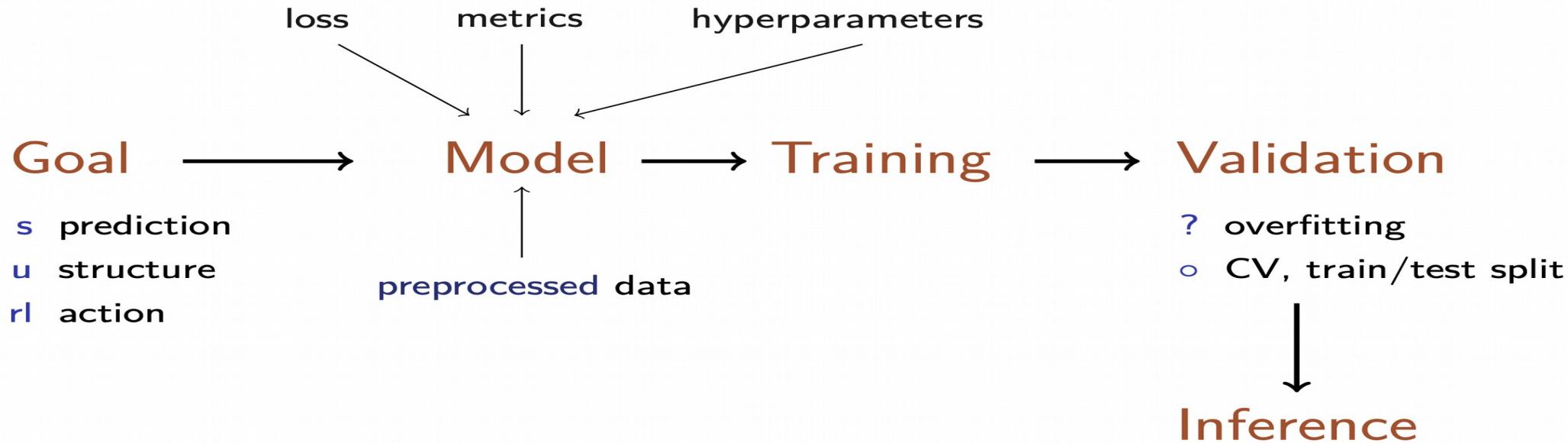
## nice one on metrics [1], [2]

- **MSE** =  $\frac{1}{\ell} \sum_{i=1}^{\ell} (y_i - a(\mathbf{x}_i))^2$ 
  - sensitive to outliers
  - easy to optimize
- **MAE** =  $\frac{1}{\ell} \sum_{i=1}^{\ell} |y_i - a(\mathbf{x}_i)|$ 
  - robust to outliers
  - gradient problems (no)
- $R^2 = 1 - \frac{\sum_{i=1}^{\ell} (y_i - a(\mathbf{x}_i))^2}{\sum_{i=1}^{\ell} (y_i - \bar{y})^2}$ 
  - explained variance
  - comparison with naive mean model
- **Logloss**

## Metrics Regression



# Pipeline



# Examples

# ML challenges

- **Precise and fast particle tracking** (single tracks, shower, jets)
- **Particle identification**
- **Fast and accurate online data processing and filtering**
- **Anomaly detection** (data quality monitoring, infrastructure monitoring)
- **Detector design optimization**
- **Data Analysis (signal from background separation)**
- **Simulation** (speed-up simulation using **generative models**, simulator parameters optimization )
- .....

## **Tracking system features**

- Particle momentum
- Particle charge
- Track parameters
- Quality of track fit
- Number of track hits
- ...

## **RICH features**

- Angle theta
- Quality of angle reconstruction
- Reconstructed particle type
- Reconstructed particle energy
- Light intensity
- ...

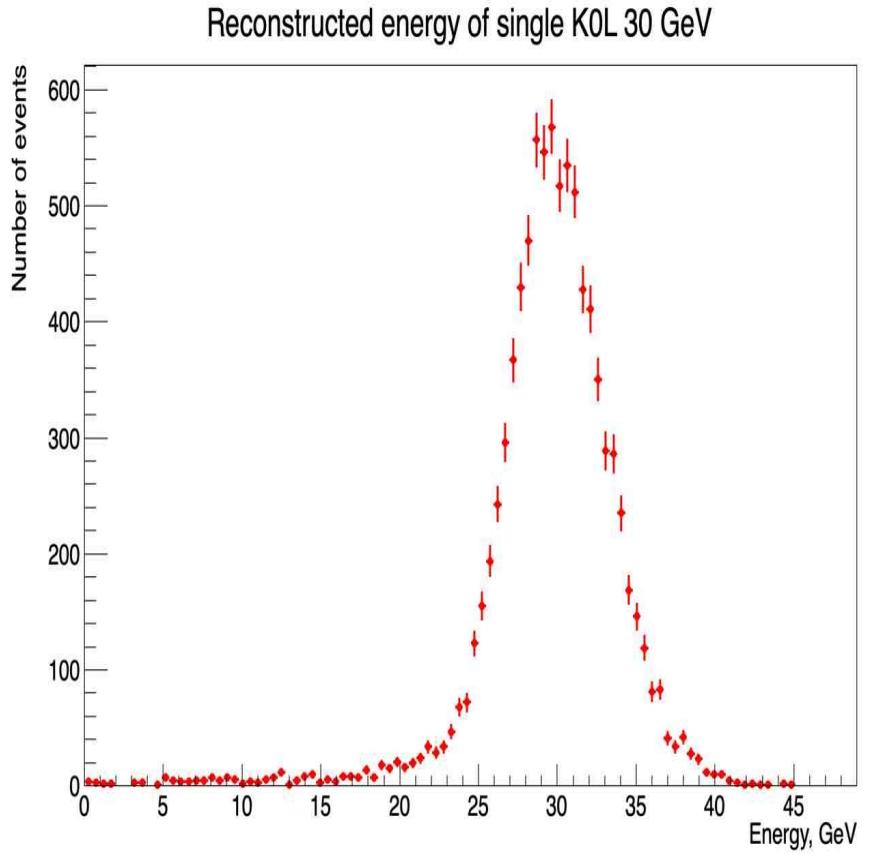
## **Calorimeter features**

- Measured particle energy
- Shower parameters: width, ...
- Number of clusters in each layer Intensity of the clusters
- Distance from track of the original particle
- ...

## **Muon detector features**

- Muon track parameters
- Quality of track fit
- Number of active layers
- Distance between the track and the active layers
- ...

- Energy in ecal and hcal
- Energy hit in both calorimeters
- Energy from muon detector
- Number of hits in both calorimeters
- Radius of the shower (in both calorimeters)
- Average number of layers in both calorimeters
- Number of hits from muon detector
- Average number of layers from muon detector
- .....



