Introduction to AI
Neural Nets basics

BROAD INSTITUTE
HARVARD MEDICAL SCHOOL
MASSACHUSETTS GENERAL HOSPITAL

Tools we'll see

PyTorch

NVIDIA

Seaborn

scikit learn

jupyter

CO

https://colab.research.google.com/drive/1vQsY-BLEN5TEITqTC_0ePo0TaZ3sQbM_

Types of applications: Supervised (classification)

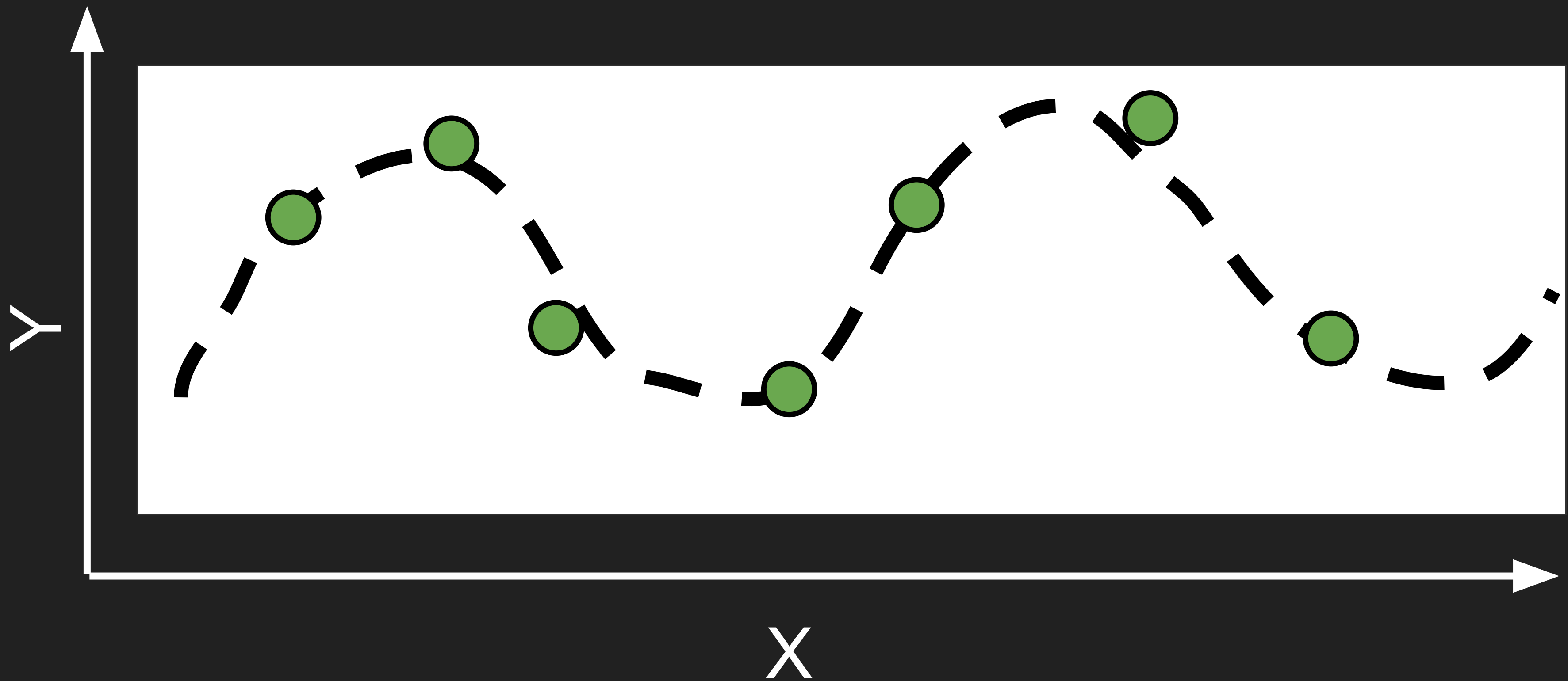
Data

Label / Target

Dog

Cat

Types of applications: Supervised (regression)

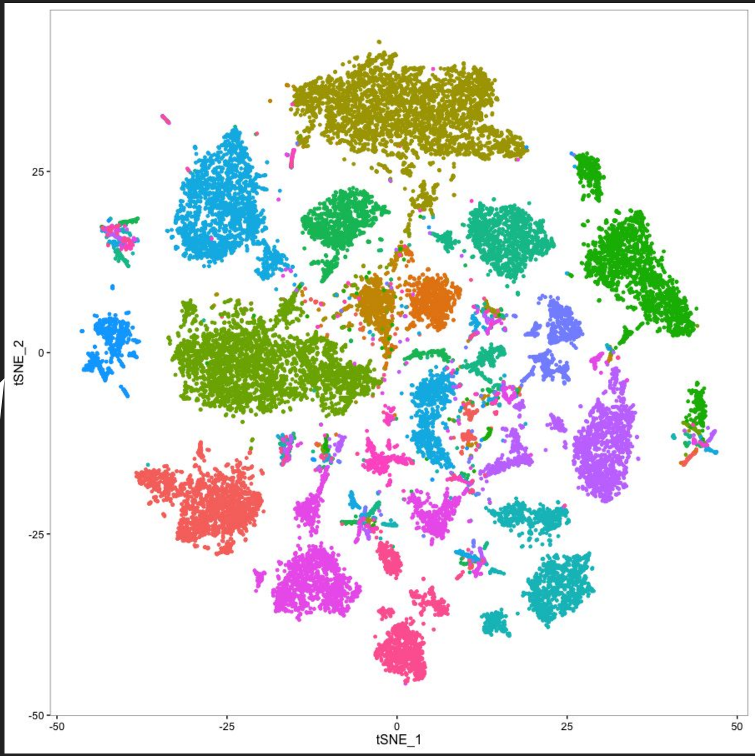


Model a fonction from examples

Types of applications: Unsupervised

- No labels / Targets
- Discover structures in the dataset
  - Clustering
  - Visualization

