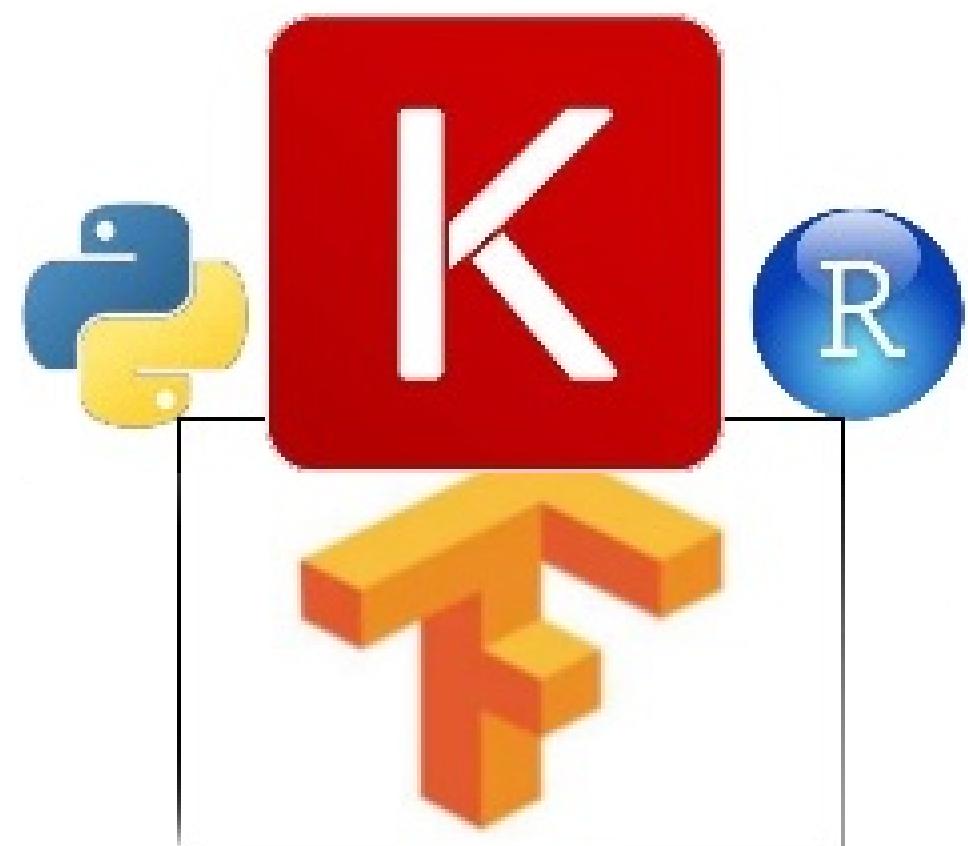


# **Image Classification with Keras and TensorFlow in Python**

## **Workshop**



Dr. Shirin Glander

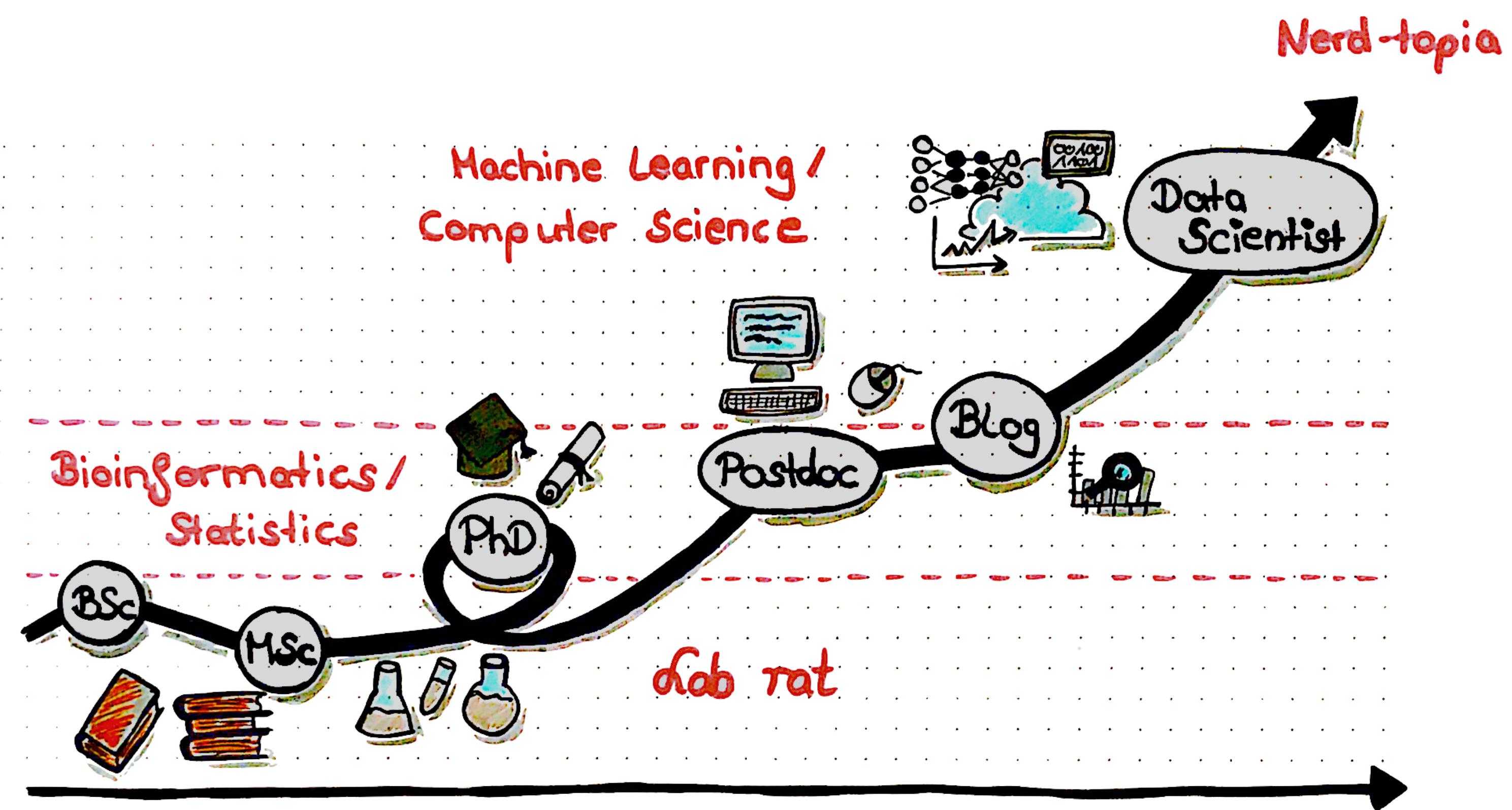
24.10.2018 - Bitmarck

# About this Workshop

- Learn what neural nets are and how they can be used in computer vision.
- Understand how a computer learns to "see".
- Apply pre-trained nets and modify them.
- Build a model from scratch that differentiates between fruits on images.
- Visualise convolutions and layers.
- (Bonus: Explaining our classifications with LIME)

Material on Github: [https://github.com/ShirinG/image\\_classification\\_keras\\_tf](https://github.com/ShirinG/image_classification_keras_tf)

# About me



# How does a computer learn to "see"?

## Convolutional Neural Nets

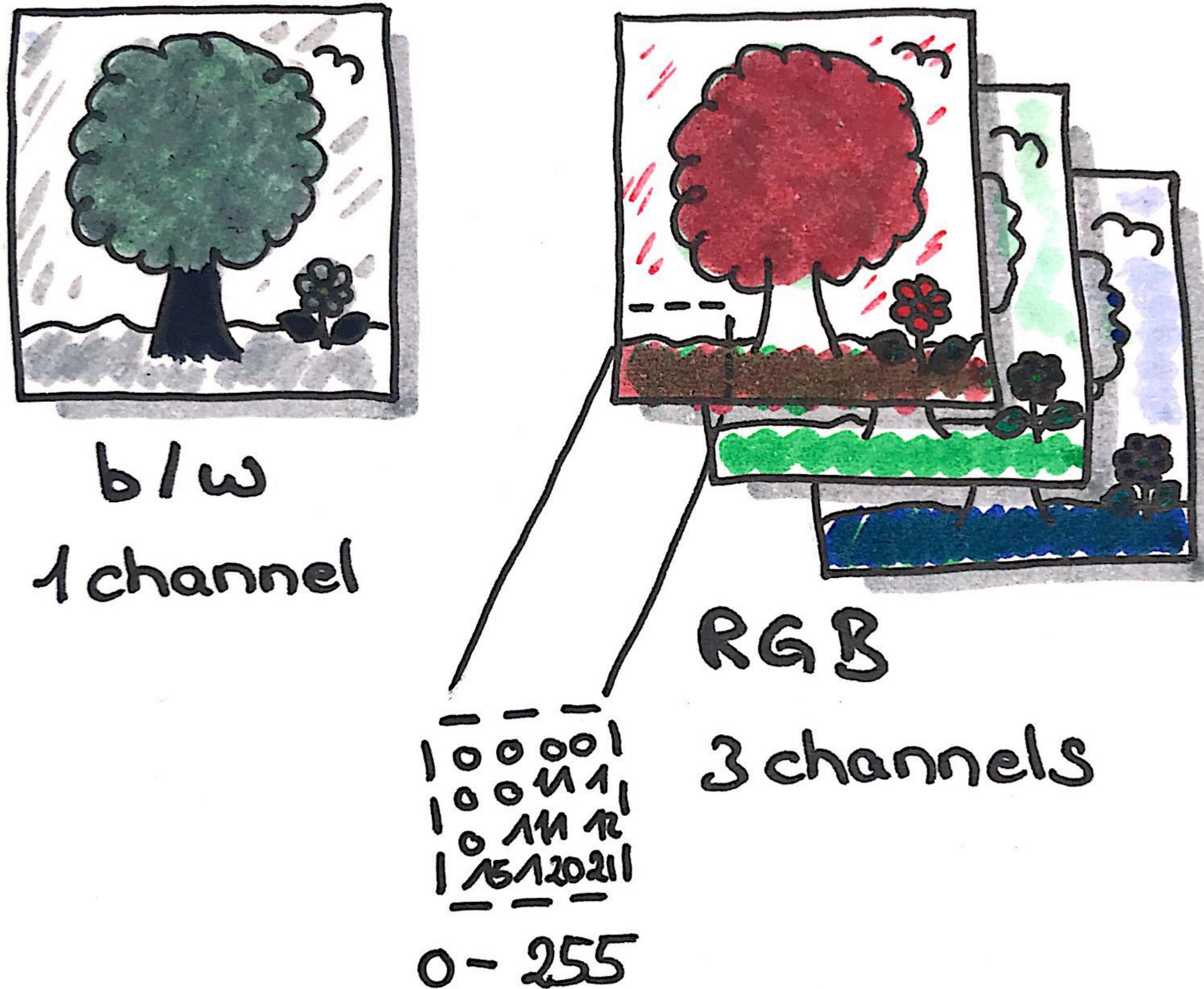


- image classification  
class = tree
- object detection  
flower

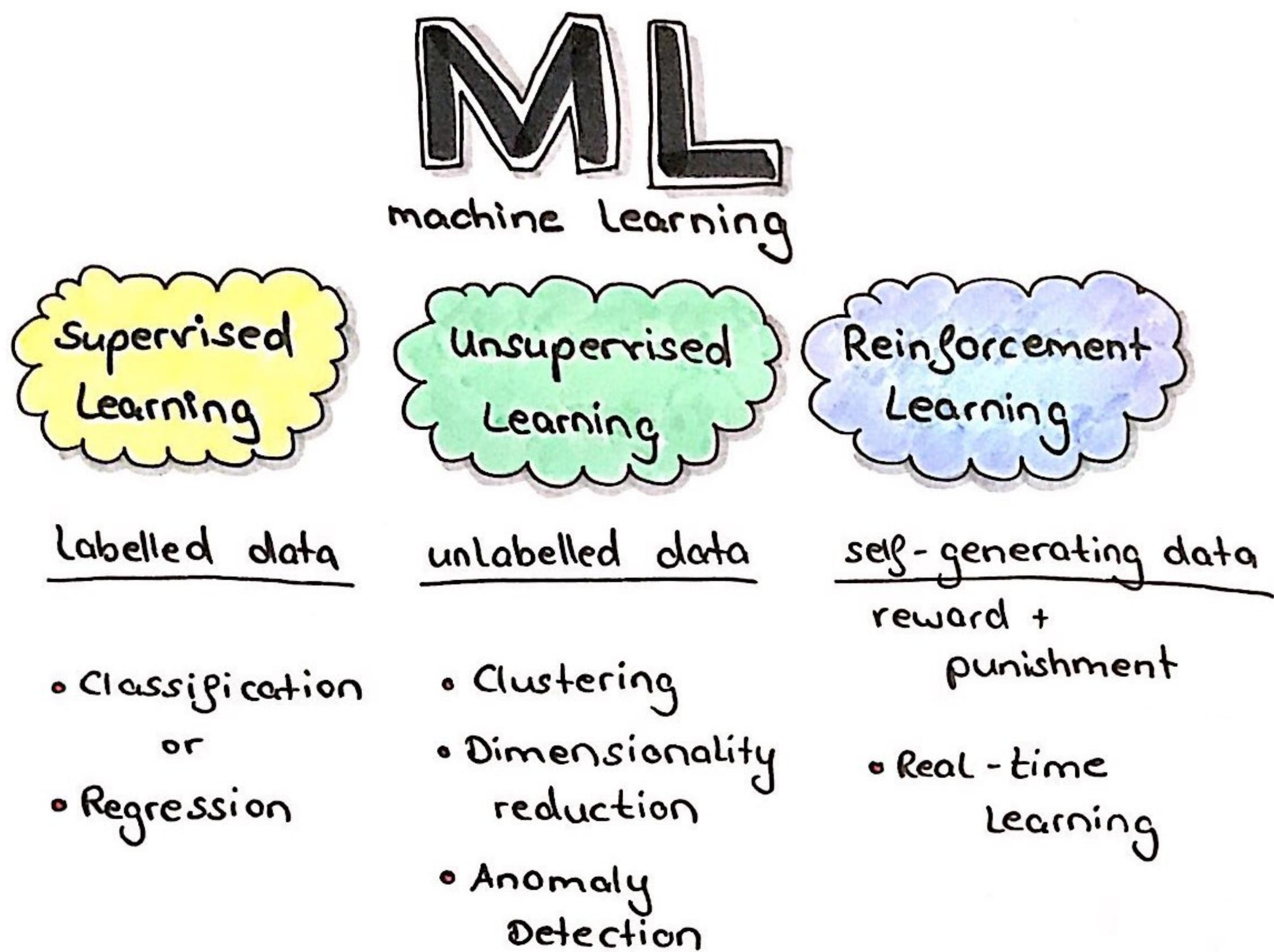
## Computer Vision

See also [this video from codecentric.AI](#) for more on Computer Vision Basics!

