THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

Machine Learning & Deep Learning – COMP4423 Computer Vision

Xiaoyong Wei (魏驍勇)

x1wei@polyu.edu.hk

Department of Computing
電子計算學系

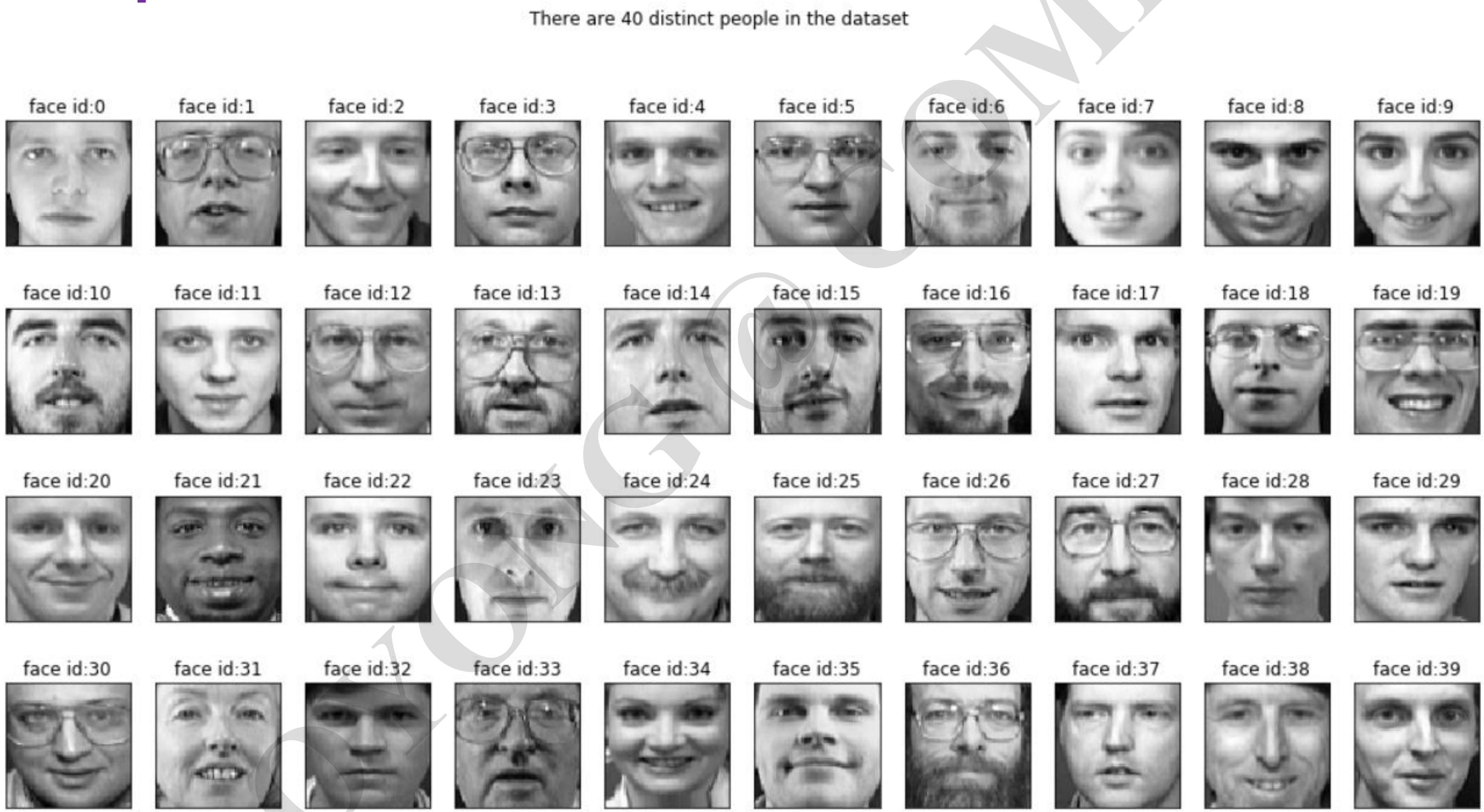
Opening Minds • Shaping the Future
啟迪思維 • 成就未來

Outline

>Traditional machine learning vs. deep learning
>Gradient decent
>Neural networks
>Deep neural networks
>Convolutional neural networks (CNN)
>Layers, pooling, and activation
>AlexNet, VGG, and ResNet

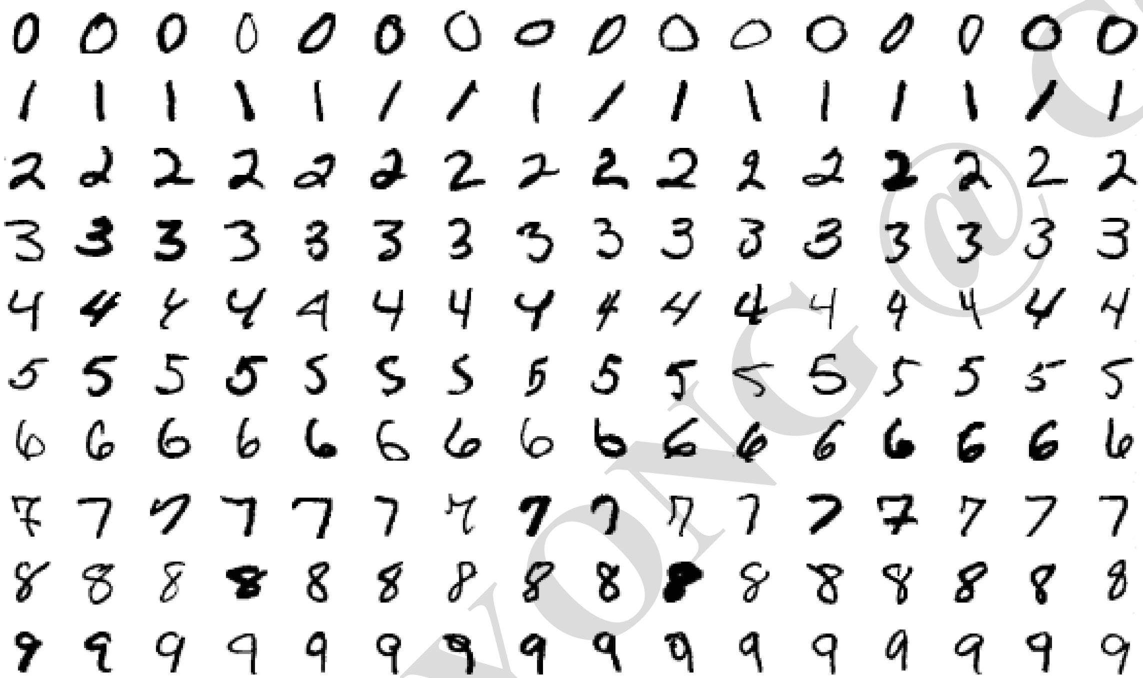
Traditional classification methods work well for simple tasks. Models are usually built in a controlled environment (e.g., lab setting) to eliminate the variations of illumination, viewpoints, scales, and so on.

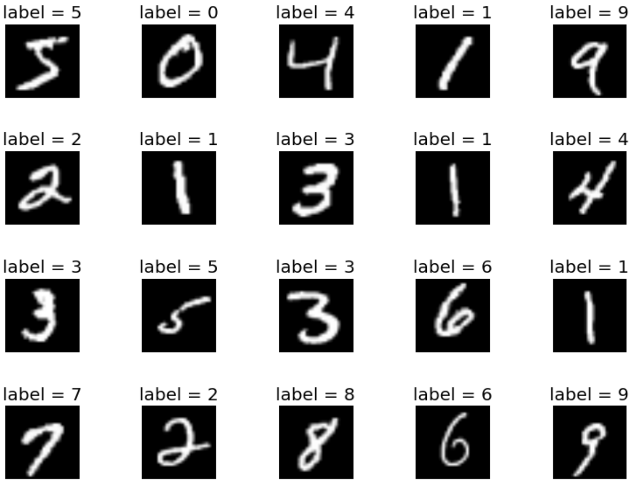
Popular traditional datasets



Olivetti Face Dataset, AT&T

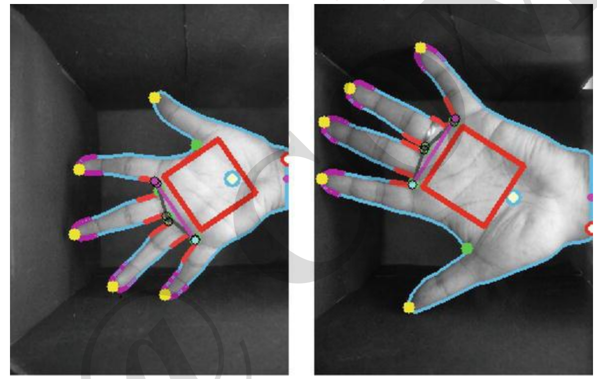
Popular traditional datasets

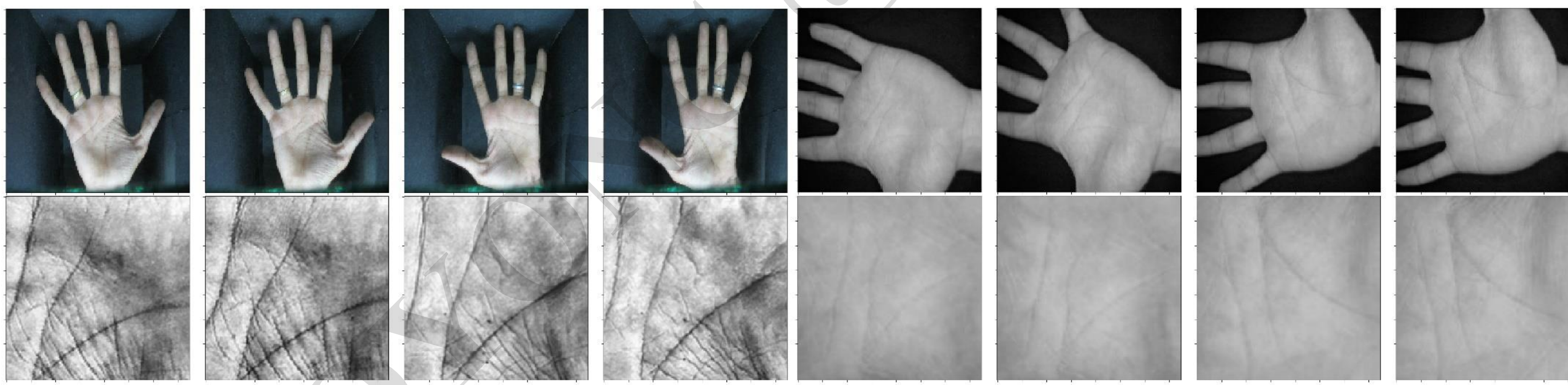




MINIST Handwritten Digits

Popular traditional datasets

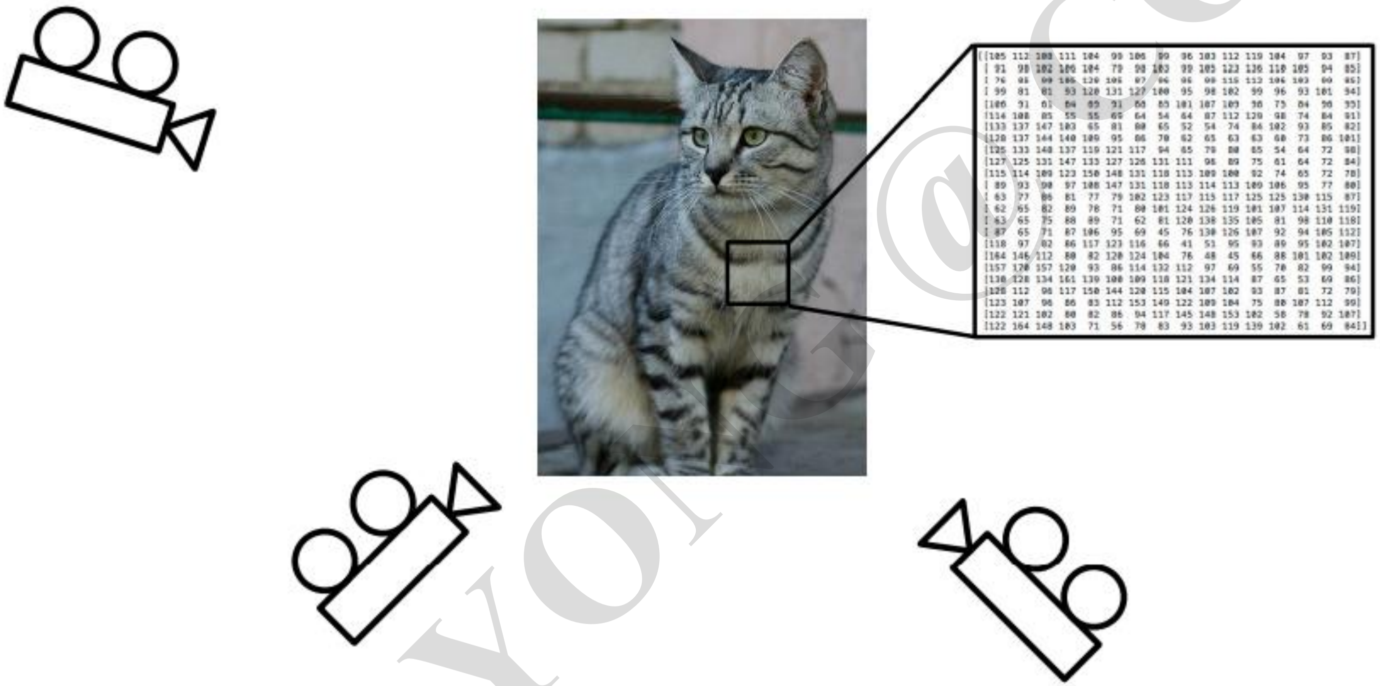




Palmprint Acquisition and Datasets

However, in real applications, those are inevitable.

Viewpoints



All pixels change when the camera moves!

