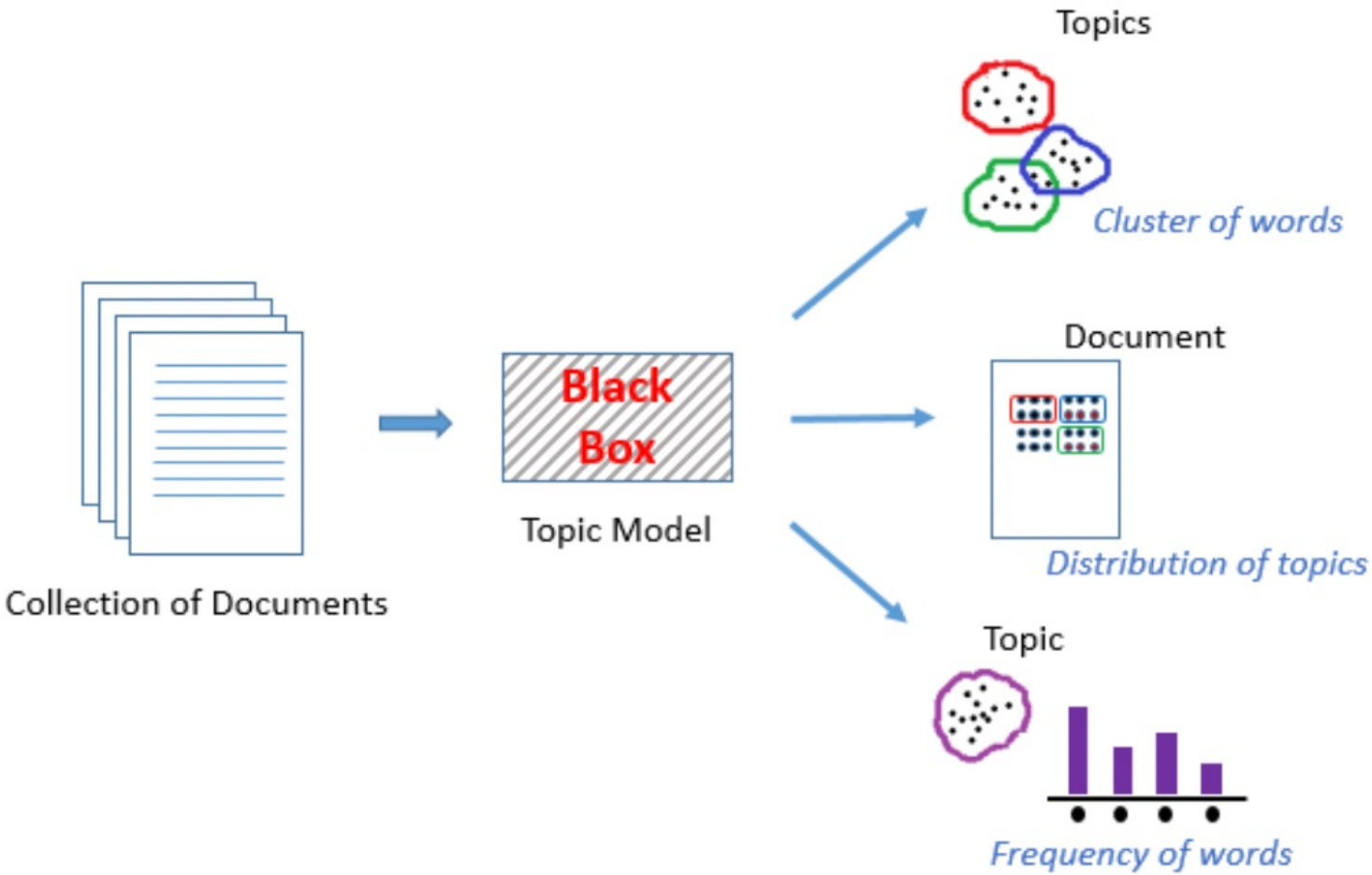
Lec08. Data Analytics for Images

With Deep Learning

Recap: Topic Modeling



Recap: Latent Semantic Indexing

- Represent Matrix M as M = K.S.D
  - Such a decomposition always exists and is unique

M = K x S x D

Words	Documents 0	1	2	3	4
banana	2	0	4	2	0
kiwi	1	0	5	0	0
apple	1	1	7	4	0
computer	0	1	0	0	4
screen	0	1	0	0	1

=

Words	Topics A	B	C
banana	0.5	0	0.1
kiwi	0.3	0	0.2
apple	0.2	0.2	0.3
computer	0	0.4	0.2
screen	0	0.4	0.2

x

a	0	0
0	b	0
0	0	c

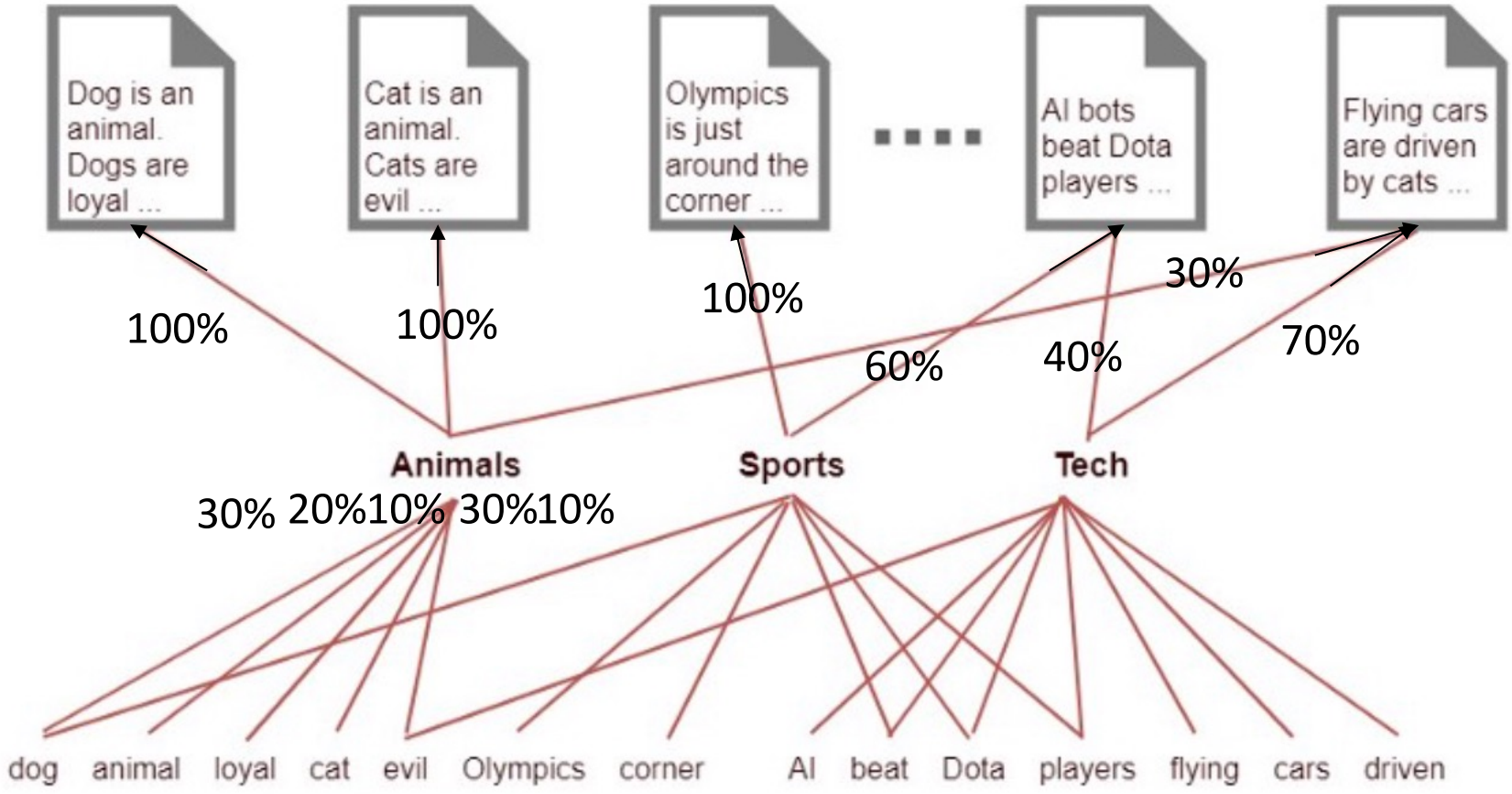
x

Topics	Documents 0	1	2	3	4
A	0.8	0	1	0.7	0
B	0	0.9	0	0	1
C	0.2	0.1	0	0.3	0

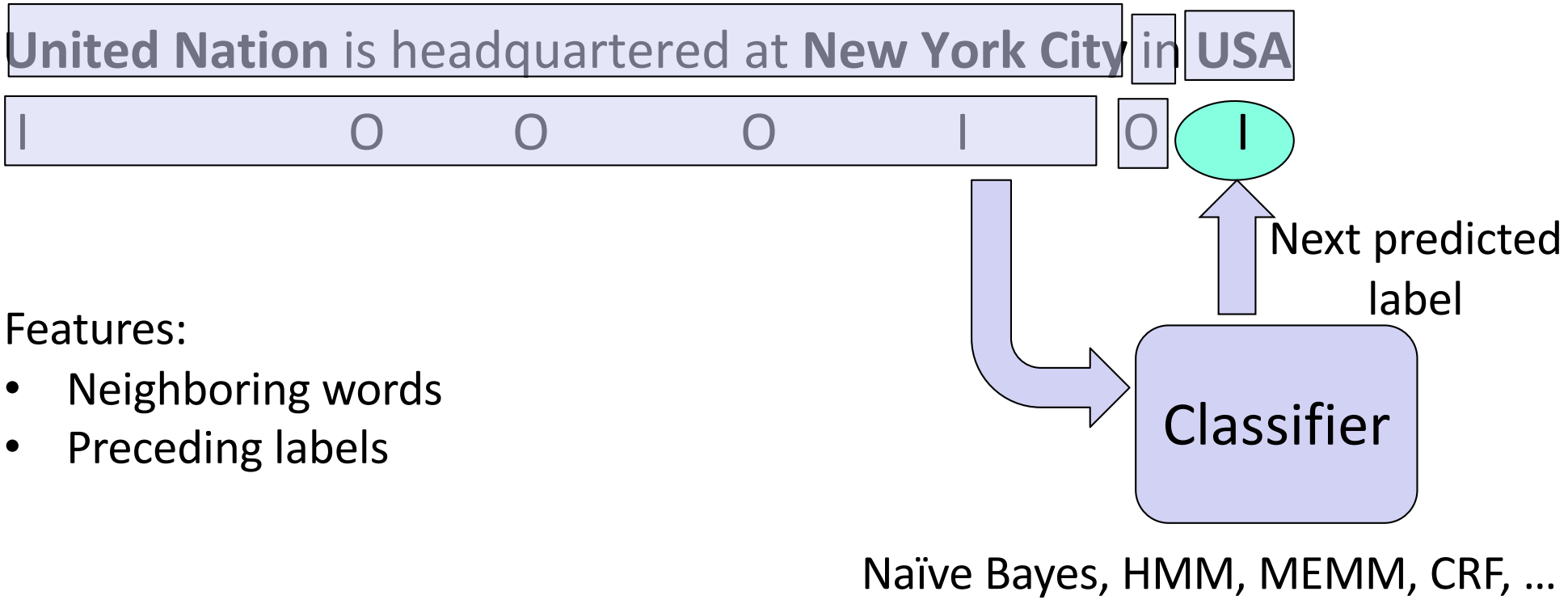
- S is a diagonal matrix of singular values in decreasing order: each value represents the weight of the corresponding topic
- K is the term-topic matrix
- D is the document-topic matrix

Recap: Latent Dirichlet Allocation

- **Idea**: assume a document collection is (randomly) generated from a known set of topics (probabilistic generative model)



Recap: Named Entity Recognition



Features:

- Neighboring words
- Preceding labels

Naïve Bayes, HMM, MEMM, CRF, …

Course structure

W1. Data Processing with Python

W2. Data Exploration with Python

W3. Data Modeling with Pytyhon

W4. Data Analytics for Timeseries

Holiday

W5-6-7. Data Analytics for Texts

W8. **Data Analytics for Images**

W9. Data Analytics for Graphs

W10-11. Data Analytics for Other Data

W12. Revision

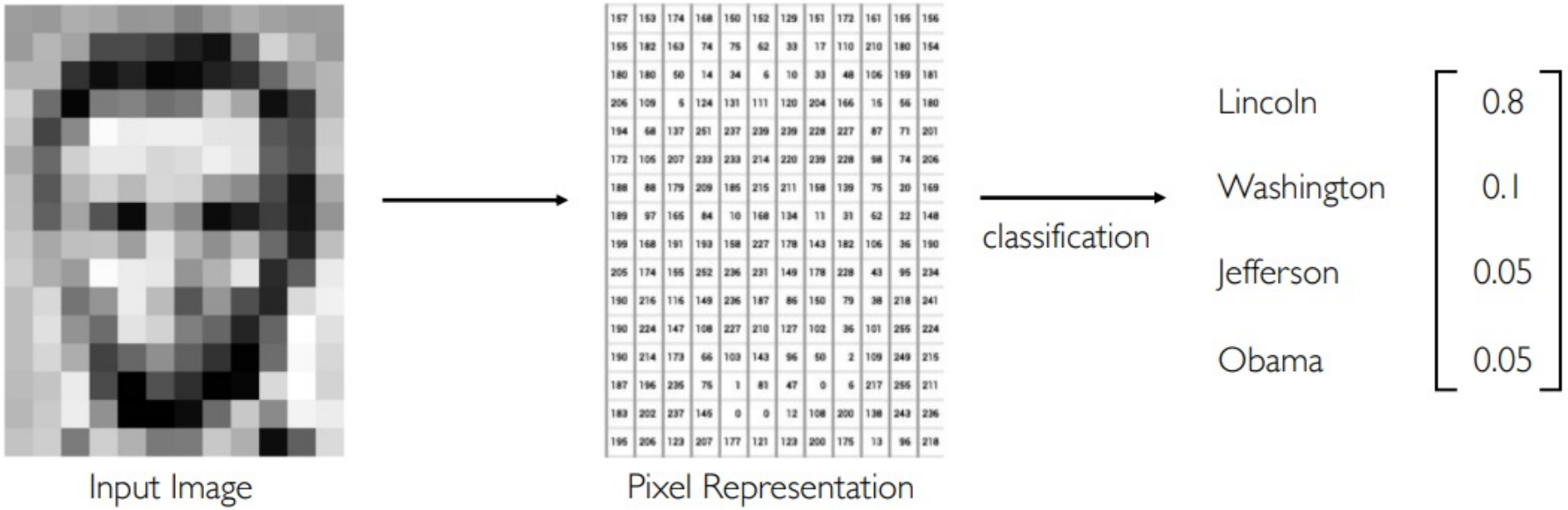
Lecture Content

- Image Analysis
- Object Detection
- Face Detection/Recognition

IMAGE ANALYSIS

What are images?