# Machine Learning Pipeline

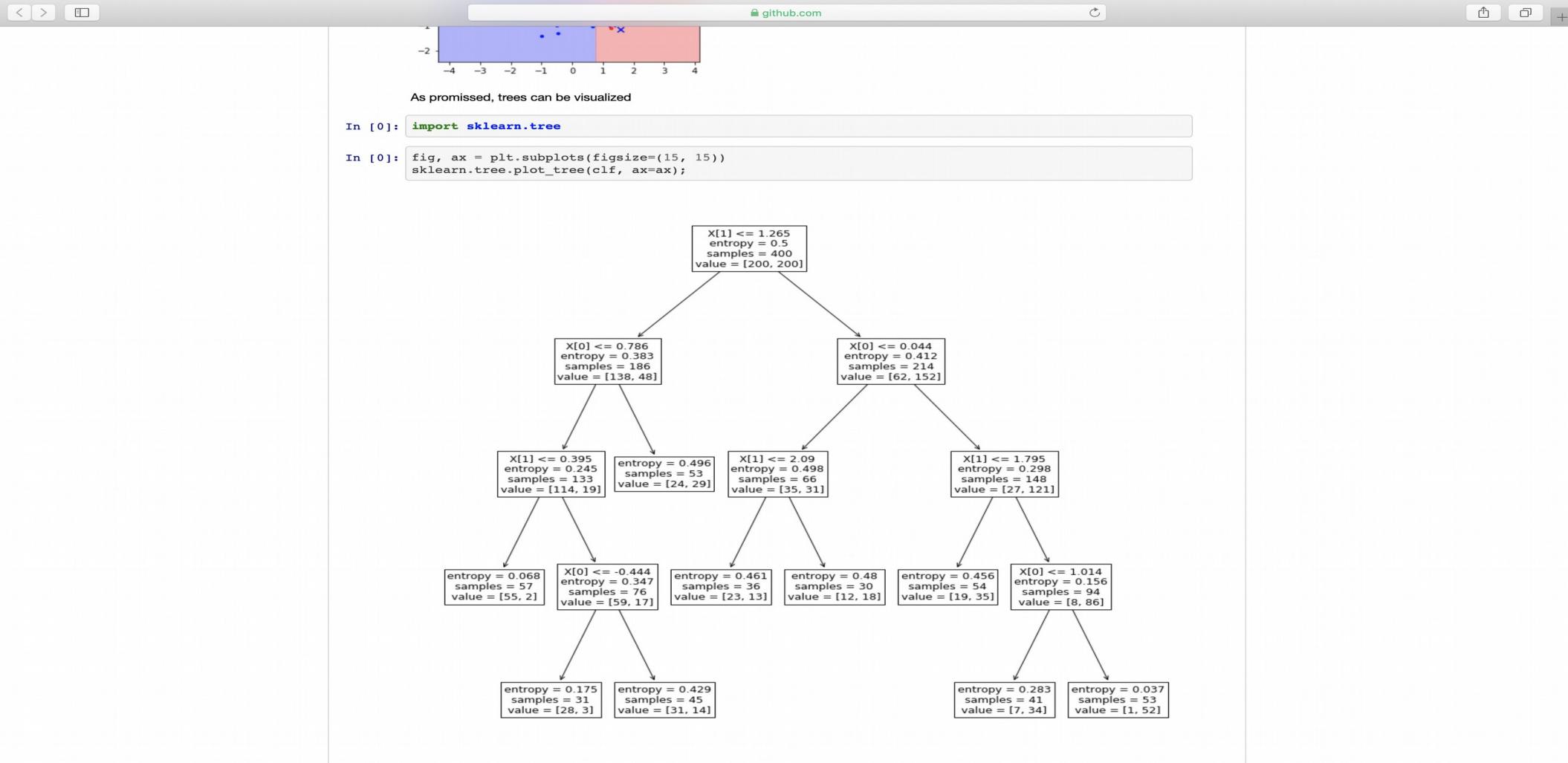
What Most People Think



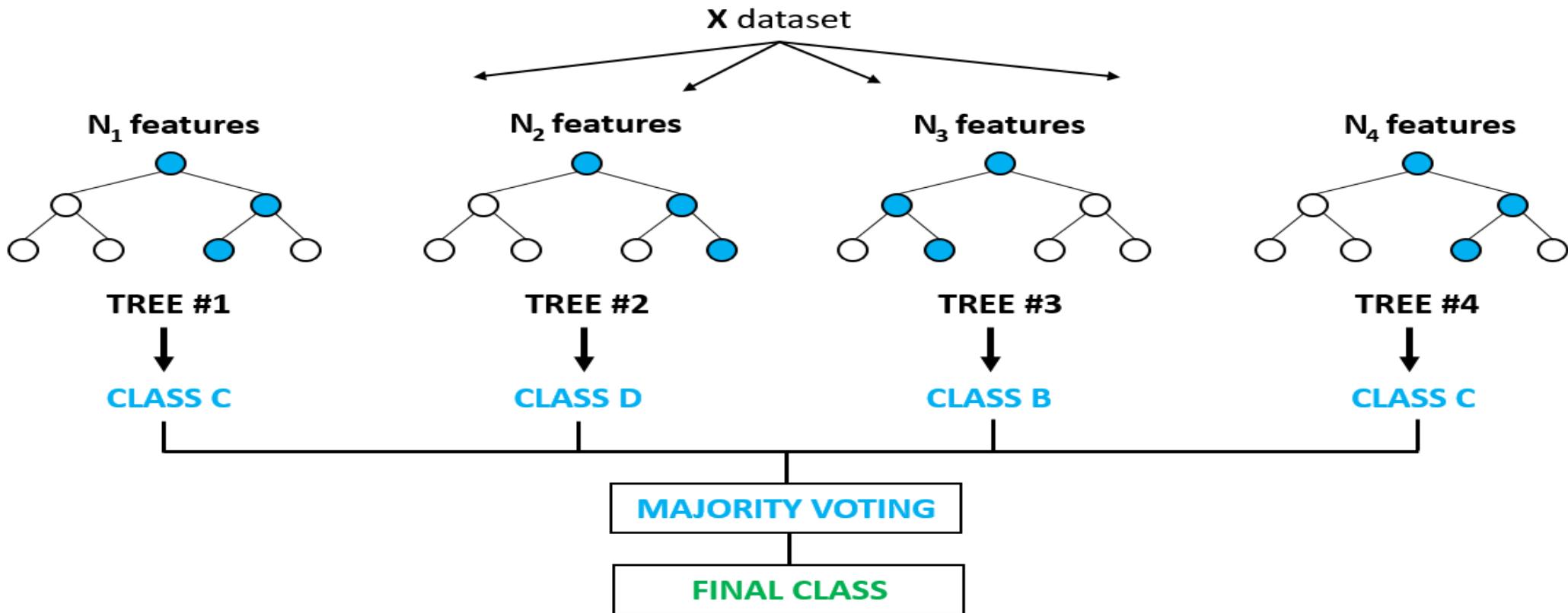
What Successful People Know



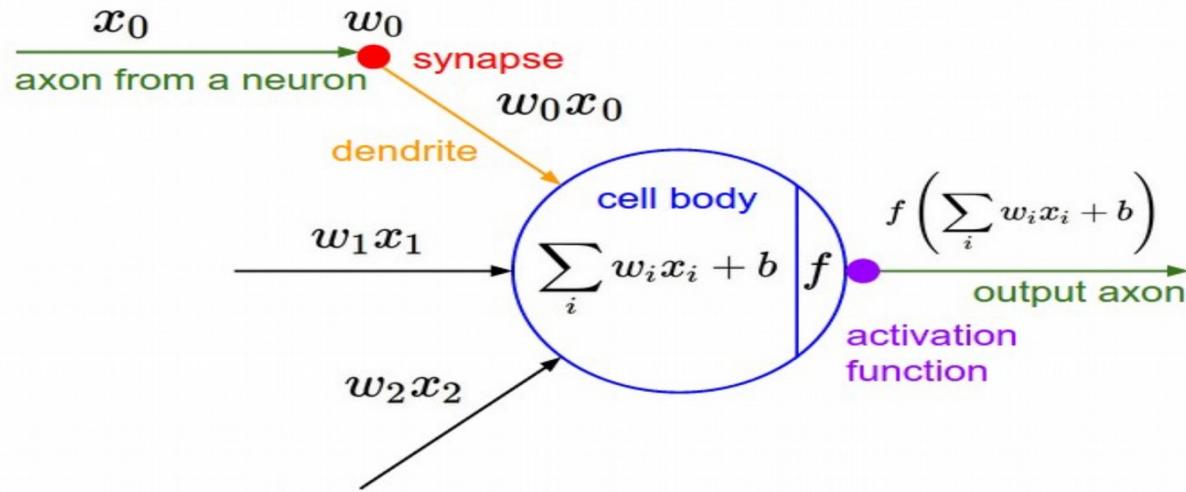
# BDT / Random Forest



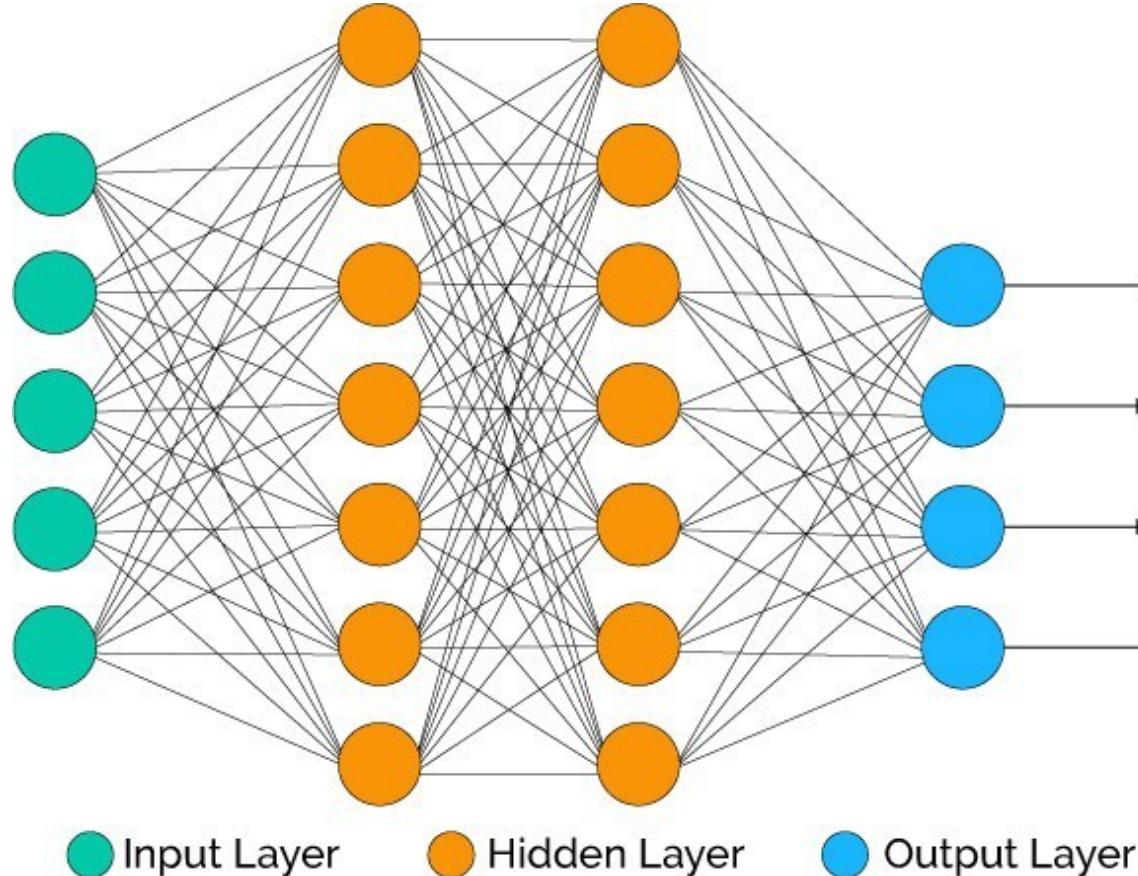
# Random Forest



# Artificial Neural Network (ANN)



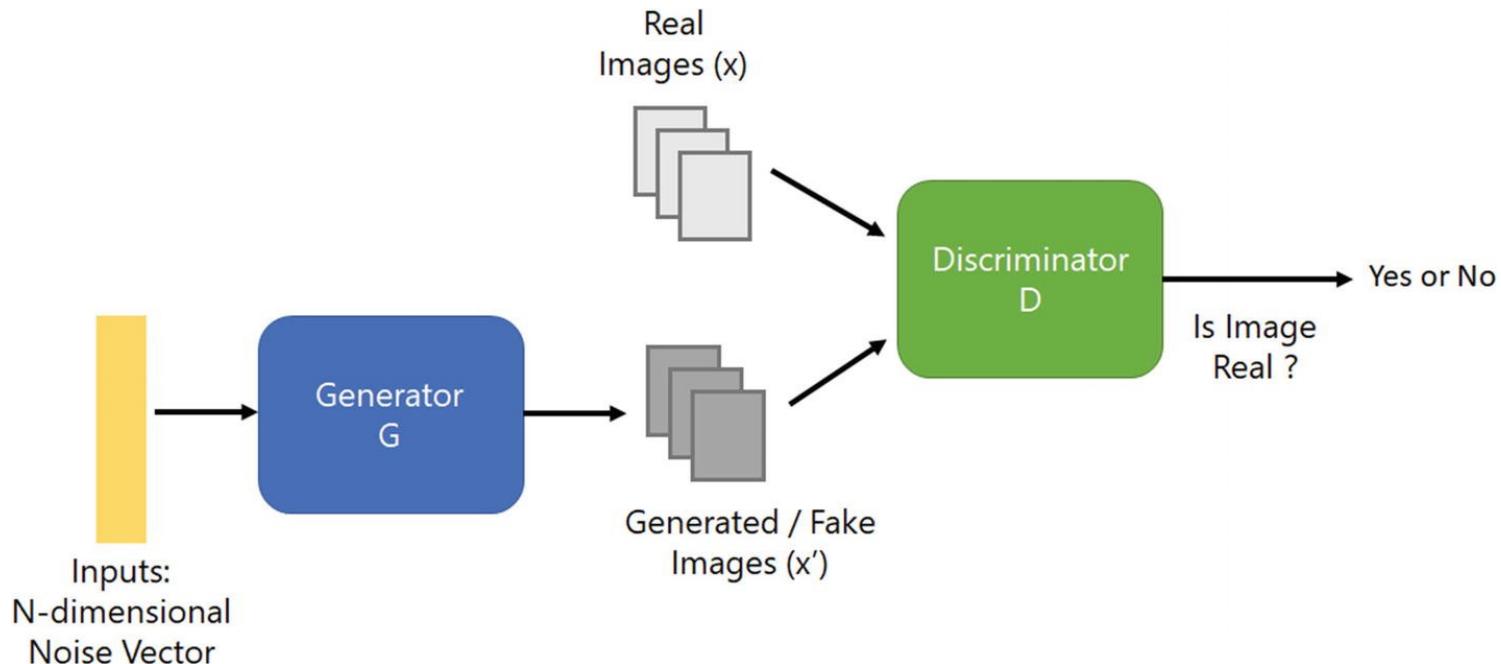
# Artificial Neural Network (ANN)



