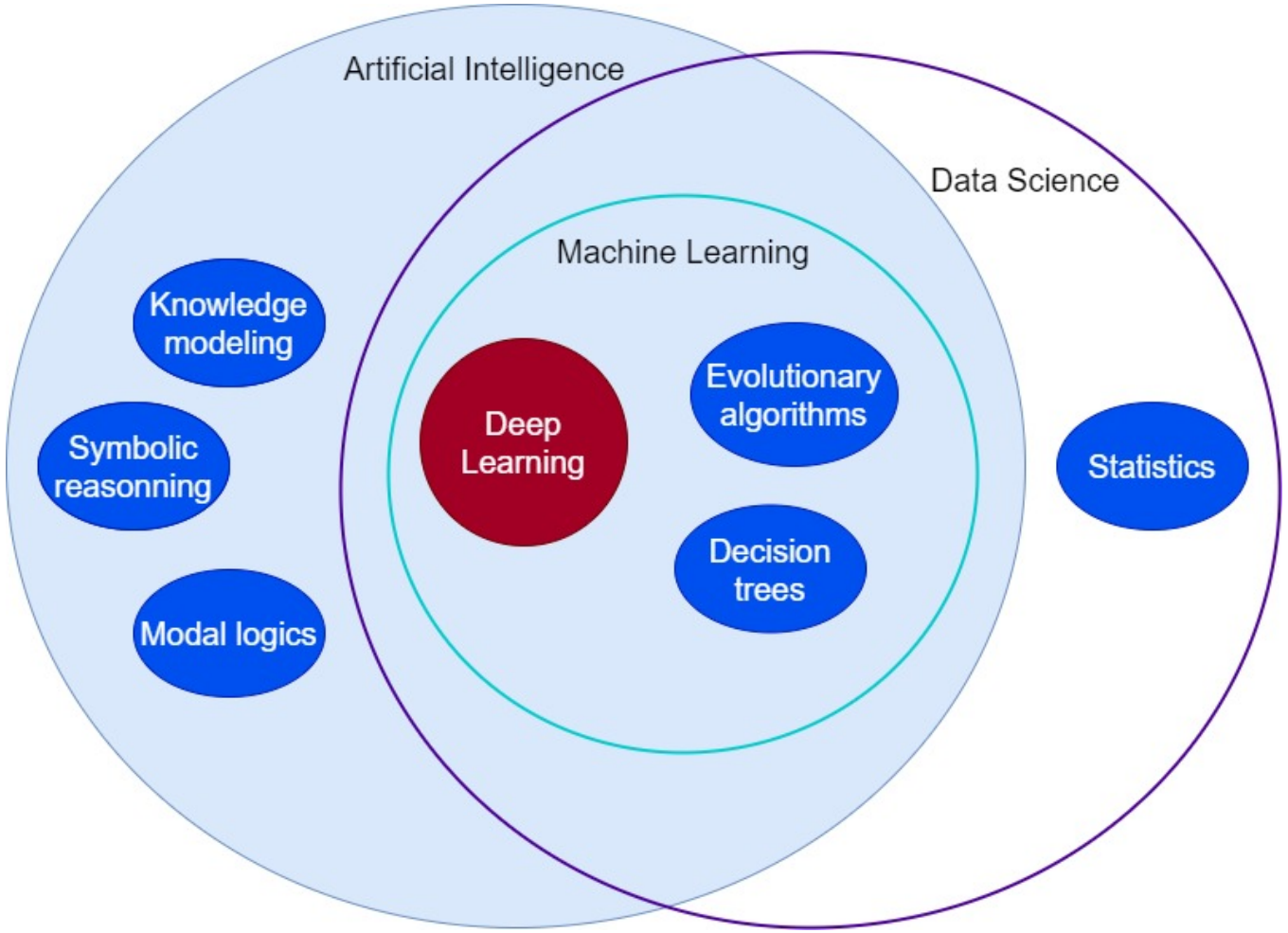
Artificial intelligence for signal applications

Corentin Menier
Phd student

Université Gustave Eiffel

corentin.menier@esiee.fr

What is AI



Artificial Intelligence
A technique which enables machines to mimic human behavior.

Machine Learning
A subset of AI technique which use statistical methods to enables machines to improve with experience.

Deep Learning
A subset of machine learning make the computation of multi-layer neural networks feasible.

https://www.slidesalad.com/product/deep-learning-powerpoint-template-designs/

Machine learning vs Deep learning

Machine learning	Deep learning
A subset of AI	A subset of machine learning
Can train on smaller data sets	Requires large amounts of data
Requires more human intervention to correct and learn	Learns on its own from environment and past mistakes
Shorter training and lower accuracy	Longer training and higher accuracy
More interpretable results	Makes non-linear, complex correlations
Can train on a CPU (central processing unit)	Needs a specialized GPU (graphics processing unit) to train

https://www.coursera.org/articles/ai-vs-deep-learning-vs-machine-learning-beginners-guide

Types of training

- Supervised Learning

  $y = f(x)$

  Known inputs and labels

- Unsupervised Learning

  $f(x)$

  Known inputs

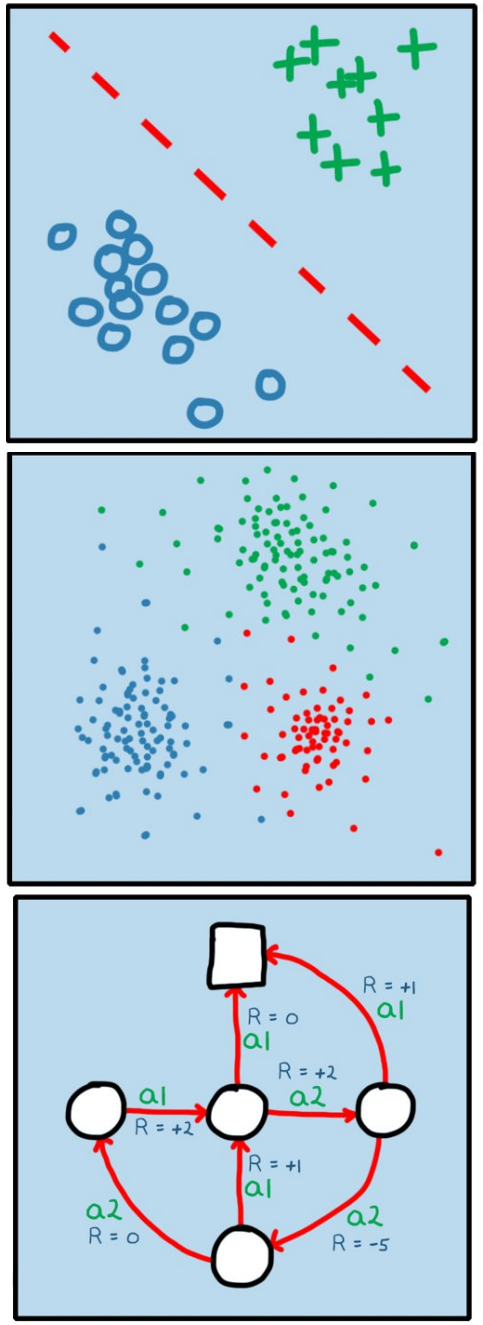
  No known labels

- Reinforcement Learning

  $y = f(x)$

  z

  Known inputs and environment outputs

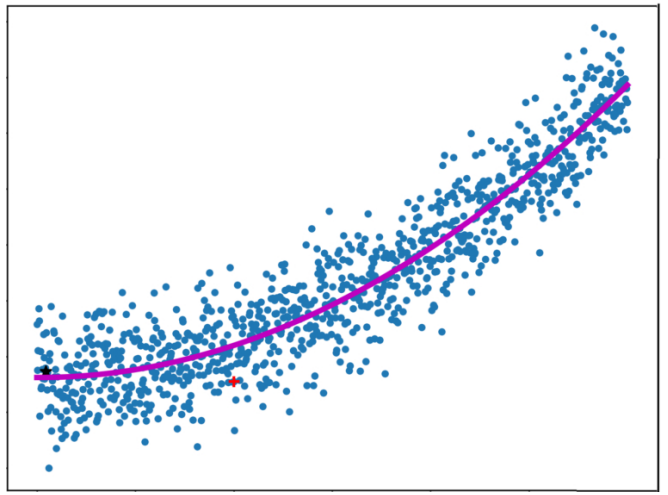


https://fr.mathworks.com/discovery/reinforcement-learning.html

Supervised Learning

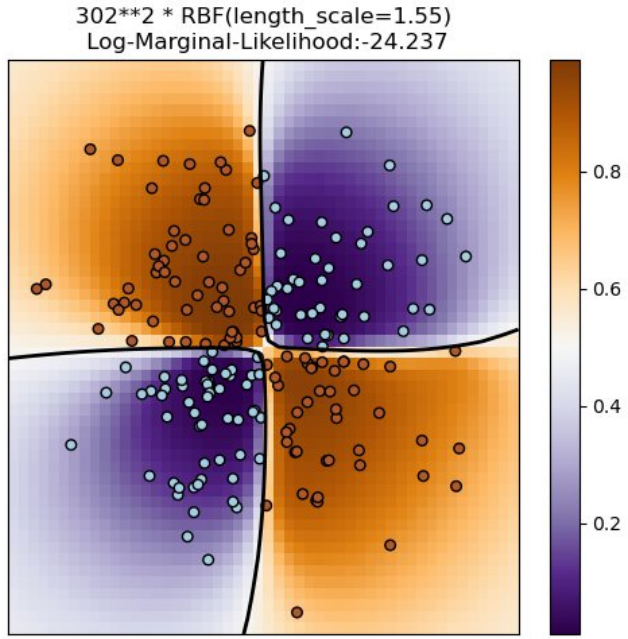
Fit a function : $y = f(x)$

given pair : $\{(x_i, y_i)\}$



https://www.engineersgarage.com/machine-learning-algorithms-classification/

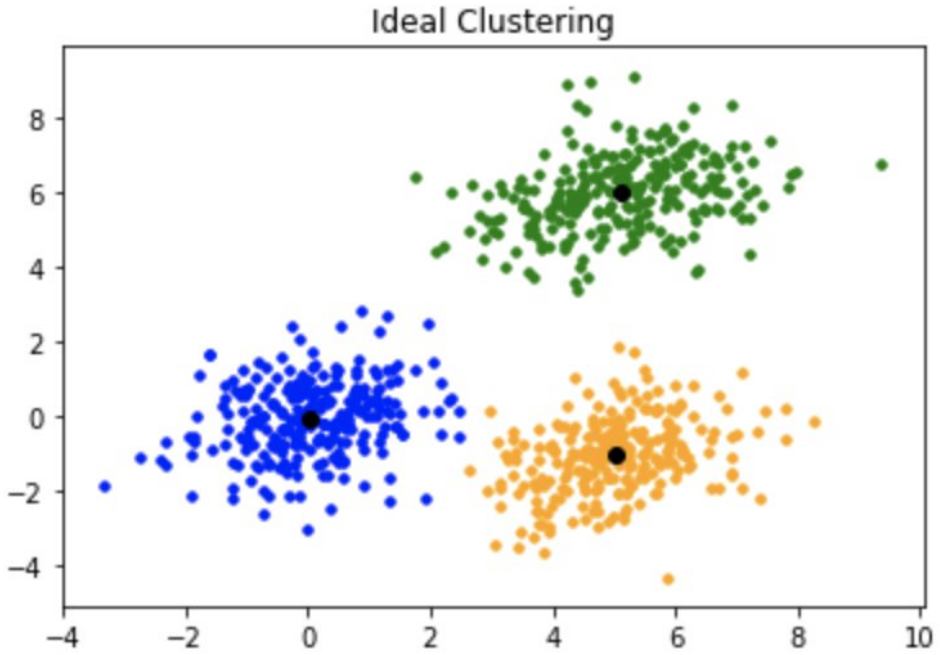
- Two main tasks :
  - Regression : data is continuous (temperatures, house pricing ...)
  - Classification : data is discrete (image of cat, dog, both...)
- Based on data/label pairs



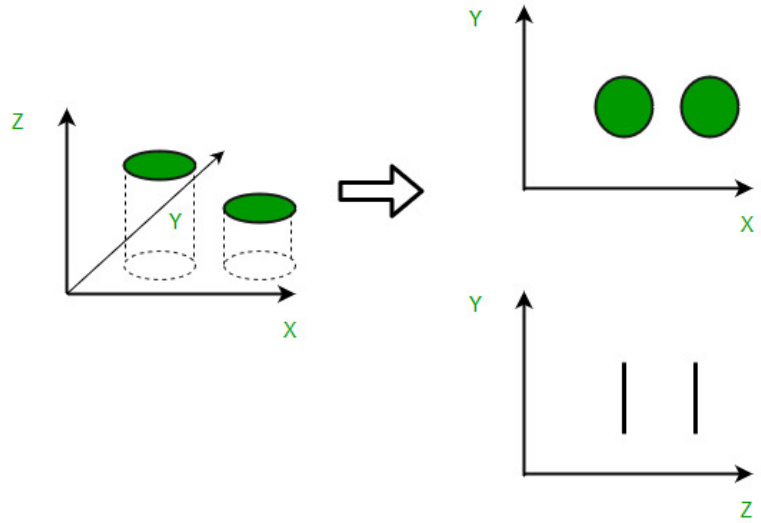
https://scikit-learn.org/stable/modules/gaussian_process.html#gaussian-process-regression-gpr

Unsupervised Learning

- Learn the underlying structure in the data without any label
- Two main tasks :
  - Clustering : find the underlying structure in data, Number of classes may not be known
  - Dimensionality reduction : identify the most impactful features in the data to make the learning process easier for the models



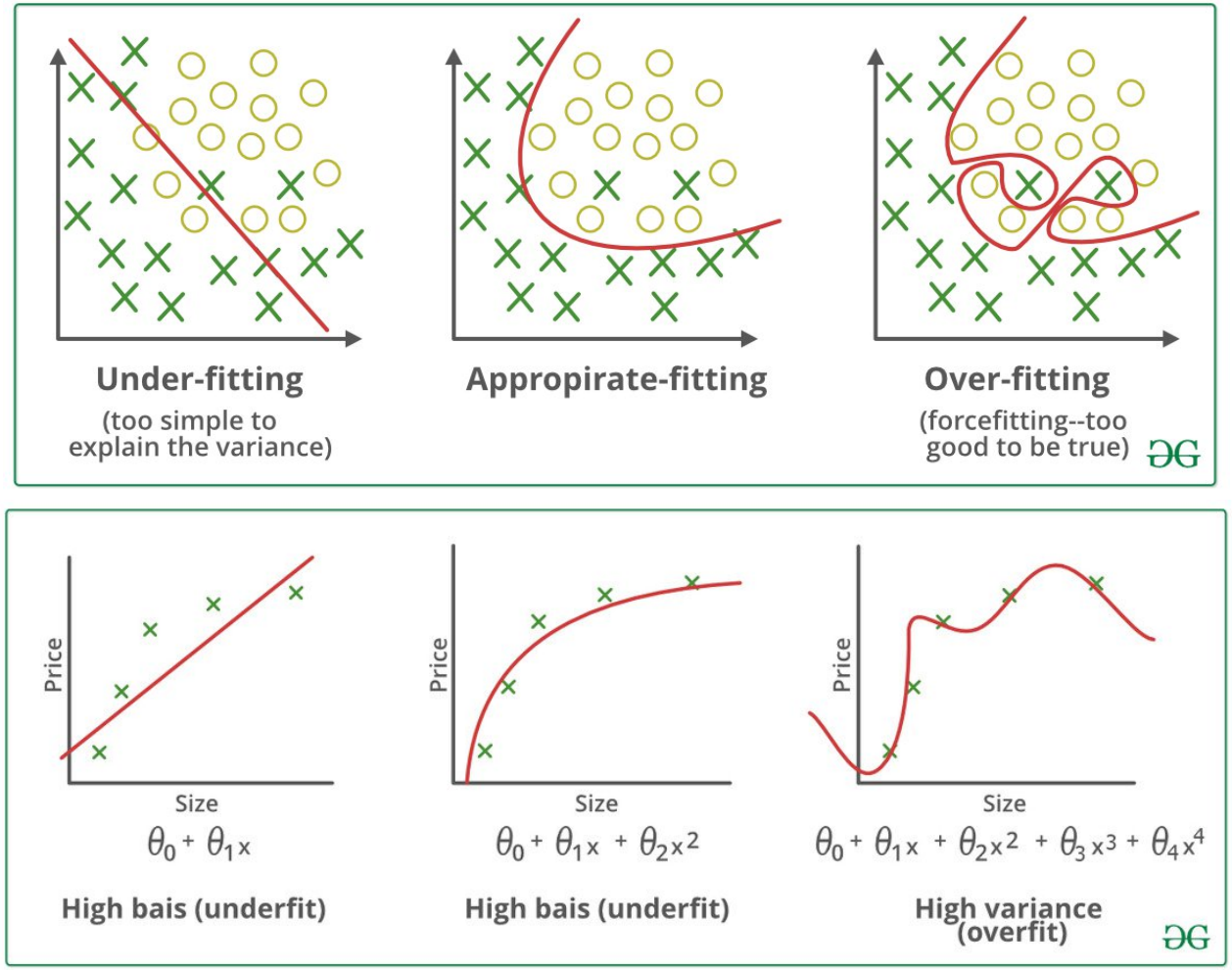
Dimensionality Reduction



source

Model Fitting

- **Underfitting** refers to a model that can neither performs well on the training data nor generalize to new data
- **Overfitting** : Even though the model has low error on training data, it cannot make accurate prediction on new data because of too many details and noise



https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

Model Fitting

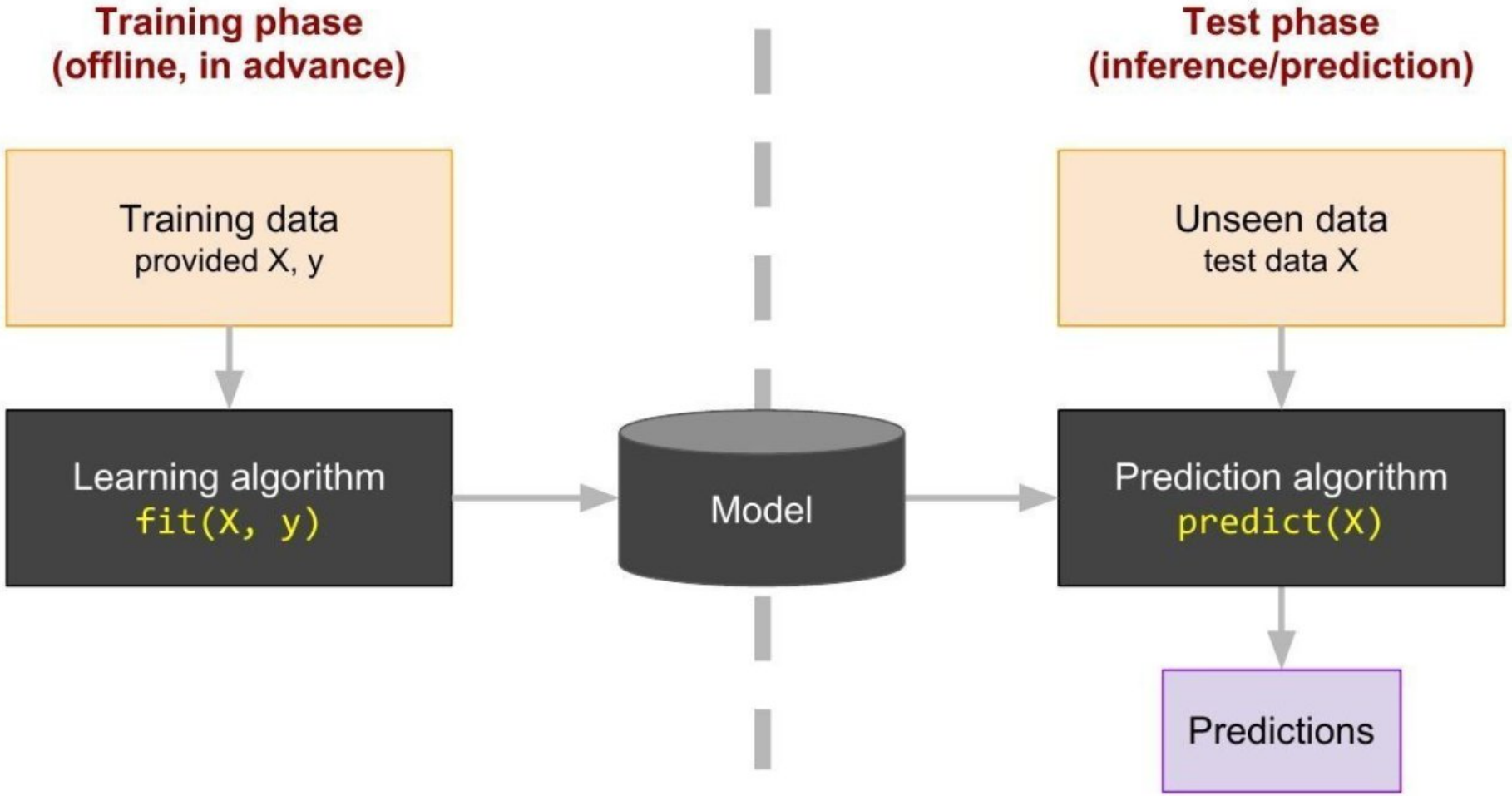
- **Reasons for underfitting** :
  - High bias and low variance
  - The size of the training dataset used is not enough.
  - The model is too simple.
  - Training data is not cleaned and also contains noise in it.

- **Reasons for Overfitting:**
  - High variance and low bias
  - The model is too complex
  - The size of the training data

- **Techniques to reduce underfitting:**
  - Increase model complexity
  - Increase the number of features, performing feature engineering
  - Remove noise from the data.
  - Increase the duration of training to get better results.

- **Techniques to reduce overfitting:**
  - Increase training data.
  - Reduce model complexity.
  - Early stopping during the training phase (have an eye over the loss over the training period as soon as loss begins to increase stop training).

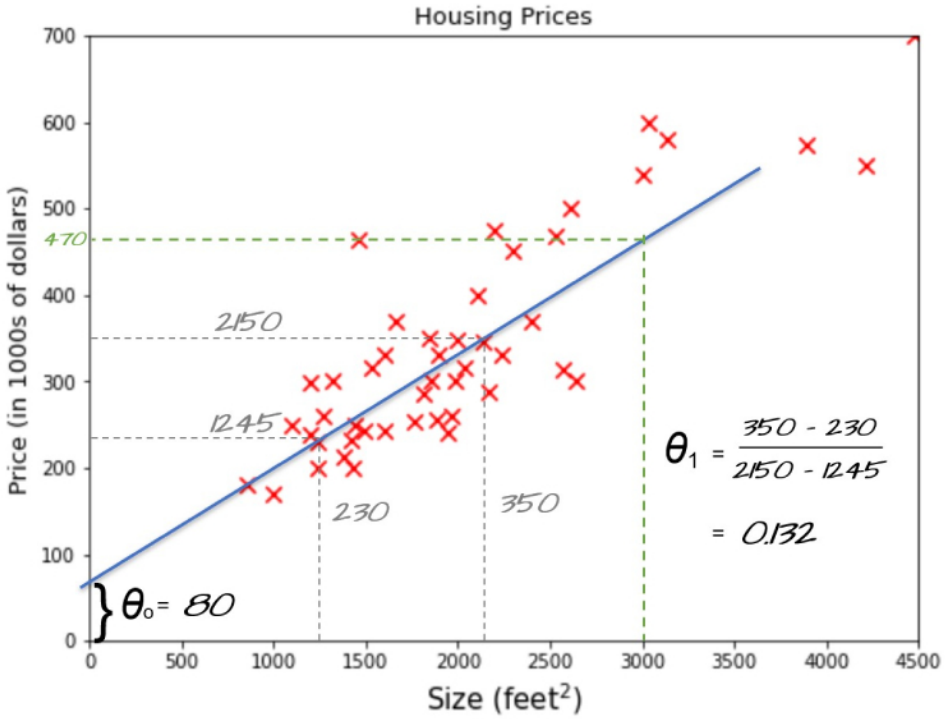
Black Box abstraction of supervised learning



Regression example

- Features x could be :
  - Square foot
  - Number of windows
  - Energy efficiency
- Target y is sale price
- Regression : target is continuous