- In computers, images are just numbers
  - We can still apply traditional analysis methods



- **Regression**: output variable takes continuous value
- **Classification**: output variable takes class label. Can produce probability of belonging to a particular class

However, images are not that easy

- Unlike other kind of data, images have **spatial structure**!
  - Indeed, the same object can be seen from various viewpoints, deformed, scaled, occluded, etc...

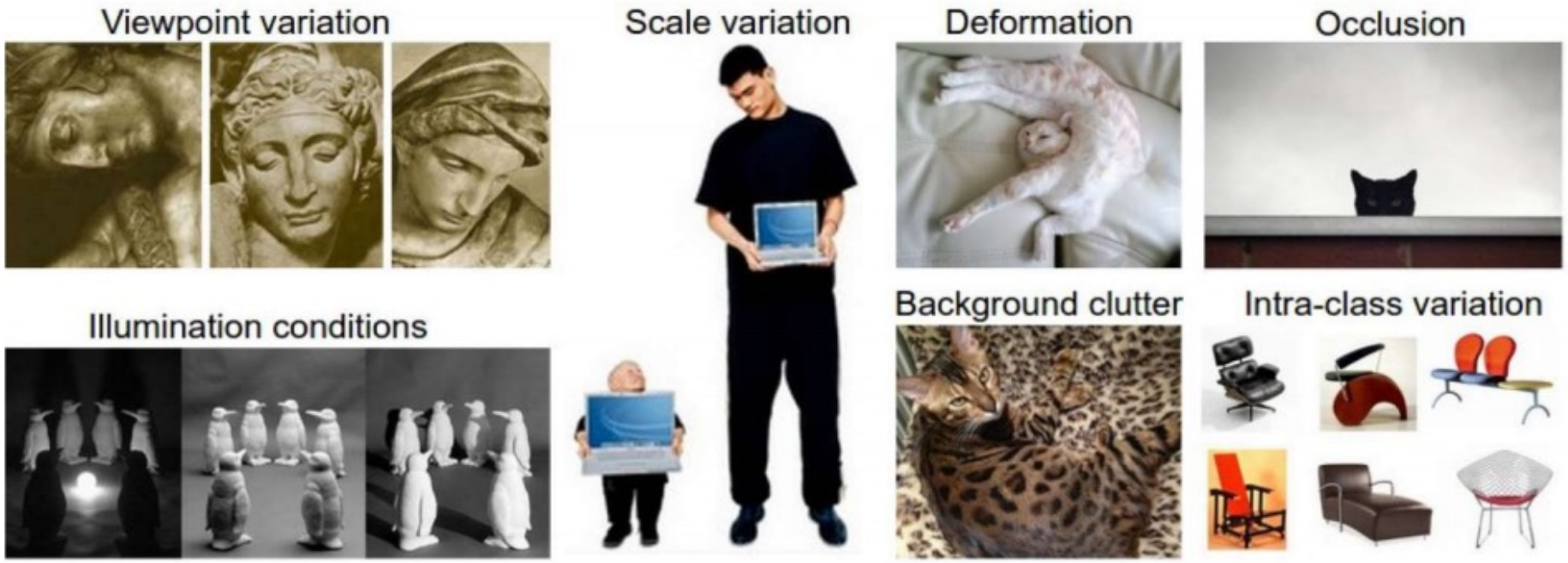
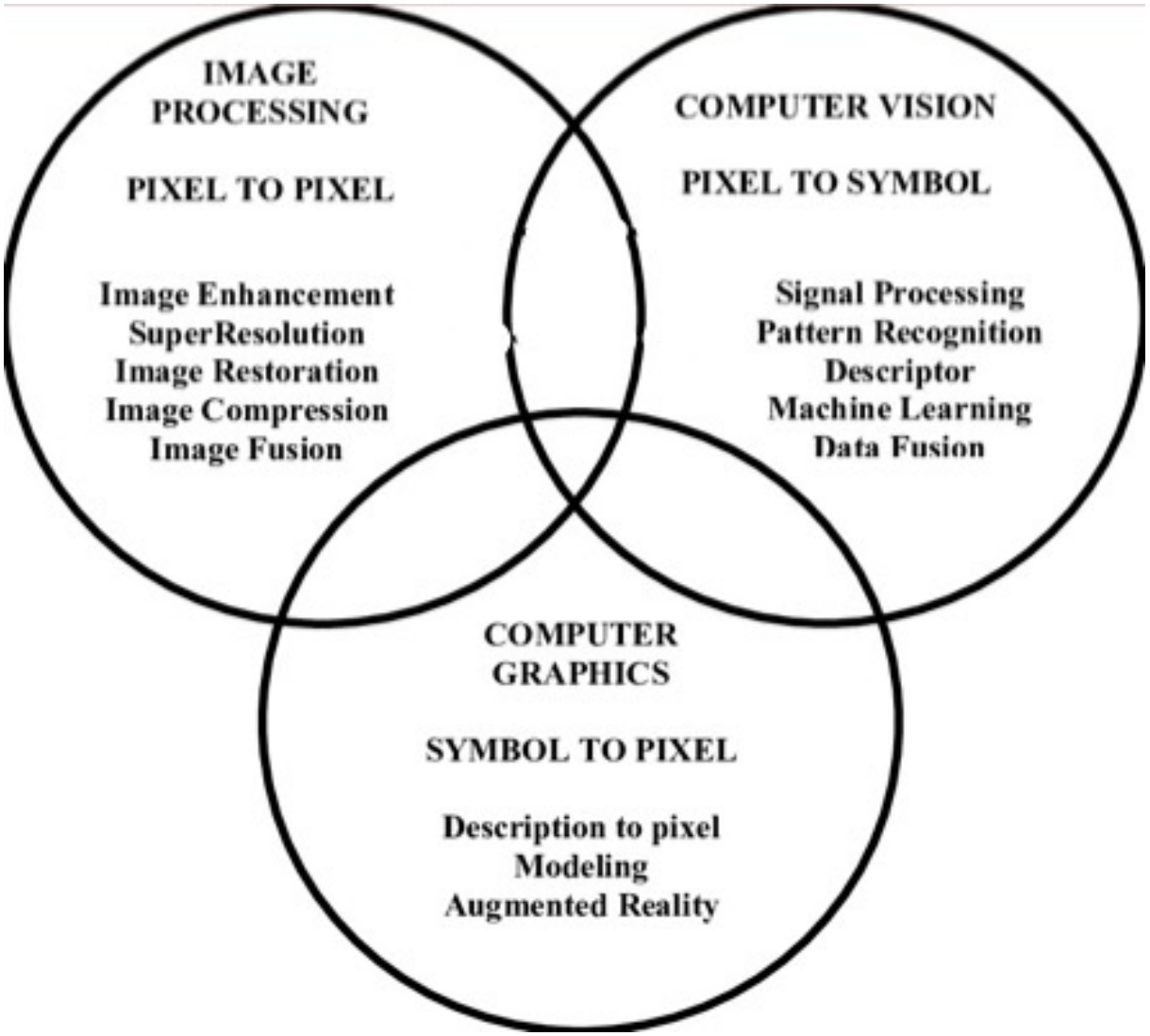
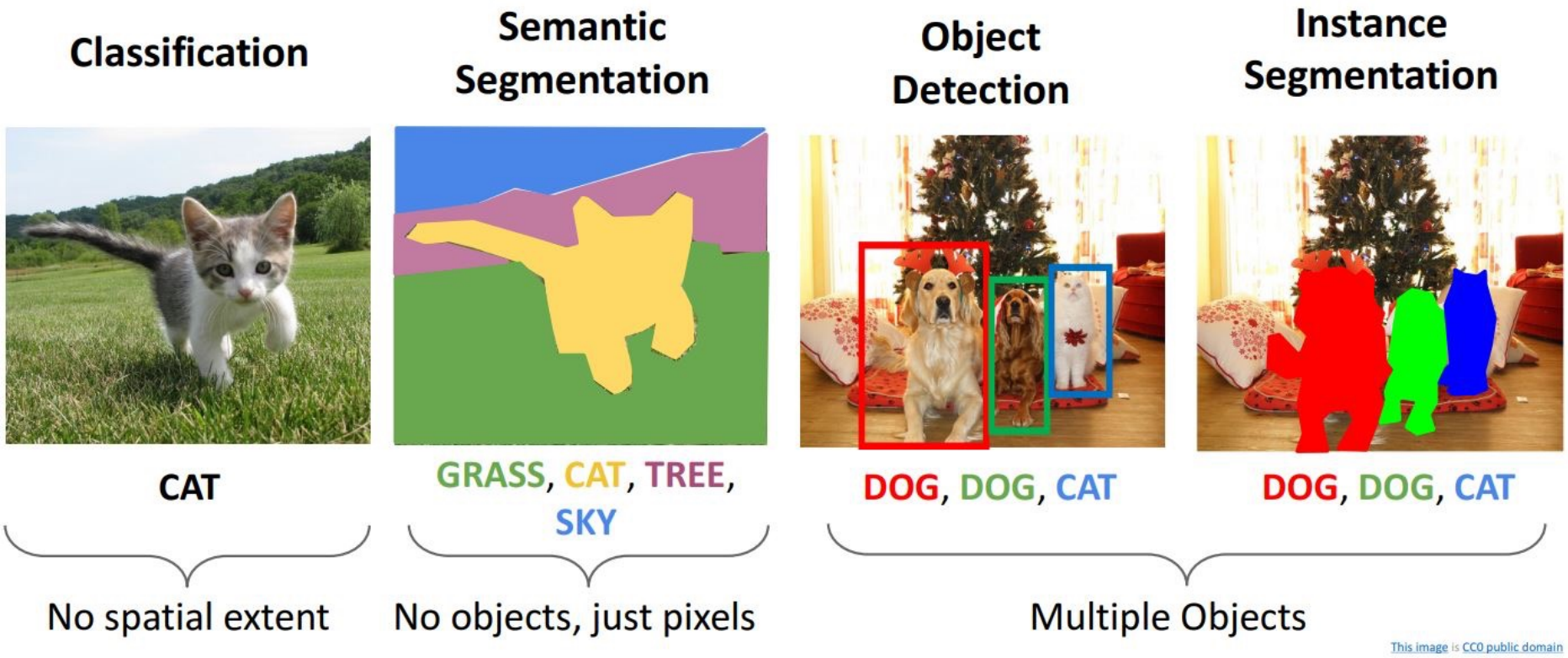


