Workshop: Building Neural Networks with Keras and TensorFlow in R

Dr. Shirin Glander

08./09.11.2018 - Munich

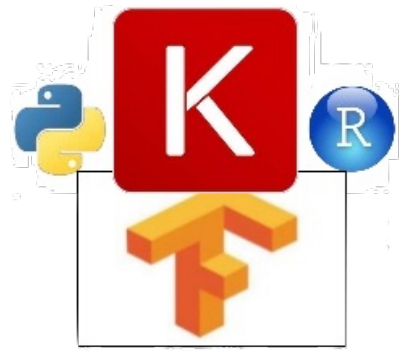
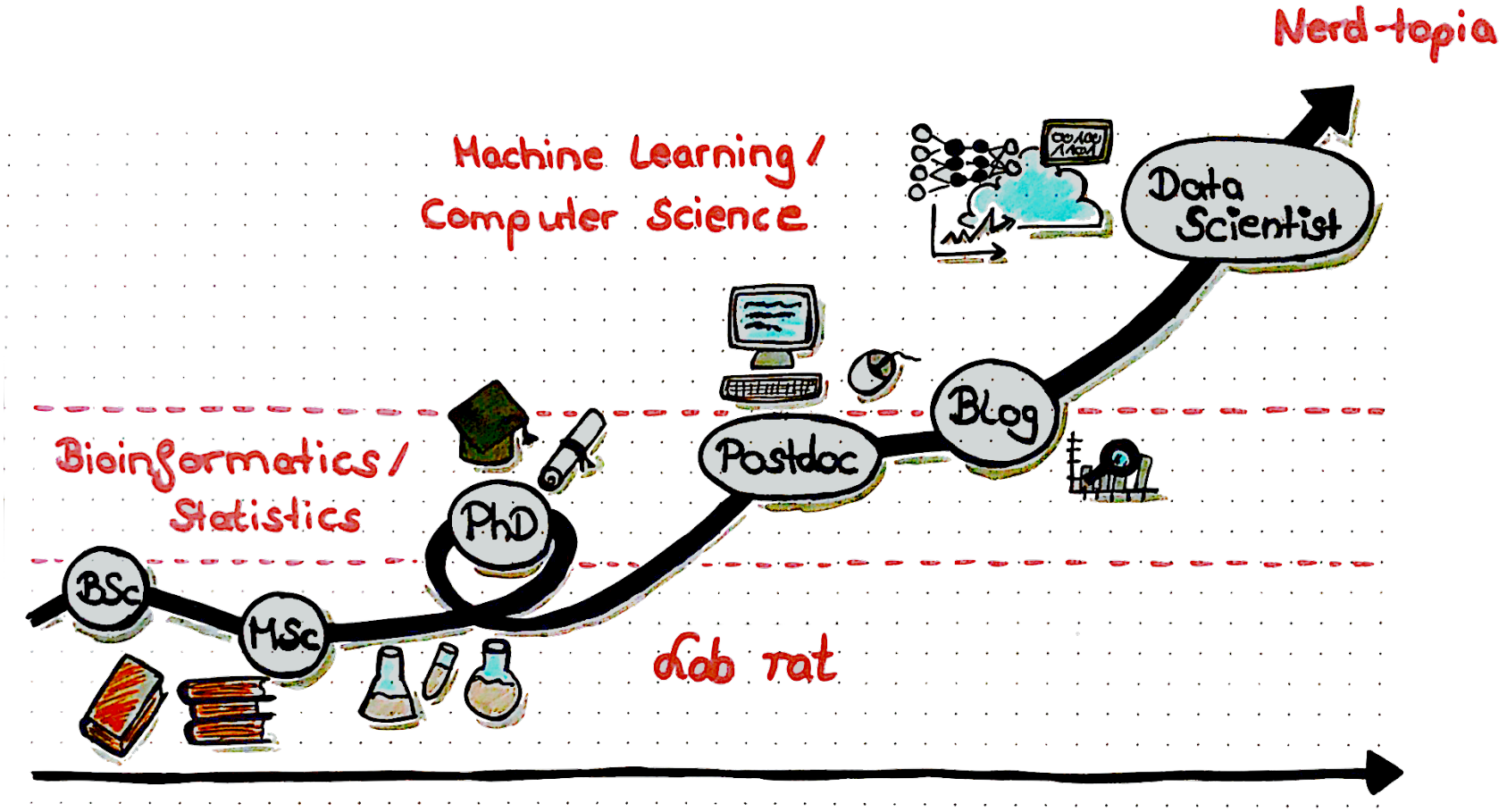


Table of Contents

About me

How I ended up here



Keras

What is Keras?

- high-level API
- library in Python that allows you to build neural networks
- works on top of TensorFlow, Theano, CNTK

Why use Keras?

- enables easy and fast prototyping
- highly modular & flexible
- supports both convolutional networks and recurrent networks, as well as combinations
- highly extensible (layers)
- runs on CPU and GPU.

About this workshop

You will learn

- the basics of deep learning
- what cross-entropy and loss is
- about activation functions
- how to optimize weights and biases with backpropagation and gradient descent
- how to build (deep) neural networks with Keras and TensorFlow
- how to save and load models and model weights
- how to visualize models with TensorBoard
- how to make predictions on test data

Workshop material

Scripts and code

```
git clone \
  https://github.com/ShirinG/workshop_keras_tensorflow_r.git
```

Keras and TensorFlow

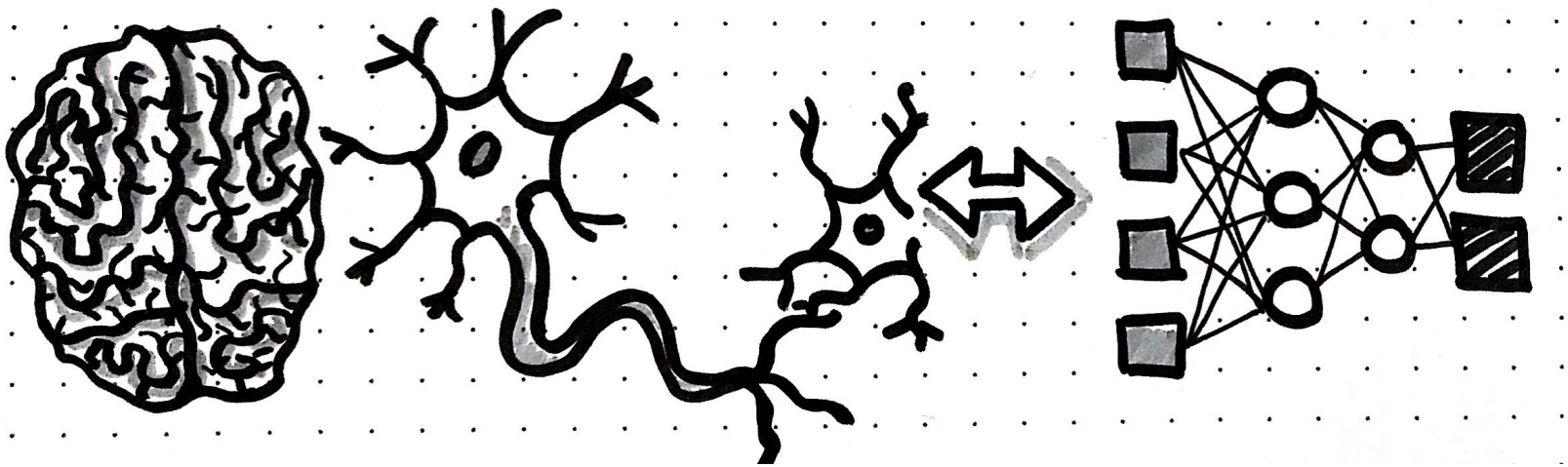
```
docker pull shiringlander/r-keras
docker run -d -p 8787:8787 \
  -v /path/to/keras_tensorflow_workshop_material:/home/rstudio/ \
  shiringlander/r-keras
```

see also 00_setup.html

Neural Networks

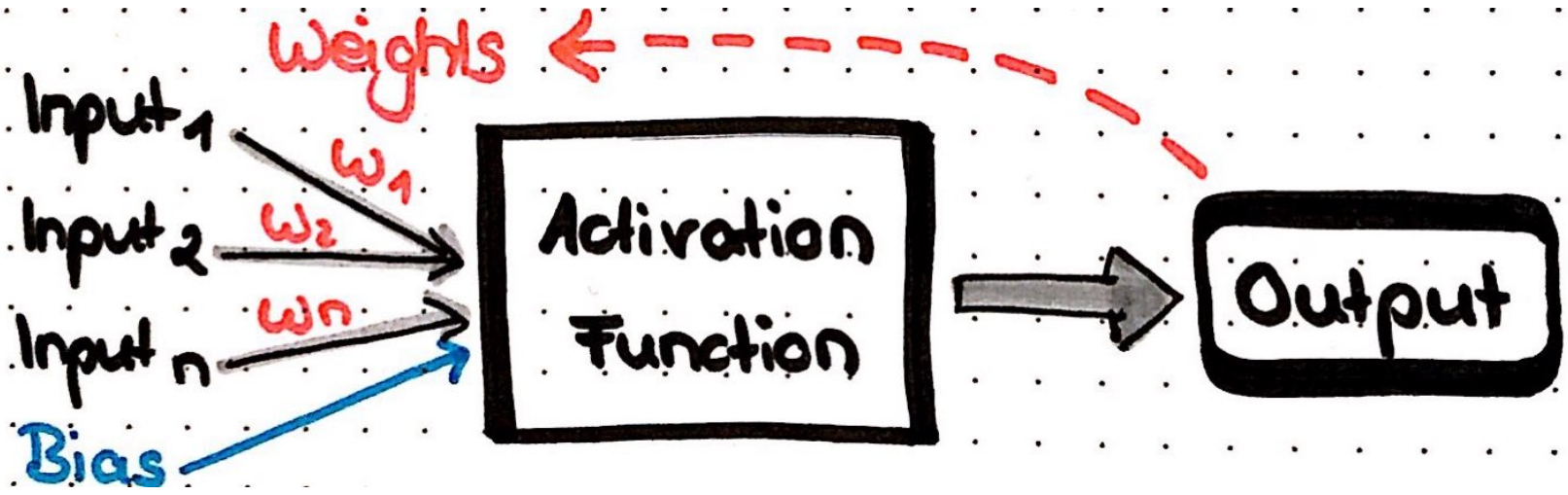
Introduction to Neural Networks (NN)

- machine learning framework
- attempts to mimic the learning pattern of the brain (biological neural networks)
- Artificial Neural Networks = ANNs