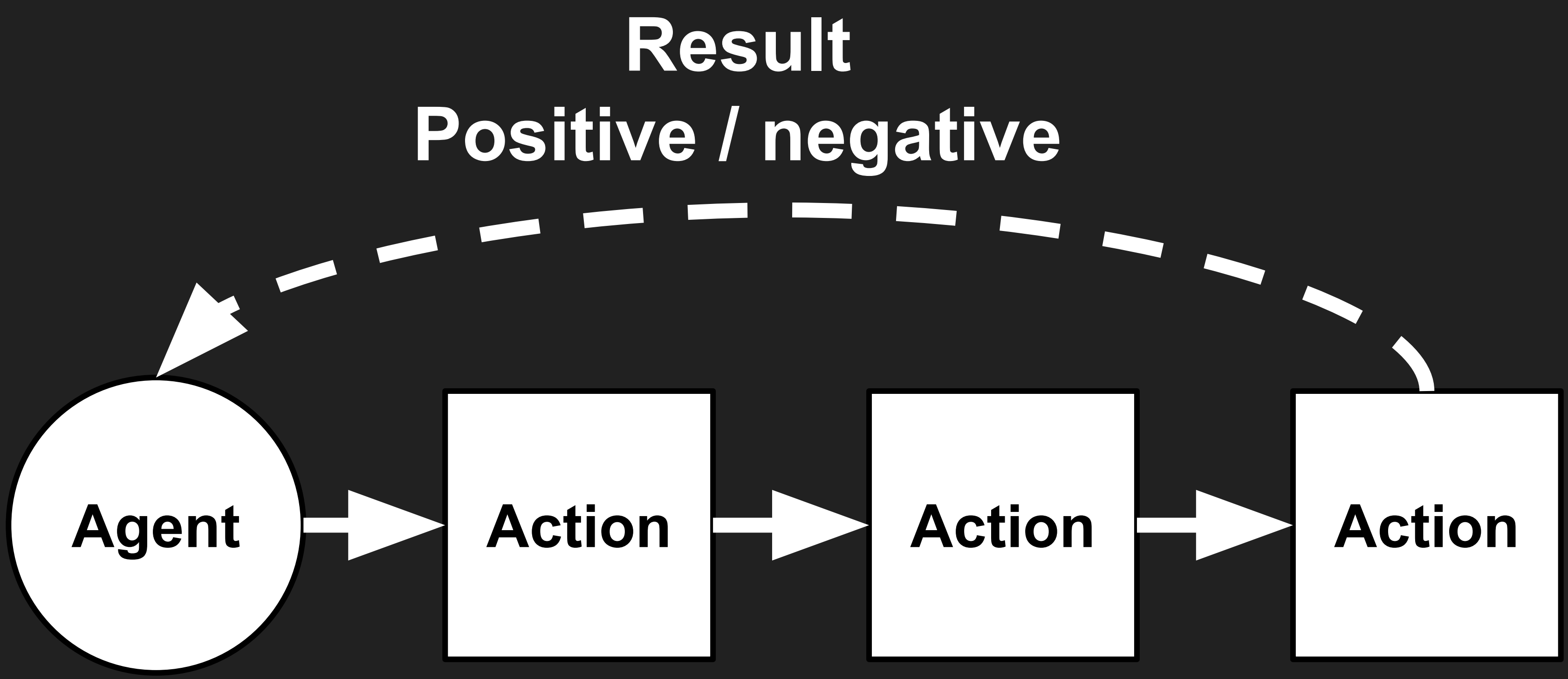
Gene expression
1 Gene: 20k values



Satija lab

1 Gene: 2D

Types of applications: Reinforcement learning

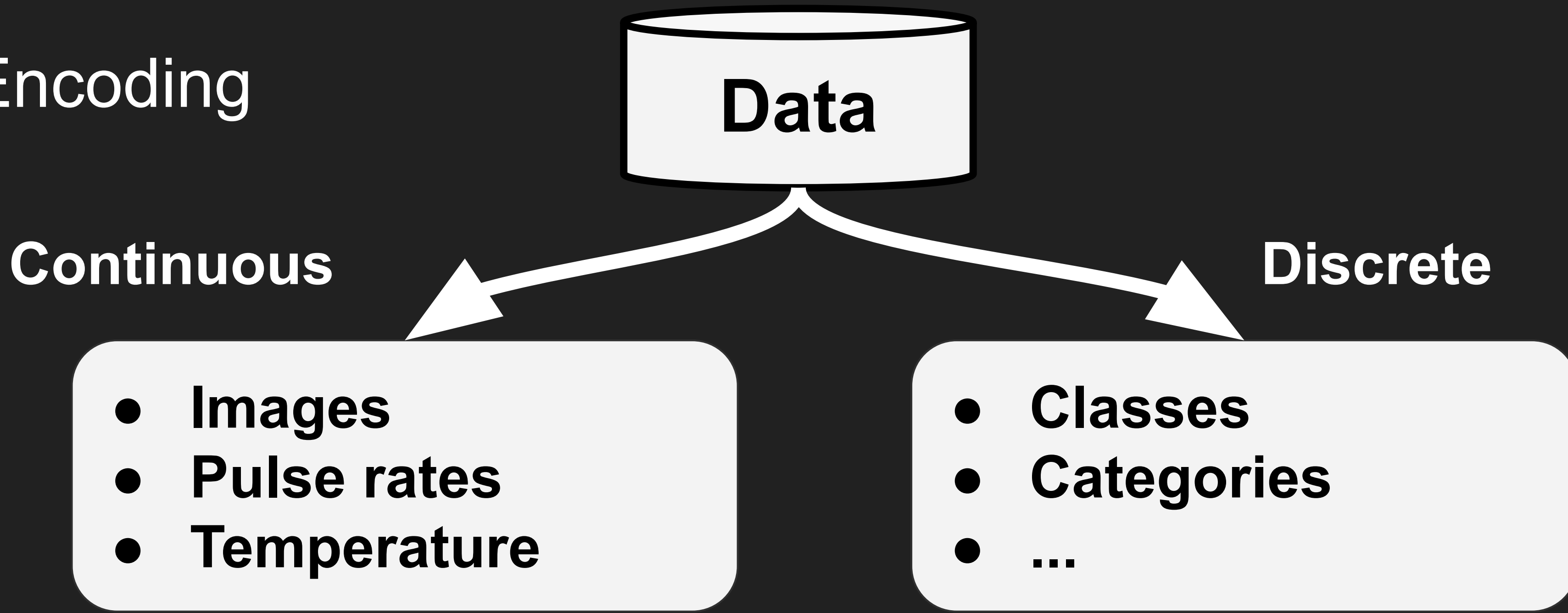


Types applications: Generative models

- Learn to generate examples similar to training examples

Data handling

Encoding



- Reduce range between values
  - log(values)
- Between [0, 1] (
  - - min
  - / max

1 Item => 1 unique id

0	Horse
1	Cow
2	Dog
...	...

Standard format: numpy arrays / pandas dataframes

Continues

- Images
- Pulse rates
- Temperature

float32

Discrete

- Classes
- Categories
- ...

int16

```
data = numpy.asarray(data, dtype='int16')
```

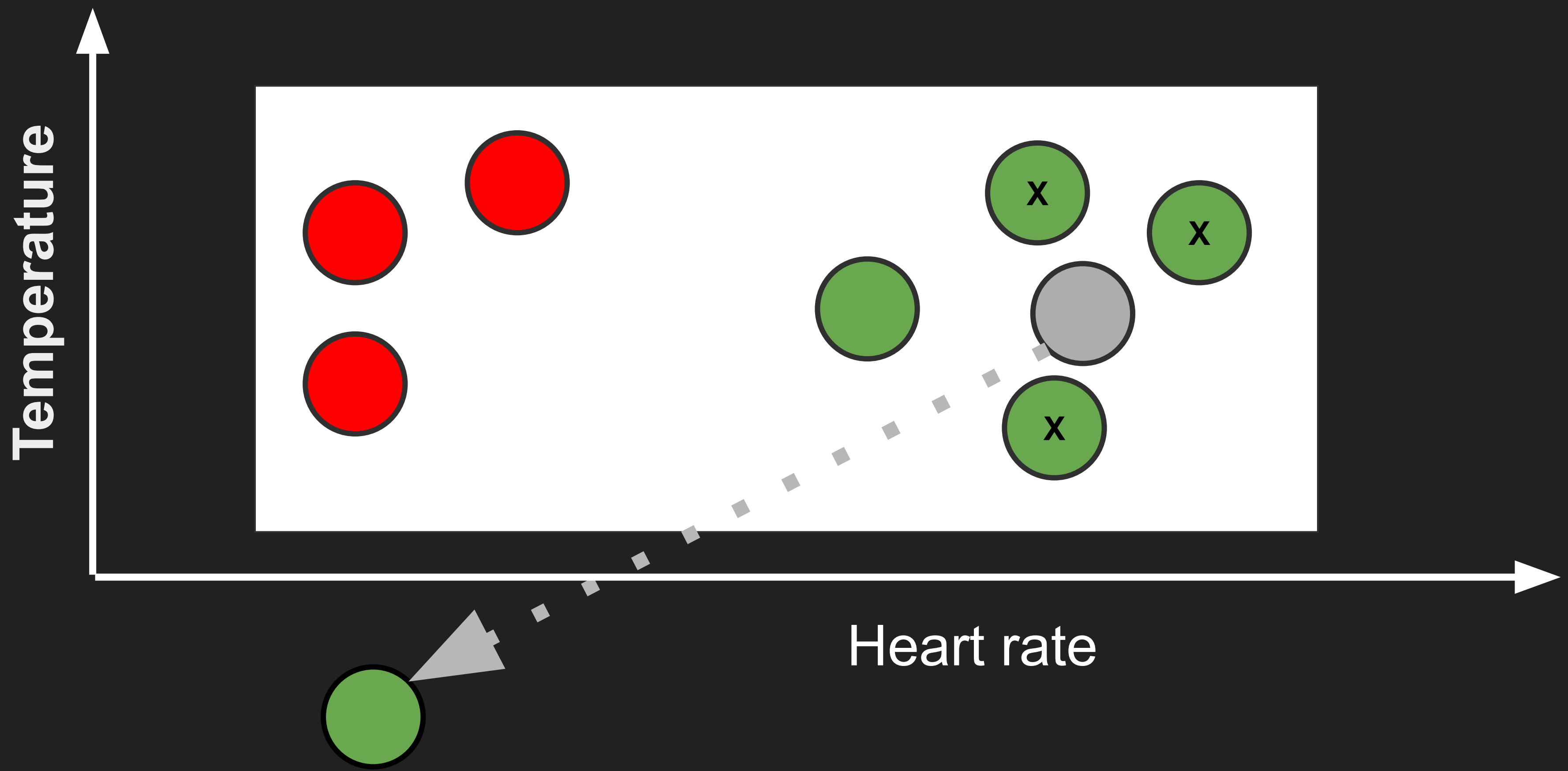
```
data = numpy.asarray(data, dtype='float32')
```