Image Analysis Paradigms



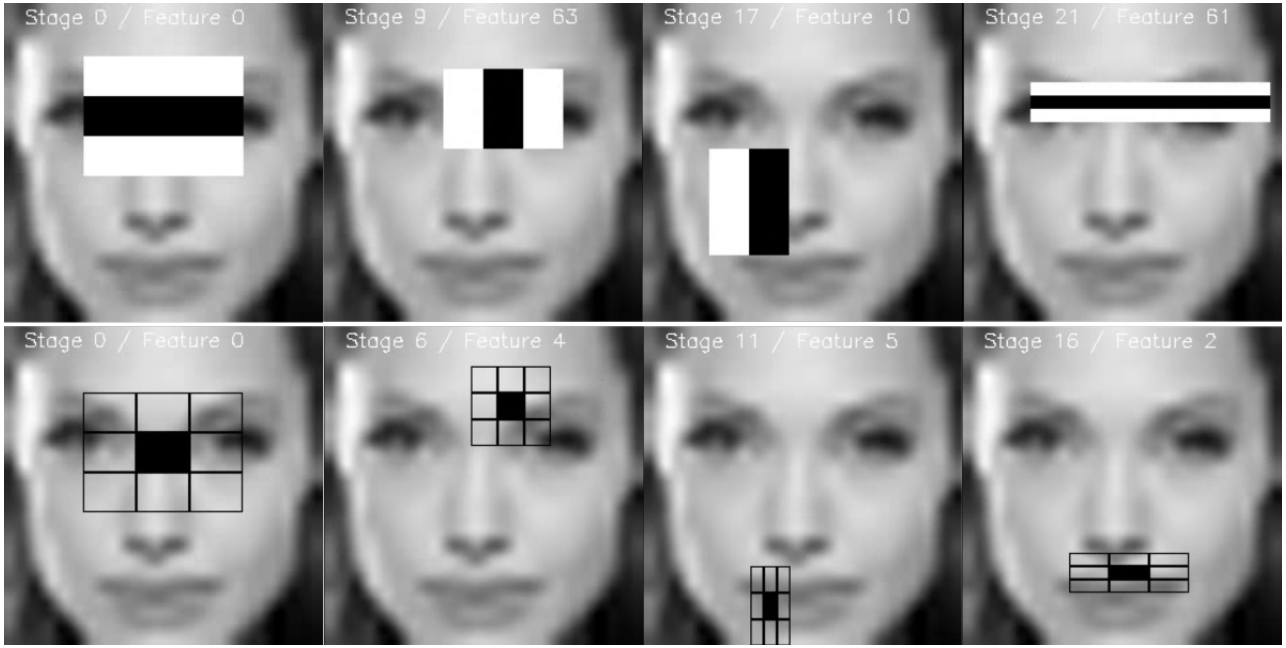
In this course, we focus on some Computer Vision applications

Computer Vision Applications



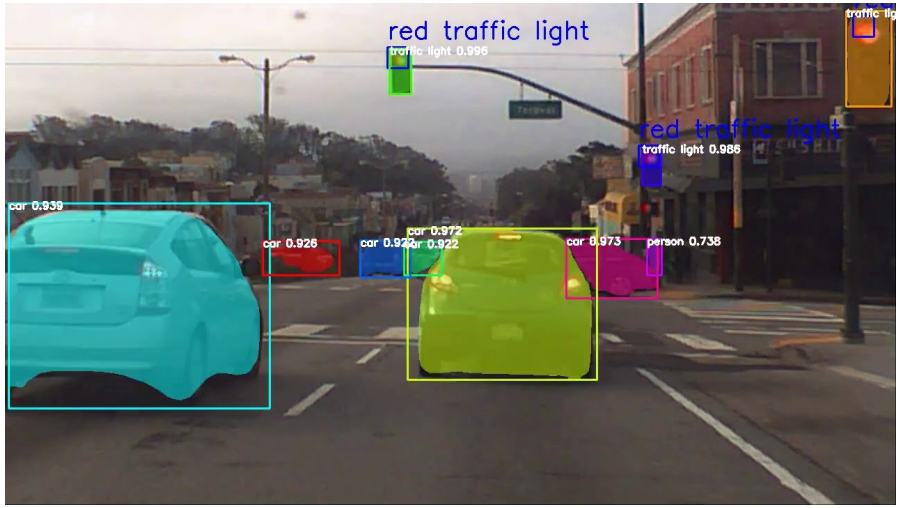
Classical Methods: Pre-defined features

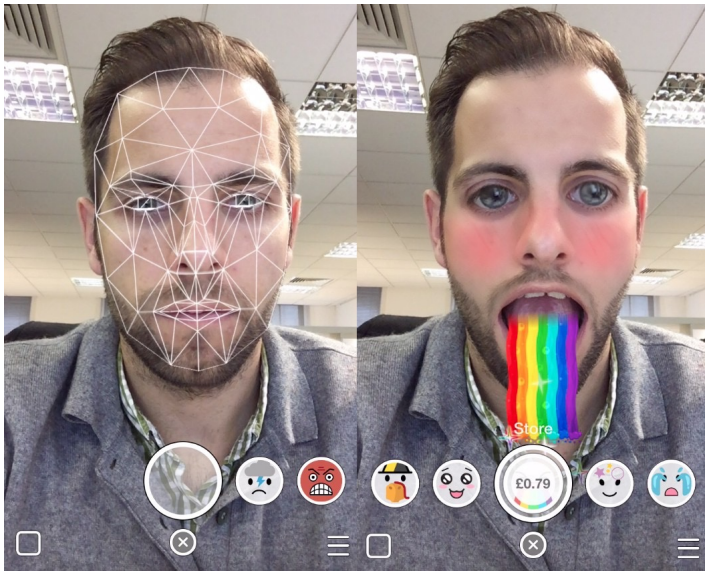
- Usecase: Face detection – locates where the face is in a picture
- Many cameras were most likely using a so-called Haar Cascade Classifier. This algorithm takes advantage of some features most faces share, e.g.:
  - A mouth is a horizontal line
  - Eyes region is darker than the cheeks
  - Nose is a brighter vertical area
  - Locations of nose, eyes, mouth...



Classical Methods: Pre-defined features

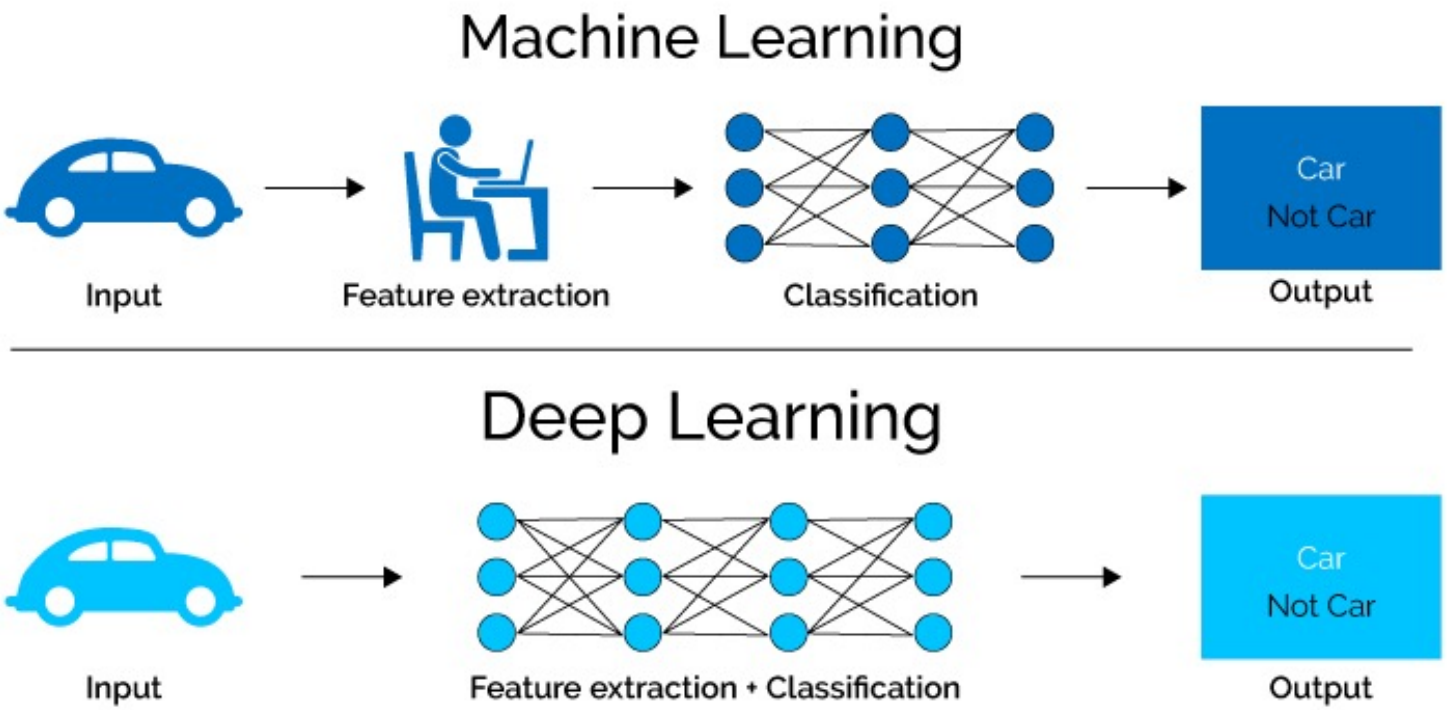
- **Pros:**
  - Specialized for an application
  - Fast
- **Cons:**
  - Not generalize well
  - Too many applications do not share the same pre-defined features





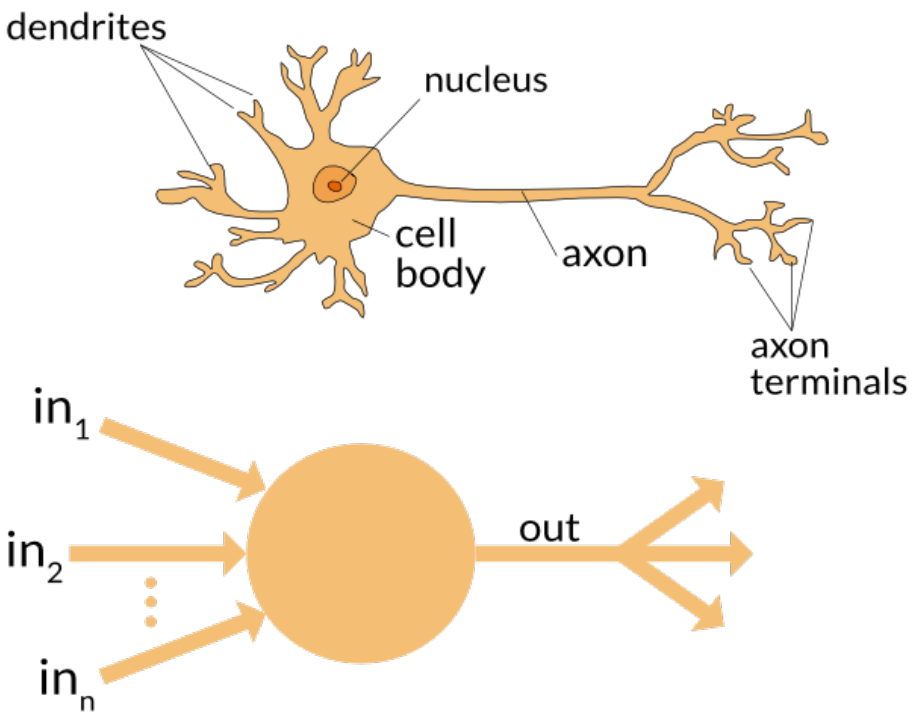
Deep Learning to the Rescue

- Features are learnt automatically from the data
- Can be applied or reused to different applications or domains



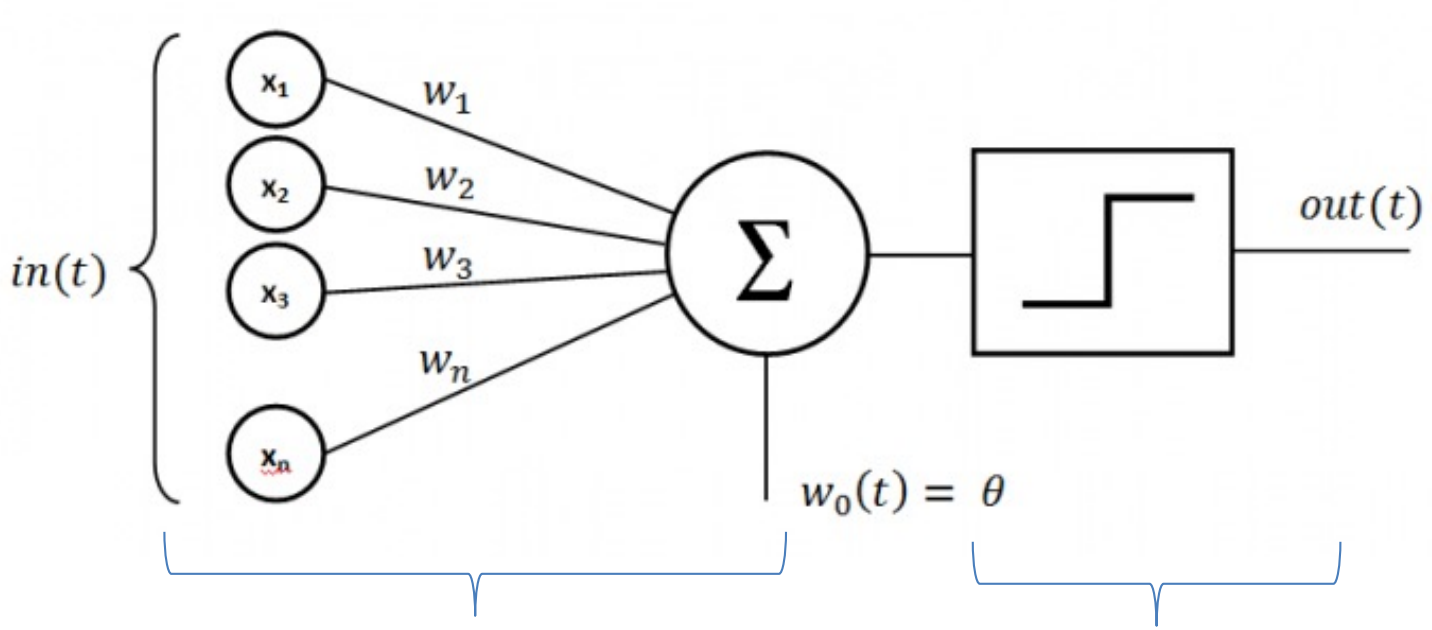
Building blocks of deep learning

- **Perceptron:** mimic human brain



Mathematical Interpretation

- A perceptron can simply be seen as a set of inputs, that are weighted and summed.



