Deep Learning 101

Astro Hack Week 2019 - Cambridge UK

Gabriella Contardo (Flatiron Institute, Center for Computational Astrophysics)
gcontardo@flatironinstitute.org

Francois Lannuse (UC Berkeley) flanusse@berkeley.edu

Outline of tutorial part I

- Machine learning principles
- Why *deep learning?*
- Neural networks:
  - Components, training
  - Fully Connected / Multi-Layer Perceptron (MLP)
  - Convolutional neural networks
- Build your own model in tensorflow !

A bit of definitions

Machine Learning (ML) :

- Relying on data analysis to automate model building
- Model learns from data, extracts patterns, and makes some ‘prediction’
- E.g. SVM, Random Forest, KNN, Deep learning….

Deep Learning :

- Subgroup of ML based on artificial neural networks
- Strongly rely on “pattern” extraction / representation learning

Learning can be **supervised, unsupervised,** or **semi-supervised.**

Some Machine Learning principles

3 main “components”:

- Data
- Machine or model
- Criterion (learning and evaluation)

Goal : Automatically extract relevant information from the data that *generalize* well to infer on new data (‘test’ or evaluation data from similar distribution, but different from training examples)

Learn a model on training data to optimize some criterion, then infer on test data.

Supervised Learning

Training data is composed of

- Examples (inputs)
- ‘Targets’ --or label(s)-- (i.e outputs) for each example.

E.g. image classification.

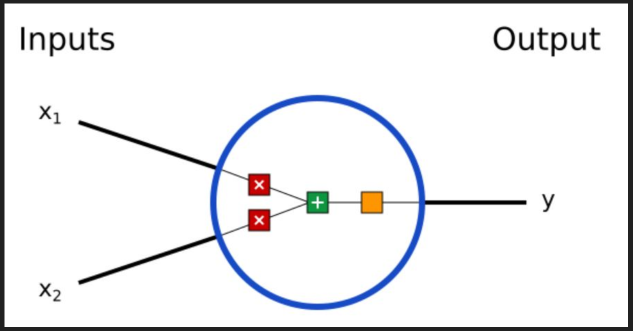
Different type of targets => different learning problem:

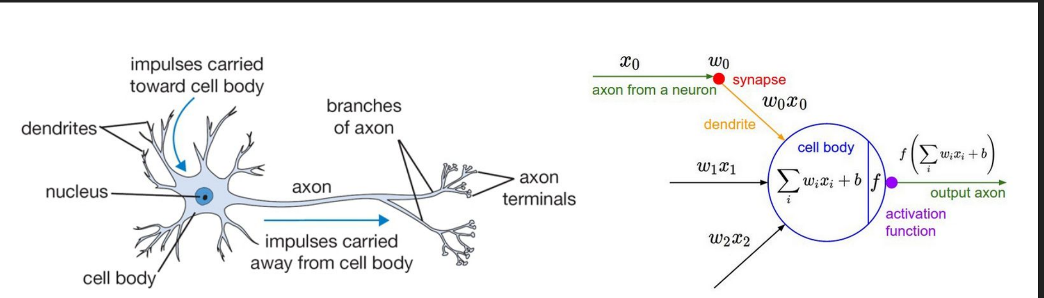
- Limited set of values, categorical : classification (multi-class, multi-label,...)
- Continuous values : regression
- Other type e.g. noisy targets, weak supervision etc.

Goal: Find an approximation of the labeling process.

What are neural networks?

- (Highly) parametric non-linear function approximators.
- Loosely inspired from brain : network of interconnected neurons that “activate” or not.
- Composition of stacked ‘*layers’* of weights (parameters) producing *neurons.*



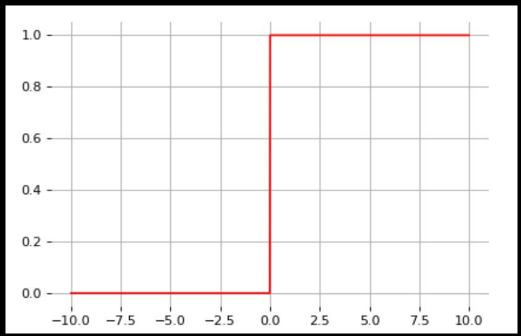


biological neuron (left) and a common mathematical model (right)

Let’s start with a very simple one: Perceptron

- Like linear regression: $\sum_i w_i x_i$
- But with an additional ***activation function***

Here : “step” function, threshold to obtain binary outputs.