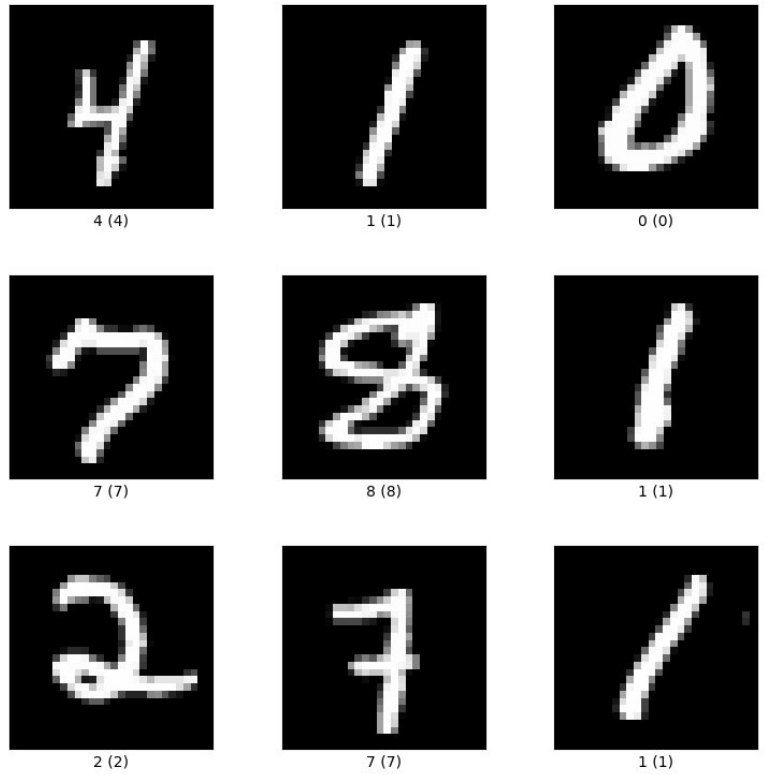
$$f : R^D \rightarrow R$$



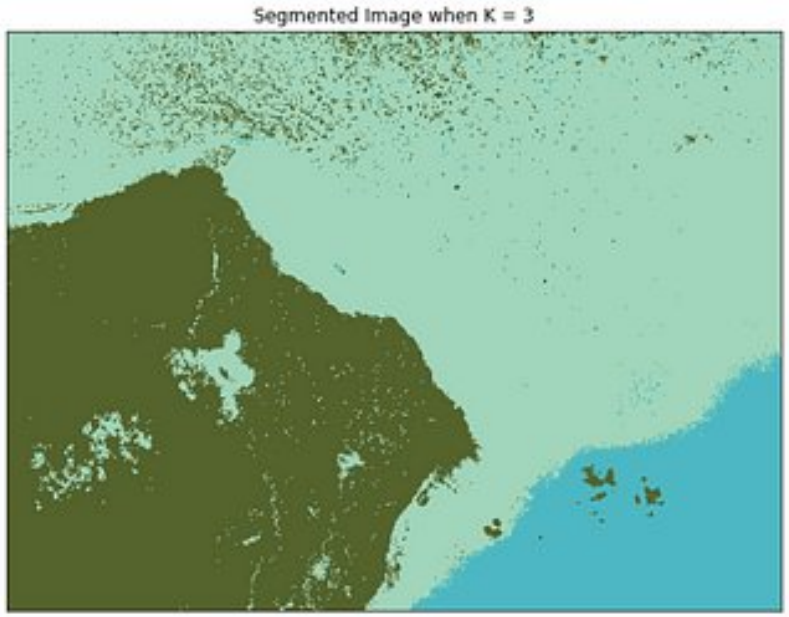
Classification example

- Produce a classifier from pixels to corresponding digits :
  - Since images are grayscale of 28x28 pixels, then $x_i \in R^{784}$
  - $y_i \in \{0,1,2,3,4,5,6,7,8,9\}$
- This is a multi-class classification task

MNIST Dataset

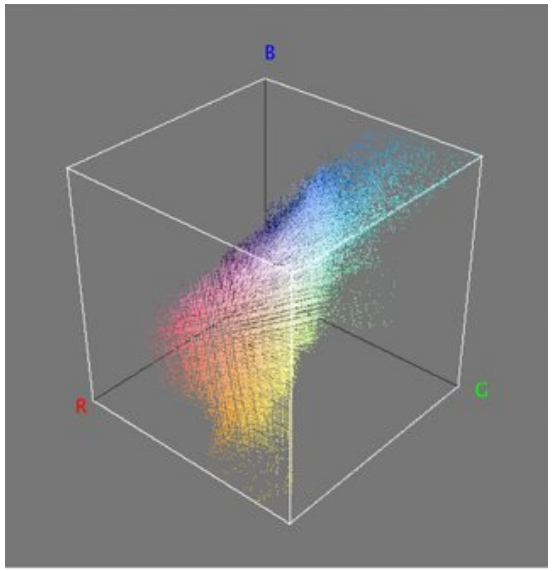


Unsupervised Segmentation example



https://www.kdnuggets.com/2019/08/introduction-image-segmentation-k-means-clustering.html

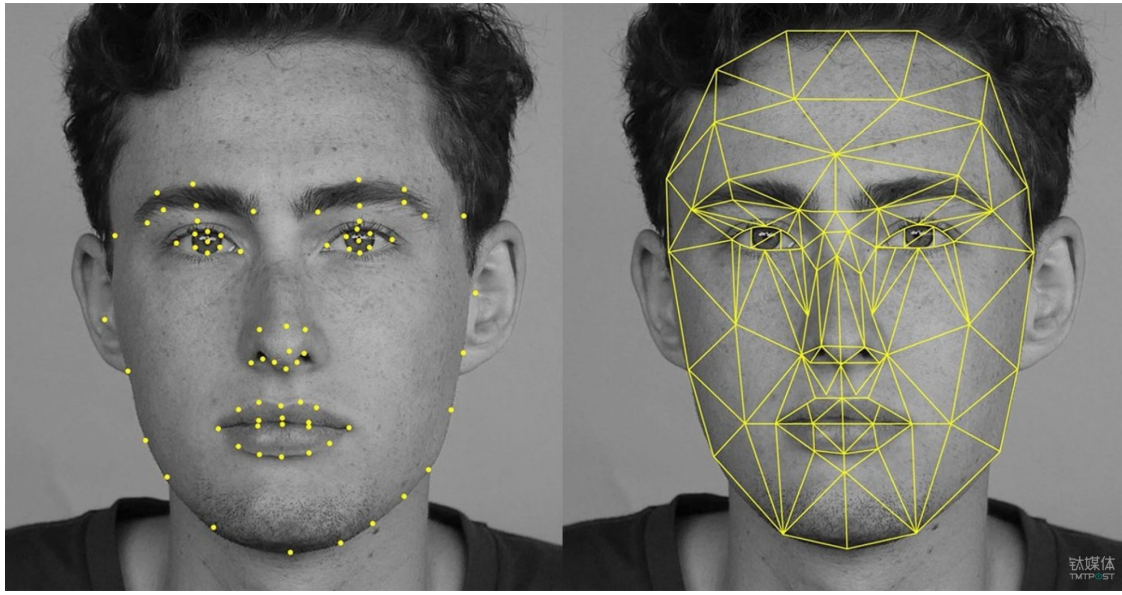
3d pixel histogram

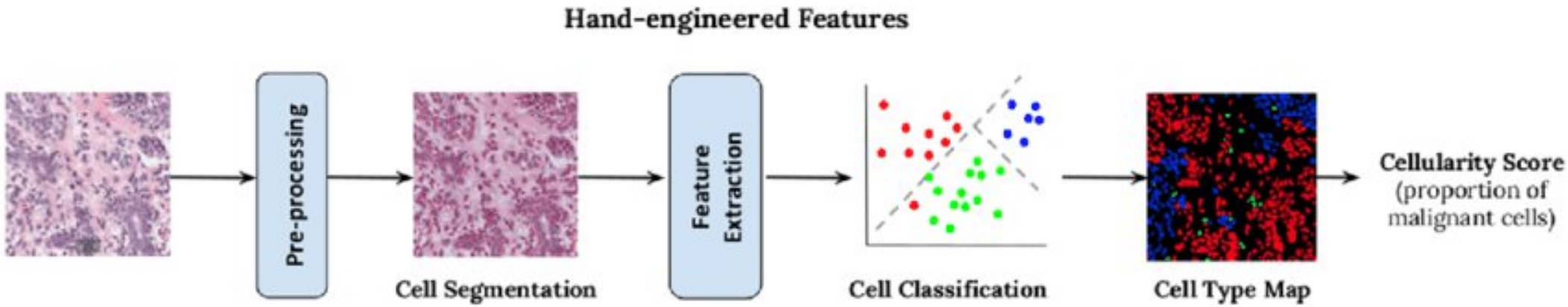


https://pixelcraft.photo.blog/2022/02/17/why-are-there-no-3d-colour-histograms/

Feature extraction

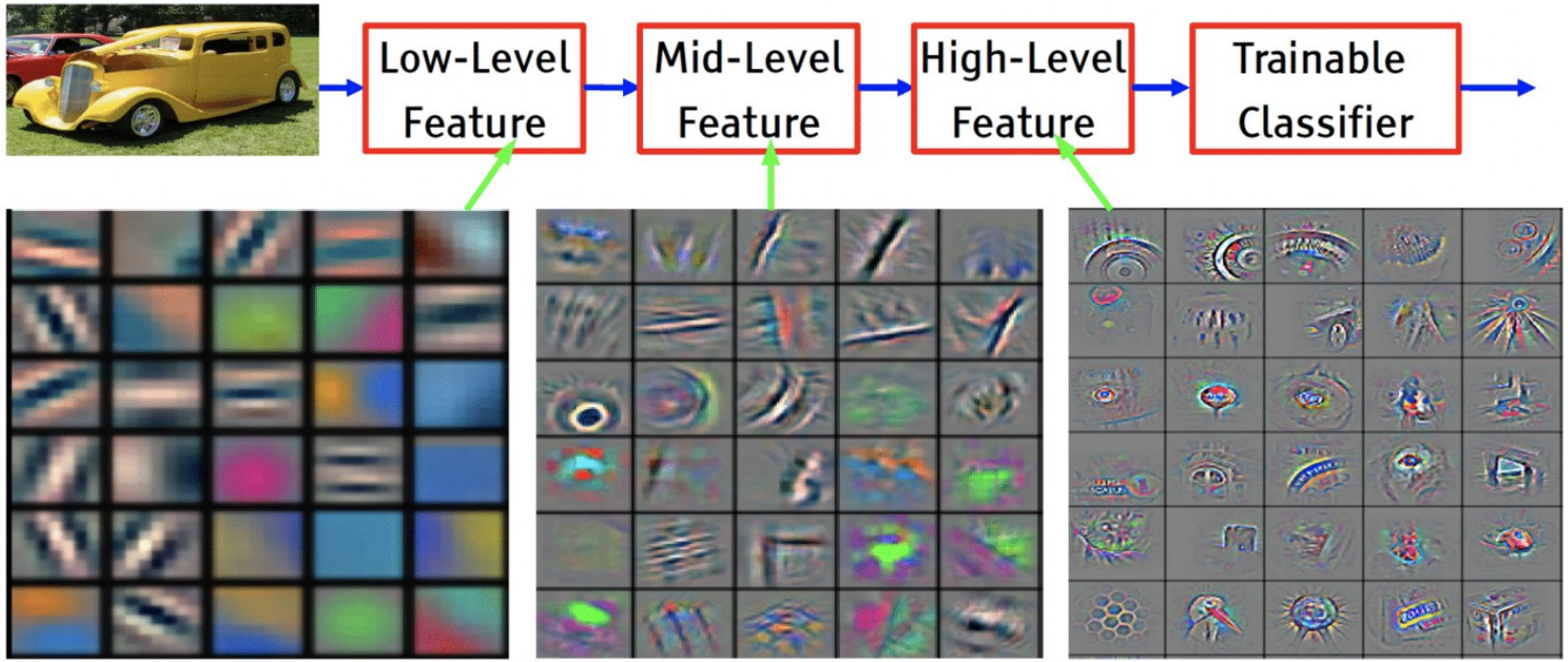
- A feature can be any pattern in the data that characterize the phenomenon we are trying to measure
- The goal of a machine learning pipeline is to identify and organize the most influencial features in the data
- Old ML algorithm relied heavely on hand engineered features, which most often required time and a good expertise to design





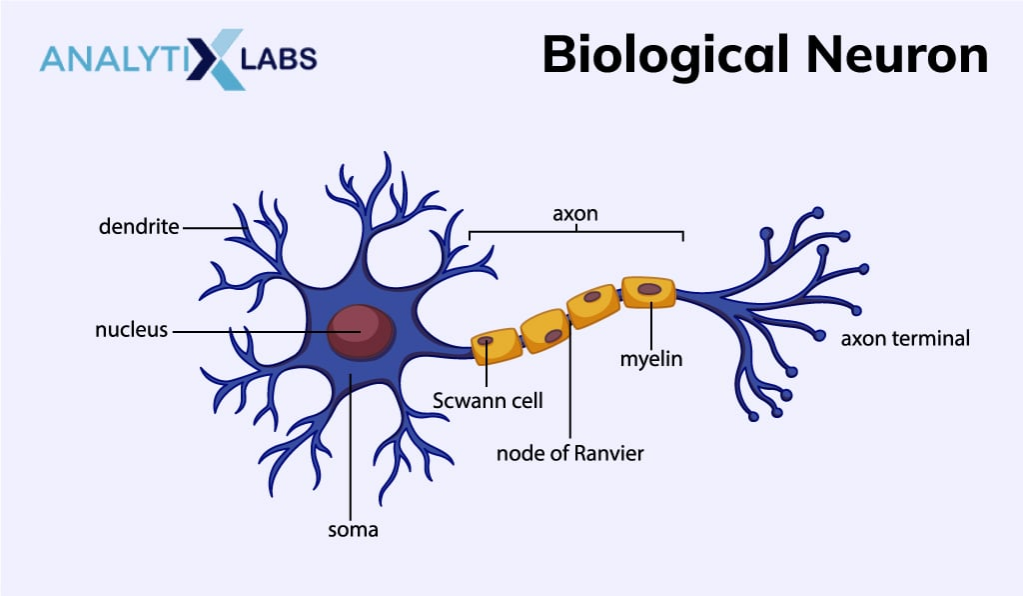
Deep learning

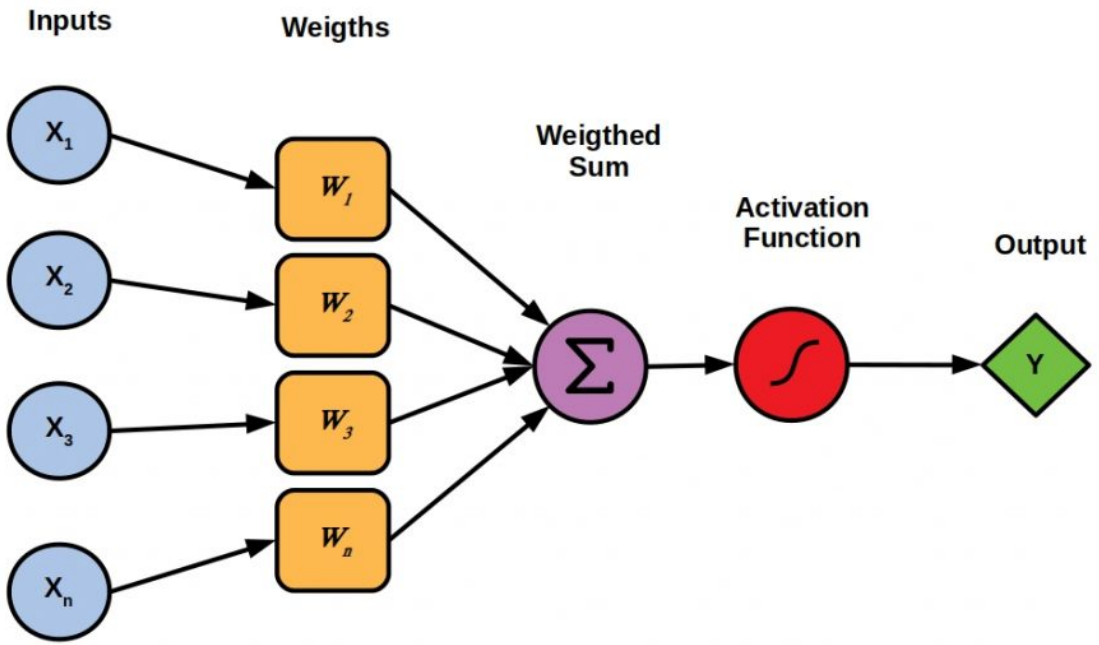
- Hand engineered features are time consuming, brittle and not scalable



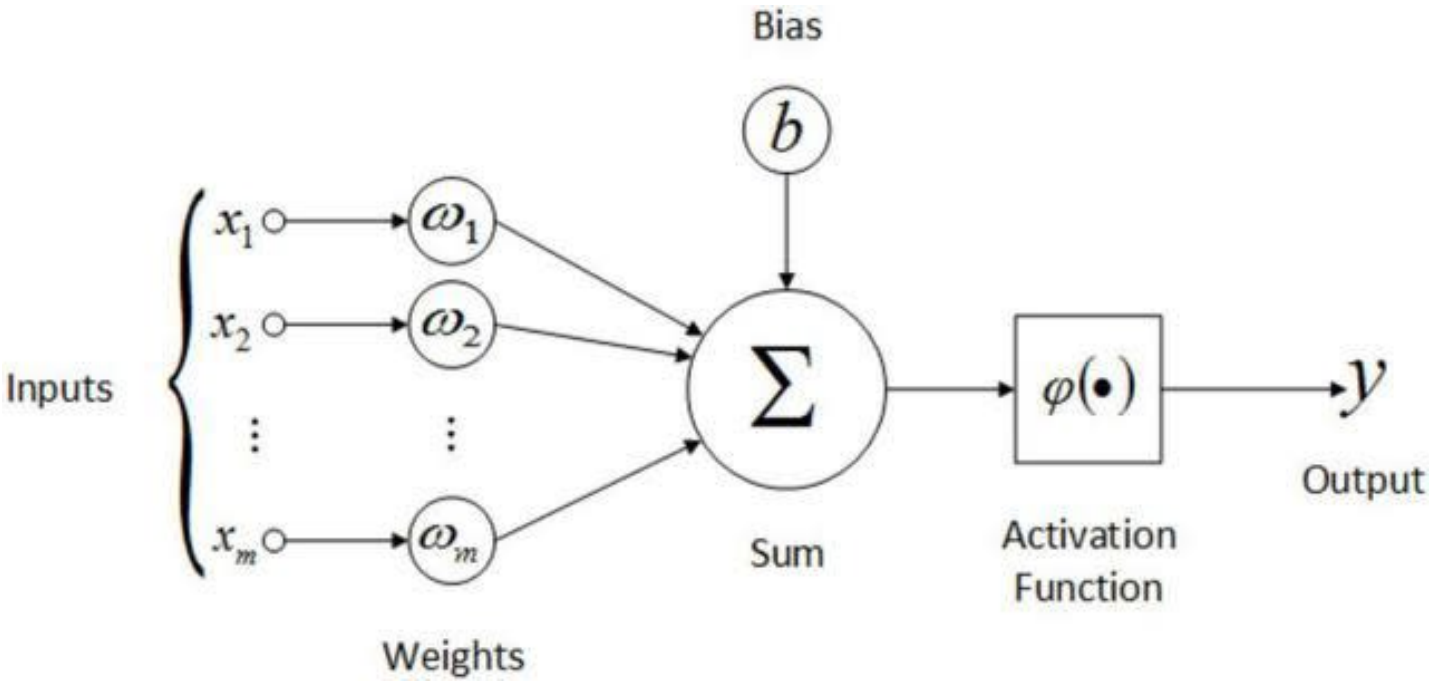
The perceptron (single neuron model)

Biological inspiration





Forward propagation (inference)



$$y = \varphi(b + \sum_{i=1}^{m} x_i . w_i)$$

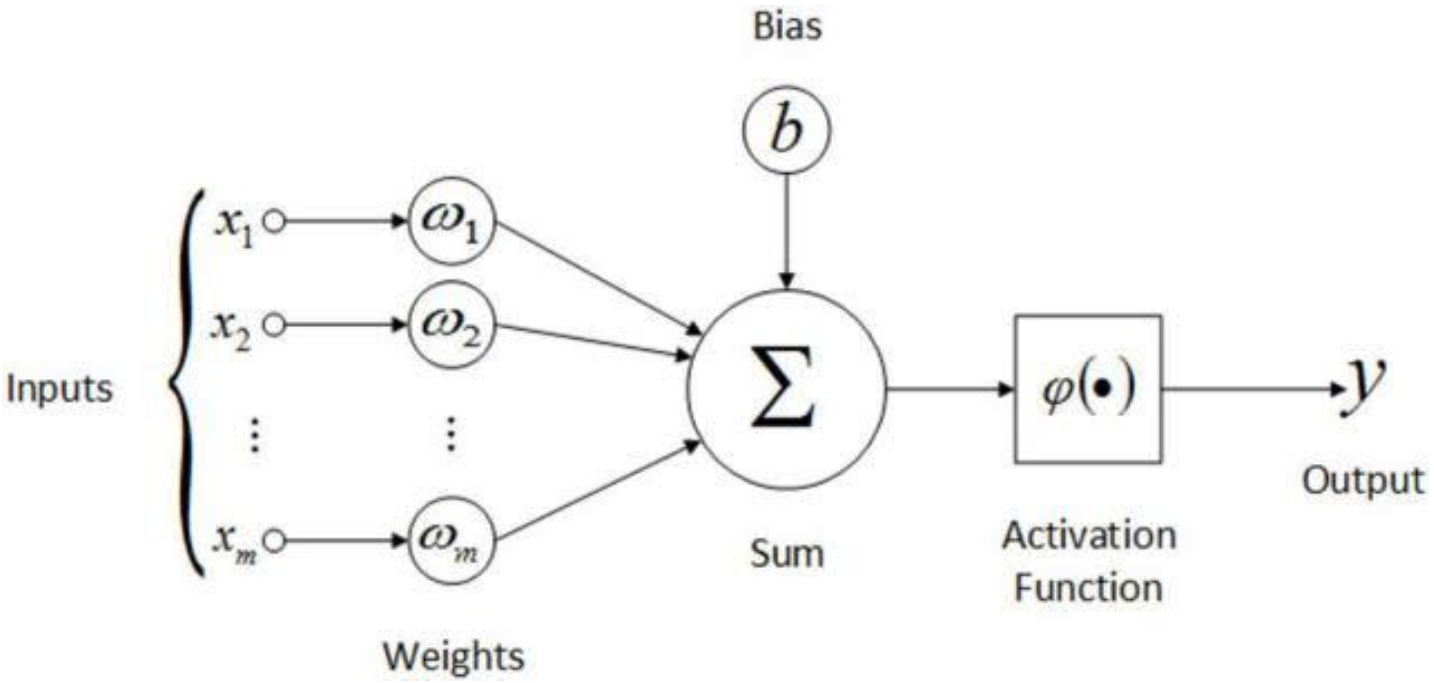
$$y = \varphi(W^T . X + b)$$

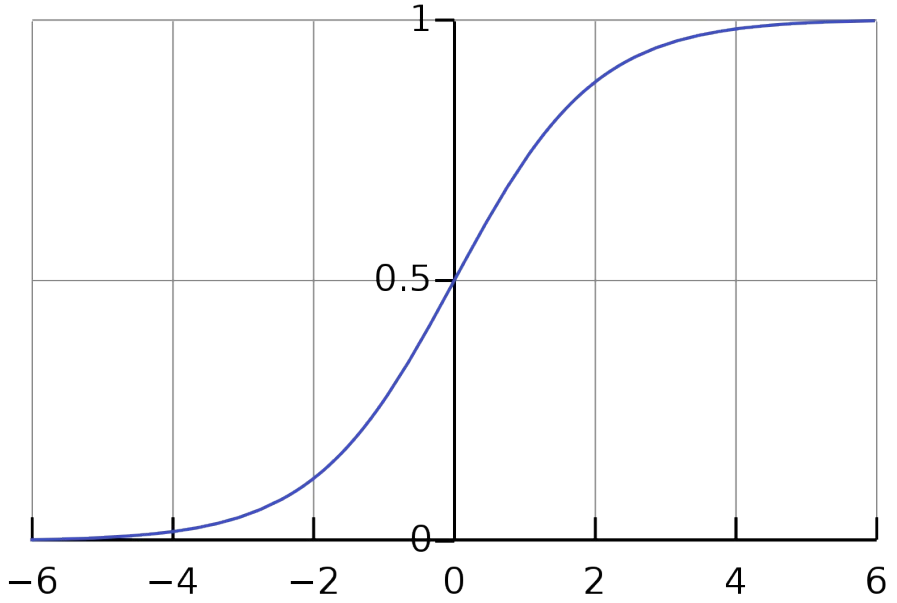
where : $X = \begin{cases} x_1 \\ x_2 \\ x_m \end{cases}$ and $W = \begin{cases} w_1 \\ w_2 \\ w_m \end{cases}$

Activation Functions

$$z = W^T.X + b$$
$$y = \varphi(z)$$

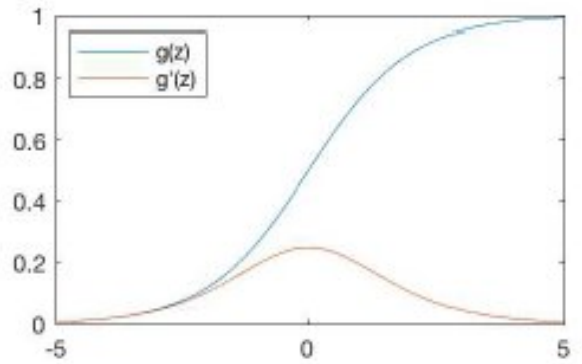
$$\varphi(z) = \sigma(z) = \frac{1}{1+e^{-z}}$$





Common activation functions

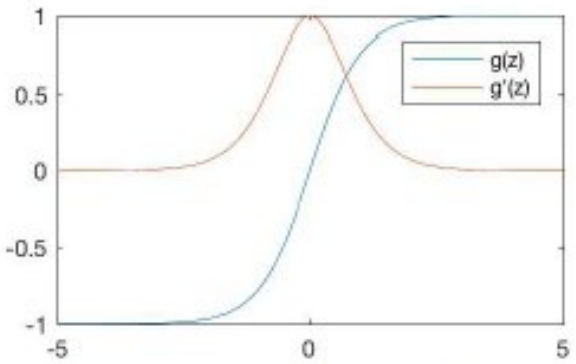
Sigmoid Function



$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

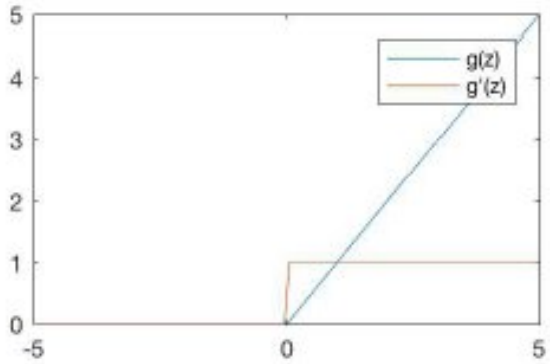
Hyperbolic Tangent



$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

Rectified Linear Unit (ReLU)