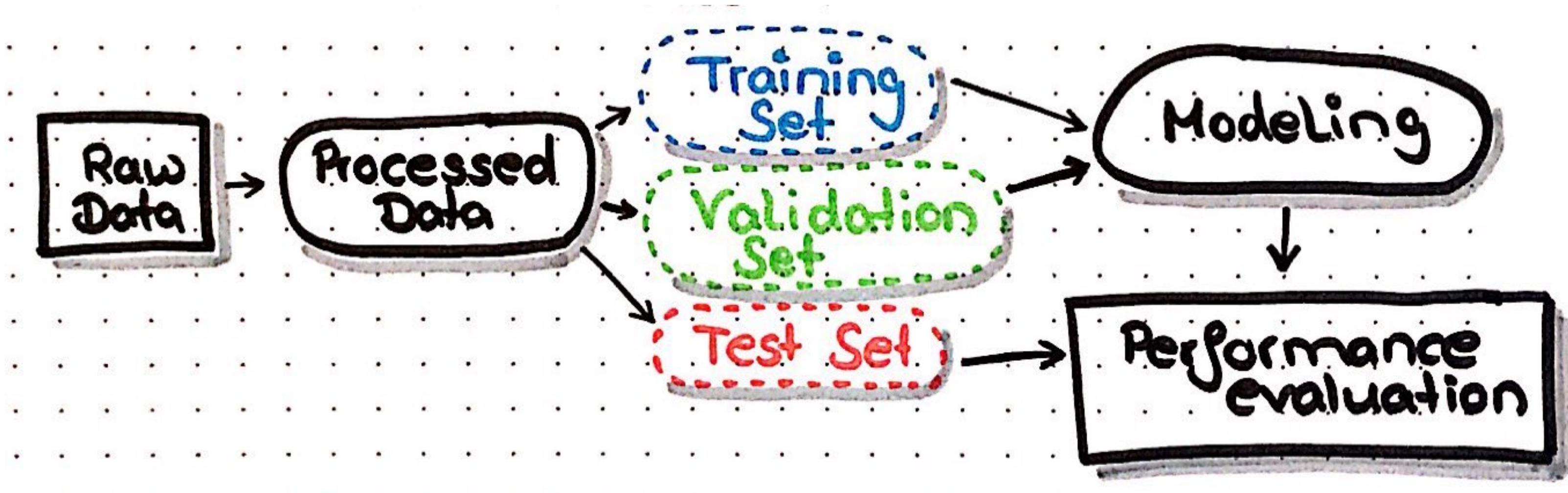
# How does a computer learn to "see"?