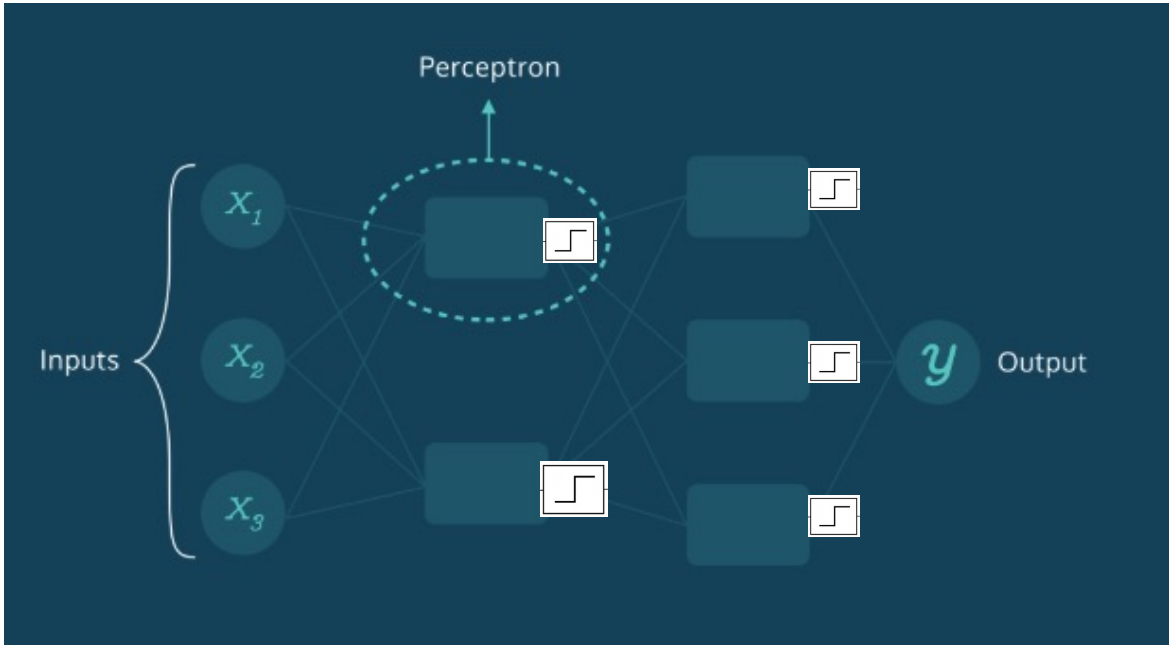
Weighted sum: $\sum_i x_i \times w_i + b$

- Input: x_i
- Weight: w_i
- Bias: b (or denoted as w_0 or θ)

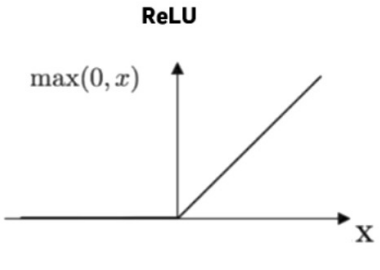
Activation function

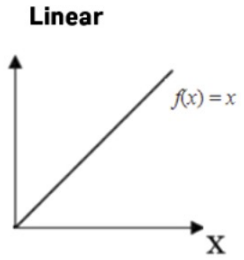
- Transform value domain
- Control linearity/non-linearity

Generic neural networks



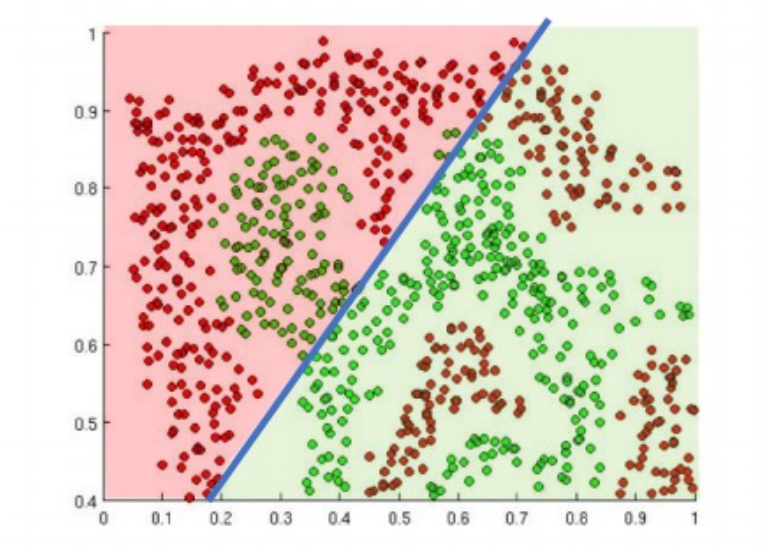
Activation functions can appear after each layer to adjust the linearity of the output (note that if you want to preserve the output as-is, use ReLU or Linear activations)



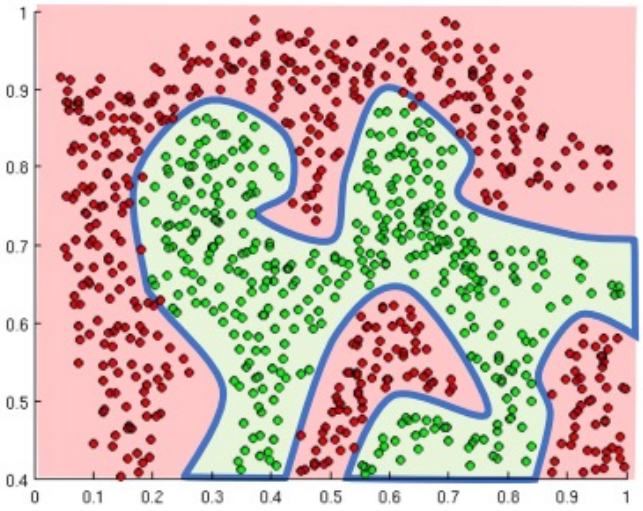


Activation functions

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



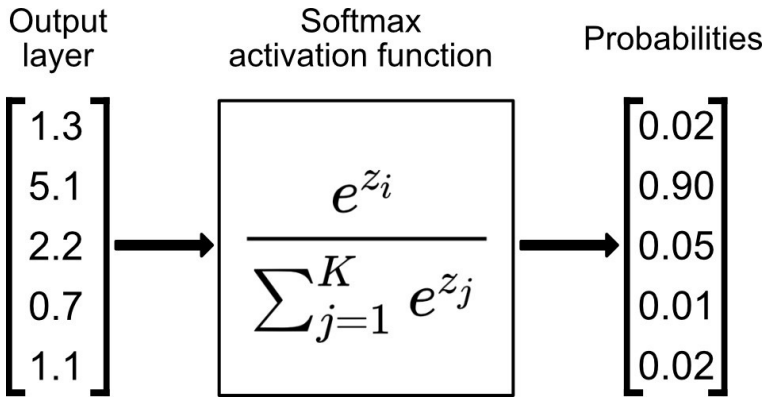
Linear Activation functions produce linear decisions no matter the network size

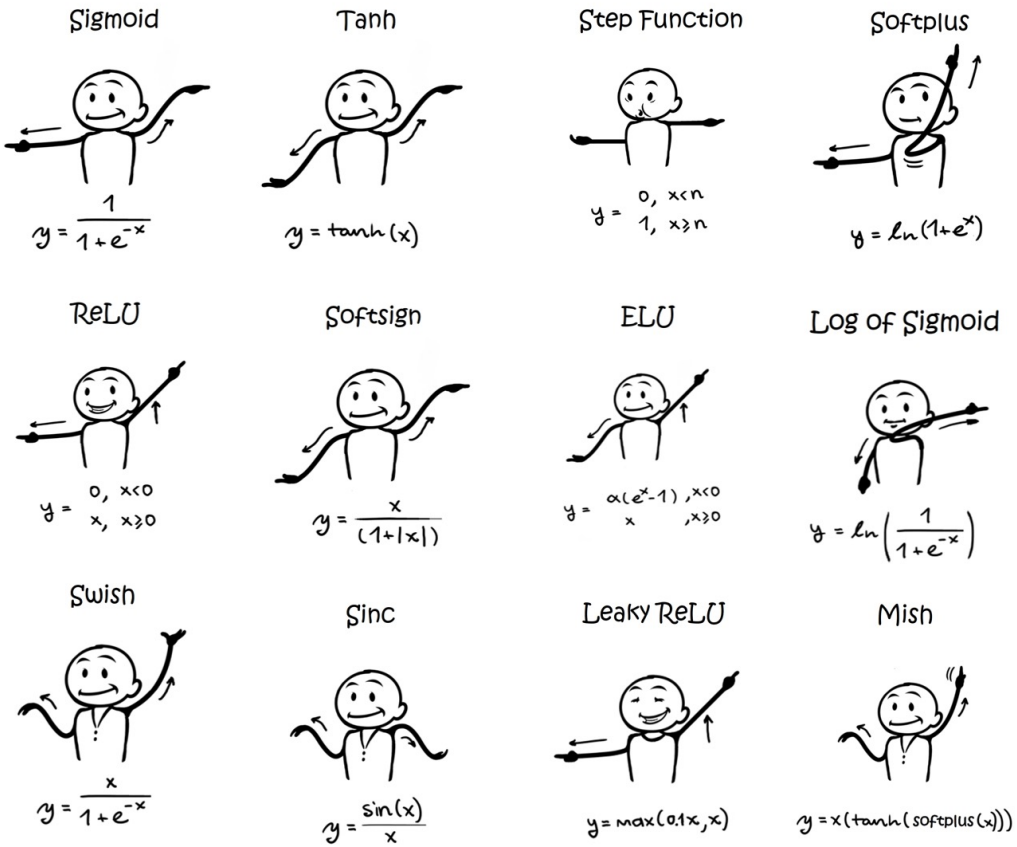


Non-linearities allow us to approximate arbitrarily complex functions

Types of activation functions

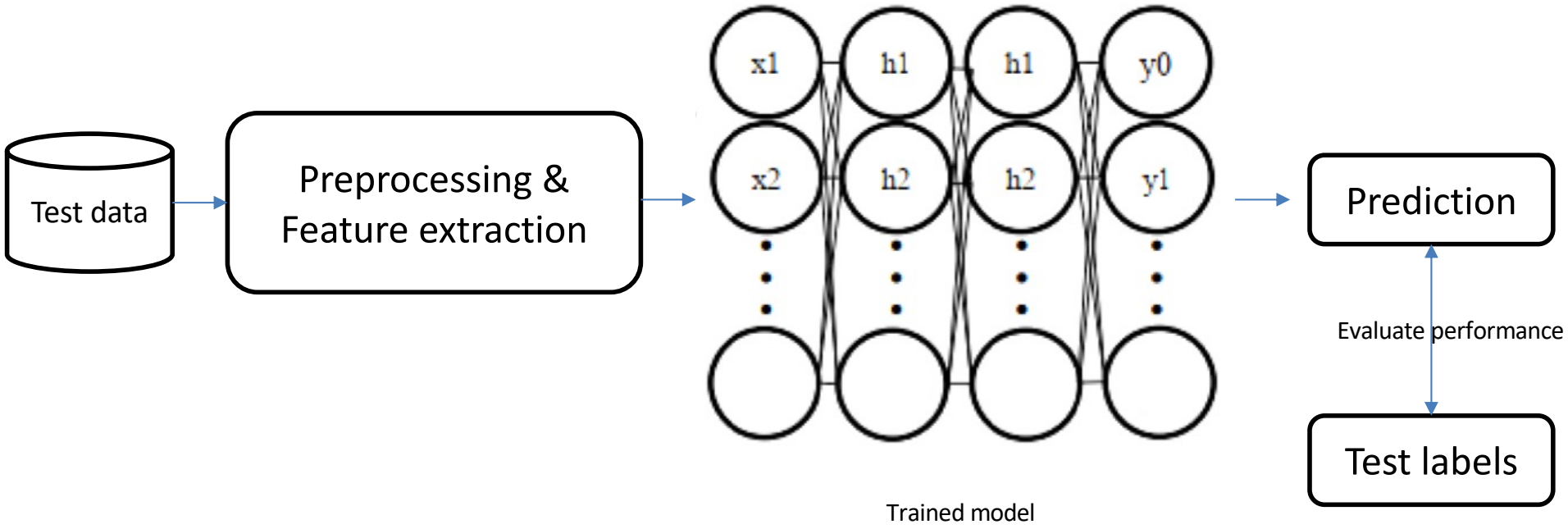
- **Popular:**
  - Sigmoid, tanh: add non-linearity
  - ReLU: remove negative values
  - Step function: binary classification
  - Softmax: multi-label classification, e.g.