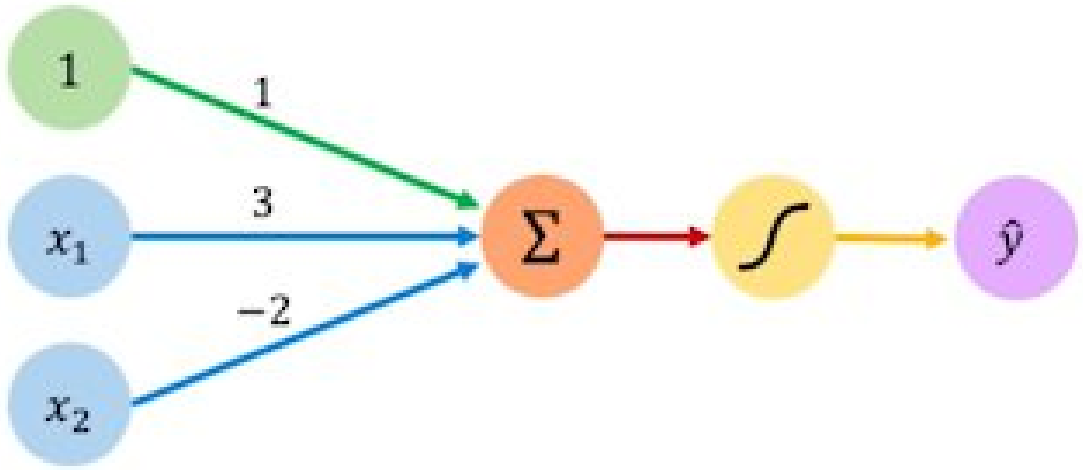


$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

https://nikhilkanta.github.io/-2019-06-18-deep-learning

Perceptron example



We have: $w_0 = 1$ and $\boldsymbol{W} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$

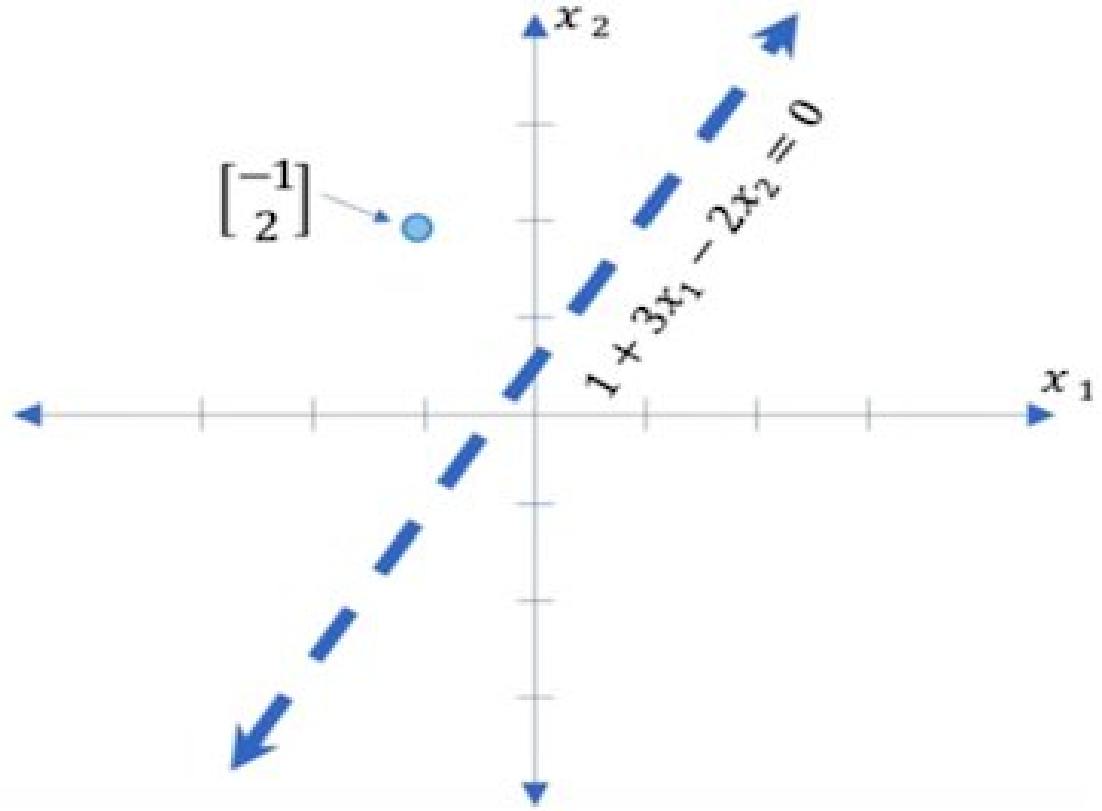
$$\hat{y} = g(w_0 + \boldsymbol{X}^T \boldsymbol{W})$$

$$= g\left(1 + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 3 \\ -2 \end{bmatrix}\right)$$

$$\hat{y} = g(\underbrace{1 + 3x_1 - 2x_2}_{\text{This is just a line in 2D!}})$$

Perceptron example

$$\hat{y} = g(1 + 3x_1 - 2x_2)$$



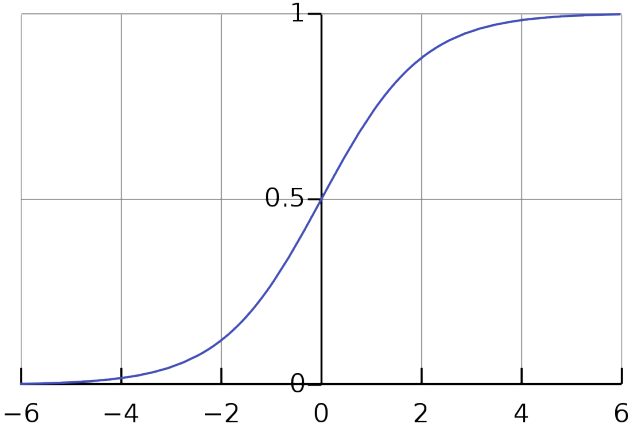
Assume we have input: $\boldsymbol{X} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$

$$\hat{y} = g\big(1 + (3 * -1) - (2 * 2)\big)$$
$$= g(-6) \approx 0.002$$

$$z = W^T.X + b$$

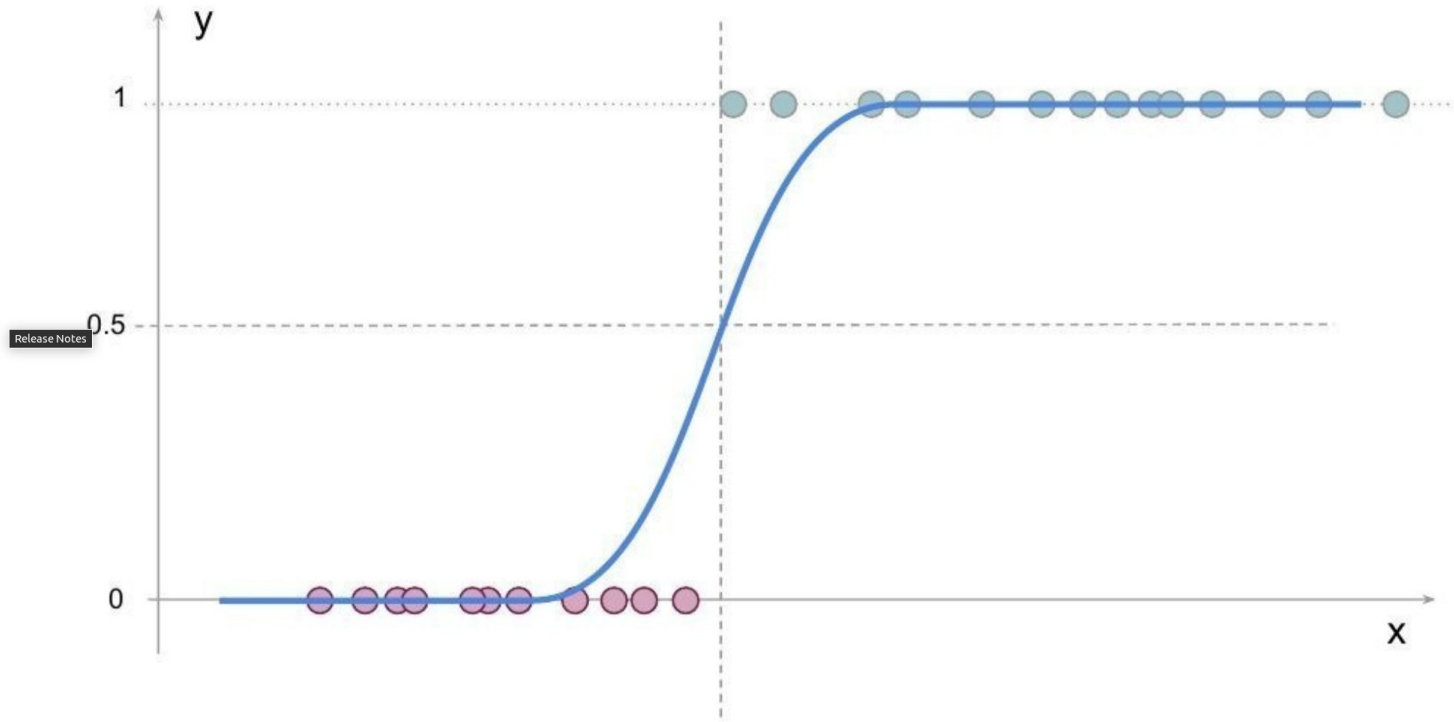
$$y = \sigma(z)$$

if $z < 0$ then $y < 0.5$



Logistic regression

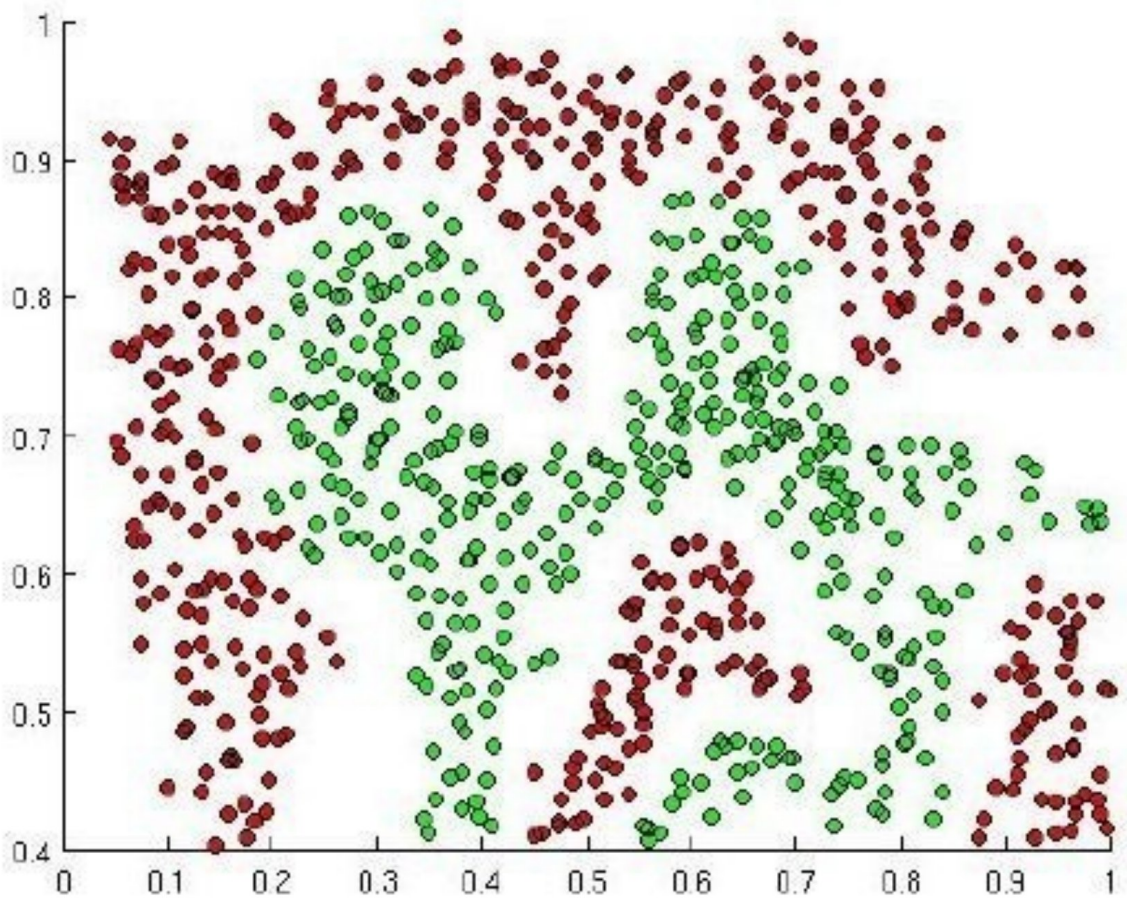
- $f(x) = \varphi(w^T.x + b)$
- Activation : sigmoid
- Mainly used in binary classification



Deep neural network model

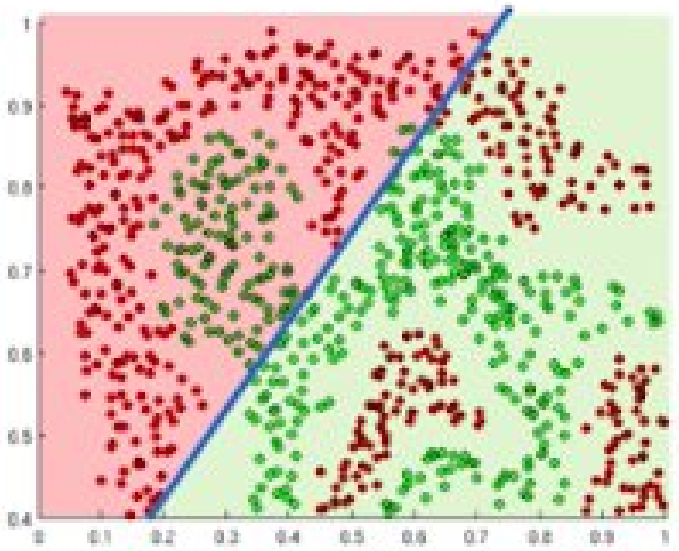
Non Linearity in the model

- Perceptrons can only produce linear decisions boundaries
- Many problems are not linearly separable
  - Images
  - Audio
  - Text

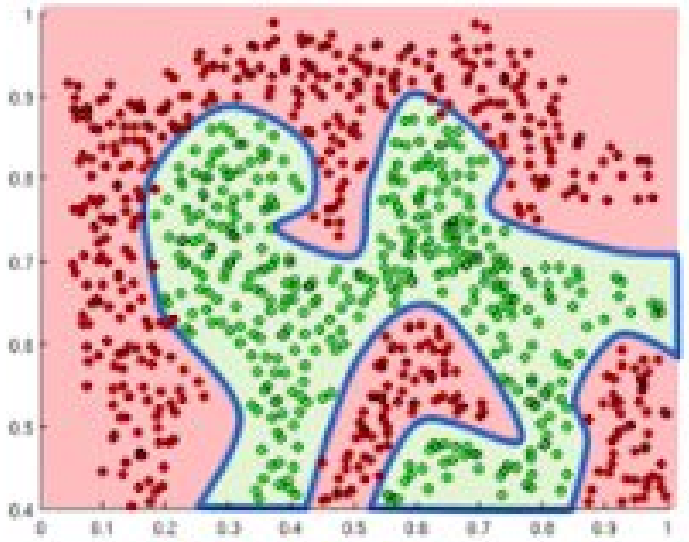


Non linearity in the model

The purpose of activation functions is to ***introduce non-linearities*** *into the network*

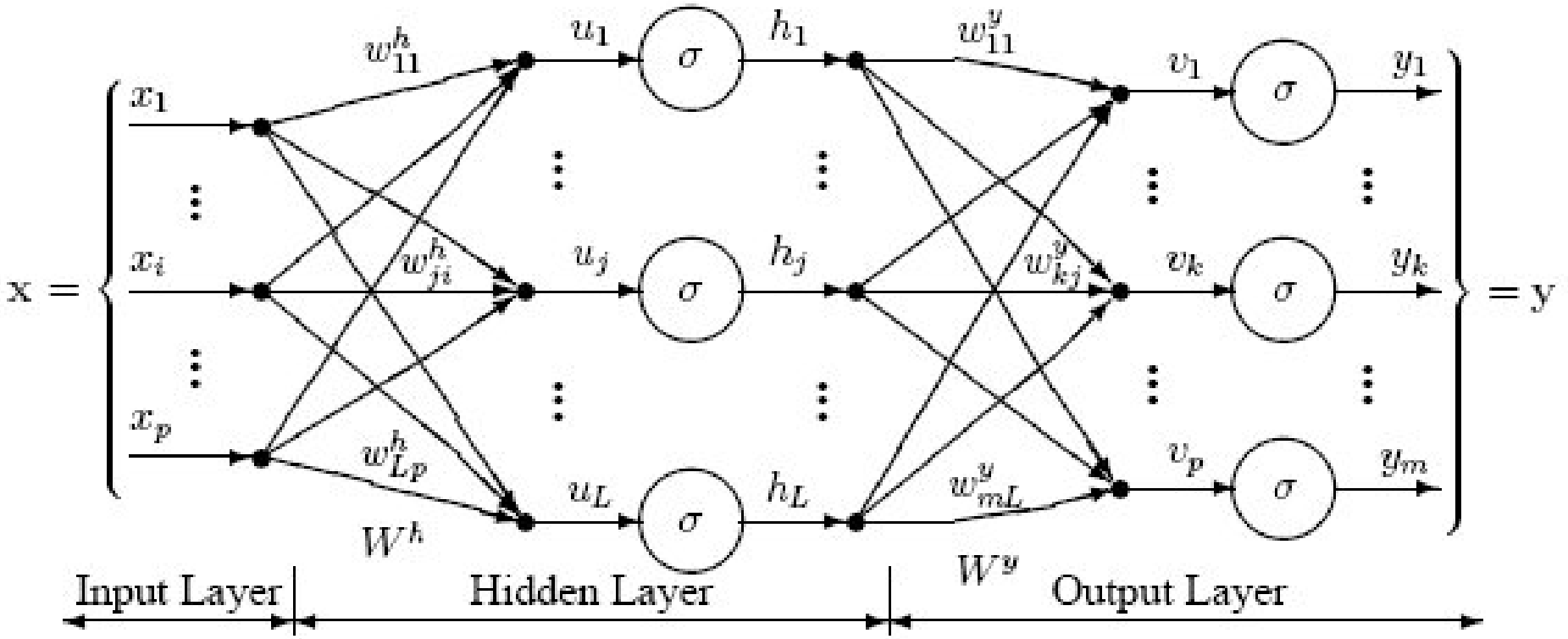


Linear activation functions produce linear decisions no matter the network size



Non-linearities allow us to approximate arbitrarily complex functions

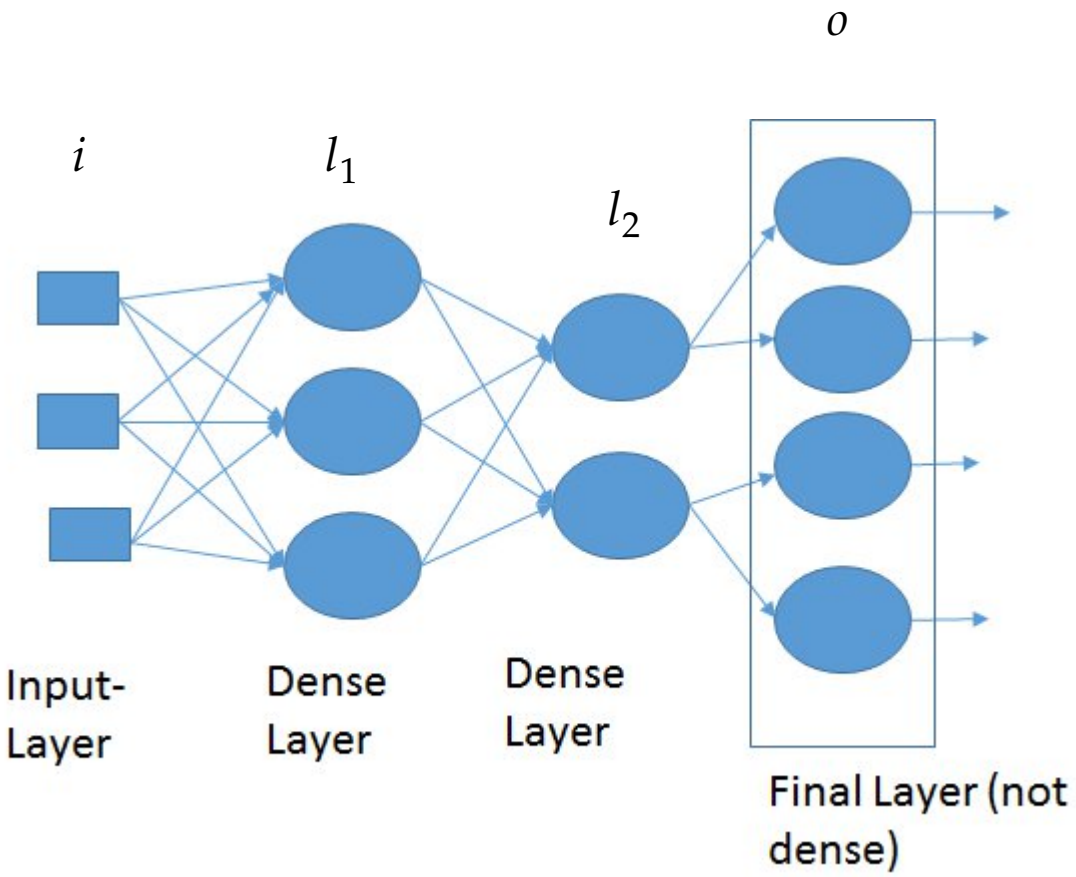
Multi layer model (deep model)



- $h_1 = \sigma(w^h_{11}.x_1 + b^h_1)$
- $y_1 = \sigma(w^y_{11}.h_1 + b^y_1)$
- Non linearity is necesseary to avoid all layers being equal to one layer

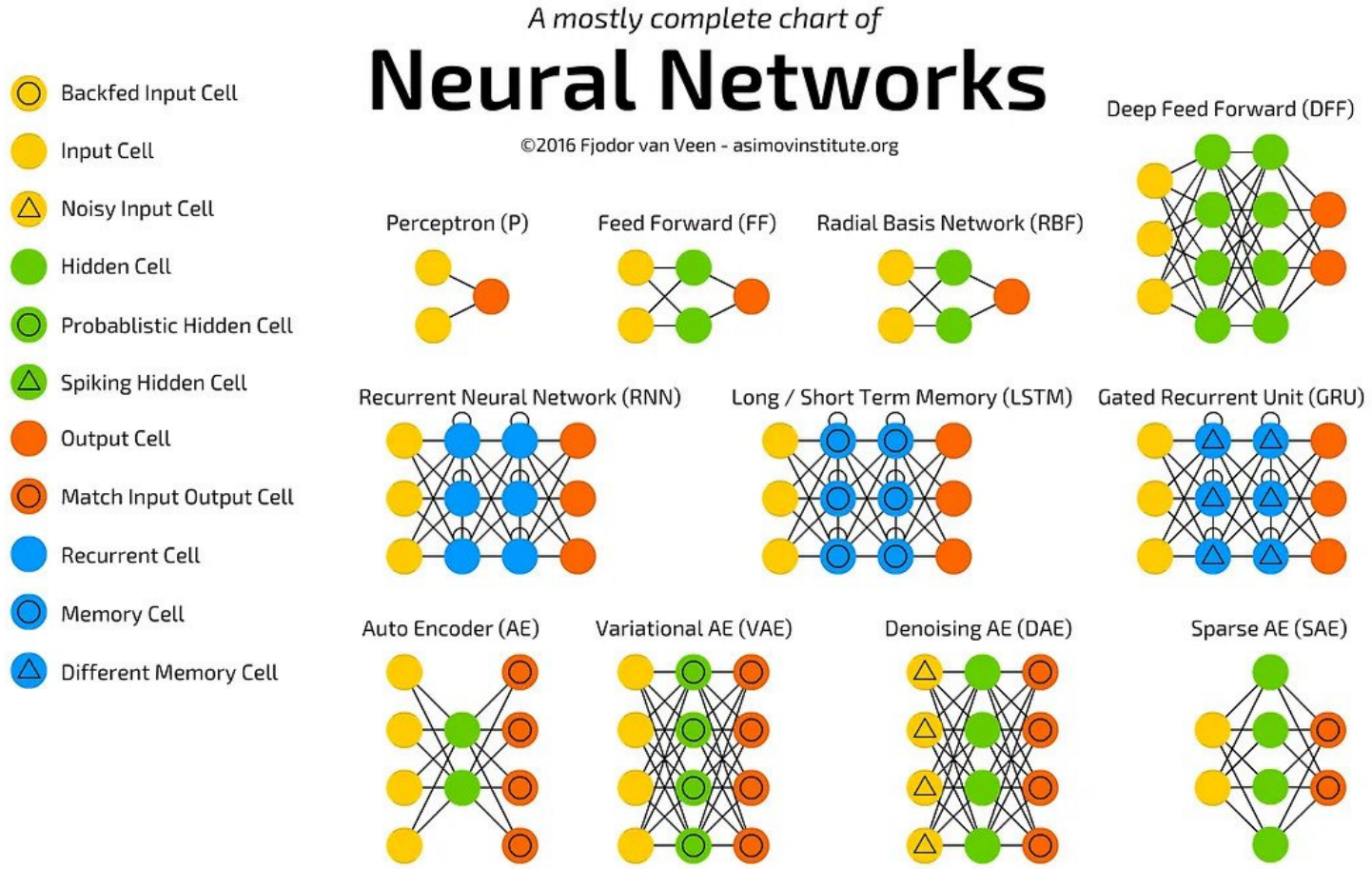
Dense Layer Type

- Hidden layers
- Two hidden layers and one output layer
- Dense layers :
  - most basic types
  - high computationality cost
- Number of parameters on a layer: $N^i_{params} = N^i_{inputs} * N^i_{units} + N^i_{units}$
- Many other hidden layer types exist