# Part 2

# Generative adversarial network (GAN)

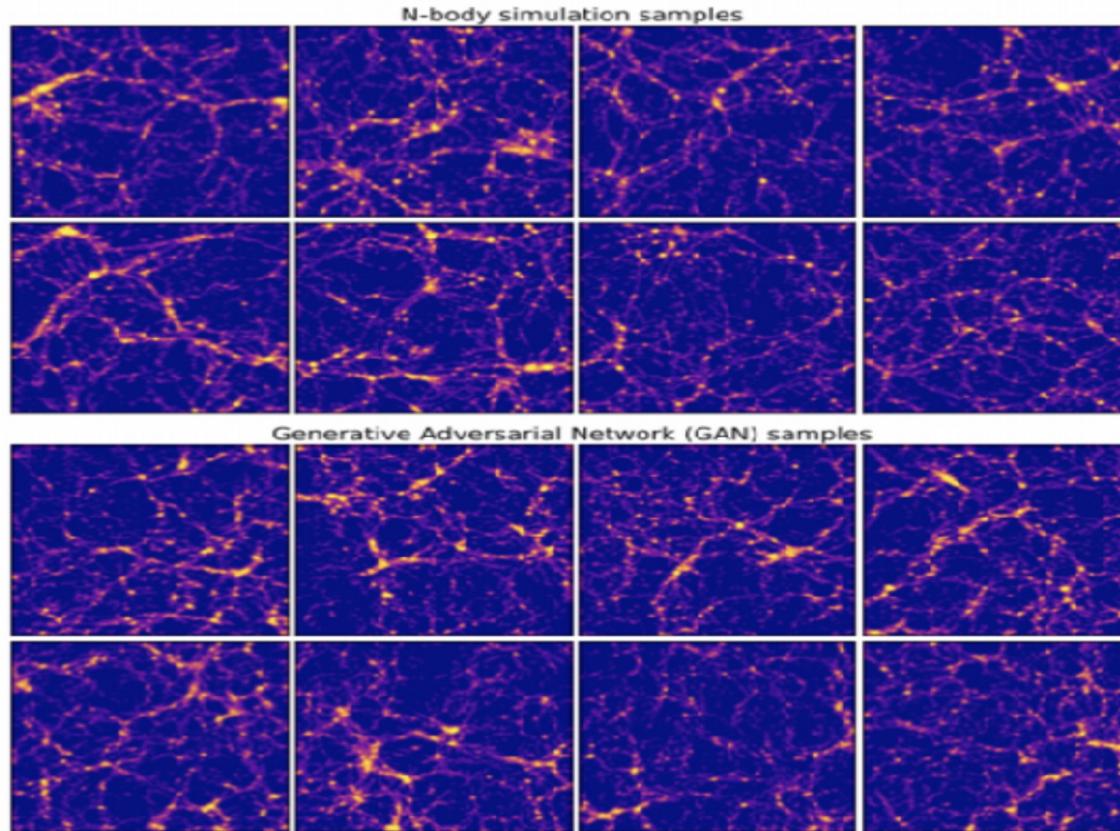
# Generative adversarial network (GAN)





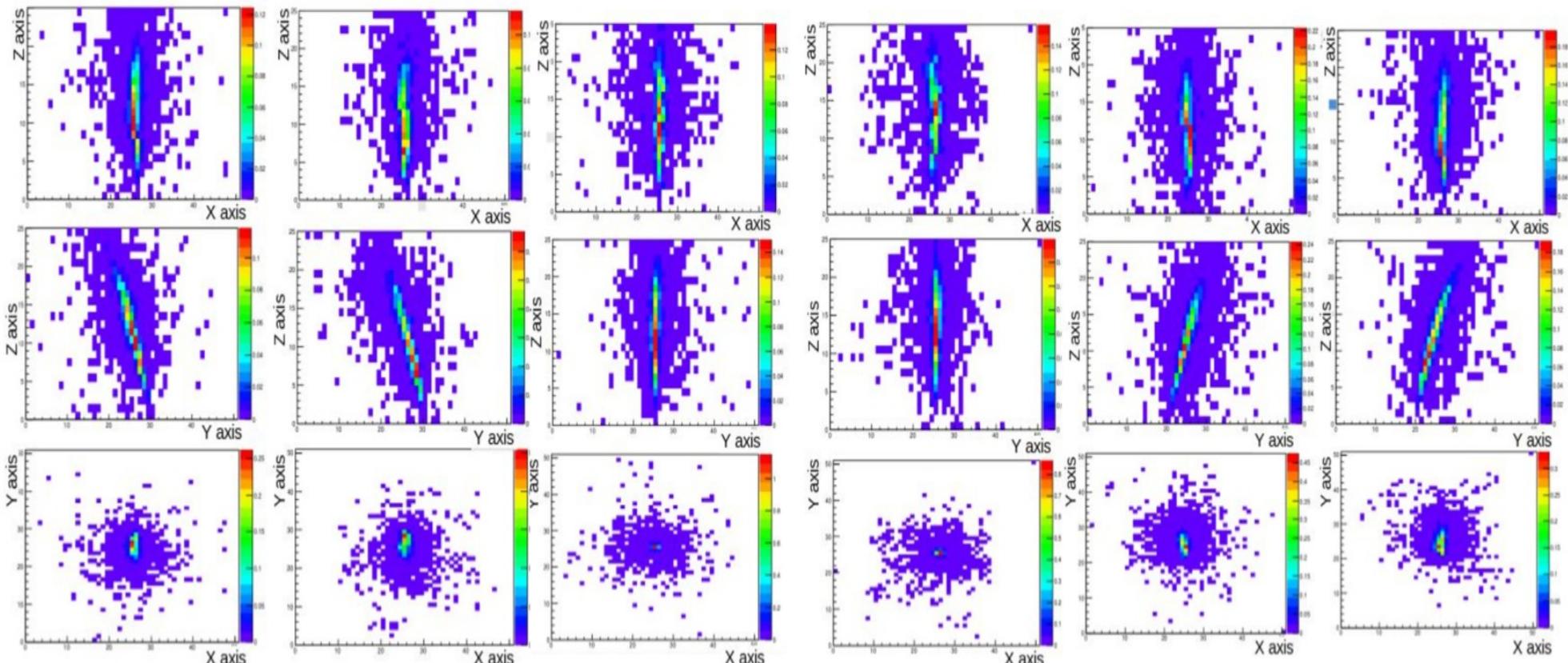


# Generative Models in Cosmology



Rodríguez et al,  
arXiv:1801.09070

**Figure 3:** Samples from N-body simulation (top two rows) and from GAN (bottom two rows) for the box size of 100 Mpc. In this figure, transformation S1 with  $k = 7$  was applied.



G4

GAN

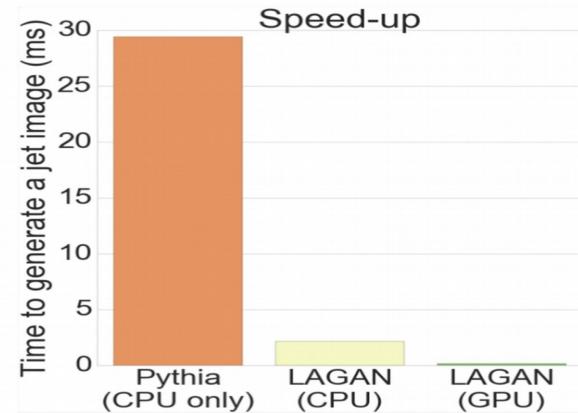
G4

GAN

G4

GAN

- **Really fast simulation**  
in fact  $10^2 \div 10^5$  faster than Geant
- Save space
- Rare events