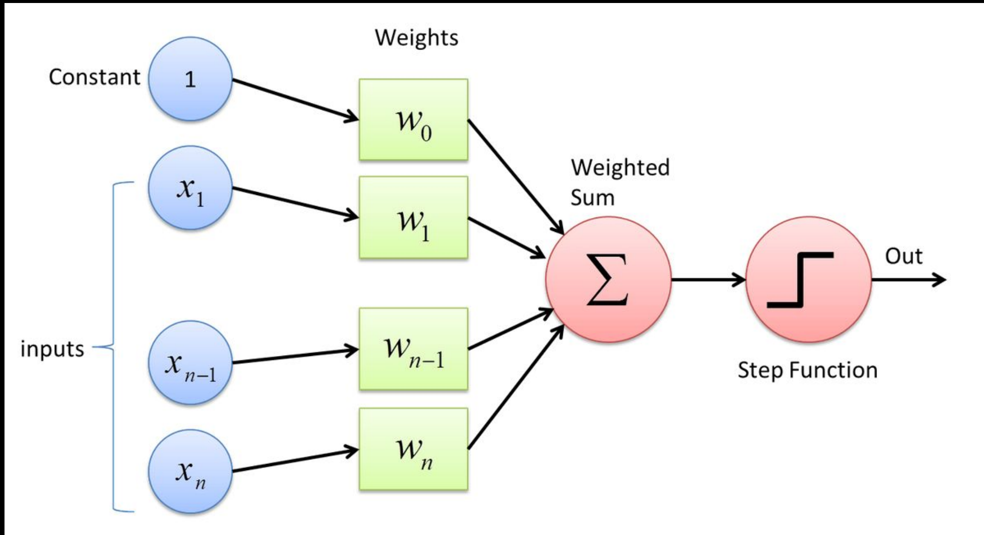


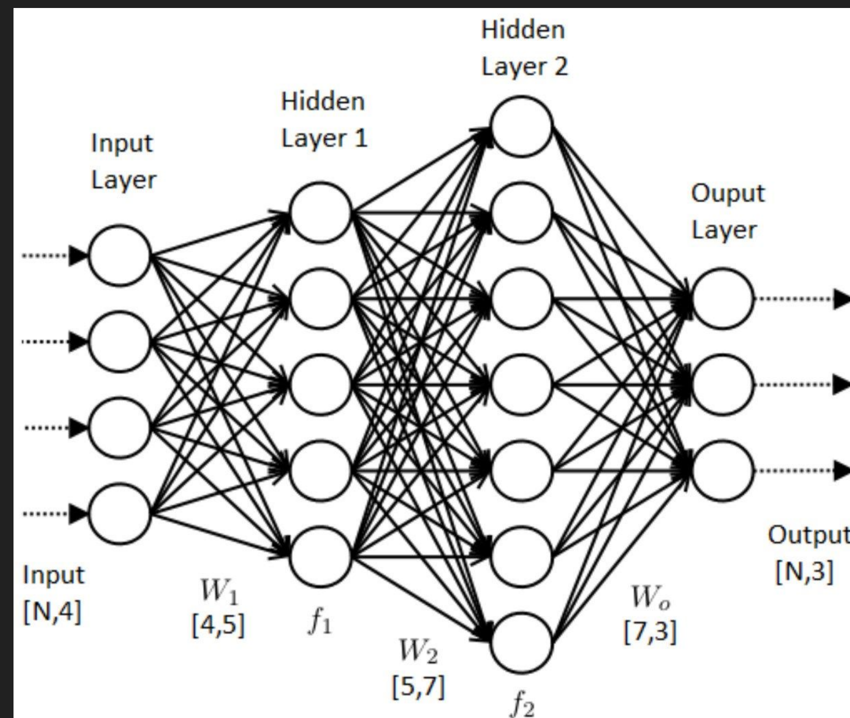
Fig : Perceptron

‘Vanilla’ Neural Networks : Multi Layer Perceptron (MLP)