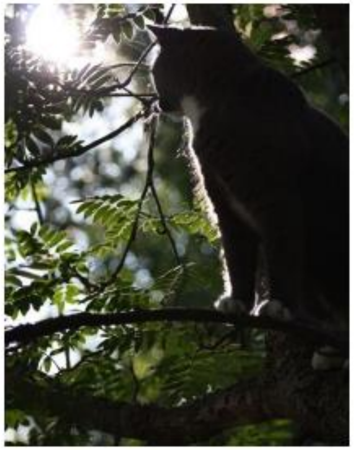
Fei-Fei Li, Ranjay Krishna, Danfei Xu, Image Classification: A Core Task in Computer Vision

Illumination

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain



This image is CC0 1.0 public domain

Fei-Fei Li, Ranjay Krishna, Danfei Xu, Image Classification: A Core Task in Computer Vision

Occlusions

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

This image by jonsson is licensed under CC-BY 2.0

Fei-Fei Li, Ranjay Krishna, Danfei Xu, Image Classification: A Core Task in Computer Vision

Background Clutter

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

Fei-Fei Li, Ranjay Krishna, Danfei Xu, Image Classification: A Core Task in Computer Vision

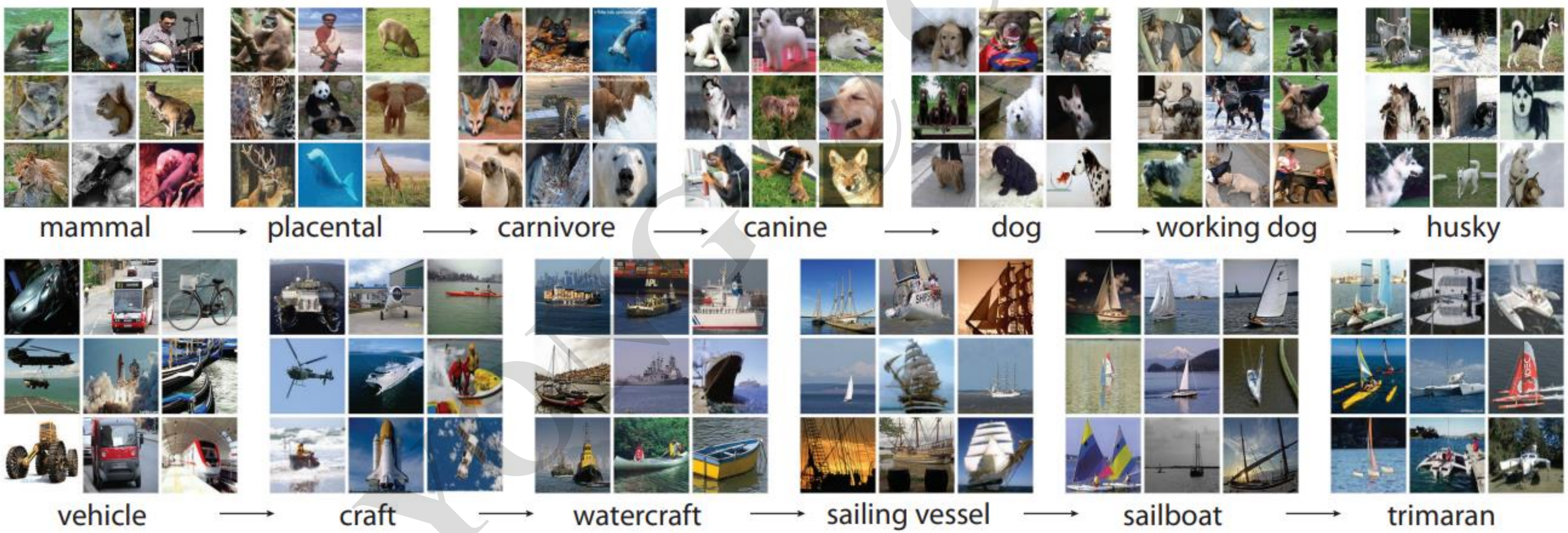
Intra-class Variations

Fei-Fei Li, Ranjay Krishna, Danfei Xu, Image Classification: A Core Task in Computer Vision

Hand Gesture Recognition

ImageNet

ImageNet: 12 subtrees with 5247 synsets and 3.2 million images in total

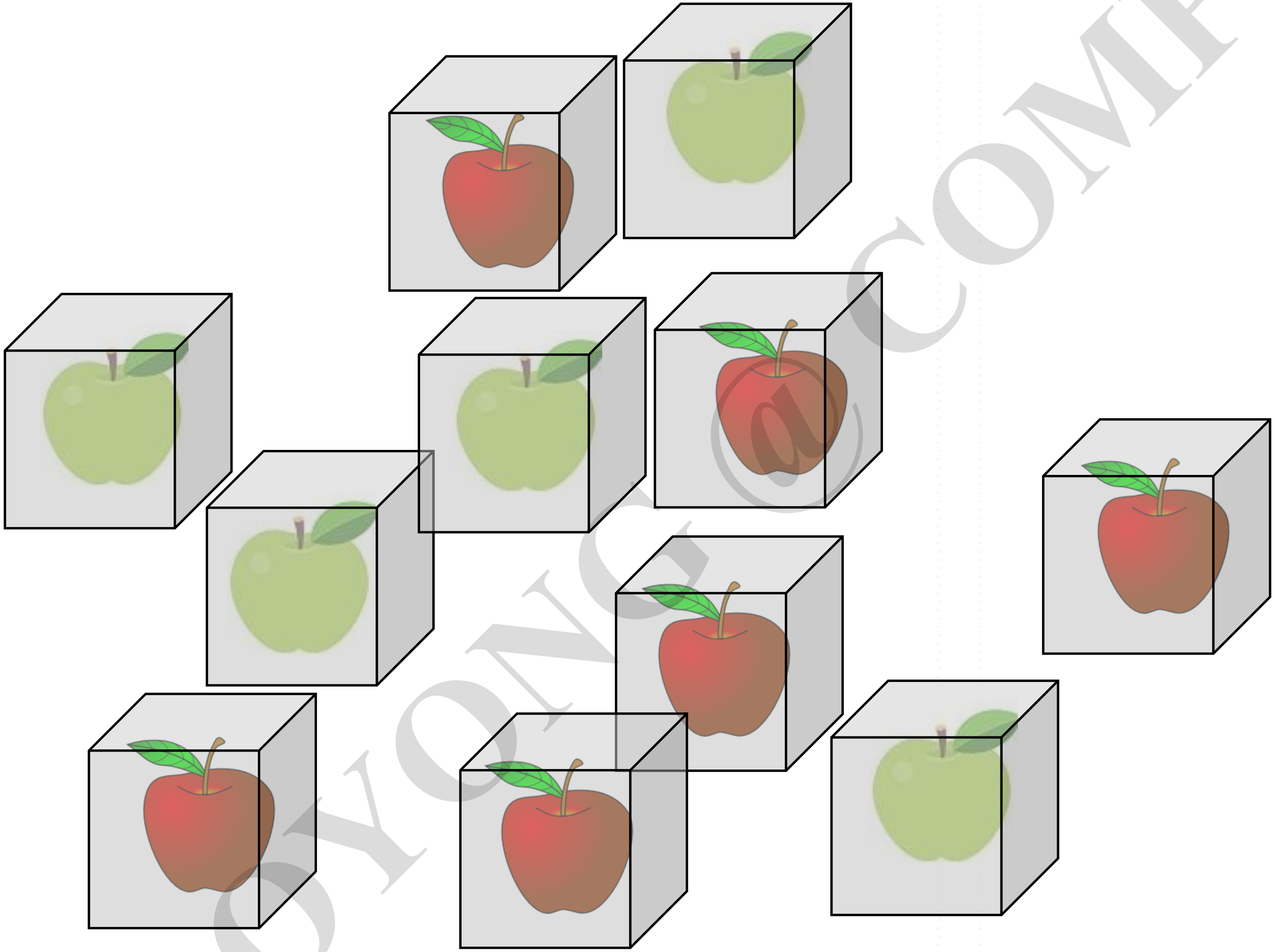


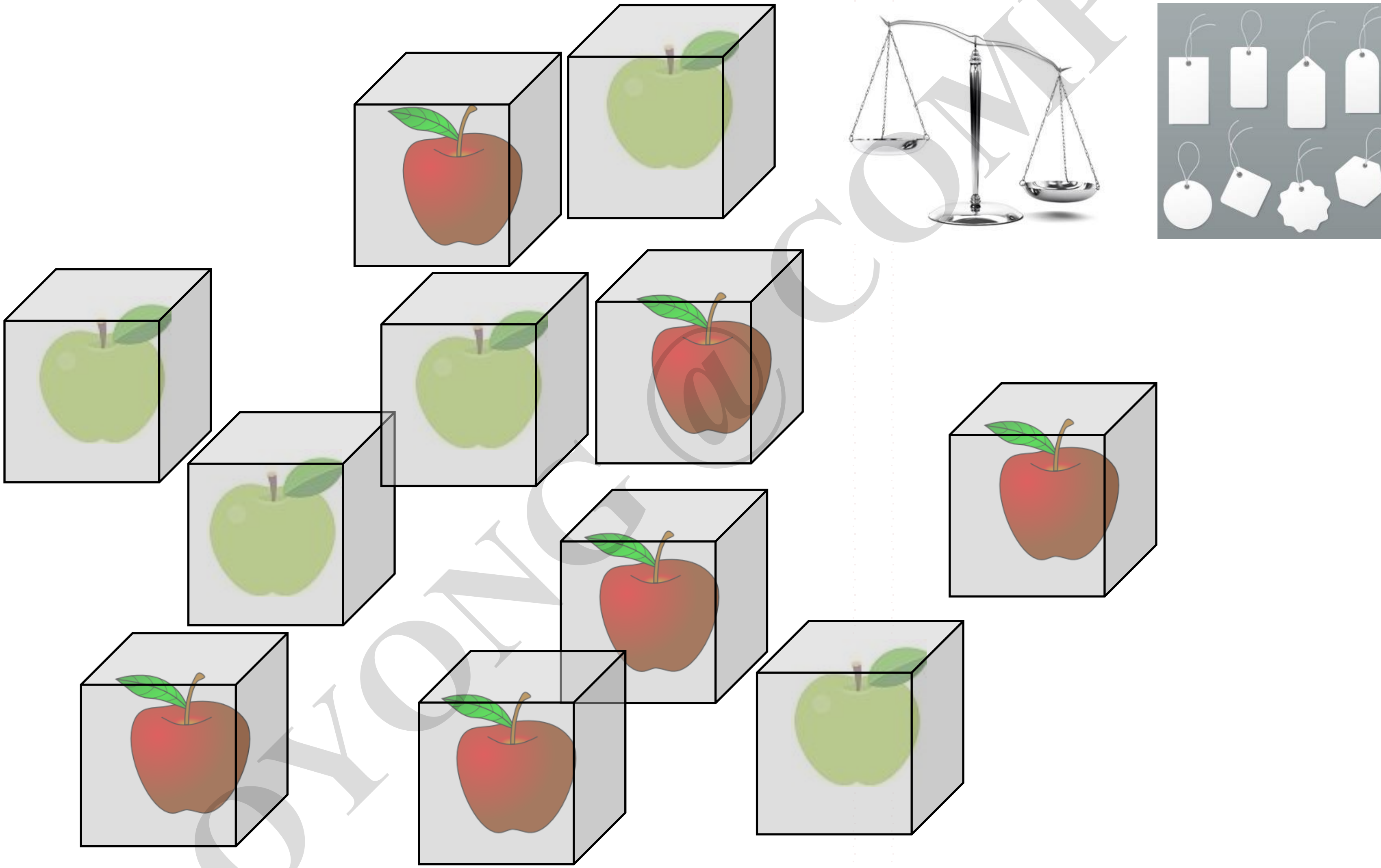
J. Deng, W. Dong, R. Socher, L. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

Deep Learning is a popular solution to address these challenges.

(This is what you're waiting for. LOL!)

Let’s start by reviewing the learning of decision boundary through an example – to classify the red and green apples.

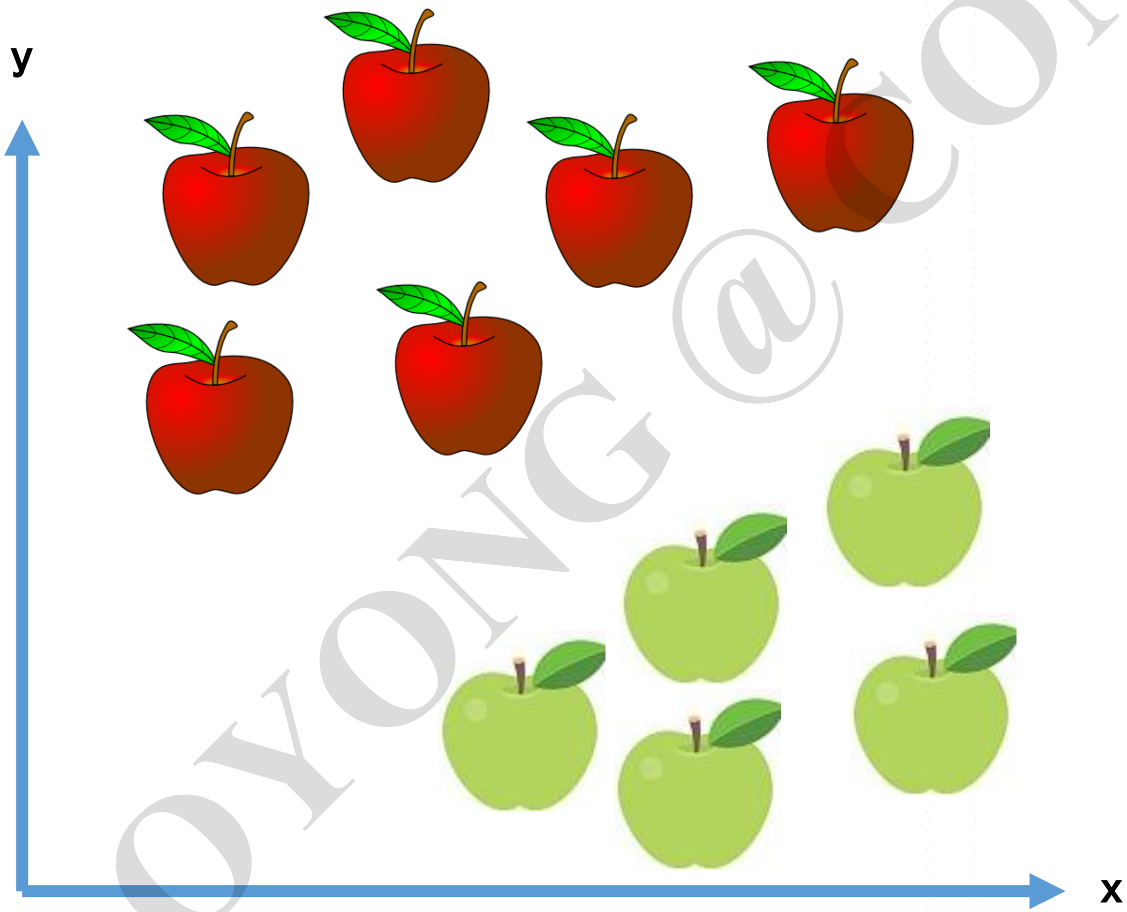




List of apples

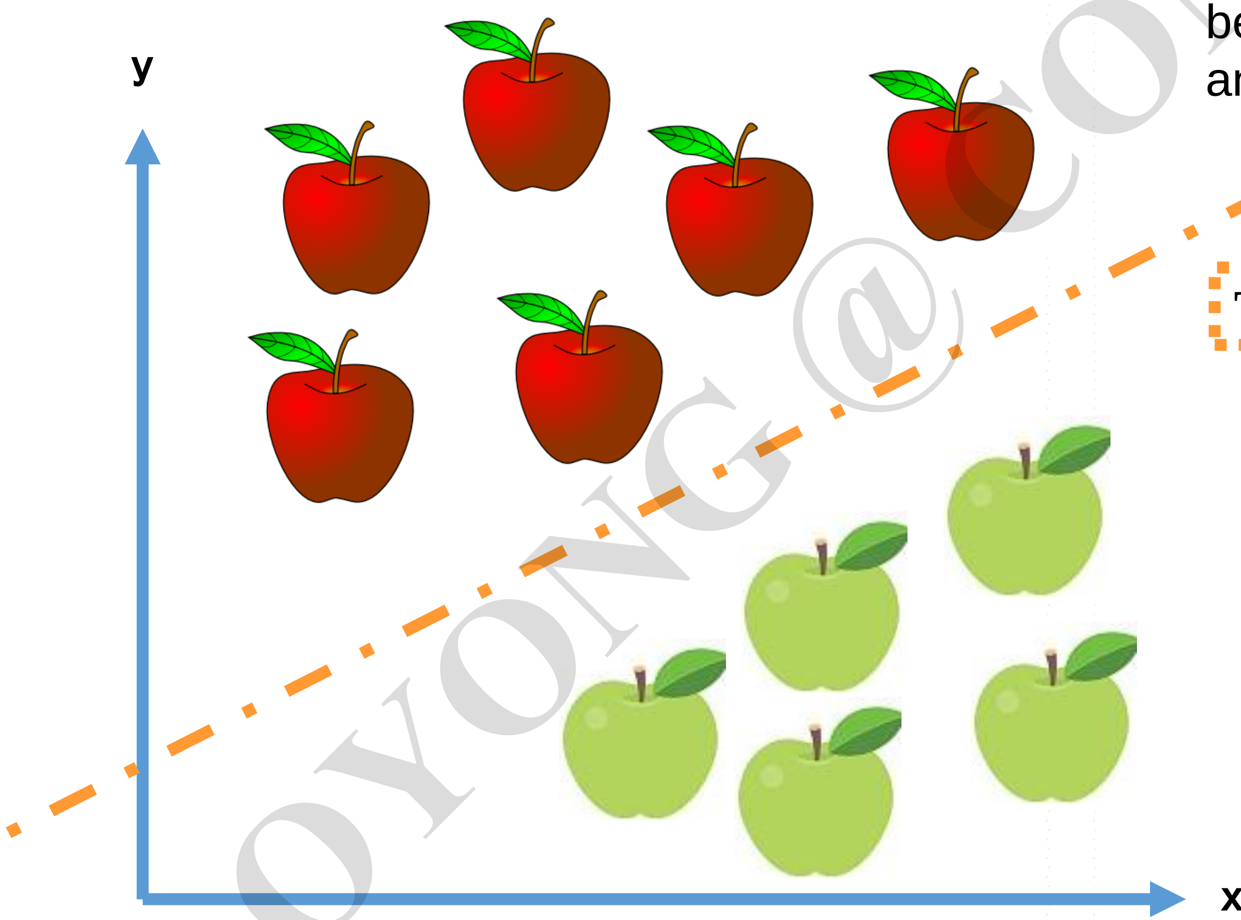
No.	x	y	others	Color (z)
1	9	53	...	Red (1)
2	25	45	...	Green (-1)
3	225	56.7	...	Red (1)
4	576	52.9	...	Green (-1)
5	676	60.2	...	Red (1)
6	900	55.7	...	Green (-1)
7	...	...	...	...