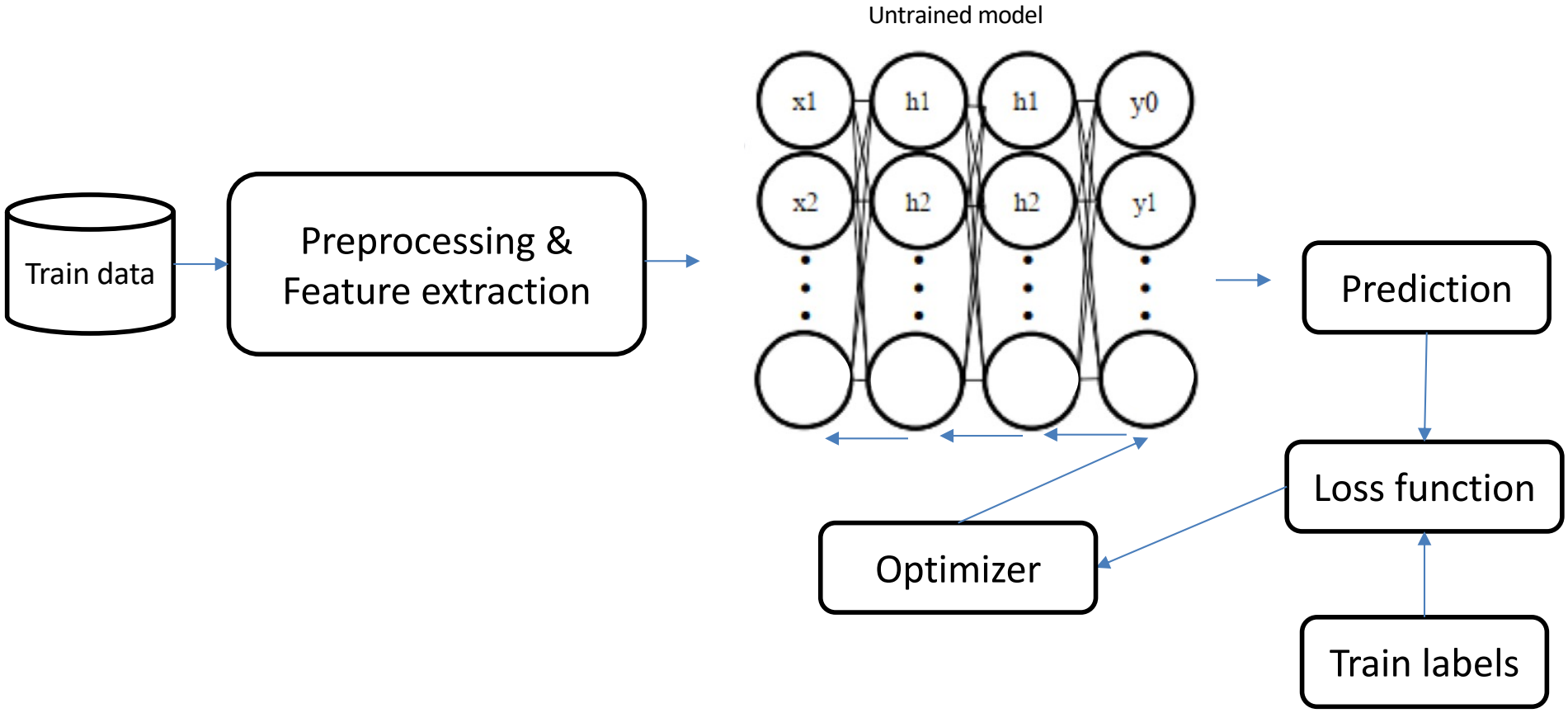
Forward propagation

- Use the weights to compute output from an input



Backward propagation

- Use the difference (loss) between train output and current prediction to adjust the weights



Loss Functions

- **Regression output:**
  - E.g. Mean Squared Error Loss
    - Penalize big errors

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

- **Binary classification output:**
  - E.g. Binary Cross-entropy
    - The ground-truth is a one-hot vector
    - Favor clear distinction between output probabilities

$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}(Y_i \cdot \log \hat{Y}_i + (1 - Y_i) \cdot \log (1 - \hat{Y}_i))$$

- **Multi-class Classification output:**
  - E.g. Categorical Cross-entropy
    - Generialization of binary cross-entropy
    - The ground-truth is a one-hot vector
    - Favor clear distinction between output probabilities

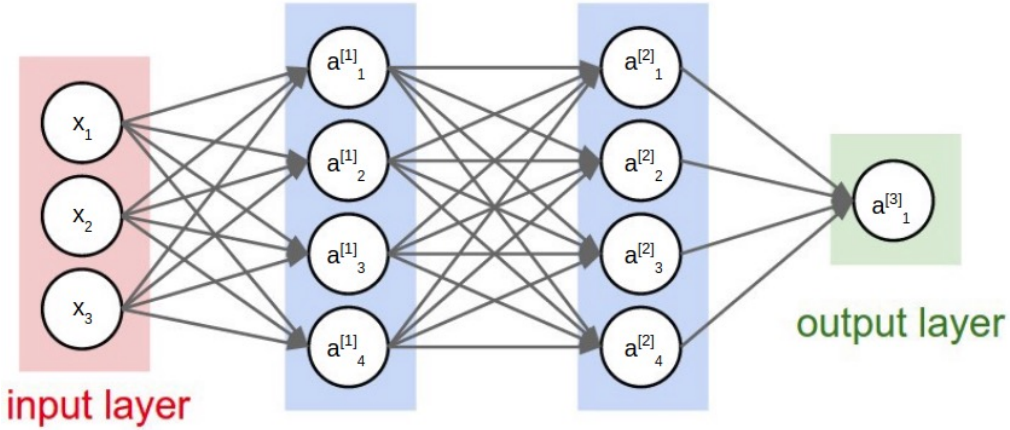
$$\text{Loss} = -\sum_{i=1}^{\substack{\text{output}\\ \text{size}}} y_i \cdot \log \hat{y}_i$$

https://playground.tensorflow.org/

Limitations of Dense Neural Networks

- Too many paramters to train → Slow
- Redundant: does not capture spatial correlations:
  - In a face the eye is always close to the nose, and the nose is always in the middle of the face.
  - A bike always has two wheels.
  - All numbers and letters have a particular shape.

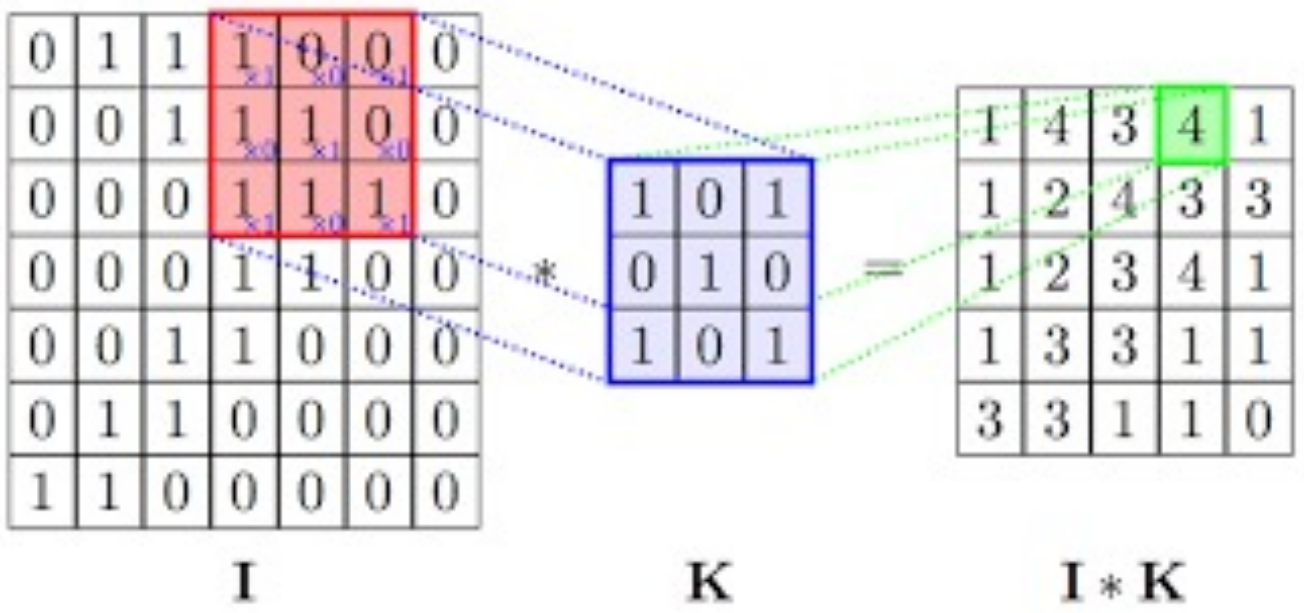
Input Image



We do not want every neuron in the second layer connected to every input pixel!

Convolution Operation

- We look at a small region at the same time only
- A convolution is a mathematical operation that computes features out of a small region.
  - For example, below is the convolution result of an image with a 3x3 kernel of ones:



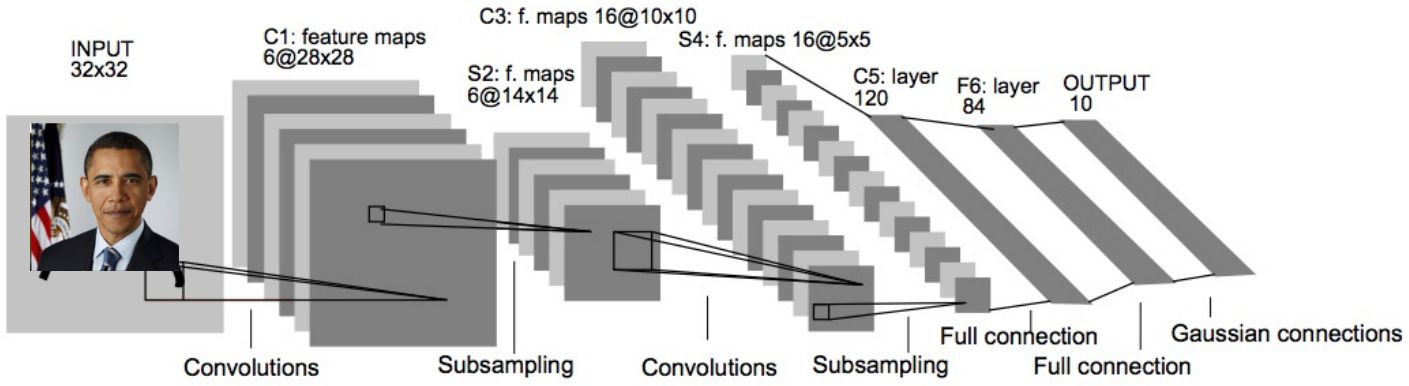
Convolutional Kernels

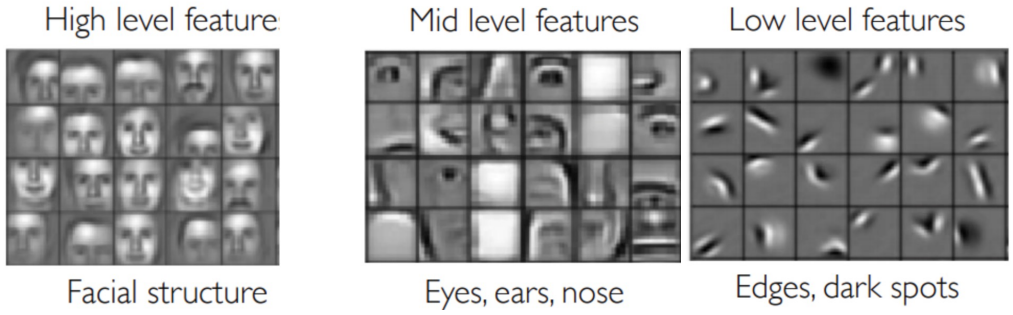
- A wisely chosen kernel can be really useful to detect patterns in images. Here is a list of commonly used kernels and what they do on a given image

Kernel	Usage	Example
$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	Identity	
$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	Edge detection	
$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	Identity	
$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	Edge detection	
$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	Sharpening	
$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	Gaussian blur	

Secret of DL: Multiple convolutions

- History:
  - CNNs were not always famous because of hardware limitation.
  - CNNs exploded with AlexNet in 2012, a CNN that made a breakthrough in the Computer Vision field thanks to deep architecture and GPU.
- Intuition:
  - Look at the big region then look at smaller regions





Regularization by Subsampling (Pooling Layer)

- A pooling layer is a layer that will reduce the size of the feature maps by taking the average or max value of a given number of pixels.
- → Regularization: Force the neural network to learn abstract features, instead of just remembering data

Below is an example of pooling layer, with a kernel of 2x2 and a stride of 2 applied on an image of 4x4