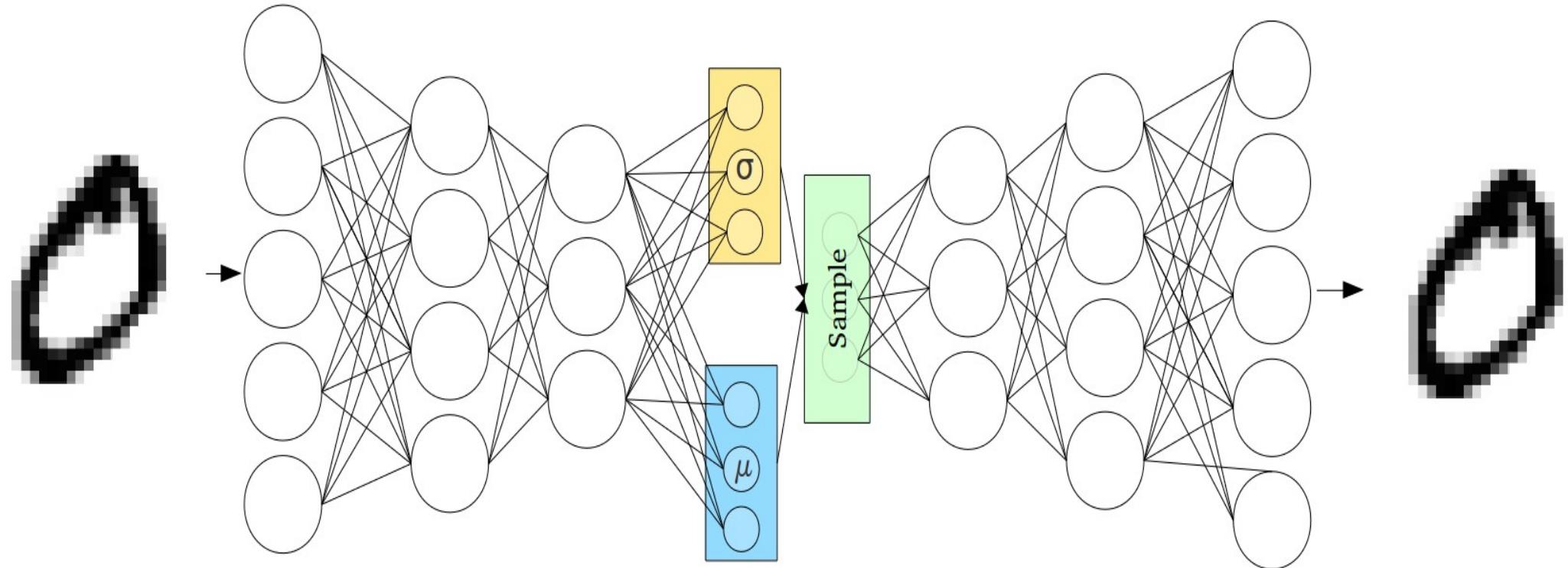


| Generation Method | Hardware | Batch Size | milliseconds/shower |
|-------------------|----------|------------|---------------------|
| GEANT4            | CPU      | N/A        | 1772                |
|                   |          | 1          | 13.1                |
|                   |          | 10         | 5.11                |
|                   |          | 128        | 2.19                |
|                   |          | 1024       | 2.03                |
|                   | GPU      | 1          | 14.5                |
|                   |          | 4          | 3.68                |
|                   |          | 128        | 0.021               |
|                   |          | 512        | 0.014               |
|                   |          | 1024       | 0.012               |

- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorical GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWWN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- IcGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

# Variational autoencoder (VAE)

# Variational autoencoder (VAE)



# ML Frameworks

# The Toolkit for Multivariate Data Analysis with ROOT (TMVA)



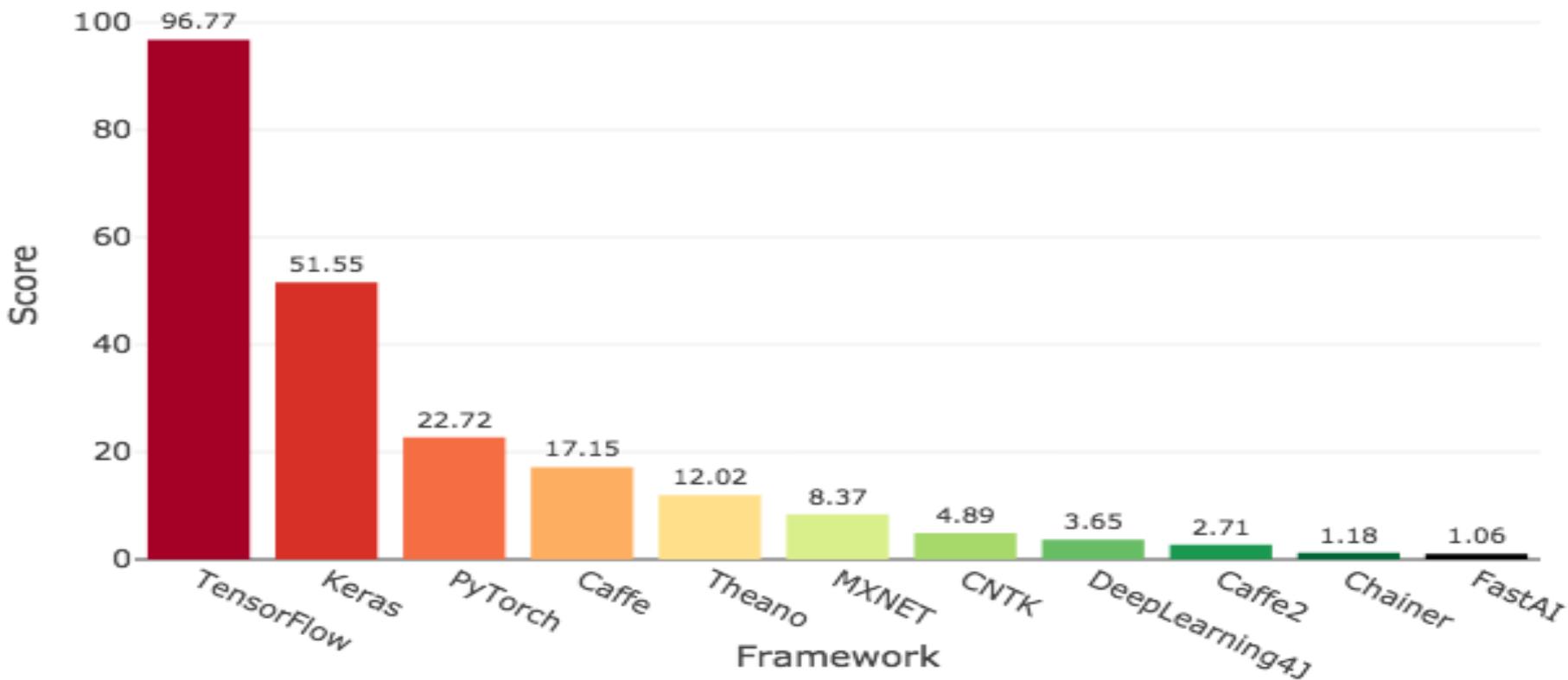
- ROOT: is the analysis framework used by most (HEP)-physicists
- Idea: rather than just implementing new MVA techniques and making them available in ROOT:
  - Have one common platform / interface for all MVA classifiers
  - Have common data pre-processing capabilities
  - Train and test all classifiers on same data sample and evaluate consistently
  - Provide common analysis (ROOT scripts) and application framework
  - Provide access with and without ROOT, through macros, C++ executables or python

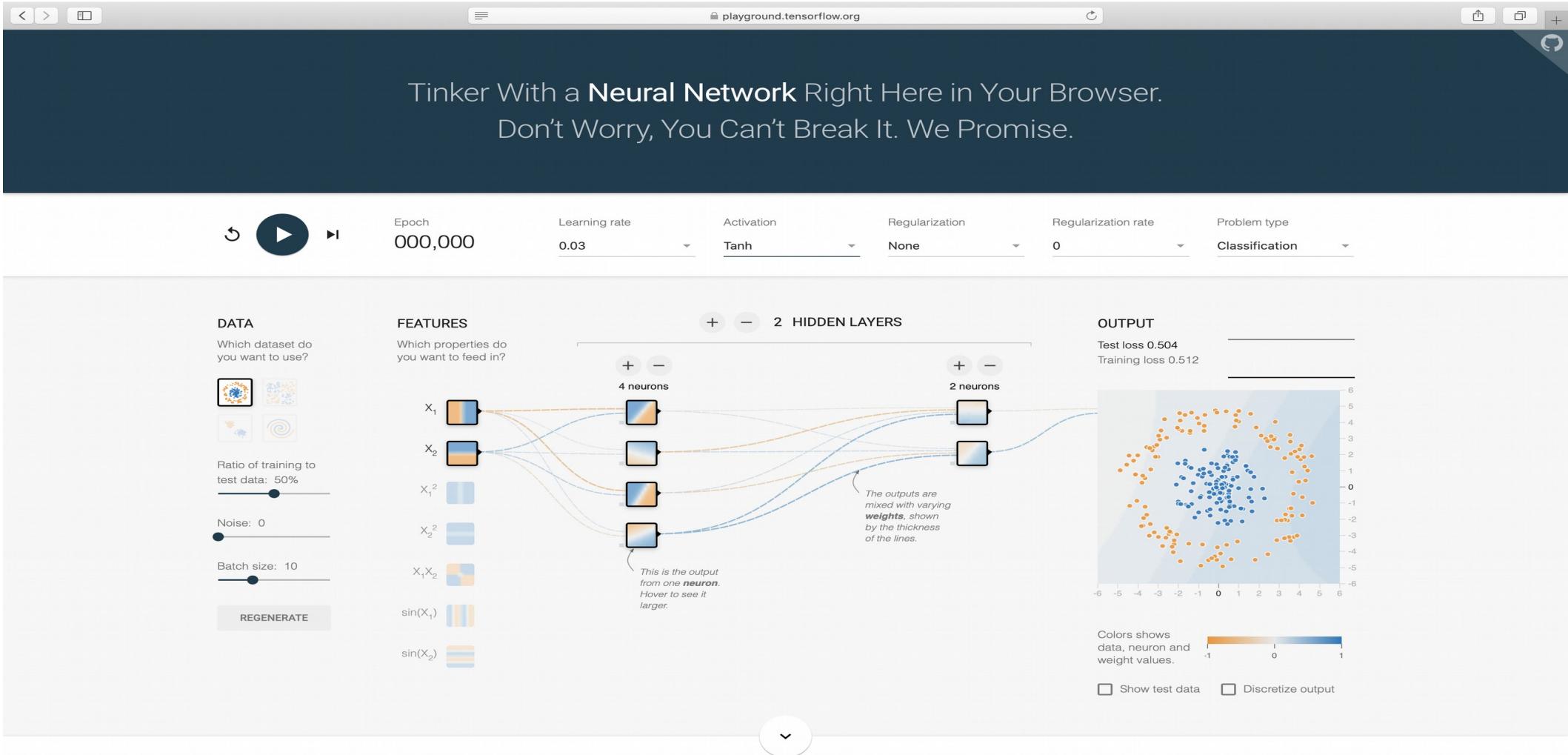
Information from Multivariate Data Analysis with TMVA (June 19, 2008)  
by Andreas Hoecker

- TensorFlow
- Keras (TF & Theano back-end)
- PyTorch
- ? Theano



## Deep Learning Framework Power Scores 2018





localhost

jupyter ann Last Checkpoint: Last Monday at 2:56 AM (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from scipy import stats

import tensorflow.keras as keras
import tensorflow.keras.backend as k

from tensorflow.keras.models import Model, load_model, Sequential
from tensorflow.keras.layers import Input, Dense, Activation, BatchNormalization, Dropout
from tensorflow.keras.callbacks import ReduceLROnPlateau, ModelCheckpoint, Callback, TensorBoard
from tensorflow.keras.utils import Sequence, plot_model

from IPython.display import Image
```

In [2]:

```
from root_pandas import read_root
Welcome to JupyterROOT 6.18/04
```

In [3]:

```
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import Ridge
```

In [4]:

```
fDataName = "./gun_k0L_allgev_FTFP_BERT_90000evt_IID_l5_v02steel.root"
fDataName_mc = "./mc_gun_k0L_allgev_FTFP_BERT_90000evt_IID_l5_v02steel.root"
fTreeName = "tree"
data = read_root(fDataName, fTreeName)
data_mc = read_root(fDataName_mc, fTreeName)
```

In [5]:

```
x = data
y = data_mc
```

In [6]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size = 0.1)
```

In [7]:

```
x_train.shape
```

Out[7]:

```
(7962, 10)
```

In [8]:

```
sns.pairplot(data_mc, kind="kde")
```

Out[8]:

```
<seaborn.axisgrid.PairGrid at 0x15478cad0>
```

```

In [ ]: def metric1(y_true, y_pred):
         return ( y_pred - y_true ) / ( y_true )

In [ ]: def metric2(y_true, y_pred):
         return k.mean( ( y_pred - y_true ) / ( y_true ) )

In [ ]: model = Sequential()
        model.add(Dense(64, activation='tanh', input_dim=x_train.shape[1]))
        model.add(Dropout(0.5))
        model.add(Dense(32, activation='tanh'))
        model.add(Dropout(0.5))
        model.add(Dense(4, activation='tanh'))
        model.add(Dropout(0.5))
        model.add(Dense(1))

        ...
        a = Input(shape=x_train.shape[1])
        b = Dense(8)(a)
        b = Dropout(rate=0.5)(b)
        b = Activation('tanh')(b)
        b = Dense(4)(a)
        b = Dropout(rate=0.5)(b)
        b = Activation('tanh')(b)
        c = Dense(1)(b)
        model = Model(inputs = a, outputs = c)
        ...

        model.compile(loss='mean_squared_error', metrics=[metric1, metric2], optimizer='adam')
        #model.compile(loss=loss1, metrics=[metric1, metric2], optimizer='adam')

        model.summary()

In [ ]: history = model.fit(x_train, y_train, epochs = 200, validation_data = (x_test, y_test))

In [ ]: #a = model.evaluate( x_train, y_train )
        #b = model.evaluate( x_test, y_test )

In [ ]: history.history.keys()

In [ ]: plt.plot(history.history['loss'])
        plt.plot(history.history['val_loss'])
        plt.title('Model loss')
        plt.ylabel('Loss')
        plt.xlabel('Epoch')

```

### 3.3 Building our Neural Network

Our NN consists of input, output and 1 hidden layer. We are using ReLU as activation function of the hidden layer and softmax for our output layer. As an additional bonus we will use Dropout — simple way to reduce overfitting during the training of our network. Let's wrap our model in a little helper function:

```
def multilayer_perceptron(x, weights, biases, keep_prob):
    layer_1 = tf.add(tf.matmul(x, weights['h1']), biases['b1'])
    layer_1 = tf.nn.relu(layer_1)
    layer_1 = tf.nn.dropout(layer_1, keep_prob)
    out_layer = tf.matmul(layer_1, weights['out']) + biases['out']
    return out_layer
```

Let's set the number of neurons in the hidden layer to 38 and randomly initialize the weights and biases considering their proper dimensions:

```
n_hidden_1 = 38
n_input = train_x.shape[1]
n_classes = train_y.shape[1]

weights = {
    'h1': tf.Variable(tf.random_normal([n_input, n_hidden_1])),
    'out': tf.Variable(tf.random_normal([n_hidden_1, n_classes]))
}

biases = {
    'b1': tf.Variable(tf.random_normal([n_hidden_1])),
    'out': tf.Variable(tf.random_normal([n_classes]))
}

keep_prob = tf.placeholder("float")
```

We will train our model for 5,000 epochs (training steps) with a batch size

```
31
32 # the model will be a sequence of layers
33 model = torch.nn.Sequential()
34
35
36 # ANN with layers [784] -> [500] -> [300] -> [10]
37 # NOTE: the "p" is p_drop, not p_keep
38 model.add_module("dropout1", torch.nn.Dropout(p=0.2))
39 model.add_module("dense1", torch.nn.Linear(D, 500))
40 model.add_module("relu1", torch.nn.ReLU())
41 model.add_module("dropout2", torch.nn.Dropout(p=0.5))
42 model.add_module("dense2", torch.nn.Linear(500, 300))
43 model.add_module("relu2", torch.nn.ReLU())
44 model.add_module("dropout3", torch.nn.Dropout(p=0.5))
45 model.add_module("dense3", torch.nn.Linear(300, K))
46 # Note: no final softmax!
47 # just like Tensorflow, it's included in cross-entropy function
48
49
50 # define a loss function
51 # other loss functions can be found here:
52 # http://pytorch.org/docs/master/nn.html#loss-functions
53 loss = torch.nn.CrossEntropyLoss(size_average=True)
54 # Note: this returns a function!
55 # e.g. use it like: loss(logits, labels)
```



# sklearn

# Машинное обучение в несколько строк

```
from sklearn.linear_model import LogisticRegression  
  
model = LogisticRegression()  
model.fit(X_train, y_train)  
predictions = model.predict(X_test)
```

# scikit-learn

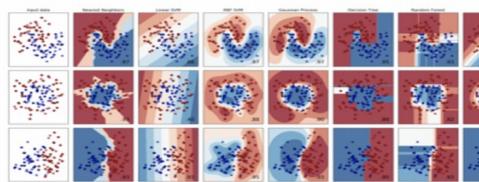
Machine Learning in Python

[Getting Started](#)[What's New in 0.22.2](#)[GitHub](#)

## Classification

Identifying which category an object belongs to.

**Applications:** Spam detection, image recognition.  
**Algorithms:** SVM, nearest neighbors, random forest, and more...

[Examples](#)

## Dimensionality reduction

Reducing the number of random variables to consider.

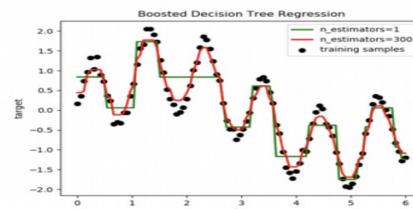
**Applications:** Visualization, Increased efficiency  
**Algorithms:** k-Means, feature selection, non-negative matrix factorization, and more...

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

## Regression

Predicting a continuous-valued attribute associated with an object.

**Applications:** Drug response, Stock prices.  
**Algorithms:** SVR, nearest neighbors, random forest, and more...

[Examples](#)

## Clustering

Automatic grouping of similar objects into sets.

**Applications:** Customer segmentation, Grouping experiment outcomes  
**Algorithms:** k-Means, spectral clustering, mean-shift, and more...

[Examples](#)

## Model selection

Comparing, validating and choosing parameters and models.

**Applications:** Improved accuracy via parameter tuning  
**Algorithms:** grid search, cross validation, metrics, and more...

## Preprocessing

Feature extraction and normalization.

**Applications:** Transforming input data such as text for use with machine learning algorithms.  
**Algorithms:** preprocessing, feature extraction, and more...

scikit-learn.org

scikit-learn

Install User Guide API Examples More ▾

Prev Up Next

scikit-learn 0.22.2 Other versions

Please cite us if you use the software.

Two-class AdaBoost

Note: Click [here](#) to download the full example code or to run this example in your browser via Binder

## Two-class AdaBoost

This example fits an AdaBoosted decision stump on a non-linearly separable classification dataset composed of two "Gaussian quantiles" clusters (see `sklearn.datasets.make_gaussian_quantiles`) and plots the decision boundary and decision scores. The distributions of decision scores are shown separately for samples of class A and B. The predicted class label for each sample is determined by the sign of the decision score. Samples with decision scores greater than zero are classified as B, and are otherwise classified as A. The magnitude of a decision score determines the degree of likeness with the predicted class label. Additionally, a new dataset could be constructed containing a desired purity of class B, for example, by only selecting samples with a decision score above some value.

Decision Boundary

Decision Scores

```
print(__doc__)

# Author: Noel Dawe <noel.dawe@gmail.com>
#
# License: BSD 3 clause

import numpy as np
import matplotlib.pyplot as plt

from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import make_gaussian_quantiles

# Construct dataset
```

Toggle Menu

scikit-learn.org

Install User Guide API Examples More ▾ Go

Prev Up Next

scikit-learn 0.22.2 Other versions

Please cite us if you use the software.

sklearn.tree.DecisionTreeRegressor  
Examples using sklearn.tree.DecisionTreeRegressor

## sklearn.tree.DecisionTreeRegressor

```
class sklearn.tree.DecisionTreeRegressor(criterion='mse', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, presort='deprecated', ccp_alpha=0.0)
```

[source]

A decision tree regressor.

Read more in the [User Guide](#).

**Parameters:**

**criterion : {“mse”, “friedman\_mse”, “mae”}, default=“mse”**  
The function to measure the quality of a split. Supported criteria are “mse” for the mean squared error, which is equal to variance reduction as feature selection criterion and minimizes the L2 loss using the mean of each terminal node, “friedman\_mse”, which uses mean squared error with Friedman’s improvement score for potential splits, and “mae” for the mean absolute error, which minimizes the L1 loss using the median of each terminal node.  
*New in version 0.18:* Mean Absolute Error (MAE) criterion.

**splitter : {“best”, “random”}, default=“best”**  
The strategy used to choose the split at each node. Supported strategies are “best” to choose the best split and “random” to choose the best random split.

**max\_depth : int, default=None**  
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples.

**min\_samples\_split : int or float, default=2**  
The minimum number of samples required to split an internal node:

- If int, then consider min\_samples\_split as the minimum number.
- If float, then min\_samples\_split is a fraction and ceil(min\_samples\_split \* n\_samples) are the minimum number of samples for each split.

*Changed in version 0.18:* Added float values for fractions.

**min\_samples\_leaf : int or float, default=1**  
The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min\_samples\_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

- If int, then consider min\_samples\_leaf as the minimum number.

**Toggle Menu**

# Jupyter Notebook / Jupyterlab

File Edit View Insert Cell Kernel Help



Code Cell Toolbar: None



## Simple spectral analysis

An illustration of the [Discrete Fourier Transform](#) using windowing, to reveal the frequency content of a sound signal.