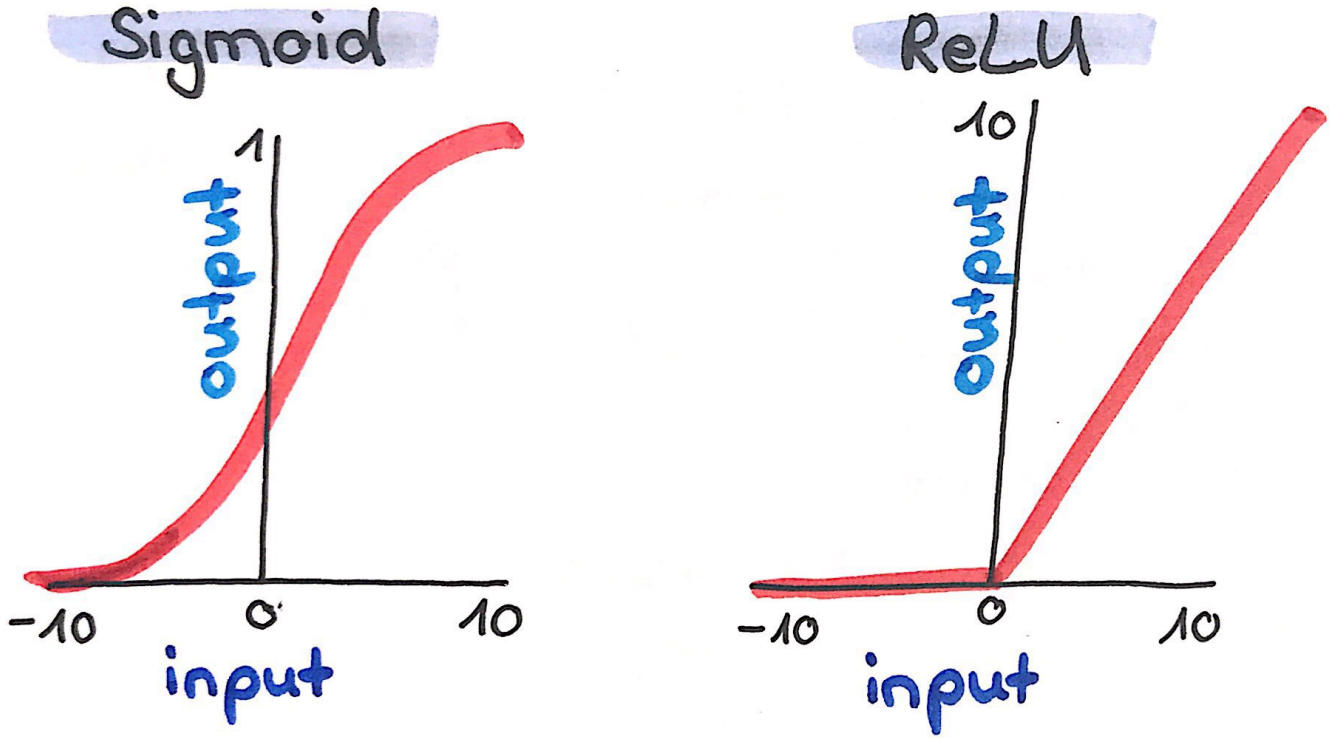


Perceptrons

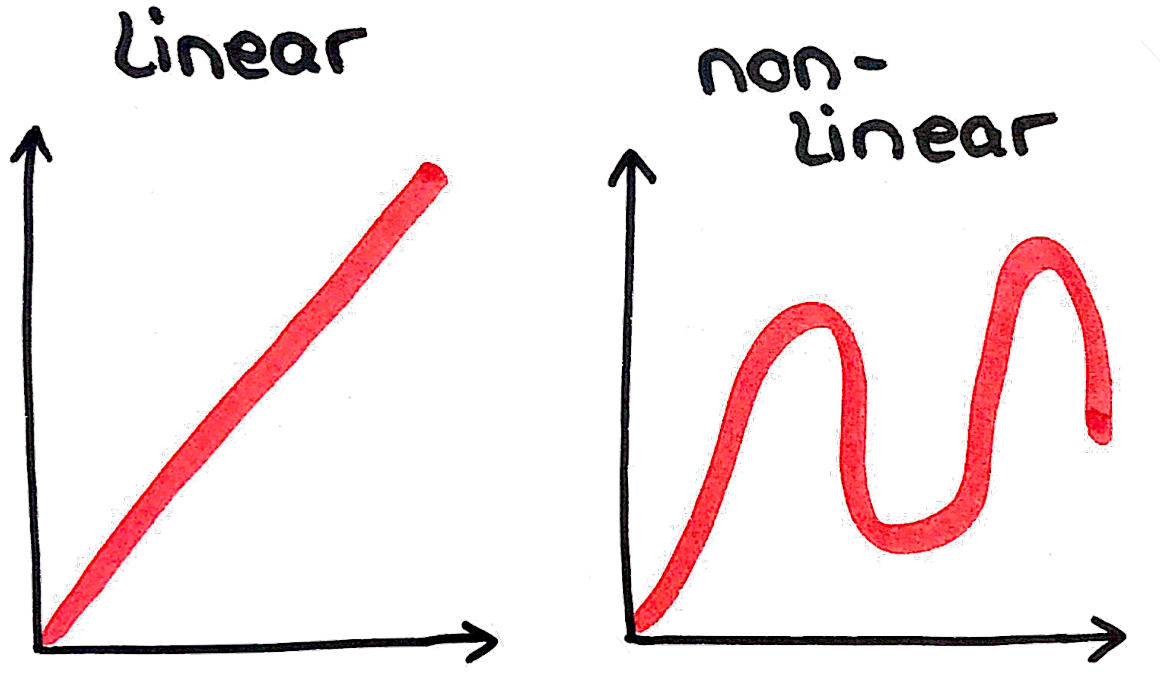
- Input (multiplied by weight)
- Bias
- Activation function
- Output



Activation functions

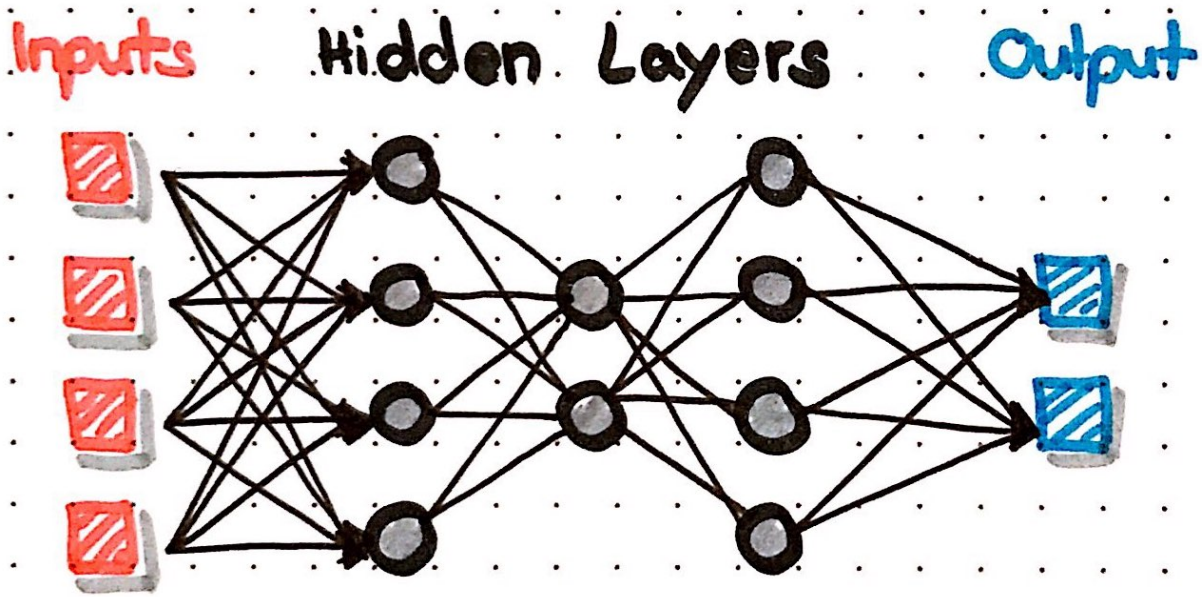


Activation functions



Multi-Layer Perceptrons (MLP)

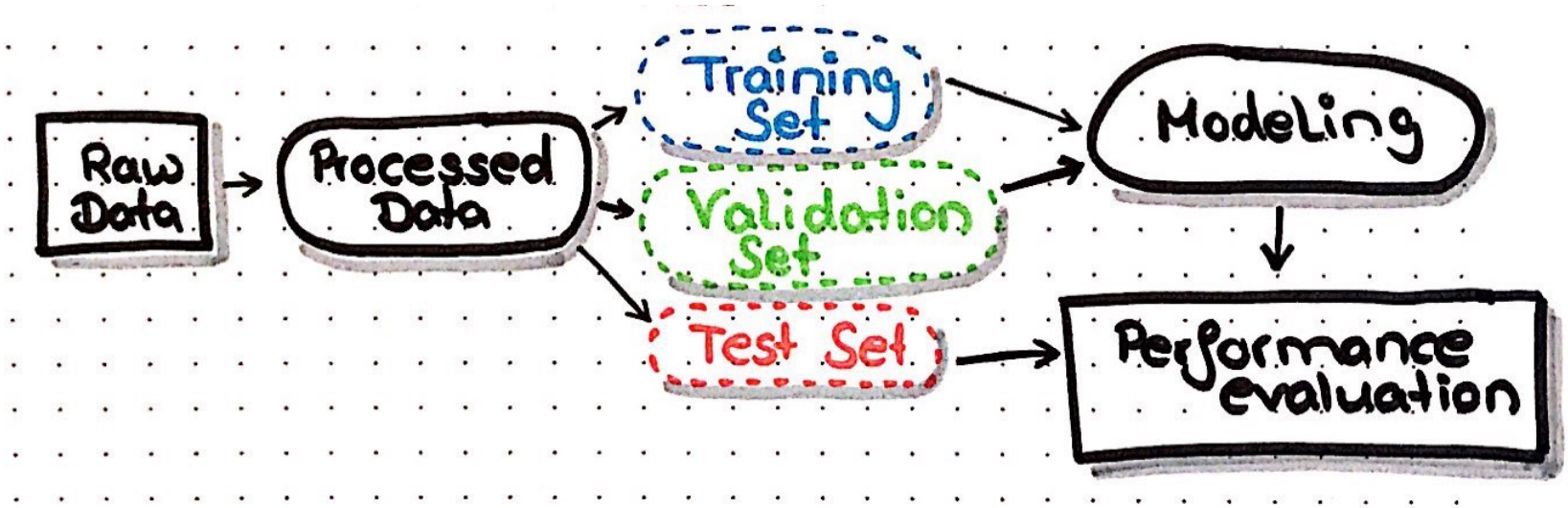
- multiple layers of perceptrons
- input
- output
- hidden layers



Practical Part 1: Neural nets in R

A general workflow

- Input data
- Data preprocessing
- Training, validation and test split
- Modeling
- Predictions and Evaluations



Why use R

- open-source, cross-platform
- established language (there are lots and lots of packages)
- high quality packages with proper documentation (CRAN)
- graphics capabilities - ggplot2!!!
- RStudio + R Markdown!!
- community support!

Features & data preprocessing

```
library(ISLR)
```

```
print(head(College))
#summary(College)
```

	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Board	Books	Personal	PhD	Terminal	S.F.Ratio	perc.alumni	Expend	Grad.Rate
Abilene Christian University	Yes	1660.00	1232.00	721.00	23.00	52.00	2885.00	537.00	7440.00	3300.00	450.00	2200.00	70.00	78.00	18.10	12.00	7041.00	60.00
Adelphi University	Yes	2186.00	1924.00	512.00	16.00	29.00	2683.00	1227.00	12280.00	6450.00	750.00	1500.00	29.00	30.00	12.20	16.00	10527.00	56.00
Adrian College	Yes	1428.00	1097.00	336.00	22.00	50.00	1036.00	99.00	11250.00	3750.00	400.00	1165.00	53.00	66.00	12.90	30.00	8735.00	54.00
Agnes Scott College	Yes	417.00	349.00	137.00	60.00	89.00	510.00	63.00	12960.00	5450.00	450.00	875.00	92.00	97.00	7.70	37.00	19016.00	59.00
Alaska Pacific University	Yes	193.00	146.00	55.00	16.00	44.00	249.00	869.00	7560.00	4120.00	800.00	1500.00	76.00	72.00	11.90	2.00	10922.00	15.00
Albertson College	Yes	587.00	479.00	158.00	38.00	62.00	678.00	41.00	13500.00	3335.00	500.00	675.00	67.00	73.00	9.40	11.00	9727.00	55.00

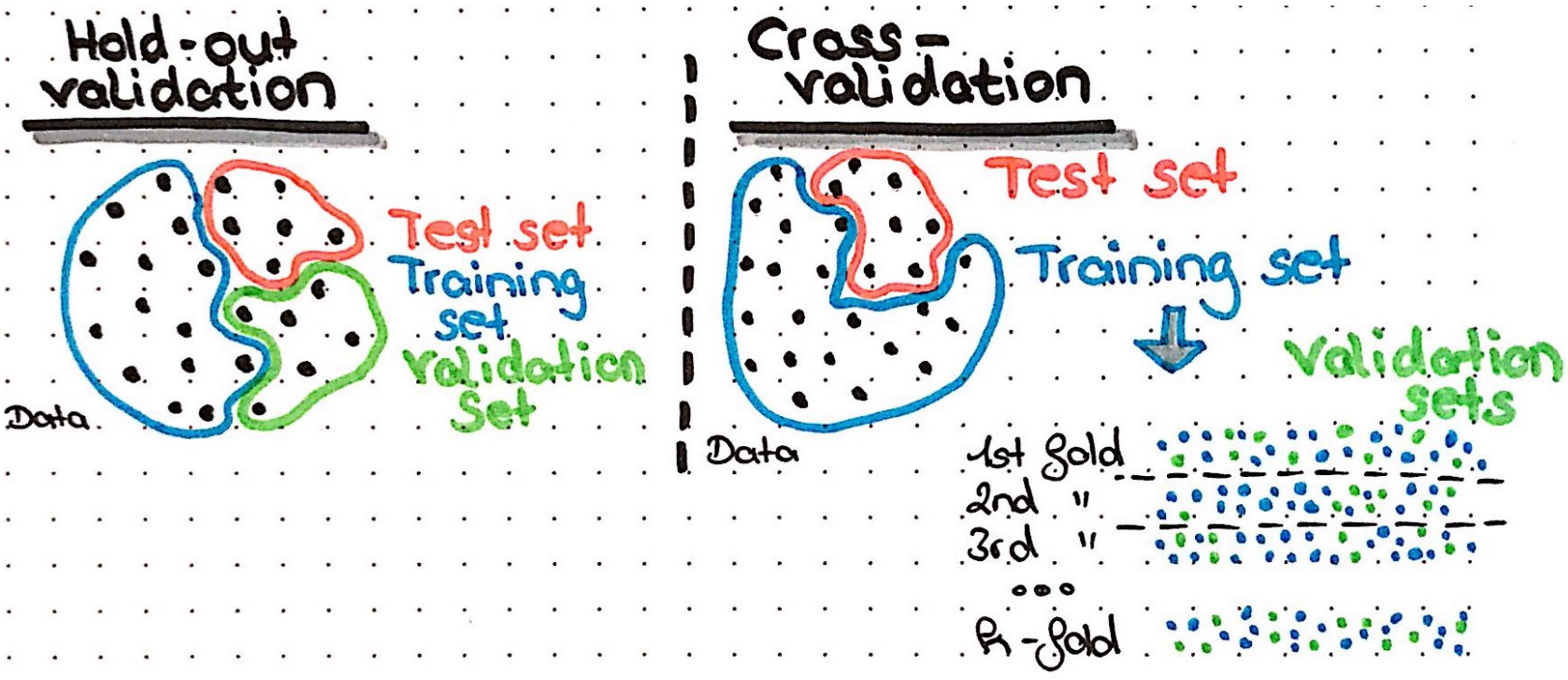
```
# Create vector of column max and min values
maxs <- apply(College[, 2:18], 2, max)
mins <- apply(College[, 2:18], 2, min)

# Use scale() and convert the resulting matrix to a data frame
scaled.data <- scale(College[, 2:18],
                     center = mins,
                     scale = maxs - mins) %>%
  as.data.frame()
```

Training, validation & testing

- balance between generalization and specificity
- to prevent over-fitting!
- validation sets or cross-validation

Training, validation & testing



Train and test split

```
# Convert column Private from Yes/No to 1/0
Private <- as.numeric(College$Private)-1
data <- cbind(Private, scaled.data)

summary(as.factor(data$Private))
```

```
##   0   1
## 212 565
```

```
library(caret)
set.seed(42)

# Create split
idx <- createDataPartition(data$Private, p = .75, list = FALSE)
train <- data[idx, ]
test  <- data[-idx, ]
```

Training with caret

```
?train
```

This function sets up a grid of tuning parameters for a number of classification and regression routines, fits each model and calculates a resampling based performance measure.