- Also called ***‘fully-connected’ feed-forward*** neural networks.
- Stack ‘perceptron’ on several *layers.*
- Each *hidden layer* has a *latent size (*output size)

Glossary:

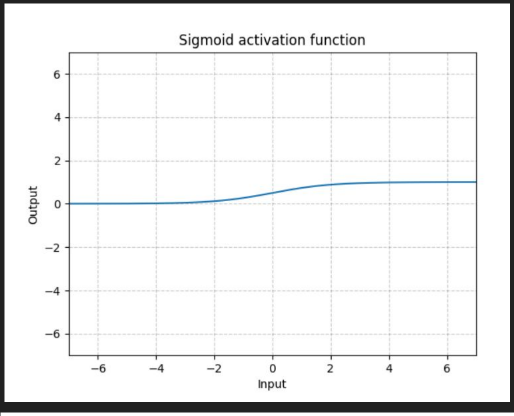
- Input nodes
- Output nodes
- Connections / weights
- Activation function
- Hidden layers



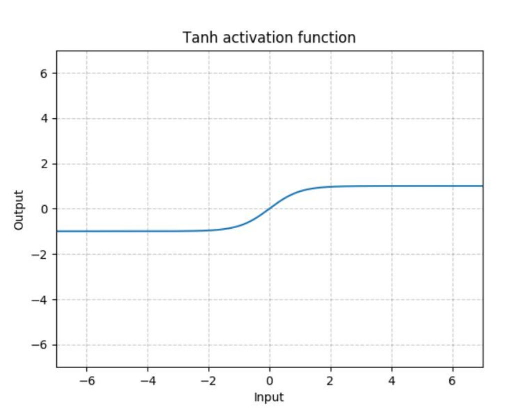
Playground.tensorflow.org

(Some) Activation Functions:

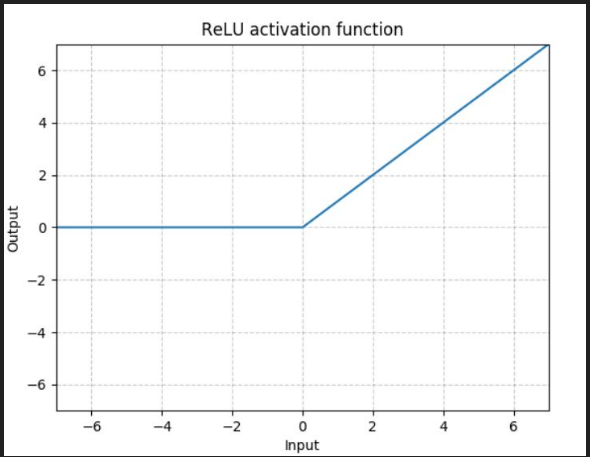
- Sigmoid:



- Tanh



- Rectified Linear Unit (ReLU) / LeakyReLU:



- Softmax : all neurons in [0,1] and sum to 1 (good for probability distribution).

Training : Finding the “best” weights (= parameters)

Remember that ‘cost’ function (criterion) ?

=> Goal is to minimize that

How ? Using gradient descent and backpropagation of the gradients !

(So you need -almost- differentiable functions)

Learning algorithm

- “Forward pass”: Feed an example (or more) to the network
- Compute error with regard to your criterion (compared to expected targets)
- Compute gradients (wait for it)
- Update weights
- Repeat until stopping criterion