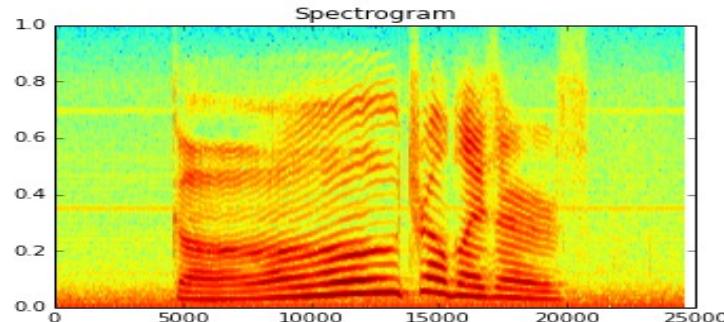
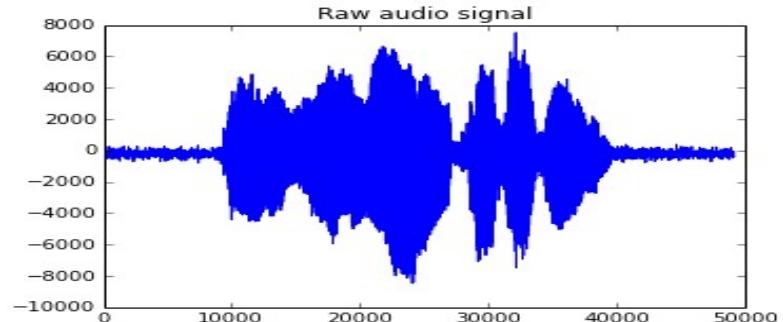
$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \quad k = 0, \dots, N - 1$$

We begin by loading a datafile using SciPy's audio file support:

```
In [1]: from scipy.io import wavfile  
rate, x = wavfile.read('test_mono.wav')
```

And we can easily view its spectral structure using matplotlib's builtin specgram routine:

```
In [2]: %matplotlib inline  
from matplotlib import pyplot as plt  
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))  
ax1.plot(x); ax1.set_title('Raw audio signal')  
ax2.specgram(x); ax2.set_title('Spectrogram');
```



# Kaggle

# Kaggle

The screenshot shows the Kaggle homepage. At the top, there's a navigation bar with links for Compete, Datasets, Notebooks, Discuss, Courses, and Register. A search bar and sign-in options are also present. The main banner features a call to action: "Help us better understand COVID-19". It includes a description: "There is a large body of research and data around COVID-19. Help the global community better understand the disease by getting involved on Kaggle.", two buttons ("Get Started" and "View Contributions"), and a central graphic of a smartphone surrounded by icons like a laptop, a clock, and a pencil, all set against a background of concentric circles and dots. Below this, a section titled "Start with more than a blinking cursor" describes Kaggle's Jupyter Notebook environment and free GPUs. It includes a "REGISTER WITH GOOGLE" button and a "Register with Email" link. Further down, a section highlights Kaggle's extensive resources: "Inside Kaggle you'll find all the code & data you need to do your data science work. Use over 19,000 public datasets and 200,000 public notebooks to conquer any analysis in no time." At the bottom, a cookie consent banner states: "We use cookies on Kaggle to deliver our services, analyze web traffic, and improve your experience on the site. By using Kaggle, you agree to our use of cookies." with "Got it" and "Learn more" buttons.

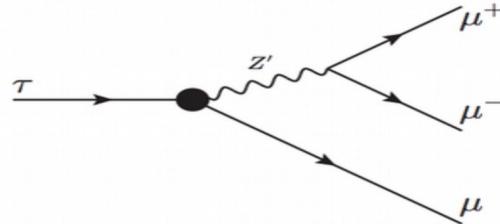
# Kaggle

- one of the largest data communities in the world
- ML competitions
- public datasets and kernels (code snippets)
- short-form AI education

open data and baselines for practice!

- 2014 Higgs Boson challenge by ATLAS [1]
- 2015  $\tau \rightarrow \mu\mu\mu$  by LHCb and Yandex [2]
- 2018 TrackML by CERN [3], [4]

huge impact on ML community in HEP



# GitHub / GitLab



**Ete Remi**  
rete

[Follow](#)

HEP computing

 DESY

 Hamburg, Germany

 remi.ete@gmail.com

 <http://lyorete.sytes.net>

[Block or report user](#)

[Overview](#)   [Repositories 63](#)   [Projects 0](#)   [Stars 18](#)   [Followers 16](#)   [Following 43](#)

#### Pinned

 [iLCSoft/MarlinMT](#)

Modular Analysis and Reconstruction for the LInear Collider - Multi-threaded version

 C++  2

 [iLCSoft/SIO](#)

Simple IO package

 C  2

 [iLCSoft/LCIO](#)

Linear Collider I/O

 C++  7  15

 [iLCSoft/LCCalibration](#)

Automated energy calibration procedure for Linear Colliders

 Python  3

 [LCEvent](#)

Linear Collider event display based on LCIO and experimental Eve7

 C++

 [DQM4hep/DQM4hep](#)

Data quality monitoring for high energy physics. Doc available at

 C++  2  6

953 contributions in the last year



Contribution activity

April 1, 2020

2020

2019

2018

2017

March 2020

rete has no activity yet for this period.

github.com

Search or jump to... / Pull requests Issues Marketplace Explore

rete / DDMarlinPandora forked from iLCSoft/DDMarlinPandora

Watch 1 Star 0 Fork 6

Code Pull requests 0 Actions Projects 0 Wiki Security Insights

Branch: master DDMarlinPandora / src / DDCaloDigi.cc Find file Copy path

andresailer DDCaloDigi: add cleanup of PpdDigi objects b9f3b0a on Jul 7, 2017

5 contributors

2116 lines (1748 sloc) | 84.9 KB Raw Blame History

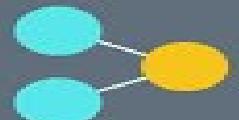
```
1 // Calorimeter digitiser for the IDC ECAL and HCAL
2 // For other detectors/models SimpleCaloDigi should be used
3 #include "DDCaloDigi.h"
4 #include <EVENT/LCCollection.h>
5 #include <EVENT/SimCalorimeterHit.h>
6 #include <IMPL/SimCalorimeterHitImpl.h>
7 #include <IMPL/CalorimeterHitImpl.h>
8 #include <IMPL/LCCollectionVec.h>
9 #include <IMPL/LCRelationImpl.h>
10 #include <marlin/Global.h>
11
12
13 #include <EVENT/LCParameters.h>
14 #include <UTIL/CellIDDecoder.h>
15 #include <UTIL/CellIDEncoder.h>
16 #include <iostream>
17 #include <string>
18 #include <algorithm>
19 #include <assert.h>
20 #include <cmath>
21 #include <cstdlib>
22 #include <sstream>
23
24 #include "CLHEP/Random/RandPoisson.h"
25 #include "CLHEP/Random/RandGauss.h"
26 #include "CLHEP/Random/RandFlat.h"
27
28 #include "DD4hep/Factories.h"
29 #include "DD4hep/DD4hepUnits.h"
30 #include "DDRec/DetectorData.h"
31 #include "DD4hep/DetType.h"
```

# References

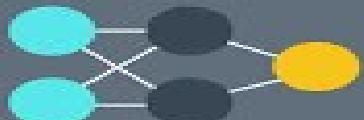
- [https://gitlab.cern.ch/ofilatov/intro\\_to\\_ml\\_lpi\\_2019](https://gitlab.cern.ch/ofilatov/intro_to_ml_lpi_2019)
- <https://mosphys.ru/2019/> - *Fedor Ratnikov*
- <https://mosphys.ru/> - *Andrey Ustyuzhanin*
- *Machine learning in high energy physics 2019 (mlhep2019)*
- .....