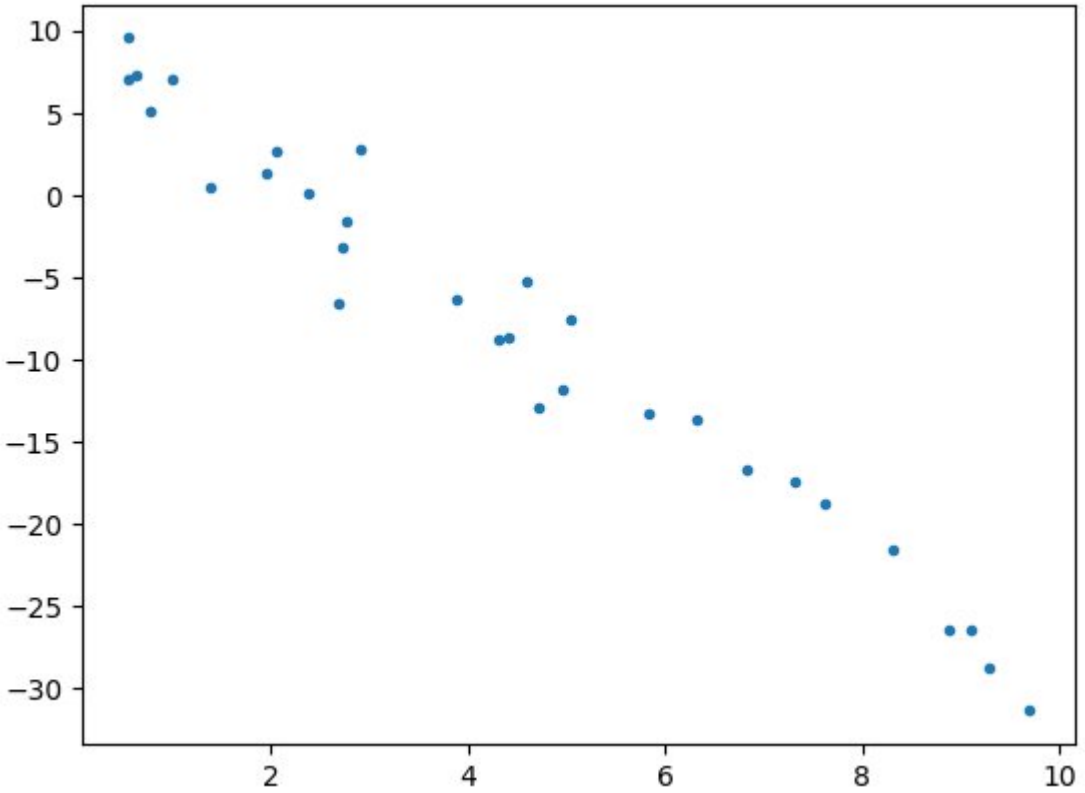
Multi layer model (deep model)

- Diversity of layer type and tasks, such as dense layer
- Output of layers, named features, can change in dimensionality and are a representation of the output
- The closer to the start, the more related to the input the features are
- The deepest features are the closest to the target structure



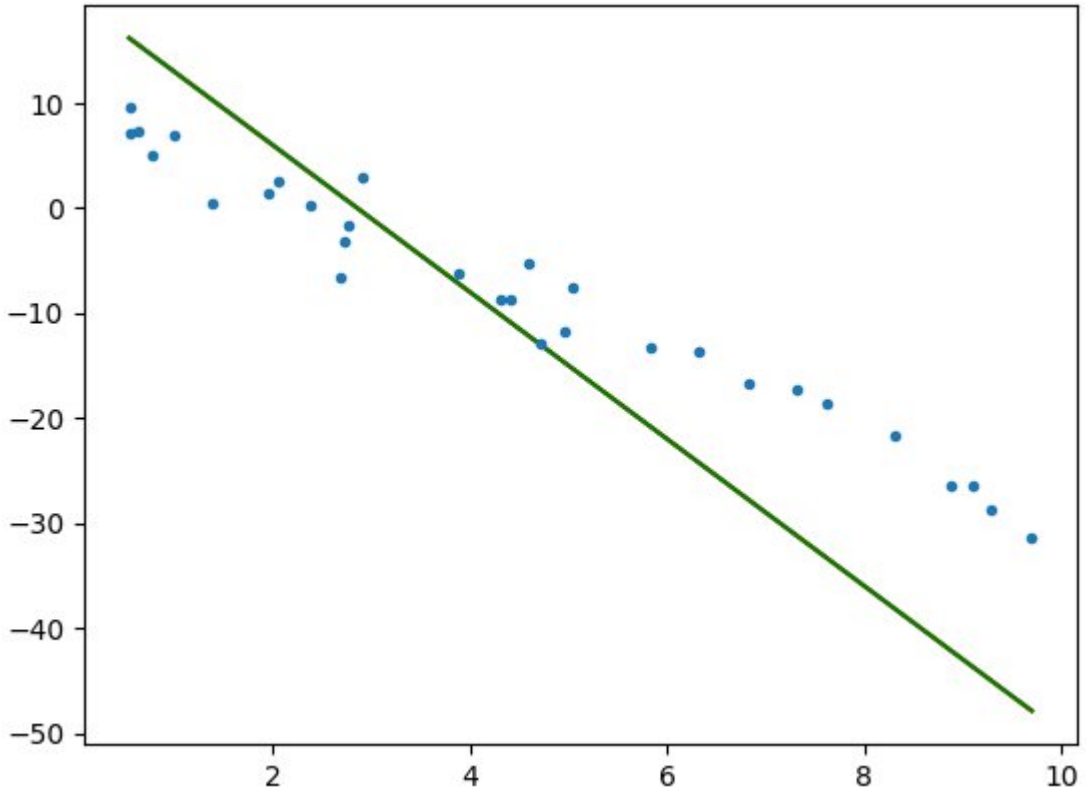
Training of DNN

Linear regression



- $y = a.x + b$
- How do i find the equation from the data ?

Linear regression



- Make a randomn estimate of the model parameters

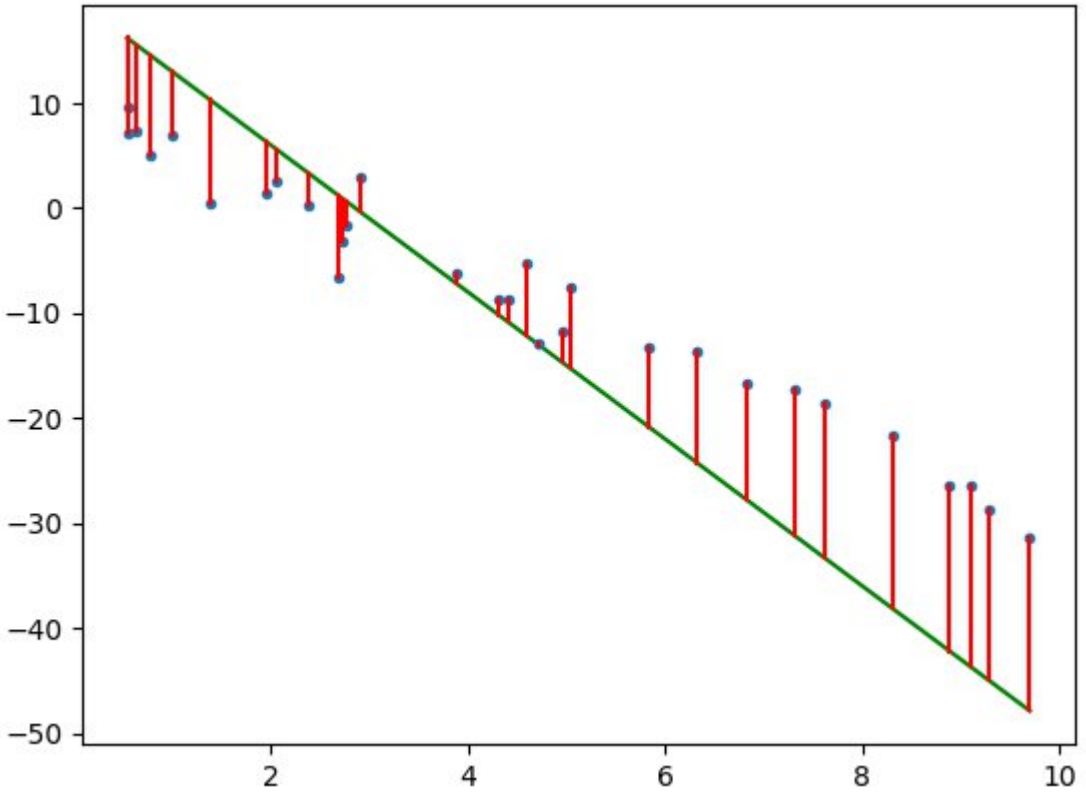
Linear regression

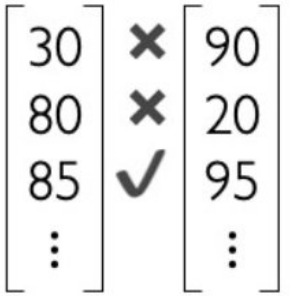
- We first need a metric to quantify the performance of our model

- Introduction of a Loss function (or cost function):

  - $loss = \mathcal{L}\left(\underbrace{f\left(x^{(i)}; \boldsymbol{W}\right)}_{\text{Predicted}}, \underbrace{y^{(i)}}_{\text{Actual}}\right)$

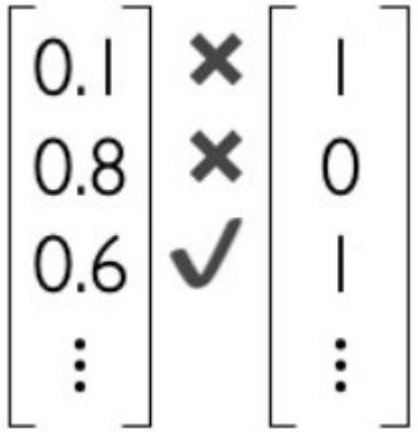
- MeanSquareError $= \frac{1}{n}\sum_{i=1}^{n}\left(\underbrace{y^{(i)}}_{\text{Actual}} - \underbrace{f\left(x^{(i)}; \boldsymbol{W}\right)}_{\text{Predicted}}\right)^2$

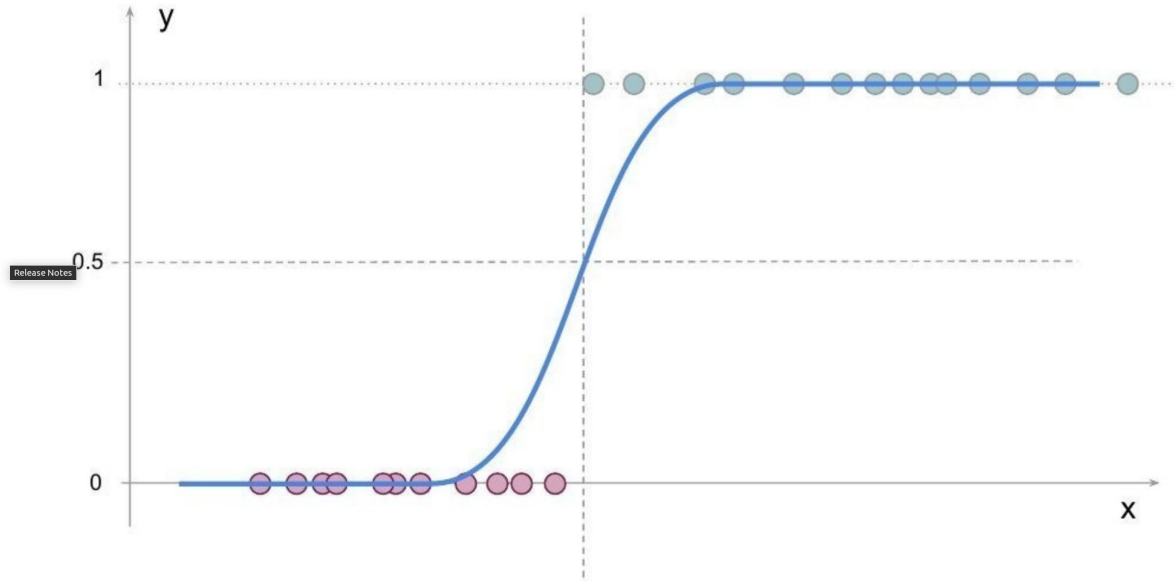




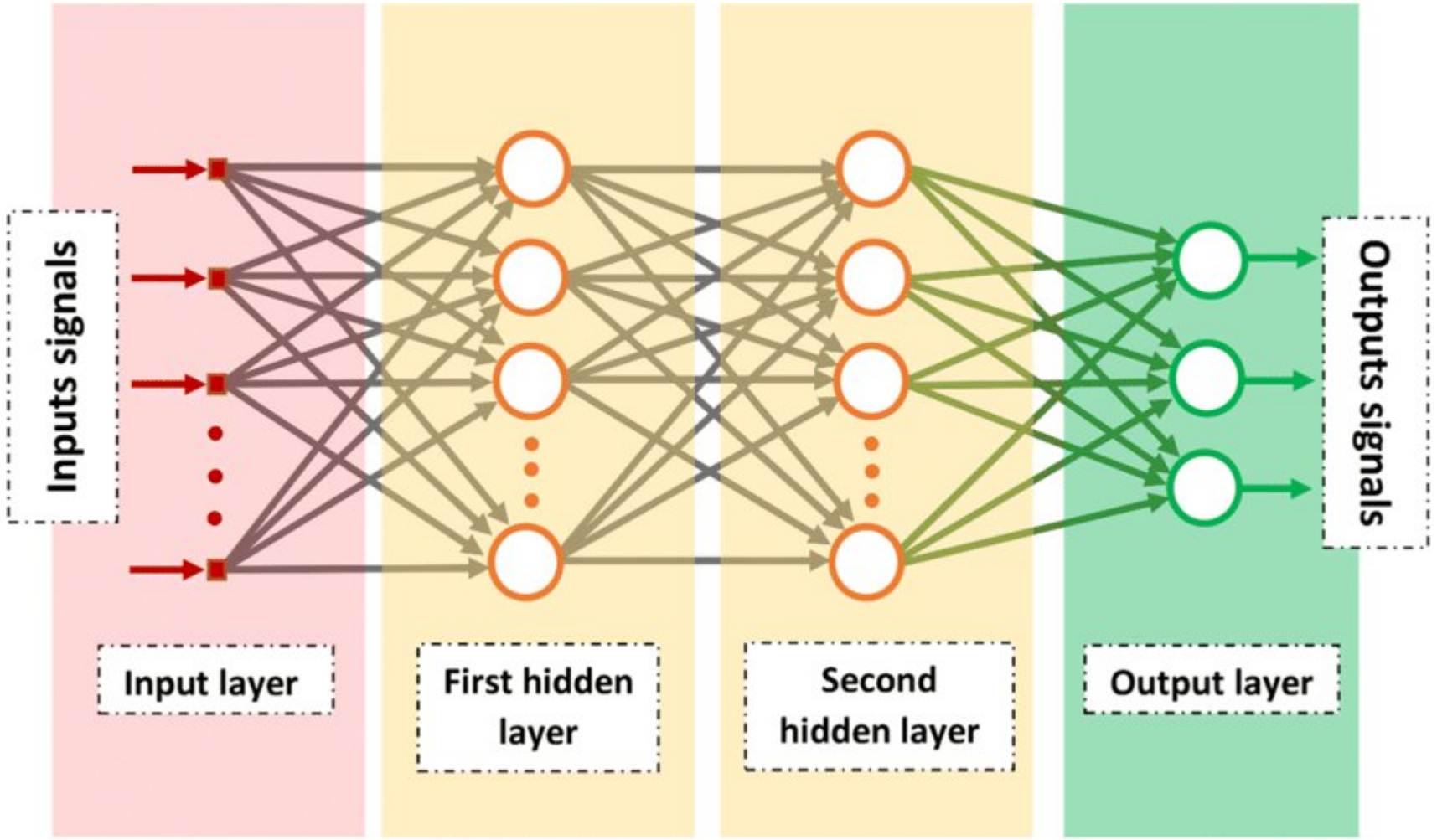
Other types of loss

- Binary cross enthropy = $-\frac{1}{n}\sum_{i=1}^{n} y^i \log f\left(x^i;W\right) + (1-y^i)\log\left(1-f(x^i;W)\right)$
- Used for cases where the output is a probability between 0 and 1
- Example : logistic regression





Multi output case



https://www.researchgate.net/publication/337718785_A_new_artificial_multi-neural_approach_to_estimate_the_hourly_global_solar_radiation_in_a_semi-arid_climate_site

One hot encoding

Label Encoding

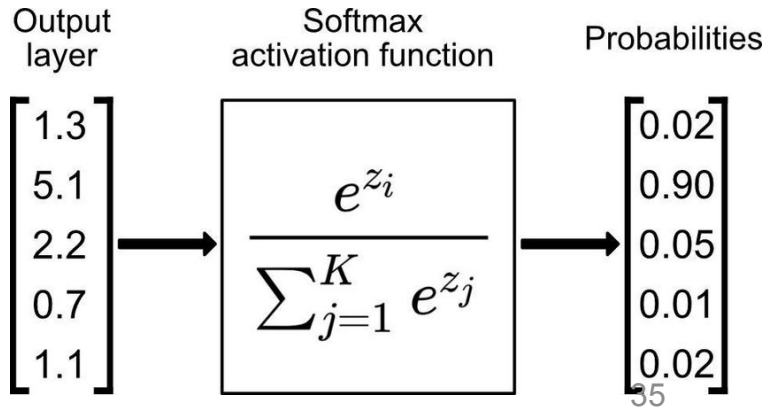
Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

- Translates to predicting the probability of y taking a value given x
- We first need to represent the labels in one hot encoding
- We then design the network output to be between 0 and 1 using a softmax activation on the last layer

source



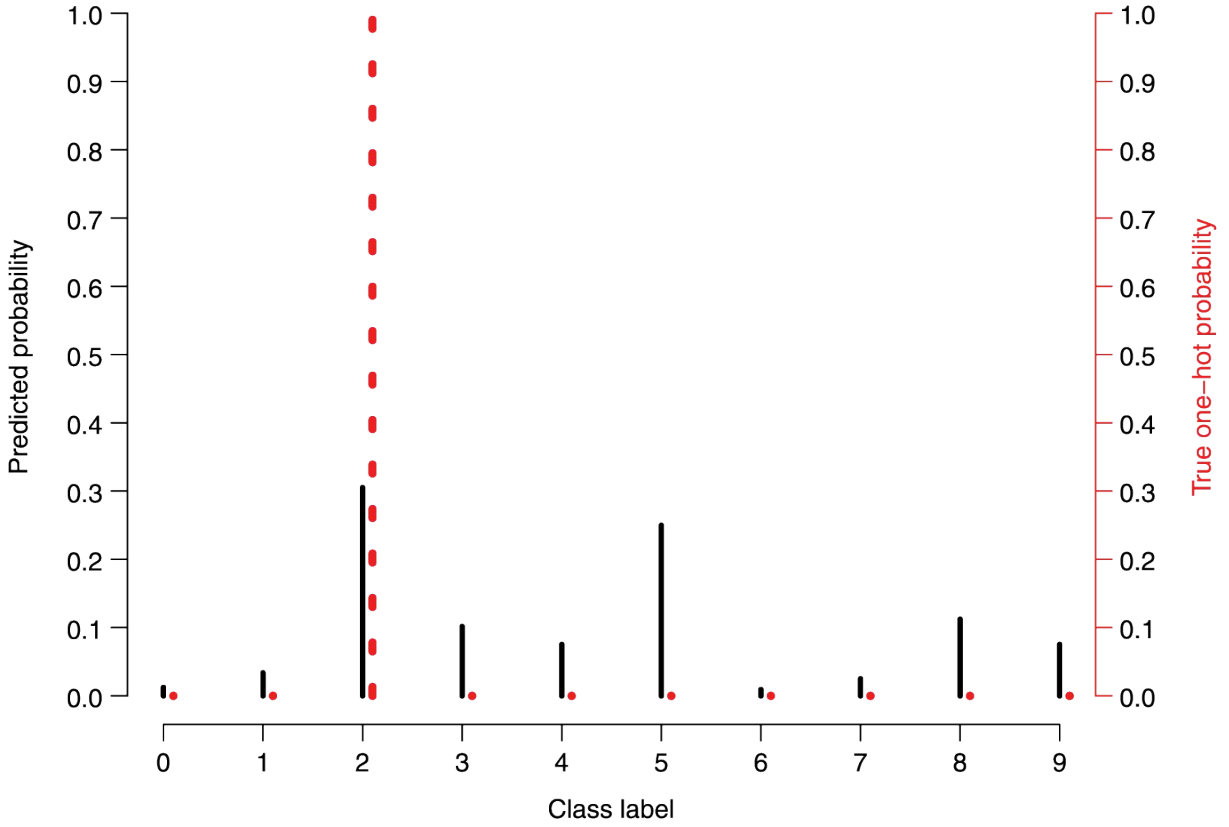
- Logistic regression does this in the binary case using the sigmoid activation, softmax allow for multiclass case

Cross enthropy loss

- Generalized case of the binary loss

- $CE = -\frac{1}{n}\sum_{i=1}^{n} y^i * \log\,(y'^i)$

- Derived from the max likelihood approach :
  - $L = \prod_{c=1}^{C} y^i y'^i$
  - $LL = \sum_{c=1}^{C} y^i \log\,(y'^i)$
  - $NLL = -\sum_{c=1}^{C} y^i \log\,(y'^i)$

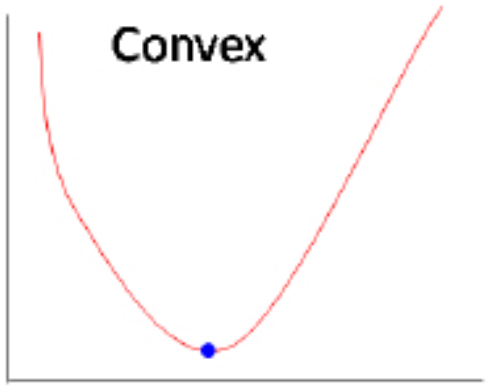


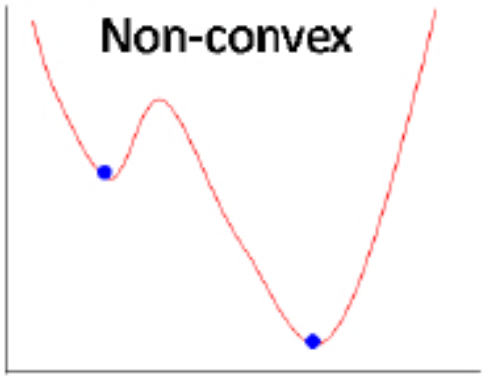
Loss optimization

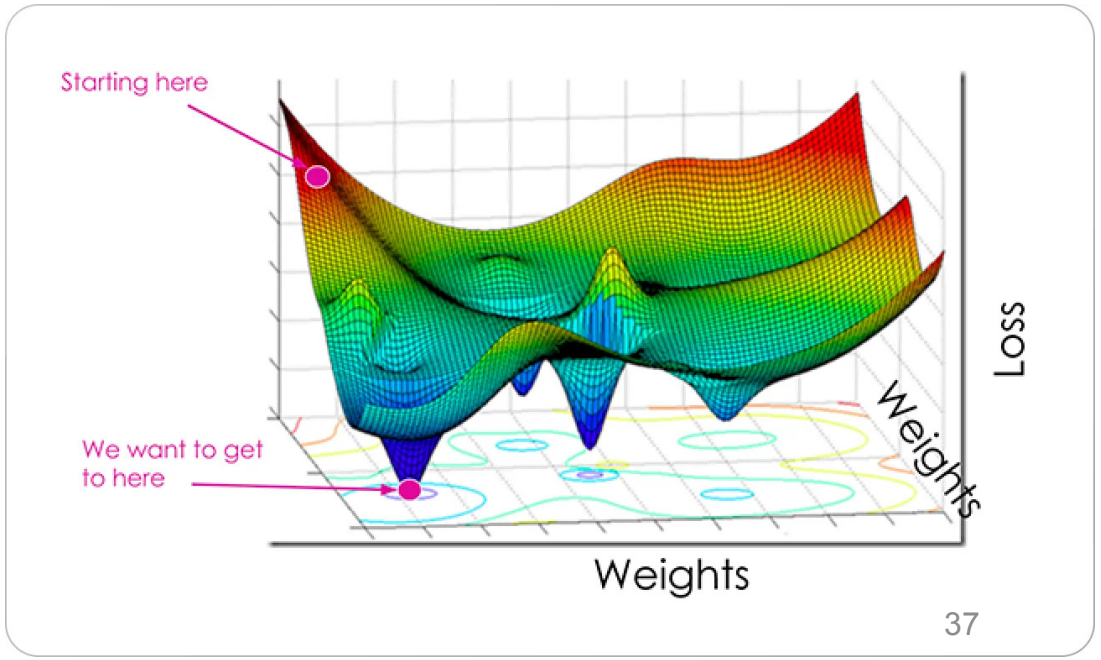
- We want to find the parameters of our model to minimize the loss :

$$W^* = \arg\min_W \left(\frac{1}{n}\sum_{i=1}^{n} \mathscr{L}\left(f\left(x^i;W\right),y^i\right)\right)$$

https://www.v7labs.com/blog/cross-entropy-loss-guide

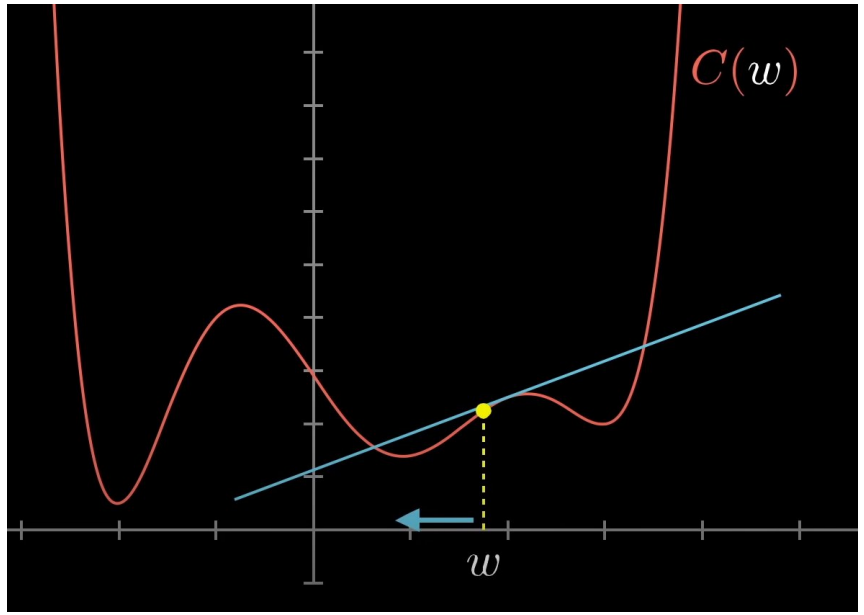






Gradient descent

- We are able to compute the gradient of the loss function with respect to its parmaeters : $\frac{d\mathcal{L}}{dw} = \nabla\mathcal{L}(w)$
- By updating its parameters following the gradient (slope in 2D) we can get closer to the minima of the function



$$w_{t+1} = w_t - \nabla\mathcal{L}(w_t)$$

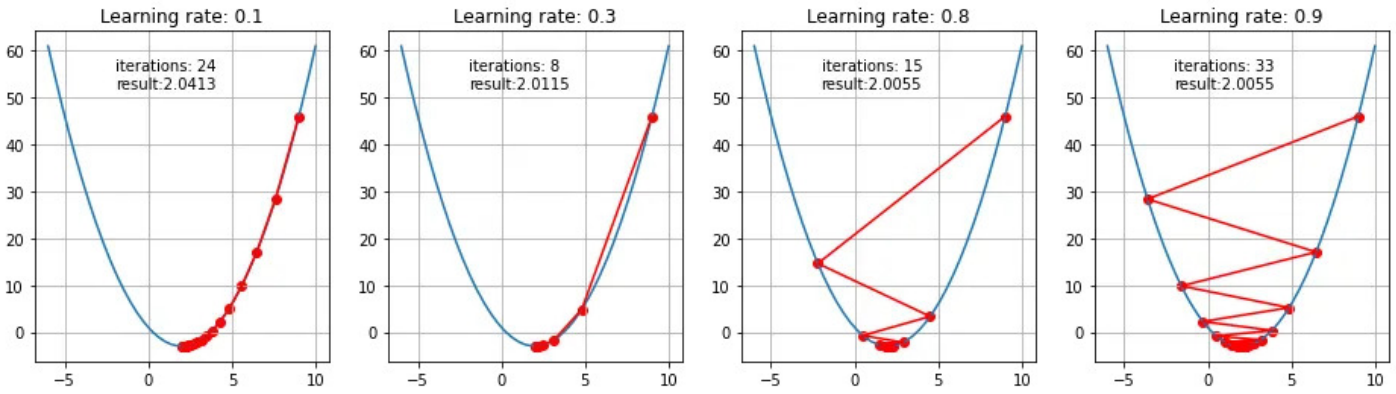
https://www.youtube.com/watch?v=IHZwWFHWa-w&t=1038s

https://www.youtube.com/watch?v=sDv4f4s2SB8

Learning rate

- We use a scaling parameter called learning rate to avoid the loss function overshooting the minimas
- This kind of parameter wich is set bu the user directly is called hyperparameters
- Different policies are often used for setting and possibly updating these parameters during the training