# What is Machine Learning (ML)



## A typical ML Workflow

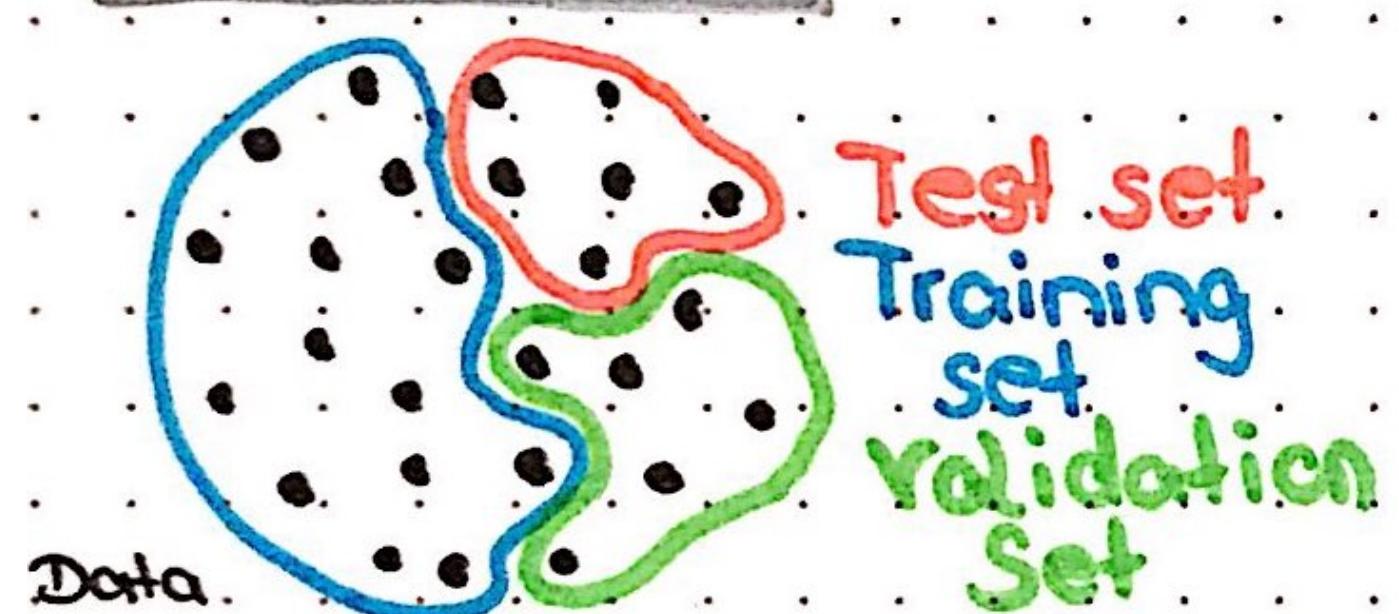


## Why use validation and test data?

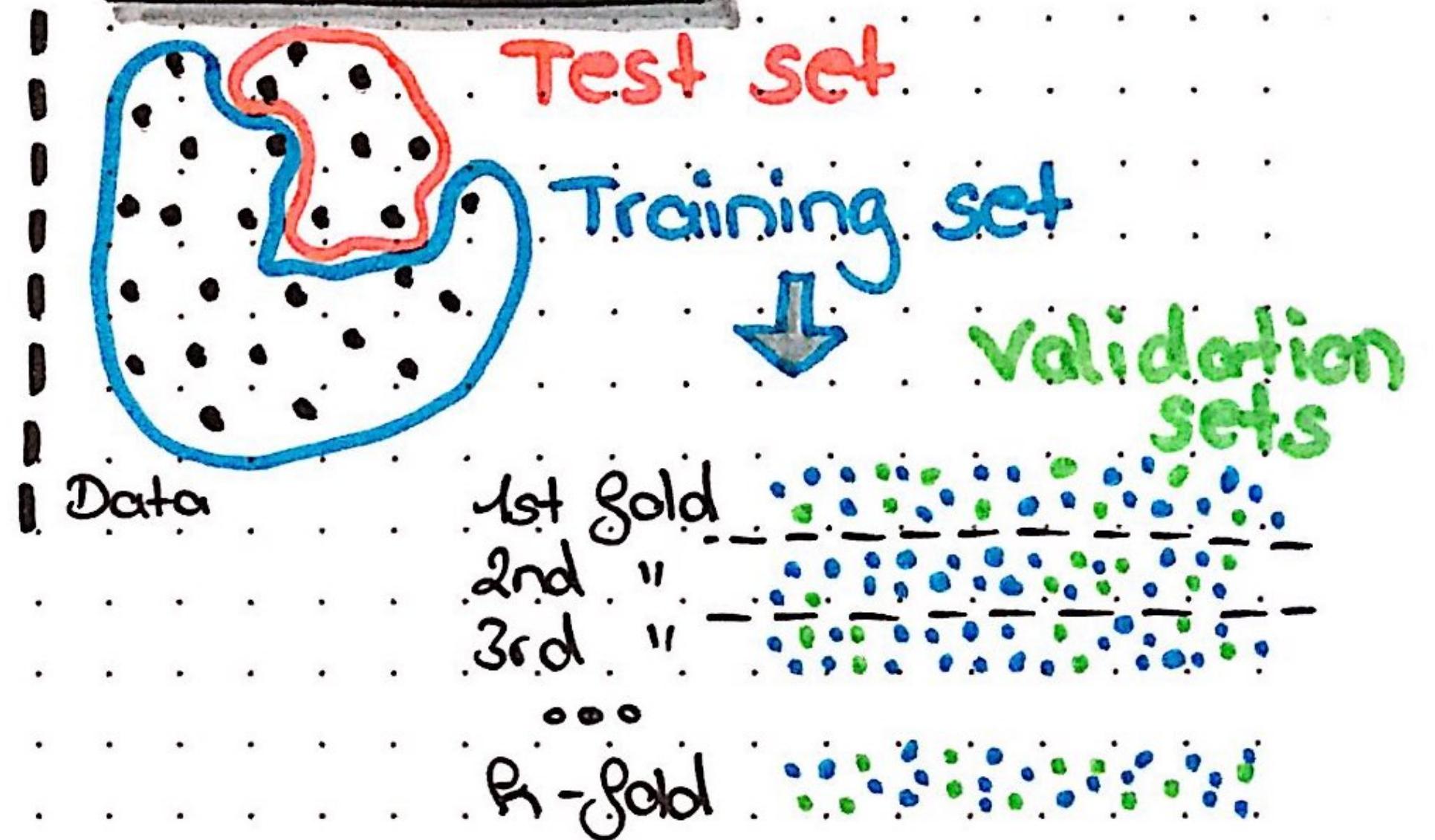


# Validation methods

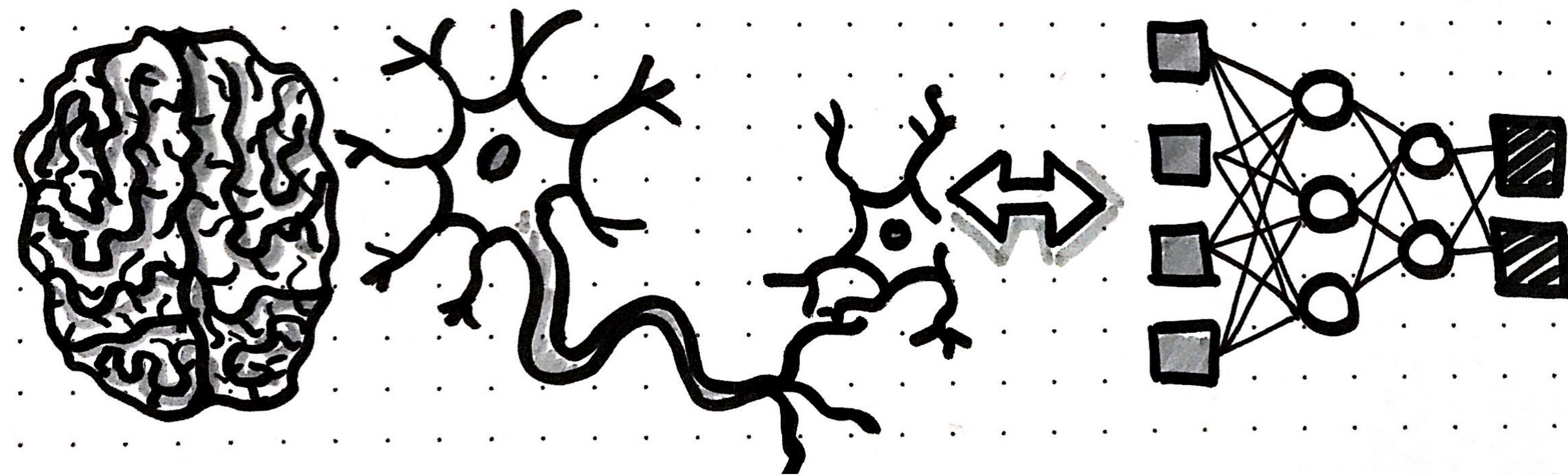
## Hold-out validation



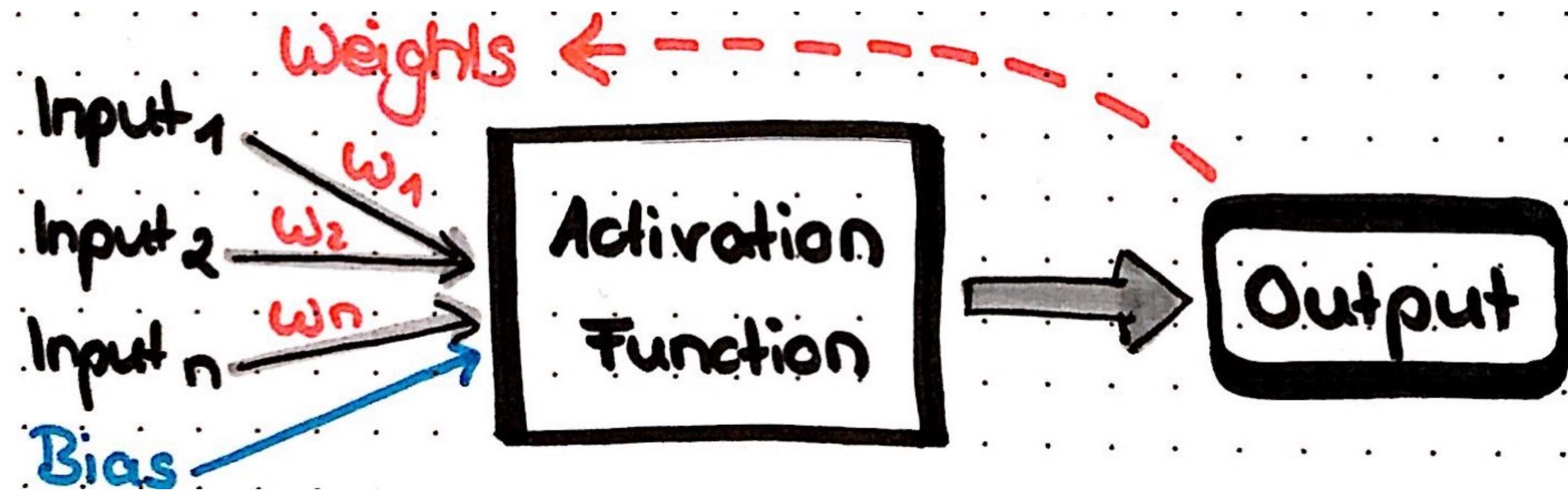
## Cross-validation



# What are neural nets?

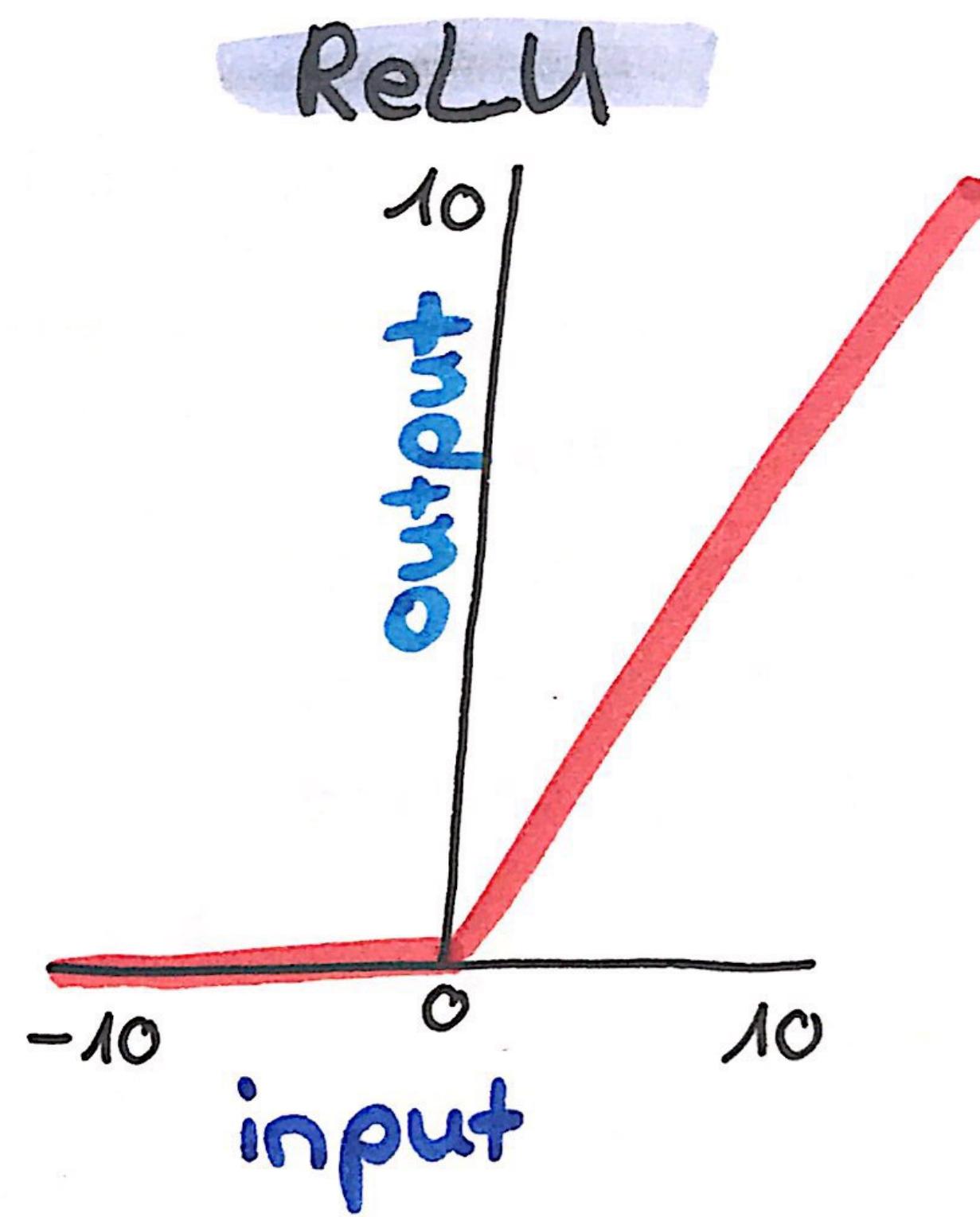
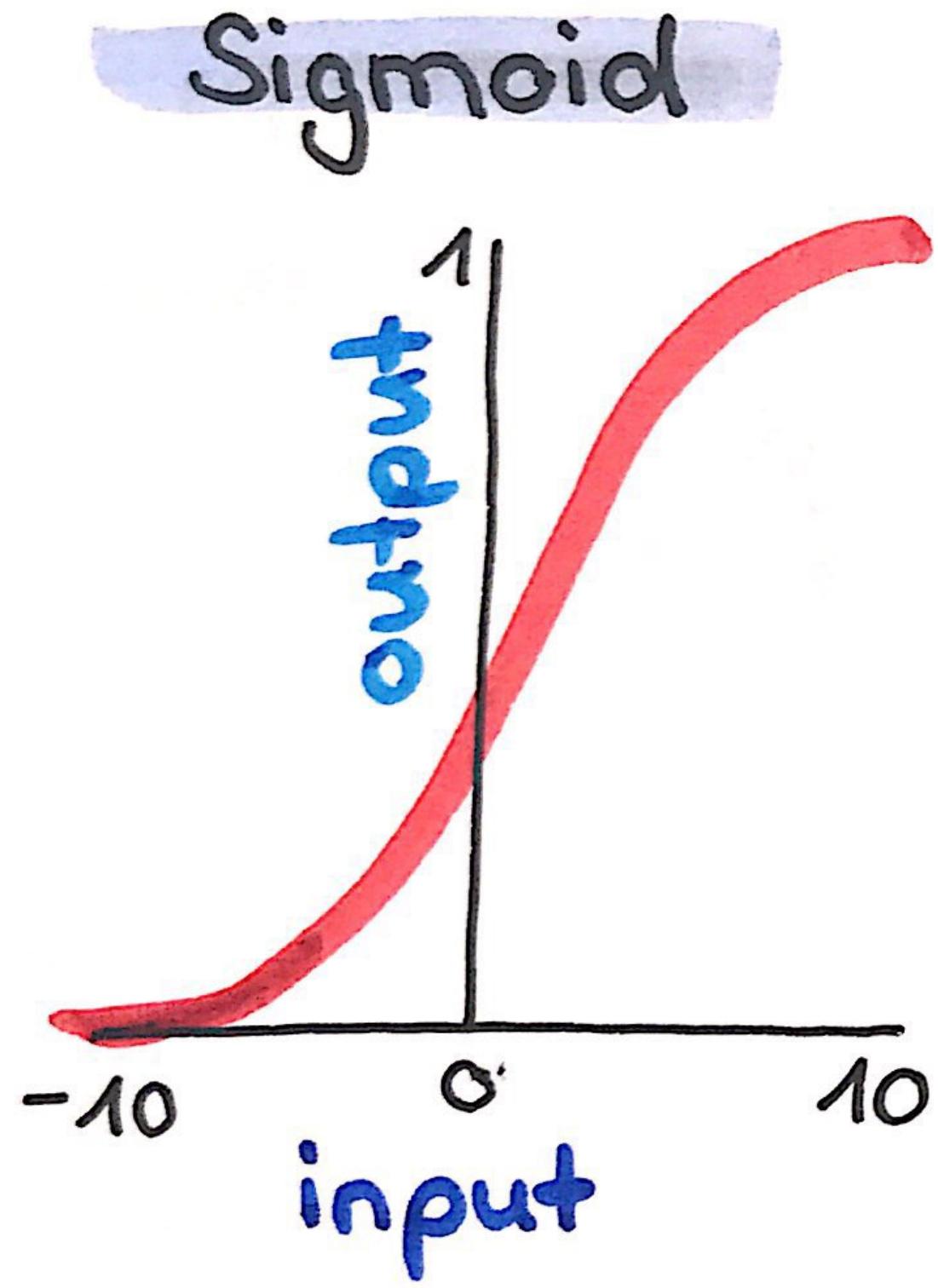


