Big data science Day 3

F. Legger - INFN Torino

https://github.com/leggerf/MLCourse-2021

Yesterday

- Big data
- Analytics
- Machine learning

Today

- **Deep learning**

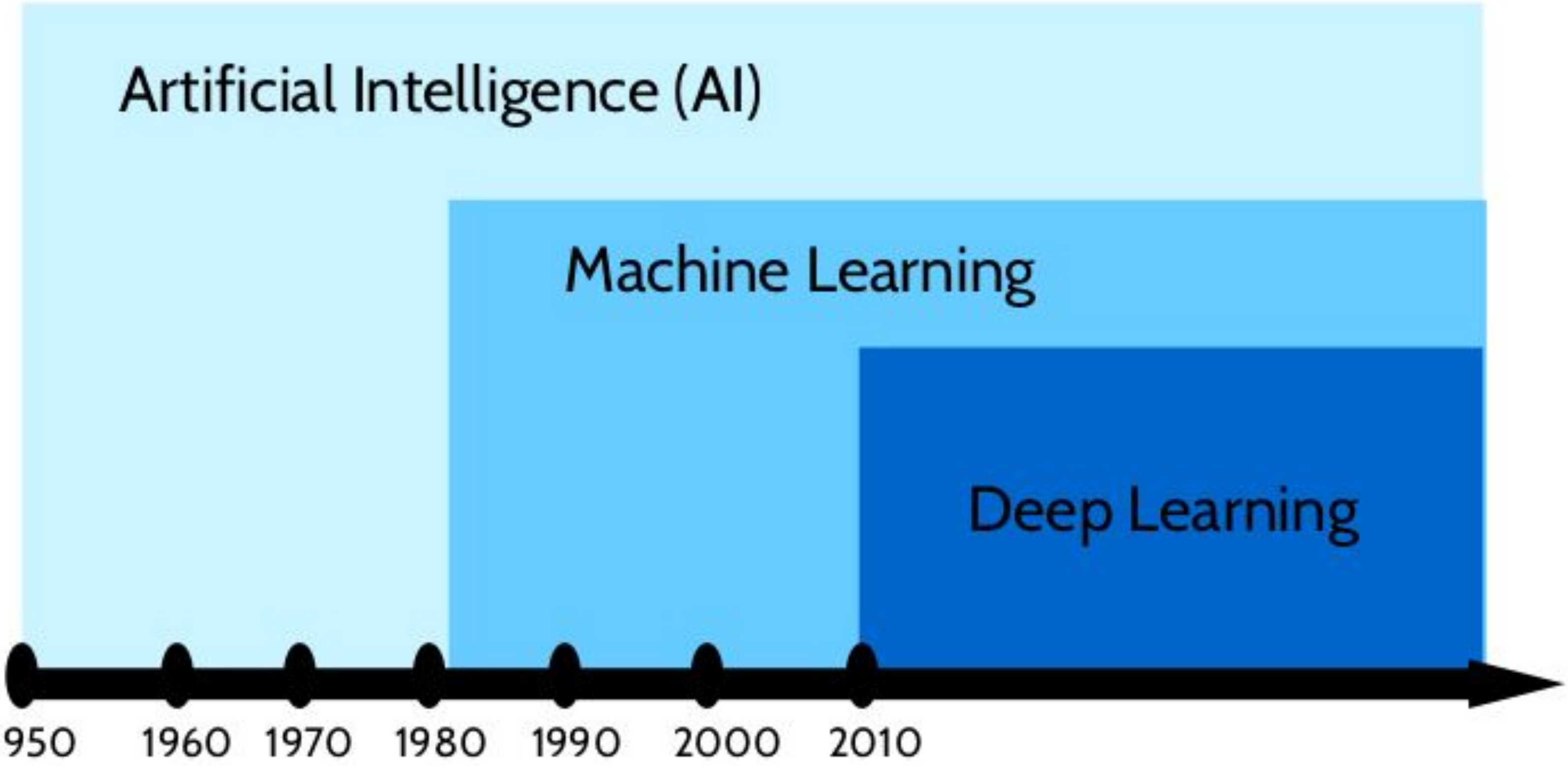
Next

- Parallelisation
- Heterogeneous architectures

Hands on

Keras

Deep Learning is a subfield of ML concerned with algorithms inspired by the structure and function of the brain called artificial neural networks [Jason Brownlee]

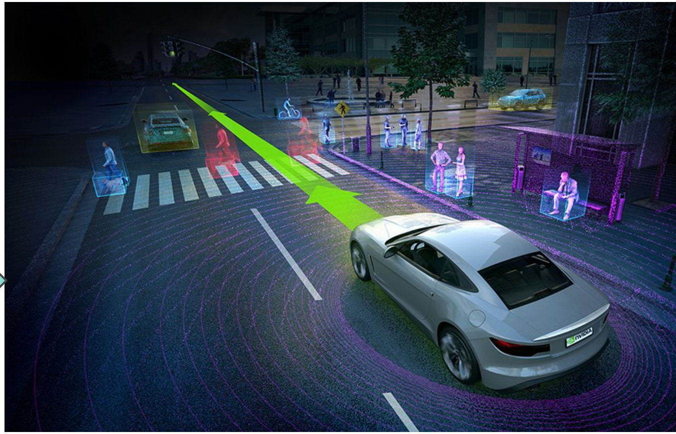


Machine translation
Real-time translation into Mandarin Chinese (2012)

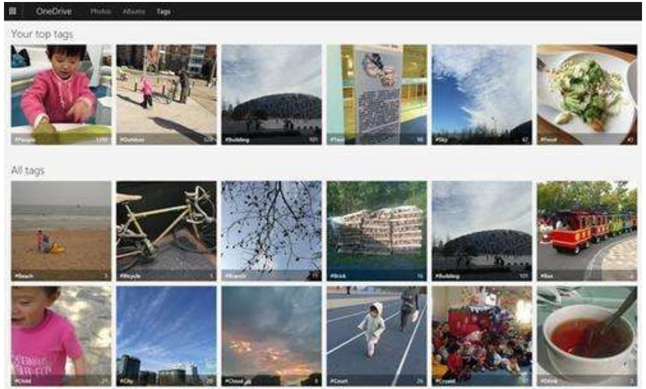
Strategy games
DeepMind beats Go world champion (2017)

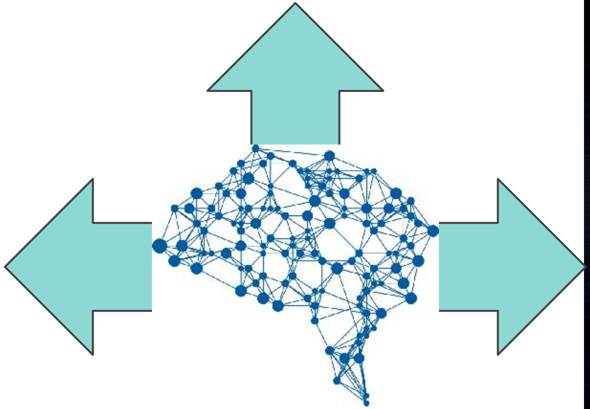
Creativity

Self driving cars

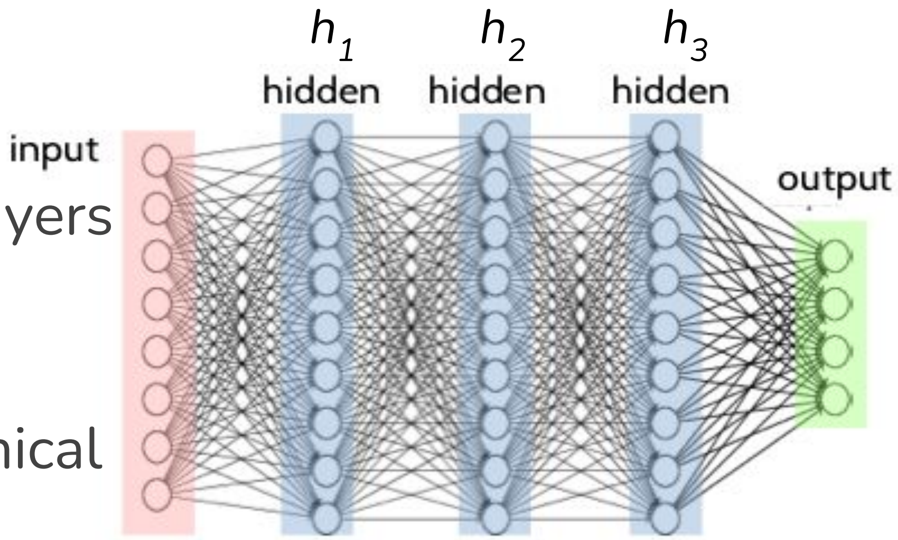


Visual recognition





Deep Learning



- Neural network with several layers
  - Deep vs shallow
- A family of parametric models which learn non-linear hierarchical representations:

$$a_L(\mathbf{x};\boldsymbol{\Theta}) = h_L(h_{L-1}(\ldots(h_1(\mathbf{x},\boldsymbol{\theta}_1),\boldsymbol{\theta}_{L-1}),\boldsymbol{\theta}_L)$$