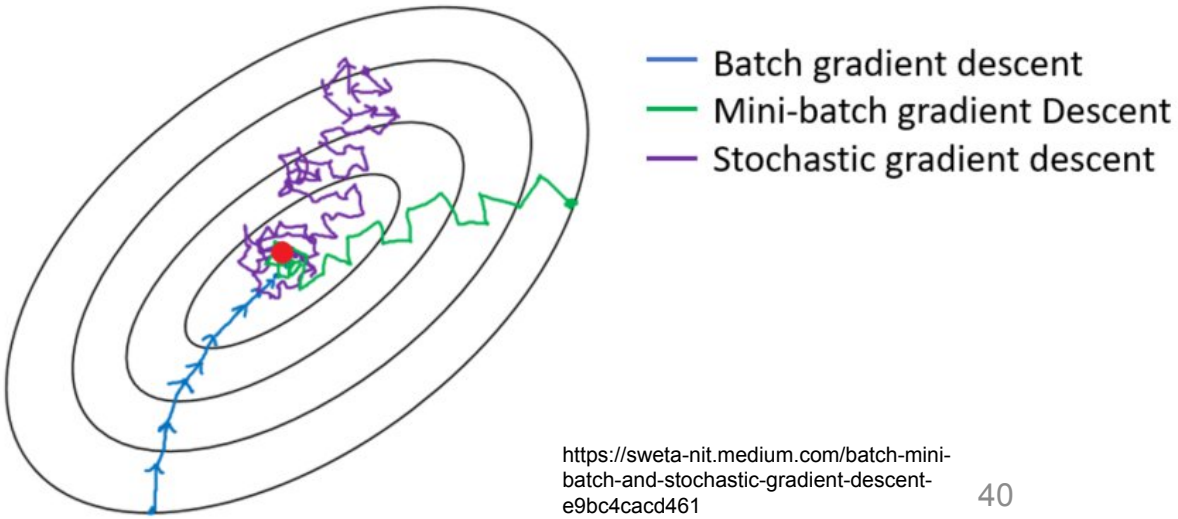
$$w_{t+1} = w_t - \alpha . \nabla \mathscr{L}(w_t)$$



https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21

Batches

- Ideally we want to compute all of the gradients at once, thats the batch case
- Based on the nature of the data, we may not be able to handle it all at once in memory
- Gradient descent algorithm may use batches instead, where only a subset of the data is processed each update
- An epoch corresponds to one loop where the model train over the whole data set



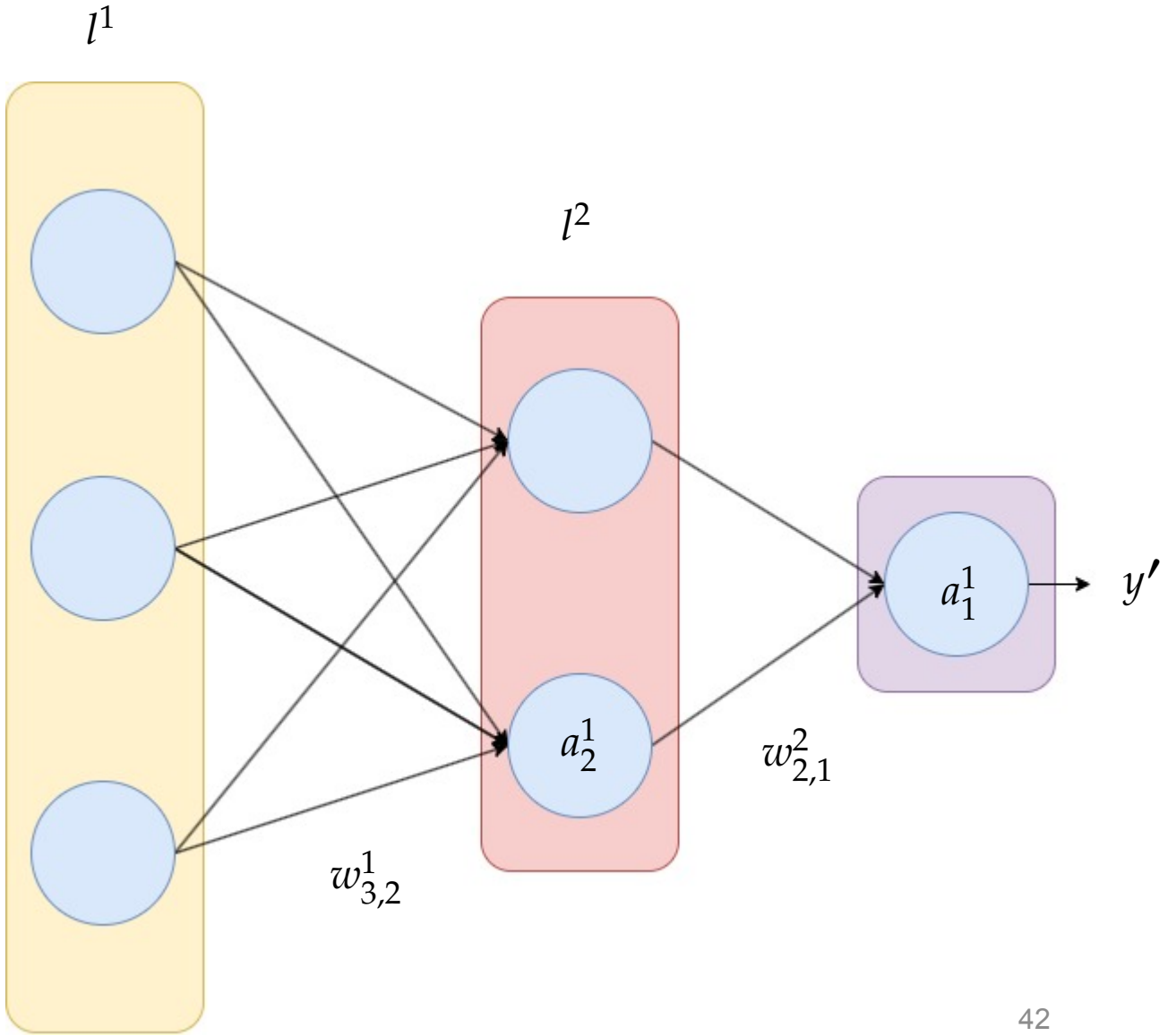
https://sweta-nit.medium.com/batch-mini-batch-and-stochastic-gradient-descent-e9bc4cacd461

Algorithm

- Initialize weights at randomn sample from $N(0,\sigma^2)$
- Main loop :
  - Forward/Infere the data through the model
  - Compute the loss and gradients
  - Update the weights
  - Check for convergence and stop if reached
- Return model

Backpropagation

- $z_r^i = b_r^i + \sum_k w_{k,r}^i * a_k^{i-1}$
- $a_r^i = \varphi(z_r^i)$
- for simplicity we set :
  φ = identity
- $\mathscr{L}(y', y) = MSE = \frac{1}{n}\sum_{i=1}^{n}(y^i - y'^i)^2$
- How do we compute $\frac{d\mathscr{L}}{dw_{2,1}^2}$?
- y' has the same formula as a but we use different notation for output

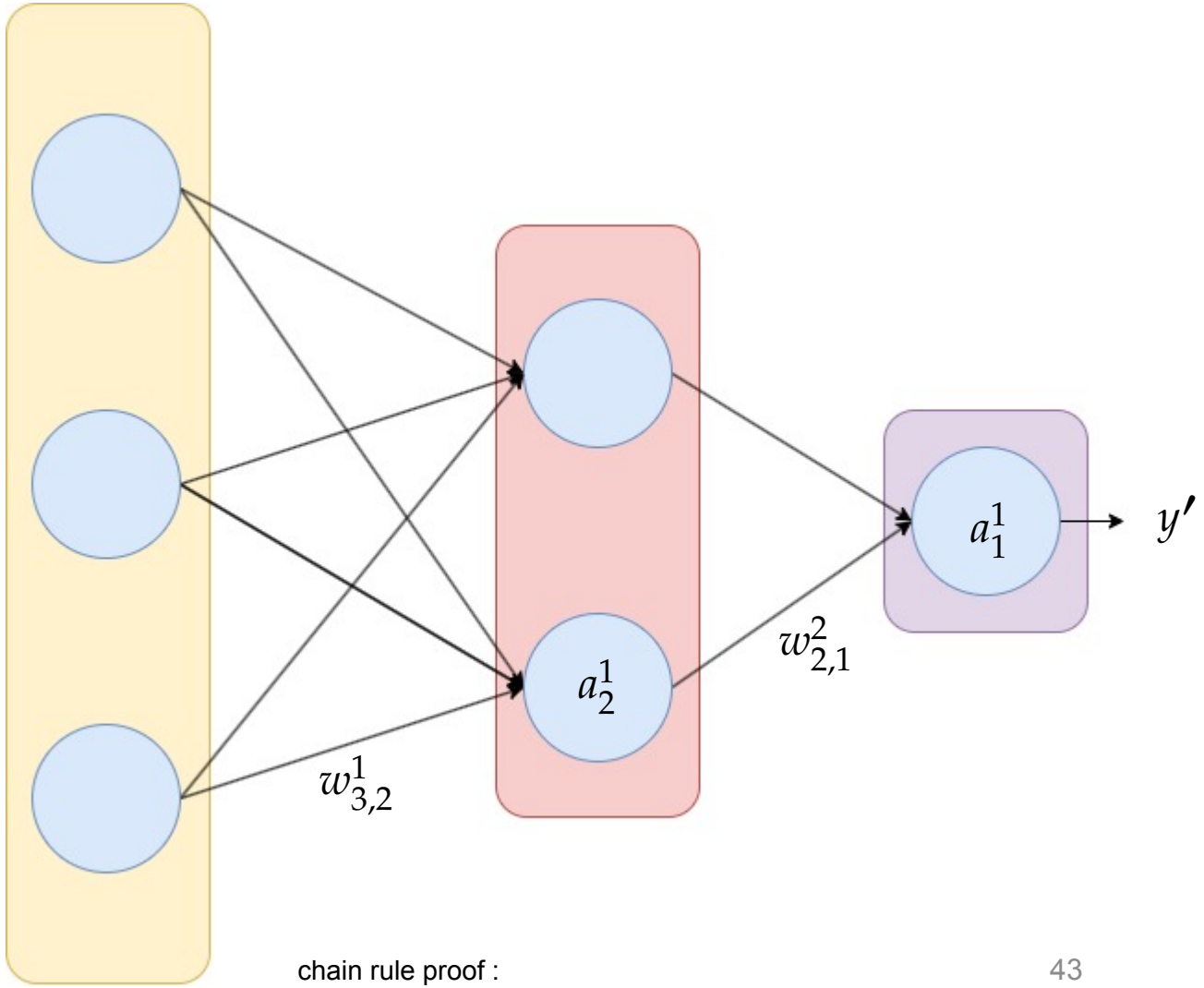


Chain rule

- $q = f \circ g \rightarrow q(x) = f(g(x))$
- $q' = (f \circ g)' = (f' \circ g) \cdot g'$
- $\frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dx}$
- $\frac{d\mathscr{L}}{dw^2_{2,1}} = \frac{d\mathscr{L}}{da^2} \cdot \frac{da^2}{dw^2_{2,1}} = \frac{d\mathscr{L}}{da^2} \cdot \frac{da^2}{dz^2} \cdot \frac{dz^2}{dw^2_{2,1}}$

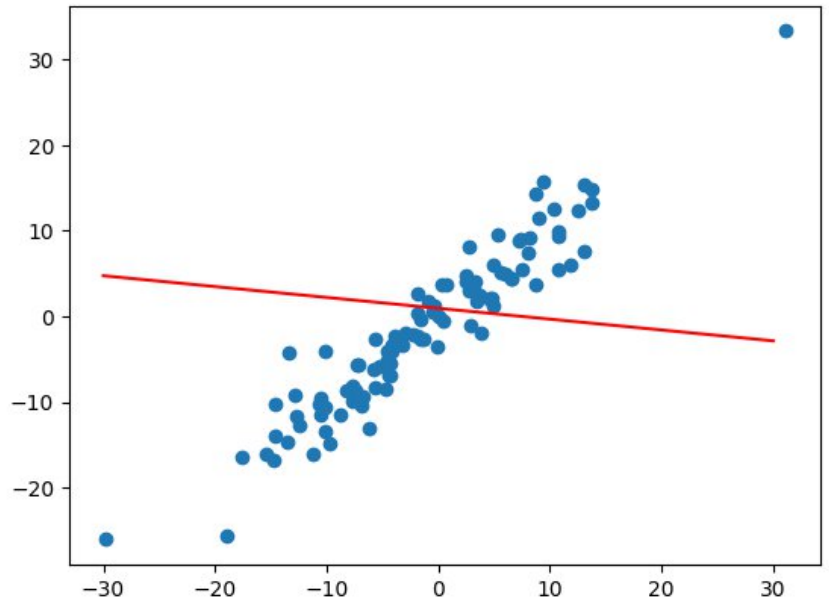
- $\frac{d\mathscr{L}}{dw^2} = \frac{d\mathscr{L}}{dy'} = \frac{1}{n}\sum_{i=1}^{n} 2(y^i - y'^i)$
- $\frac{da}{dz^2} = 1$
- $\frac{dz^2}{dw^2_{2,1}} = w_2{}^1$

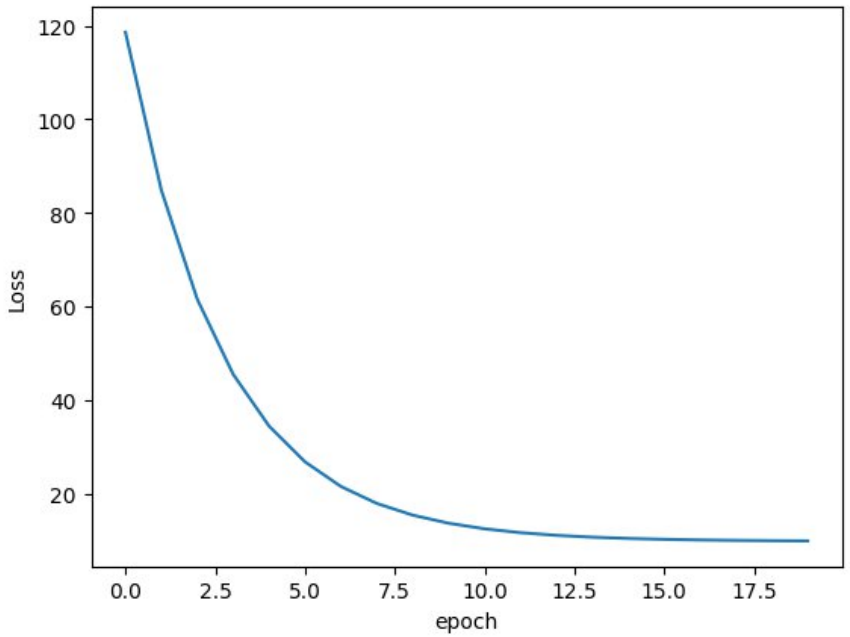
$$\frac{d\mathscr{L}}{dw^2_{k,1}} = \frac{1}{n}\sum_{i=1}^{n} 2(y'^i - y^i) * a^1_k$$

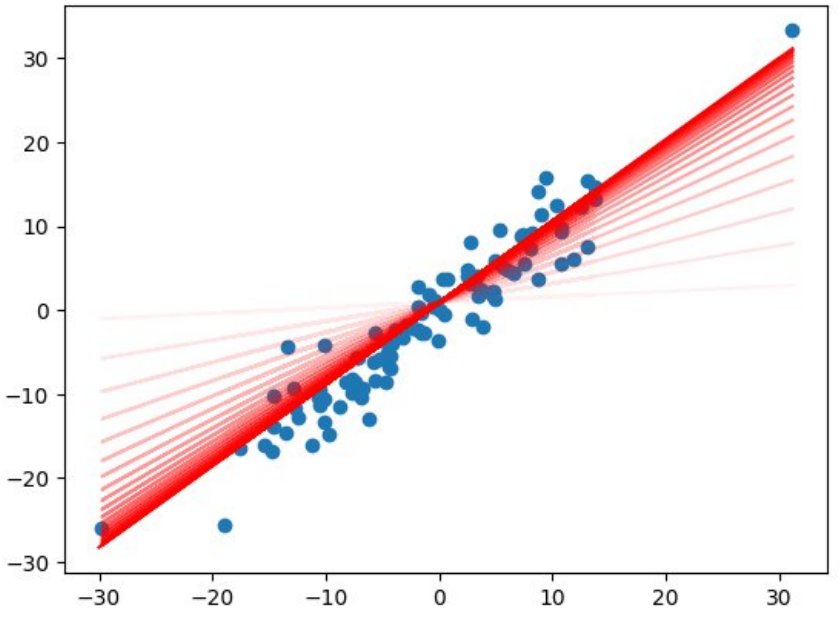


chain rule proof :
https://www.youtube.com/watch?v=m0LZX19DyyI&t=269s

Linear Regression Training

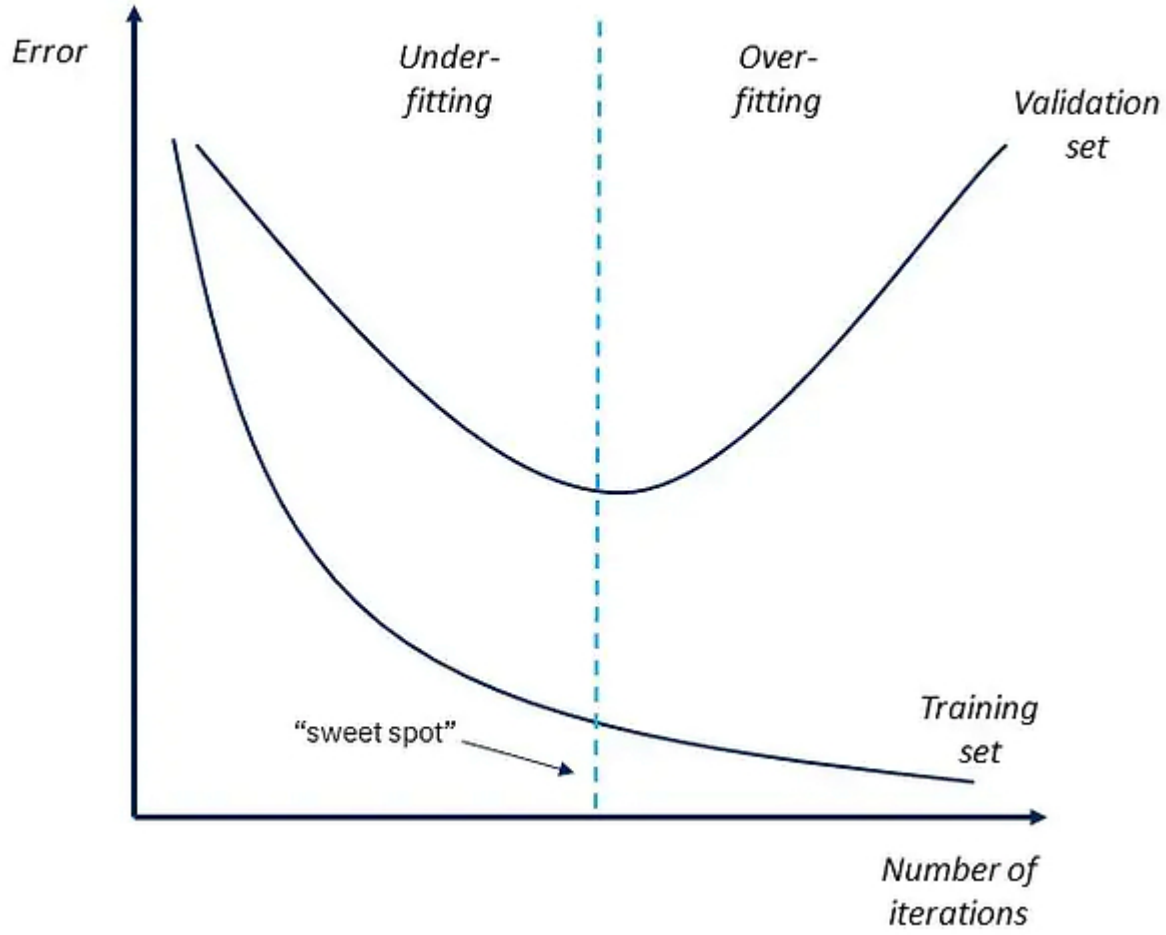






Regularization

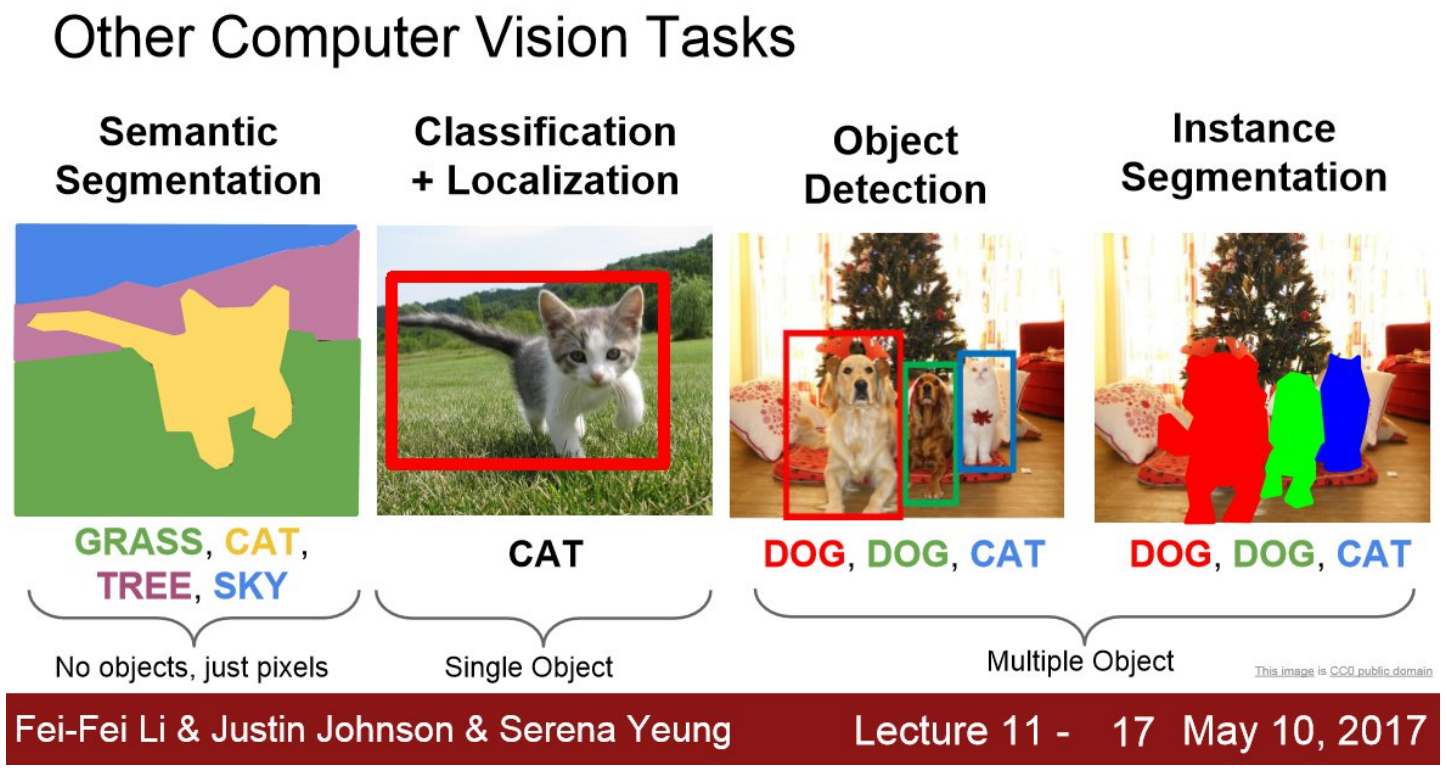
- Underfitting a model is easy to spot, overfitting might require you to make use of the validation set
- May be caused by a model too complex, a learning rate too big or too small, not enough data...
- Set hyperparameters to try to mitigate this effect
- Use of dropouts to reduce complexity