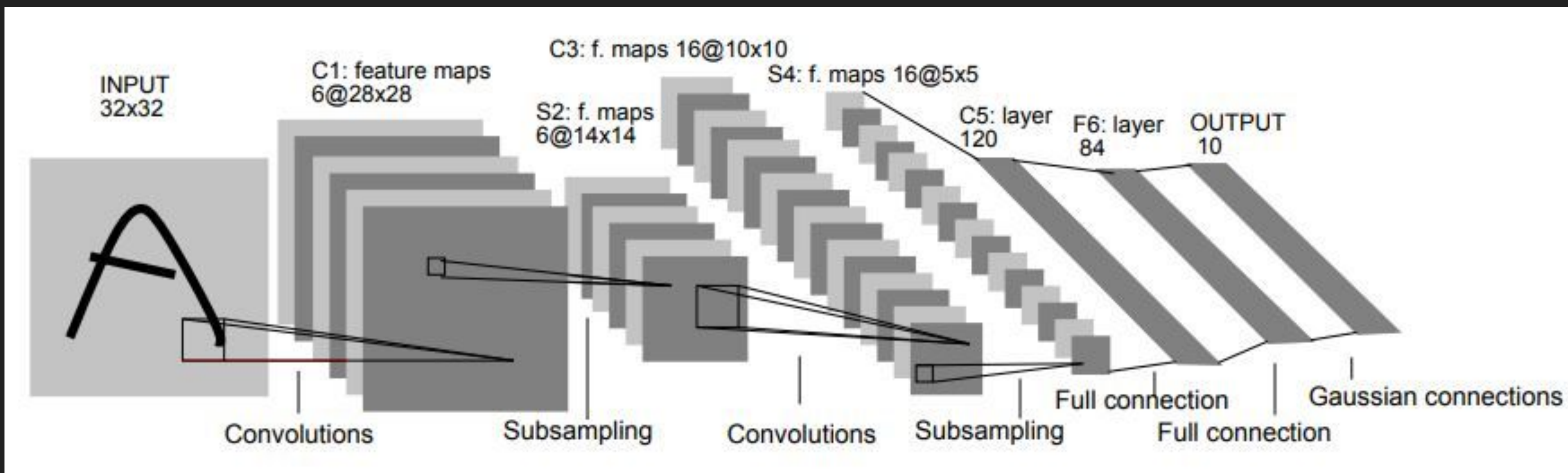
Backpropagation ?

- Goal of gradient : update the weights in the “right” direction
  - Which weight is responsible for the error and on what ‘amount’ ?
- Backpropagation relies on the chain rules to compute the gradients of a layer’s weight using the delta’s of next layer.

Different Losses

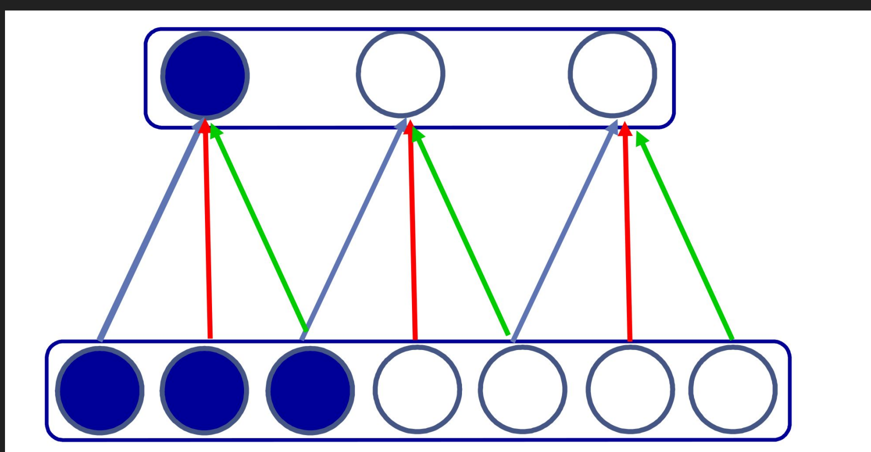
- Regression :
  - Mean-square error (MSE)
  - (Smooth) L1-Loss
- Classification :
  - Negative Log-Likelihood
  - Cross-Entropy
- And many others defined for specific problems...