input

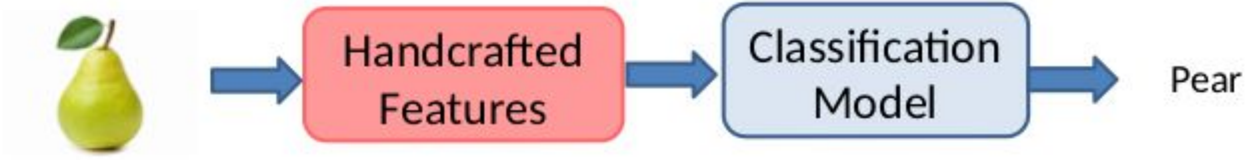
parameters of the network

non-linear activation function

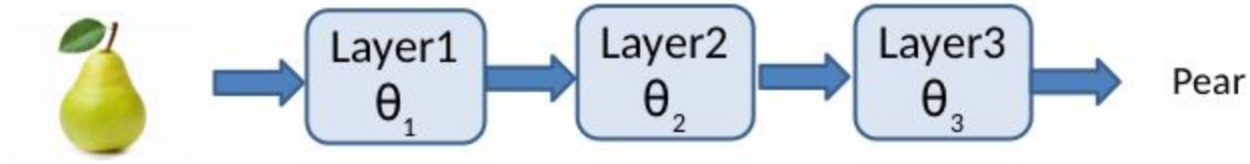
parameters of layer *L*

Learning hierarchical representations

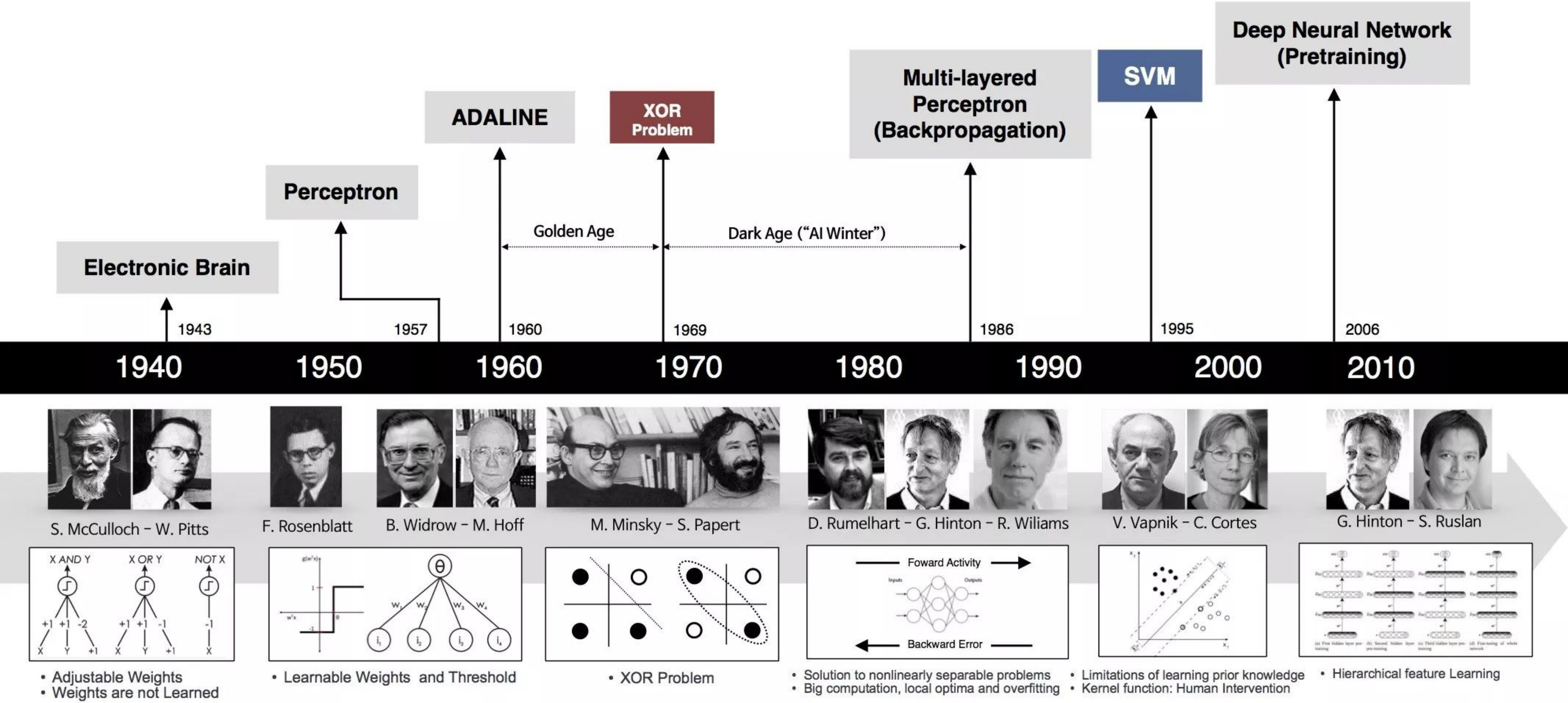
- Traditional framework



- Deep Learning

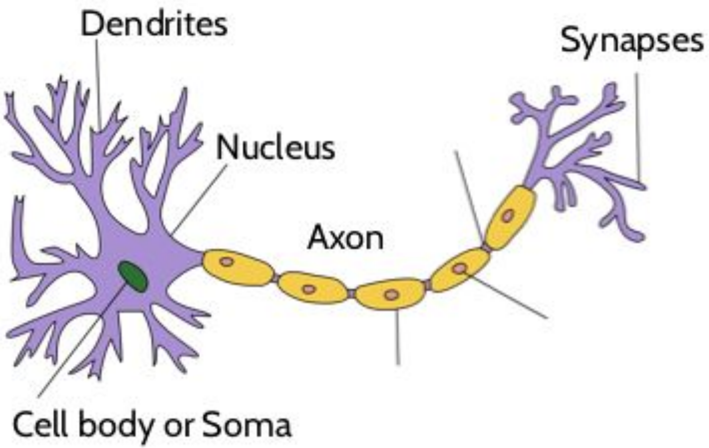


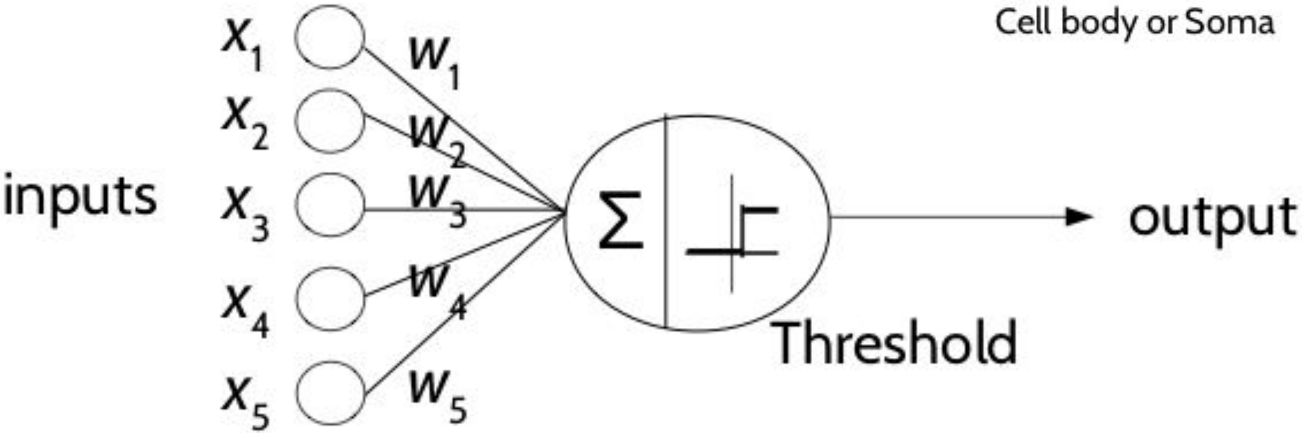
Brief history of neural networks



1943 – McCulloch & Pitts Model

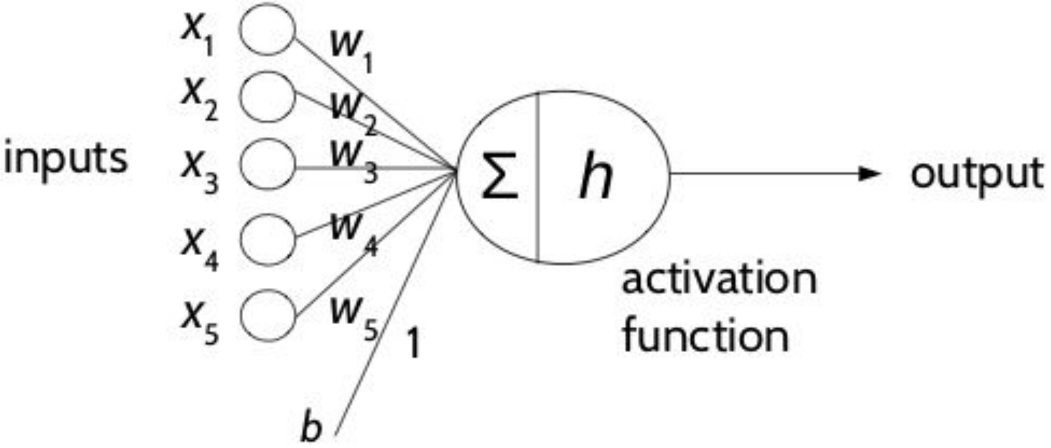
- Early model of artificial neuron
- Generates a binary output
- The weights values are fixed

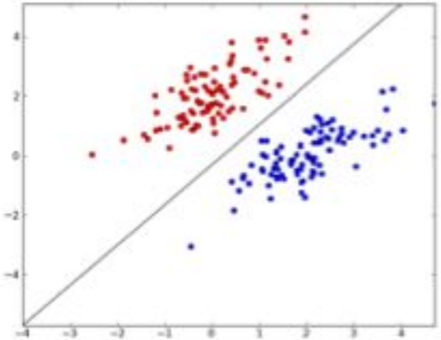




1958 – Perceptron by Rosemblatt

- Perceptron as a machine for linear classification
- Main idea: Learn the weights and consider bias.
  - One weight per input
  - Multiply weights with respective inputs and add bias
  - If result larger than **threshold** return 1, otherwise 0

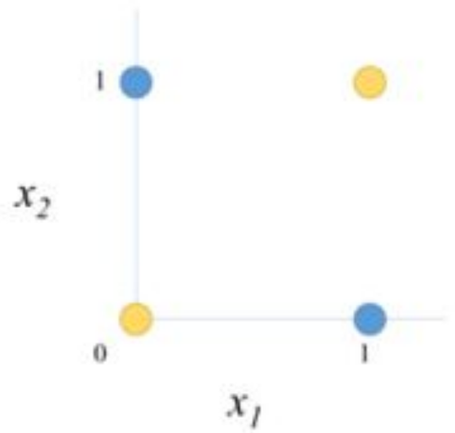


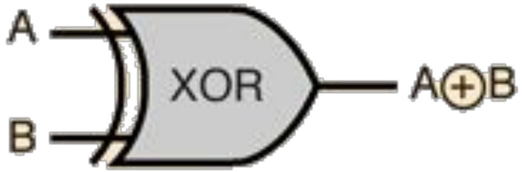


First NN winter

- 1970- Minsky. The XOR cannot be solved by perceptrons.