Deep learning application to images

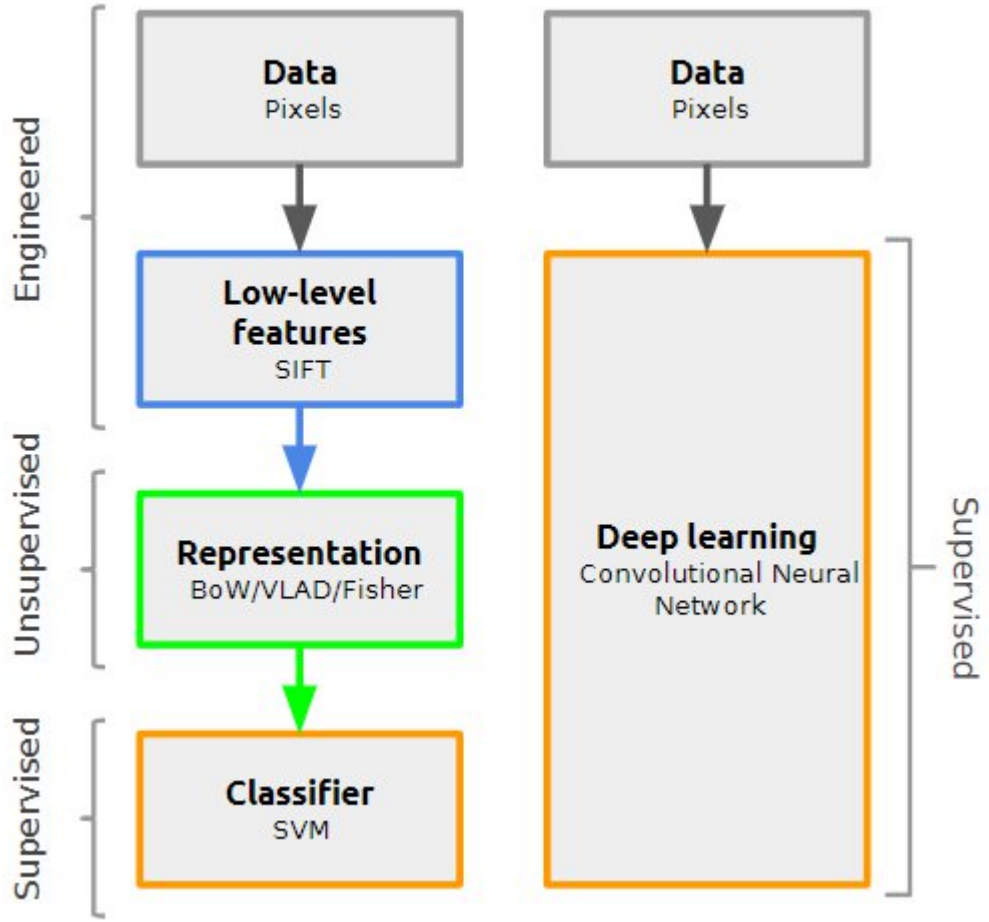
Computer Vision tasks

- Classification
- Segmentation and instance segmentation
- Object detection
- Image Generation
- Style transfer

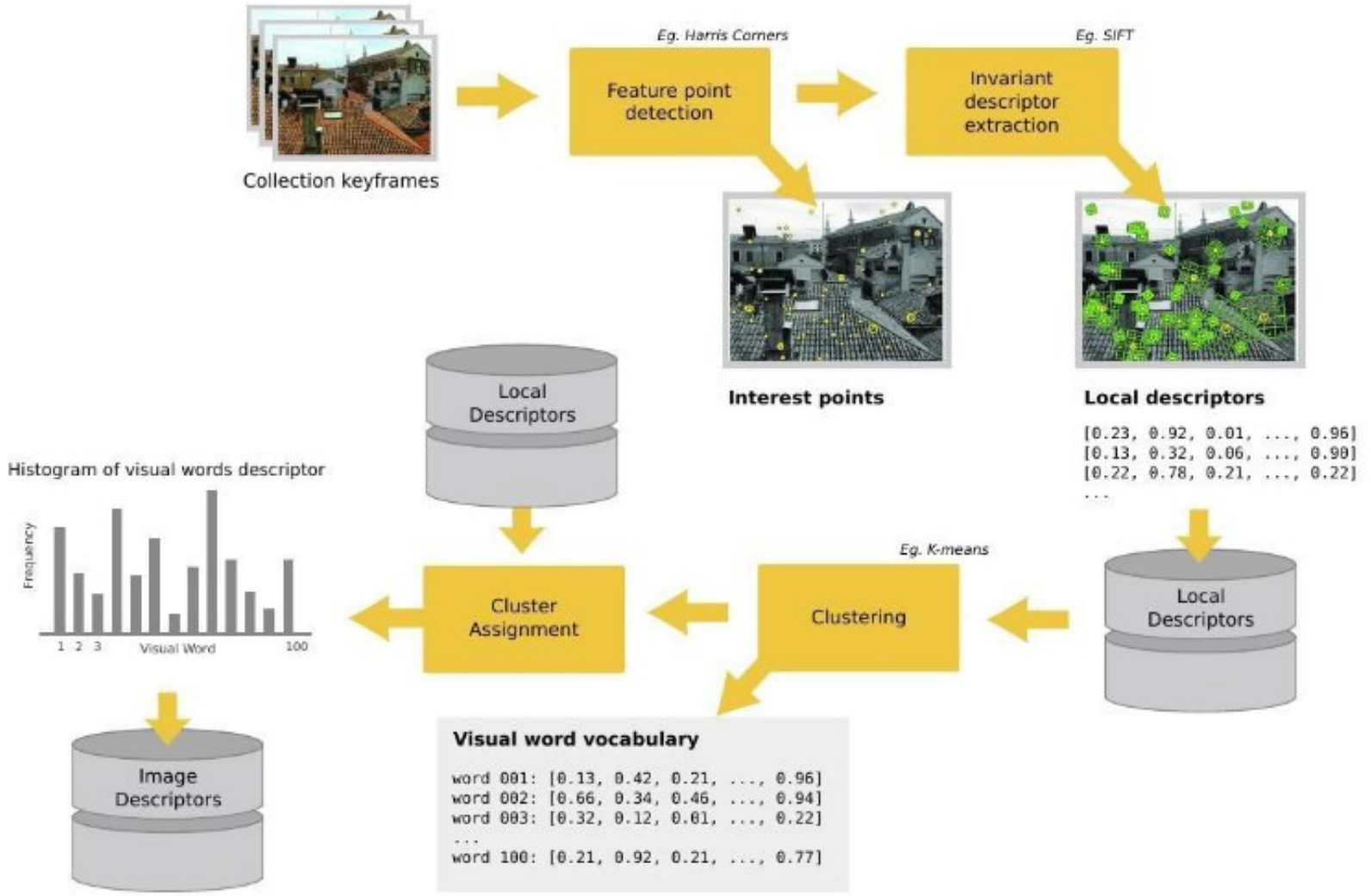


Shallow learning vs deep learning

- Old style machine learning:
  - Engineer features (by some unspecified method)
  - Create a representation (descriptor)
  - Train shallow classifier on representation
- Example:
  - SIFT features (engineered)
  - BoW representation (engineered + unsupervised learning)
  - SVM classifier (convex optimization)
- Deep learning
  - Learn layers of features, representation, and classifier in one go based on the data alone
  - Primary methodology: deep neural networks (non-convex)



Example : feature engineering pipeline

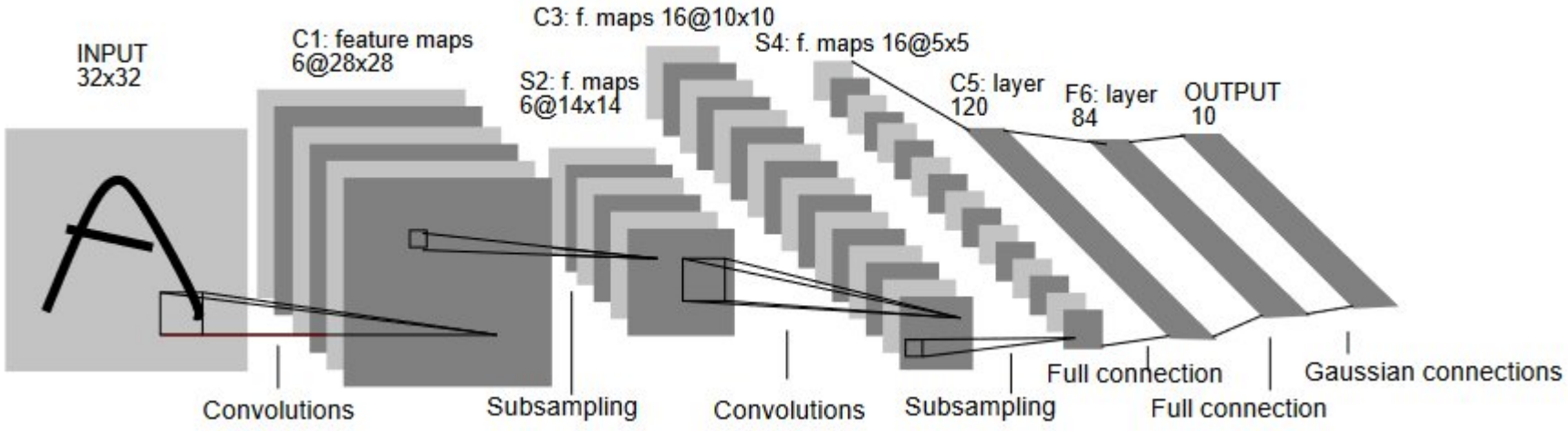


Growth of computer vision

- Has been the driving force of deep leanring for the last decade
- Multiple factors behind :
  - Higher computing power
  - Development of algorithms and theory of deep learning
  - Data availability

Convolution

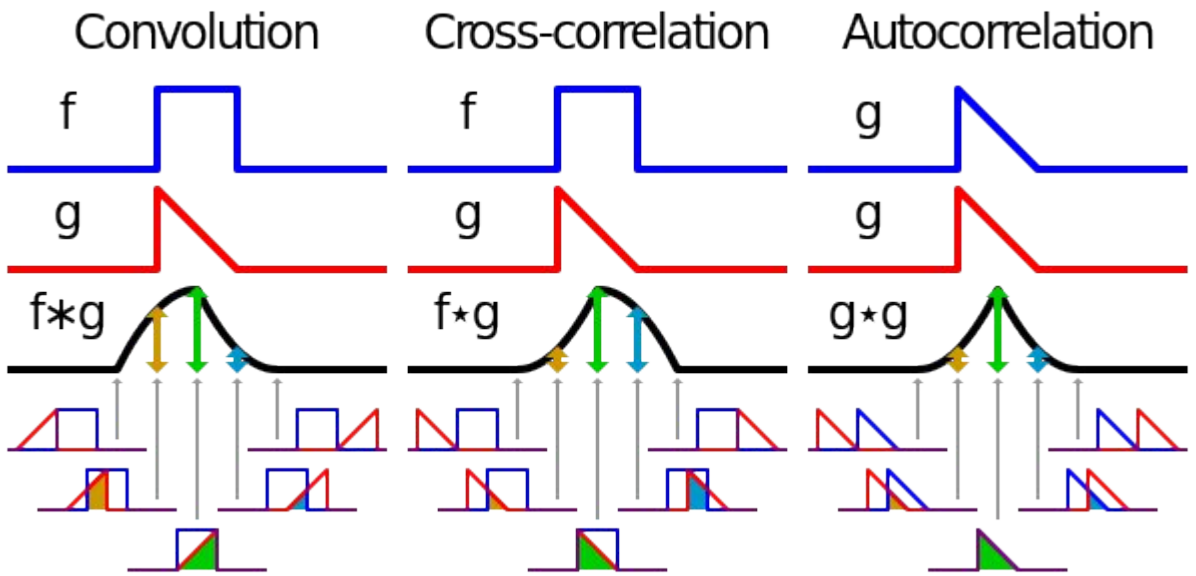
- Lecun 1990, "Gradient-Based Learning Applied to Document Recognition"
- Introduced convolution to neural networks, and the mnist dataset
- Allowed for switch from engineered features to learned features



Convolution (continuous)

$$(f * g)(x) = \int_{-\infty}^{+\infty} f(x-t)g(t)\,\mathrm{d}t = \int_{-\infty}^{+\infty} f(t)g(x-t)\,\mathrm{d}t$$

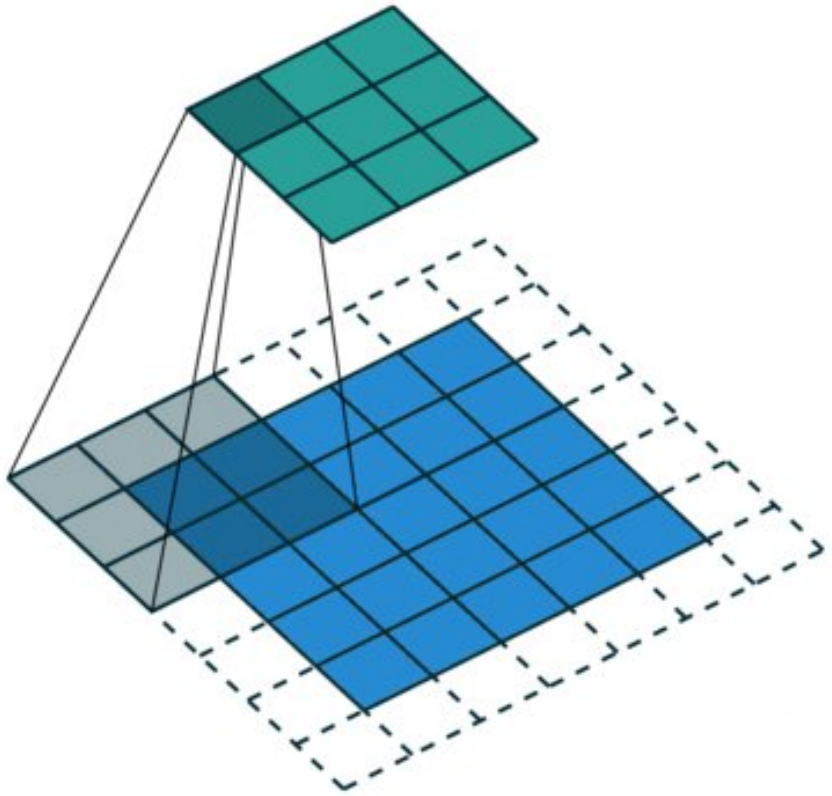
- Show the effect of a function on the other
- Used in probability, statistics, signal processing, computer vision ...
- Interesting properties with the fourier transform
- Accumulate every interaction of one signal with another



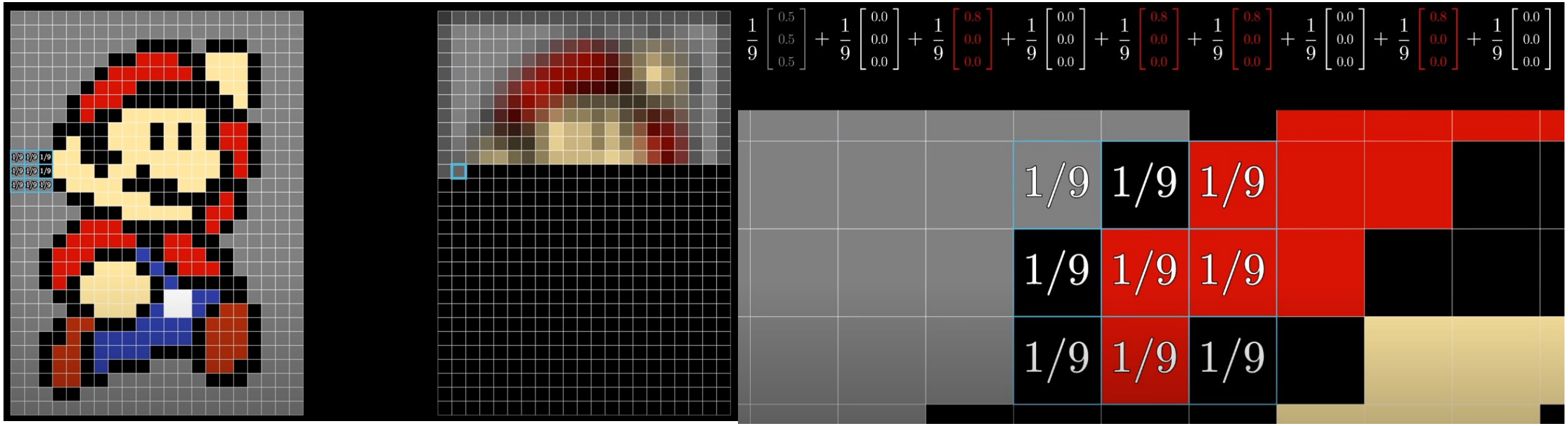
Convolution (discrete)

$$(f * g)[n] = \sum_{m=-M}^{M} f[n-m]g[m]$$

- Basic operation in image processing
- The second signal (g) is named a kernel, and the operation is also known as filtering an image
- In the discrete case, the output shape may be smaller than the input and may require padding based on the kernel size



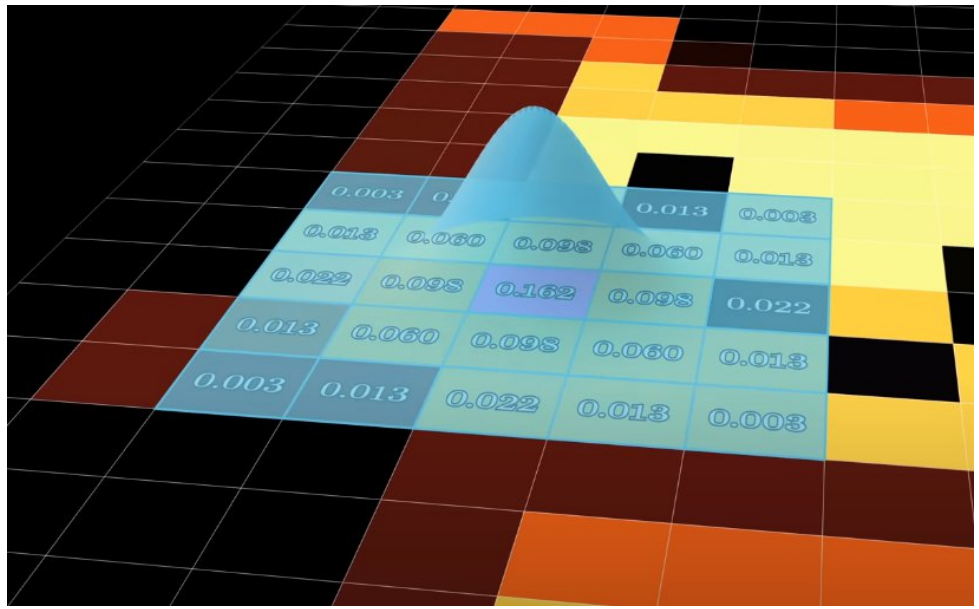
Example : Hand tuned Kernel



- Kernels sampled from a uniform distribution result in a blur of the image
- Compute the average on the kernel size of the image

https://www.youtube.com/watch?v=KuXjwB4LzSA

Example : Hand tuned Kernel



- More elegant blur is possible such as the gaussian blur sampled from a gaussian distribution
- Sobel operator widely used in edge detection algorithms (canny edge)

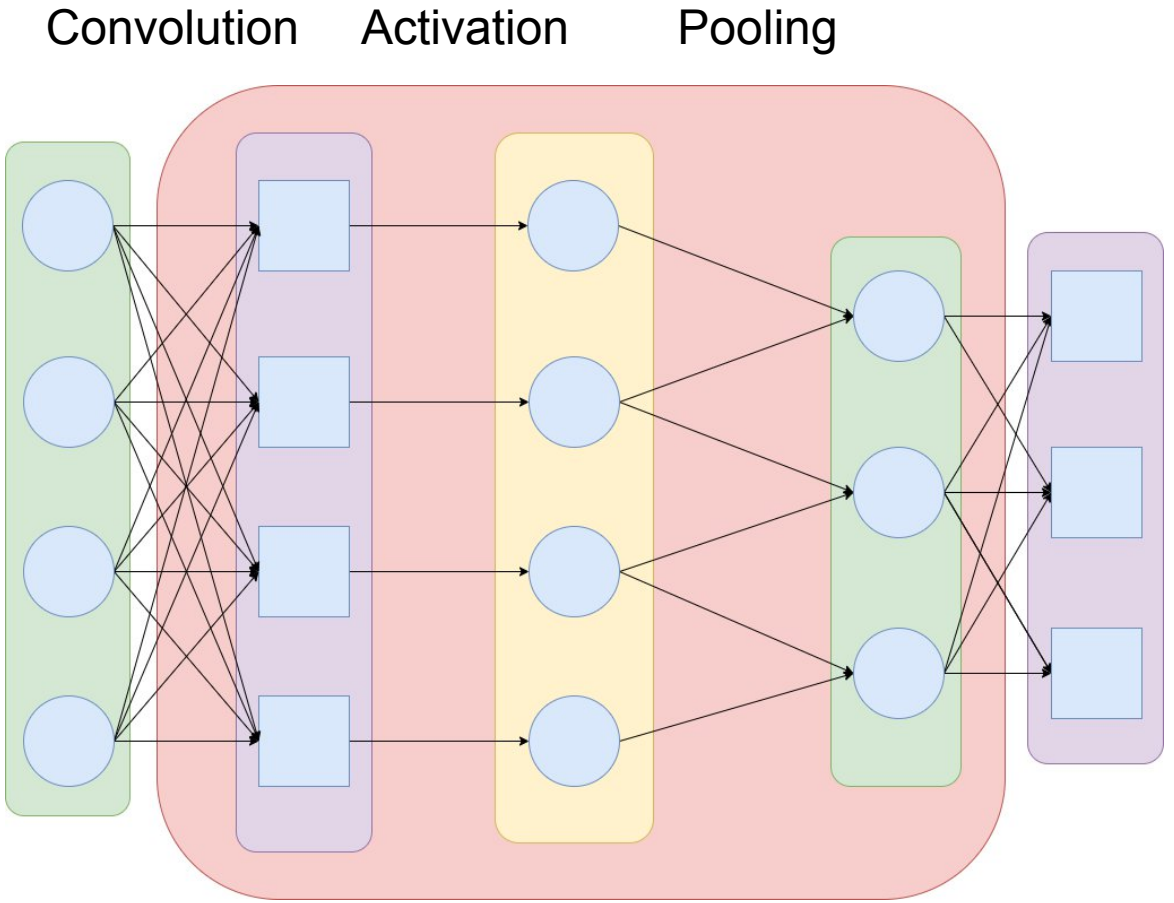
$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \mathbf{A} \quad \text{et} \quad \mathbf{G_y} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * \mathbf{A}$$

$$\mathbf{G} = \sqrt{\mathbf{G_x}^2 + \mathbf{G_y}^2}$$

Convolution layer

- Just like dense layer in FCN, convolution layer represent one of the succesive blocks in CNN
- A typical convolution layer is composed of 3 operations :
  - The convolution itself
  - The activation
  - A pooling operation