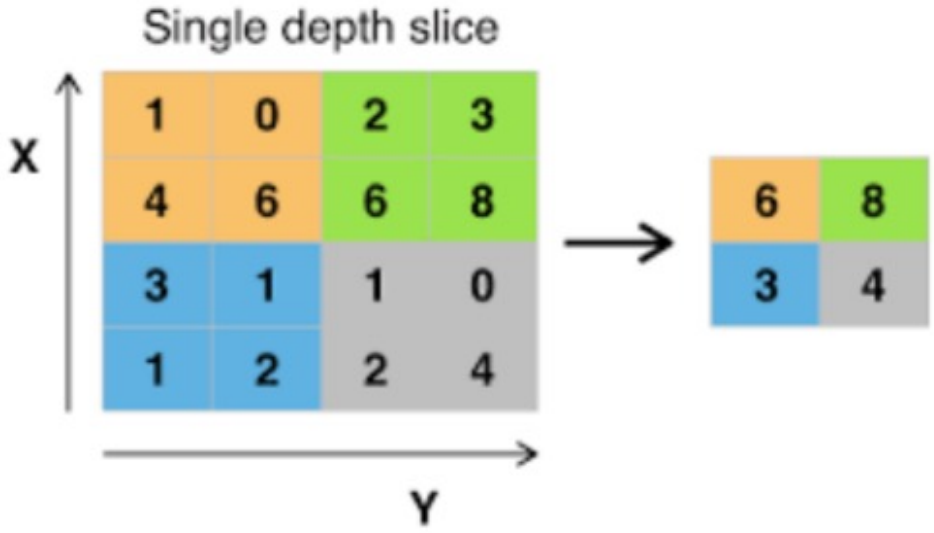
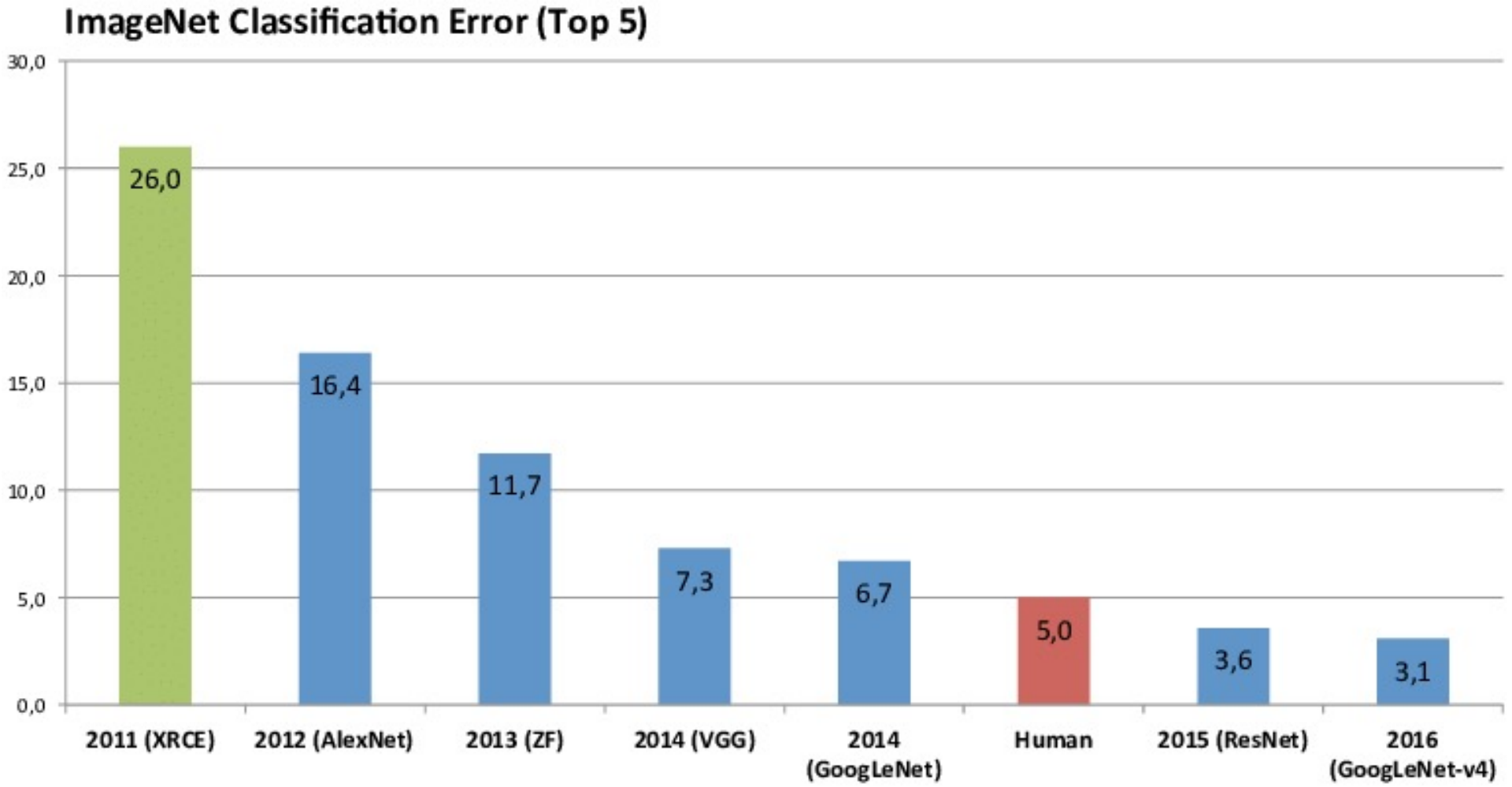


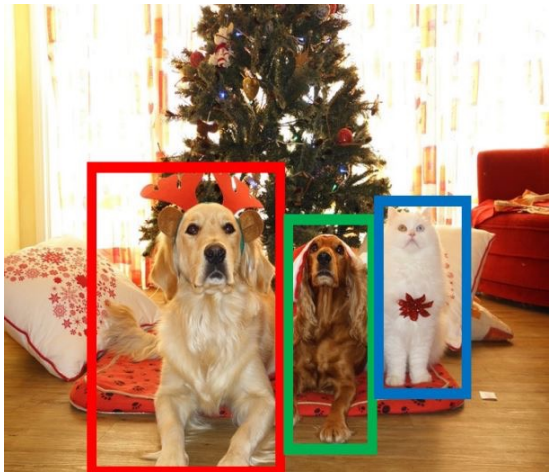
Image Analysis Performance



OBJECT DETECTION

Object Detection

- It's all well and good to be able to classify images between cats and dogs ; but what if we wanted to know **where** in the image the cat is situated ?
- Definition:
  - The **input** will be a single RGB image
  - The **output** will be a set of detected object. The aim is to predict, for each object :
    - A **category label** (*What* is the object ?), selected from a fixed, known set of categories
    - A **bounding box** (*Where* is the object ?), usually made of four informations (x, y, width, heigth)

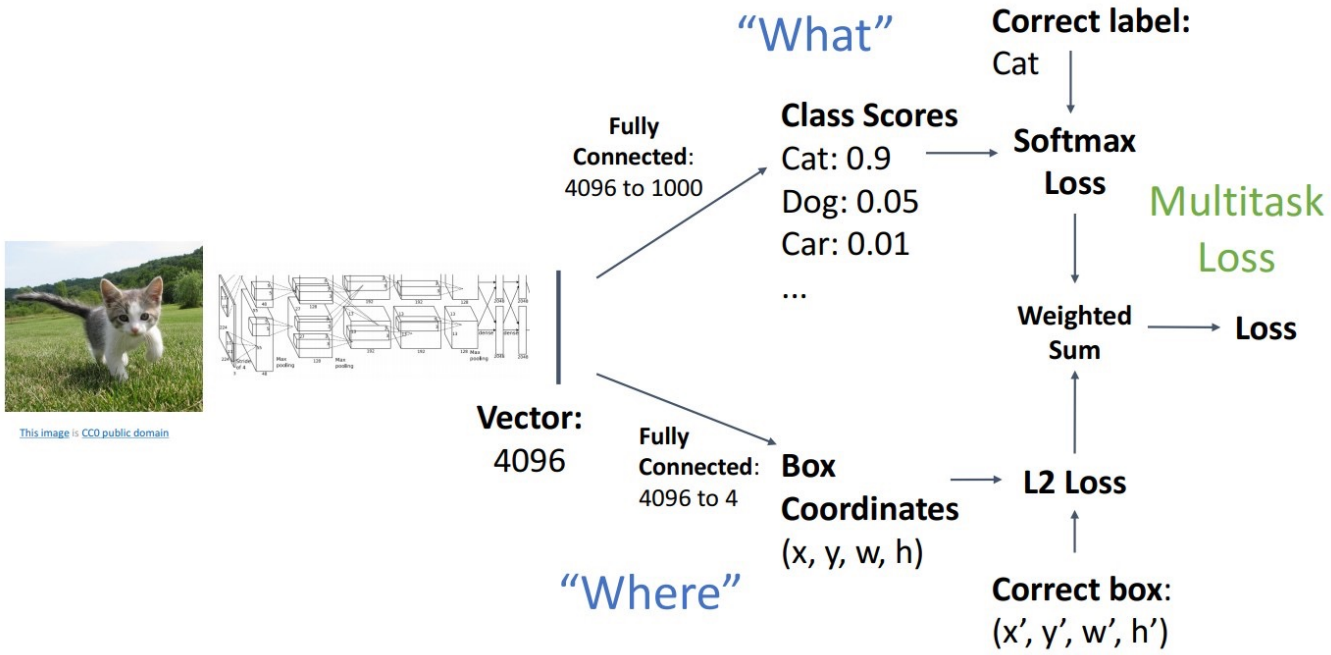


Object Detection

- Why is it so hard:
  - **Multiple ouputs** : Need to input variable numbers of objects per image
  - **Multiple types of output**: Need to predict "what" (category label) as well as "where" (bounding box)
  - **Large images**: Classification works at 224x224; need hight resolution for detection, often 800x600

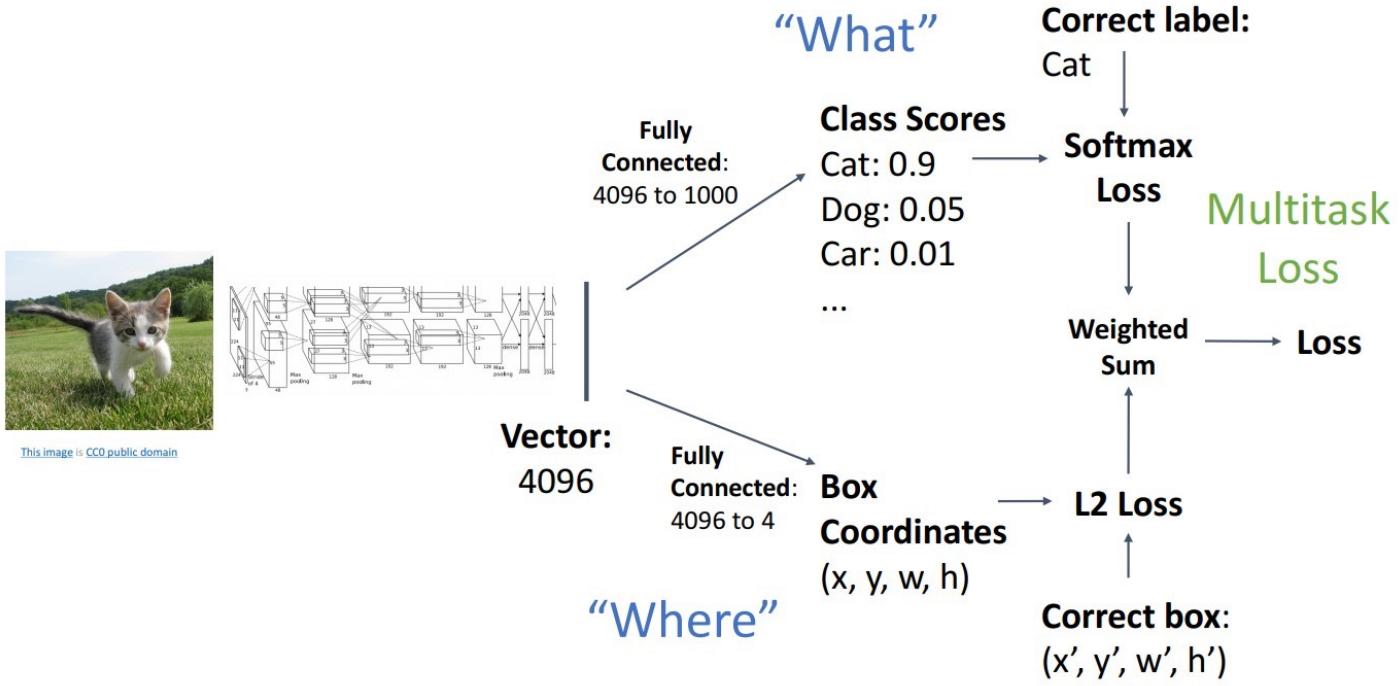
Detect Single Object

1. **First step : Identify object : "What"**
   - As seen so far, a simple classification is carried out, using a CNN.
   - After the fully connected layer (you also know it under the name of Dense layer), we obtain an array of class scores, that indicate the probability of each class.
   - With the correct label, we can apply a **softmax function** in order to calcule the loss.
2. **Second Step : Localisation the object : "Where"**
3. **Combine Loss : Multitask loss**