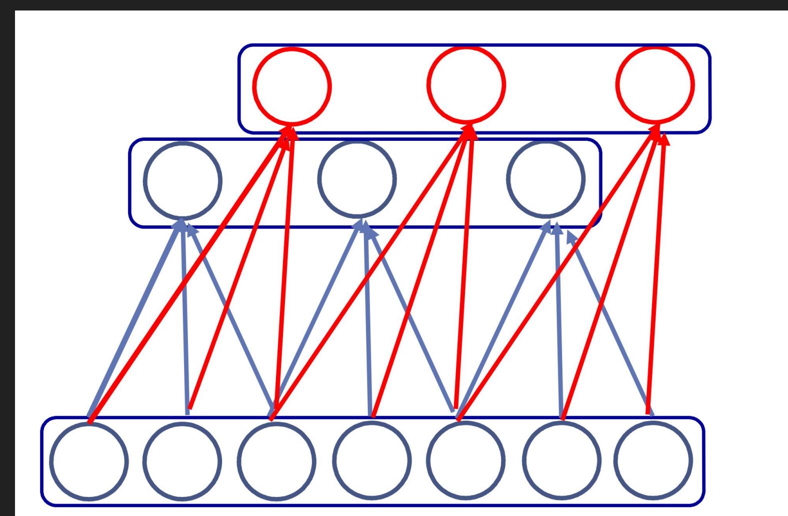
Convolutional Neural Networks



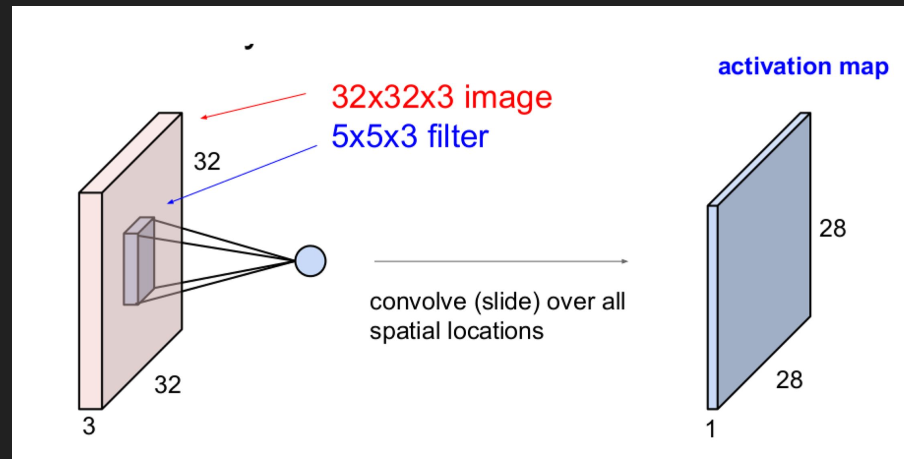
Convolutional Neural Networks

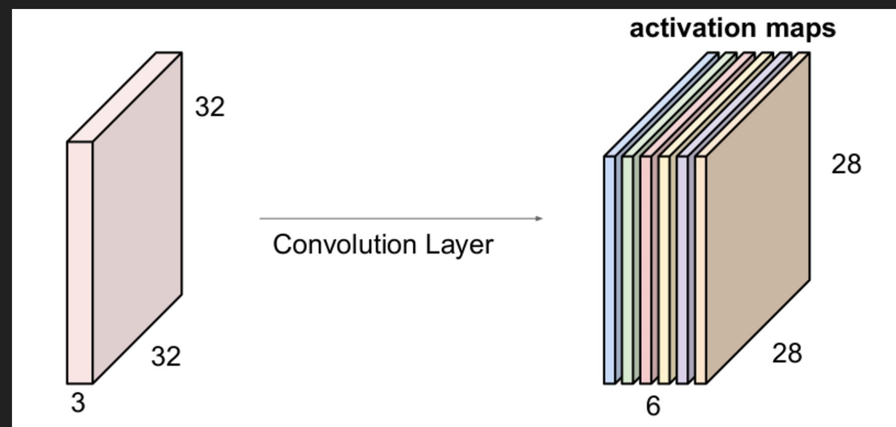
- Use local connections instead of fully-connected
- Shared weights
- Each layer can train several convolution filters : channels





- Filters extend the full depth of the input volume
- Convolve the filter with the image: Slide over the image spatially, computing dot products.
- Stack the activation map produced by each filter : new “image”