Supervised Learning

Dataset

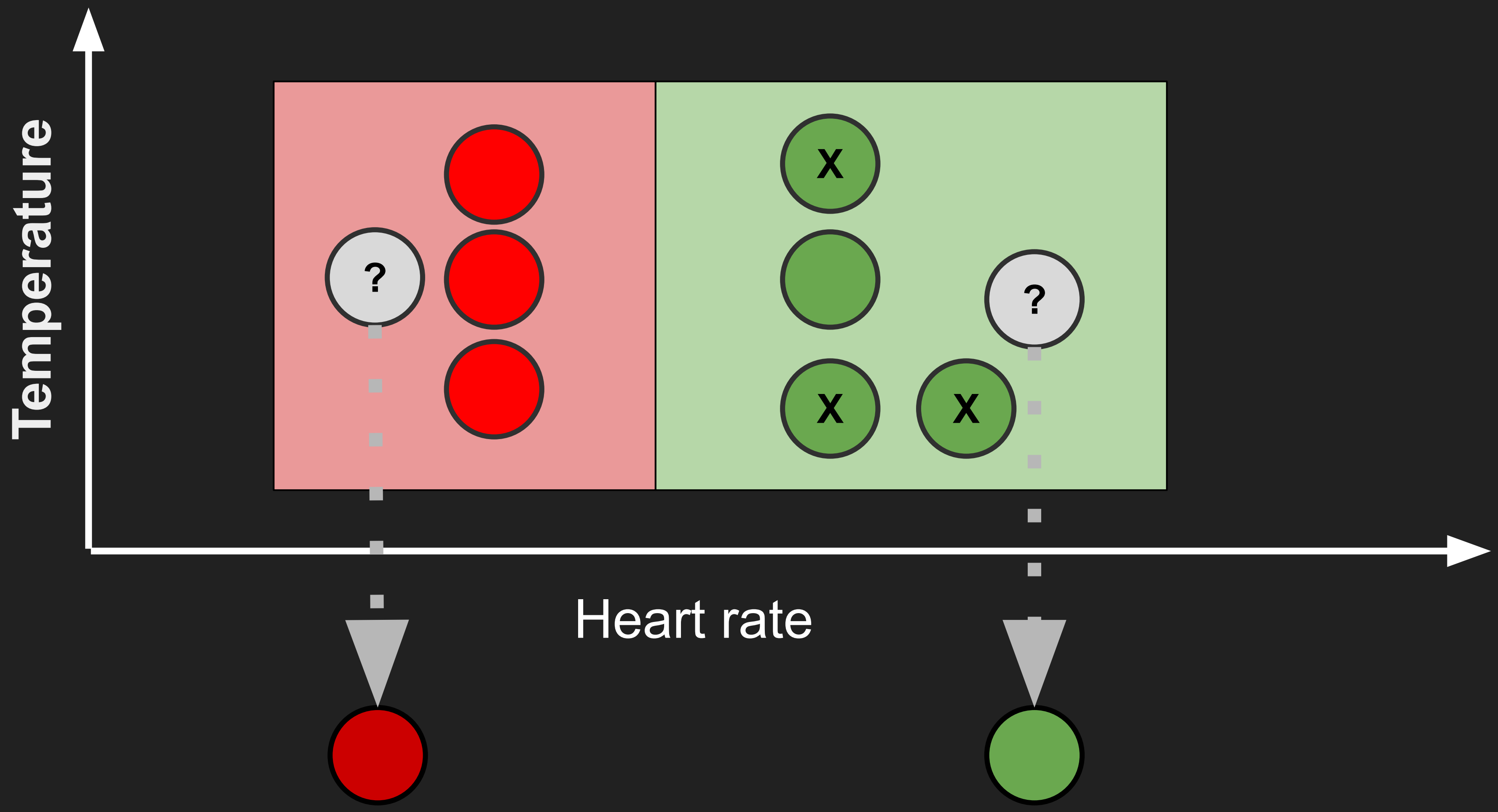
Temperature	Heart rate	Type
0.1	0.32	
0.9	0.5	
0.7	0.59	

Ex classification: K nearest neighbours (KNN)