# Perceptrons



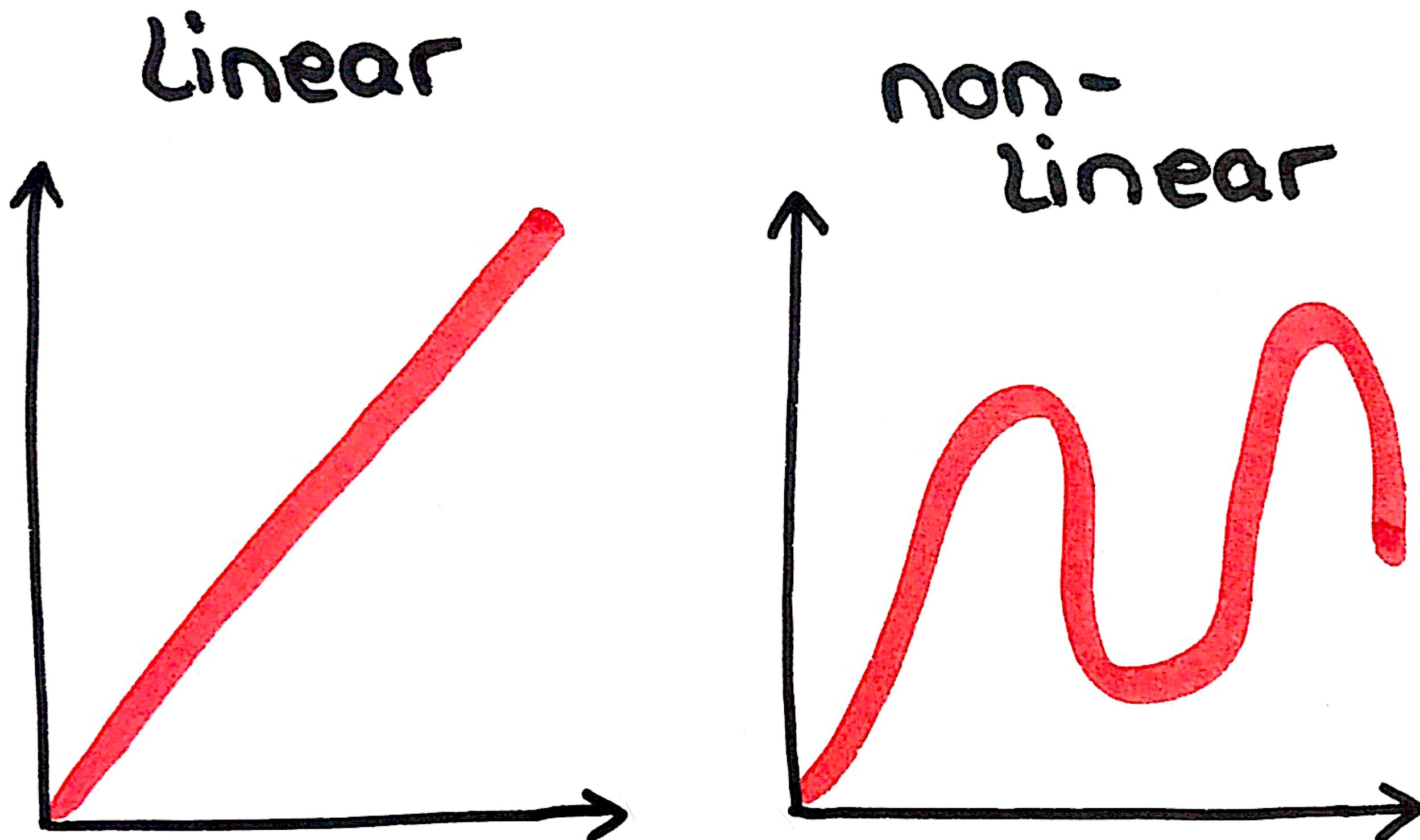
# Activation functions normalise input

- every problem can be described with a mathematical function

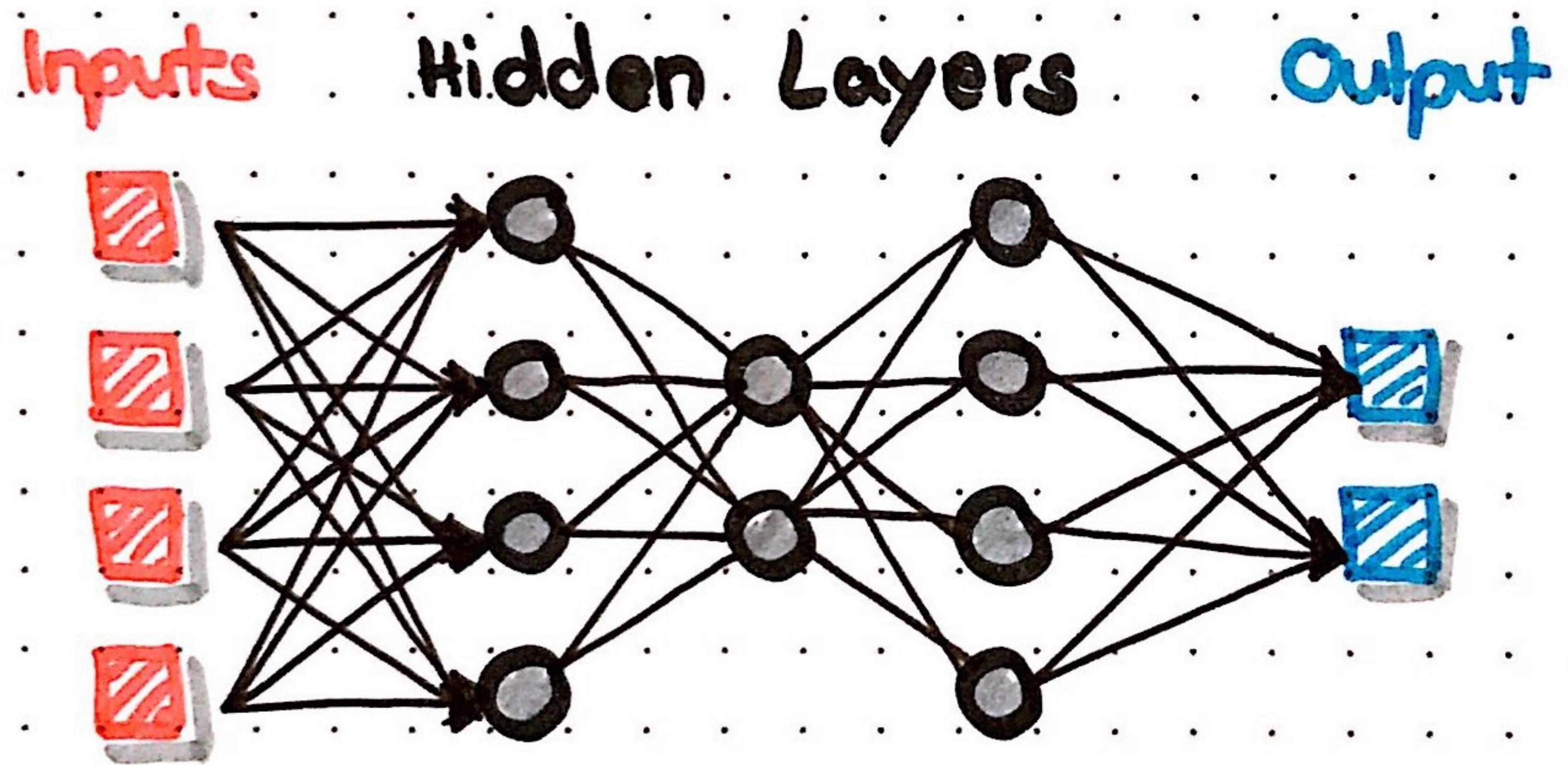


## Activation functions make non-linearity possible

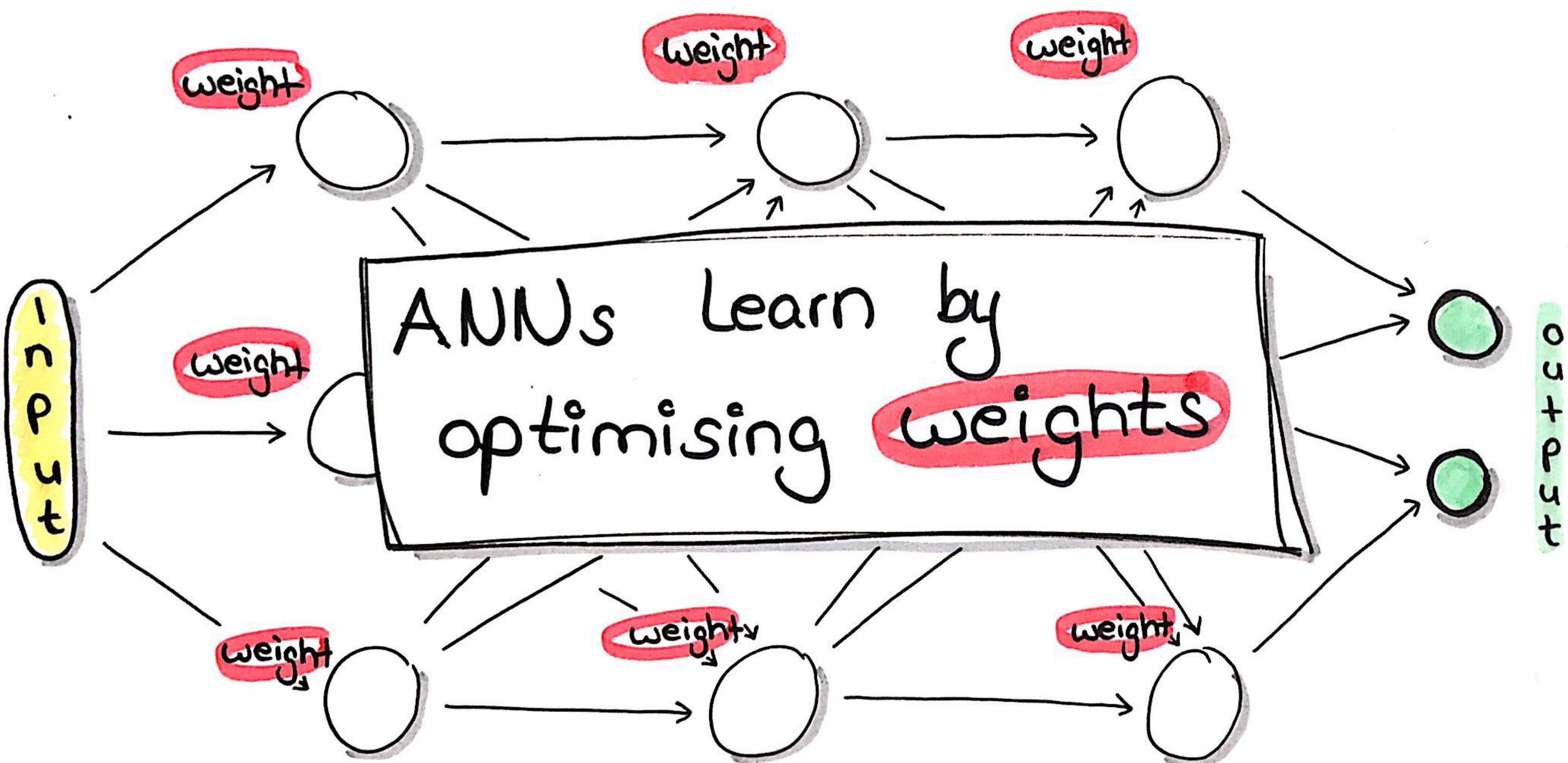
- non-linear activation functions allow us to approximate ANY mathematical formula with neural nets



# Multi-Layer Perceptrons



## How does a neural net learn?



# How does a neural net learn?

## The Softmax function

$$x * \omega + b = y$$

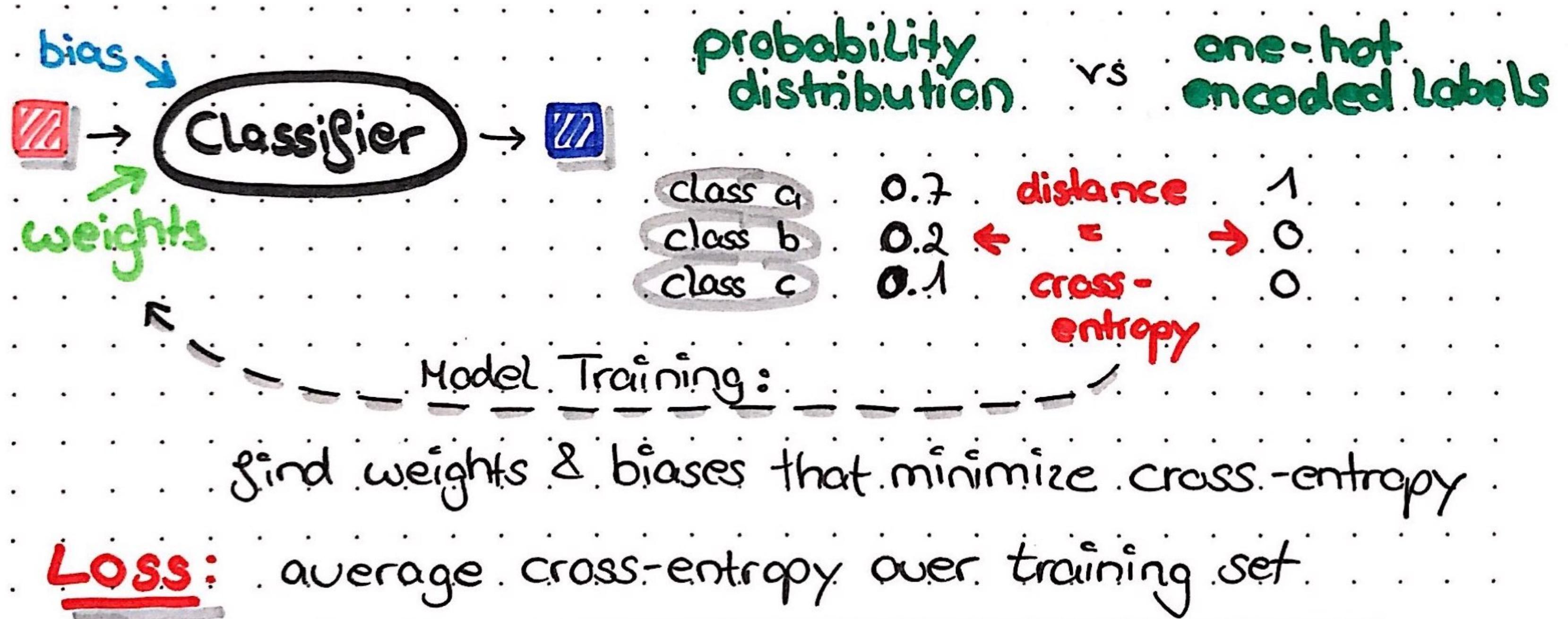
input weight bias      Output

Model Training ~ finding good weights & biases

	score	probability	
class a	2	→ 0.7	correct
class b	1	→ 0.2	
class c	0.1	→ 0.1	
			soft-max

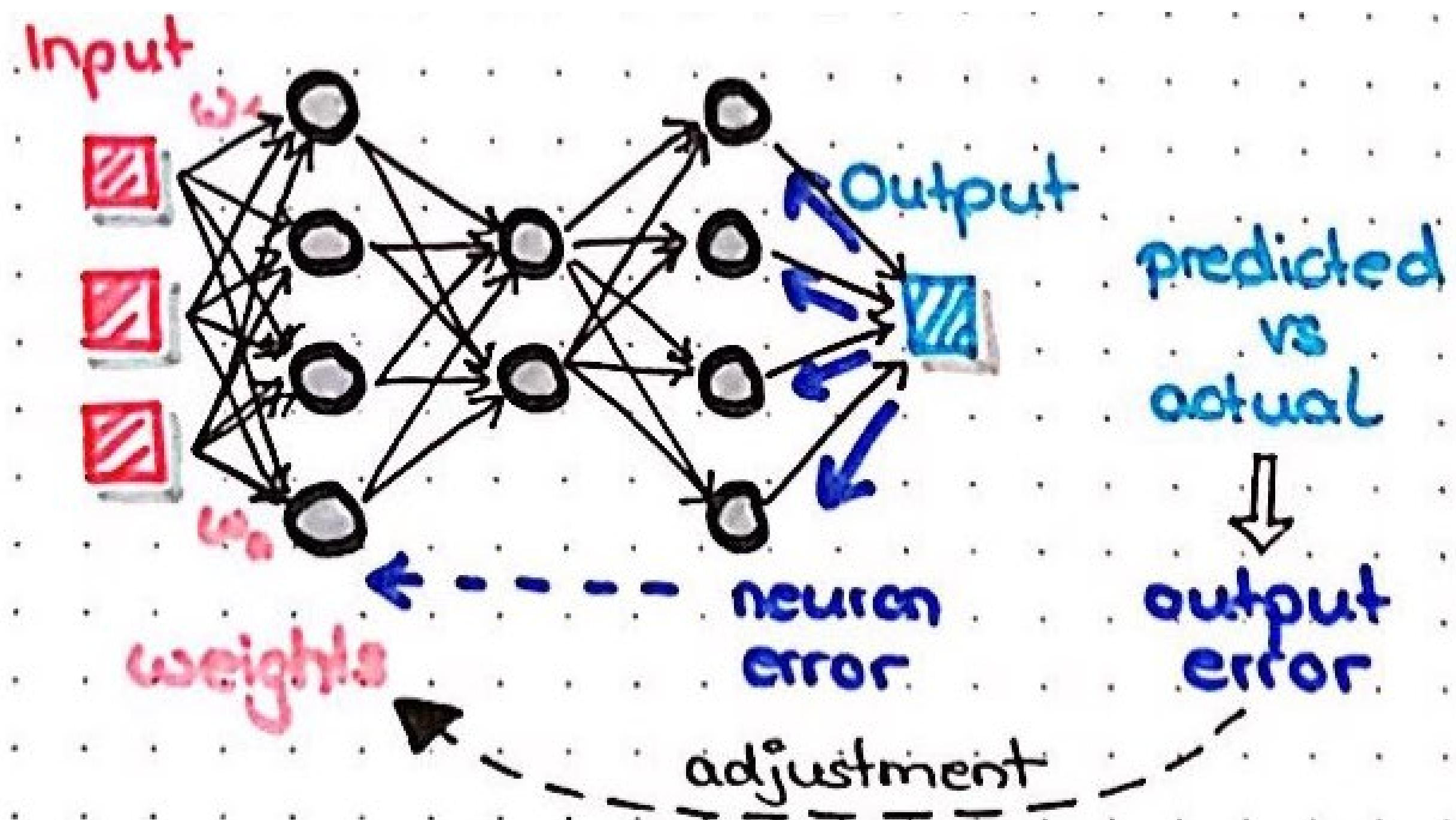
# How does a neural net learn?