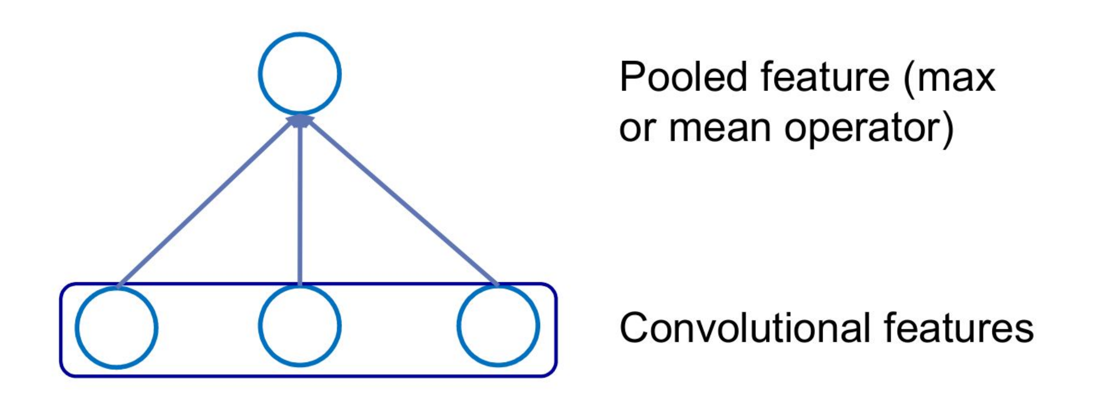


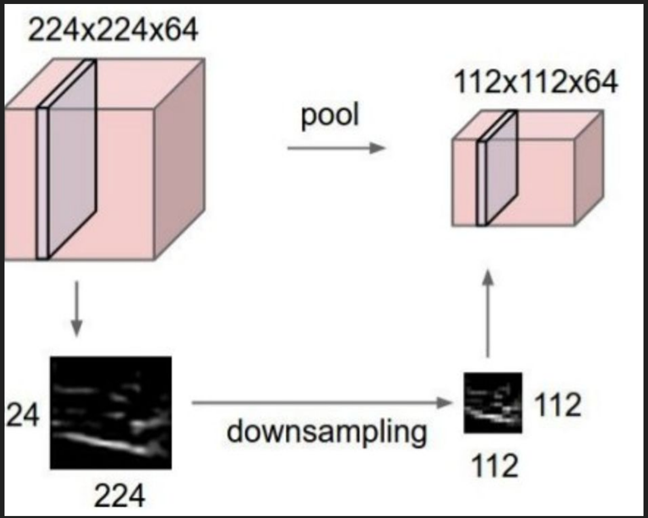


Credit Fei-Fei Li’s course

Pooling

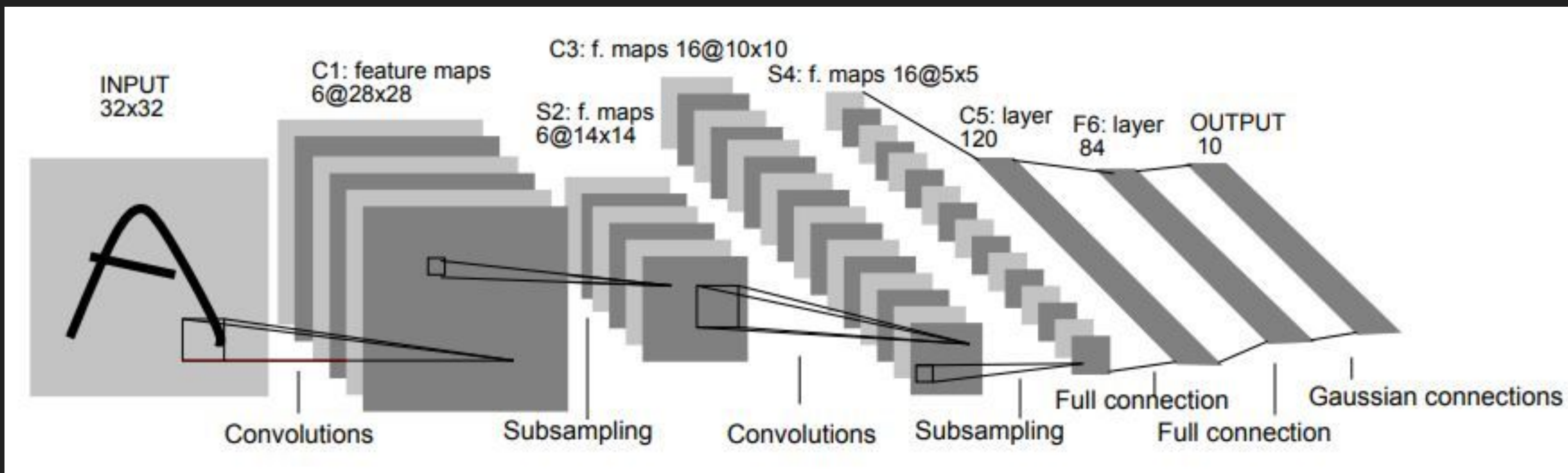
- Number of features extracted can be very large (increasing number of channels / filters)
- To reduce the size of the activation maps, one can :
  - Use strides
  - Downsample using Max or AveragePooling - applied on each channel independently