Modeling

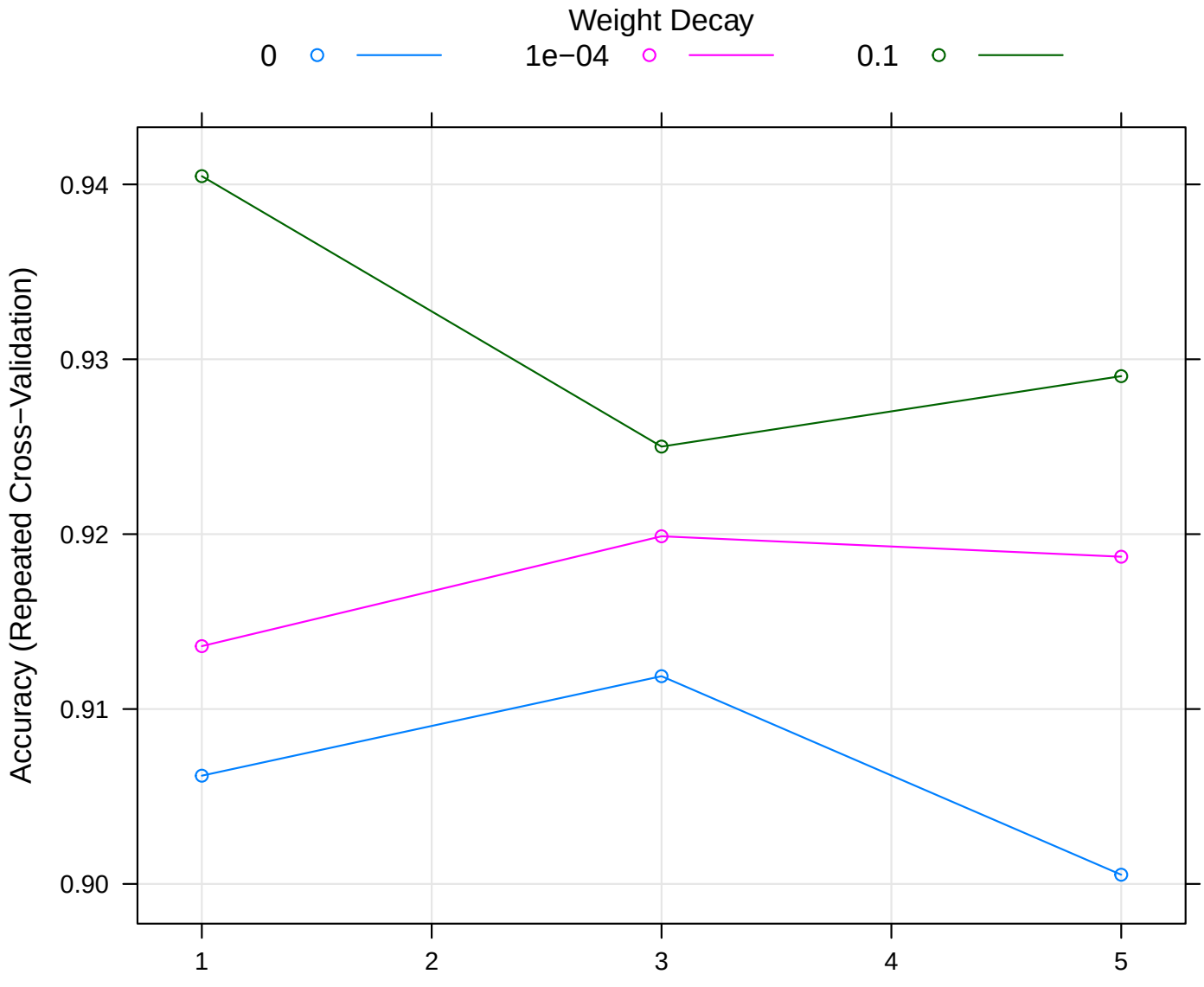
```
# 3 x 5 repeated CV
fitControl <- trainControl(method = "repeatedcv",
                           number = 5,
                           repeats = 3,
                           savePredictions = TRUE)
```

```
set.seed(42)
nn <- train(factor(Private) ~ .,
            data = train,
            method = "nnet",
            preProcess = c("scale", "center"),
            trControl = fitControl,
            verbose= FALSE)
```

```
## # weights:  20
## initial  value 305.082701
## iter  10 value 102.529848
## iter  20 value 72.014007
## iter  30 value 60.691048
```

Plotting

```
plot(nn)
```



Predictions and Evaluations

```
#predicted.nn.values <- predict(nn, test[, 2:18], type = "prob")
predicted.nn.values <- predict(nn, test[, 2:18], type = "raw")

cbind(predicted.nn.values, test) %>%
  count(predicted.nn.values, Private)
```

```
## # A tibble: 4 x 3
##   predicted.nn.values Private     n
##   <fct>               <dbl> <int>
## 1 0                       0    46
## 2 0                       1     5
## 3 1                       0     4
## 4 1                       1   139
```

```
# print results
print(head(predicted.nn.values, 3))
```

```
## [1] 1 1 1
## Levels: 0 1
```

Confusion Matrix

```
caret::confusionMatrix(predicted.nn.values, as.factor(test$Private))
```

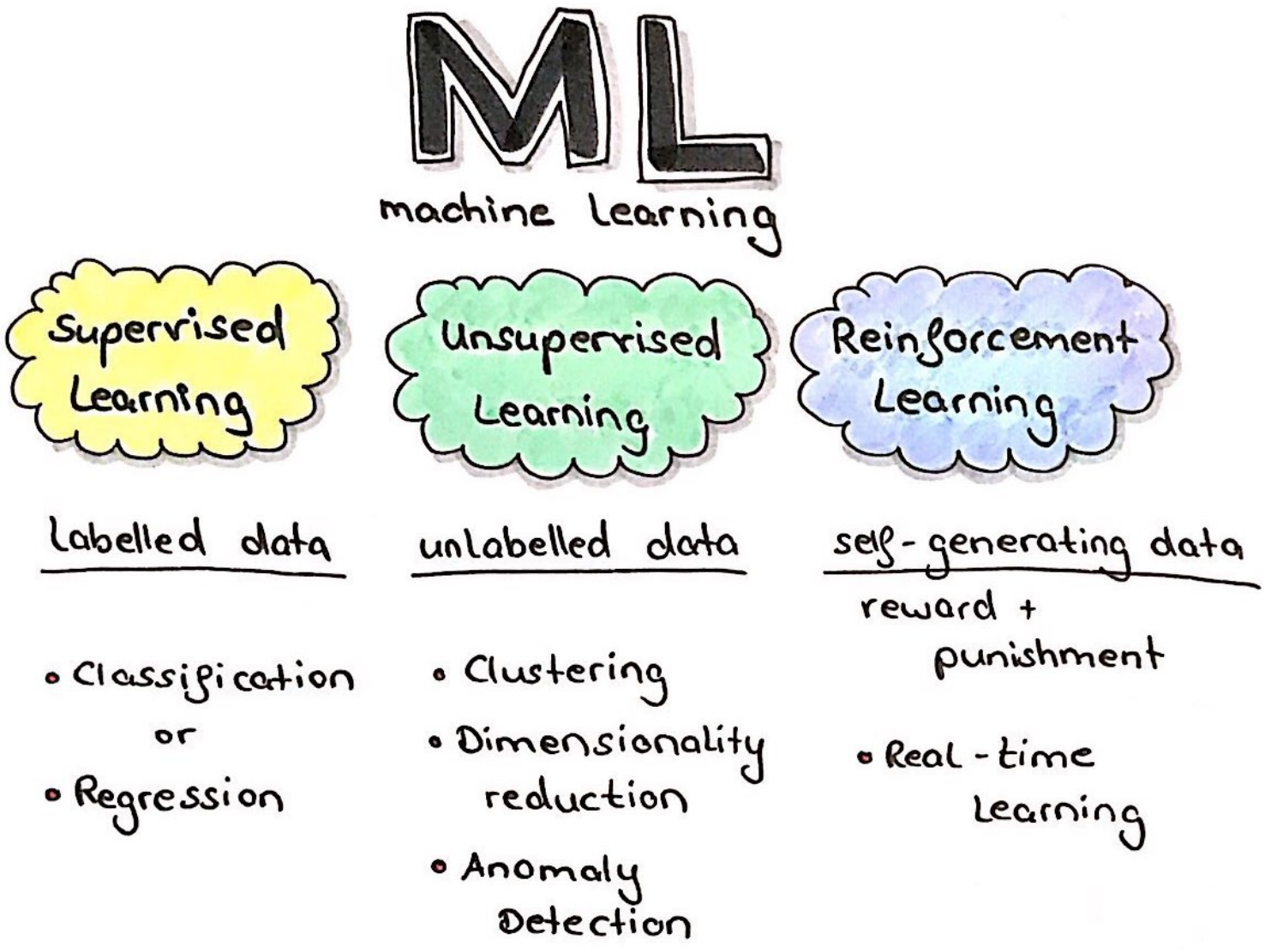
```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction   0   1
##          0  46   5
##          1   4 139
##
##                Accuracy : 0.9536
##                  95% CI : (0.9138, 0.9786)
##     No Information Rate : 0.7423
##     P-Value [Acc > NIR] : 5.846e-15
##
##                   Kappa : 0.8795
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9200
##             Specificity : 0.9653
```

...continue on your own...

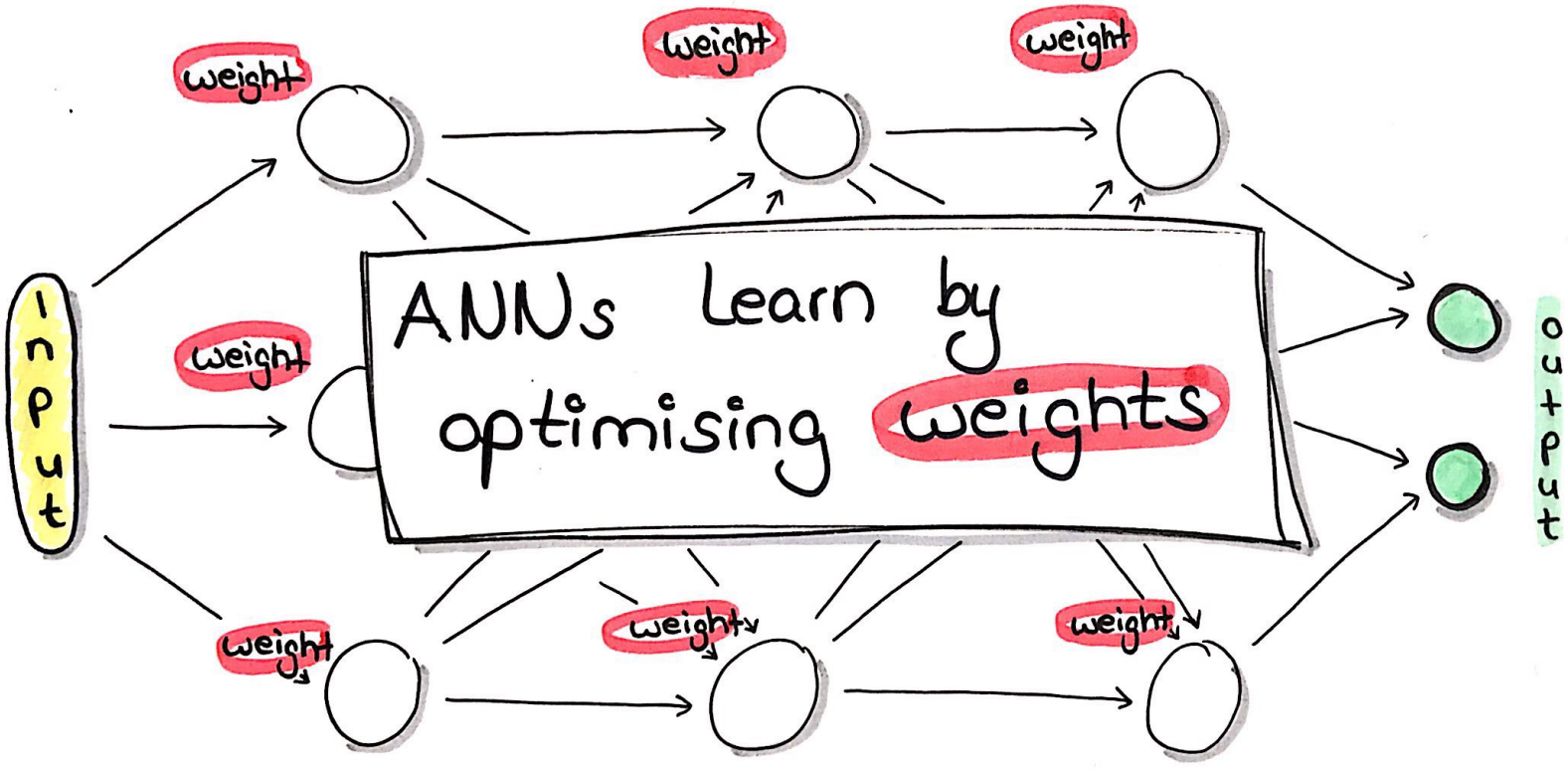
- get acquainted with R
- try out different hidden layer combinations
- set other hyperparameters in the train() function (see ?train)
- try different ratios of training and test data
- explore different normalization techniques, e.g. dividing by the maximum value of each column
- ...

How do Neural Nets learn?