Apple Space



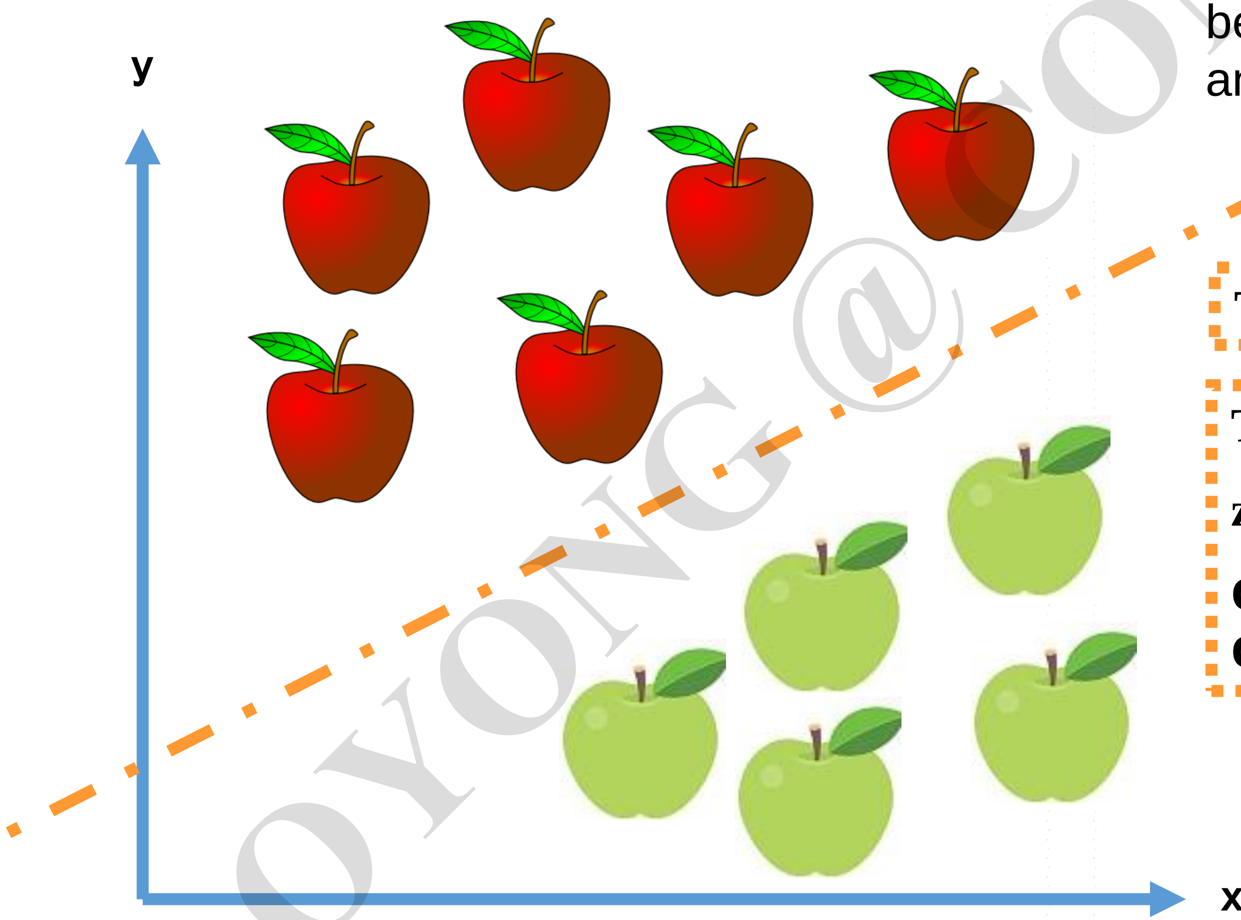
Apple Space

To find the best line dividing the two groups of apples is to find the best **parameters** of **a** and **b**



The line: y =a*x+b

Apple Space



To find the best line dividing the two groups of apples is to find the best **parameters** of **a** and **b**

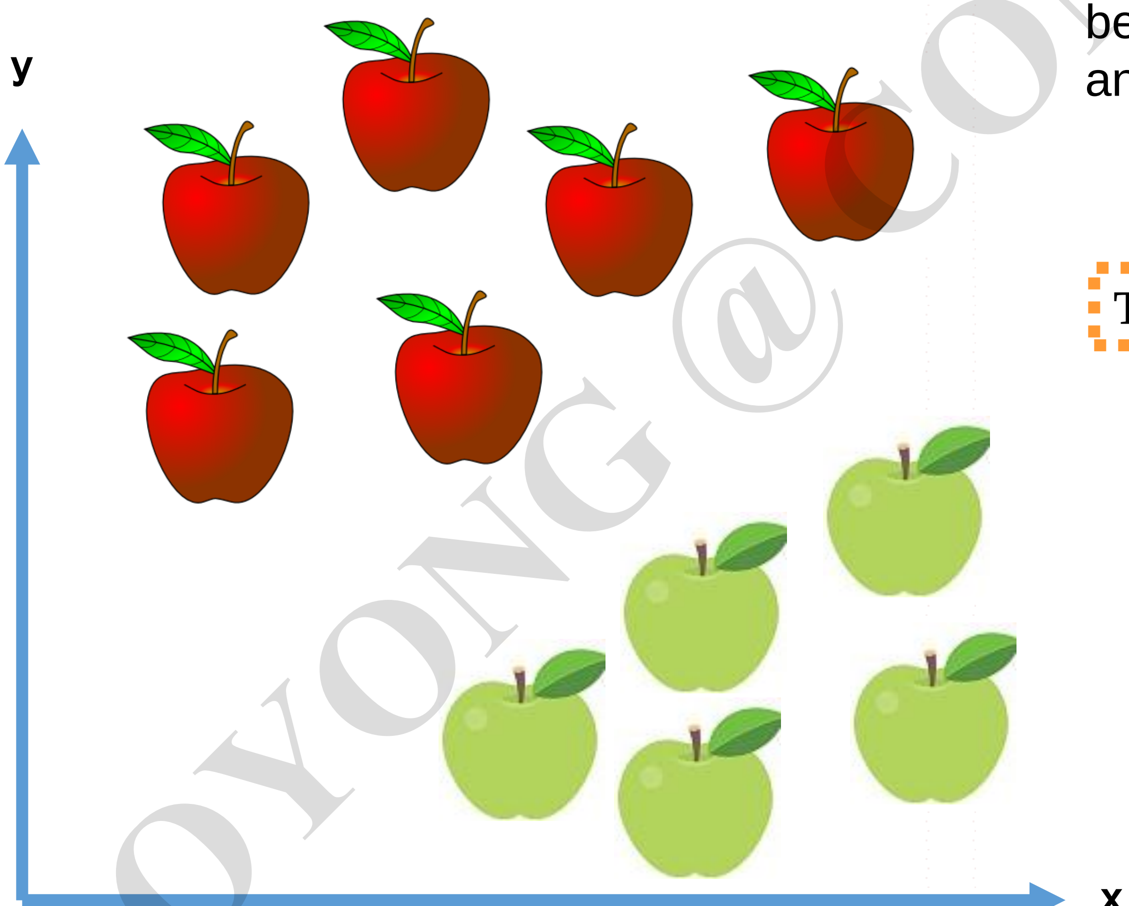
The line: y =a*x+b

The model:

z=a*x-y+b

Outputs 1 if z>0
Outputs -1 if z<=0

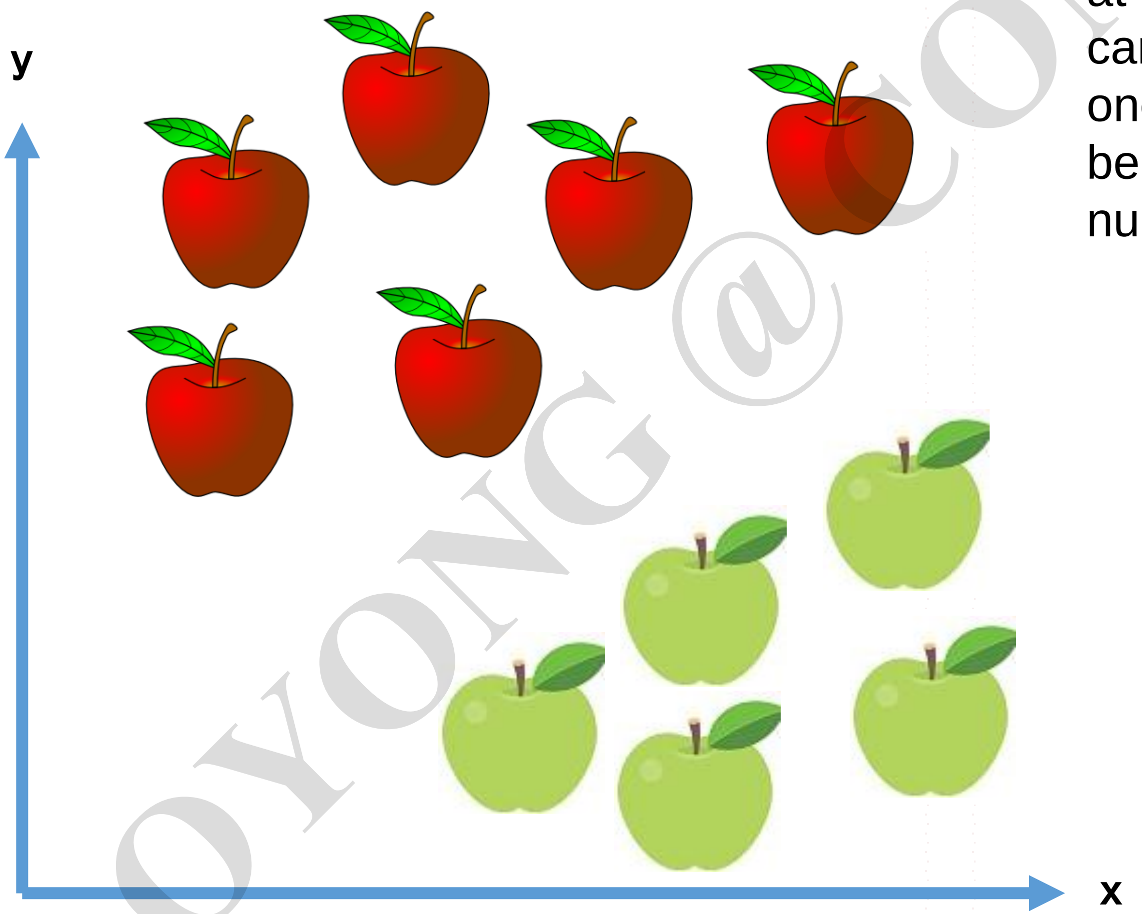
Apple Space



To find the best line dividing the two groups of apples is to find the best **parameters** of **a** and **b**

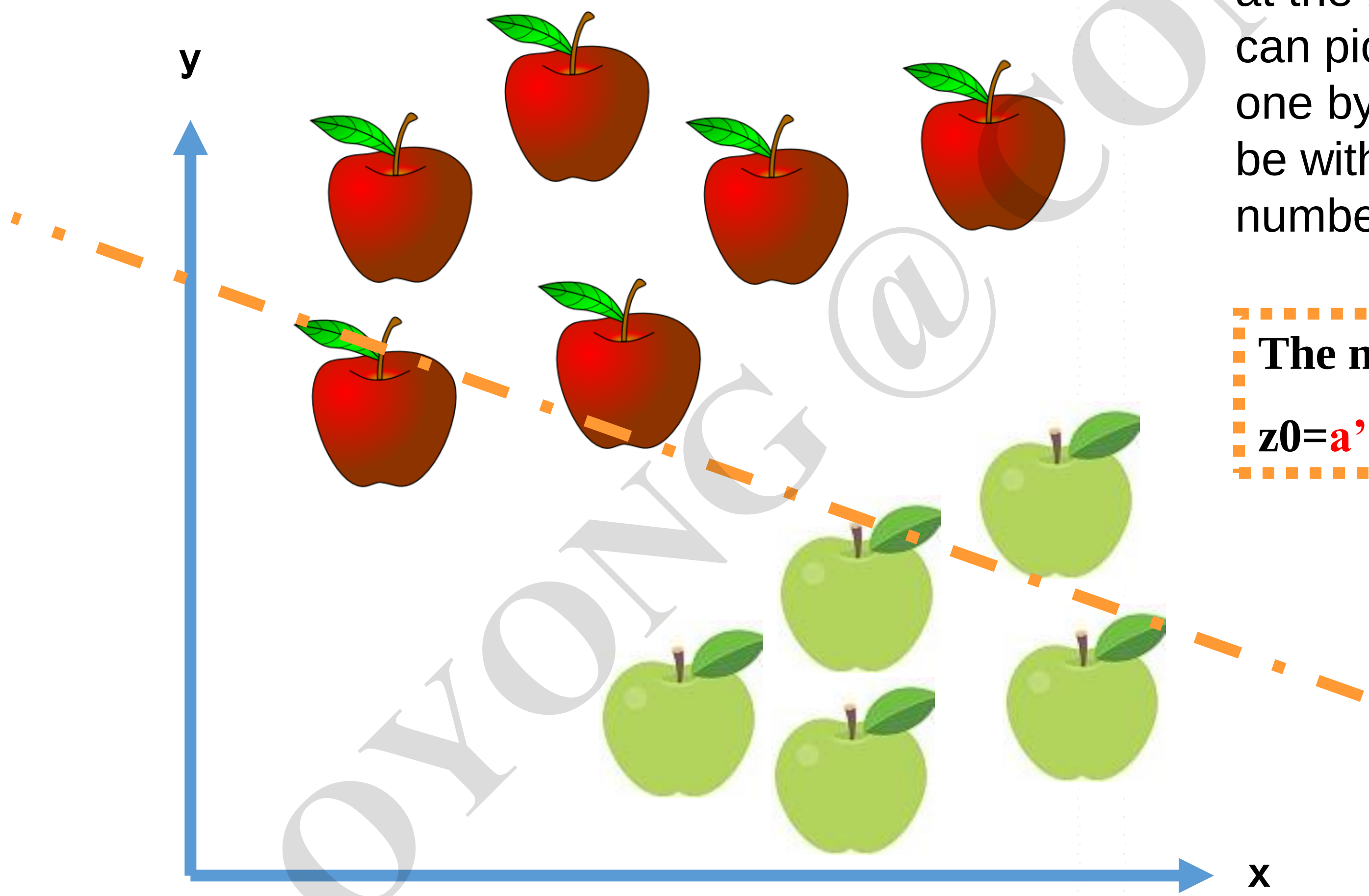
The line: y =a*x+b

Apple Space



Initialization
Without knowing which line is the best at the beginning, we can pick a random one by setting a and be with random numbers a’ and b’.

Apple Space



Initialization
Without knowing which line is the best at the beginning, we can pick a random one by setting a and be with random numbers a' and b'.

The model:

$z0=a'*x-y+b'$

How can we evaluate how good the **model** (a' and b') is?

Intuitively, we can compare the **prediction** z' to the **ground truth label** z using **(z'-z)^2**. By applying to all N samples, we have a **loss function**

$$L(a', b') = \frac{1}{N}\sum_{i=1}^{N}(z'_i - z_i)^2$$

With the “goodness” evaluated, we can update a’ and b’ by replacing them with better ones.

The updating process is so called **learning**.

But, how?