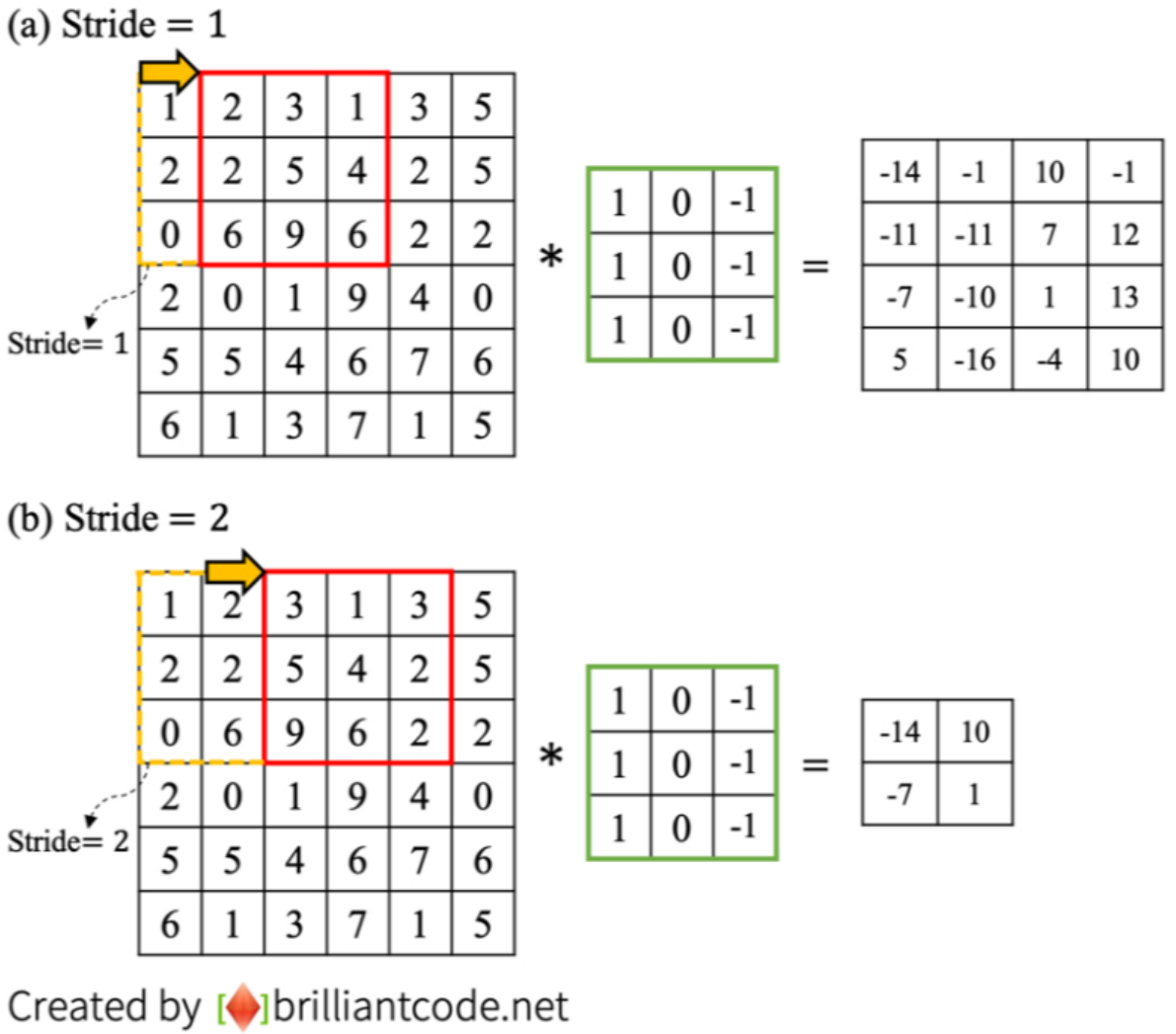
(These operation may also be referred astheir own layers)



$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(N_i, k)$$

Layer analysis : convolution

- Focused on capturing spatial structures, rather than the underlying trend such as dense layers
- The convolution modifies the shape of the forwarded data, depending on its parameters



https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html#torch.nn.Conv2d

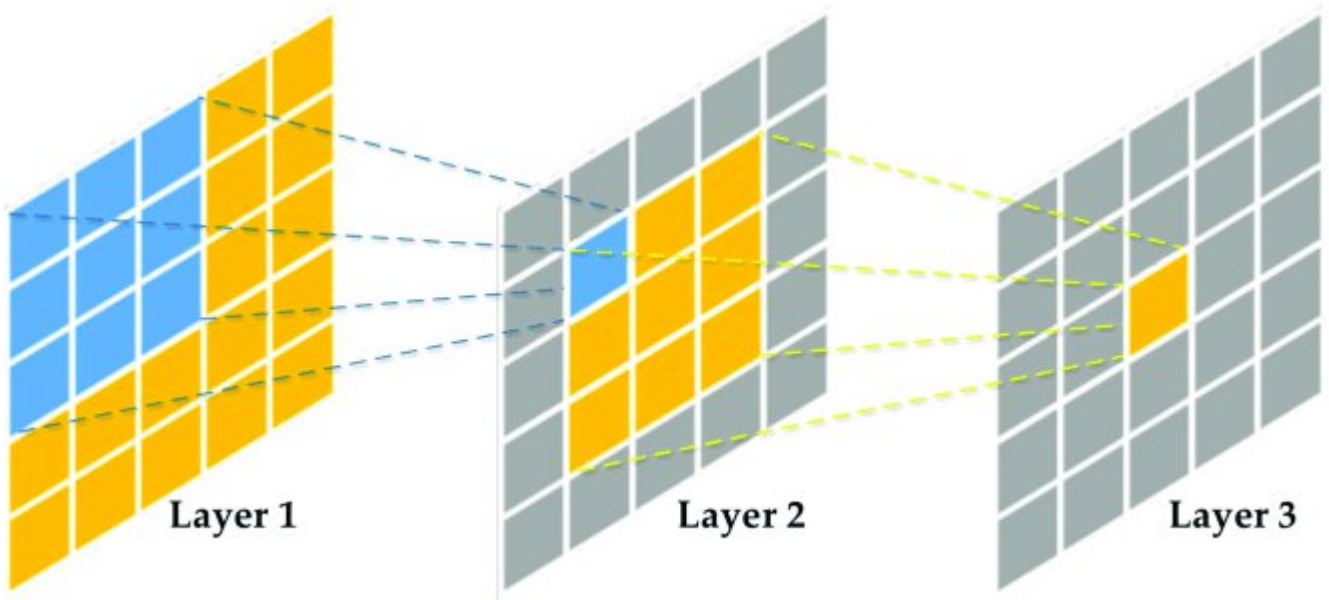
- Input: $(N, C_{in}, H_{in}, W_{in})$ or (C_{in}, H_{in}, W_{in})
- Output: $(N, C_{out}, H_{out}, W_{out})$ or $(C_{out}, H_{out}, W_{out})$, where

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

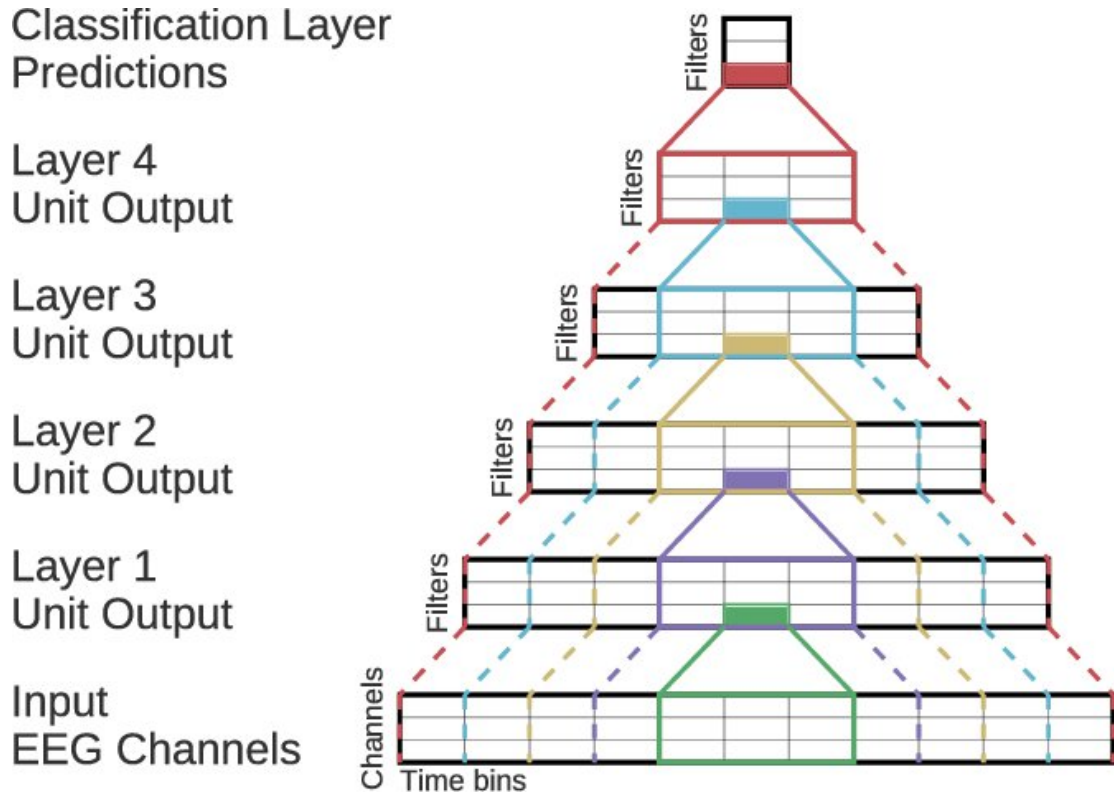
$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

Receptive Field

- Impact of inputs on a layer of a given depth
- We want it to be as large as possible



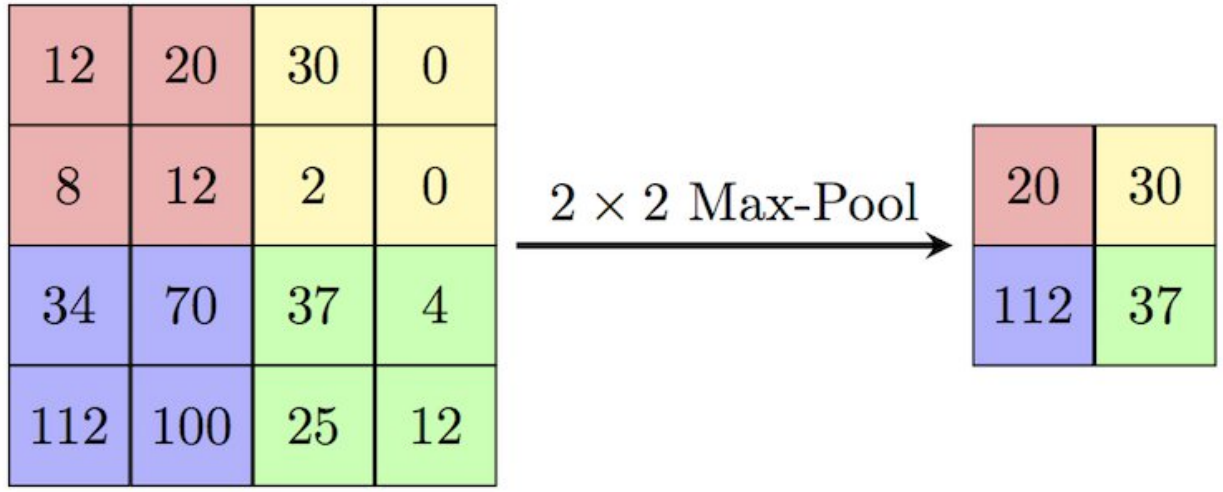
https://www.researchgate.net/figure/Schematic-diagram-of-the-receptive-field-in-CNNs_fig1_353545214



https://www.researchgate.net/figure/ConvNet-Receptive-Fields-Schema-Showing-the-outputs-inputs-and-receptive-fields-of-one_fig4_318965745

Layer analysis : Pooling

- Because of how convolution works, feature maps are sensitive to the location of the features in the input.
- We use pooling layer to try to mitigate that effect :
  - Max pooling
  - Average pooling
  - Stochastic pooling
  - Fractional pooling



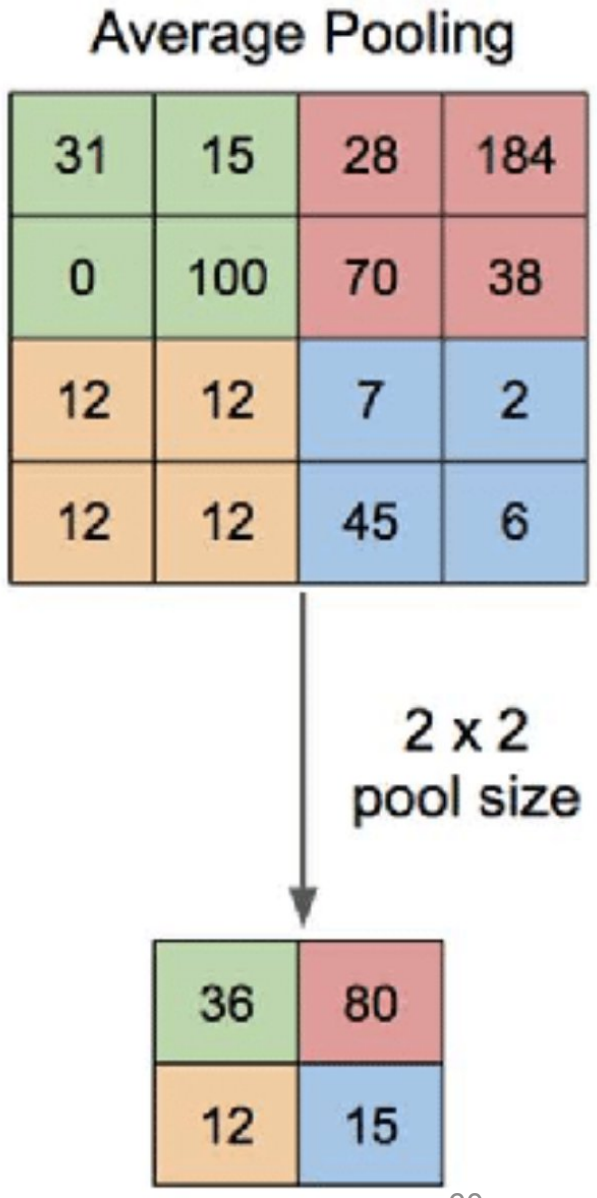
- Input: $(N, C_{in}, H_{in}, W_{in})$ or (C_{in}, H_{in}, W_{in})
- Output: $(N, C_{out}, H_{out}, W_{out})$ or $(C_{out}, H_{out}, W_{out})$, where

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

Average Pooling

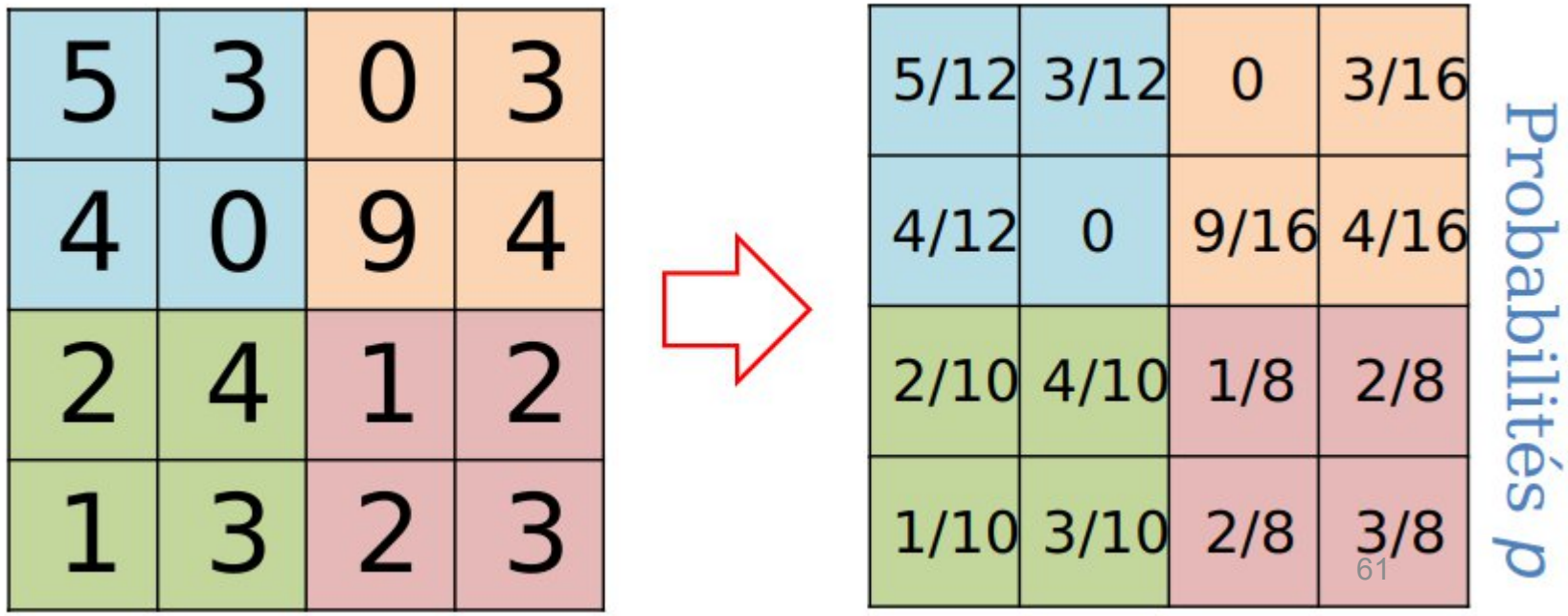
- Unlike max pooling which extract the most influencial features, average pooling takes everything into account
- Has more of a cleaning effect on the features
- All gradients are propagated to the previous layer



Stochastic pooling

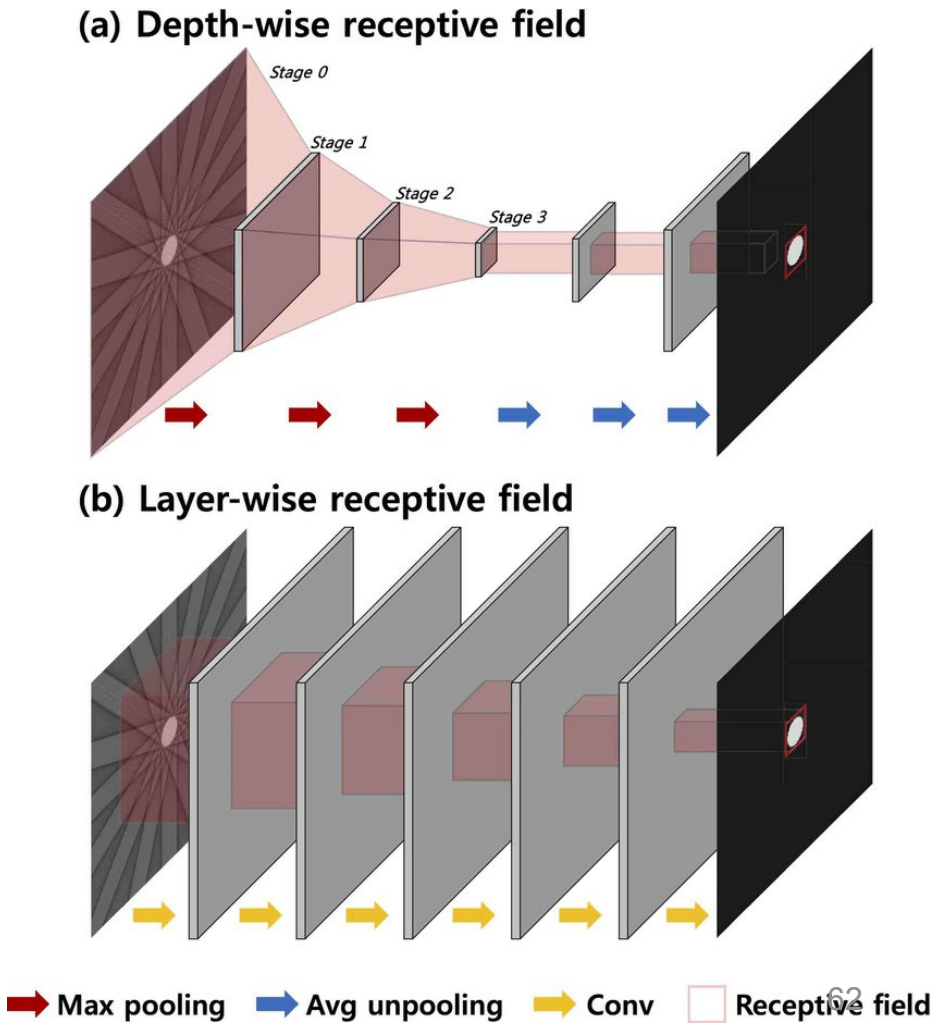
- Use features values as probabilities for pooling
- Acts as a kind of regularization
- At test and prediction time, a probabilistic average is used instead

https://arxiv.org/pdf/1301.3557.pdf



Why introduce pooling

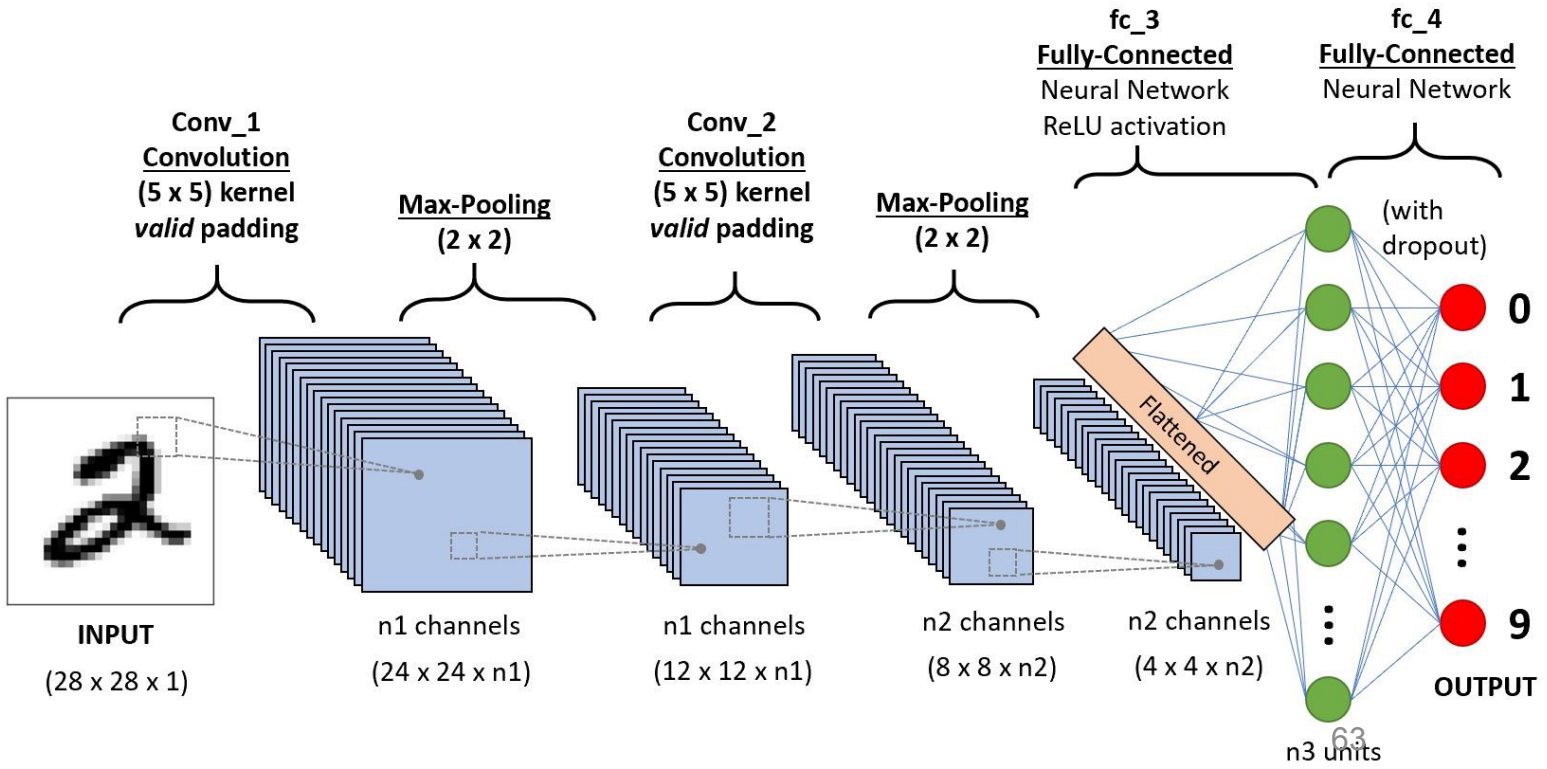
- Increase receptive field
- Summarize presence of features
- Reduce parameters number
- Allow for some invariance to localization of feature in input image



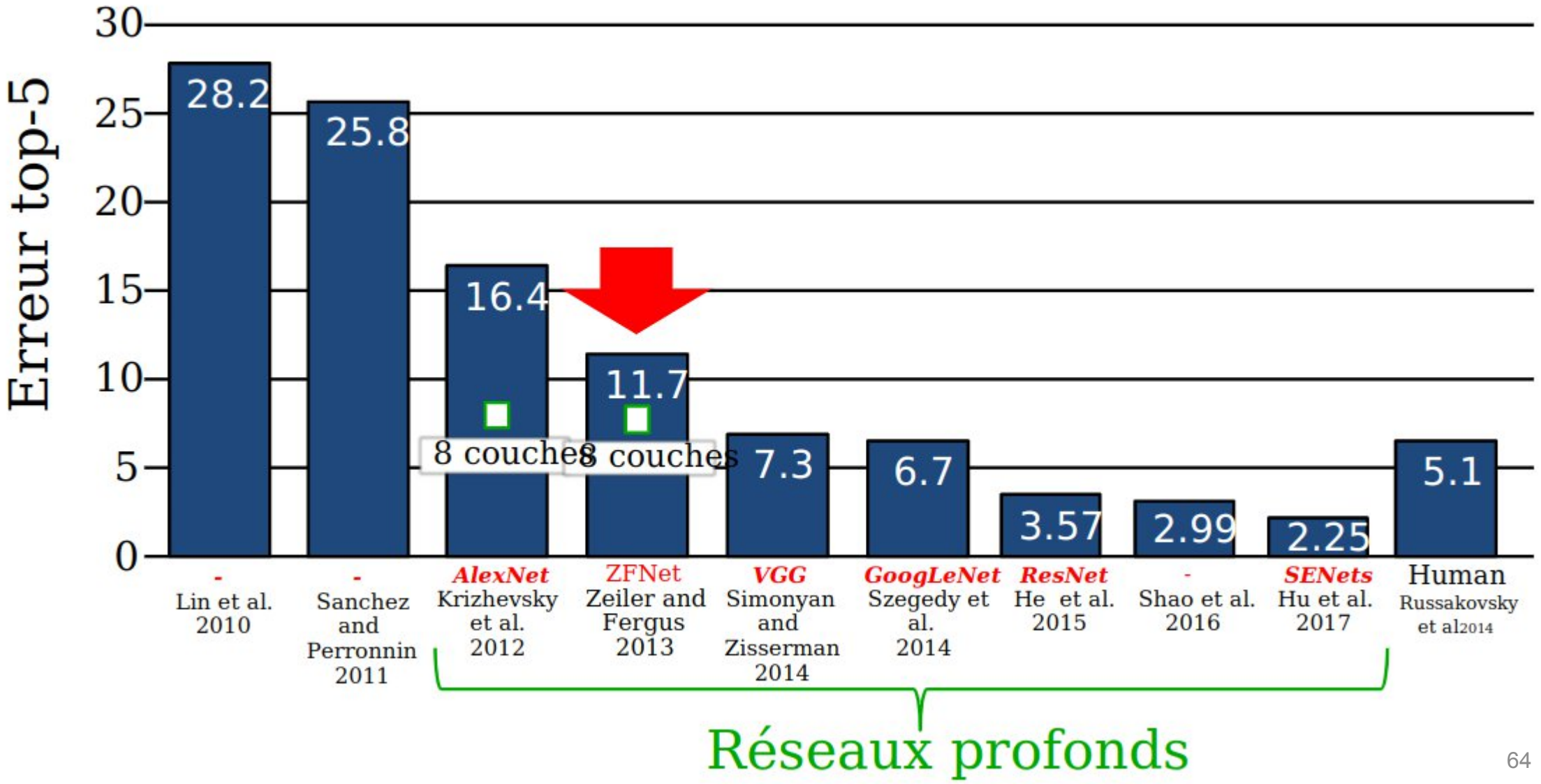
Puting it together

- Blocks building approach
- The lowest level features , known as latent representation of the input, are flattened and forwaded in dense layers for classification
- We can then use cross-enthropy to compute loss