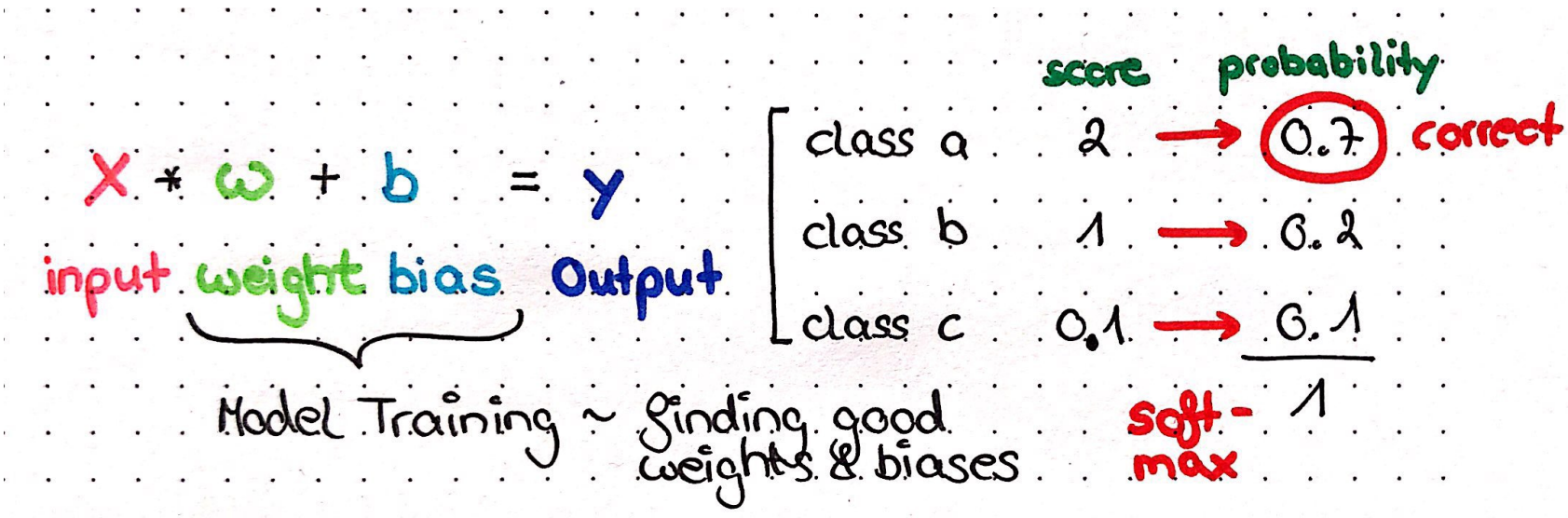
Classfication with supervised learning



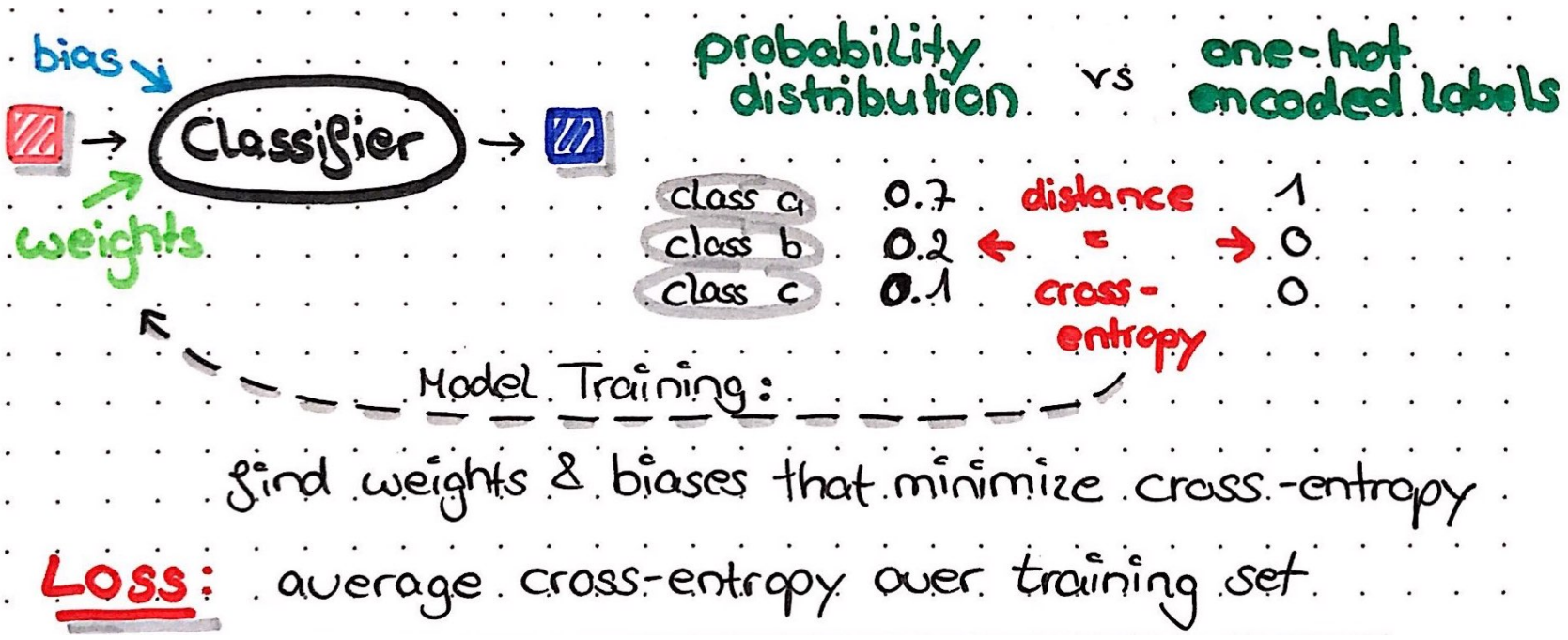
How do Neural Nets learn?