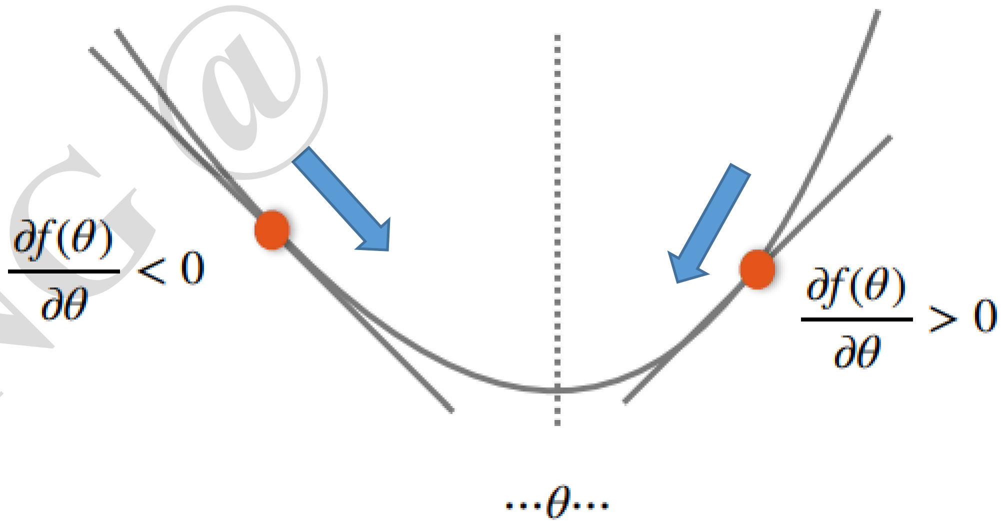
The best parameters are the ones that minimize the loss function L. The optimal parameters can thus be found at where the **gradients** of L are zeros

$$\frac{\partial L}{\partial a} = 0, \frac{\partial L}{\partial b} = 0.$$

We can update a' and b' by pushing the gradients towards zeros!

$$a' = a' - \frac{\partial L}{\partial a'}$$

$$b' = b' - \frac{\partial L}{\partial b'}$$



Learning Rate

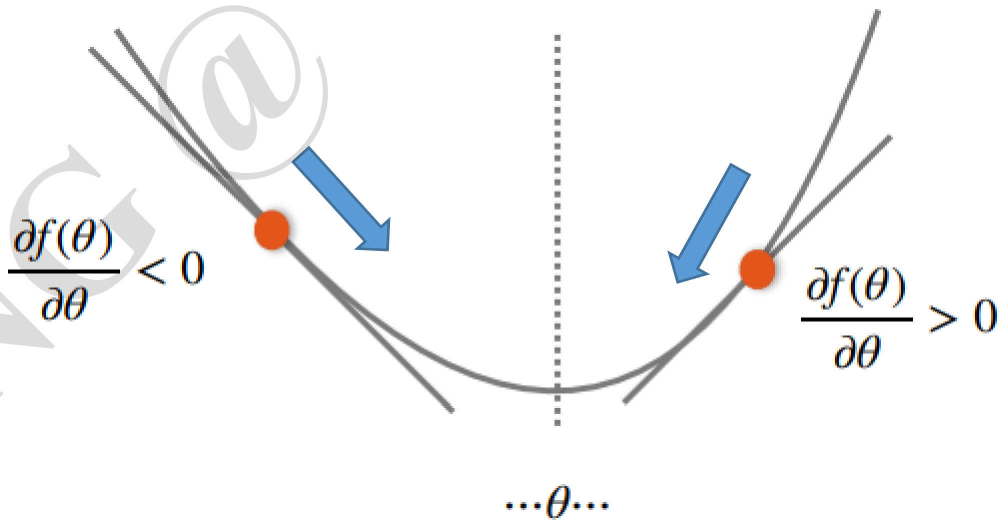
Gradient Decent

We can update a' and b' by pushing the gradients towards zeros!

$$a' = a' - \delta \frac{\partial L}{\partial a'}$$

$$b' = b' - \delta \frac{\partial L}{\partial b'}$$

Learning Rate



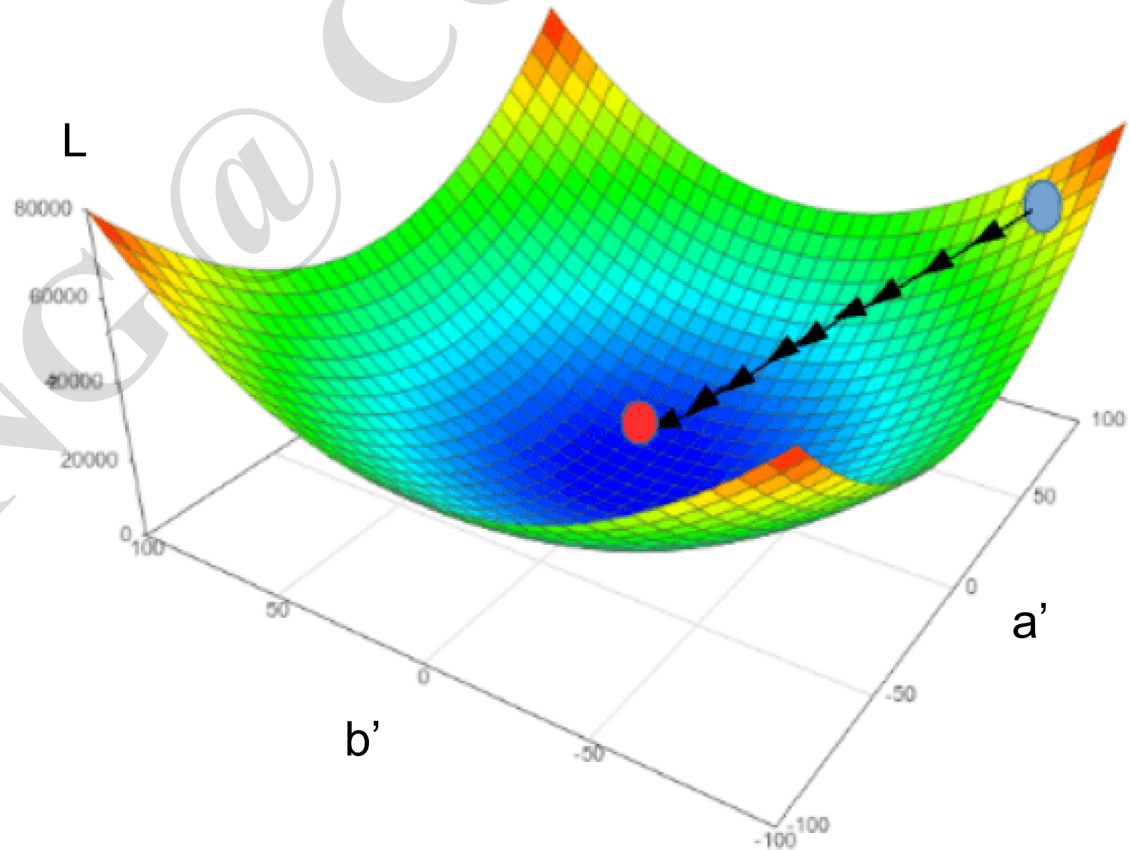
Gradient Decent

We can update a' and b' by pushing the gradients towards zeros!

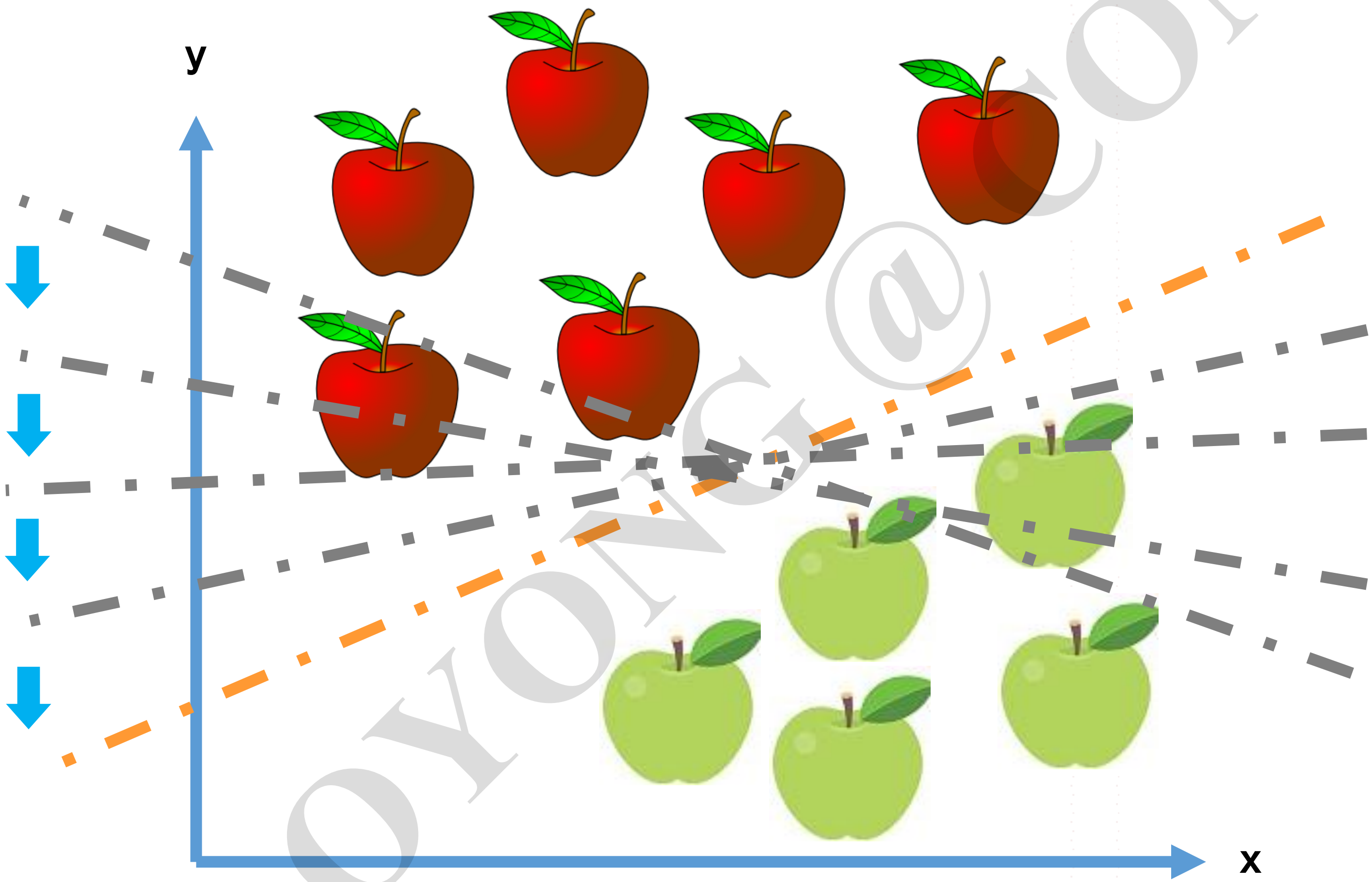
$$a' = a' - \delta \frac{\partial L}{\partial a'}$$

$$b' = b' - \delta \frac{\partial L}{\partial b'}$$

Learning Rate



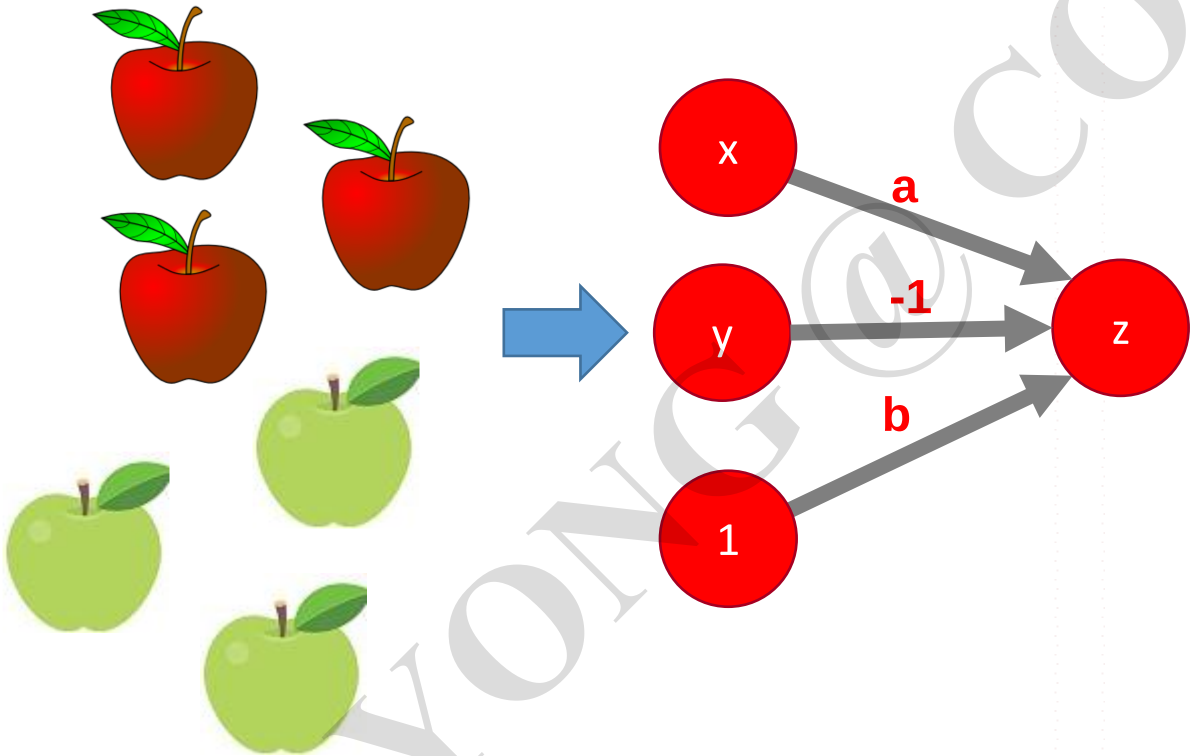
Apple Space



Machine learning is a process to find the best set of **parameters** that fits into a **model/hypothesis**. The learning is usually conducted by updating the initial parameters with a learning rate towards the optimal of a **loss function**. **Gradient Decent** is one of the most popular updating strategies.

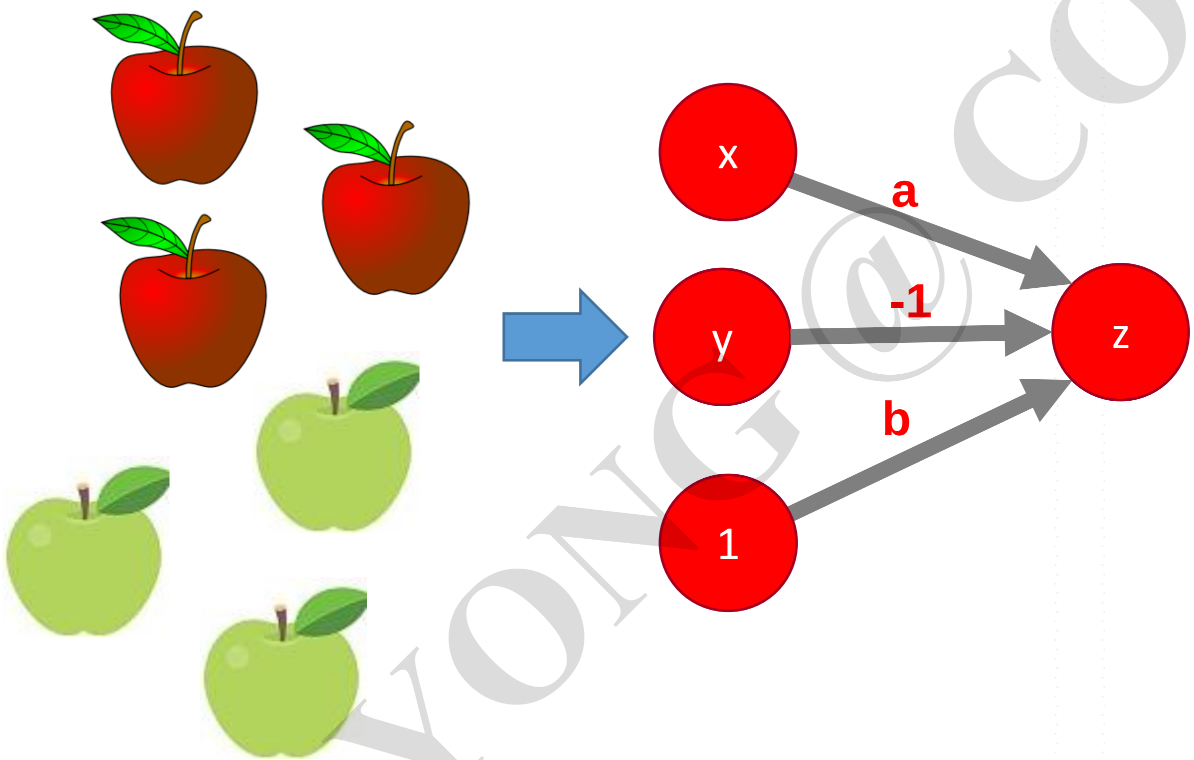
Let's implement the learning using **neural networks**.