- Neural models cannot be applied to complex tasks.



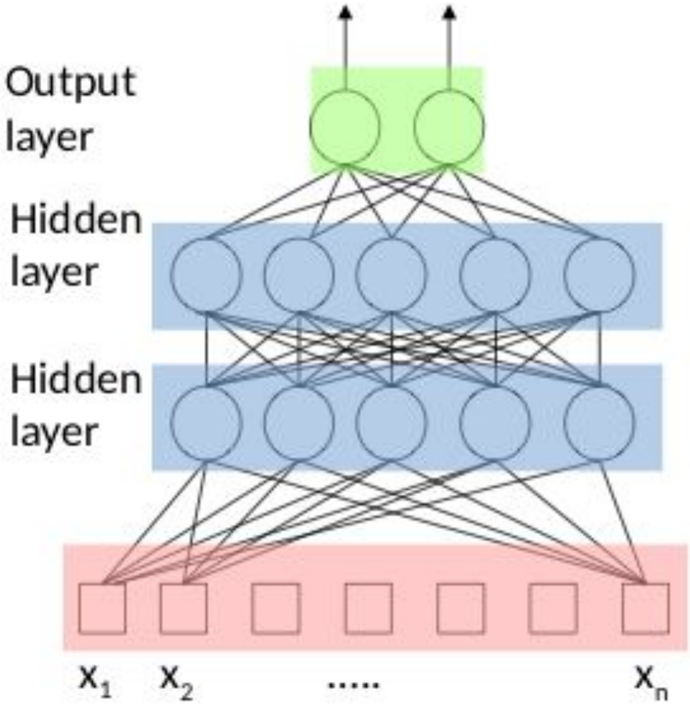


A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

Multi-layer Feed Forward Neural Network

- 1980s. Multi-layer Perceptrons (MLP) can solve XOR.

- ML Feed Forward Neural Networks:
  - Densely connect artificial neurons to realize compositions of non-linear functions
  - The information is propagated from the inputs to the outputs
  - The input data are usually *n*-dimensional feature vectors
  - Tasks: Classification, Regression



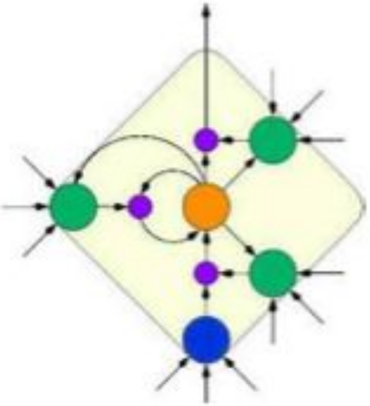
How to train it?

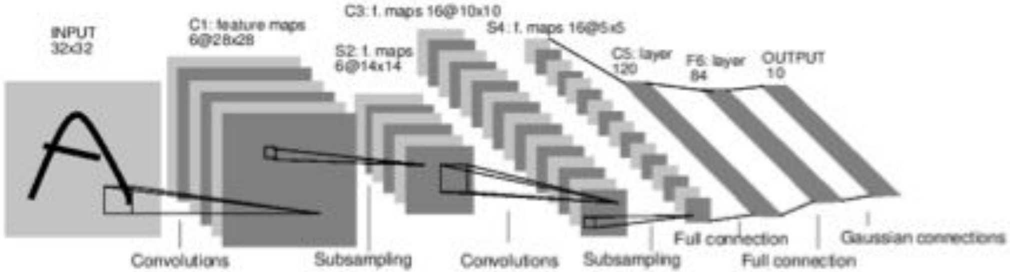
- Rosenblatt algorithm* not applicable, as it expects to know the desired target
  - For hidden layers we cannot know the desired target
- Learning MLP for complicated functions can be solved with **Back propagation (1980)**
  - efficient algorithm for complex NN which processes large training sets

* Remember? Rosenblatt developed a method to train a single neuron

1990s - CNN and LSTM

- Important advances in the field:
  - Backpropagation
  - Recurrent Long-Short Term Memory Networks (Schmidhuber, 1997)
  - Convolutional Neural Networks - LeNet: OCR solved before 2000s (LeCun, 1998).

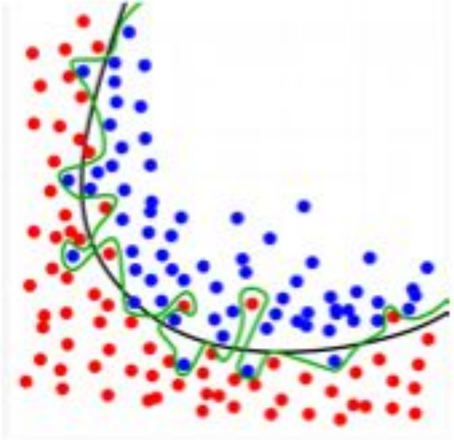


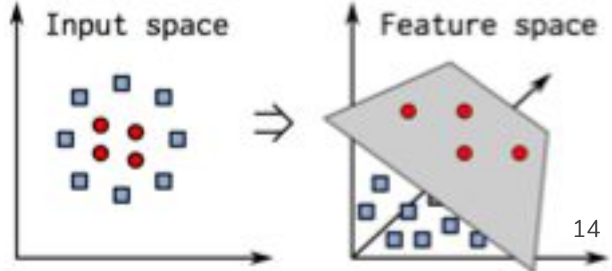


OCR: Optical character recognition

Second NN Winter

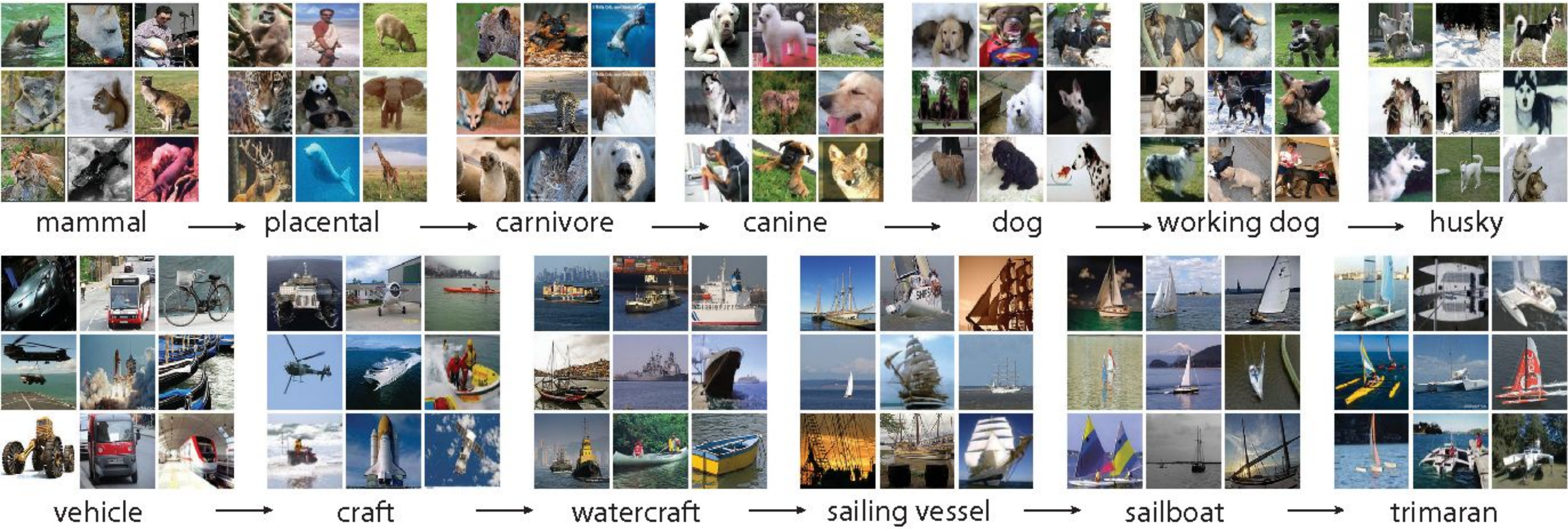
- NN cannot exploit many layers
  - Overfitting
  - Vanishing gradient (with NN training you need to multiply several small numbers → they become smaller and smaller)
- Lack of processing power (no GPUs)
- Lack of data (no large annotated datasets)
- Kernel Machines (e.g. SVMs) suddenly become very popular