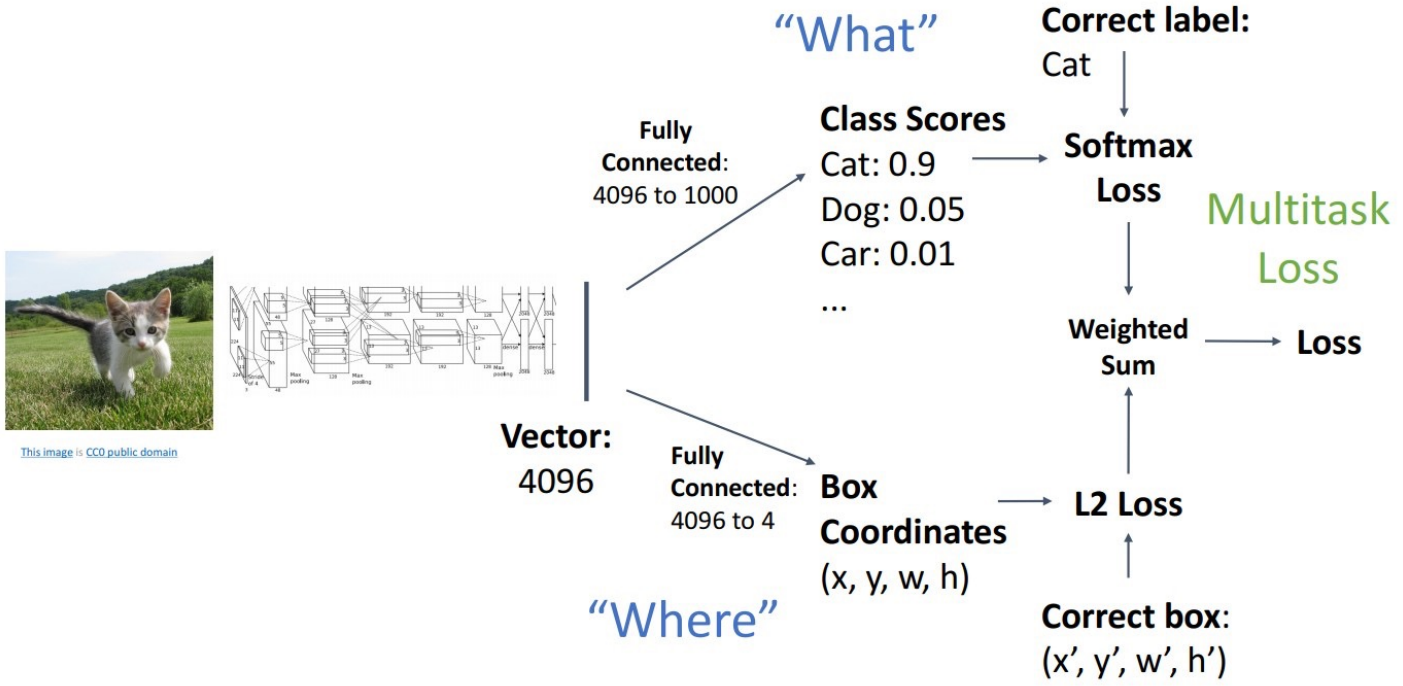
Detect Single Object

1. **First step : Identify object : "What"**
2. **Second Step : Localisation the object : "Where"**
   - Here, we can just treat the localization as a regression problem : we need to predict four continuous variables (x, y, width, heigth of the bounding box)
   - We usually use a **L2 loss** in this step.
3. **Combine Loss : Multitask loss**



Detect Single Object

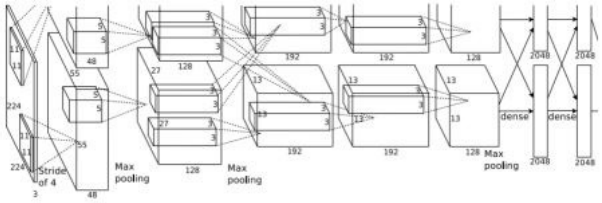
1. **First step : Identify object : "What"**
2. **Second Step : Localisation the object : "Where"**
3. **Combine Loss : Multitask loss**
   - Now we have 2 differents loss : A softmax loss and a L2 loss ;
   - **We can sum the two losses in order to have one multitask loss**
   - Armed with this multitask loss, we can train our neural network to perform object detection !



Detect Multiple Objects

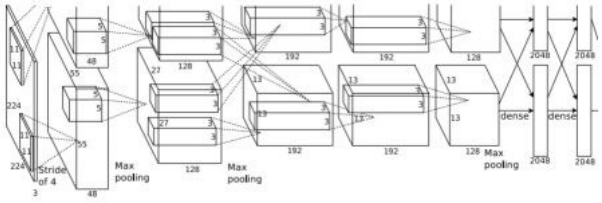
Naïve approach:

- Do multiple single-object problem
- Does not scale



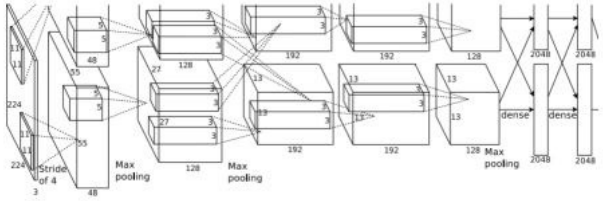
CAT: (x, y, w, h)

4 numbers



DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

16 numbers



DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Many numbers!

Detect Multiple Objects

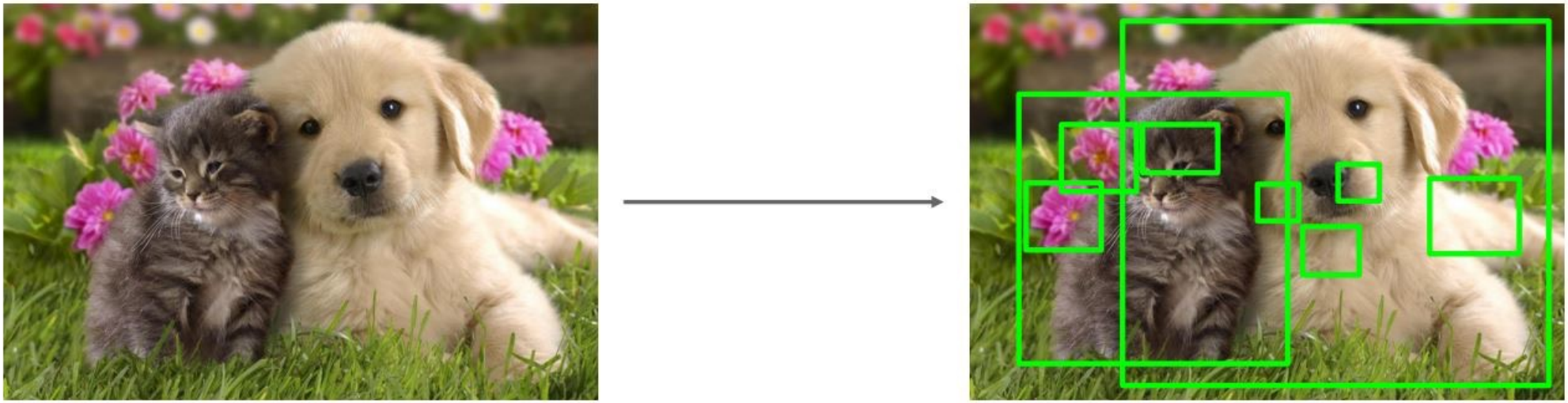
Better Solution: Sliding Window

- Let's apply a CNN to many different crops of an image ;
- Then the CNN classifies each crop as either object or background.
- Limitations:
  - Computationally expensive
  - If the window is too big, we have to use multiple windows and use overlap
    - But too computationally expensive and not always correct

Detect Multiple Objects

Best Solution: Region Proposals

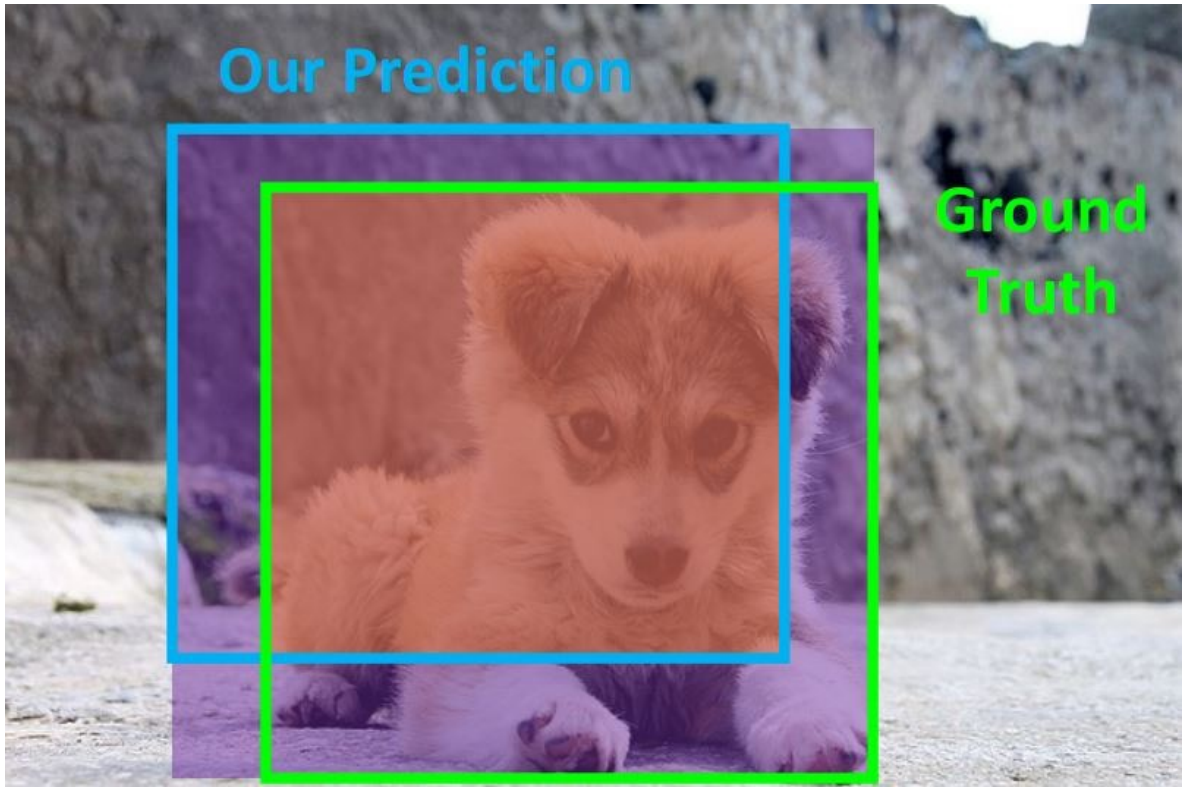
- Find a small set of boxes that are likely to cover all objects
- Often based on heuristics: e.g. look for “blob-like” image regions
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU



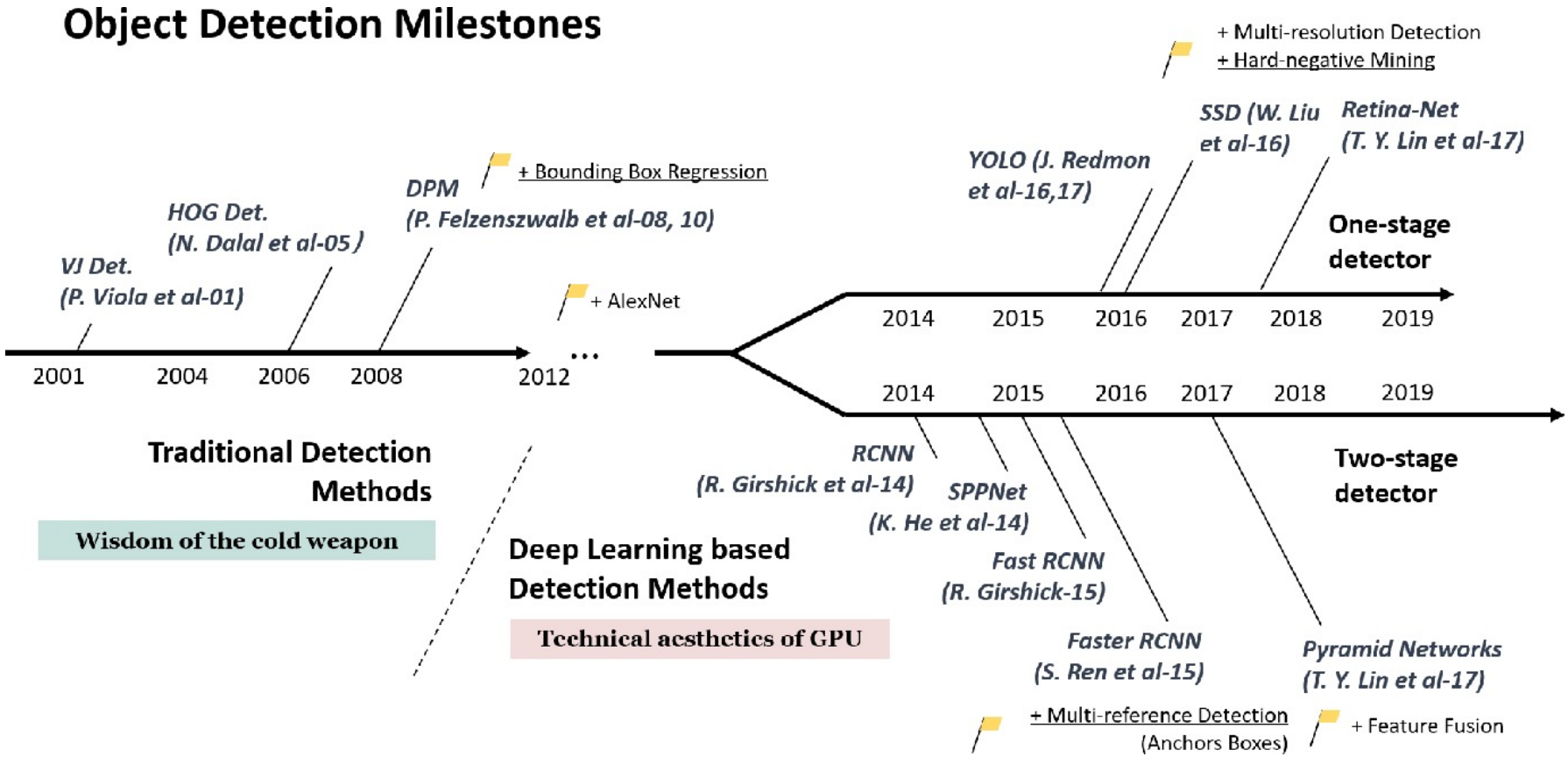
Detect Multiple Objects

Best Solution: Region Proposals

- Then resize the box and compare with the ground-truth boxes
- Use Intersection Over Union (IoU) loss (Jaccard index)