## Cross-entropy



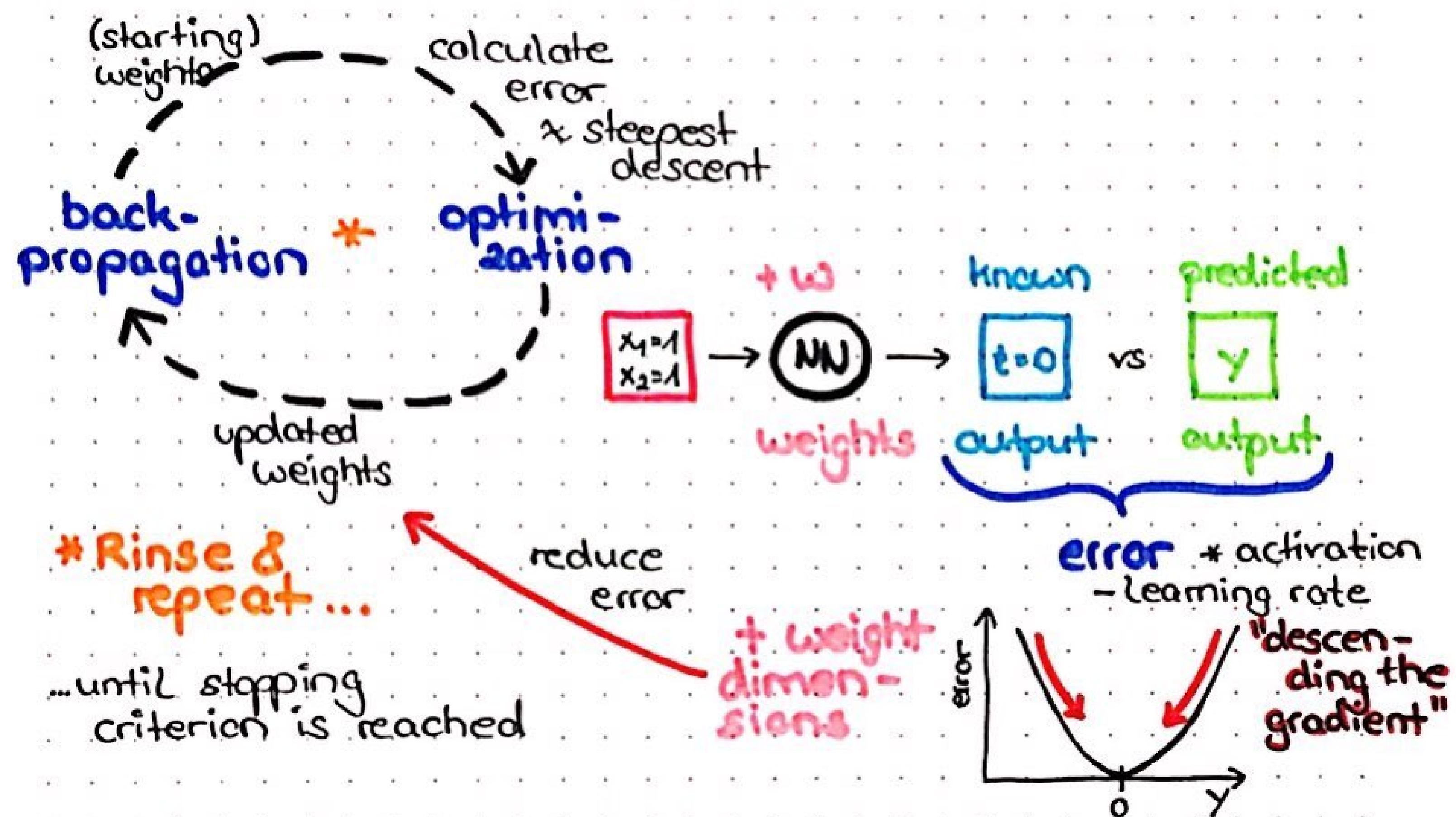
# How does a neural net learn?

## Backpropagation

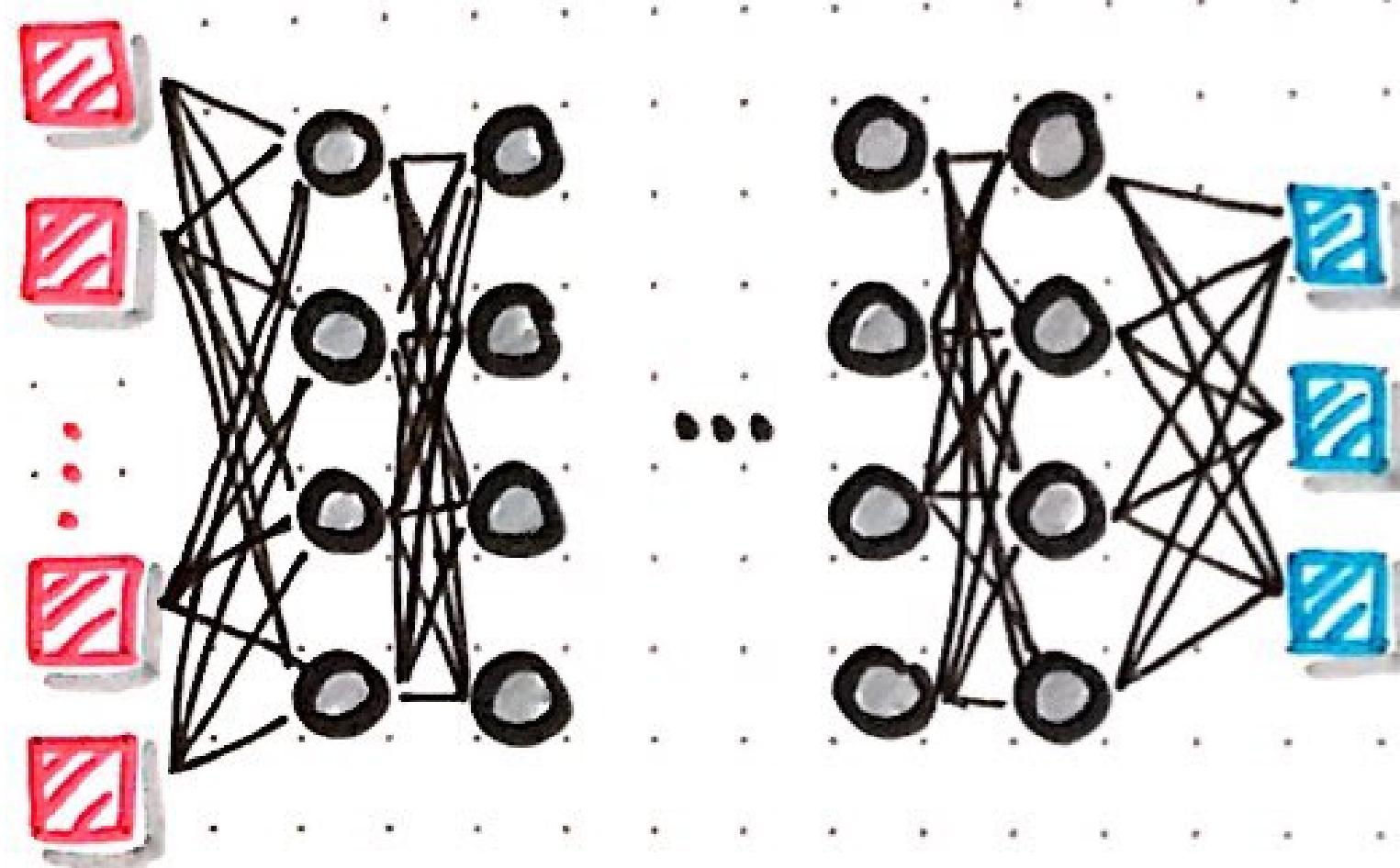


# How does a neural net learn?

## Gradient descent optimization



# Deep Learning



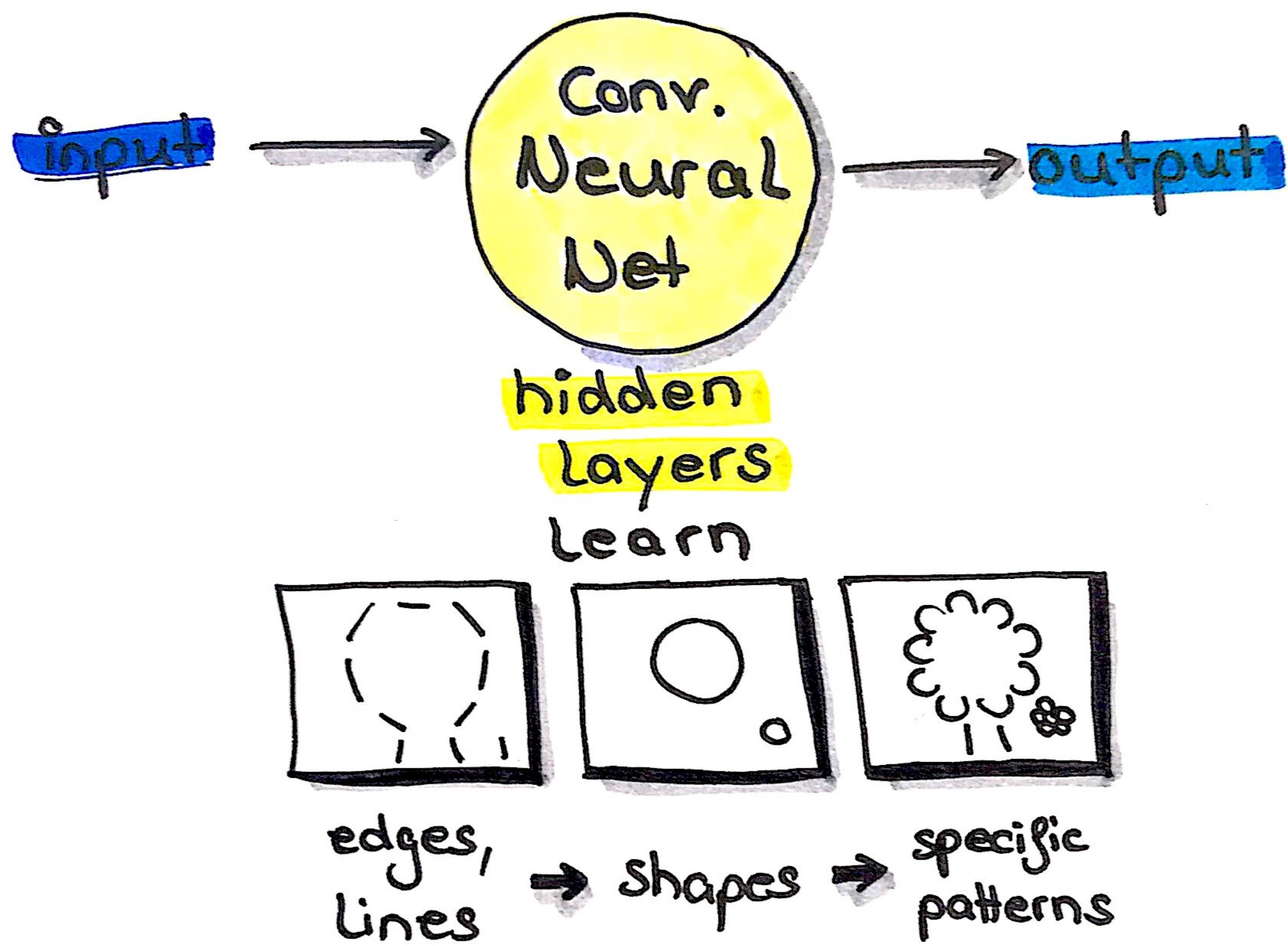
- supervised , semi - supervised or unsupervised
- NLP , speech recognition
- image recognition
- object classification
- recommender systems
- etc.

# Convolutional Neural Nets

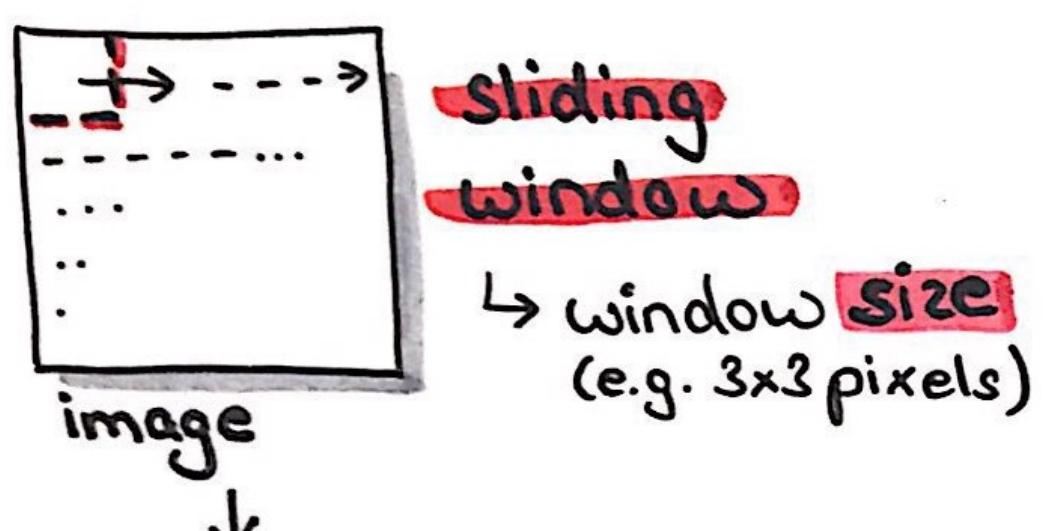
## MLPs vs CNNs

- pixels are considered independent
- computationally faster
- Learned : weights
- pixels are considered as groups of connected information (context)
- analysed as chunks (= windows)
- Learned : filters

# ConvNets



# ConvNets

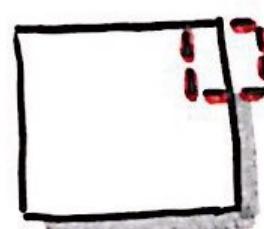


vertical Lines

horizontal Lines

- **Filters**
  - detect shapes & patterns in window chunks
  - multiple filters are combined
  - filters are learned