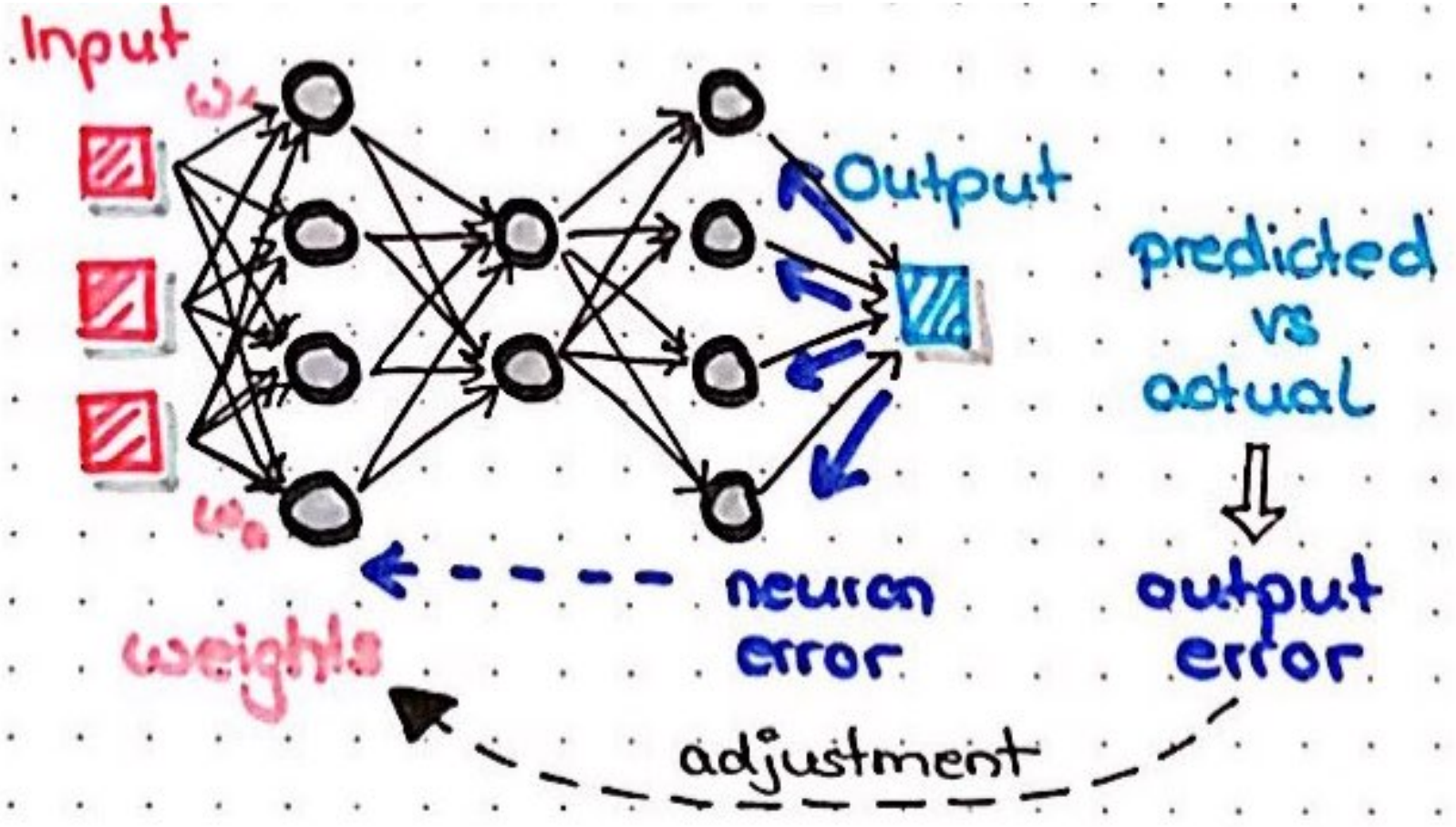
Softmax


X * ω + 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
1
soft-max

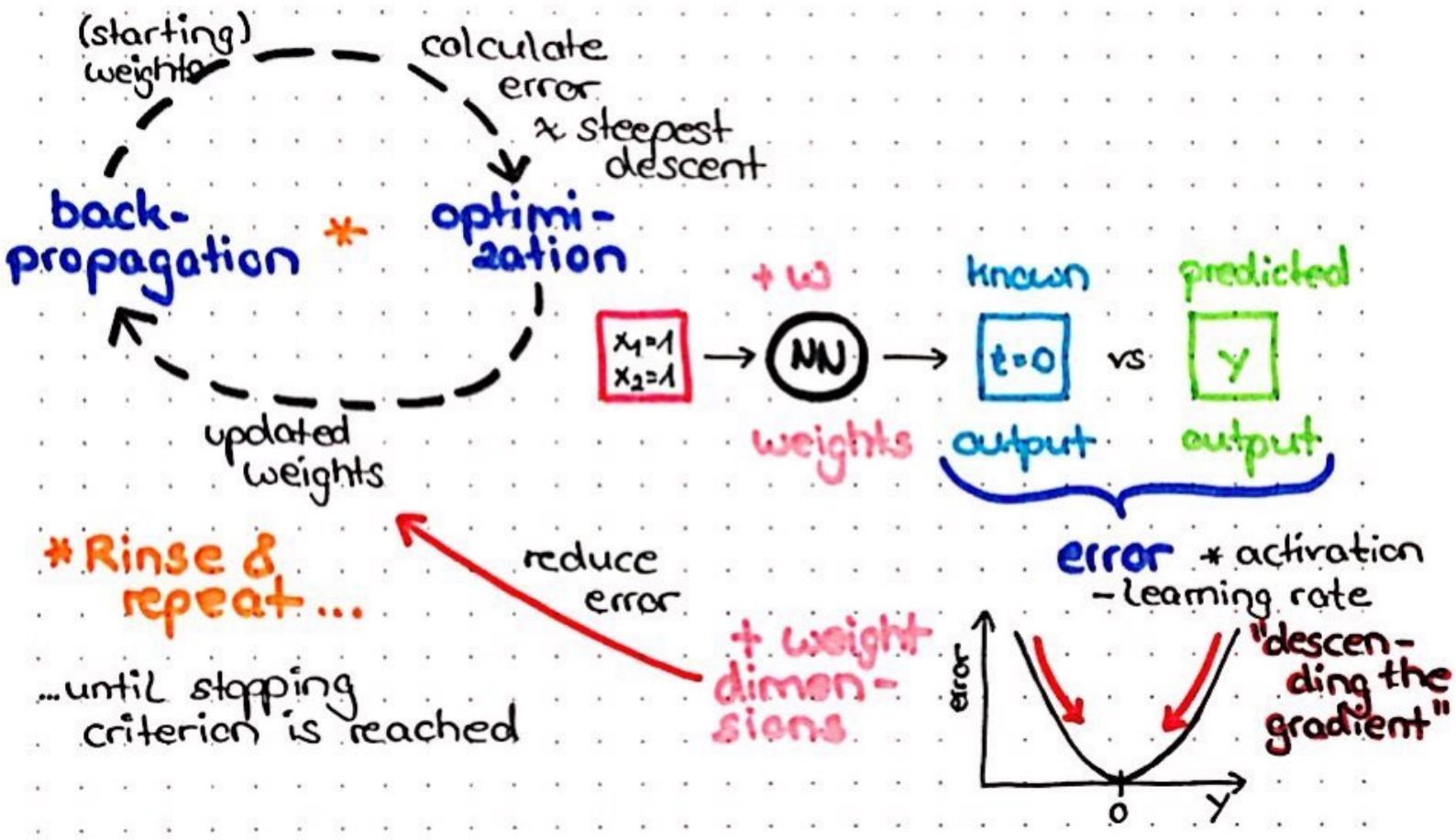
Cross-entropy



Backpropagation



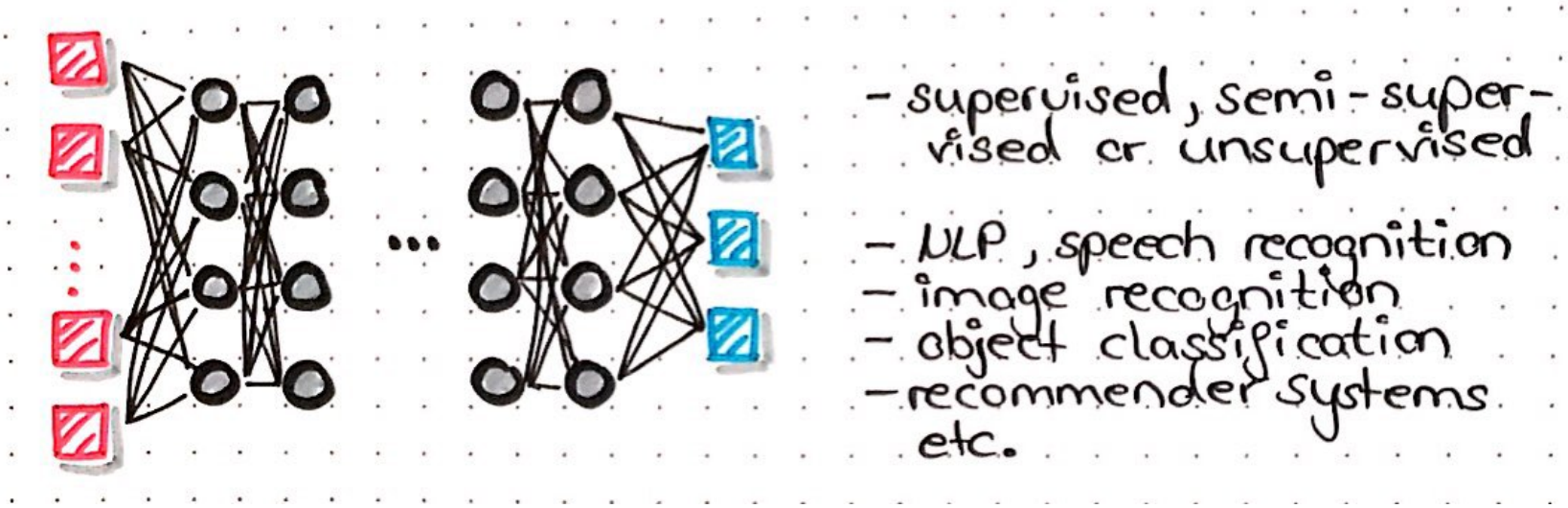
Gradient Descent Optimization



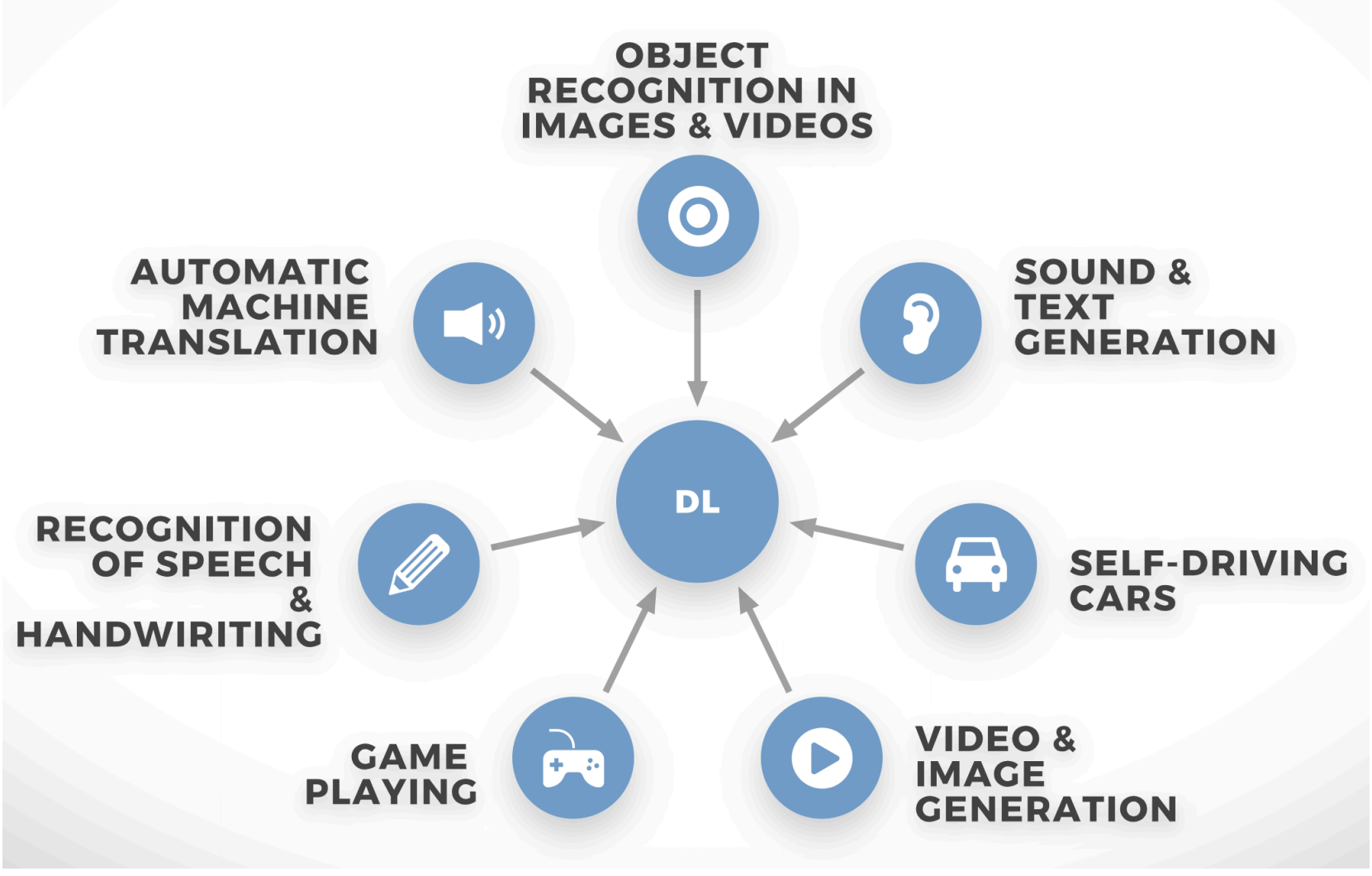
Deep Learning with Keras & TensorFlow

Deep Neural Networks

- loosely defined as a NN with more than 2 hidden layers
- use for COMPLEX problems!



Deep Neural Networks



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

Why Keras + TensorFlow?

The Keras philosophy

- Rapid prototyping
- Fast turnover from idea to model

Keras is the default TensorFlow API

- Keras is the first high-level API added to core TF at Google

Start small and simple

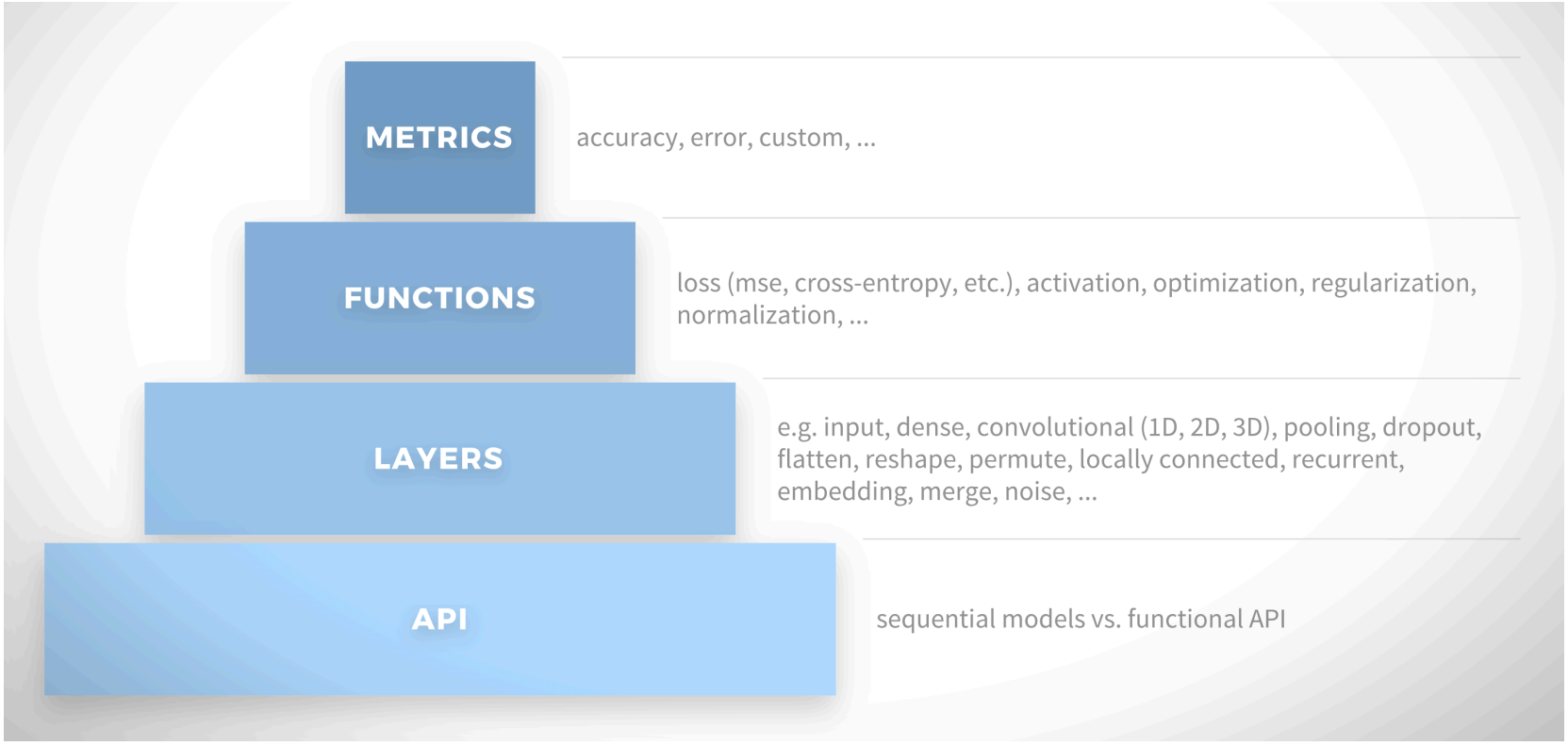
- Start simple and small to have a baseline against which to compare

Keras Basics

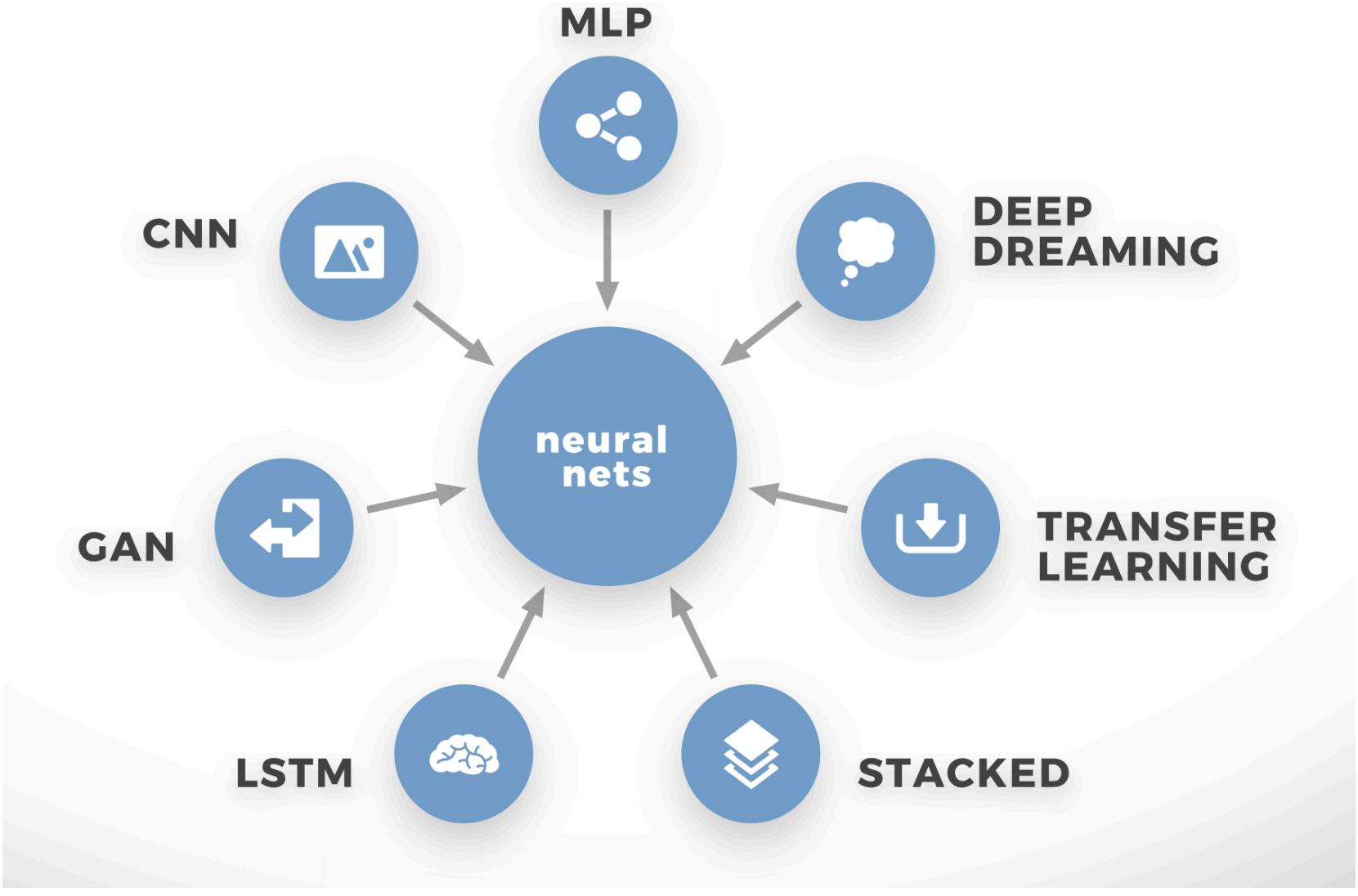
You are dealing with layers:

- input layer ... output layer
- dropout layer
- noise layer
- pooling layer
- normalization layer
- embedding layer
- activation function for each layer
- regularization techniques
- optimization functions
- you can save models, layers, weights