History of Object Detection



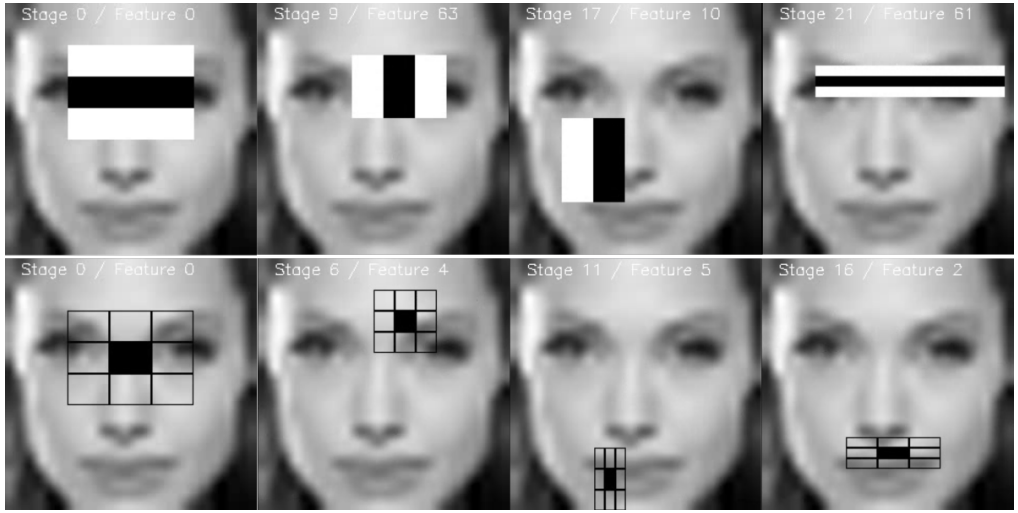
Detectron2: a state-of-the-art

- Detectron2 is Facebook's new library that implements state-of-the-art object detection algorithms.
- Object detection is hard! Don’t implement it yourself
- We will see how to use Detectron2 in the lab.

FACE DETECTION/RECOGNITION

Face Detection

- Face detection and recognition is used everyday by more and more companies. Some of the most famous applications could be the snapchat filters or the Apple faceID.
- Many cameras were most likely using a so-called Haar Cascade Classifier. This algorithm takes advantage of some features most faces share, e.g.:
  - A mouth is a horizontal line
  - Eyes region is darker than the cheeks
  - Nose is a brighter vertical area
  - Locations of nose, eyes, mouth...



Face Recognition

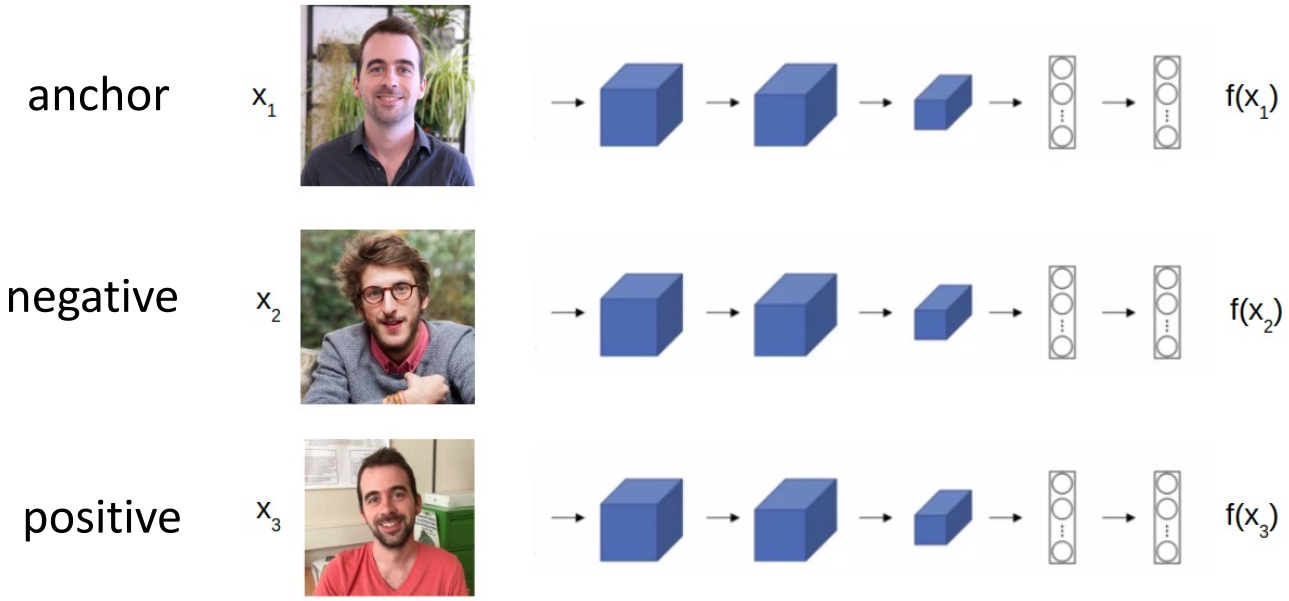
- Face recognition is a tricky problem, because most of the time **you have only few pictures per person**. As you know, it usually takes thousands of pictures to recognize a cat from a dog for a CNN. How to do that with only one image? This is called a **few-shot learning** problem.
- One of the most common applications is the face recognition of employees of a company. So you have:
  - Few pictures per employee
  - Probably not more than hundreds of pictures
  - When a new employee is hired, you don't want to retrain all your network

Face Recognition (cont'd)

- **Approach**: compute distance(picture1, picture2) that returns the dissimilarity between two person faces. We would then add a threshold:
  - if distance(picture1, picture2) ≤ threshold: the pictures are of the same person
  - if distance(picture1, picture2) > threshold: the pictures are of two different persons
  - → However, modeling such distance function is non-trivial

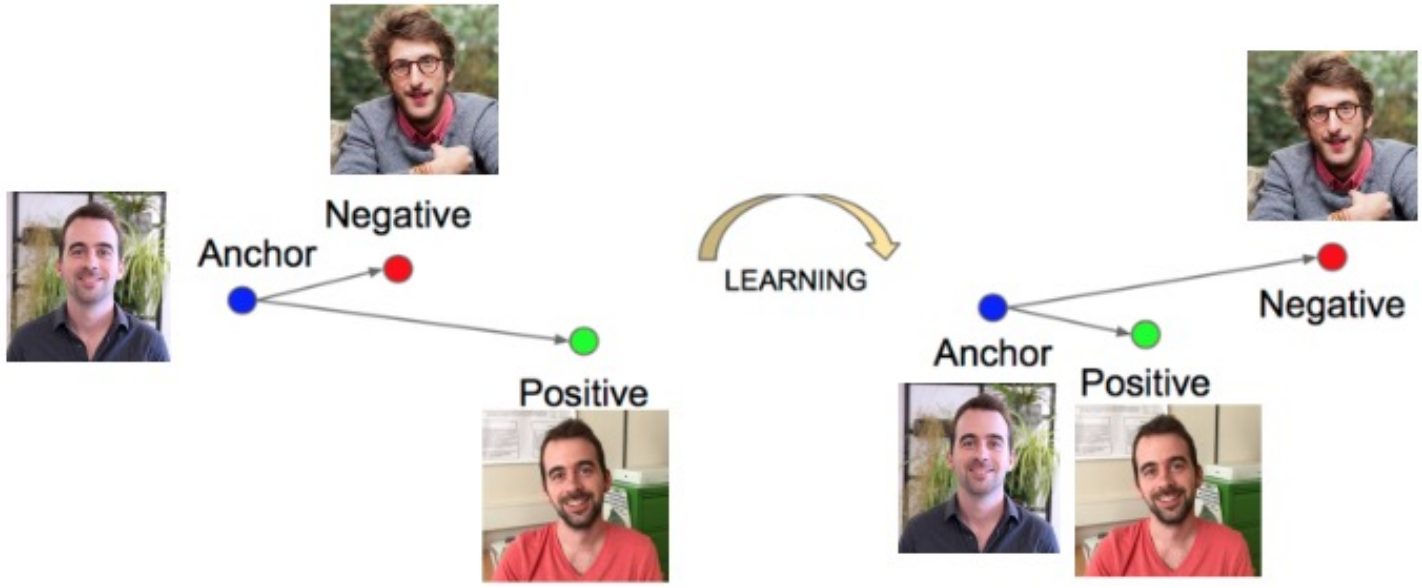
Face Recognition: DL Approach

- **Siamese Network:** compute the face features automatically
  - Use the **same normal CNN** for every person picture
  - Goal: In the below we want to have the following condition:
    - distance(f(x1), f(x2)) is large
    - distance(f(x1), f(x3)) is close to zero
- **Challenge:** how to train such model? (we don't have a lot of positive images)



Face Recognition: DL Approach (cont'd)

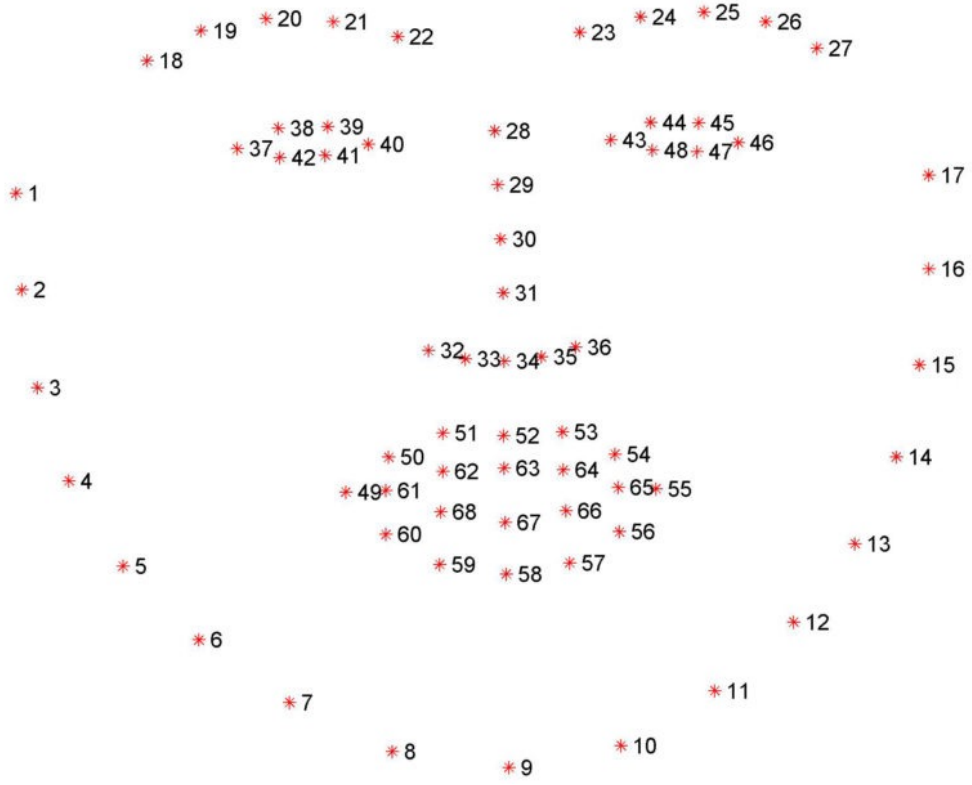
- How to train the Siamese Network:
  - Use triplet loss between 1 anchor image, 1 positive, and 1 negative image
  - Generate training data (triples) as many as we want by replacing the negative image



$$\mathcal{L}\,(Anchor, Positive, Negative) = \max\left(\| \mathrm{f}(Anchor) - \mathrm{f}(Positive)\|^2 - \| \mathrm{f}(Anchor) - \mathrm{f}(Negative)\|^2 + \alpha, 0\right)$$

Facial Landmarks

- One last application closely related to face recognition and detection is the use of facial landmarks.
- **What are facial landmarks?**
  - Specific areas of the face, that allow to get features from the face of anyone.
  - Is historically numbered from 1 to 68
- Can aid neural network models
- Can be used with graph algorithms as well



Summary

- Image Analysis:
  - Limitations of hand-craft features
  - Deep Learning to the rescue and its secrets
- Object Detection
  - Detect multiple objects and label them in the same image
  - Good solution: Sliding Window
  - Best solution: Region Proposal
- Face Detection/Recognition
  - Lack of training data: few images per person (few-shot learning)
  - Solution:
    - Siamese Network
    - Use triple loss
    - Use negative sampling to generate triples of (anchor,negative,positive)

References

[1] https://en.wikipedia.org/wiki/Perceptron

[2] https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

[3] https://sefiks.com/2020/02/02/dance-moves-of-deep-learning-activation-functions/

[4] https://playground.tensorflow.org

[5] https://en.wikipedia.org/wiki/Kernel_(image_processing)

[6] https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[7] https://www.researchgate.net/figure/The-history-of-ImageNet-The-blue-and-red-dot-dash-lines-represent-the-human_fig1_359872504