**padding**



- fake pixels are created at the edge of images to incorporate border pixels

**Convolutional Layers**

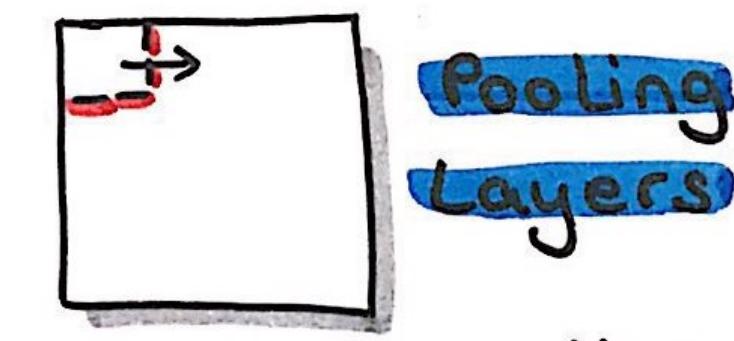
apply filters

↓

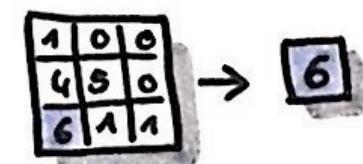
output

=  
**feature maps**

→  
**stacks of feature maps**



e.g. max pooling

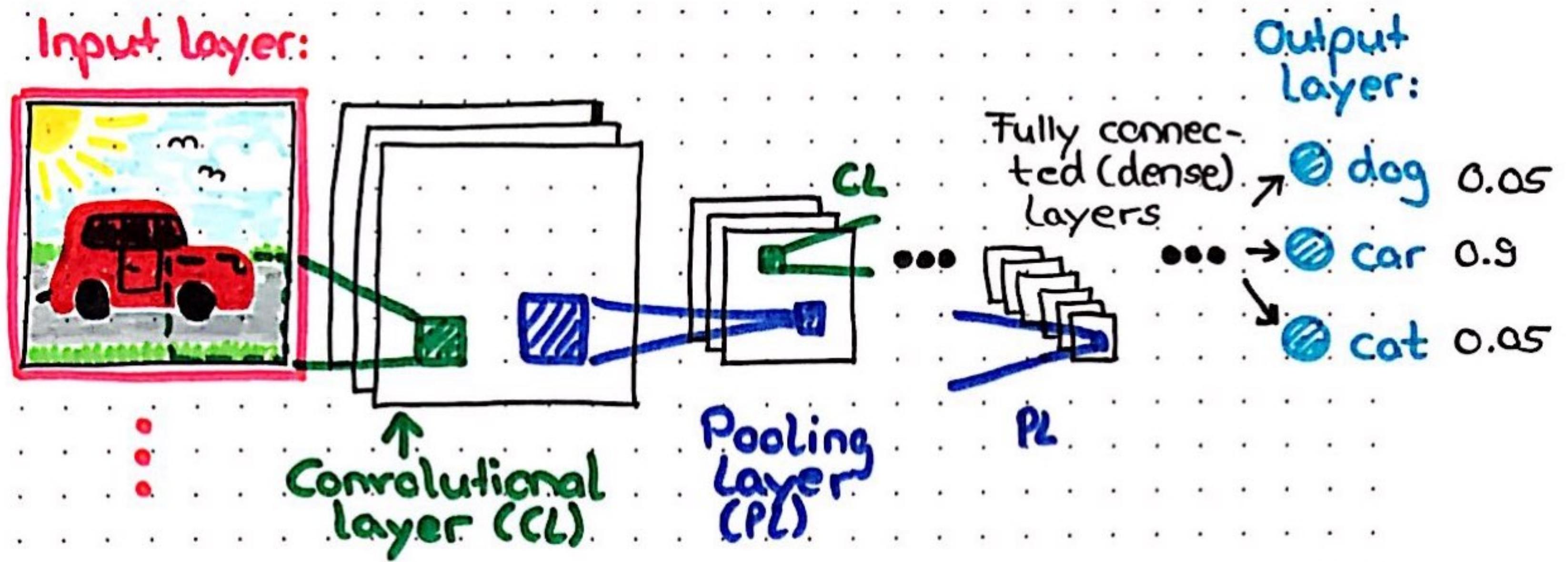


- reduces compute time
- boils information down

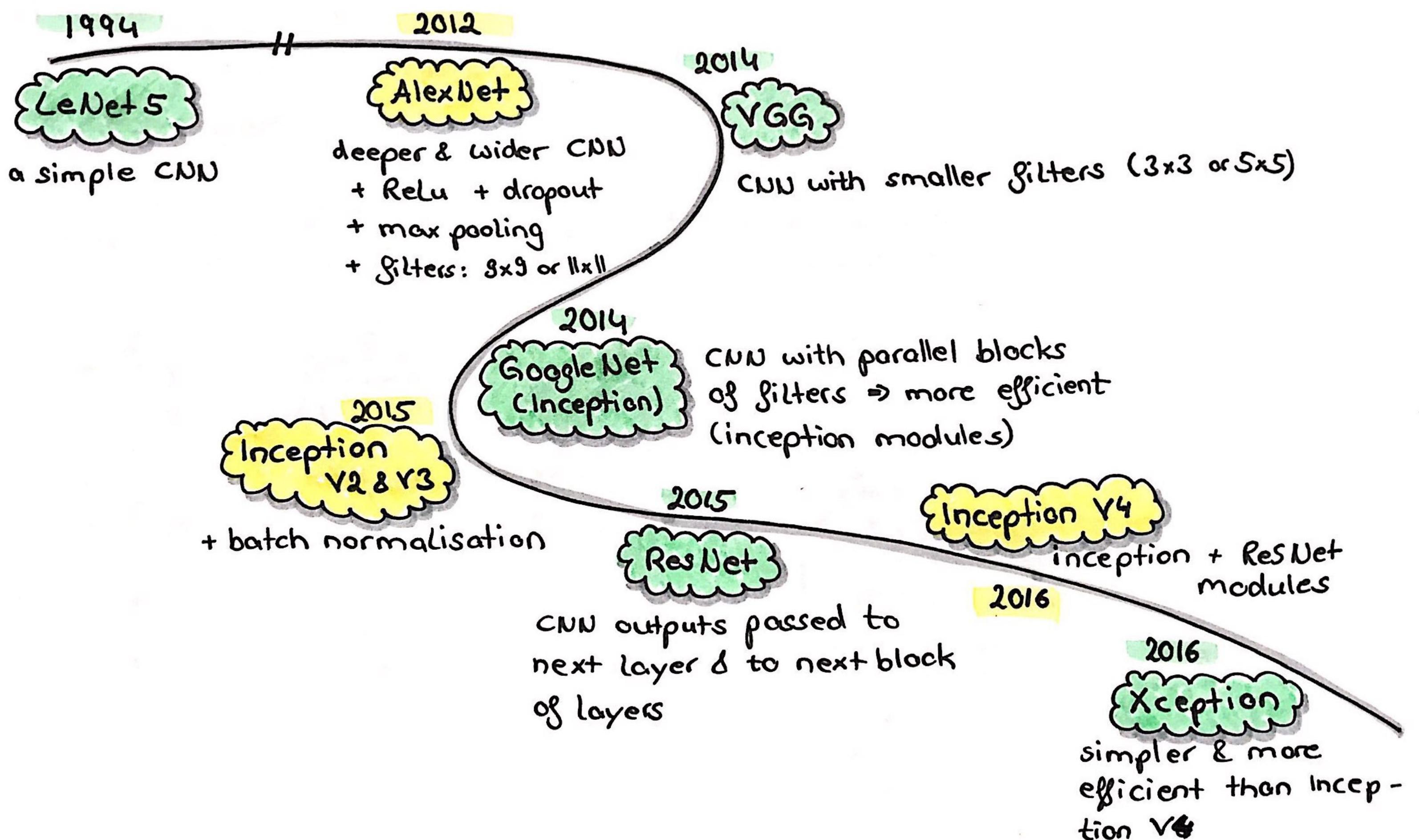
**stride**

- how much overlap to have in sliding window

# ConvNets

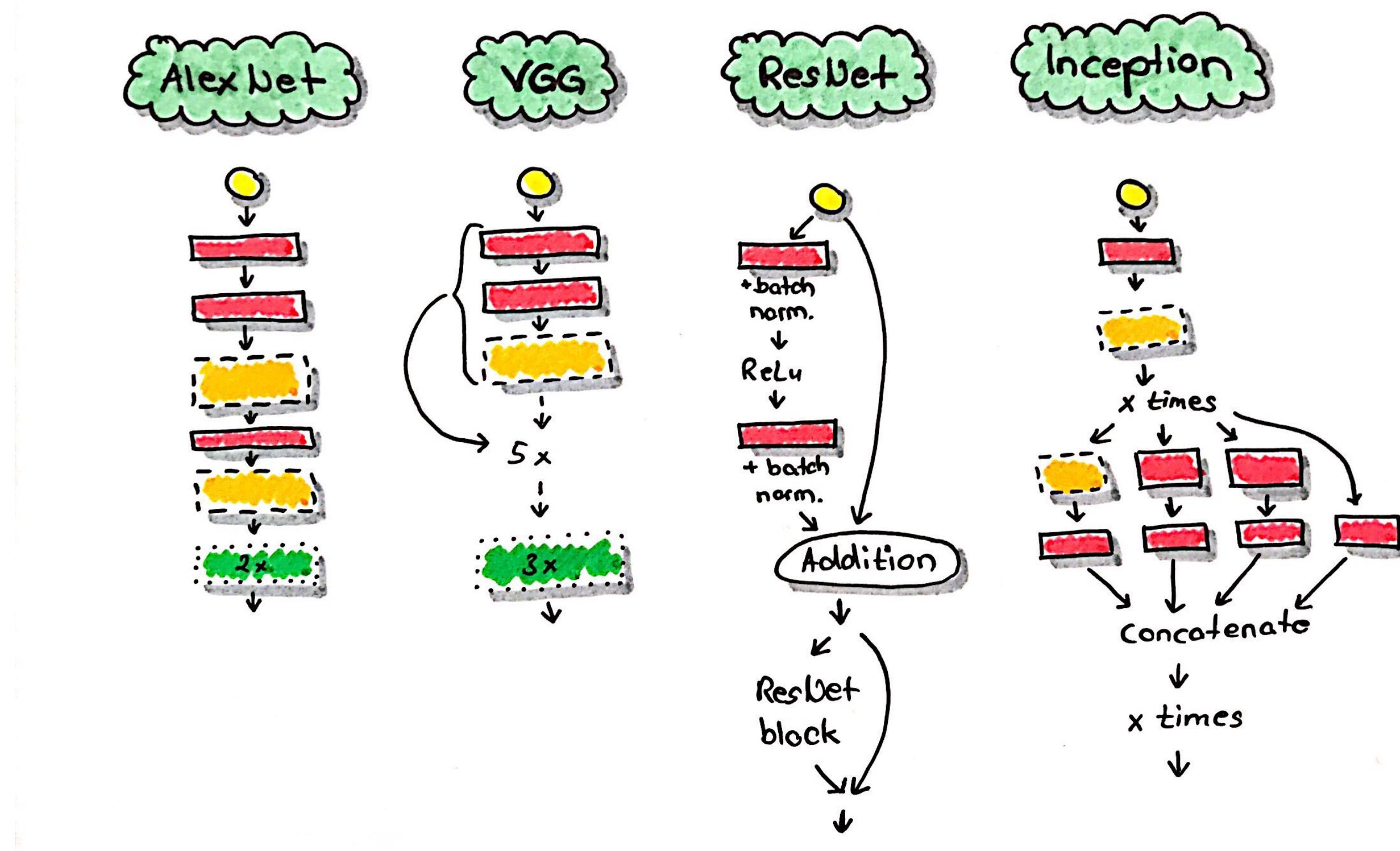


# Evolution of neural nets for image recognition

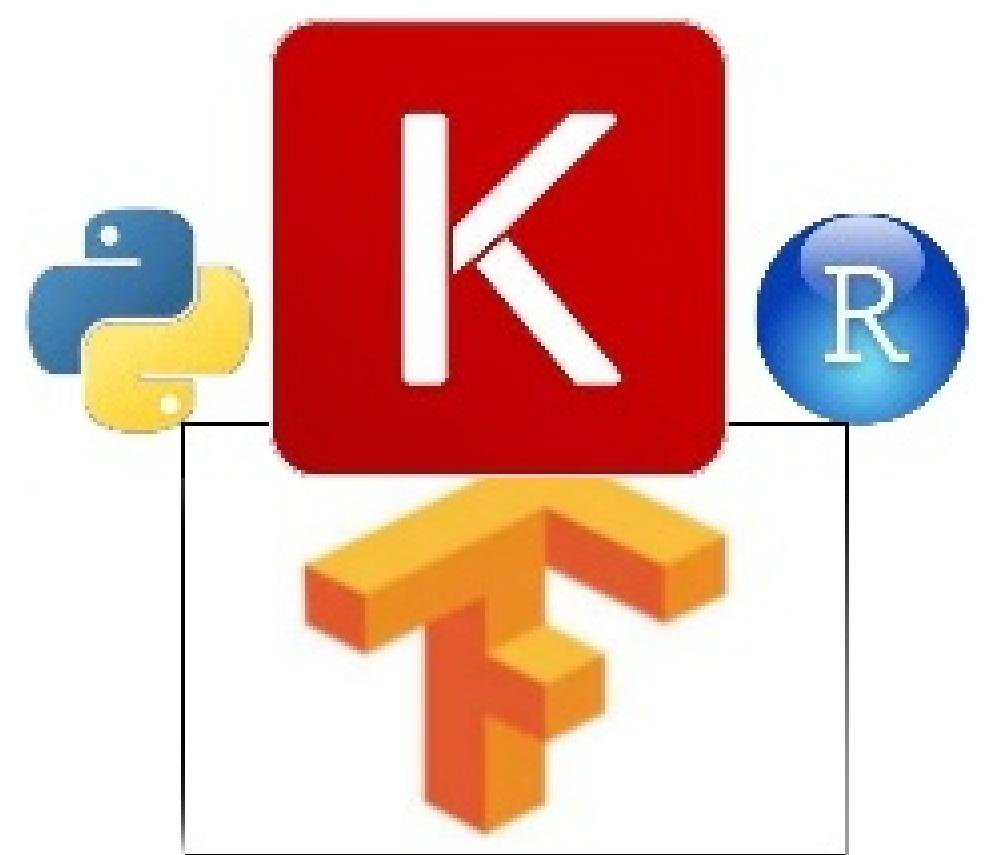


# CNN architectures

- input image
- convolutional layer
- pooling layer