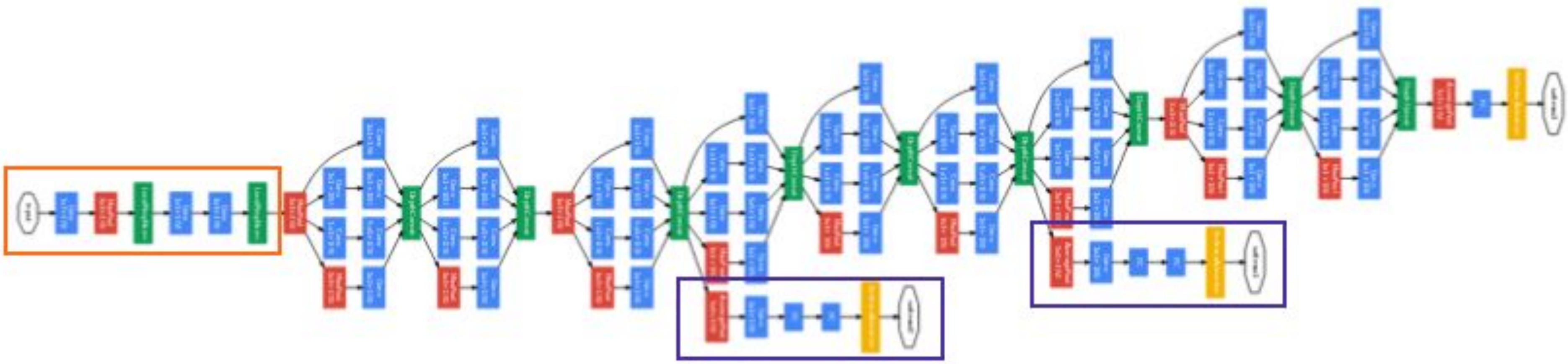


Credit Fei-Fei Li's course

Convolutional Neural Networks

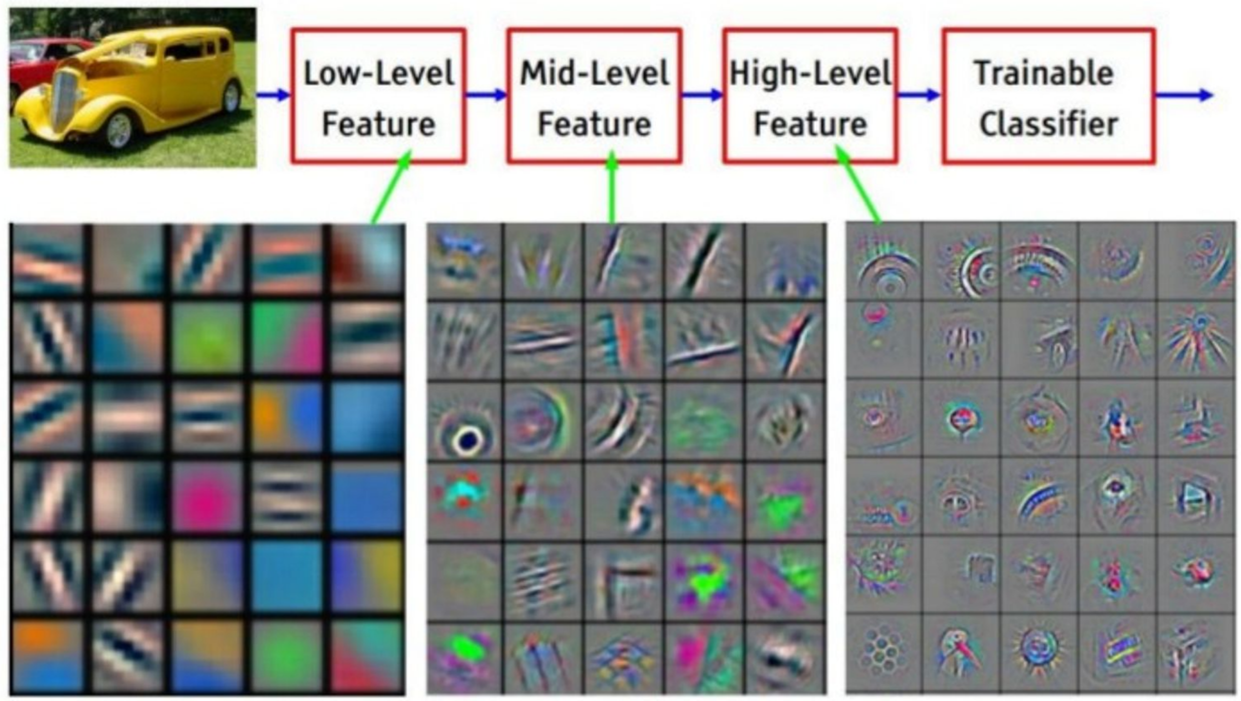


[redacted]

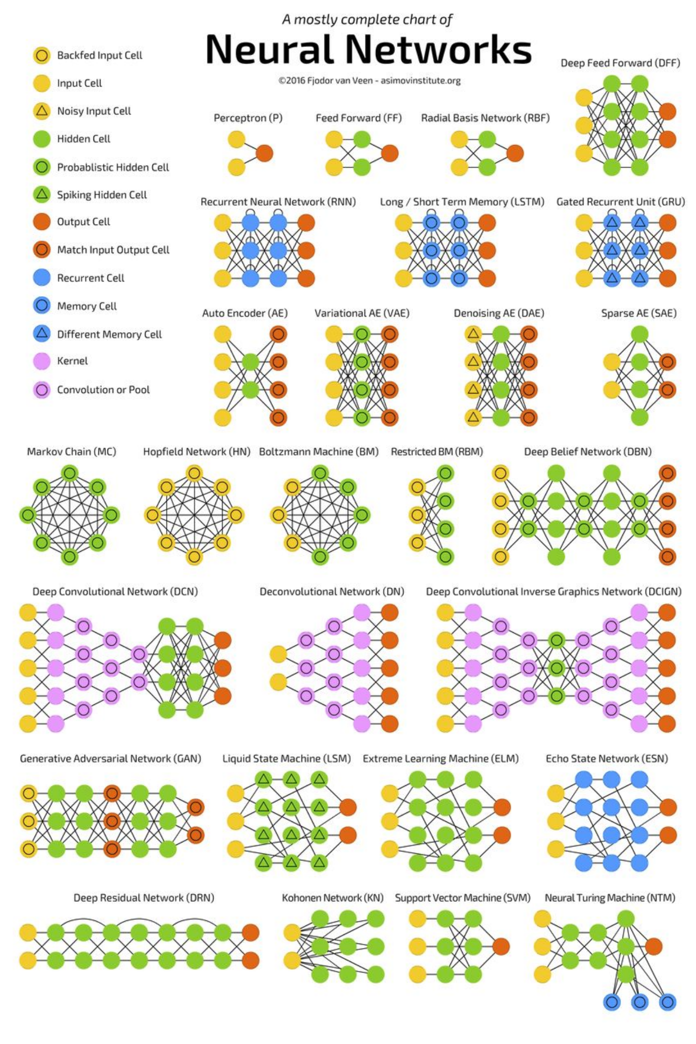


[redacted]

“Inside” deep learning : Representation Learning



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


A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

Let’s code !