Neural Network Version of the Model



z=a*x−y+b
1: if z>0
−1: if z<=0

Neural Network Version of the Model

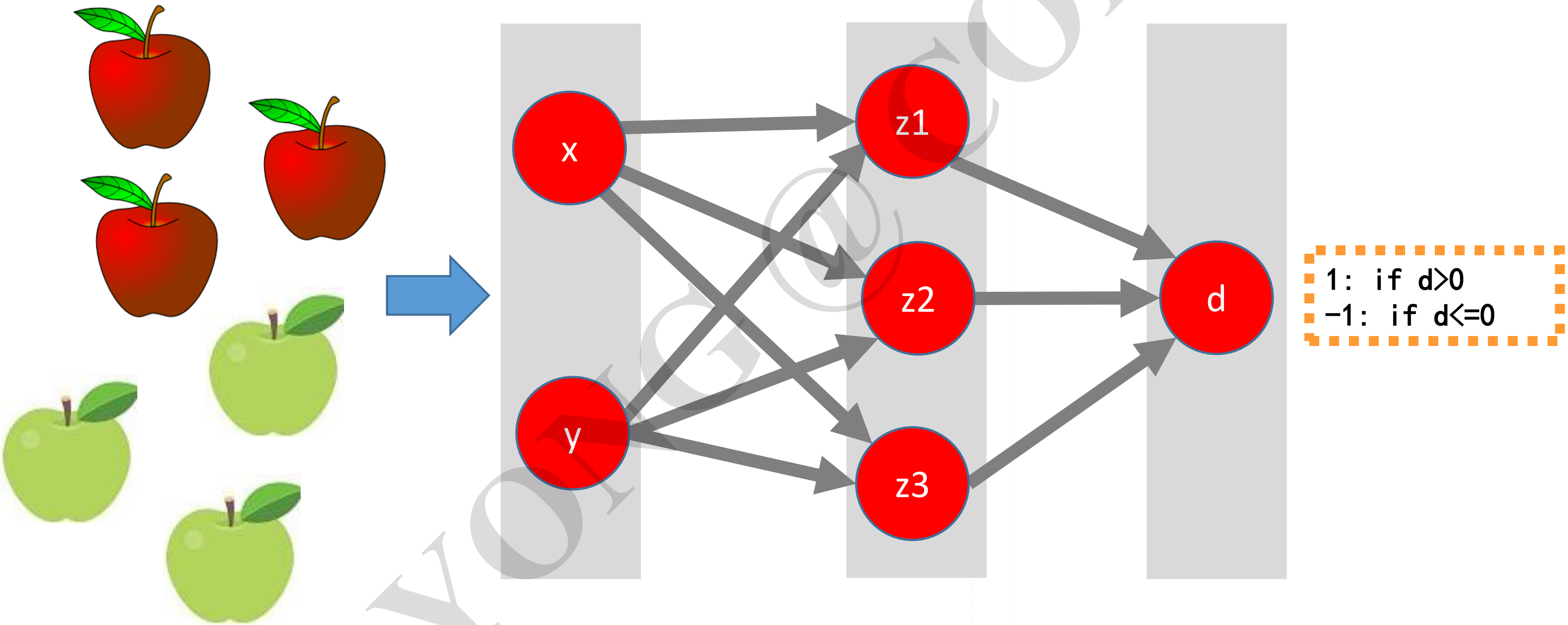


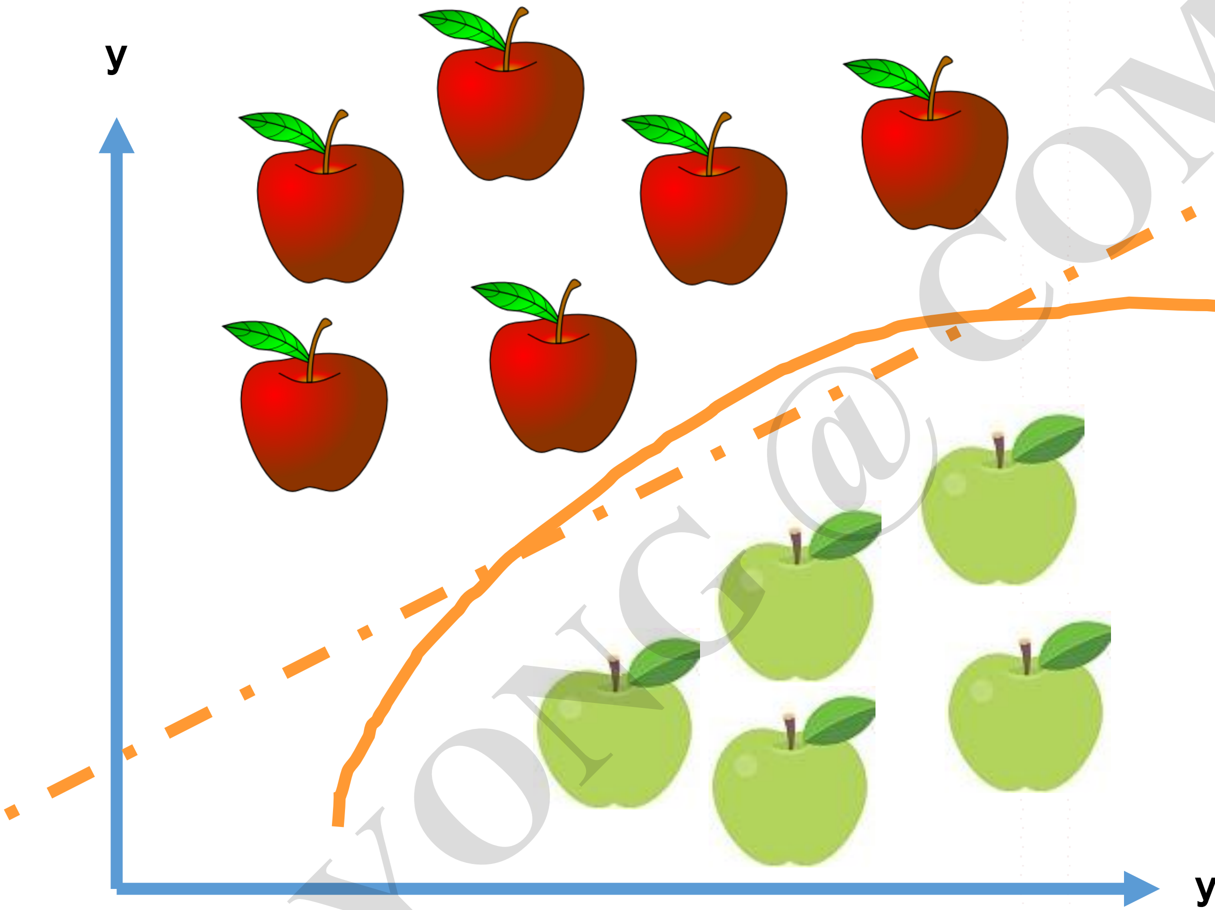
$z = a * x - y + b$

1: if $z > 0$

-1: if $z <= 0$

Neural Network Version of the Model

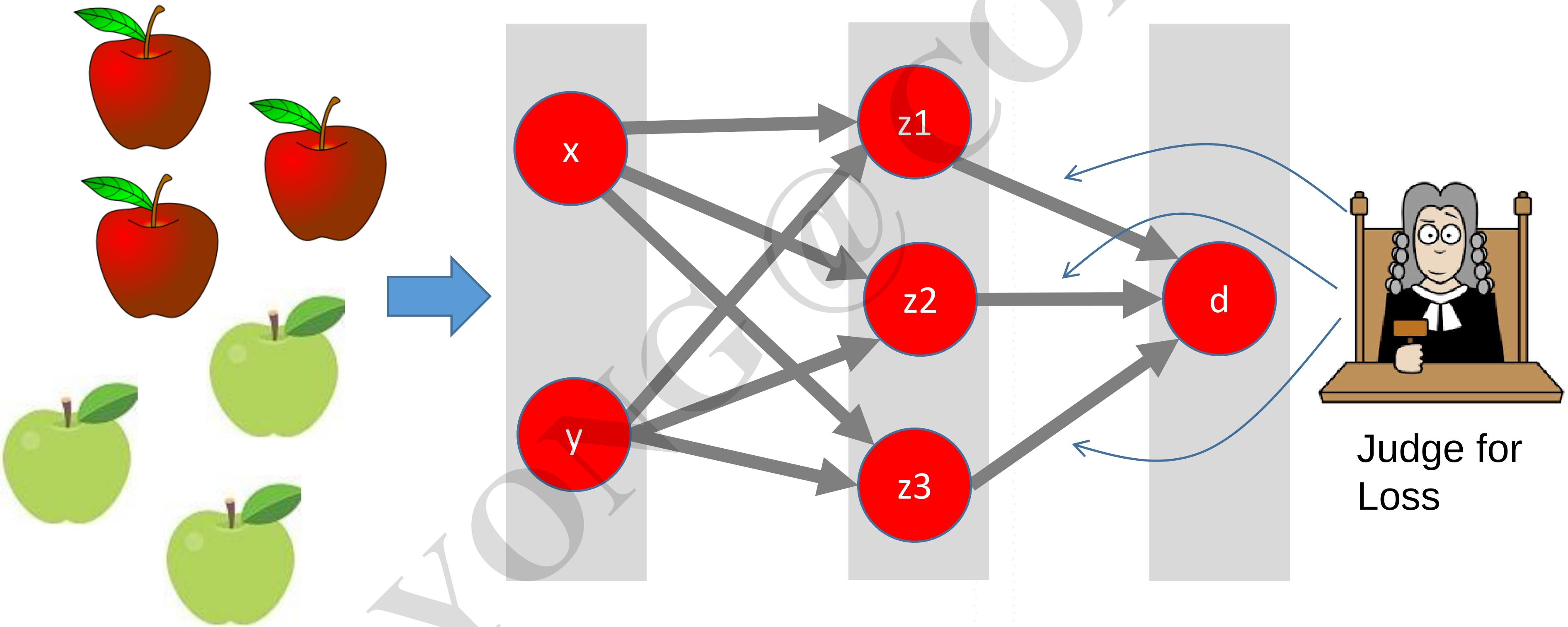




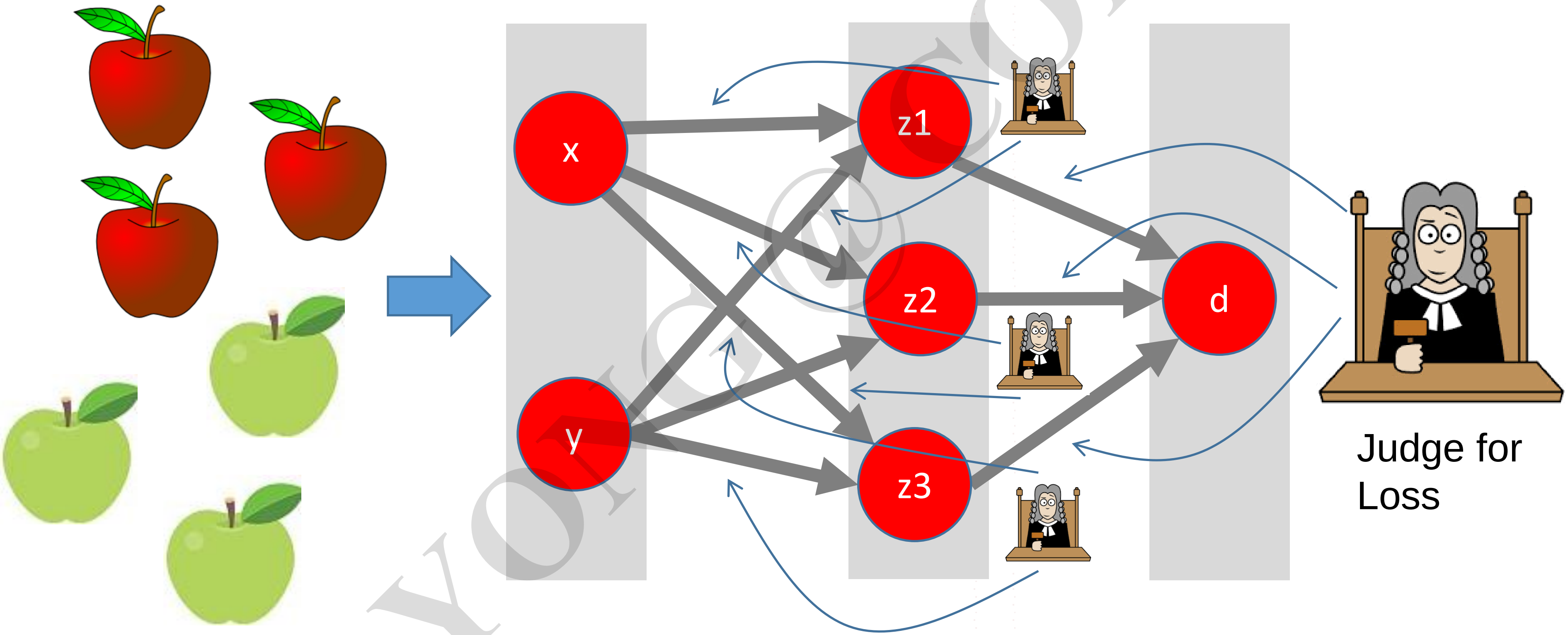
The new decision boundary

How is the learning conducted with more layers and weights?