Assign class of K (ex: 3) nearest neighbours

Classification: As space partitioning



Artificial neural networks

Dataset: predict **Temperature** from **Heart rate**

Heart rate (x)	Temperature (z)
0.02	0.2
0.04	0.4
0.08	0.8

Model

↓

$z = w * x$

↓

Find the best value for w

↓

$z = 10 * x$

Dataset

Concentration (x)	Effet (z)
0.02	3.2
0.04	3.4
0.08	3.8

Modèle

↓

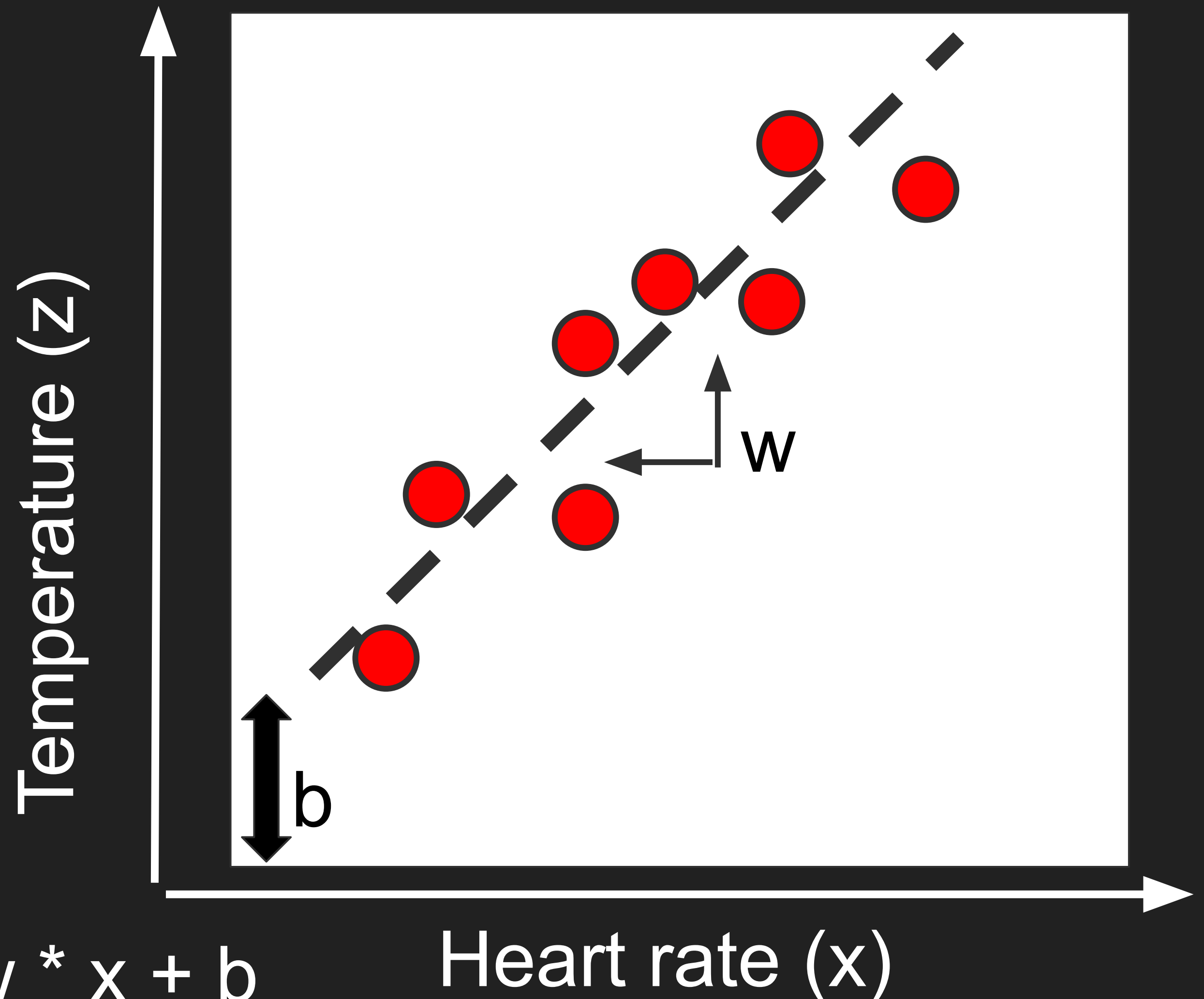
$z = w * x + b$

↓

$z = 10 * x + 3$

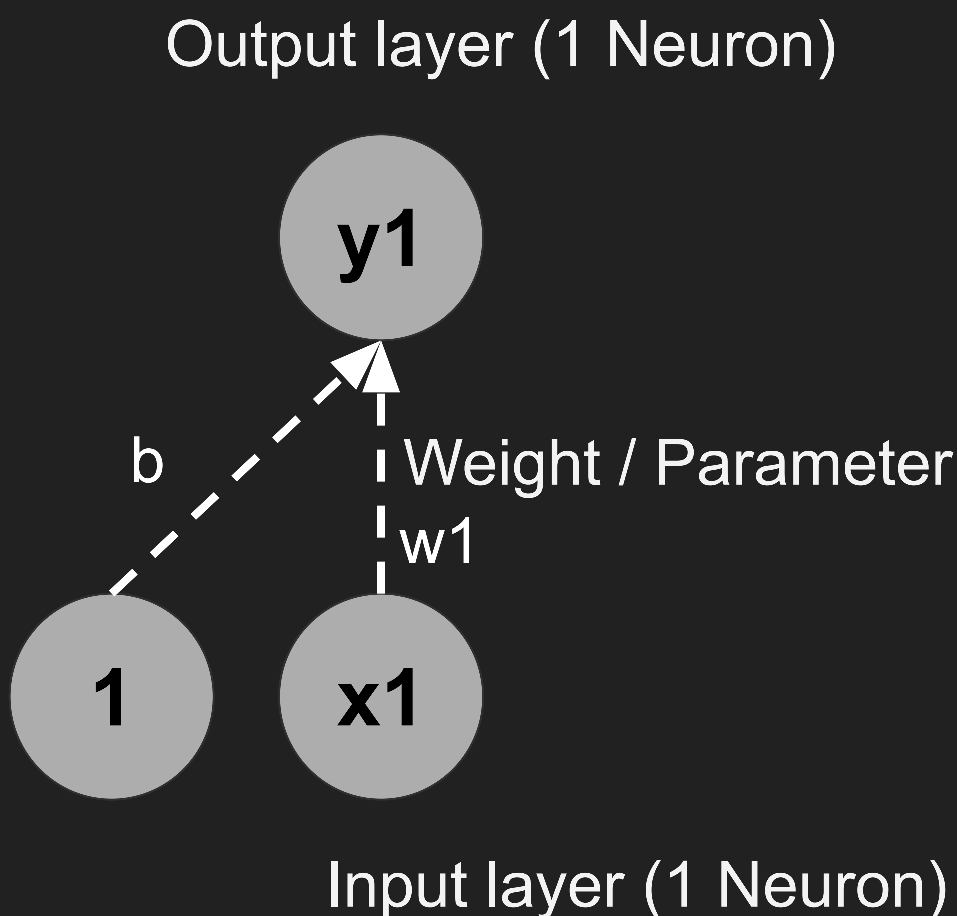
Biais (b)