Keras layers



Keras possibilities



TensorFlow

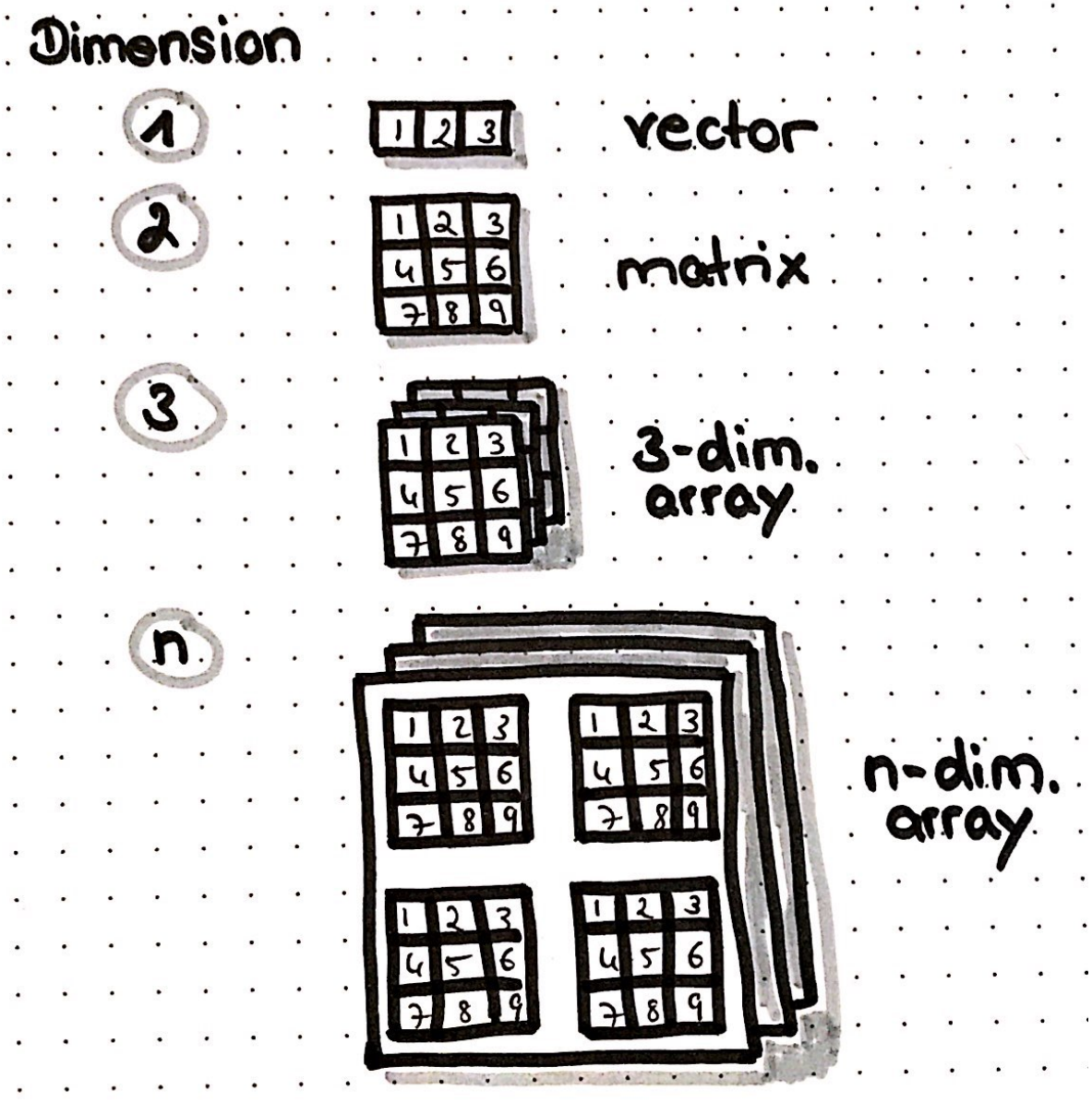
TensorFlow

- originally developed by Google Brain
- open-source software for deep learning
- based on flow charts / graphs

Tensors

- tensors ~ multidimensional arrays
- images can be converted to arrays
- 2D for black-and-white, 3x 2D for RGB
- can be processed in PARALLEL

Tensors

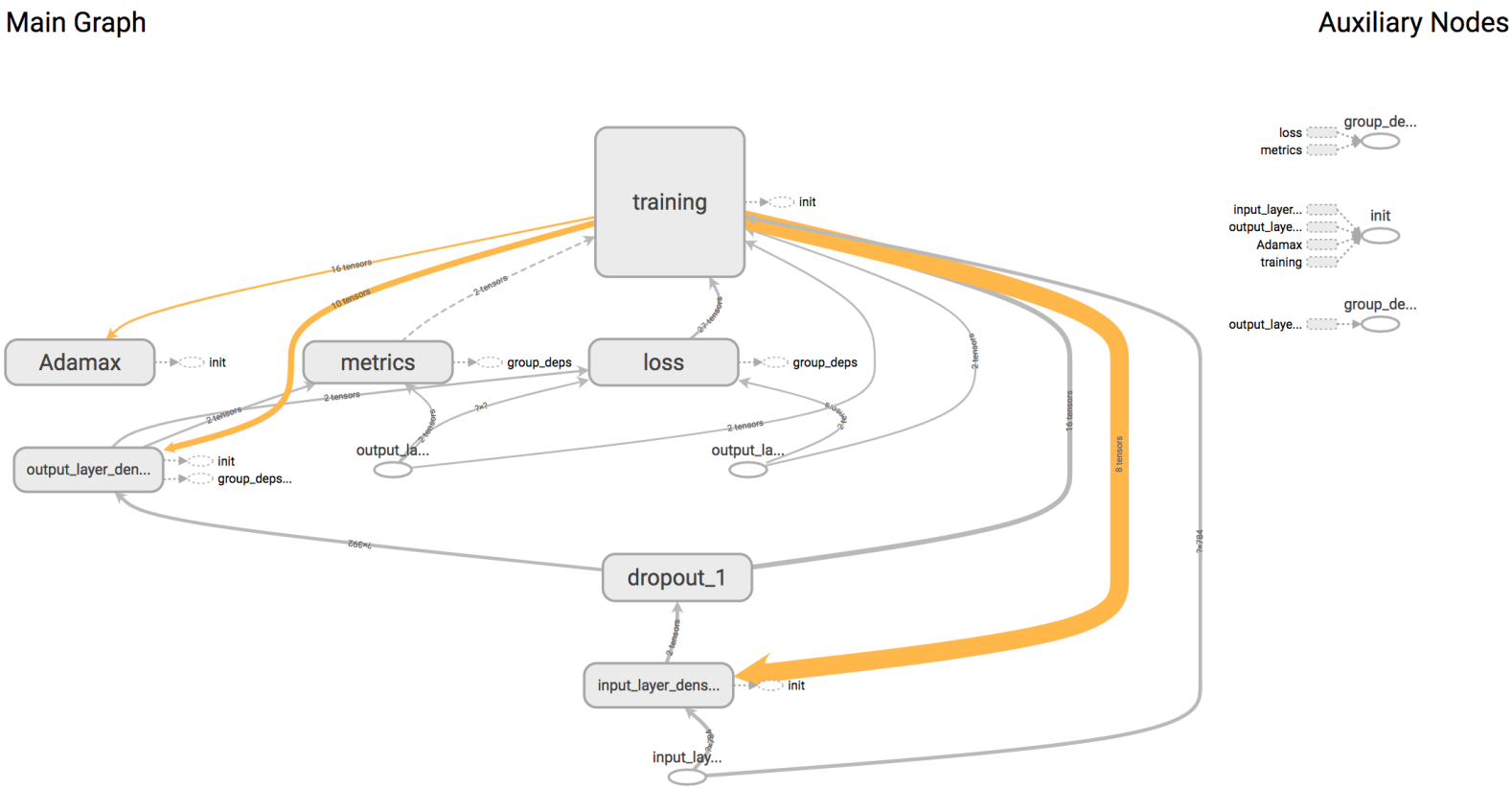


Graphs

- computational graphs to represent operations
- each node takes zero or more tensors as inputs and ...
- ... performs a mathematical operation ...
- ... to produce a tensor as an output
- visualization with TensorBoard

TensorBoard

- suite of visualization tools
- visualize TensorFlow graph,
- plot quantitative metrics about the execution of your graph,
- etc.



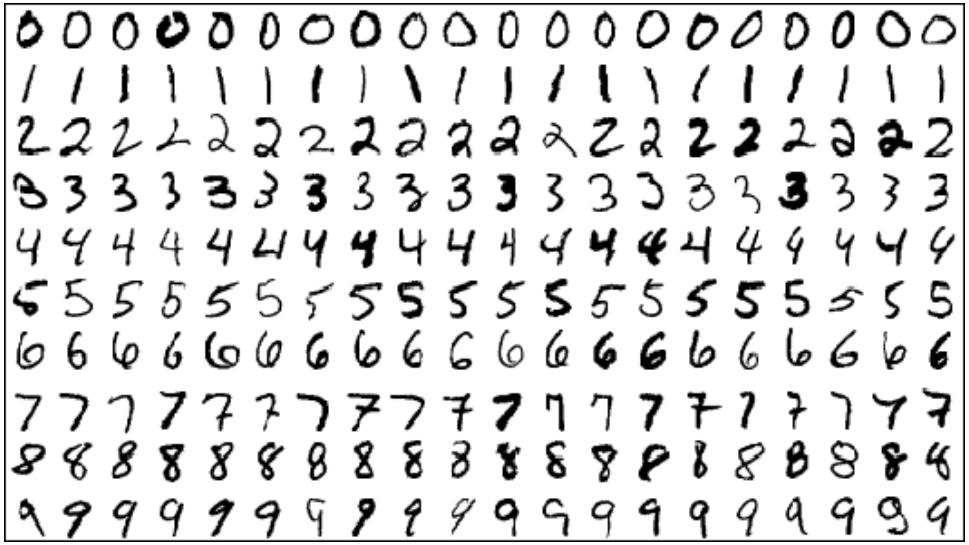
Practical Part 2: Sequential models with Keras

Sequential model

- linear stack of layers
- input layer -> hidden layer -> output layer

The MNIST example

- MNIST handwritten digits dataset
- labels: numbers from 0 - 9
- images: 28x28 pixels



...continue on your own...

- try out different hidden layer combinations
- change & set other hyperparameters in the layers, e.g. learning rate
- change optimizer
- change activation function
- change epochs
- explore different normalization techniques, e.g. scale()
- ...

Practical Part 3: Convolutional Neural Networks

How does a computer learn to see?