Gradient Decent on Neural Networks

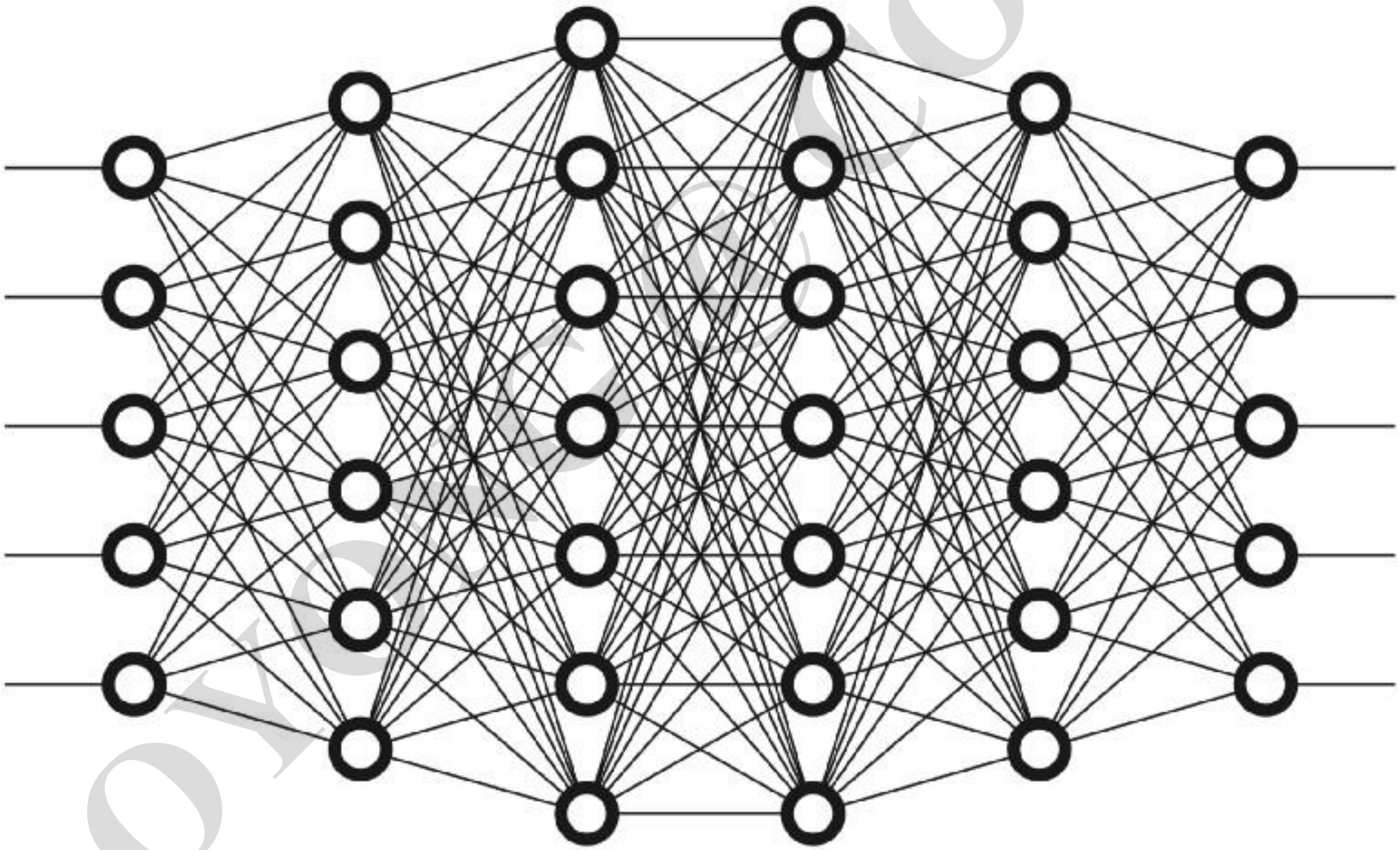


Gradient Decent on Neural Networks

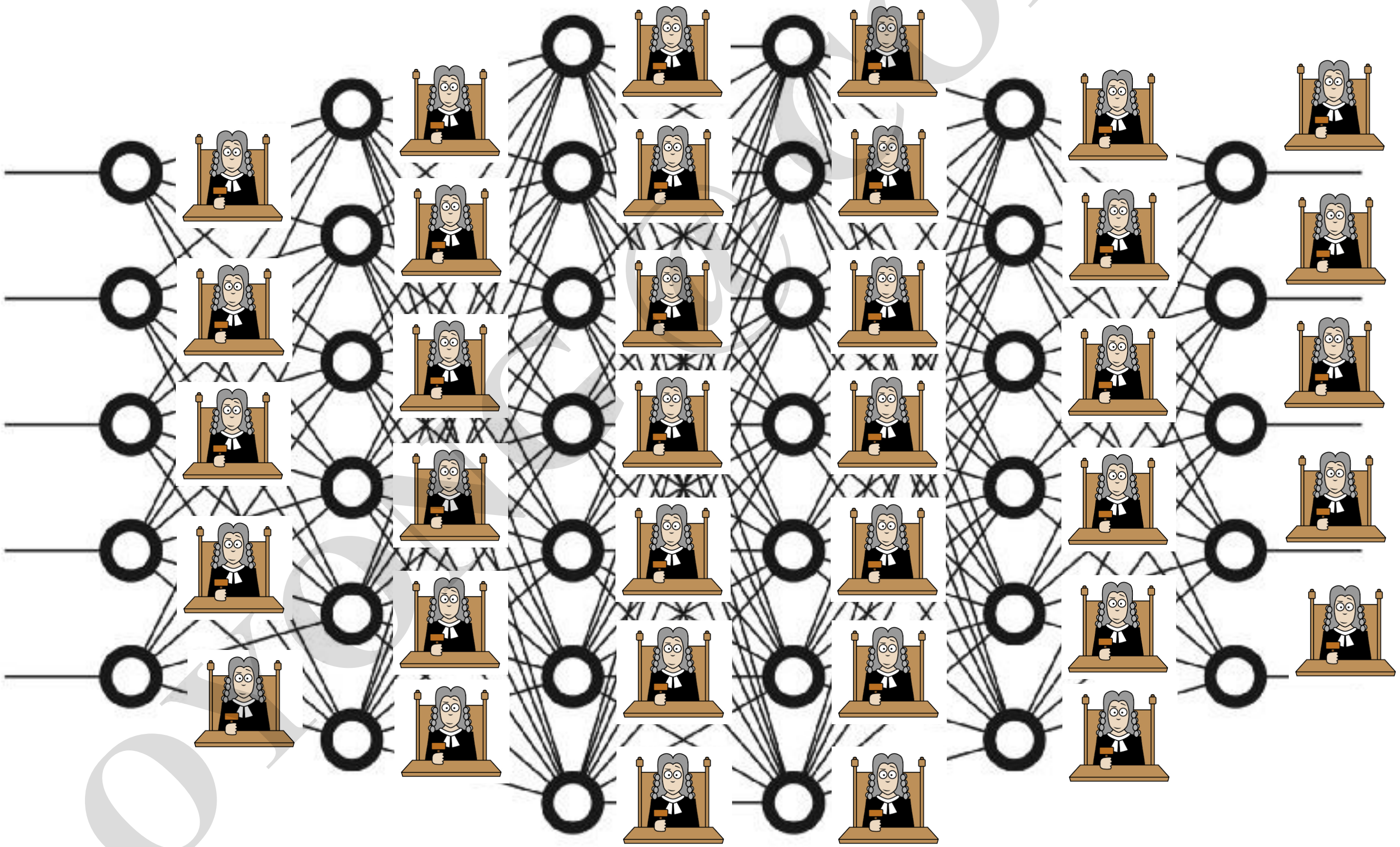


The Chain Rule for Backpropagation

By stacking more layers you have **Deep Neural Networks**

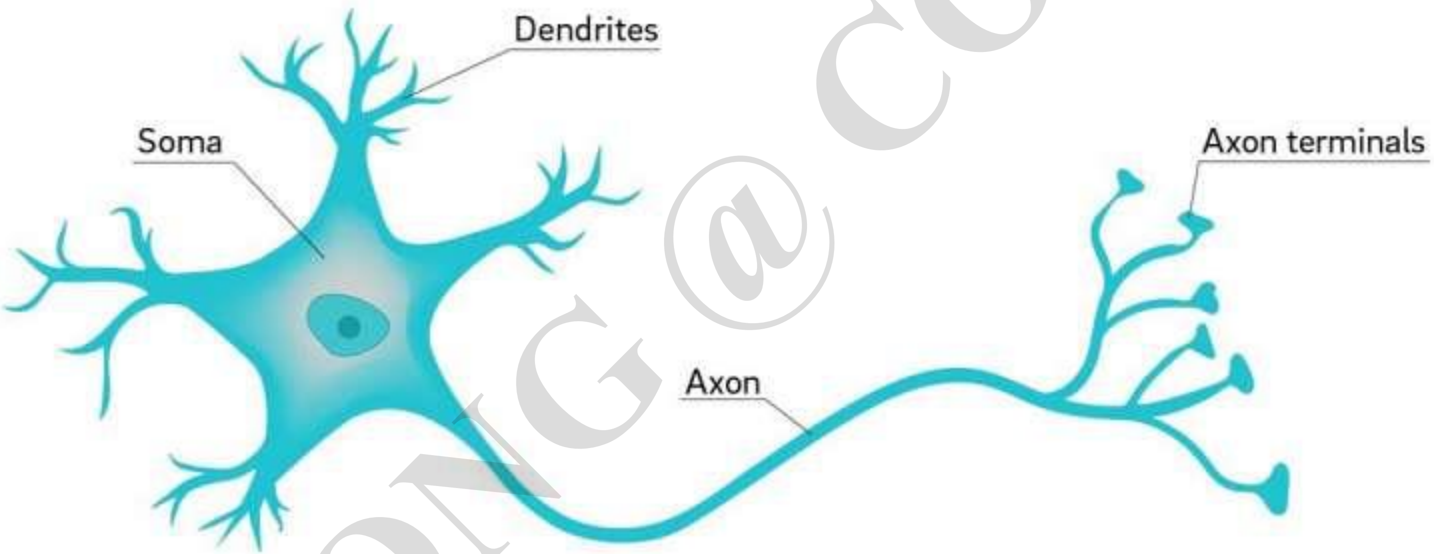


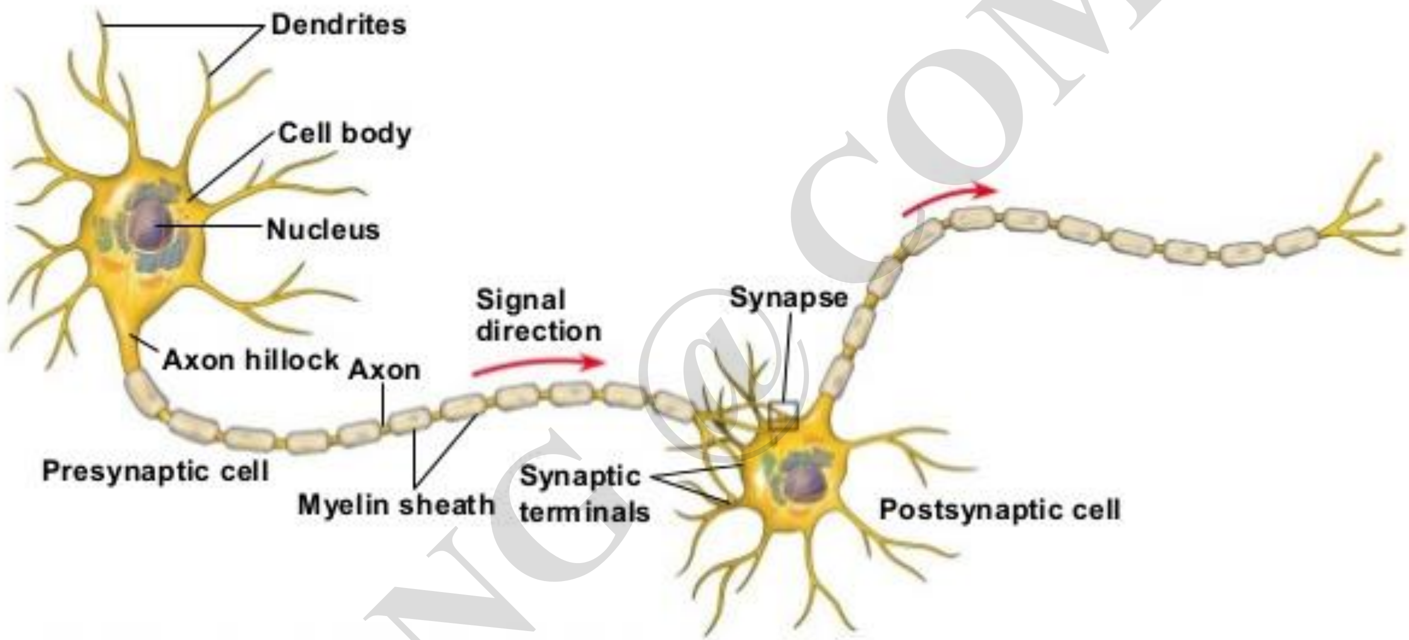
By employing more loss judges you have **Deep Learning**

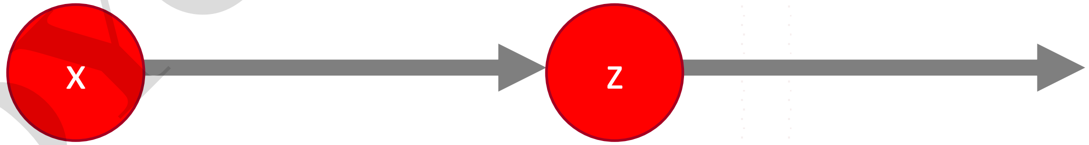


Now, we're ready for a few more concepts (tricks)?

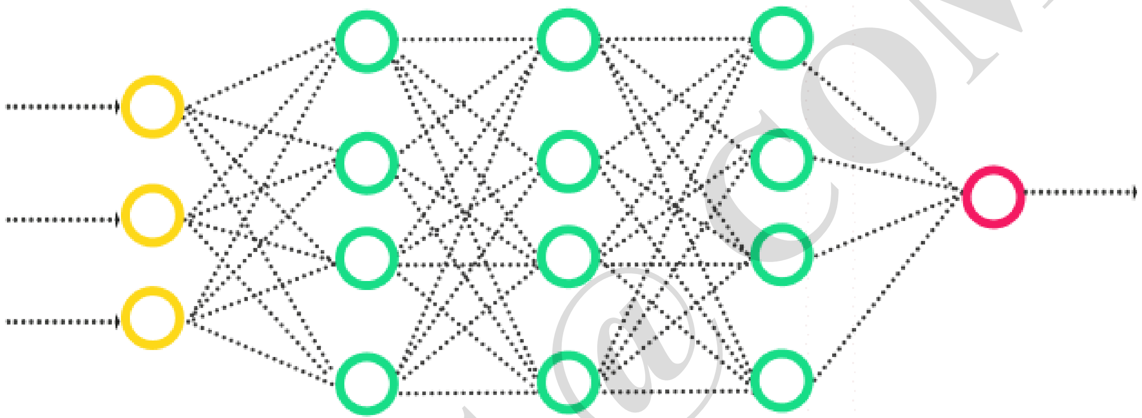
Neuron




Dendrites
Cell body
Nucleus
Signal direction
Synapse
Axon hillock
Axon
Presynaptic cell
Myelin sheath
Synaptic terminals
Postsynaptic cell


X
Z

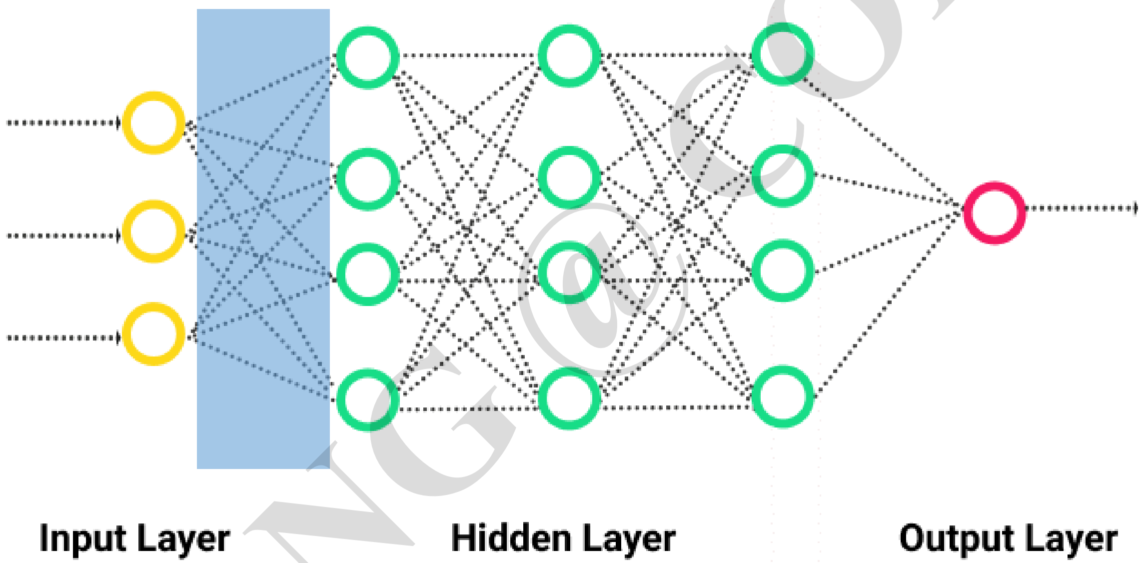
Layers



Input layer : Receive data from external sources (data files, images, sensors, etc.)
Hidden layers : process data
Output layer provides network-based functions for one or more data points

https://ipullrank.com/resources/guides-ebooks/machine-learning-guide/chapter-1

Convolutional Layers

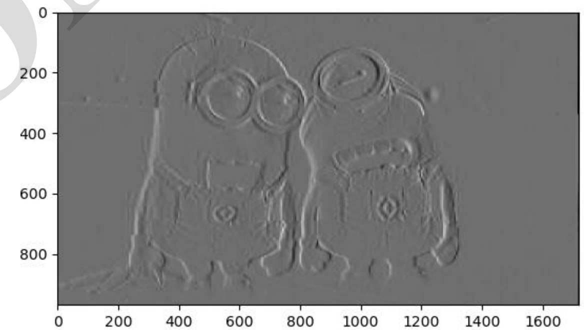


Instead of using fully connected layers, we can add a few more "partially" connected layers.