ImageNet

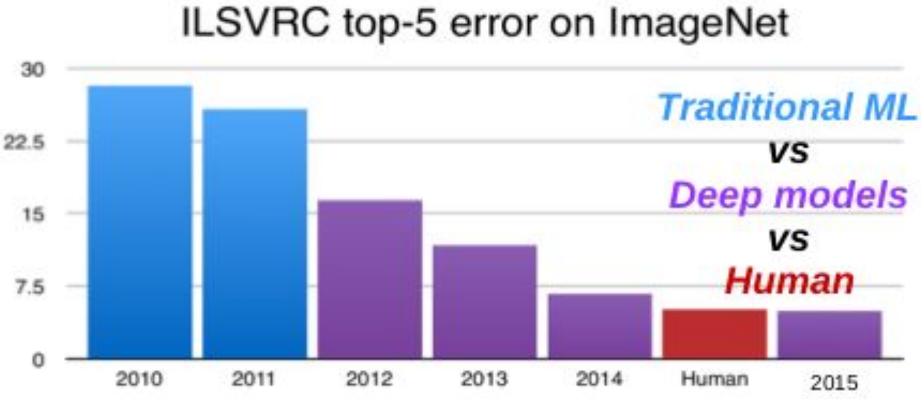
A Large-Scale Hierarchical Image Database (2009)



https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5206848&tag=1

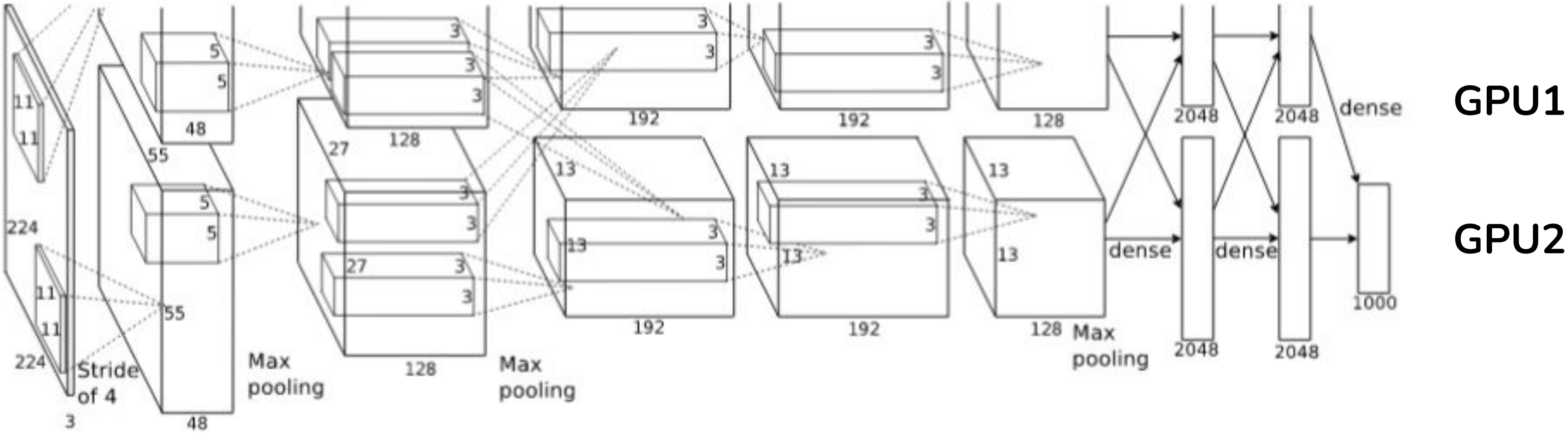
2012 - AlexNet

- Hinton's group implemented a CNN similar to LeNet [LeCun1998] but…
  - Trained on ImageNet (1.4M images, 1K categories)
  - With 2 GPUs
  - Other technical improvements (ReLU, dropout, data augmentation)



https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

AlexNet

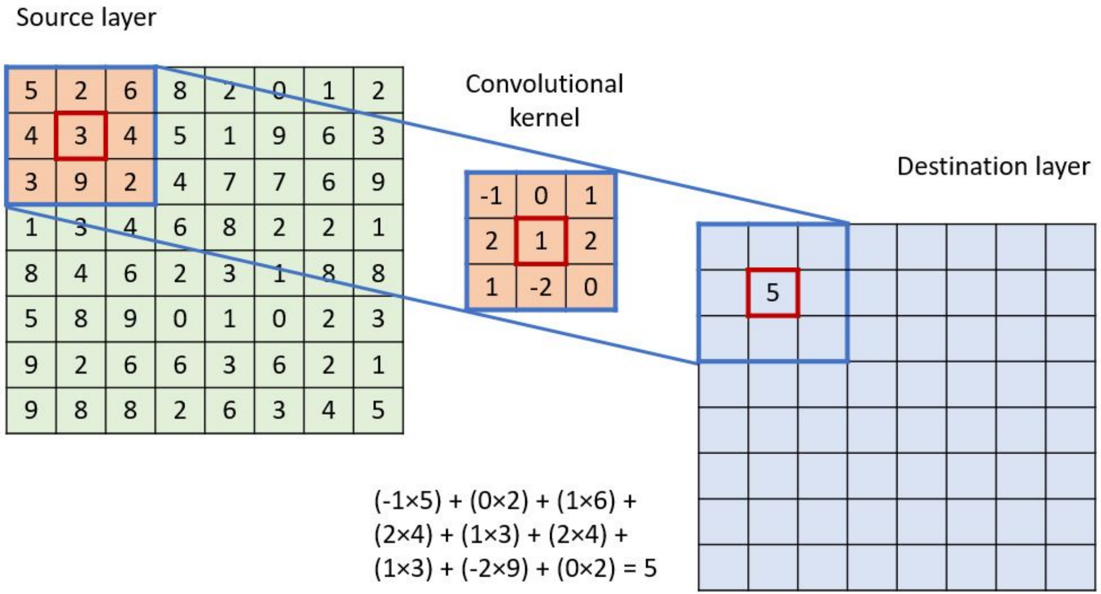


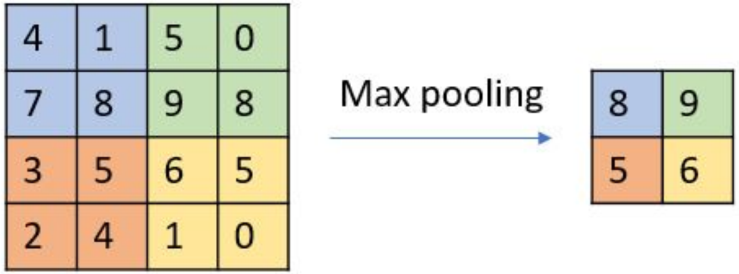
- 60M parameters
- Limited information exchange between GPUs

https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Convolutional Neural Networks (CNN)

- **Convolutional** layer: two functions produce a third that describes how the shape of one is changed by the other
- **pooling** layer: reduce dimensionality

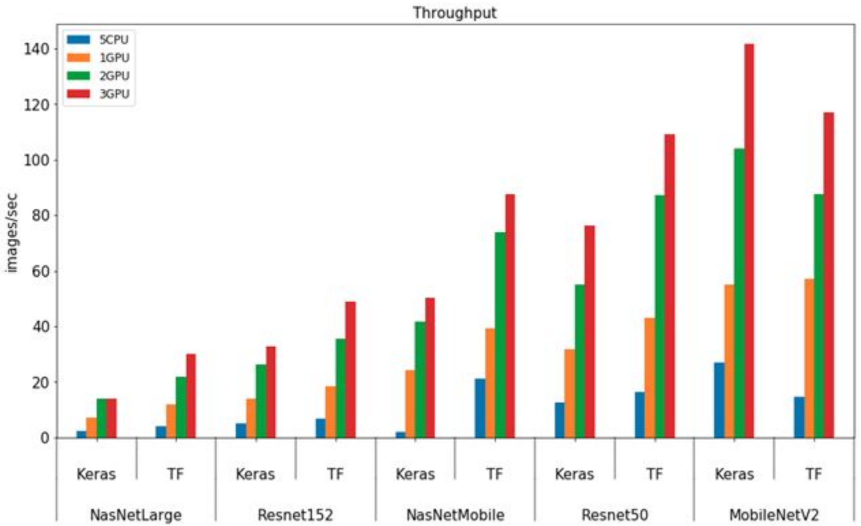




Why Deep Learning now?

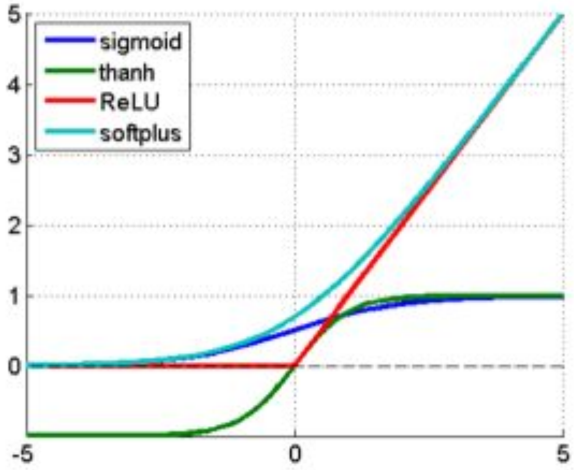
- Three main factors:
  - Better hardware
  - Big data
  - Technical advances:
    - Layer-wise pretraining
    - Optimization (e.g. Adam, batch normalization)
    - Regularization (e.g. dropout)

    ….



Rectified Linear Units (2010)

$$f(x) = \max(0, x)$$



- More efficient gradient propagation: (derivative is 0 or constant)

- More efficient computation: (only comparison, addition and multiplication).