# **Backup Slides**

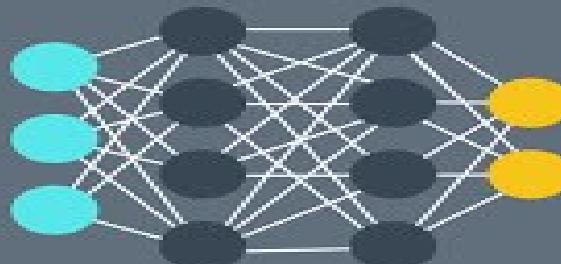
# NEURAL NETWORK ARCHITECTURE TYPES



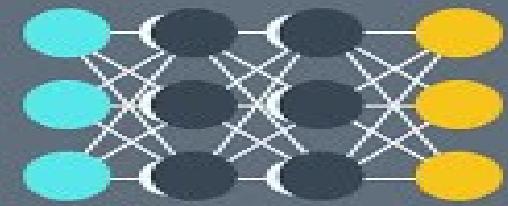
SINGLE LAYER PERCEPTRON



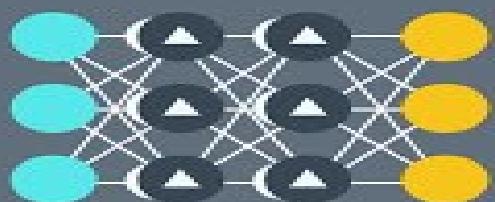
RADIAL BASIS NETWORK



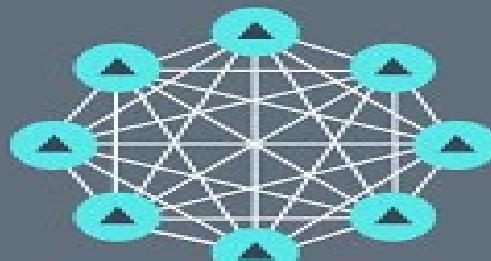
MULTI LAYER PERCEPTRON



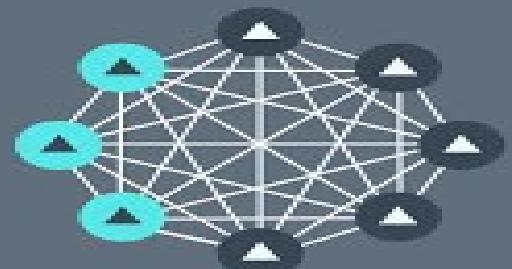
RECURRENT NEURAL NETWORK



LSTM RECURRENT NEURAL NETWORK



HOPFIELD NETWORK



BOLTZMANN MACHINE



INPUT UNIT



OUTPUT UNIT



HIDDEN UNIT



FEEDBACK WITH MEMORY UNIT



BACKFED INPUT UNIT

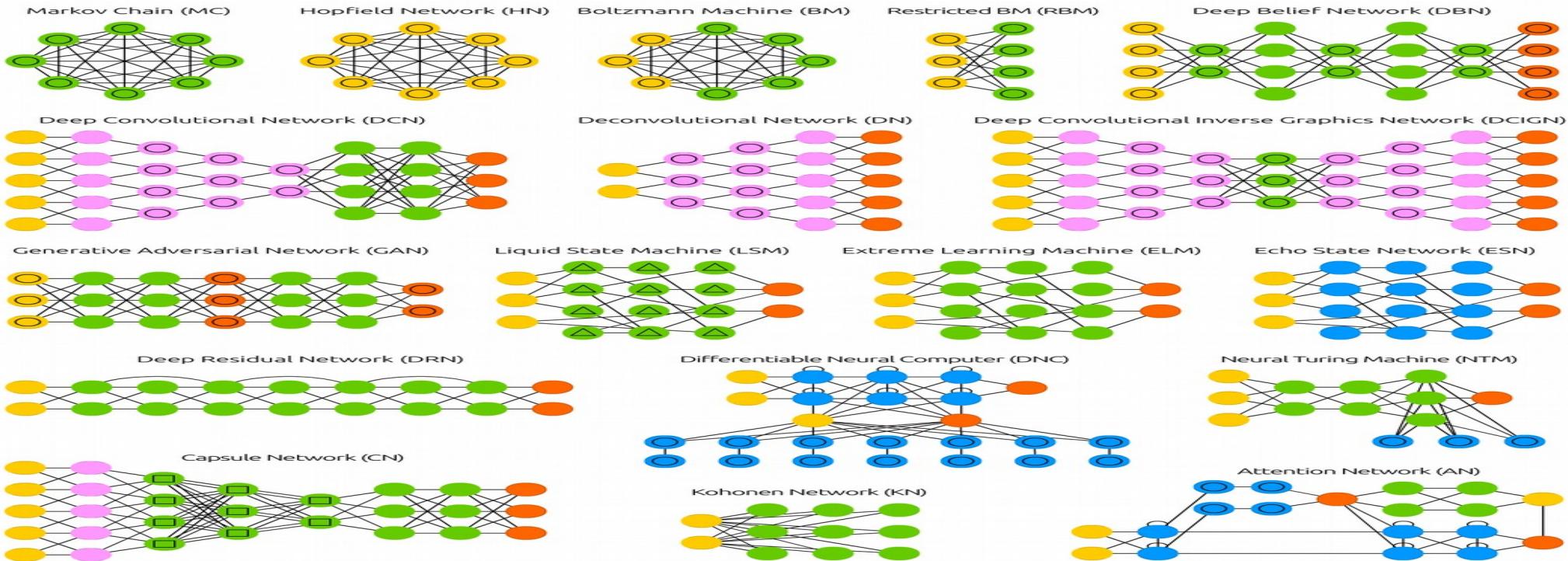


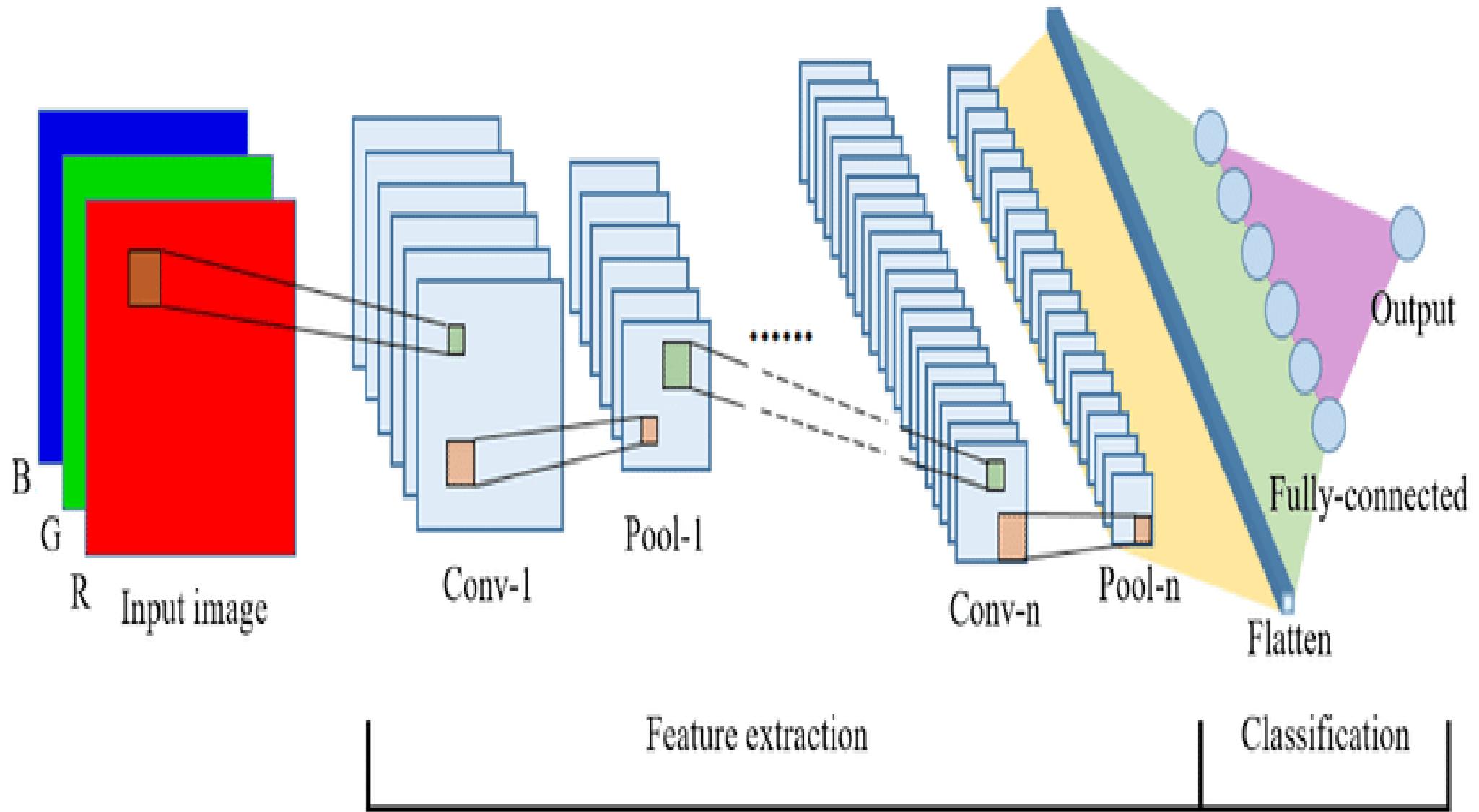
PROBABILISTIC HIDDEN UNIT

A mostly complete chart of  
**Neural Networks**

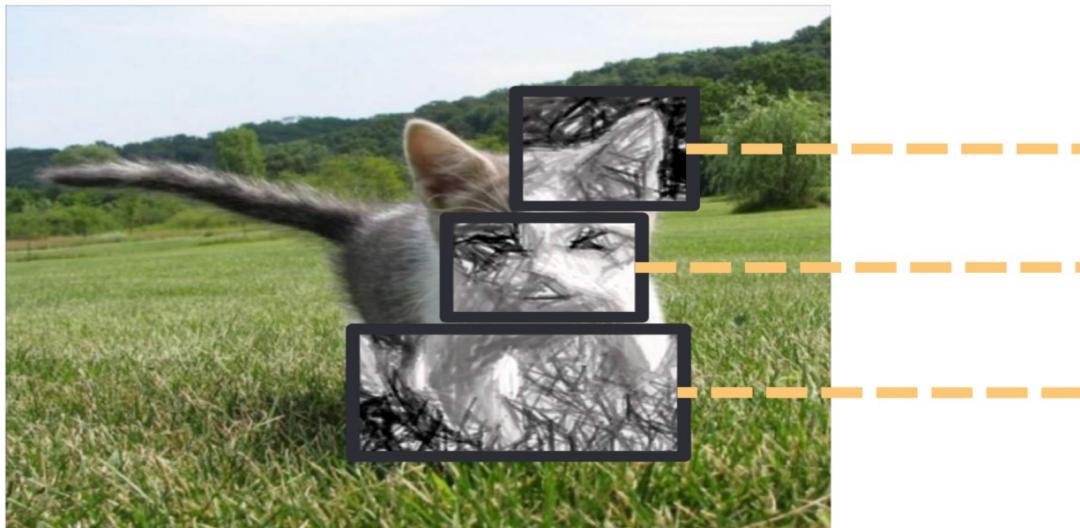
©2019 Fjodor van Veen & Stefan Leijnen asimovinstitute.org

- Input Cell
- Backfed Input Cell
- △ Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- △ Spiking Hidden Cell
- Capsule Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- △ Gated Memory Cell
- Kernel
- Convolution or Pool