Recall the Filters and Convolutions

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \qquad G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

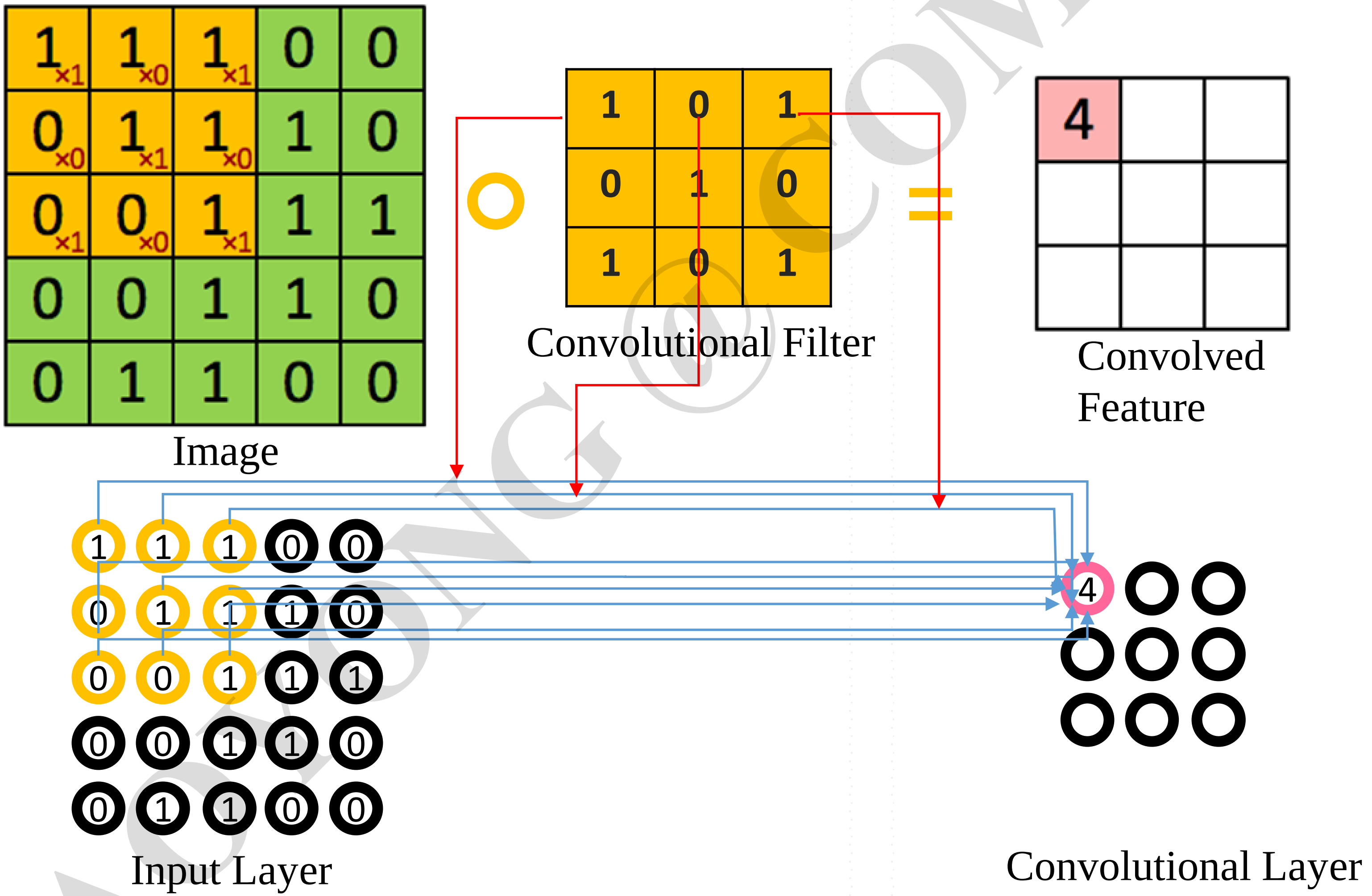
Prewitt filter for vetical edge detection **Prewitt filter for horizontal edge detection**



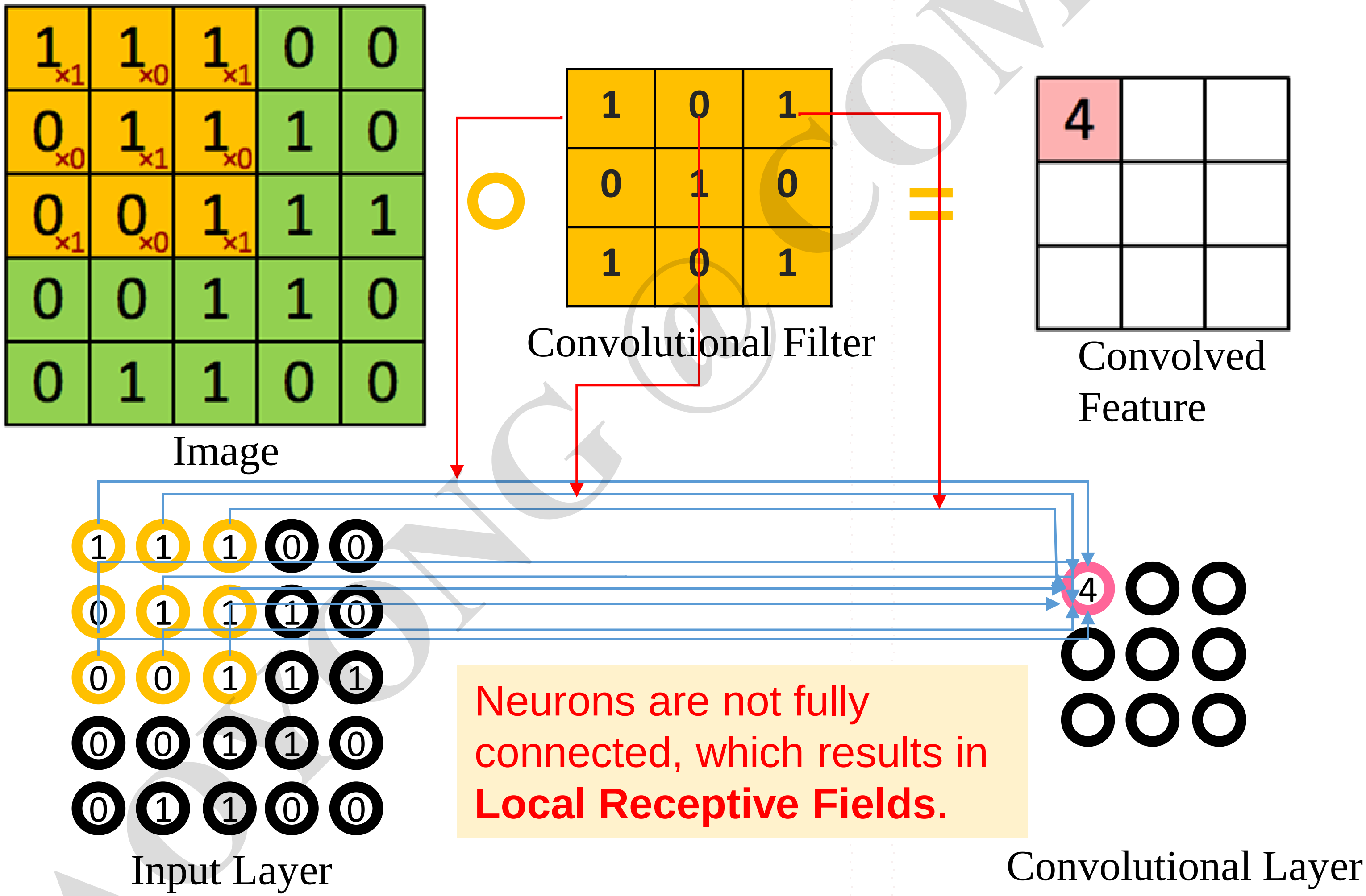
-1	0	1
-2	0	2
-1	0	1

Sauber Filter

Implement with Neural Networks



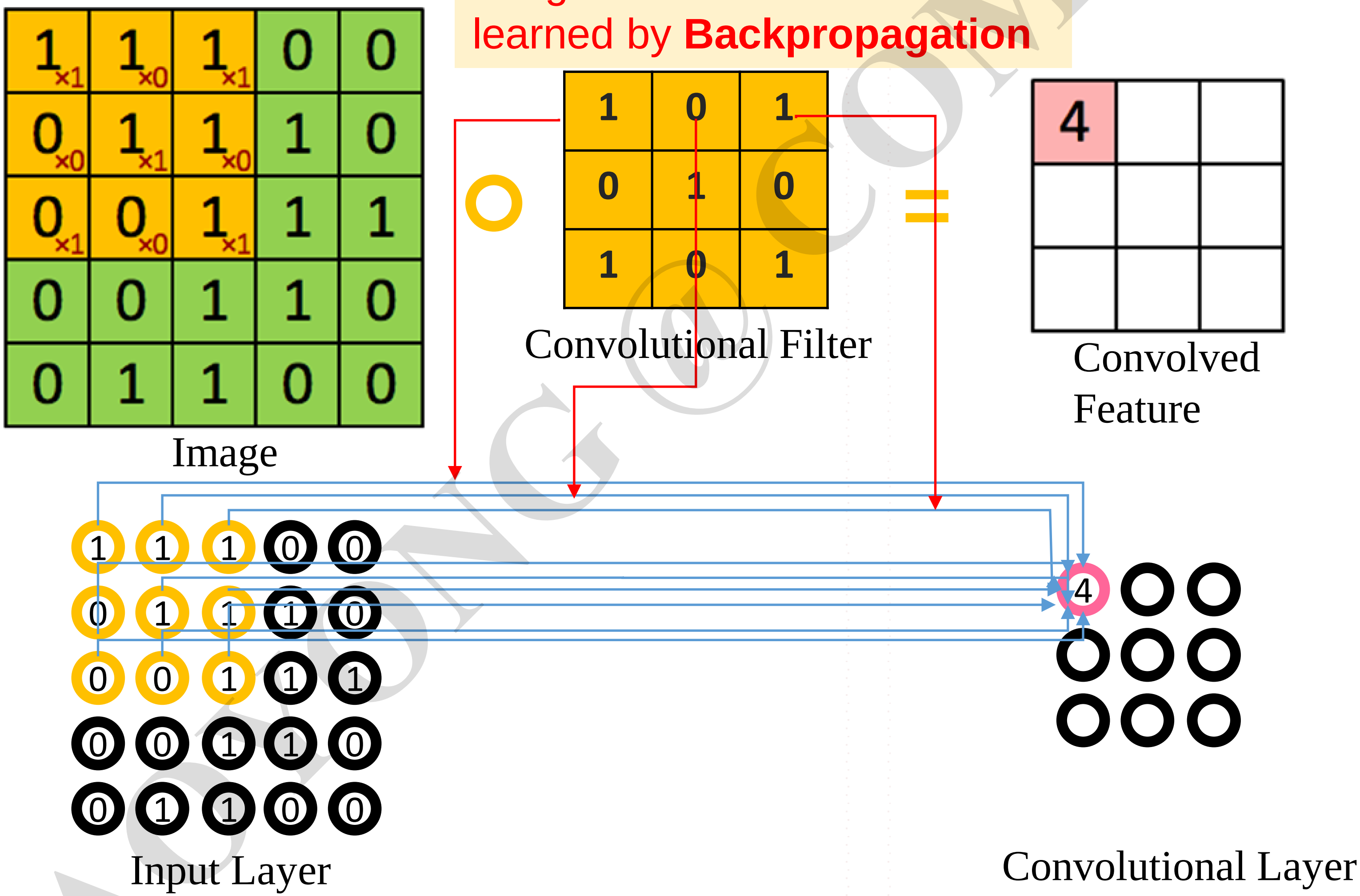
Implement with Neural Networks



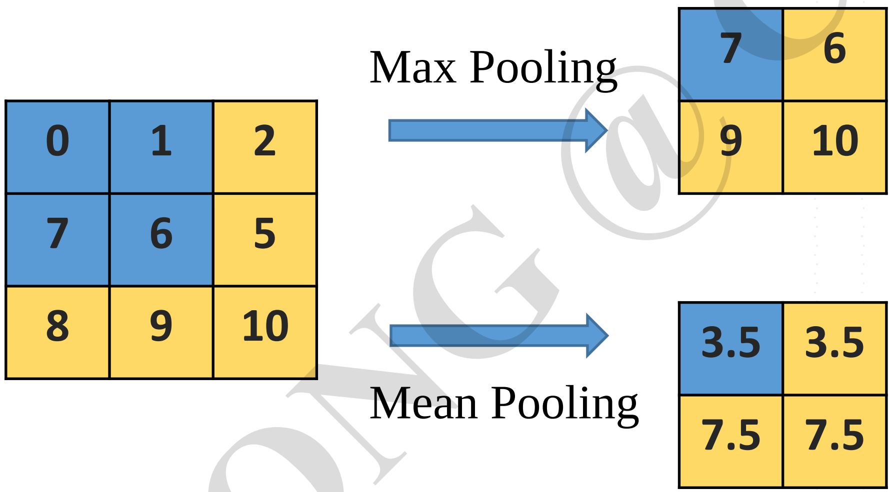
Neurons are not fully connected, which results in **Local Receptive Fields.**

Implement with Neural Networks

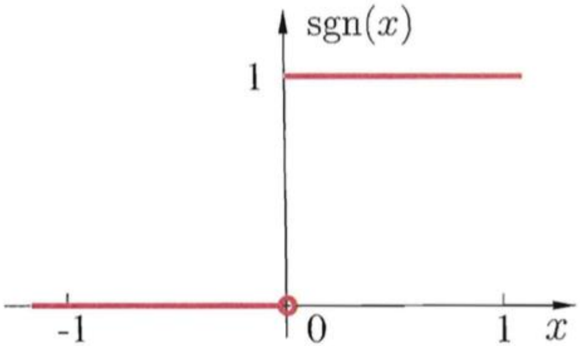
Weights of the filters can be learned by **Backpropagation**



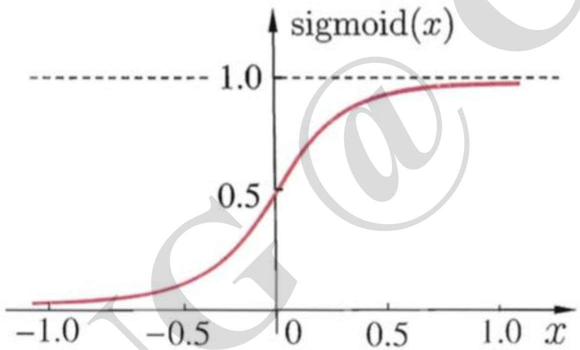
Pooling



Activation

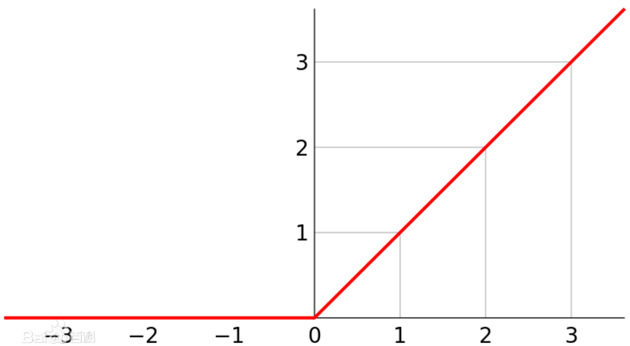


$$\operatorname{sgn}(x) = \begin{cases} 1, & x \geqslant 0; \\ 0, & x < 0. \end{cases}$$