Linear regression

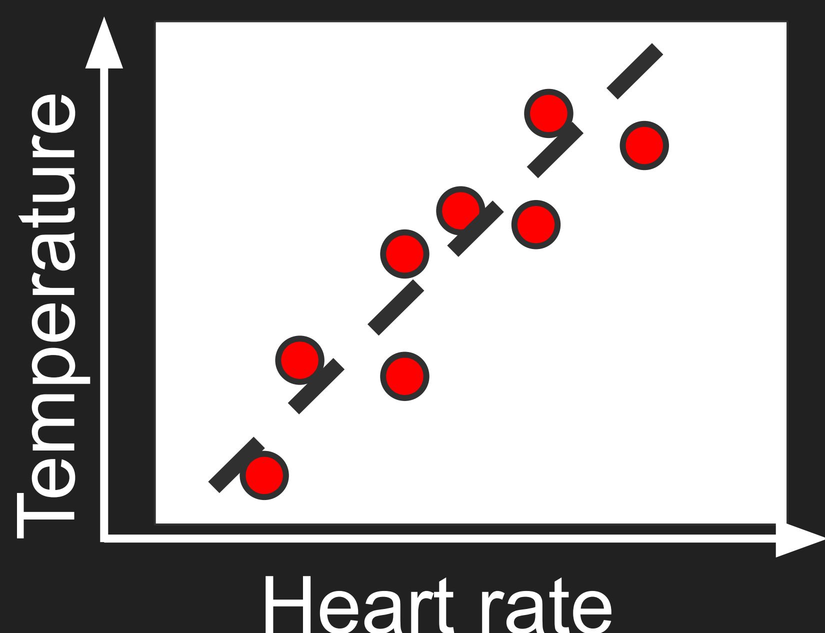


$z = w * x + b$

Baby neural network



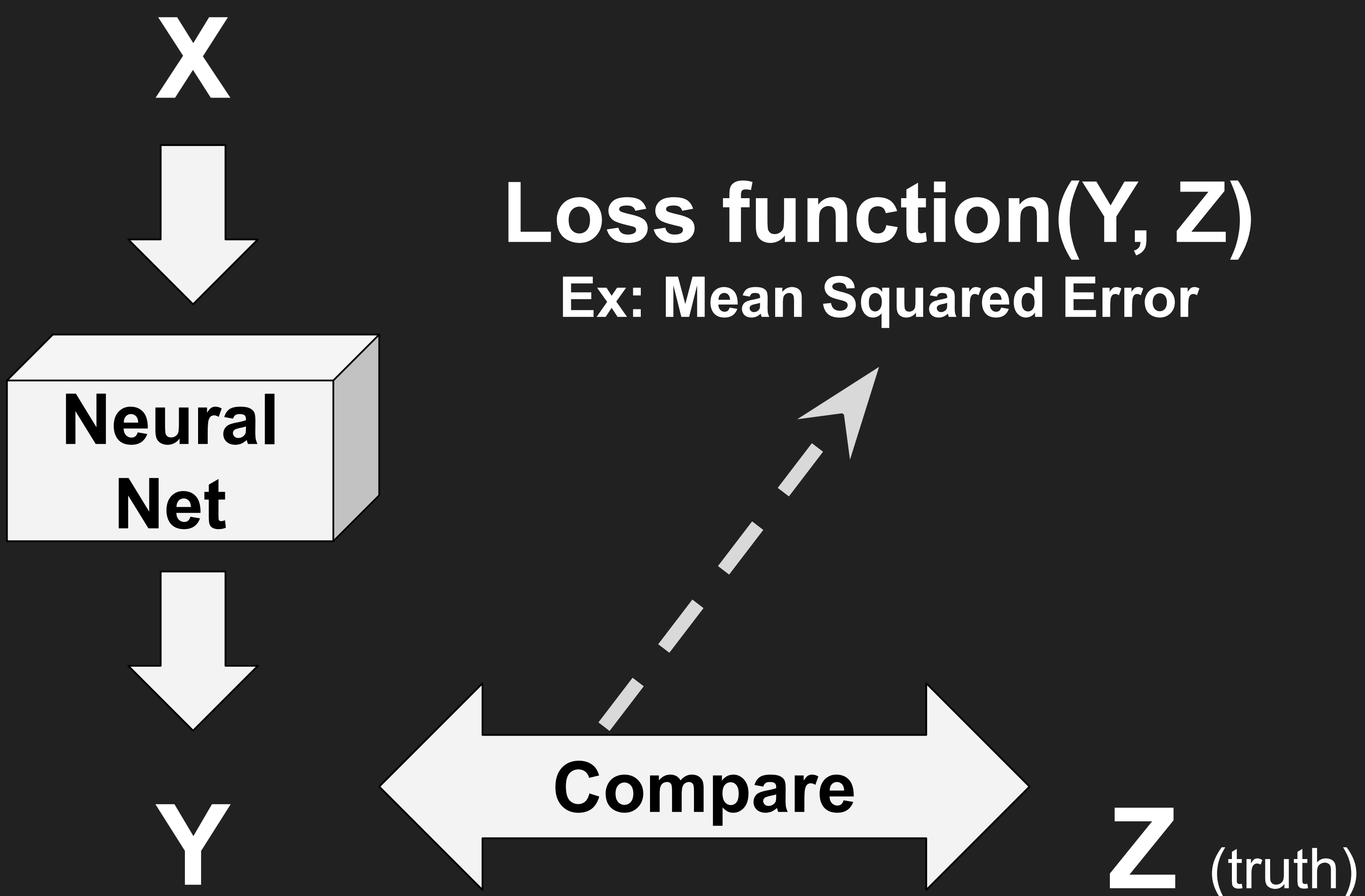
$$F(x) = w1 * x + b$$



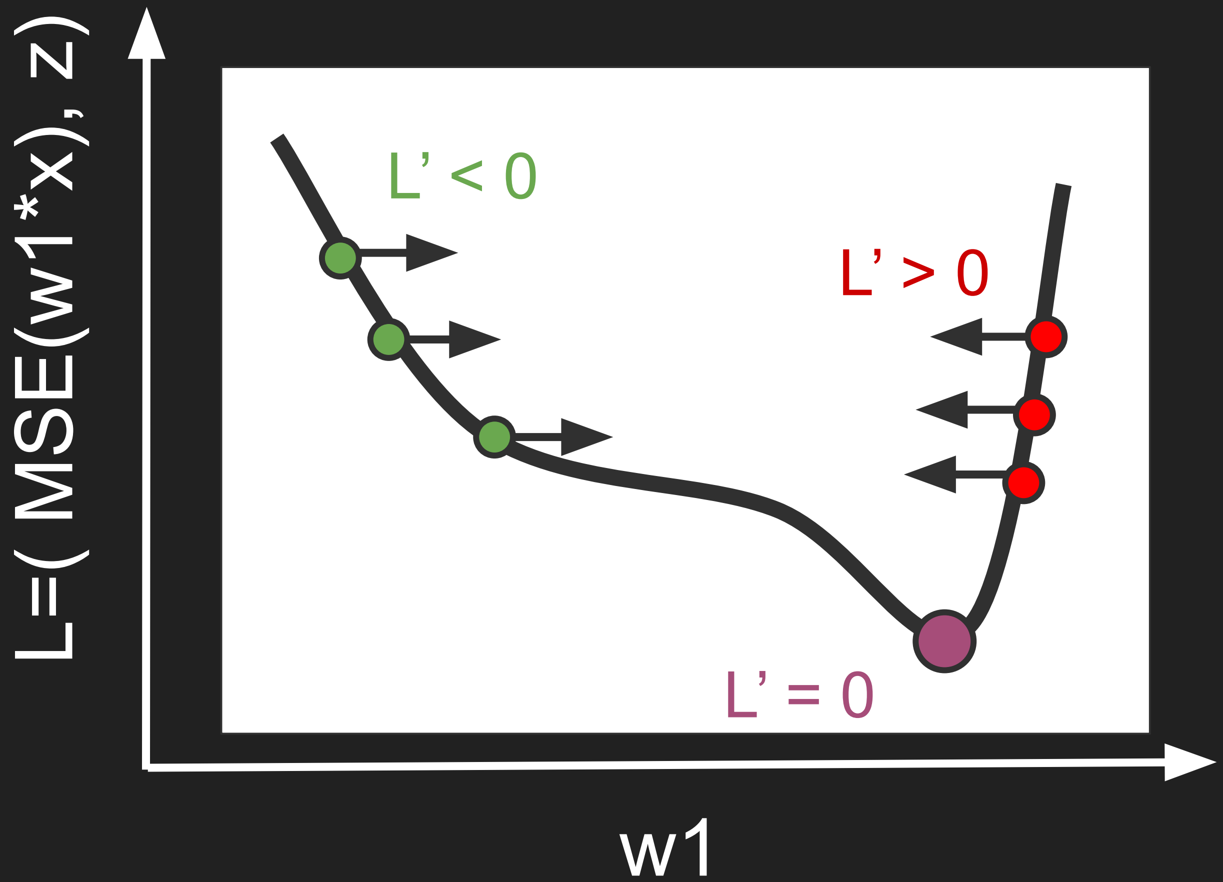
ML = Find w1 automatically

How can we find w1 automatically?

How far are form the truth?



Find a value of w1 that reduces L, how?

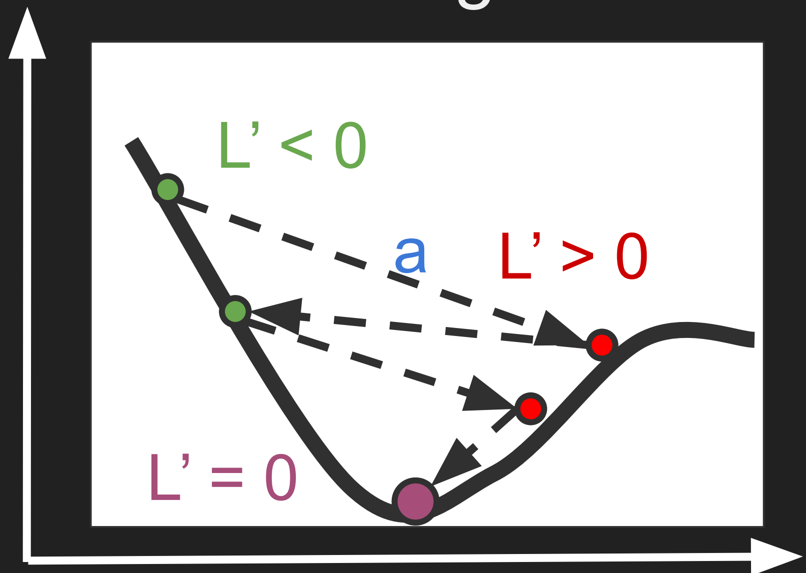


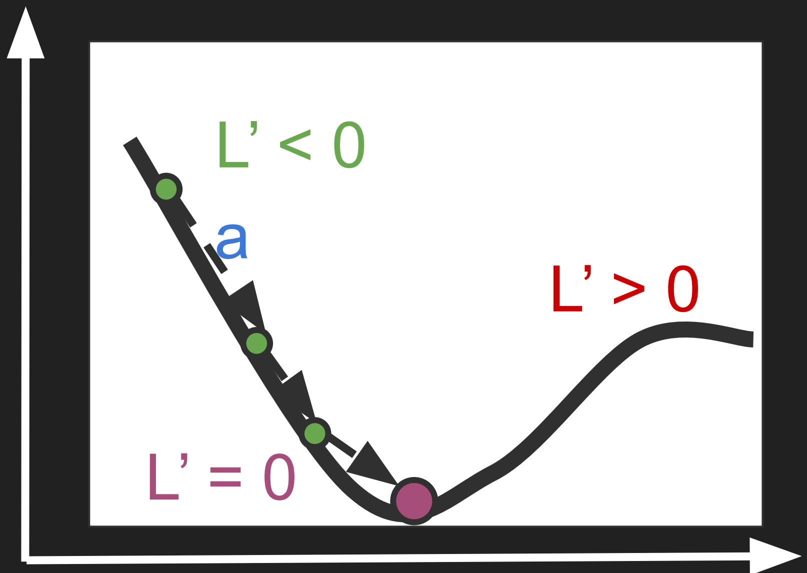
The variable is **W1** Not X

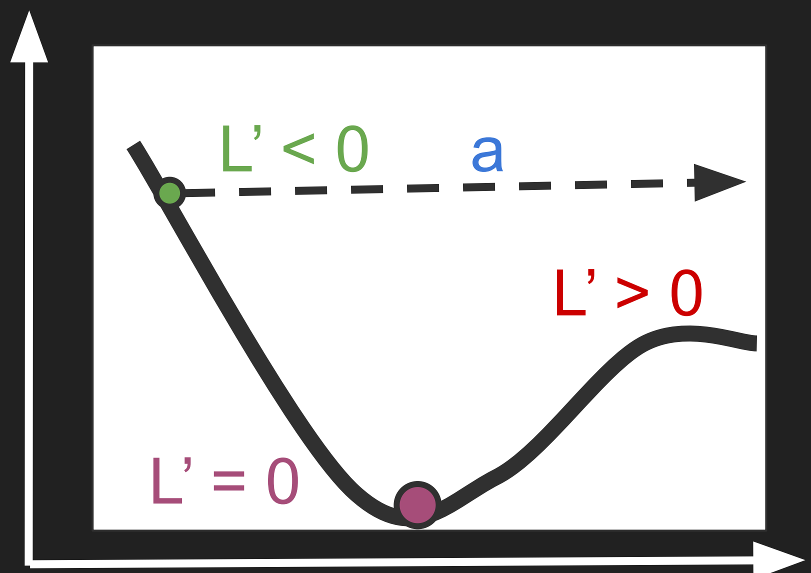
Gradient Descent

W1 = W1 - L’