- Sparse activation: e.g. in a randomly initialized networks, only about 50% of hidden units are activated (having a non-zero output)

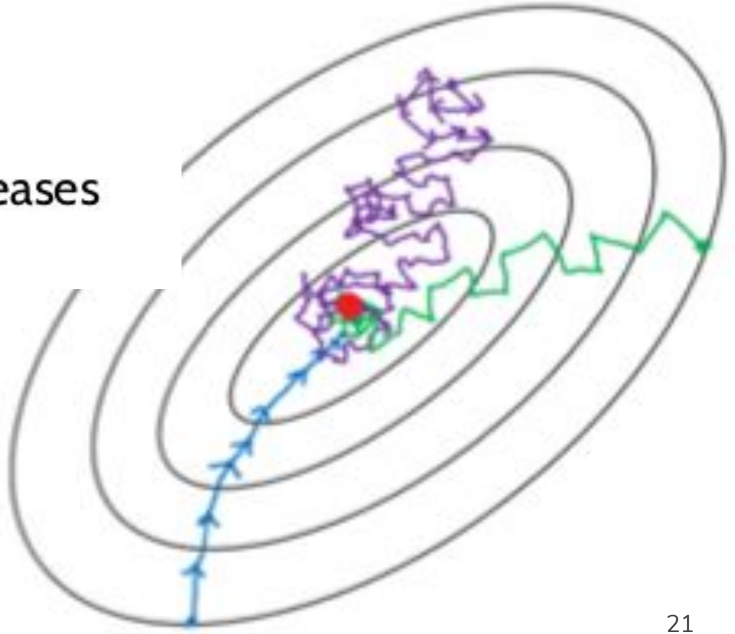
https://www.cs.toronto.edu/~fritz/absps/reluICML.pdf

Stochastic gradient descent (SDG)

- Use mini-batch **sampled** in the dataset for gradient estimate.

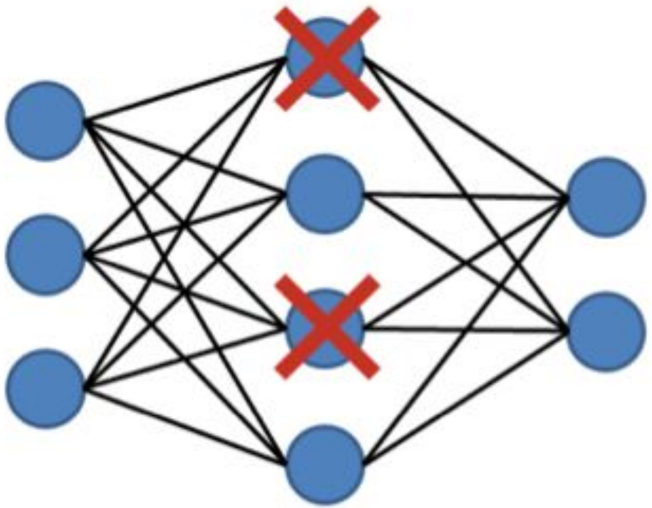
$$\boldsymbol{\Theta}^{t+1} = \boldsymbol{\Theta}^{t} - \frac{\eta_t}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_{\Theta} \mathcal{L}_i$$

- Sometimes helps to escape from local minima
- Noisy gradients act as regularization
- Variance of gradients increases when batch size decreases
- Not clear how many sample per batch



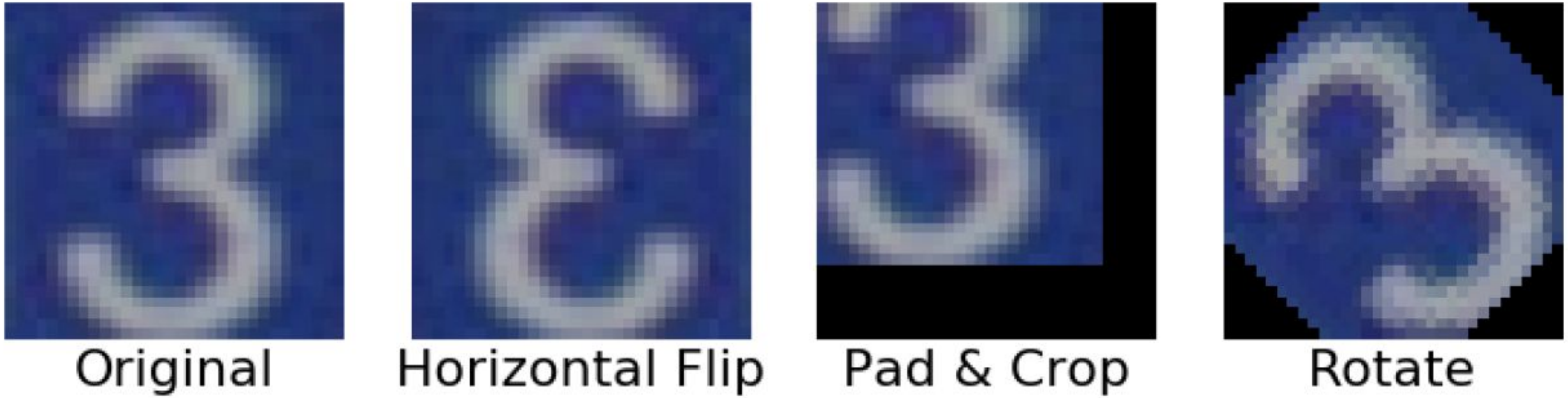
Regularization - Dropout

- For each instance drop a node (hidden or input) and its connections with probability *p* and train
- Final net just has all averaged weights (actually scaled by 1-*p*)
- As if ensembling 2^n different network substructures



Data augmentation

- Techniques to significantly increase the diversity of data available for training models, without actually collecting new data
- Data augmentation techniques such as cropping, padding, and horizontal flipping are commonly used to train large neural networks



Training a neural network

Name	Weight (lb)	Height (in)	Gender
Alice	133	65	F
Bob	160	72	M
Charlie	152	70	M
Diana	120	60	F

- Predict gender from weight and height

Feature engineering

- Symmetrize numeric values
- Category -> numbers

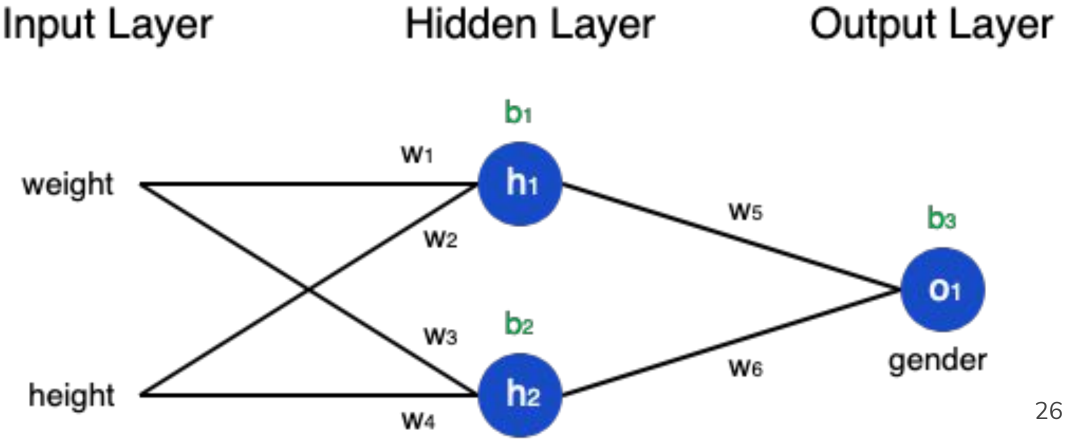
Name	Weight (lb)	Height (in)	Gender
Alice	133	65	F
Bob	160	72	M
Charlie	152	70	M
Diana	120	60	F

Name	Weight (minus 135)	Height (minus 66)	Gender
Alice	-2	-1	1
Bob	25	6	0
Charlie	17	4	0
Diana	-15	-6	1

Ingredients

- **n** : 4, number of samples (Alice, Bob, Charlie, Diana)
- **y** : variable being predicted (Gender)
- $\mathbf{y_{true}}$: true value of y, $\mathbf{y_{pred}}$: predicted value of y = **o**
- Loss function **L: MSE**
- Activation function **f**
- Outputs of the hidden layer **h**
- Unknown parameters: weights **w** and biases **b**

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_{true} - y_{pred})^2$$



Back propagation

- **Training the network == trying to minimize its loss**
  - Find weights **w** and biases **b**
  - $L(w_1, w_2, w_3, w_4, w_5, w_6, b_1, b_2, b_3)$
- Minimization taking partial derivatives (**back propagation**)

$$\boxed{\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{pred}} * \frac{\partial y_{pred}}{\partial h_1} * \frac{\partial h_1}{\partial w_1}}$$

For very simple case: with only Alice in the dataset, n=1

$$L = (1 - y_{pred})^2$$

$$\frac{\partial L}{\partial y_{pred}} = \frac{\partial (1 - y_{pred})^2}{\partial y_{pred}}$$

$$y_{pred} = o_1 = f(w_5 h_1 + w_6 h_2 + b_3)$$

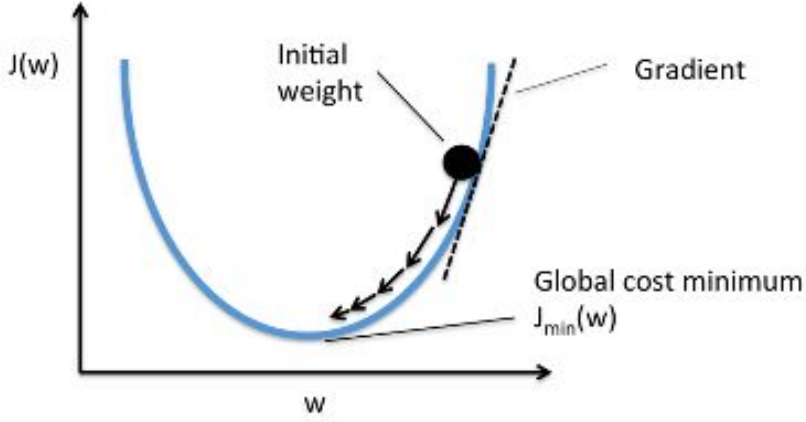
$$\frac{\partial y_{pred}}{\partial h_1} = \boxed{w_5 * f'(w_5 h_1 + w_6 h_2 + b_3)}$$

$$h_1 = f(w_1 x_1 + w_2 x_2 + b_1)$$

$$\frac{\partial h_1}{\partial w_1} = \boxed{x_1 * f'(w_1 x_1 + w_2 x_2 + b_1)}$$