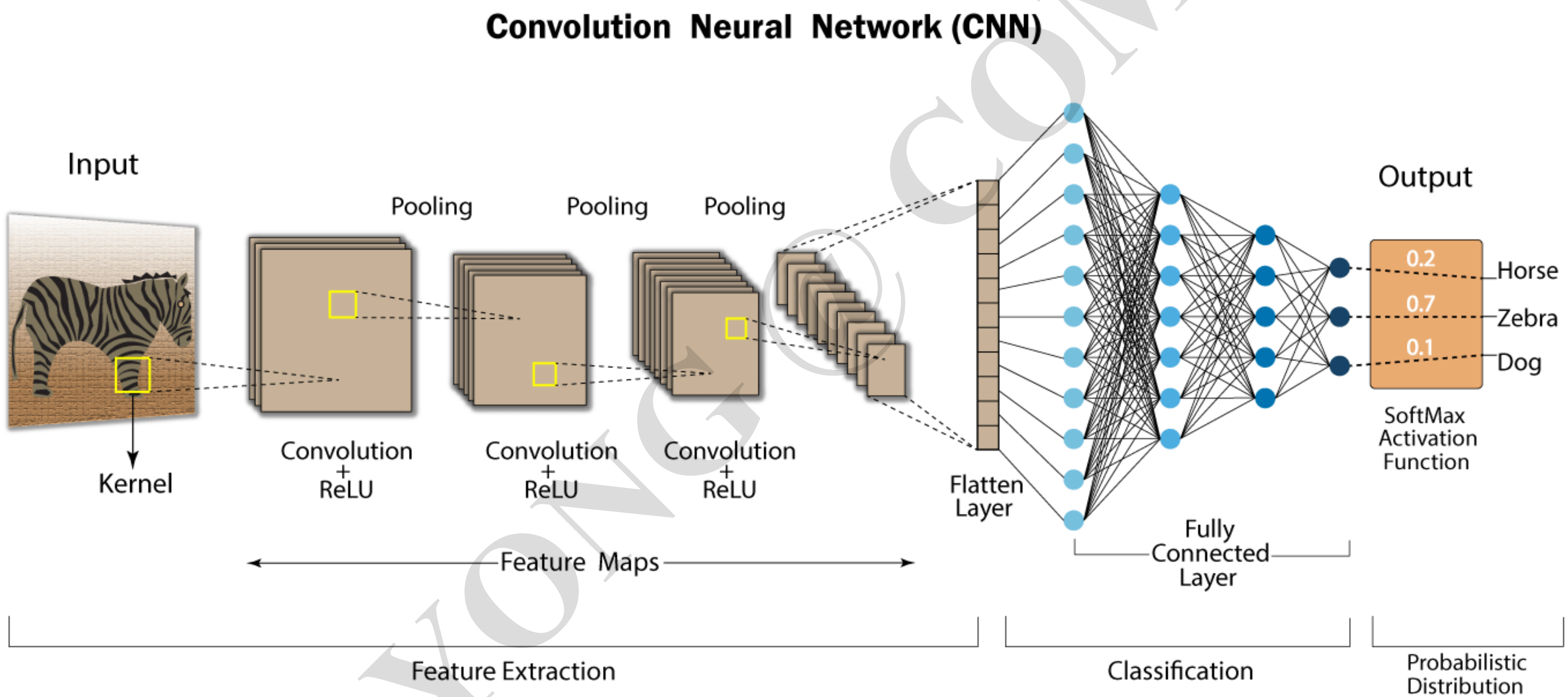
$$\operatorname{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

$ReLU(x)$



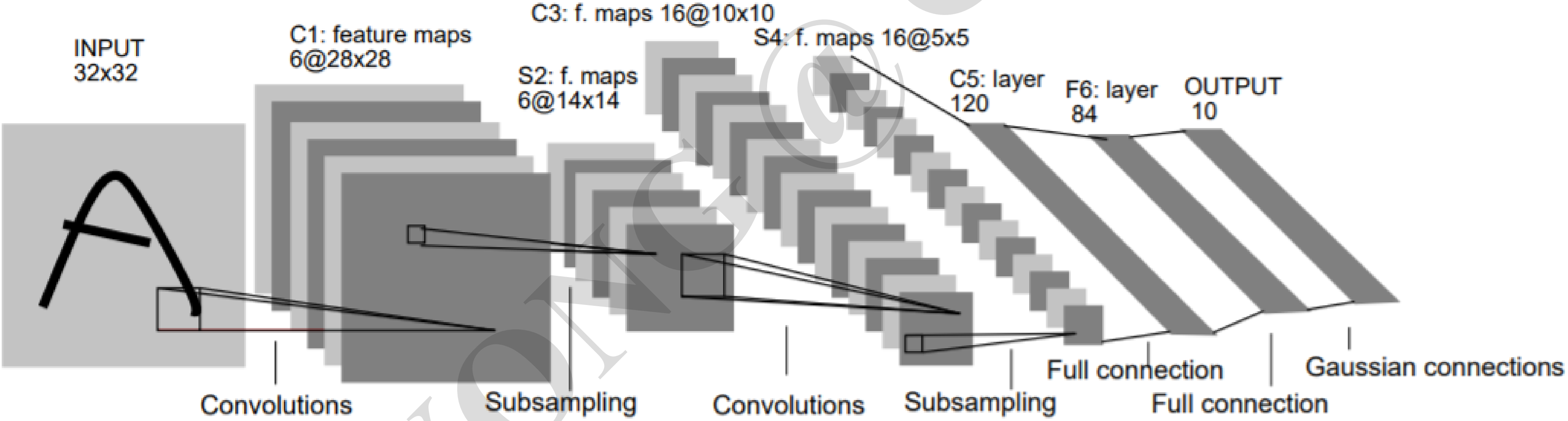
$$ReLU(x) = \max(0, x)$$

Convolutional Networks

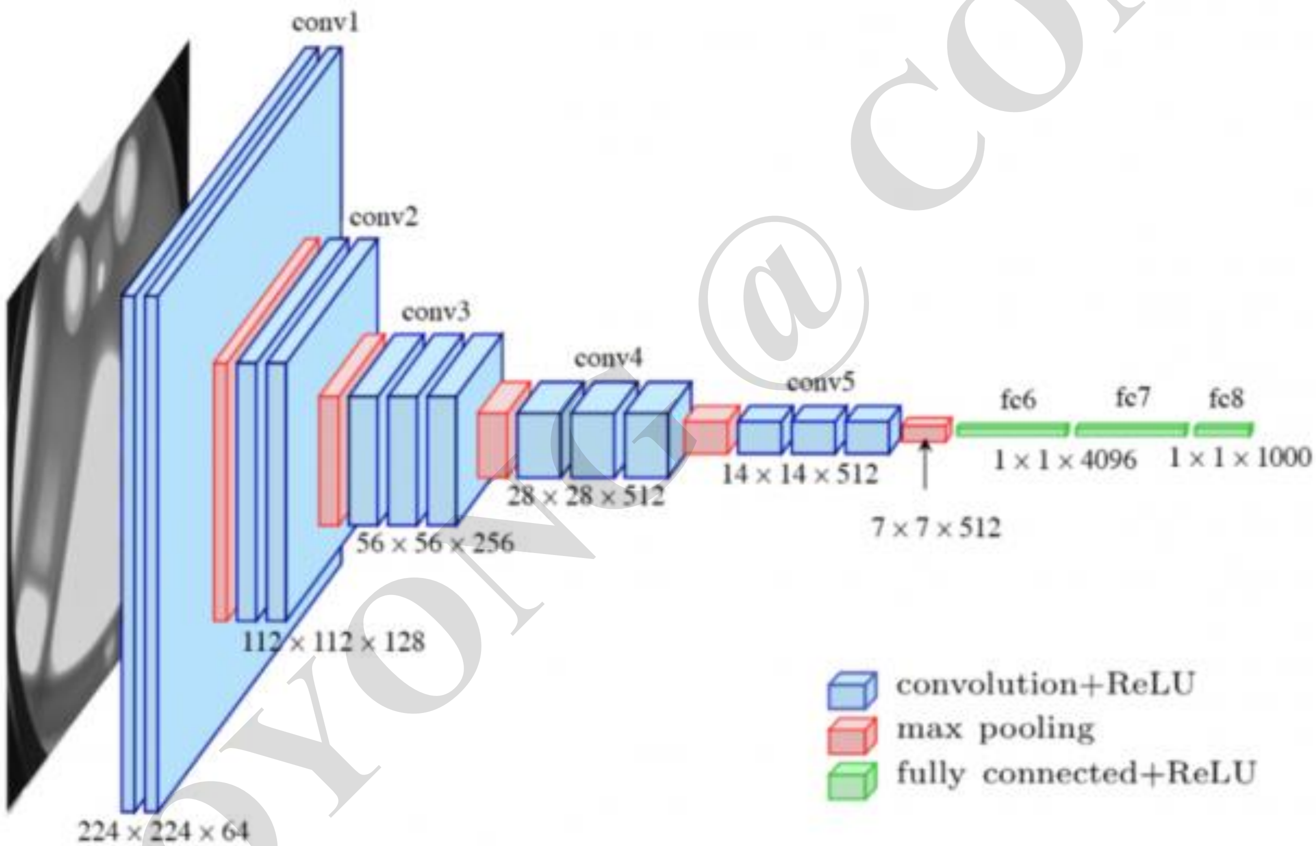


https://discuss.boardinfinity.com/t/what-do-you-mean-by-convolutional-neural-network/8533

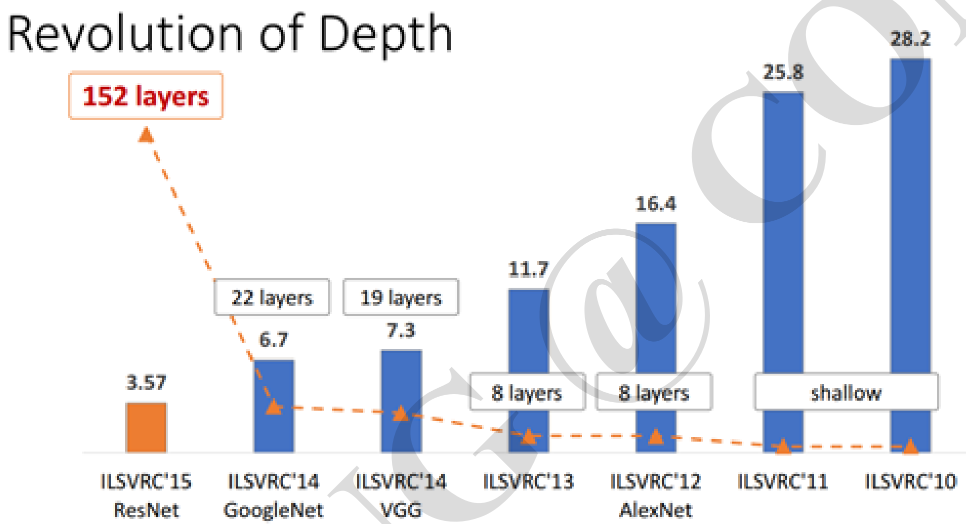
AlexNet by Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton



VGG16 by Karen Simonyan, Andrew Zisserman @ Oxford



ResNet by K. He and *et al.*



ResNet @ ILSVRC & COCO 2015 Competitions

1st places in all five main tracks

ImageNet Classification: "Ultra-deep" 152-layer nets

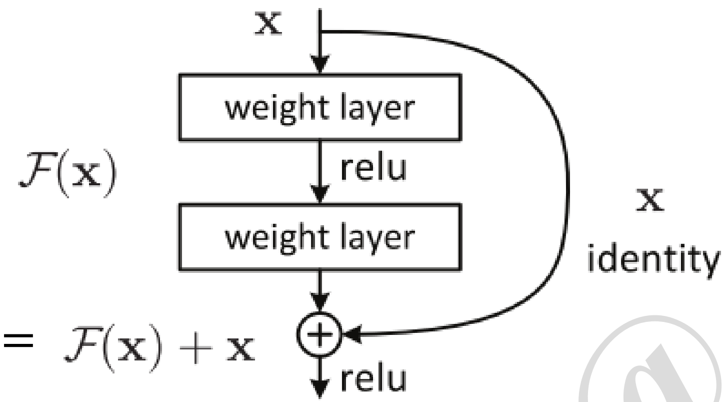
ImageNet Detection: 16% better than 2nd

ImageNet Localization: 27% better than 2nd

COCO Detection: 11% better than 2nd

COCO Segmentation: 12% better than 2nd

Vanishing Gradients and Residual Learning

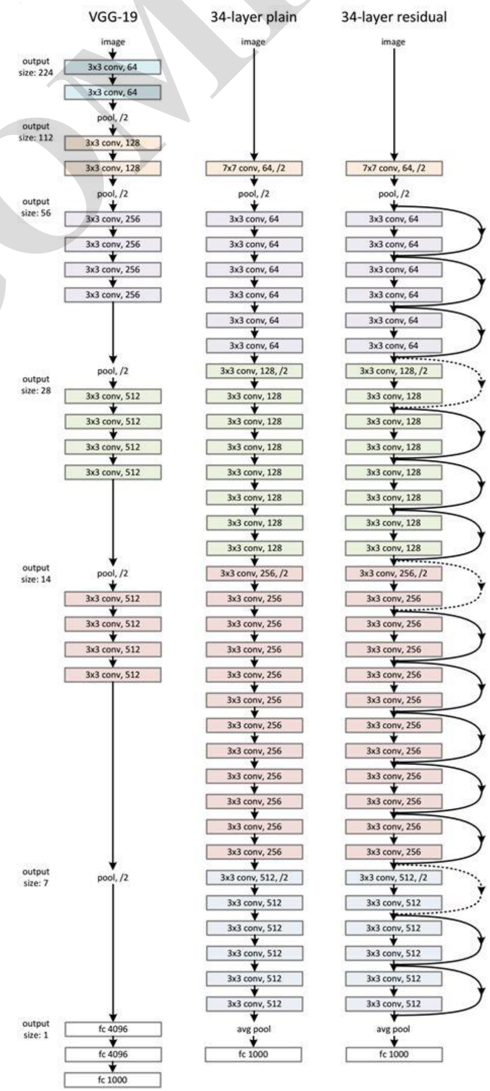


$$H(x) = \mathcal{F}(\mathbf{x}) + \mathbf{x}$$

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial H(x)} \frac{\partial H(x)}{\partial x}$$

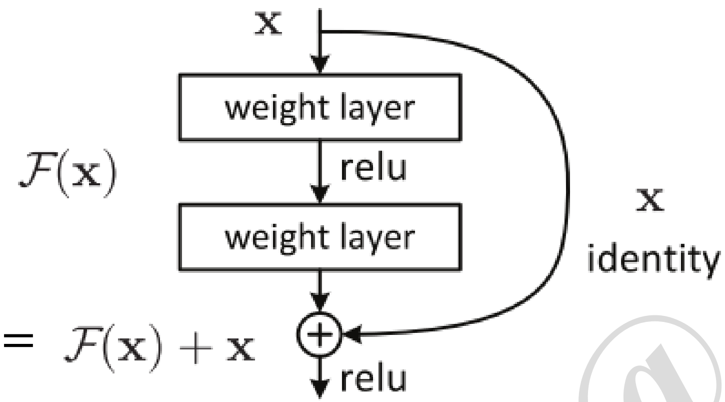
$$\frac{\partial H(x)}{\partial x} = \frac{\partial (F(x) + x)}{\partial x}$$

$$= \frac{\partial F(x)}{\partial x} + 1$$



He K, Zhang X, Ren S, et al. Deep residual learning for image recognition, Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

Vanishing Gradients and Residual Learning

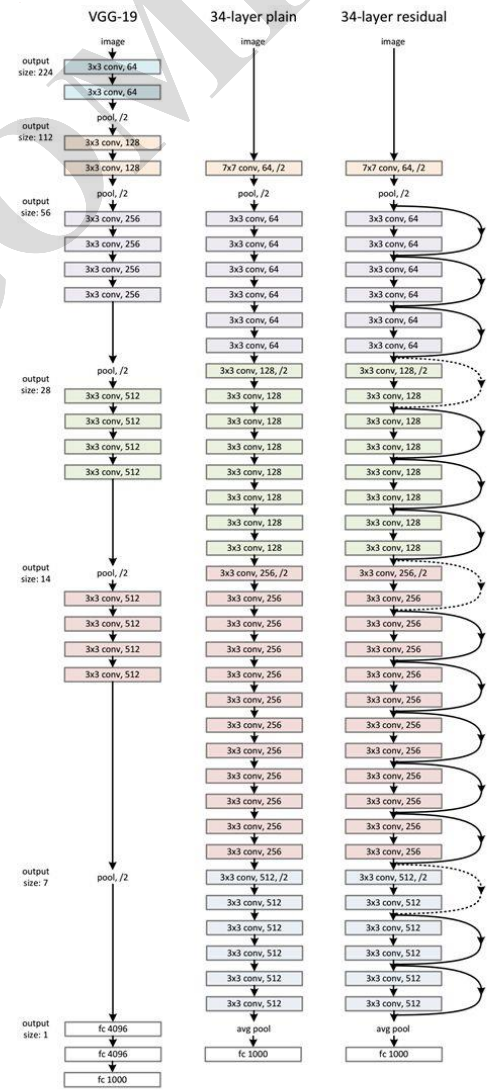


$$H(x) = \mathcal{F}(\mathbf{x}) + \mathbf{x}$$

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial H(x)} \frac{\partial H(x)}{\partial x}$$

$$\frac{\partial H(x)}{\partial x} = \frac{\partial (F(x) + x)}{\partial x}$$

$$= \frac{\partial F(x)}{\partial x} + 1$$



He K, Zhang X, Ren S, et al. Deep residual learning for image recognition, Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

Thank you!