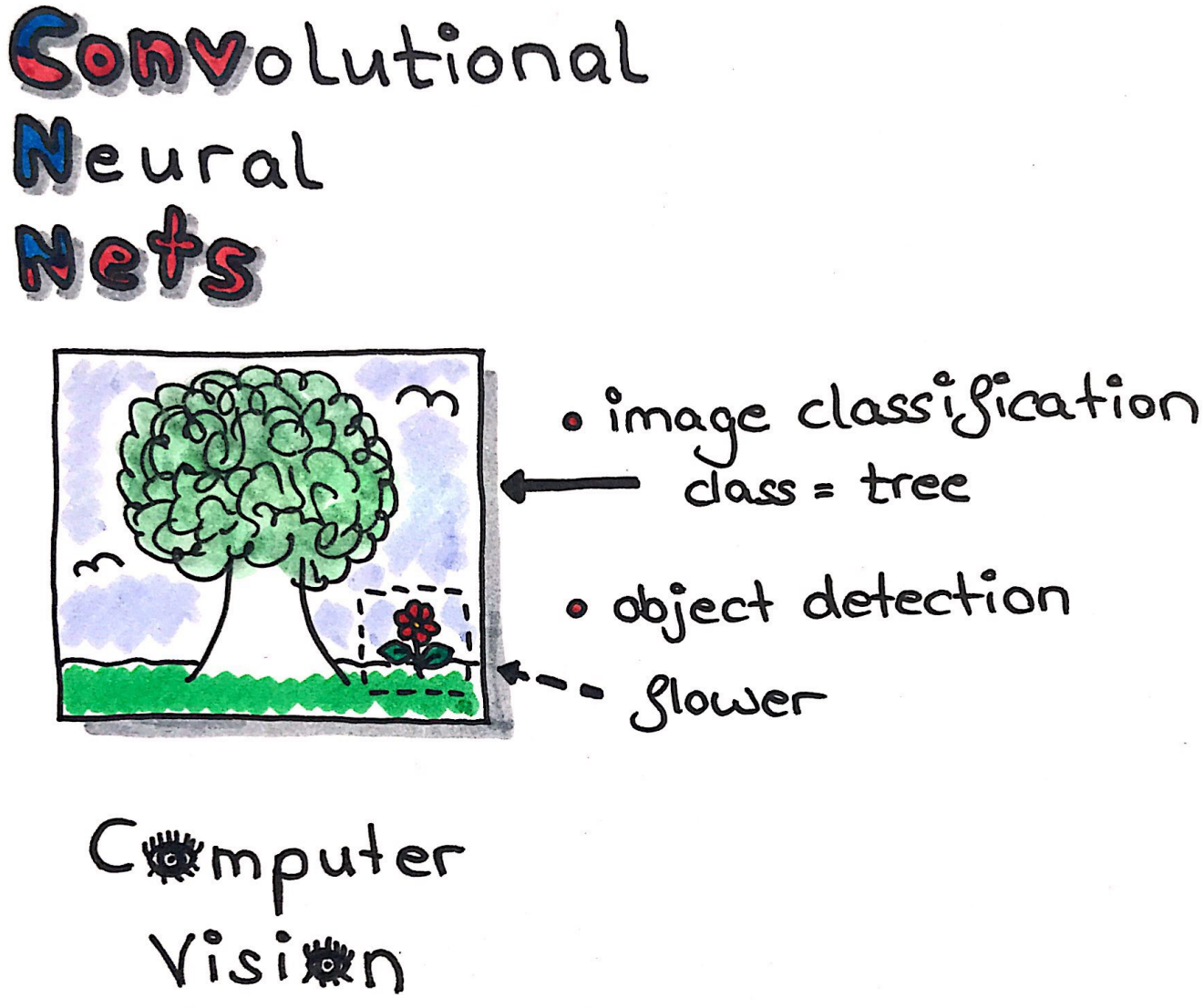
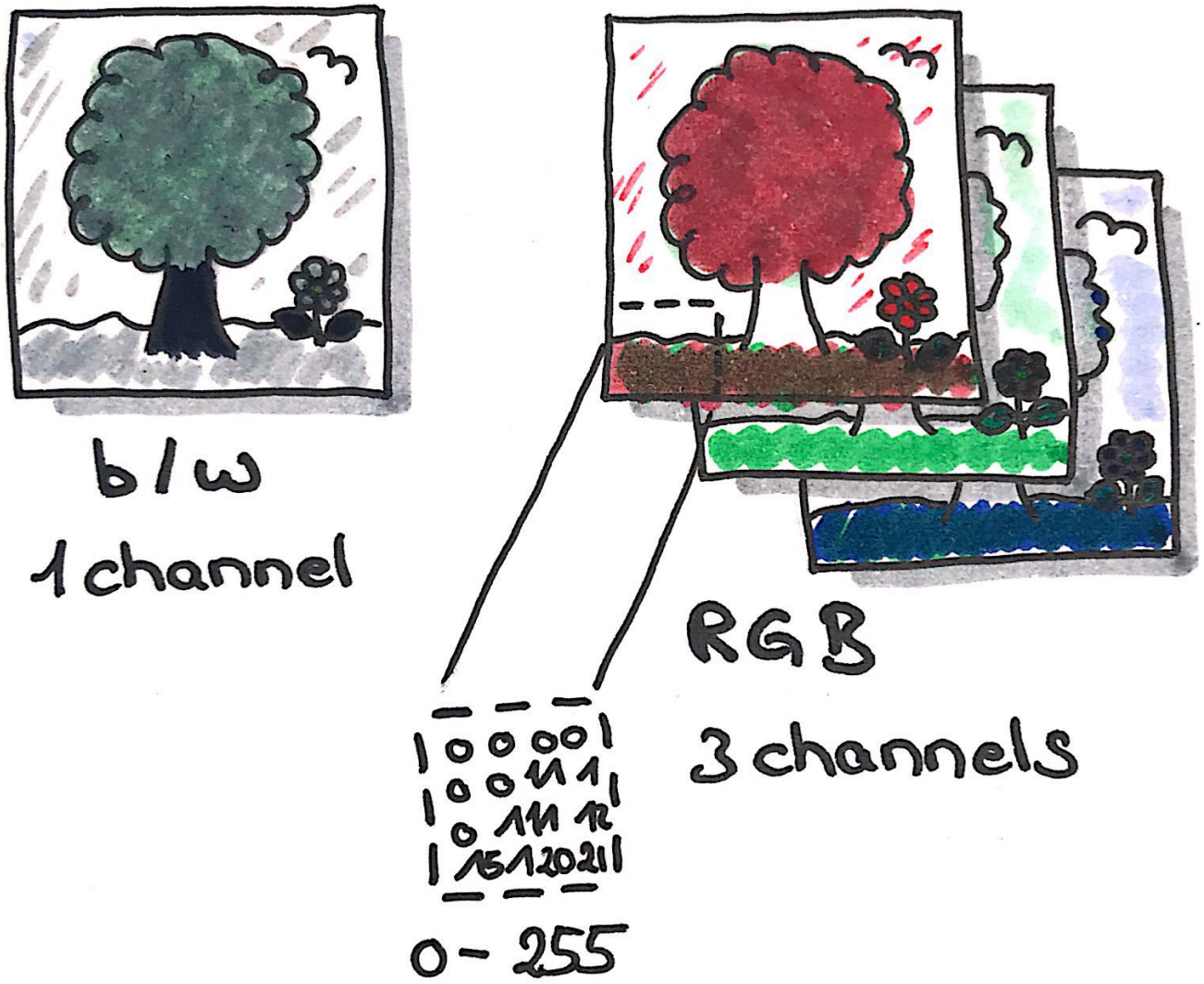
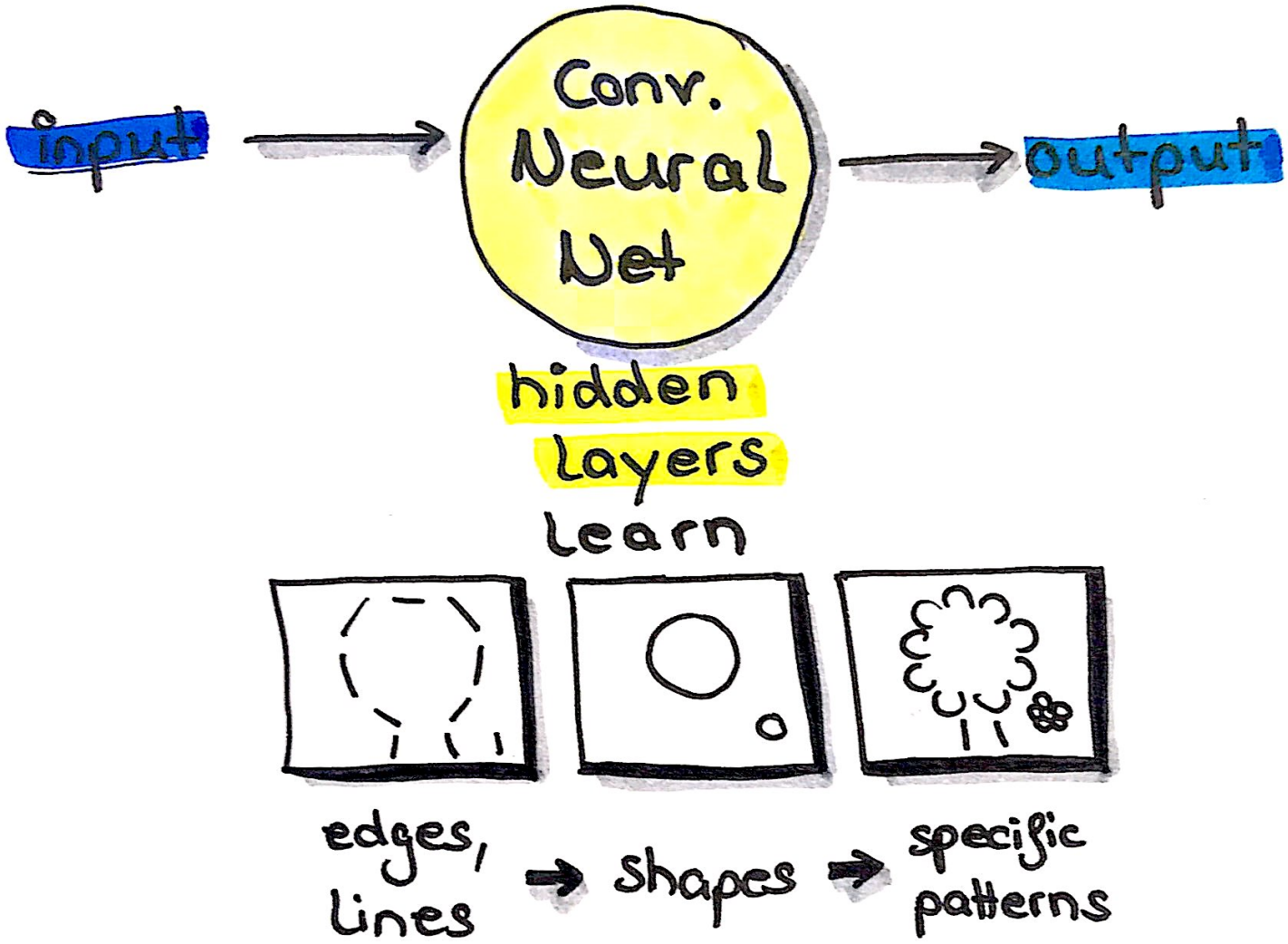


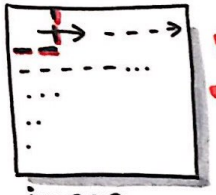
Image input data



Convolutional Neural Networks (CNNs)



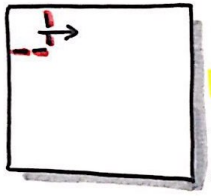
CNN layers



sliding window

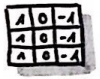
↳ window **size** (e.g. 3×3 pixels)

image

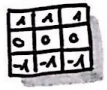


apply **filters** to every sliding window

e.g.



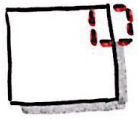
vertical lines



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

stride

- how much overlap to have in sliding window

Convolutional Layers

apply filters

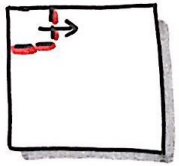
output

=

feature maps

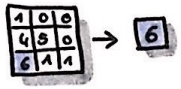
stacks of

feature maps



Pooling Layers

e.g. max pooling



- reduces compute time
- boils information down

Convolution & Pooling

Convolution

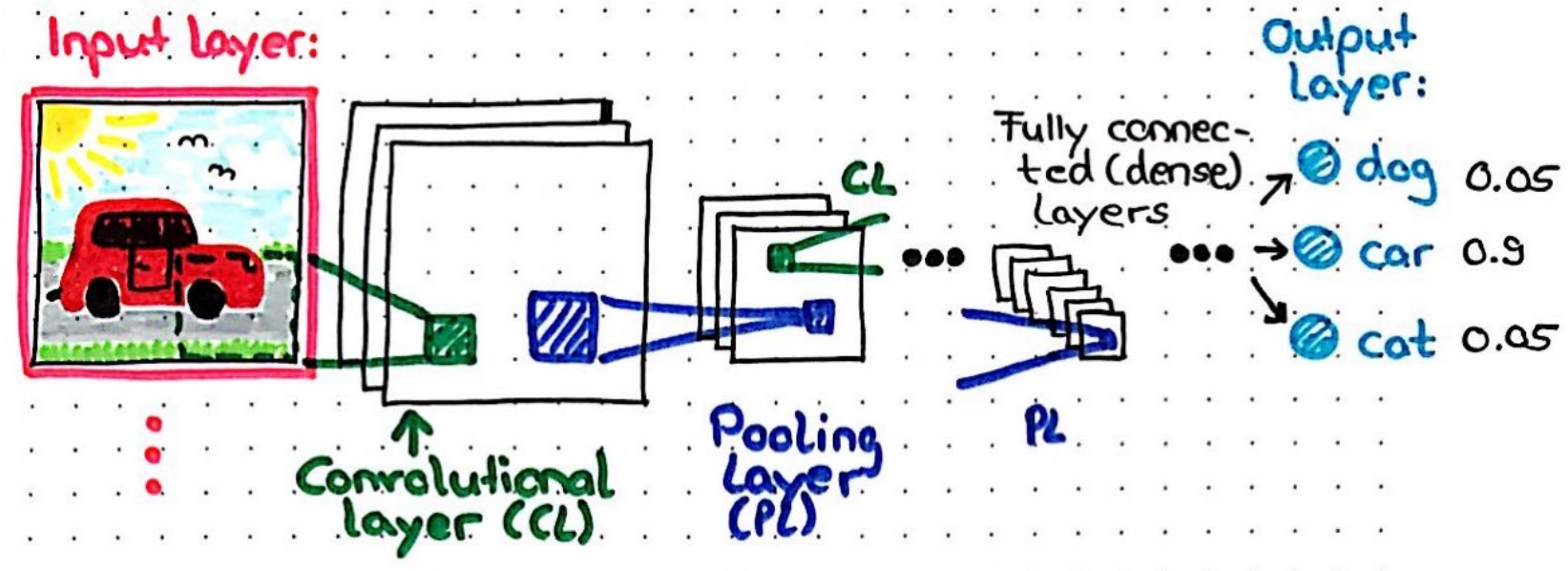
The convolution process calculates an activation map for each region that will be triggered if the learned network recognizes a typical feature used for classification.

Pooling

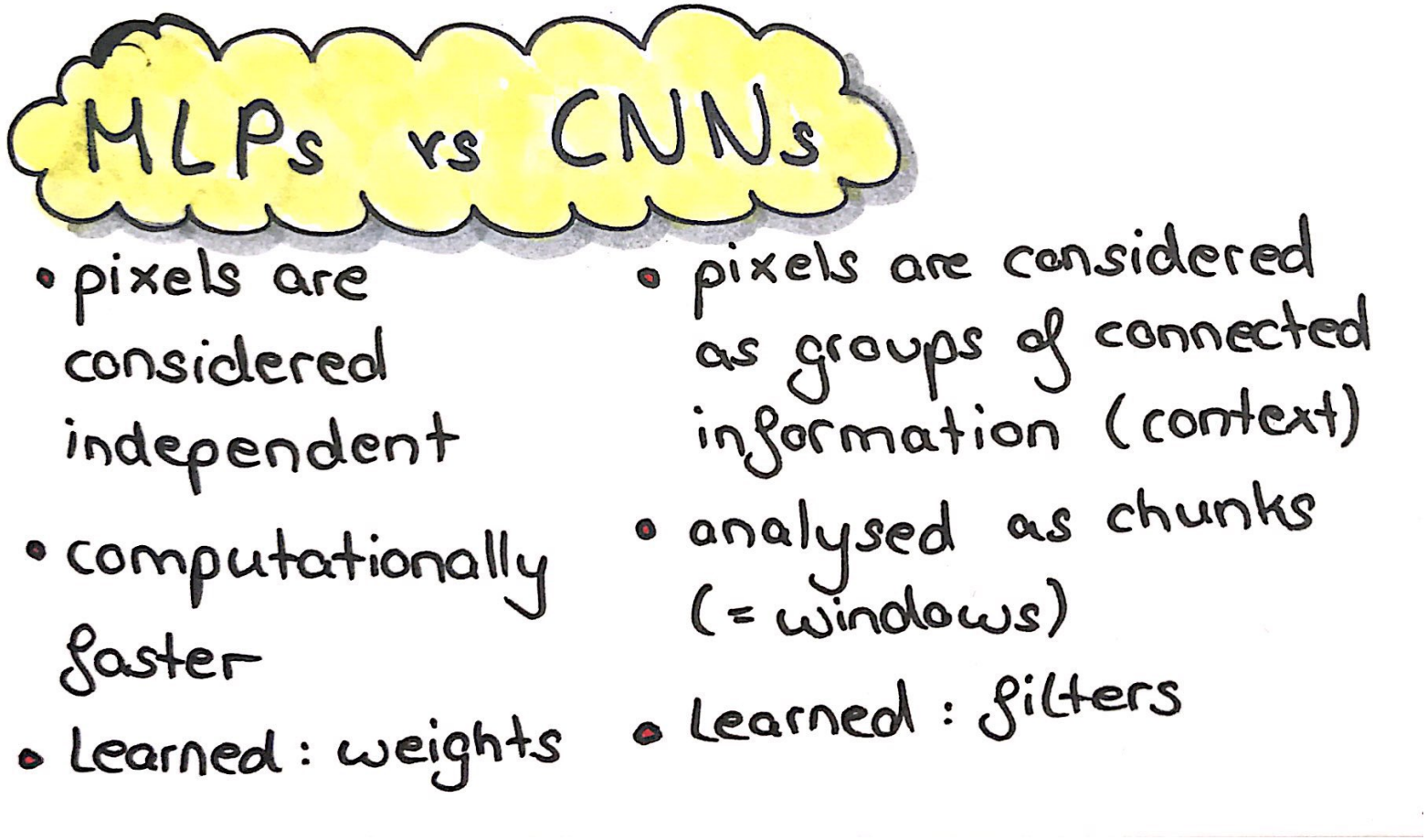
Pooling layers combine the stacked outputs (i.e. the activation maps) of all subsets of neurons from preceding convolutional layers.