Stochastic Gradient Descent (SGD)

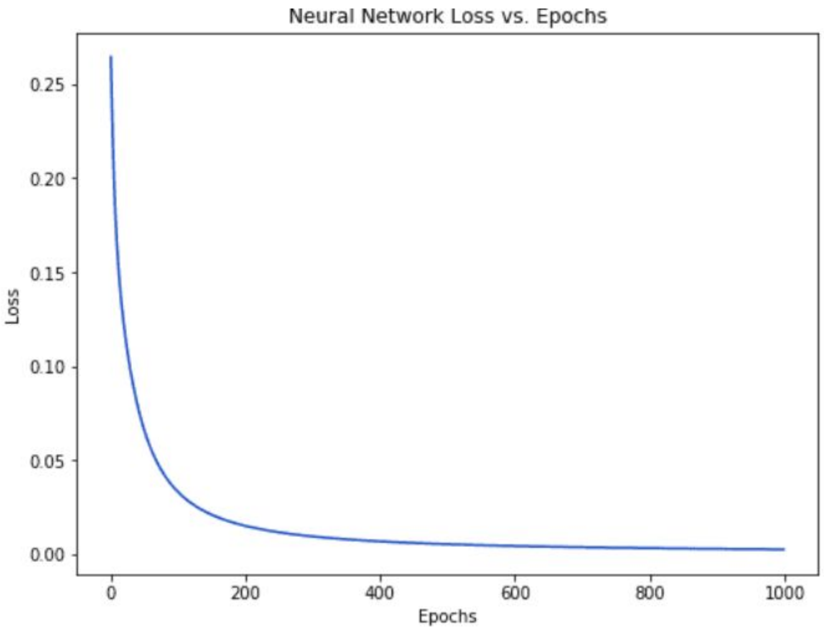
- **Stochastic** -> the parameters are updated using only a single training instance (usually randomly selected) in each iteration
- optimization algorithm to change our weights and biases to minimize loss
- **update equation**: $w_1 \leftarrow w_1 - \eta \frac{\partial L}{\partial w_1}$
- **η** is a constant called the **learning rate** that controls how fast we train



- If $\frac{\partial L}{\partial w_1}$ is positive, w_1 will decrease, which makes L decrease.
- If $\frac{\partial L}{\partial w_1}$ is negative, w_1 will increase, which makes L decrease.

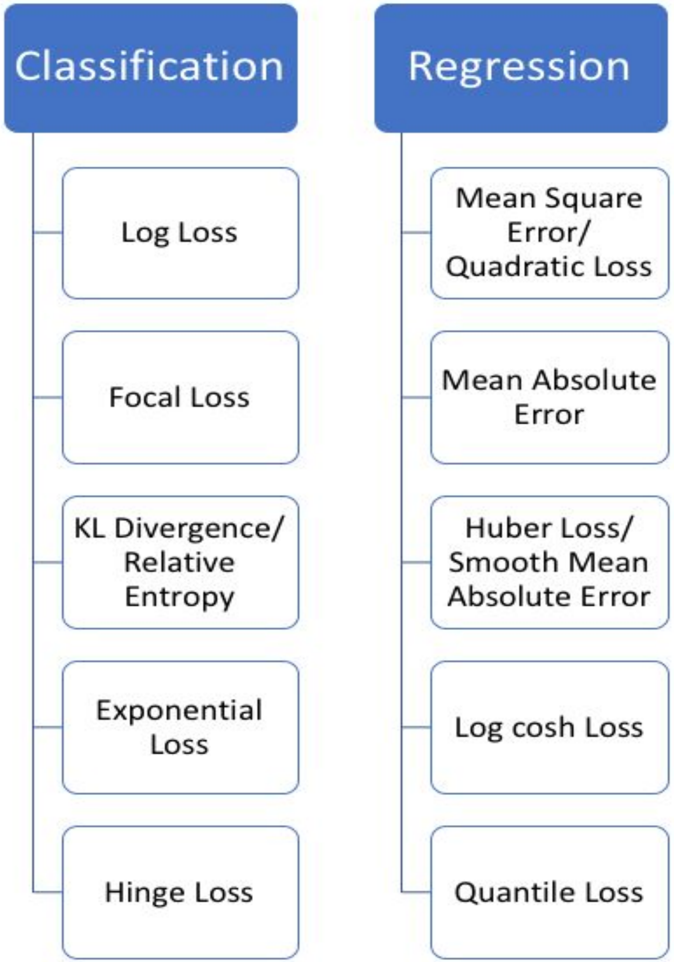
Training the network

- Choose one sample from our dataset
    - This is what makes it stochastic gradient descent - only operate on one sample at a time
- Calculate all the partial derivatives of loss with respect to weights or biases
- Use the update equation to update each weight and bias
- Iterate



Loss functions

- https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0
- https://www.wikiwand.com/en/Loss_functions_for_classification



Regularization

- One of the major aspects of training the model is overfitting -> the ML model captures the noise in your training dataset
- The **regularization** term is an addition to the loss function which helps generalize the model
    - **L1** or Lasso regularization adds a penalty which is the sum of the absolute values of the weights
        - L1+MSE

$$Min(\sum_{i=1}^{n}(y_i - w_i x_i)^2 + p\sum_{i=1}^{n}|w_i|)$$

    - **L2** or Ridge regularization adds a penalty which is the sum of the squared values of weights
        - L2+MSE

$$Min(\sum_{i=1}^{n}(y_i - w_i x_i)^2 + p\sum_{i=1}^{n}(w_i)^2)$$

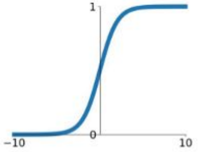
- **Dropout** in NN context: hidden nodes are dropped randomly
- **Early Stopping** is a time regularization technique which stops training based on given criteria

Activation functions

- Classification: sigmoid functions
  - sigmoids and tanh functions are sometimes avoided due to the vanishing gradient problem
- **ReLU** function is a general activation function
- dead neurons in our networks -> the leaky ReLU
- ReLU function should only be used in the hidden layers
- As a rule of thumb, start with ReLU

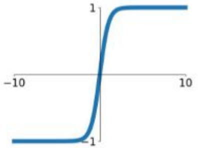
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



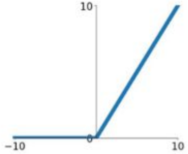
tanh

$\tanh(x)$



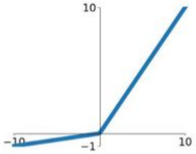
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