Good convergence Vs Catastrophic bounces







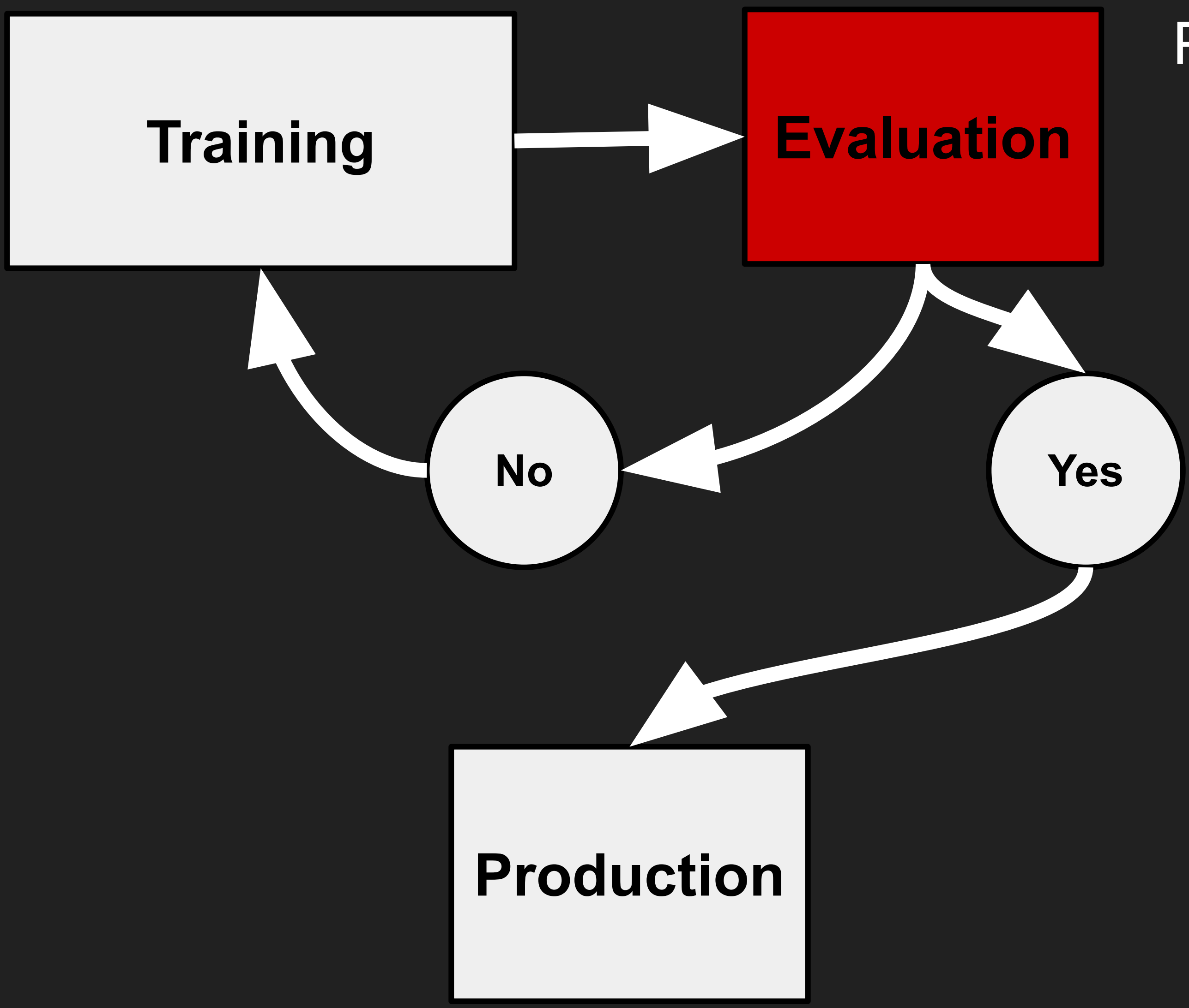
W1 = W1 - <u>a</u>*L’

a: Learning rate

Tensors, is just a name for (almost) everything

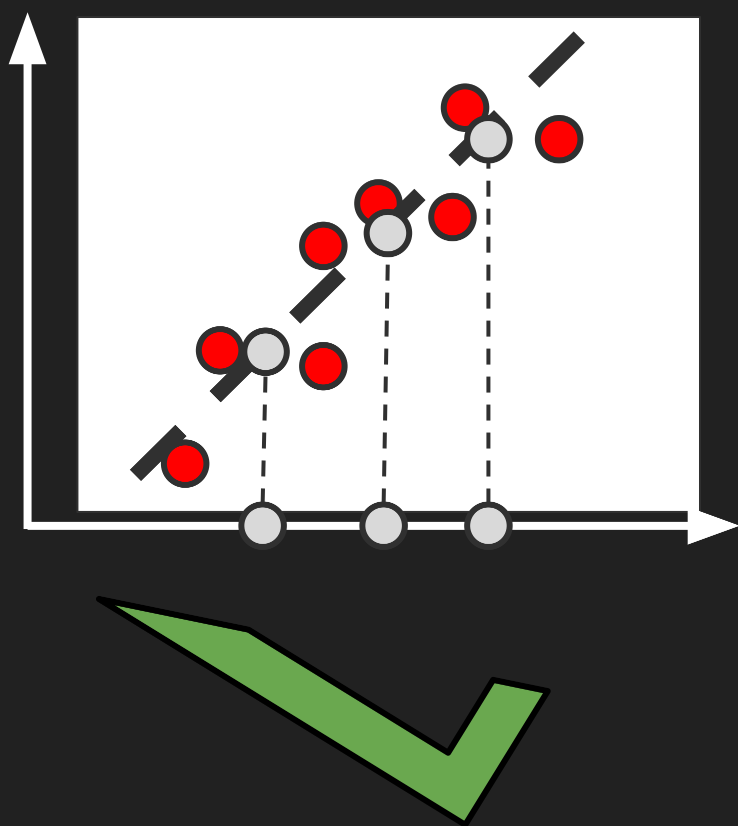
- Number: 0D tensor
- Vector: 1D tensor
- Matrix: 2D tensor
- ….