- dense layer



# Introduction to TensorFlow



# What are tensors?



# Tensors = multidimensional arrays

Dimension

1

1	2	3
---	---	---

vector

2

1	2	3
4	5	6
7	8	9

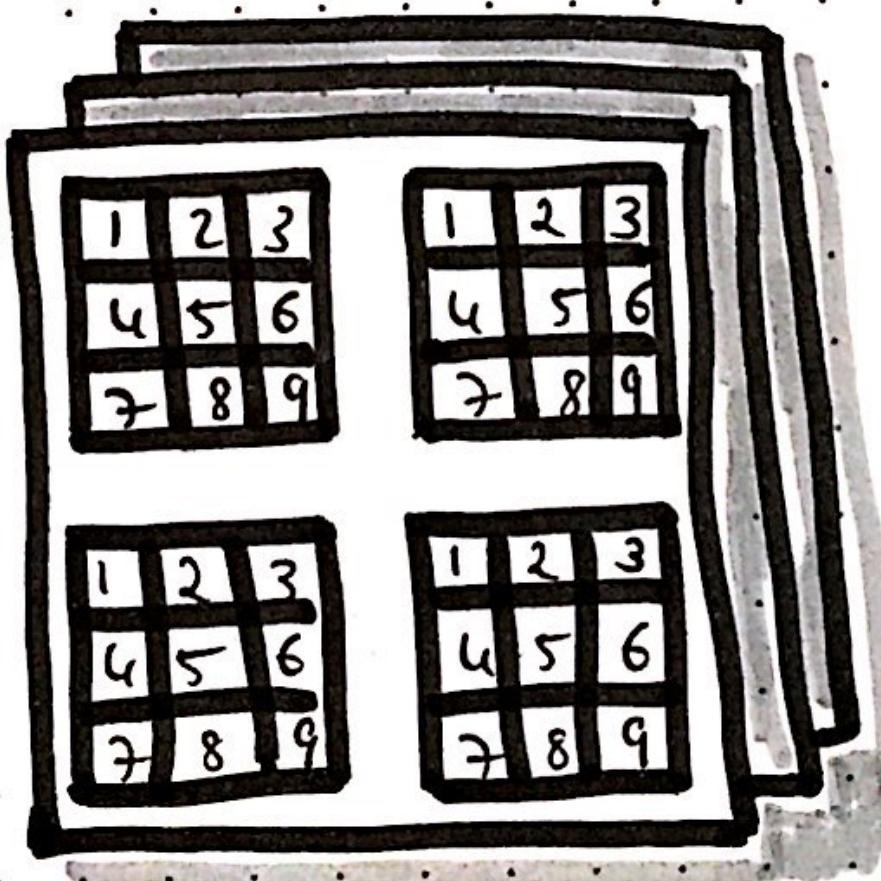
matrix

3

1	2	3
4	5	6
7	8	9

3-dim.  
array

n



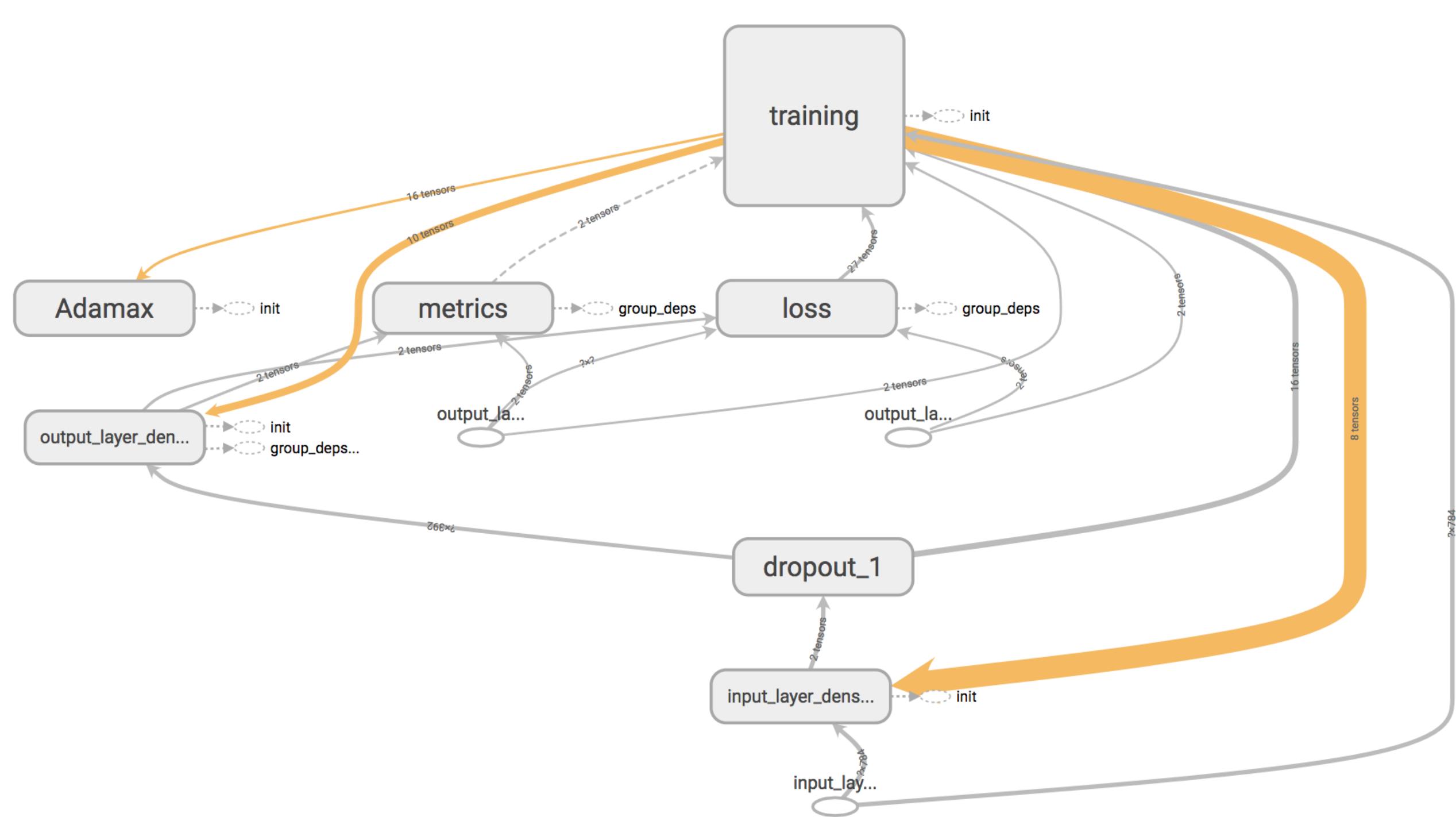
n-dim.  
array

## Tensor "Flow"



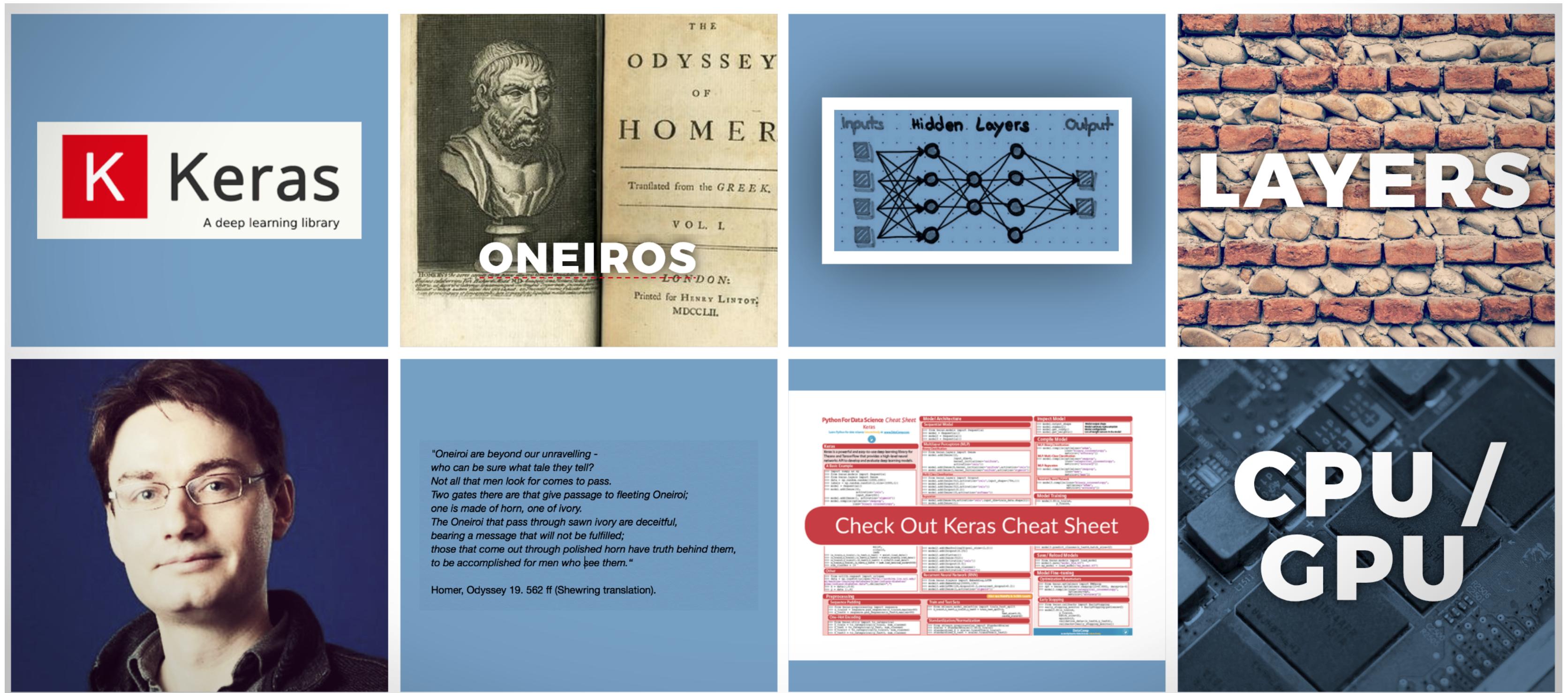
# Graphs in TensorBoard

## Main Graph

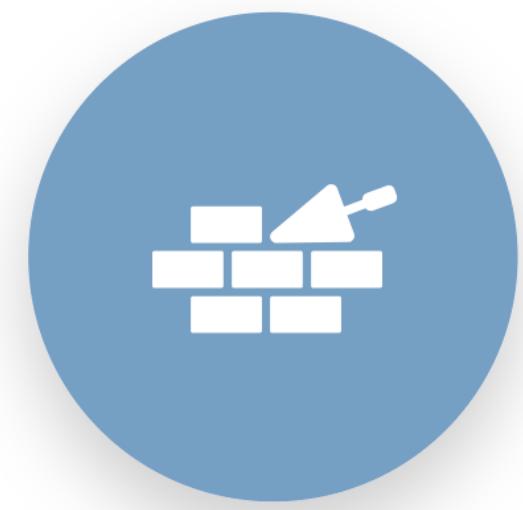


## Auxiliary Nodes

# Keras High-Level API for TensorFlow



# Keras APIs



## SEQUENTIAL MODELS

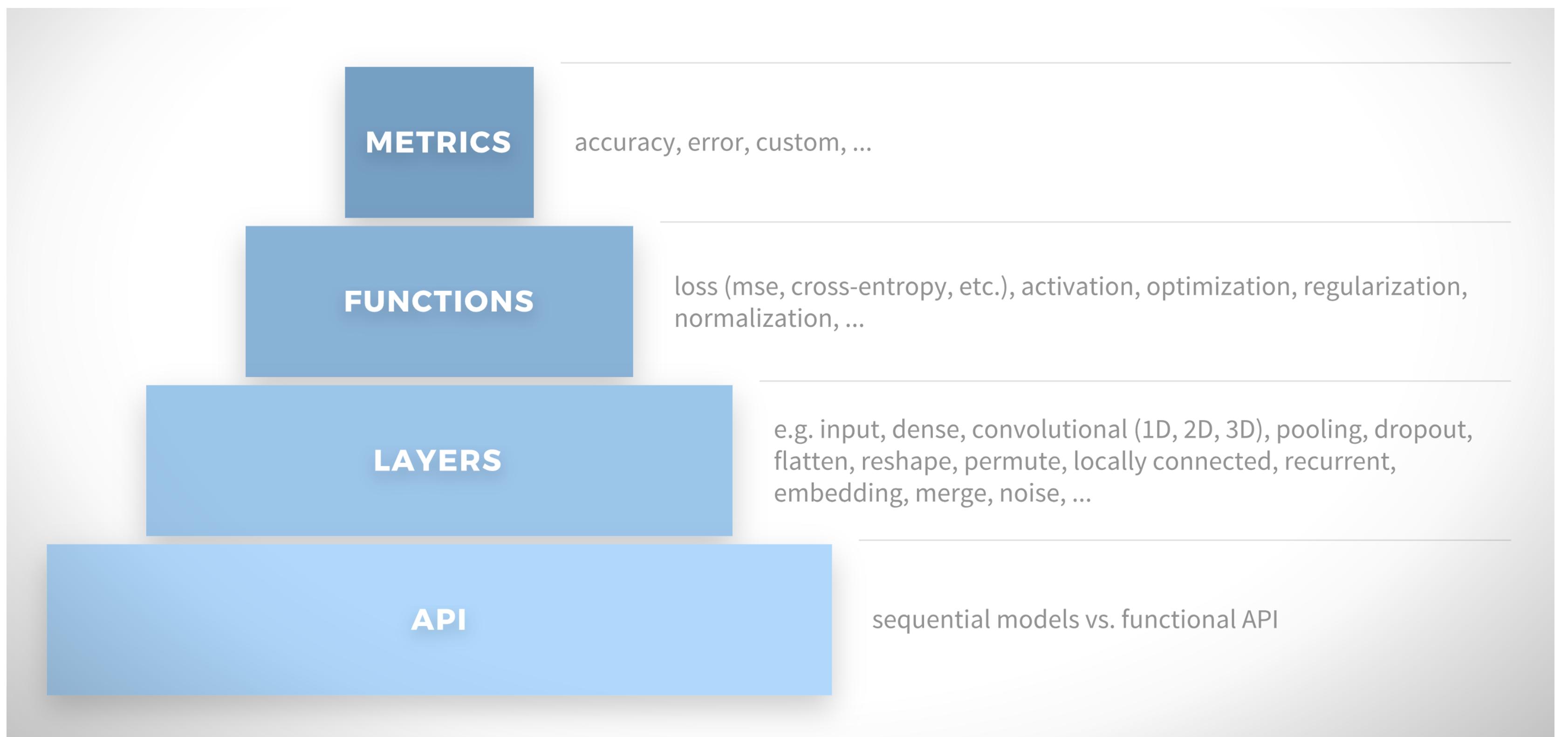
simple  
suitable for most cases  
linear order of layers  
only one direction from input to output