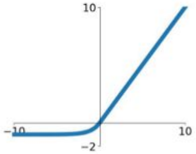


Maxout

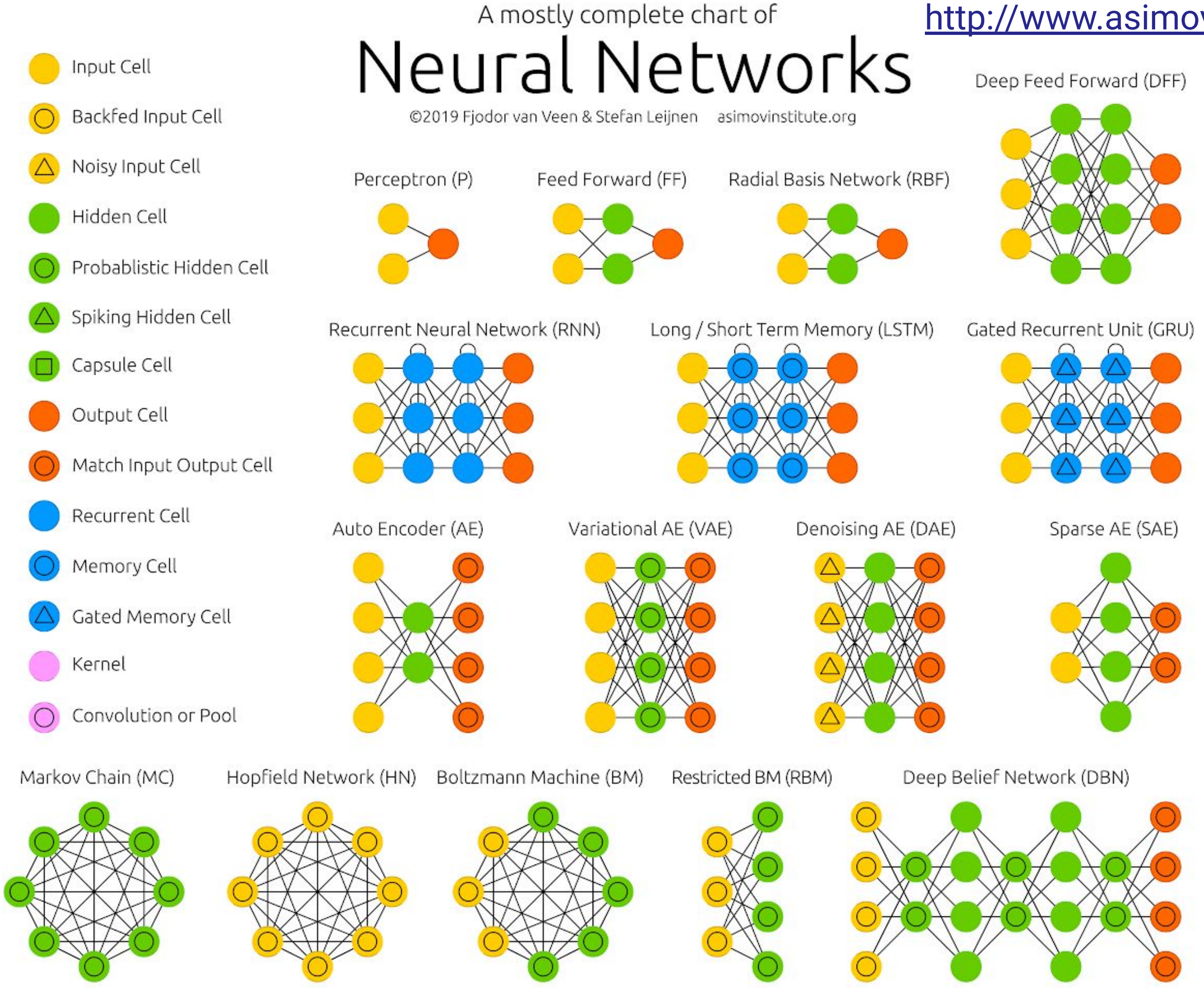
$\max(w_1^T x + b_1, w_2^T x + b_2)$

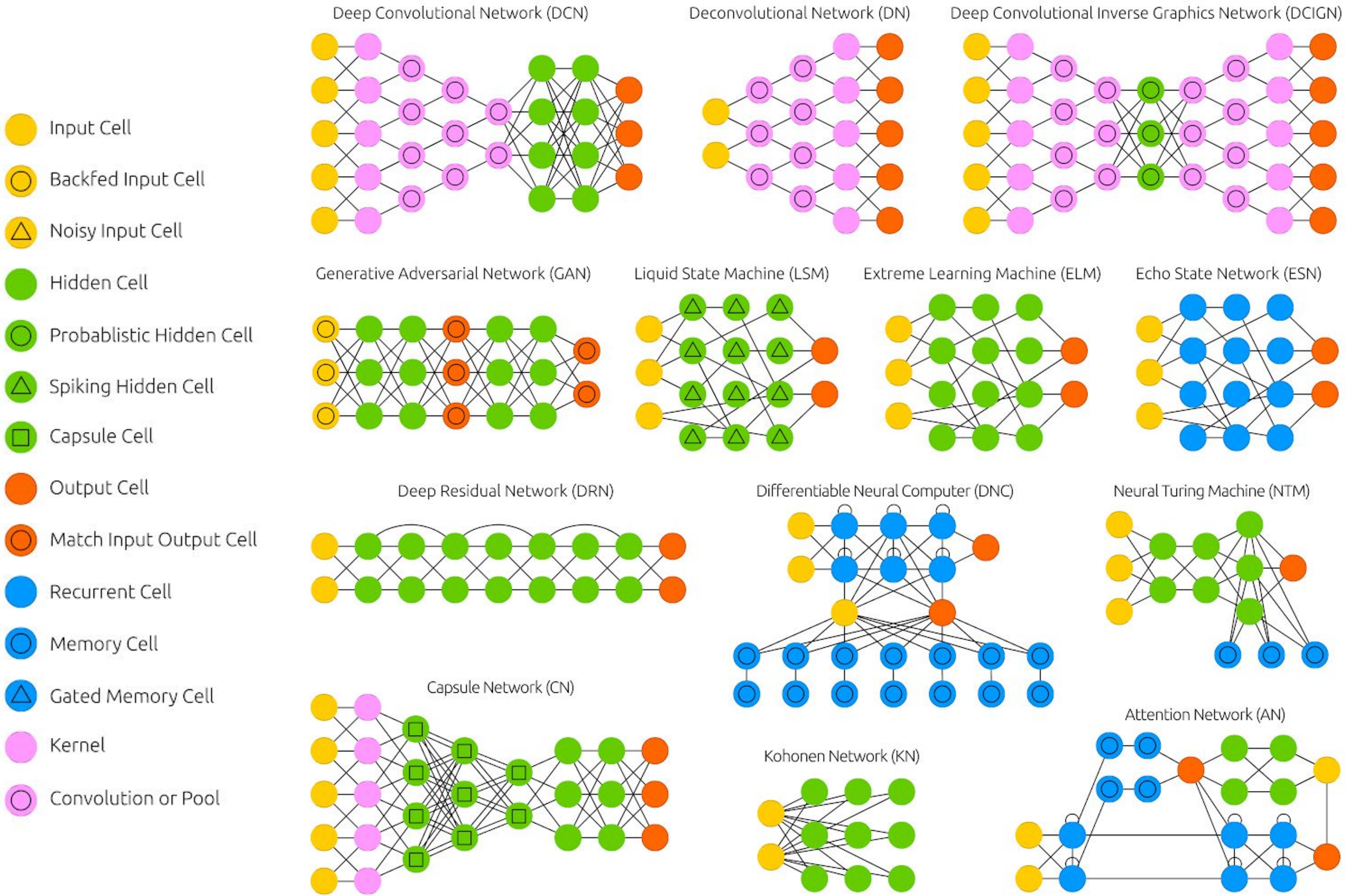
ELU

$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$



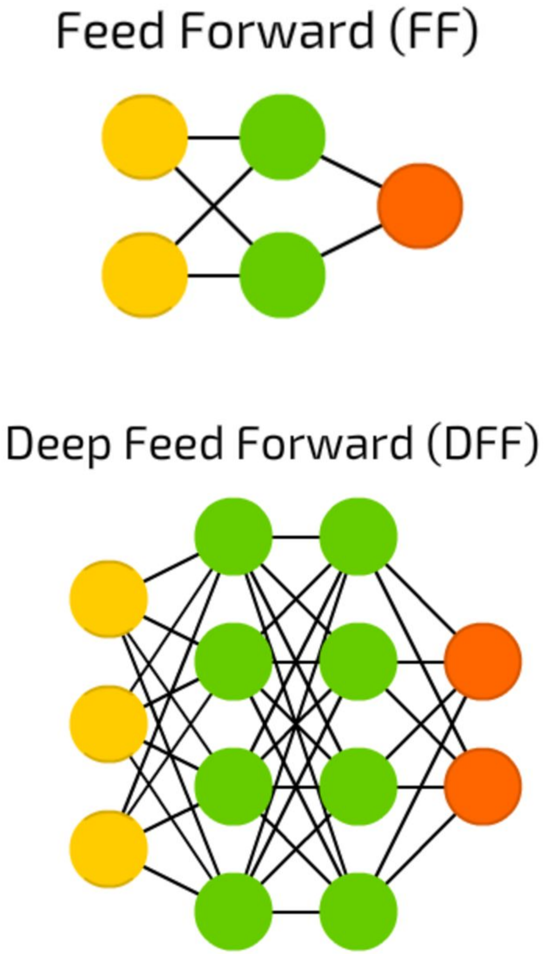
http://www.asimovinstitute.org/neural-network-zoo/




Input Cell
Backfed Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Capsule Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Gated Memory Cell
Kernel
Convolution or Pool
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)
Differentiable Neural Computer (DNC)
Neural Turing Machine (NTM)
Capsule Network (CN)
Kohonen Network (KN)
Attention Network (AN)

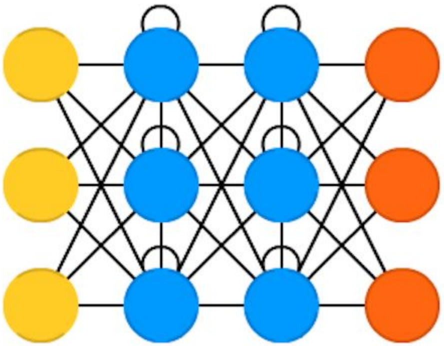
Feed Forward

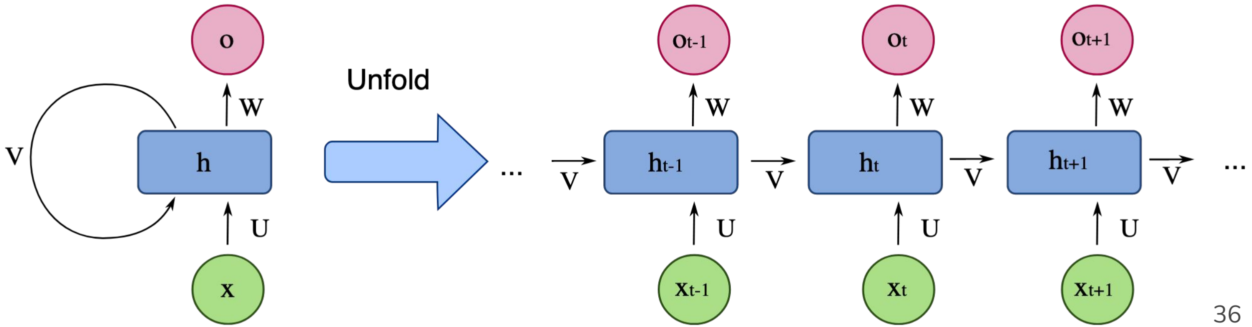
- Supervised, simplest form of NN
- The data passes through input nodes and exits on the output nodes
- Easy to implement and combine with other type of ML algorithms
- Typically trained with backpropagation
- Used in many ML tasks, speech, image recognition, classification, computer vision
- input/outputs are vectors of fixed length
- **DFF** is a FF NN with more than one hidden layer



Recurrent Neural Network (RNN)

- FFNN with **Recurrent Cells**: hidden cell that receives its own output with fixed delay
- RNNs permit to operate on **sequences of vectors** (in the input, the output, or both)
- RNNs, once unfolded in time, can be seen as very deep FF networks in which all the layers share the same weights