CNN architecture

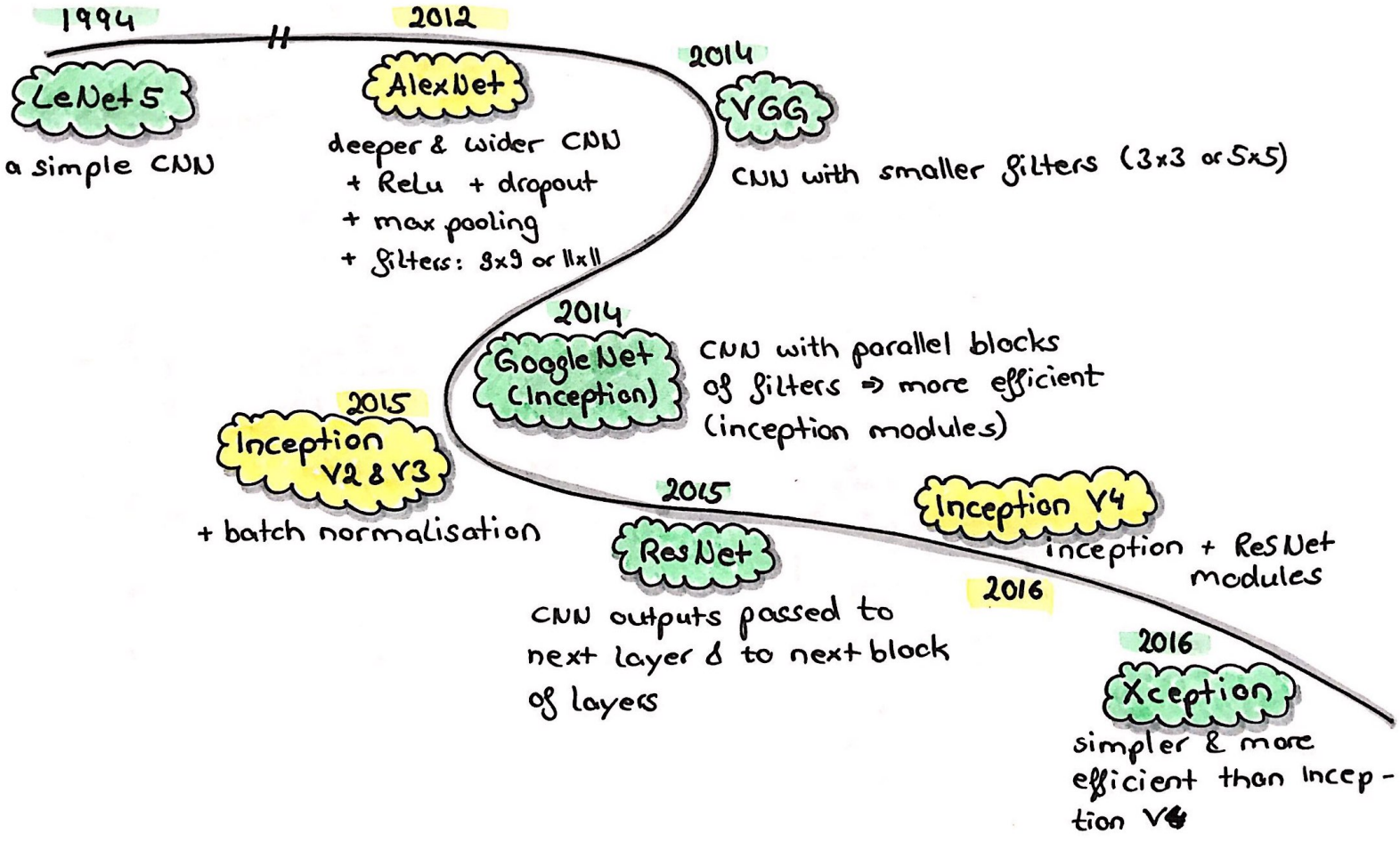
- most commonly used in image recognition
- not fully connected
- convolutional layers + pooling layers



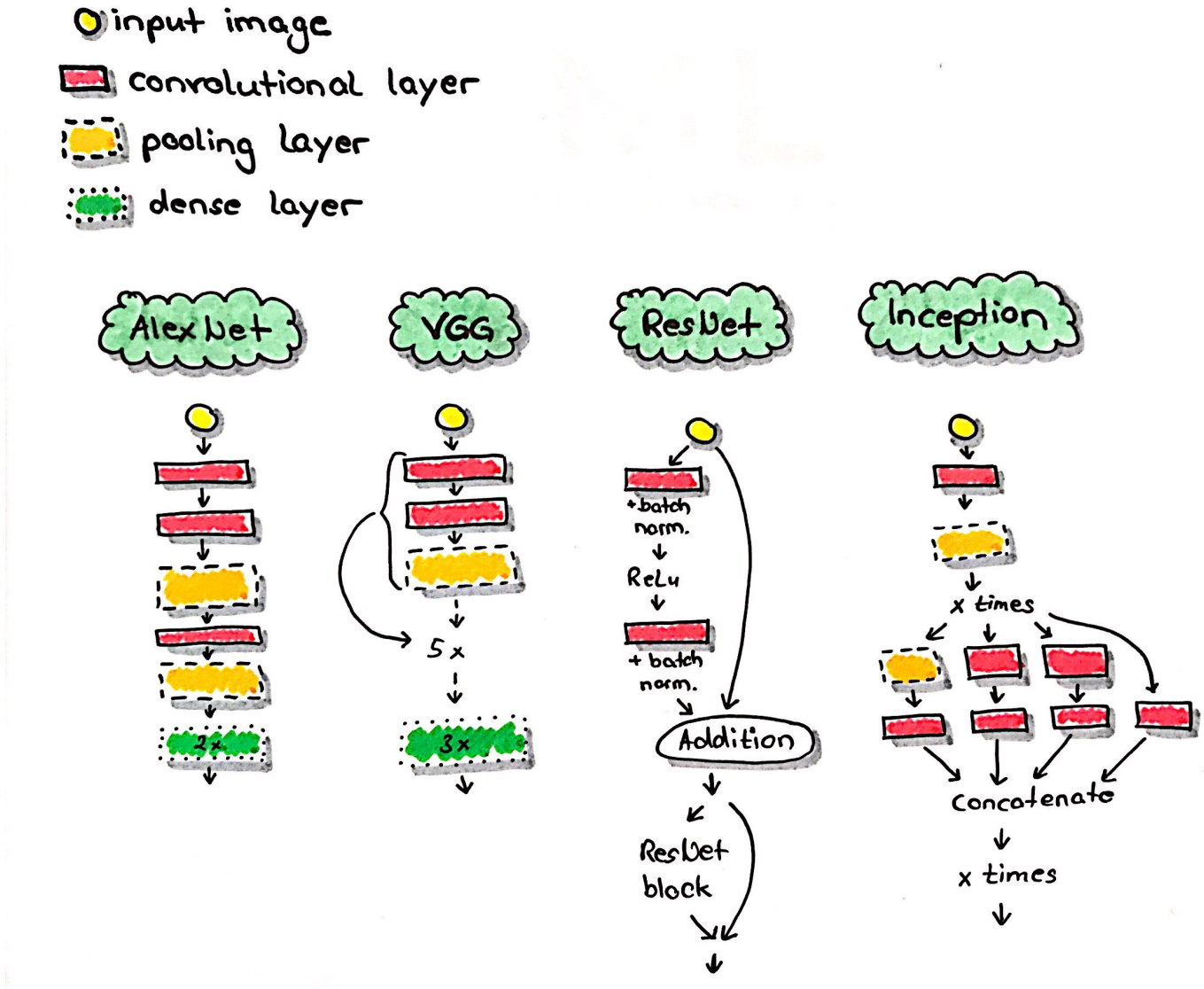
MLP vs CNN


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

History of CNNs



CNN architectures



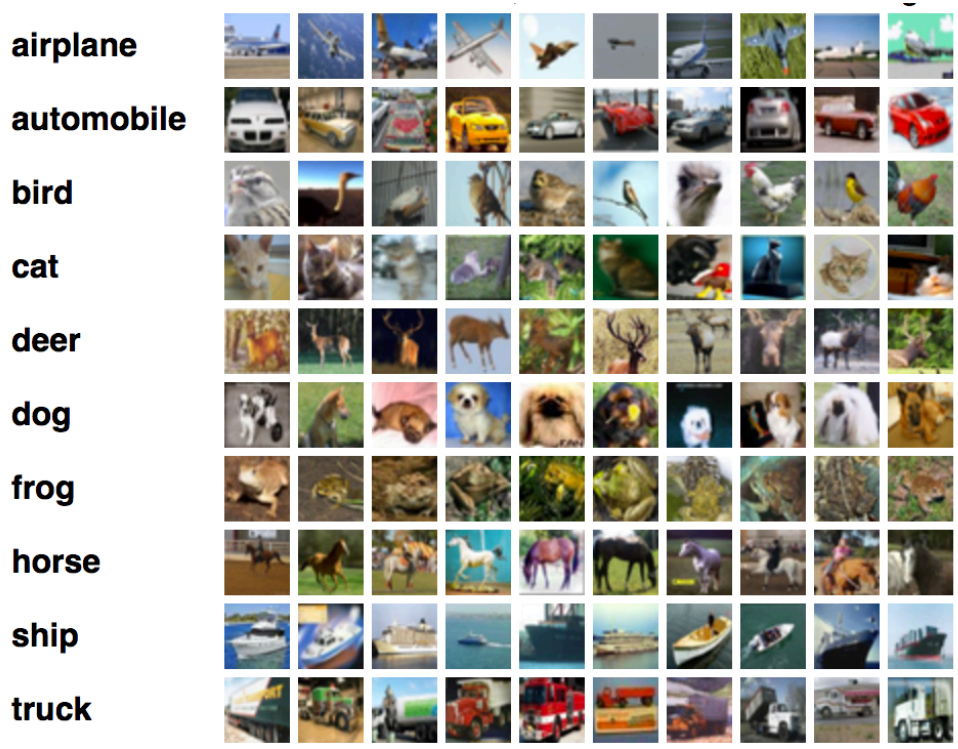
...continue on your own...

Build a CNN with the MNIST dataset

- try out different combinations of layers
- try out different hyperparameters
- ...

The CIFAR example

- image classification dataset
- labels: 10 classes
- images: 32x32 color (RGB)



...continue on your own...

Build a CNN with the CIFAR dataset

- try out different combinations of layers
- try out different hyperparameters
- ...

CNNs can also be used for text classification

How does a computer learn to understand natural language?

- document term matrix / term document matrix
- word embedding matrices

Document term matrix

- generate list of words in the complete corpus
- count words per document
- optional: normalize (e.g. tf/idf)
- optional: remove stop words, filter common words, etc.

Embedding matrices

- generated from vector representation of words (word2vec or index)
- high-dimensional matrix representation of words
- learned with neural nets, PCA/t-SNE/UMAP or co-occurrence matrices

...continue on your own...

Build a CNN with the IMDB dataset

- try out different combinations of layers
- try out different hyperparameters
- ...

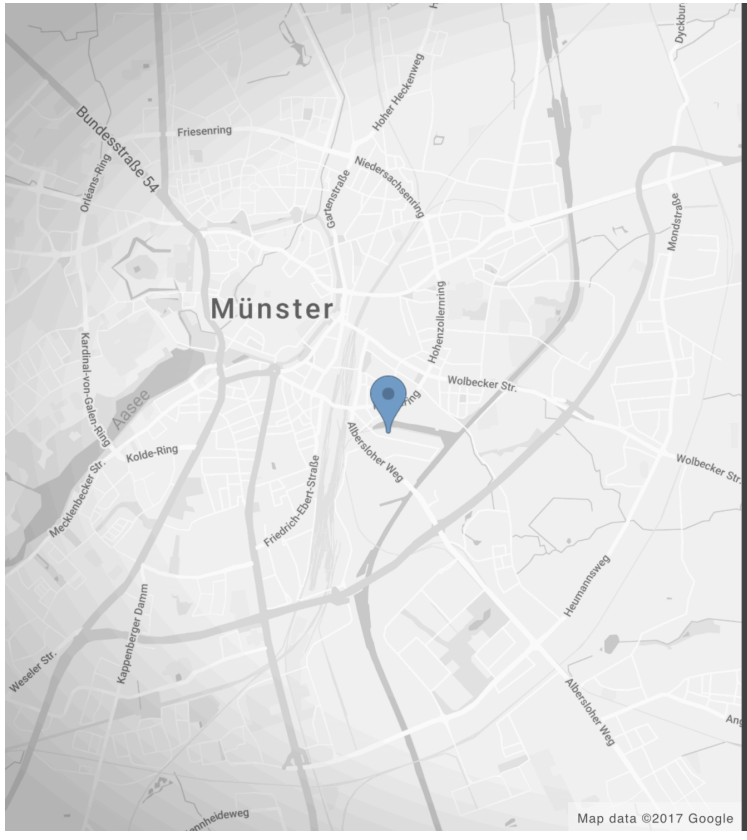
Additional resources

Follow links:

- Classify a fruit dataset from Kaggle...
- ... and explain it with LIME
- Image clustering with Keras and k-Means
- Use pre-trained models, either directly or modify them
- Free notebooks from “Deep Learning with R”
- Examples in Python for image & text classification

Thank you!

Thank you



Thank you!
And stay connected

Am Mittelhafen 14, Münster

ShirinGlander

shirin.glander@codecentric.de

www.codecentric.ai
https://www.youtube.com/codecentricAI
www.shirin-glander.de