Workflow

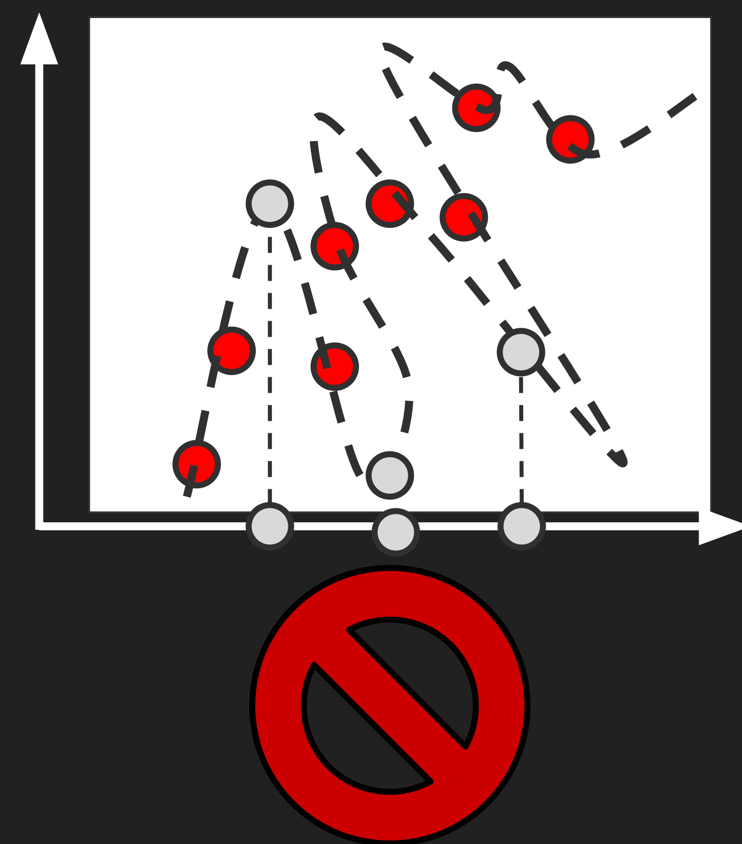


Performances on "*real life*" data

Overdoing it (overfitting): Regression

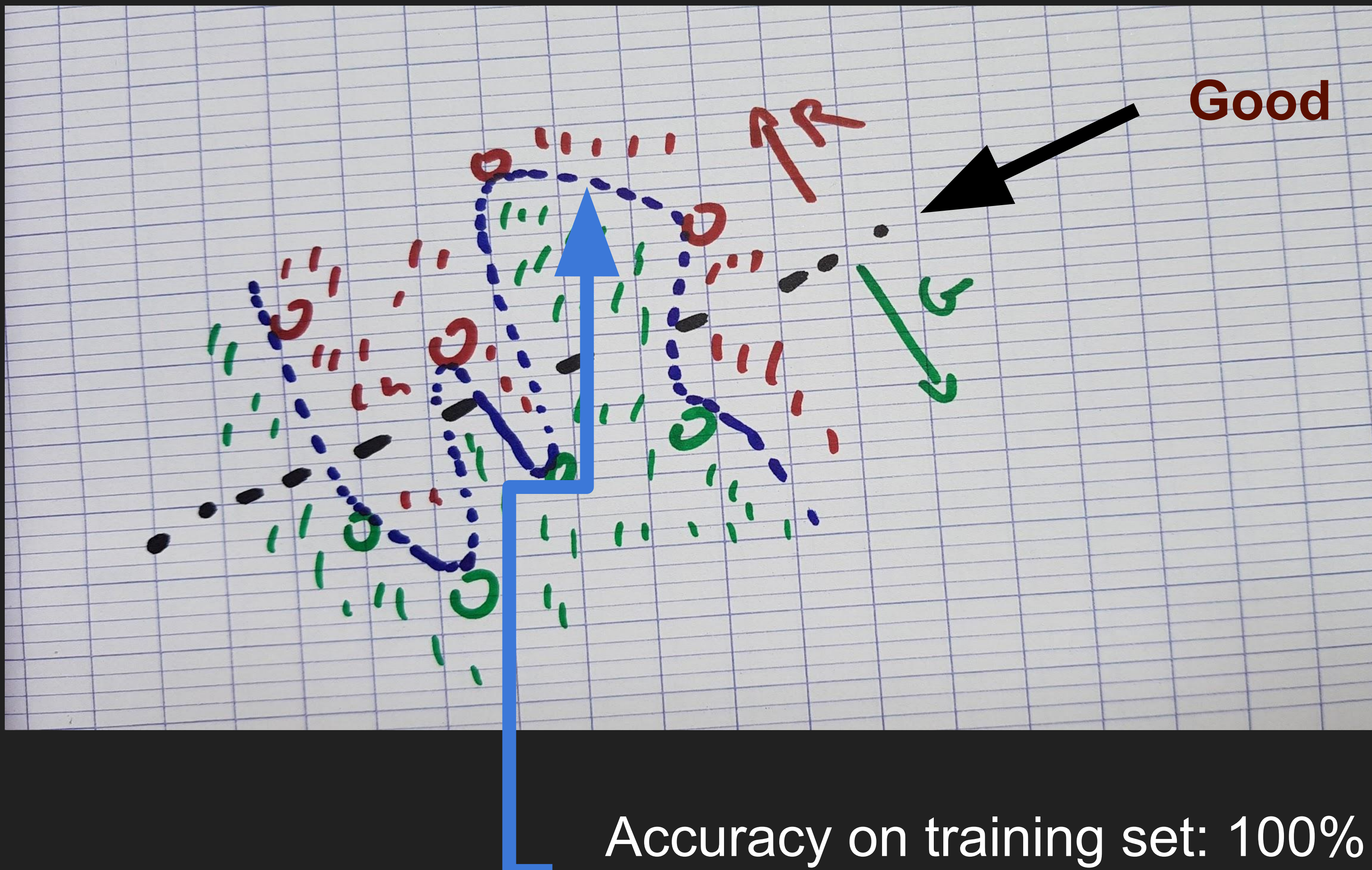


Good



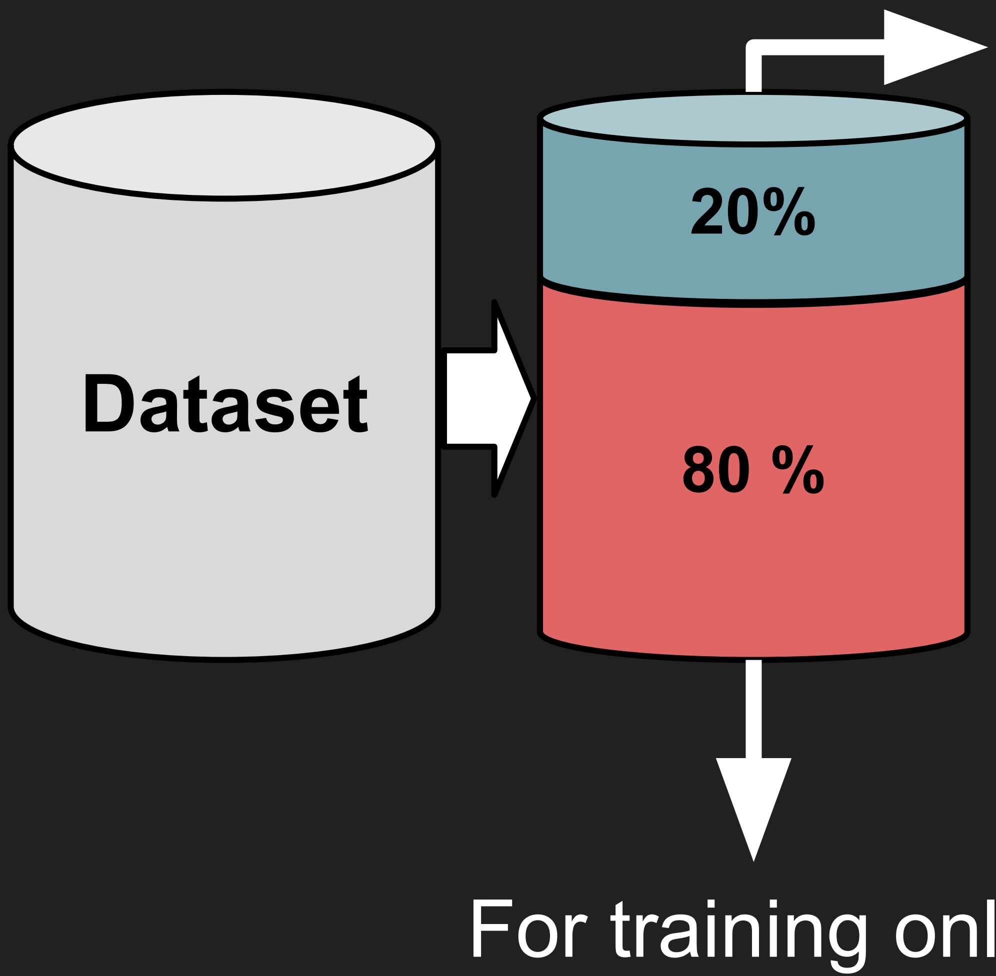
Accuracy on training set: 100%
Real life: Very bad