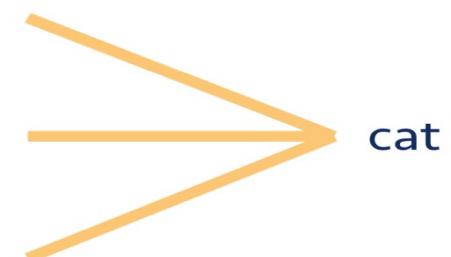
# Convolutional NN patterns



filter 1

filter 2

filter 3

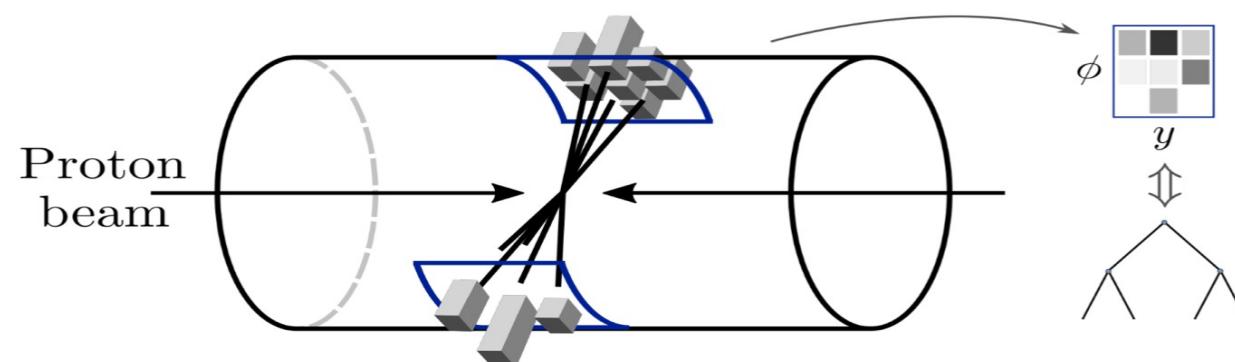


# What's a jet, anyway?

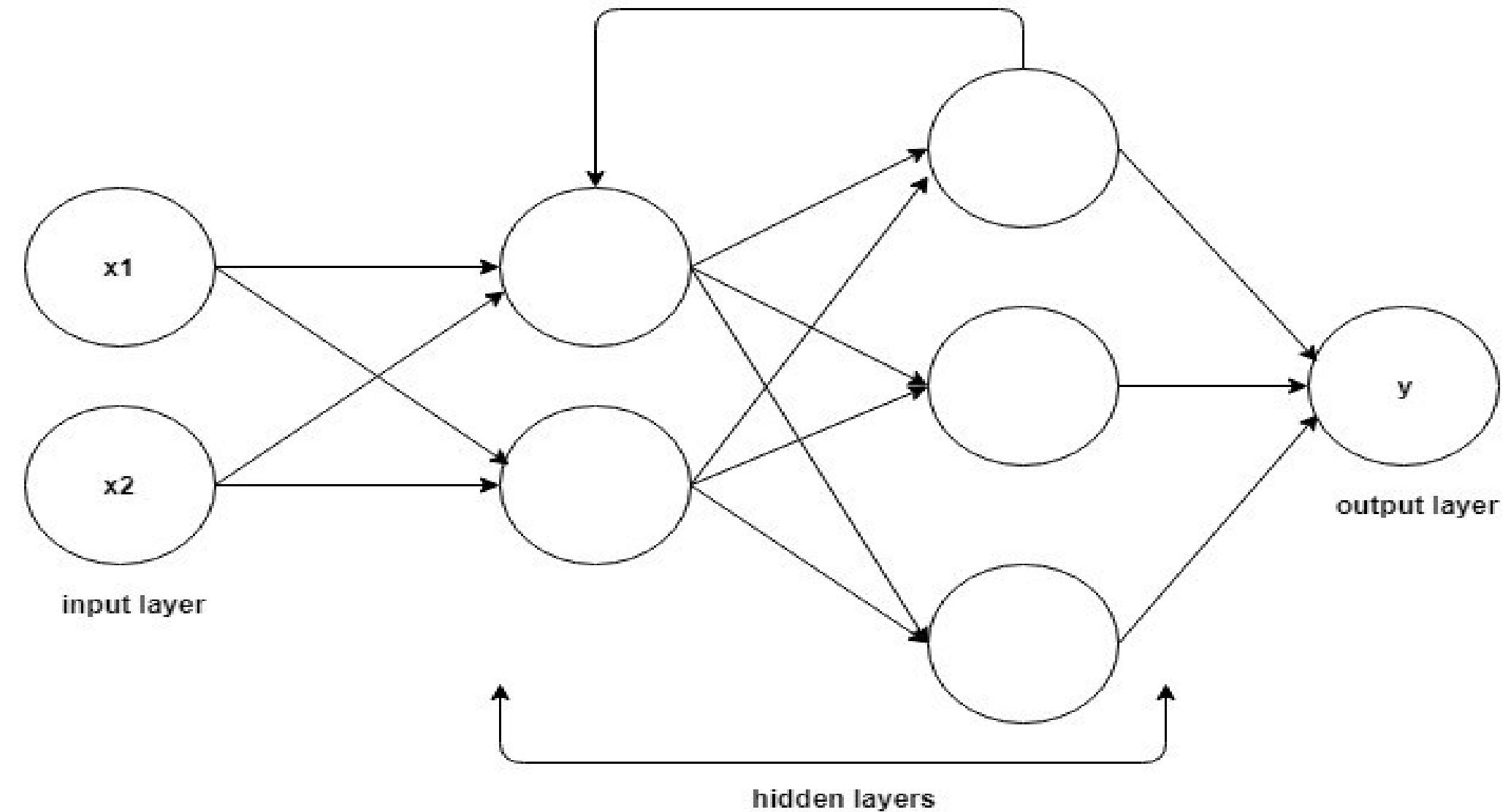
A jet can be described in a variety of ways

- ▶ Sum of its particle constituents
- ▶ Image of energy deposits in rapidity-azimuth
- ▶ Complete basis of observables (e.g. energy flow polynomials)
- ▶ Image of primary Lund declustering sequence
- ▶ Binary tree associated with the jet clustering sequence
- ▶ ...

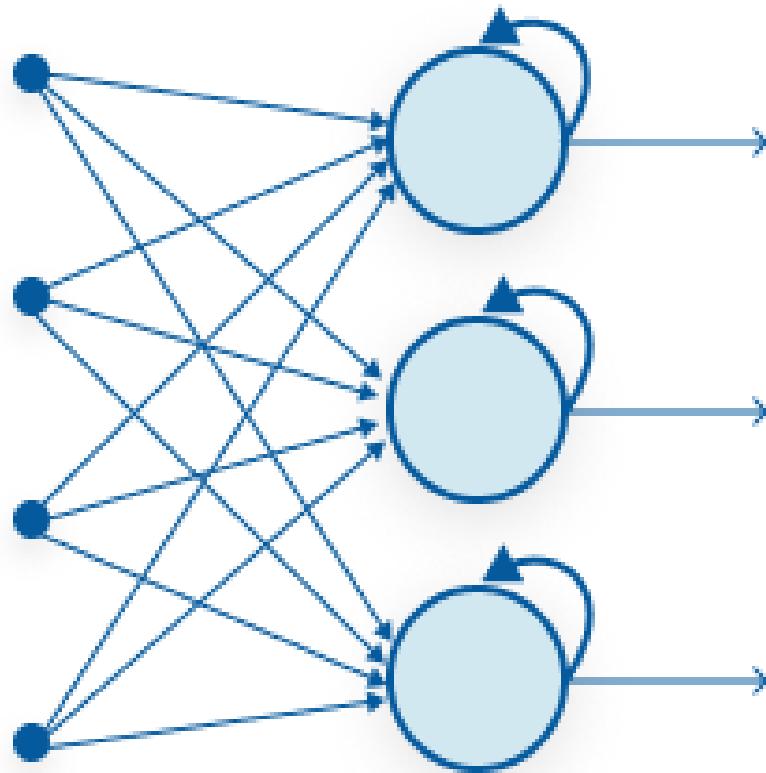
Most of them involve some trade-off between interpretability, representability, robustness and performance.



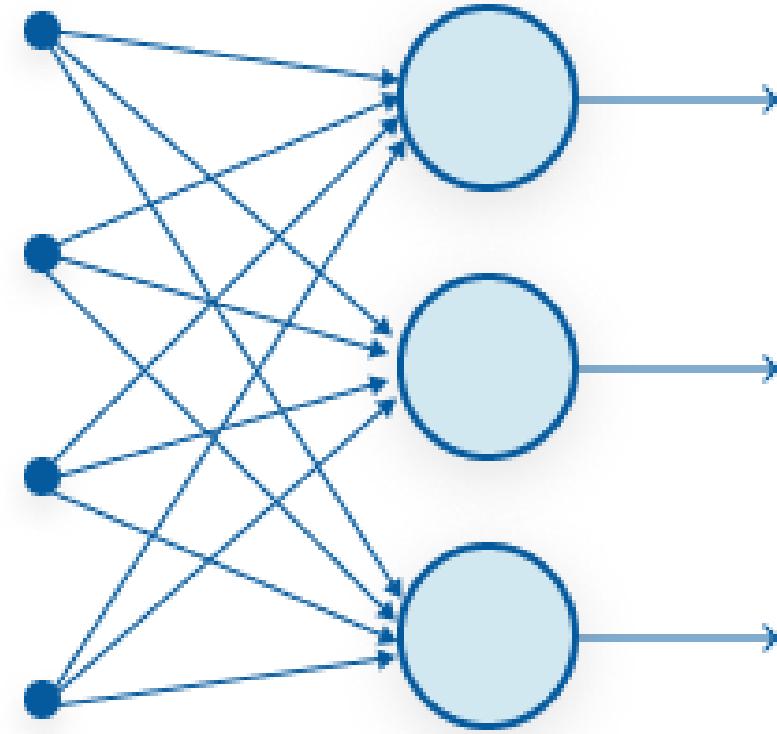
Recurrent network



# Recurrent Neural Network structure



Recurrent Neural Network



Feed-Forward Neural Network

# Codecademy

The screenshot shows the Codecademy website homepage. At the top, there's a navigation bar with links for 'Catalog', 'Pricing', and 'For Business'. On the right side of the bar are buttons for 'Try Pro For Free', 'Log in', and 'Sign up' (which is highlighted in purple). Below the navigation, there's a large image of a smiling man wearing headphones, with a purple overlay containing the text: 'Join the Millions Learning to Code with Codecademy'. To the right of this image is a 'Get Started For Free' form with fields for 'Email' and 'Password', and a 'Start coding now' button. Below the form, there's a note about agreeing to the Terms of Service and Privacy Policy. Further down, there's a section for 'Or, use another account:' with icons for LinkedIn, Google, Facebook, and GitHub. At the bottom of the page, there's a purple bar with the text 'Not sure'.

codecademy

Catalog Pricing For Business

Try Pro For Free Log in Sign up

Join the Millions Learning to Code with Codecademy

Get Started For Free

Email

Password

Start coding now

By signing up for Codecademy, you agree to Codecademy's [Terms of Service](#) & [Privacy Policy](#).

Or, use another account:

LinkedIn Google Facebook GitHub

Not sure

# Linear Regression

Model:

$$x \longrightarrow wx + b \longrightarrow y^{\text{pred}}$$

Objective function:

$$L = \frac{1}{N} \sum_i (y_i - y_i^{\text{pred}})^2$$

Optimization (iterative):

$$w_{i+1} := w_i - \alpha \cdot \frac{\partial L}{\partial w_i}$$

# Recap: Gradient Descent

Update:

$$w_{i+1} := w_i - \alpha \cdot \frac{\partial L}{\partial w_i}$$

- $\alpha$  – learning rate  $\alpha << 1$
- $L$  – loss function