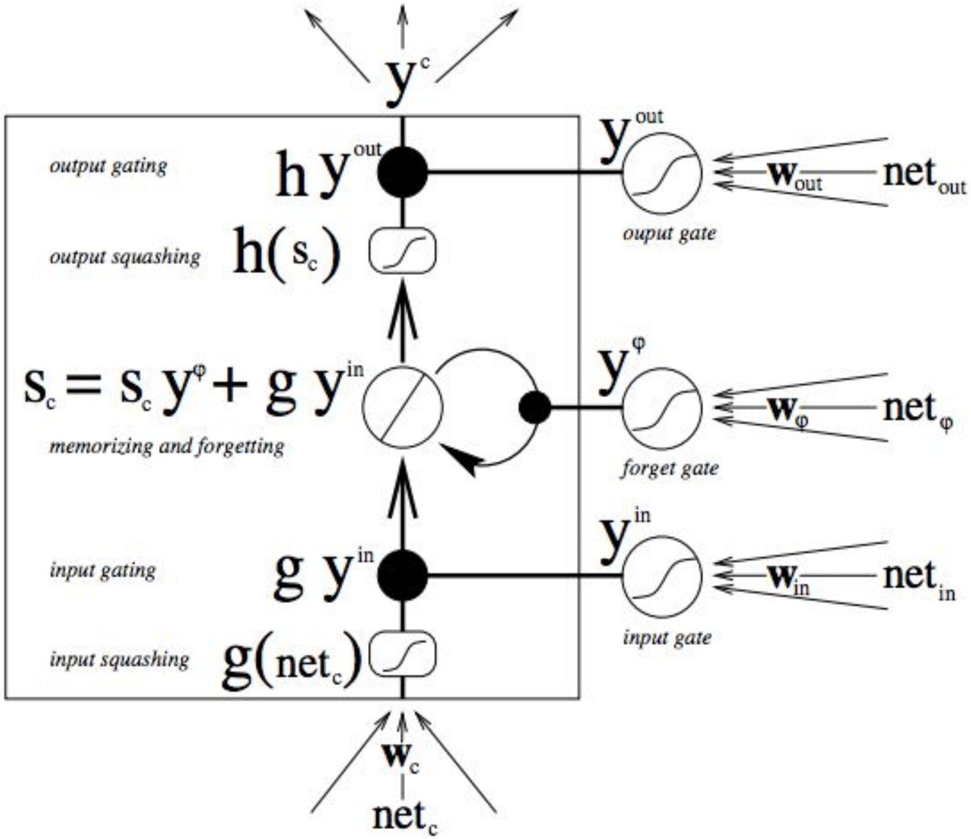
Recurrent Neural Network (RNN)

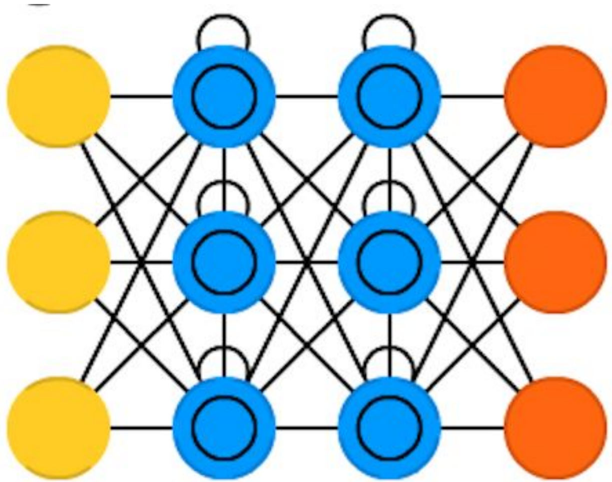
- The parameters to be learned are shared by all time steps in the network
- The gradient at each output depends also on the calculations of the previous time steps
- context is important, decision from past iterations can influence current state
- a word can be analyzed only in context of previous words or sentences

Long/Short Term Memory (LSTM)

- RNNs not really capable of learning long term dependencies
  - Due to vanishing gradient with increasing time steps
  - -> add cells with memory: “keep in mind” previous info, e.g. something that happened many frames ago
- A common LSTM unit is made of a **cell**, an **input**, an **output** and a **forget** gate
- The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell

Long/Short Term Memory (LSTM)

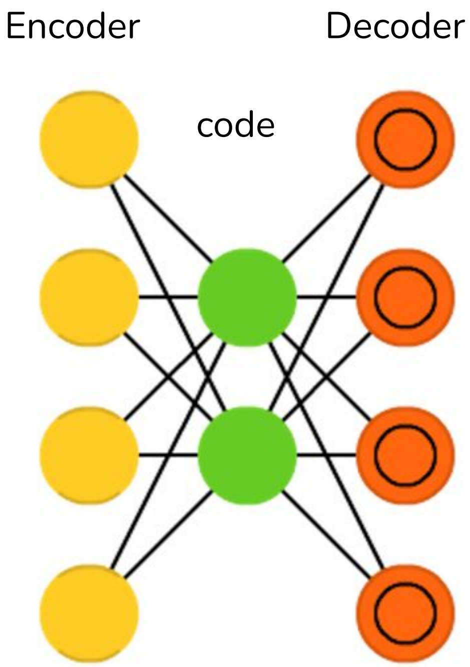




Auto Encoders

- **Unsupervised** learning
  - find smaller representation of given input and search for common patterns
- Compress (encode) information automatically.
- An **encoder** is a deterministic mapping f that transforms an input vector x into hidden representation y
- A **decoder** maps back the hidden representation y to the reconstructed input z via g.
- **Autoencoder**: compare the reconstructed input z to the original input x and try to minimize the reconstruction error
- used for classification, clustering and feature compression

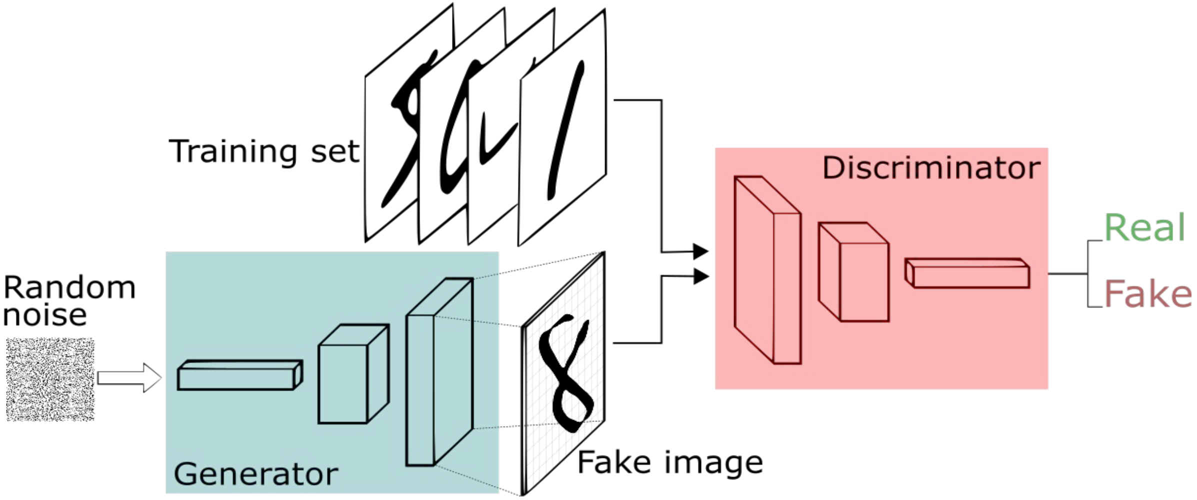
Auto Encoders

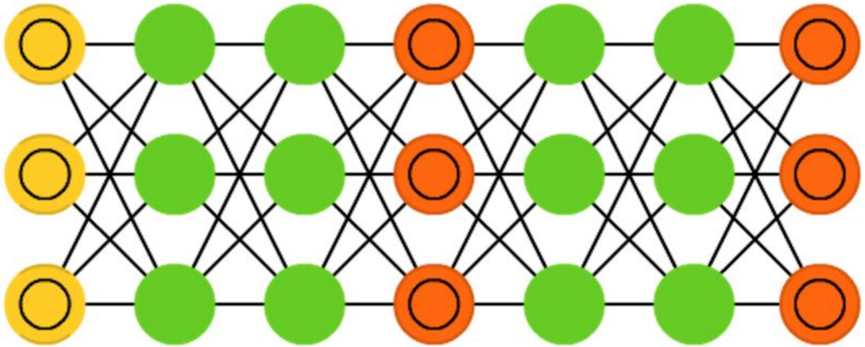


Generative Adversarial Networks (GAN)

- GANs represent a huge family of double networks that are composed from a **generator** net and a **discriminator** net
  - The generator produces samples close to training samples
  - Discriminator net (adversary) must differentiate between samples from the generative net and the training set
  - Use error feedback to improve task of both nets, until discriminator can no longer distinguish
  - Discriminator net is discarded at test time
- Can be used to generate samples of data without prior knowledge of the data

Generative Adversarial Networks (GAN)





NN evolution in the past 10 years

- 2009: handwriting recognition prizes with LSTM
- 2011 DanNet (CNN) beats humans in **visual pattern recognition**
- 2014: GANs, Alexa, Tesla AutoPilot
- 2015: FaceNet starts facial recognition programs
- 2016: AlphaGo **beats Go champion**
- 2017: Google **Translate** and Facebook Translate based on two LSTMs
- 2018: Cambridge analytica, deep fakes
- 2019: DeepMind defeats professional StartCraft players with RL + LSTM, Rubik's Cube solved with a Robot Hand

Hands-on today

1. **Point your browser to:** https://yoga.to.infn.it
2. **Open a terminal:**
    - cd MLCourse-1920
    - git pull
    - cp Notebooks/Day3/* ../
3. **From JupyterHub Home tab:**
    - start and run *Keras.ipynb*

- *Deep learning with* Keras
    - FF NN