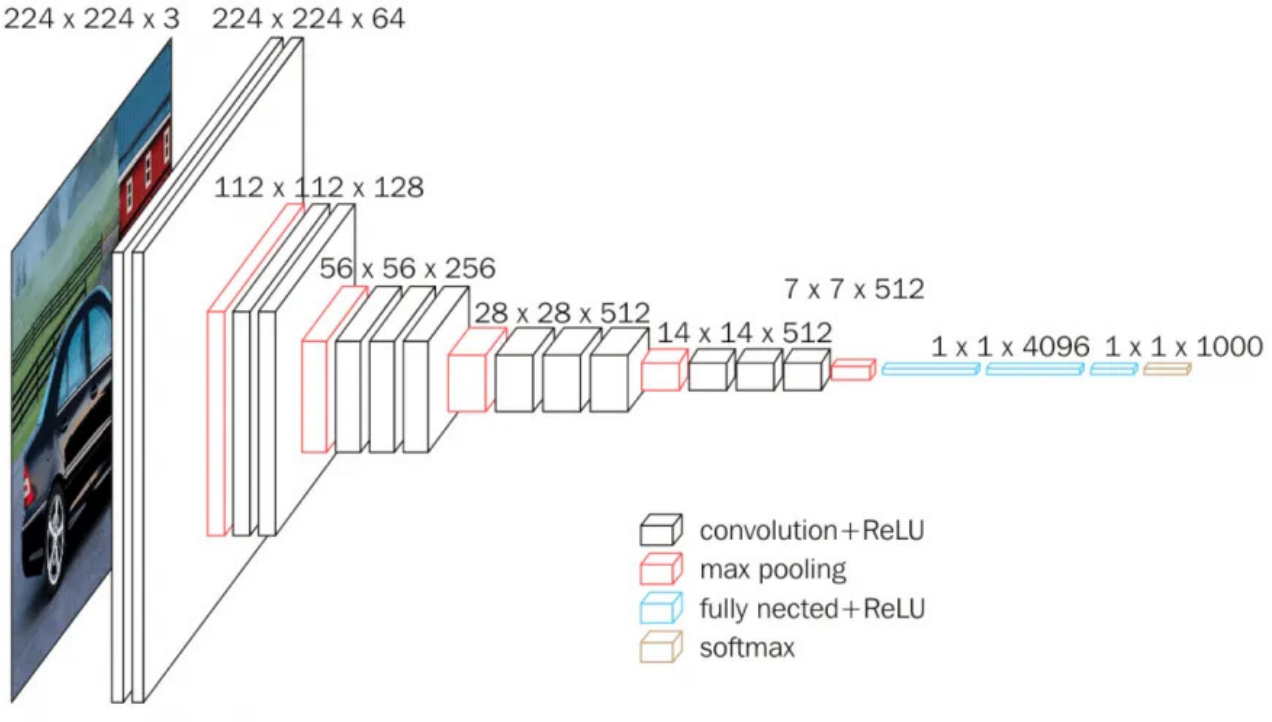
- Makes use of the dimensionality reduction to increase depth



ImageNet over the years



VGG16 architecture

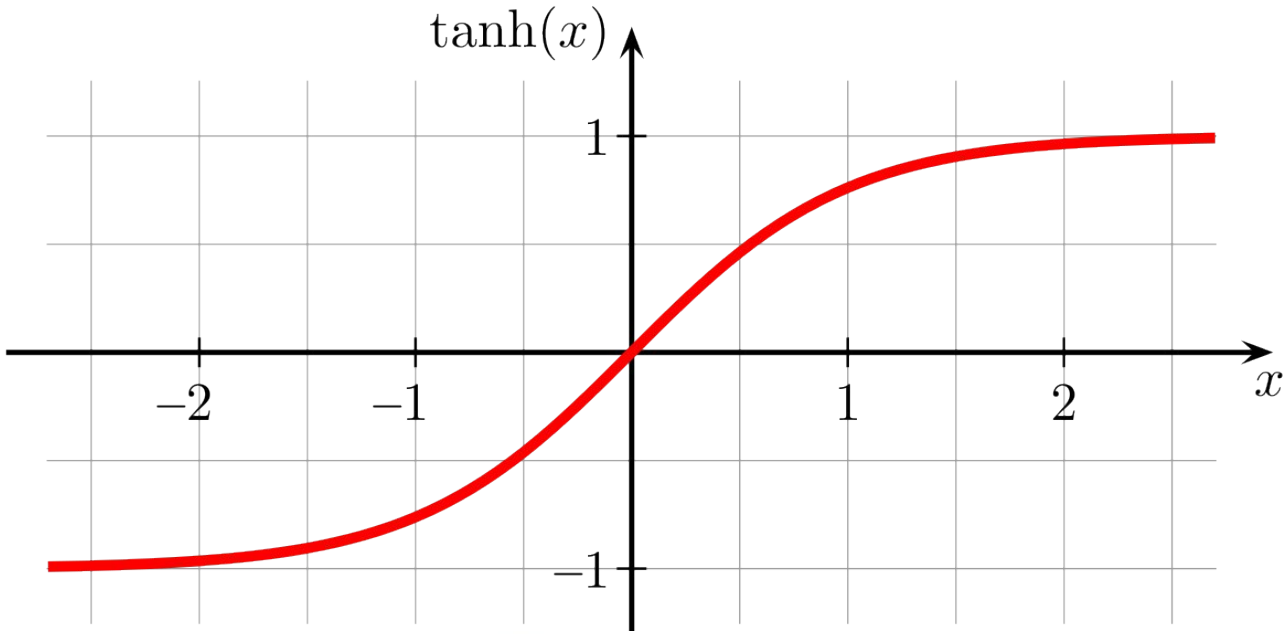


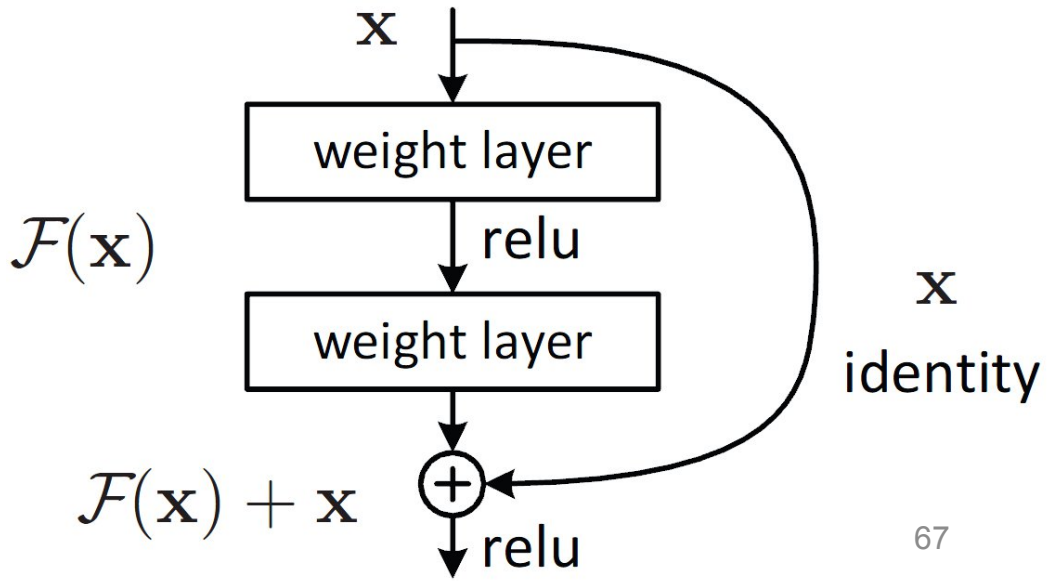
VGG16 - Structural Details													
#	Input Image			output			Layer	Stride	Kernel		in	out	Param
1	224	224	3	224	224	64	conv3-64	1	3	3	3	64	1792
2	224	224	64	224	224	64	conv3064	1	3	3	64	64	36928
	224	224	64	112	112	64	maxpool	2	2	2	64	64	0
3	112	112	64	112	112	128	conv3-128	1	3	3	64	128	73856
4	112	112	128	112	112	128	conv3-128	1	3	3	128	128	147584
	112	112	128	56	56	128	maxpool	2	2	2	128	128	65664
5	56	56	128	56	56	256	conv3-256	1	3	3	128	256	295168
6	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080
7	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080
	56	56	256	28	28	256	maxpool	2	2	2	256	256	0
8	28	28	256	28	28	512	conv3-512	1	3	3	256	512	1180160
9	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808
10	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808
	28	28	512	14	14	512	maxpool	2	2	2	512	512	0
11	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
12	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
13	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
	14	14	512	7	7	512	maxpool	2	2	2	512	512	0
14	1	1	25088	1	1	4096	fc		1	1	25088	4096	102764544
15	1	1	4096	1	1	4096	fc		1	1	4096	4096	16781312
16	1	1	4096	1	1	1000	fc		1	1	4096	1000	4097000
Total													138,423,208

- In the early days, performance was mostly limited by hardware
- The « better » models heavely relied on building deeper and deeper models
- At some point, performance would decrease in deeper model, which led DL engineer to design a wide diversity of models

Vanishing gradient problem and residual

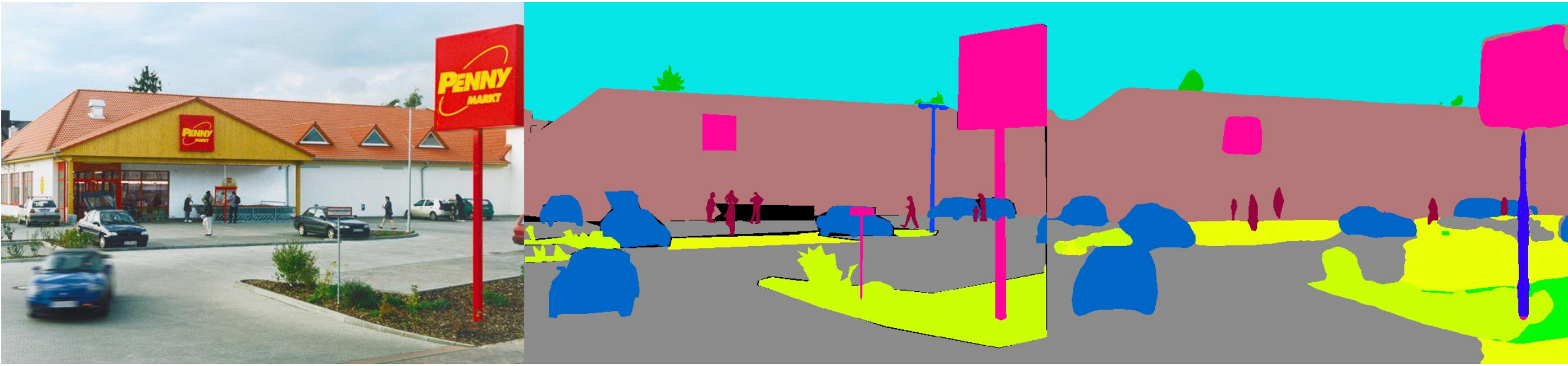
- Chain rule in very deep NN might result in an extremely small gradient
- For example : tanh derivatives are comprised between 0 and 1
- The residual module was then design to tackle this issue





Segmentation in computer vision

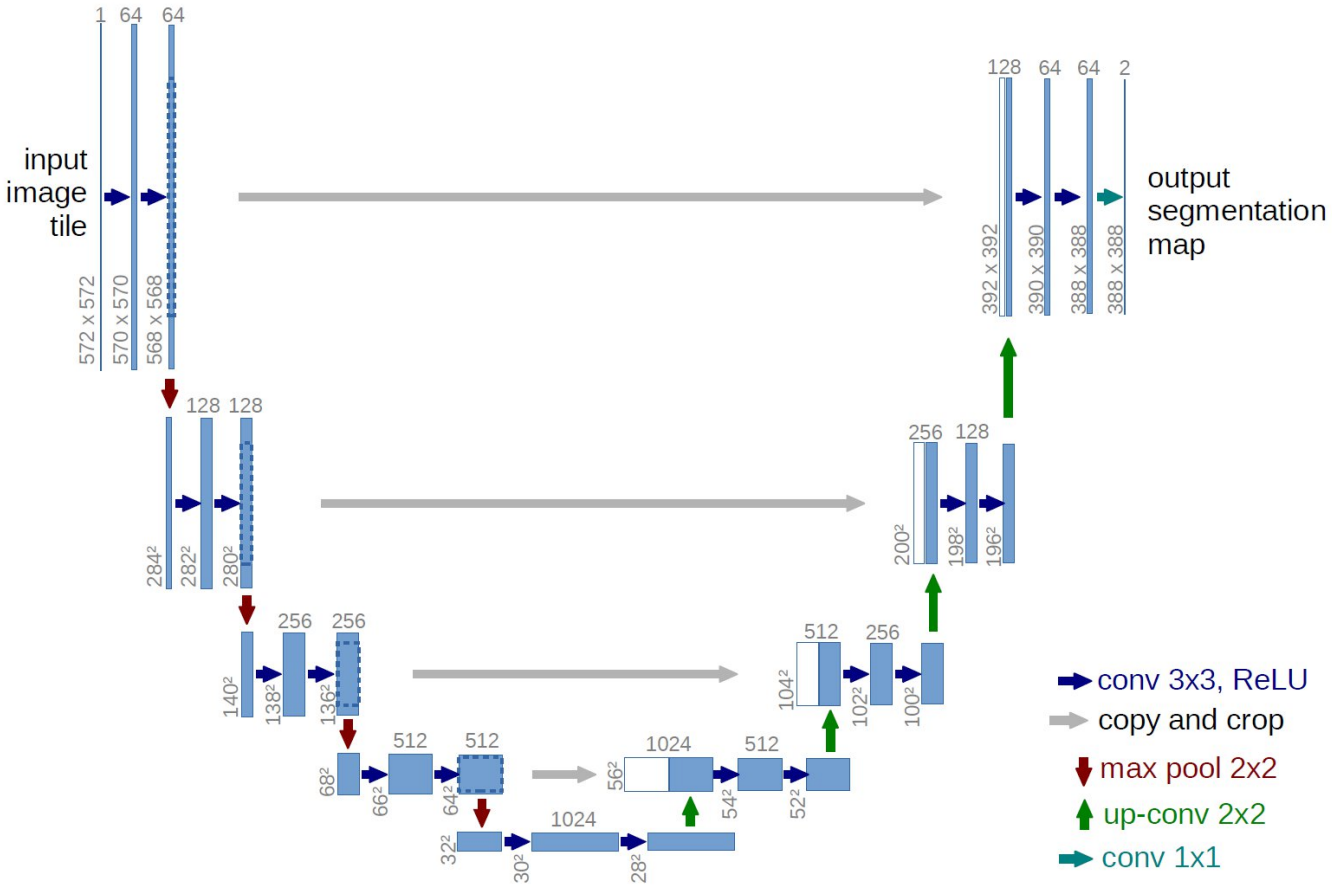
input ground truth segmented output



https://paperswithcode.com/paper/semantic-understanding-of-scenes-through-the

- As opposed to classification, the label dimension has the same shape as the input dimension
- This requires a different types of model to resolve this task, which makes it possible to use different kinds of losses
- When building and training a model, analysing the dimensions along the network is generally a good idea to avoid mistakes

Example : Unet architecture for segmentation



- Output has same shape as input :
  - for an input of shape $H * W * C$, C being the number of channels, the model return a shape $H * W * N$, with N classes
  - we are then able to use cross-enthropy- or other type of losses
  - example : Jaccar loss
- Skip connection are used which fill the same roles as residual connections
- There exists different techniques to upsample an image back to its original shape :
  - classical nearest value upsample
  - upsample with interpolation
  - deconvolution
  - ...

Jaccard Loss

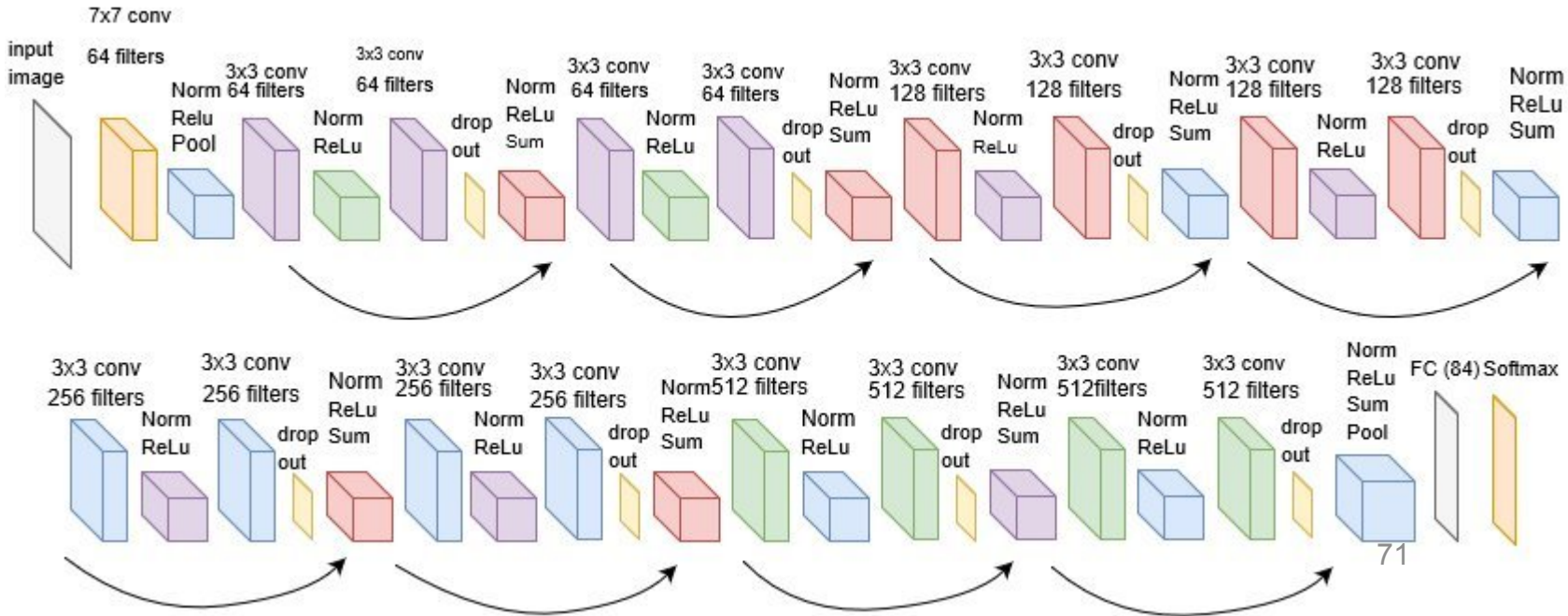
IOU : intersection over union : $\frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$

For minimization purposes, we use the Jaccar Distance :

$$J_d = 1 - \frac{|A \cap B|}{|A \cup B|} = 1 - \frac{y.y'}{y + y' - y.y'}$$

ResNet18 - Structural Details														
#	Input Image			output			Layer	Stride	Pad	Kernel		in	out	Param
1	227	227	3	112	112	64	conv1	2	1	7	7	3	64	9472
	112	112	64	56	56	64	maxpool	2	0.5	3	3	64	64	0
2	56	56	64	56	56	64	conv2-1	1	1	3	3	64	64	36928
3	56	56	64	56	56	64	conv2-2	1	1	3	3	64	64	36928
4	56	56	64	56	56	64	conv2-3	1	1	3	3	64	64	36928
5	56	56	64	56	56	64	conv2-4	1	1	3	3	64	64	36928
6	56	56	64	28	28	128	conv3-1	2	0.5	3	3	64	128	73856
7	28	28	128	28	28	128	conv3-2	1	1	3	3	128	128	147584
8	28	28	128	28	28	128	conv3-3	1	1	3	3	128	128	147584
9	28	28	128	28	28	128	conv3-4	1	1	3	3	128	128	147584
10	28	28	128	14	14	256	conv4-1	2	0.5	3	3	128	256	295168
11	14	14	256	14	14	256	conv4-2	1	1	3	3	256	256	590080
12	14	14	256	14	14	256	conv4-3	1	1	3	3	256	256	590080
13	14	14	256	14	14	256	conv4-4	1	1	3	3	256	256	590080
14	14	14	256	7	7	512	conv5-1	2	0.5	3	3	256	512	1180160
15	7	7	512	7	7	512	conv5-2	1	1	3	3	512	512	2359808
16	7	7	512	7	7	512	conv5-3	1	1	3	3	512	512	2359808
17	7	7	512	7	7	512	conv5-4	1	1	3	3	512	512	2359808
	7	7	512	1	1	512	avg pool	7	0	7	7	512	512	0
18	1	1	512	1	1	1000	fc					512	1000	513000
Total														11,511,784

Example Segmentation architecture : resnet 18



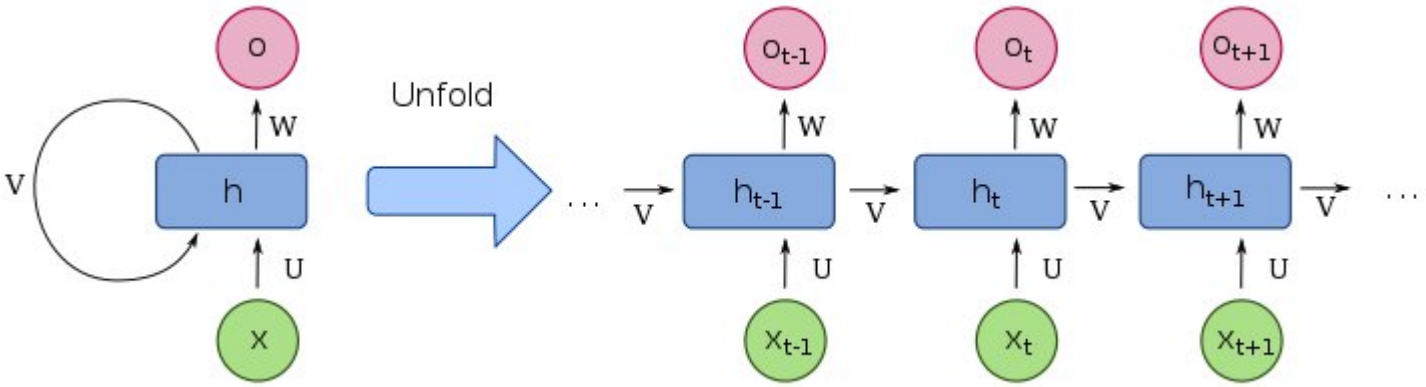
Many different types of module for computer vision

- Researchers came up with a lot of different designs over the years
- Most of the computer vision deep learning networks takes an « encoder / decoder » approach to extract the latent representation, and then analyze it
- You can acces some of the most popular state of the art implementation on those sources :
  - https://github.com/huggingface/pytorch-image-models
  - https://github.com/qubvel/segmentation_models.pytorch

Each deep learning tasks has its own architecture

- Different tasks require different design
- For examapl, audio or text analysis may not fit image segmentation models
- A wide variety of architecture, more or less complex exists

Recurrent Neural Networks



Transformer mechanism (chat-GPT)

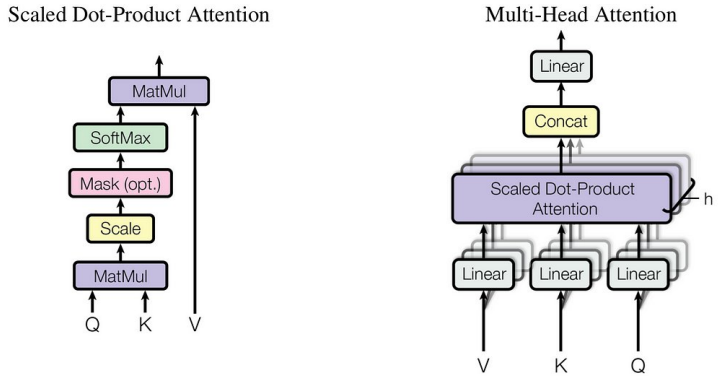


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

Conclusion

- Deep learning may look scary at first, but the theory behind it stays reltively simple once you understand the concept
- There exists a lot of different sources to build your knoweldge and intuition on what goes on behind the scene of an AI :
  - MIT deep learning series on YT : https://www.youtube.com/@AAmini
  - 3blue1brown excellent visualisation series : https://www.youtube.com/@3blue1brown/videos
  - Michael Nielsen book for a math oriented explanation : http://neuralnetworksanddeeplearning.com/
  - Andrew Ng's famous course on coursera : https://www.coursera.org/instructor/andrewng