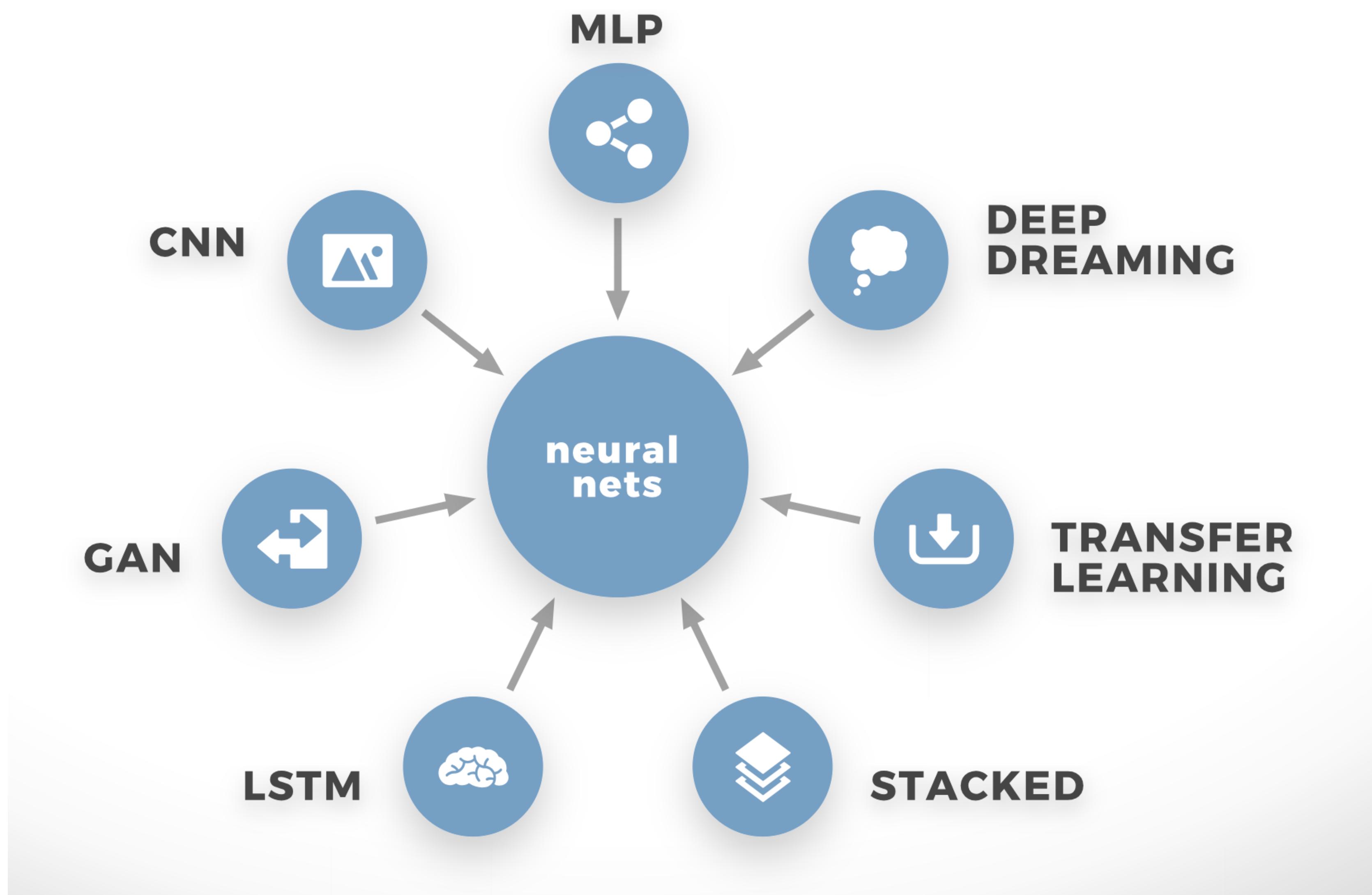
## FUNCTIONAL API

more complex  
suitable for complex models  
can have multiple in- or outputs  
layers can be non-sequential, e.g. LSTM

# Keras layers



# Endless possibilities



# Let's get our hands dirty!

## EDA

[https://github.com/ShirinG/image\\_classification\\_keras\\_tf/blob/master/notebooks/01-eda.ipynb](https://github.com/ShirinG/image_classification_keras_tf/blob/master/notebooks/01-eda.ipynb)

```
In [10]: plot_images(imgs=train_images[:10], rows=2, columns=5)
```



## Apply pretrained nets

- VGG16

[https://github.com/ShirinG/image\\_classification\\_keras\\_tf/blob/master/notebooks/02-pretrained-1.ipynb](https://github.com/ShirinG/image_classification_keras_tf/blob/master/notebooks/02-pretrained-1.ipynb)

- Xception

[https://github.com/ShirinG/image\\_classification\\_keras\\_tf/blob/master/notebooks/02-pretrained-2.ipynb](https://github.com/ShirinG/image_classification_keras_tf/blob/master/notebooks/02-pretrained-2.ipynb)

- Try out: ResNet50

# Modify pretrained nets

[https://github.com/ShirinG/image\\_classification\\_keras\\_tf/blob/master/notebooks/03-modify-pretrained.ipynb](https://github.com/ShirinG/image_classification_keras_tf/blob/master/notebooks/03-modify-pretrained.ipynb)

## Modify VGG16

- transfer learning (freeze all but the penultimate layer and re-train the last Dense layer) and
- fine tuning (un-freeze the lower convolutional layers and retrain more layers)

Validation set: fit\_generator has no option validation\_split

<https://keras.io/applications/#usage-examples-for-image-classification-models>

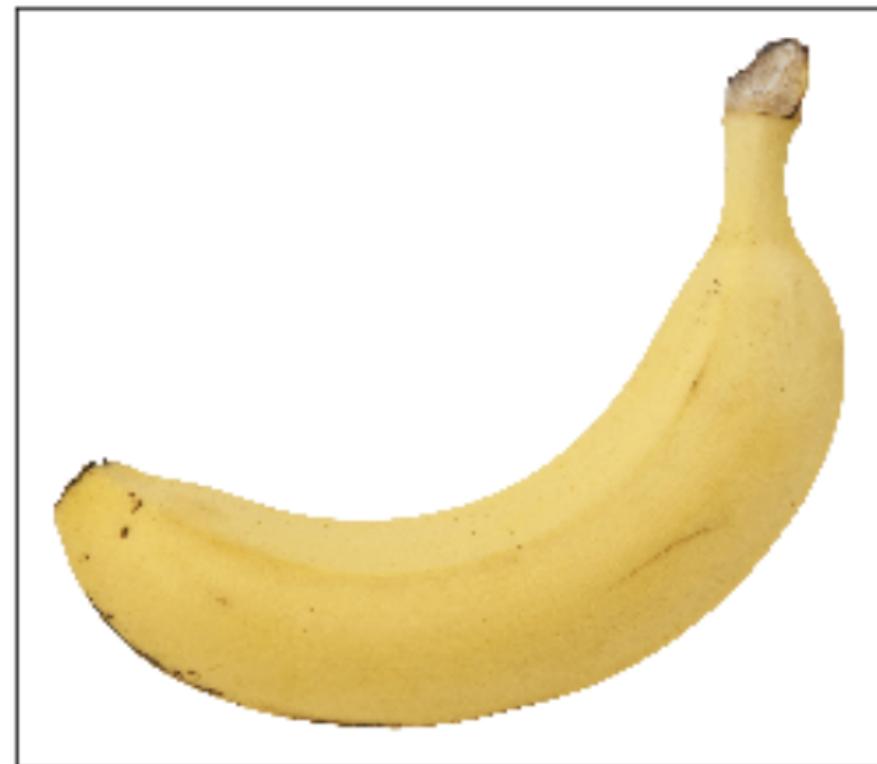
```
In [6]: # important: exclude top layers
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(75, 75, 3))
#base_model.summary()
```

```
In [7]: # Freeze the layers except the last 4 layers
for layer in base_model.layers[:-4]:
    layer.trainable = False
```

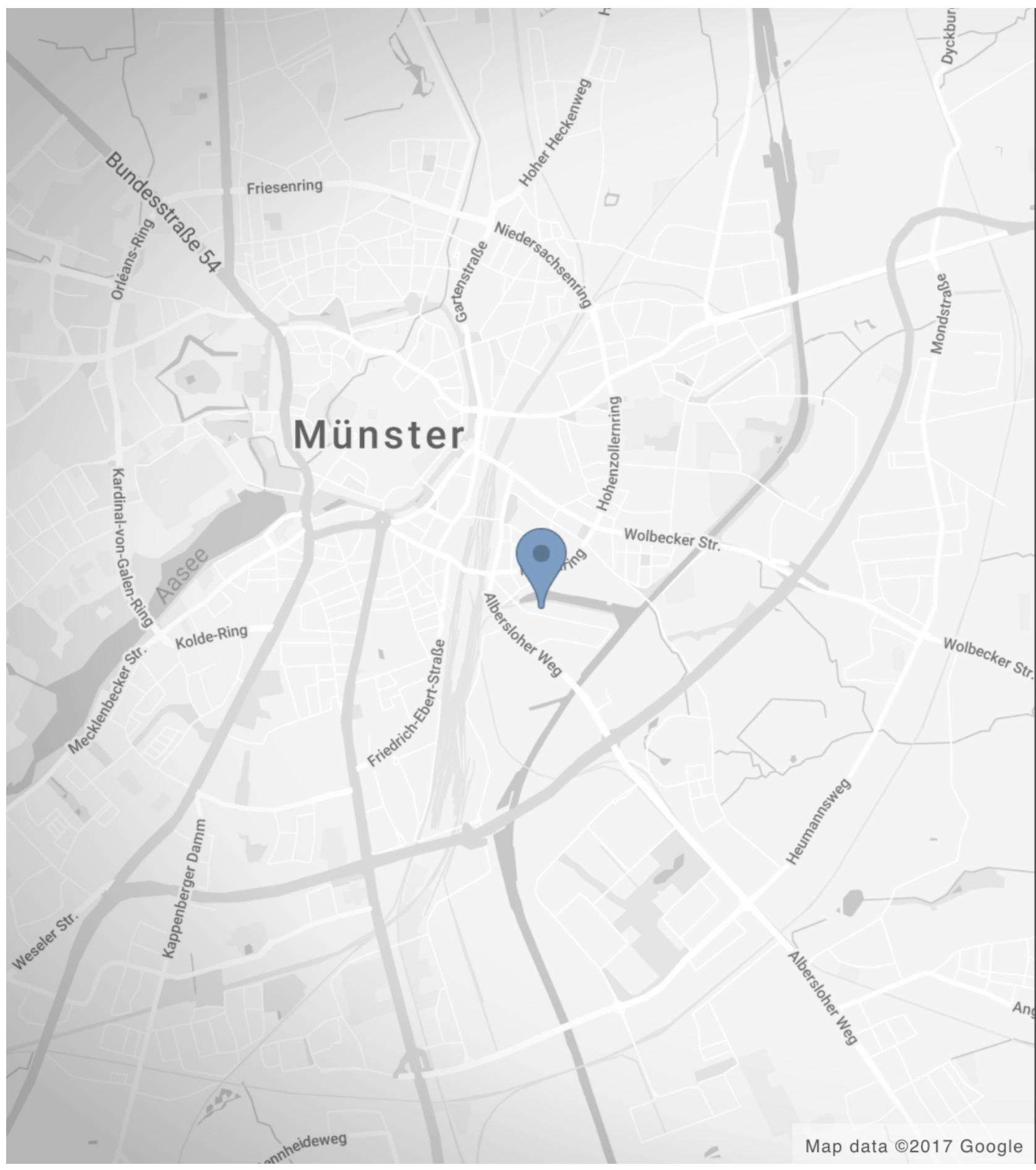
# Build your own CNN

[https://github.com/ShirinG/image\\_classification\\_keras\\_tf/blob/master/notebooks/04-fruits-cnn.ipynb](https://github.com/ShirinG/image_classification_keras_tf/blob/master/notebooks/04-fruits-cnn.ipynb)

```
In [16]: classify_image_model(test_images[0])
```



Predicted class: Banana with probability 99.99985694885254%



**Thank  
you!  
And stay  
connected**

Am Mittelhafen 14, Münster

 ShirinGlander

 shirin.glander@codecentric.de

[www.codecentric.ai](http://www.codecentric.ai)

<https://www.youtube.com/codecentricAI>

[www.shirin-glander.de](http://www.shirin-glander.de)