Overdoing it (overfitting): Classification



Accuracy on training set: 100%
Real life: **Very bad**

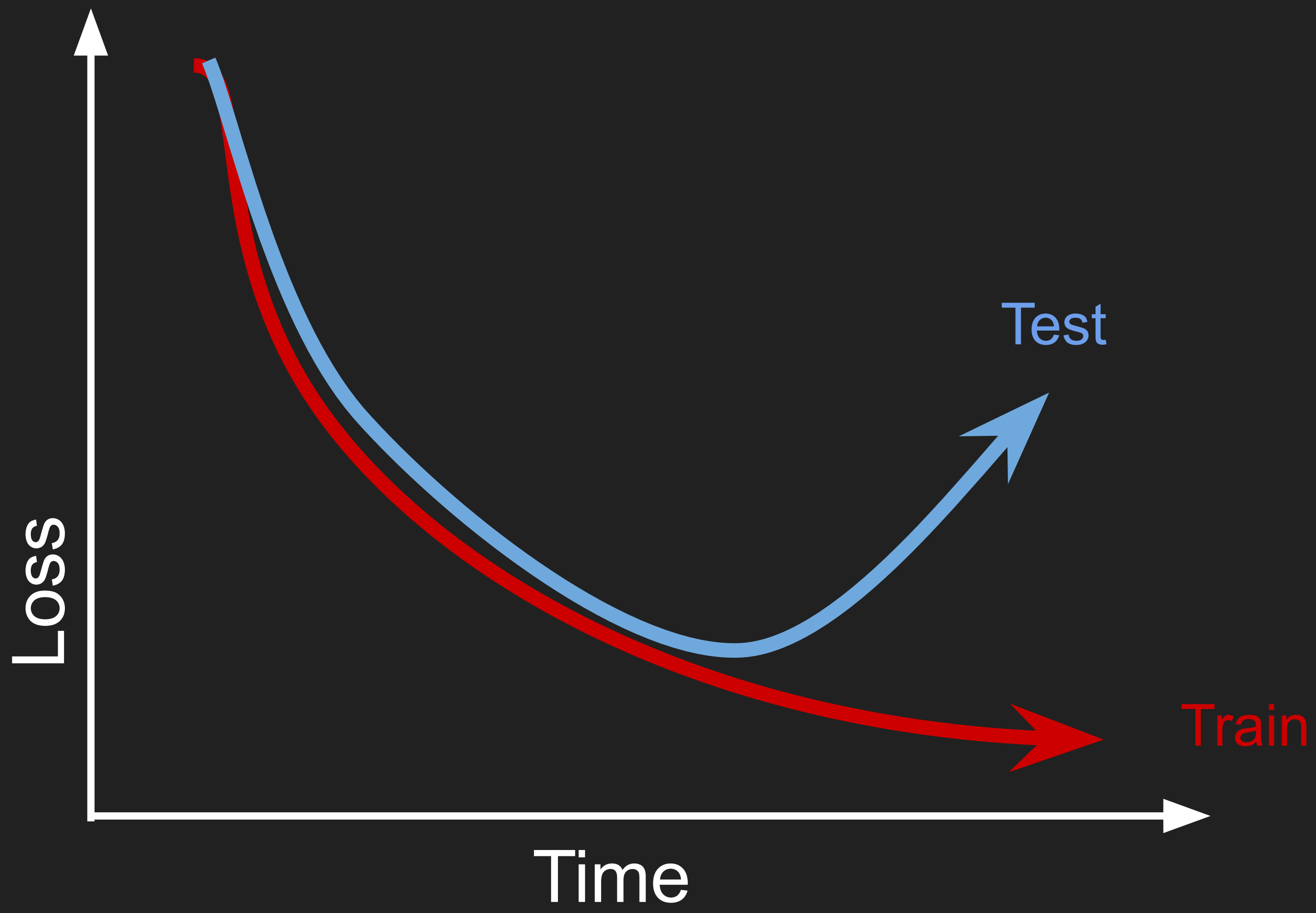
Evaluation: Train / Test split



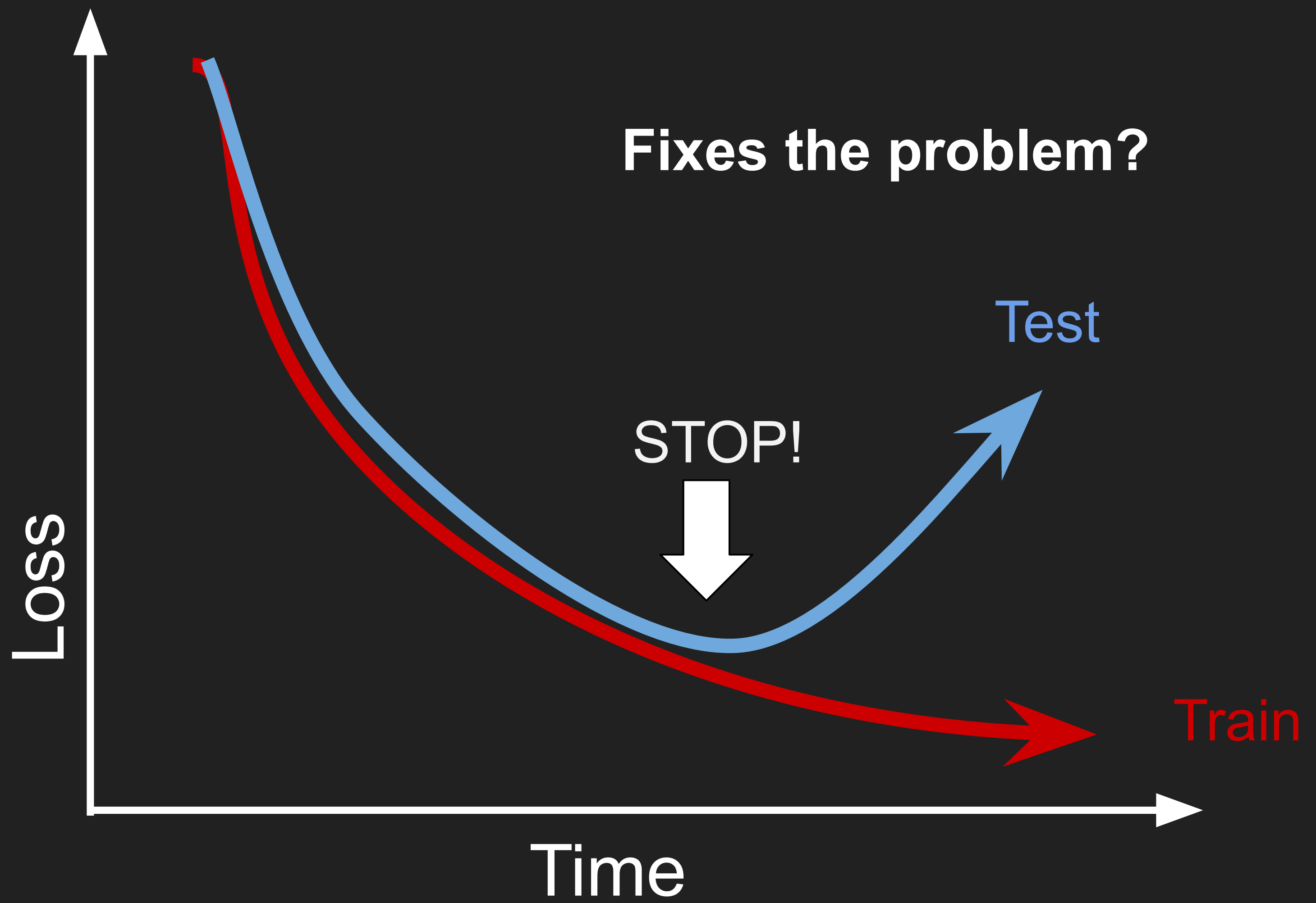
Test

- **Never** used for training!
- Mesure model performances on real life data

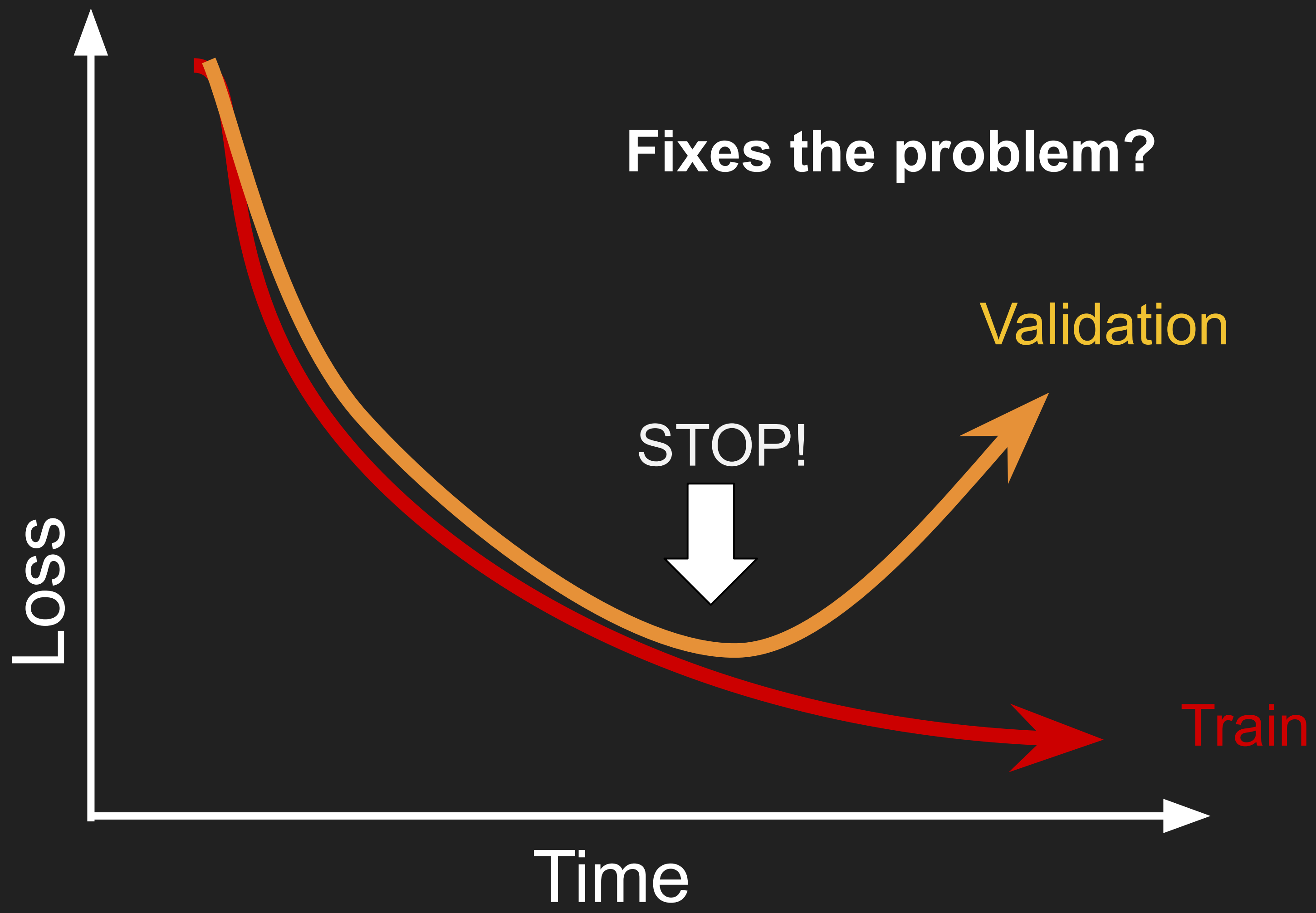
Spotting overfitting



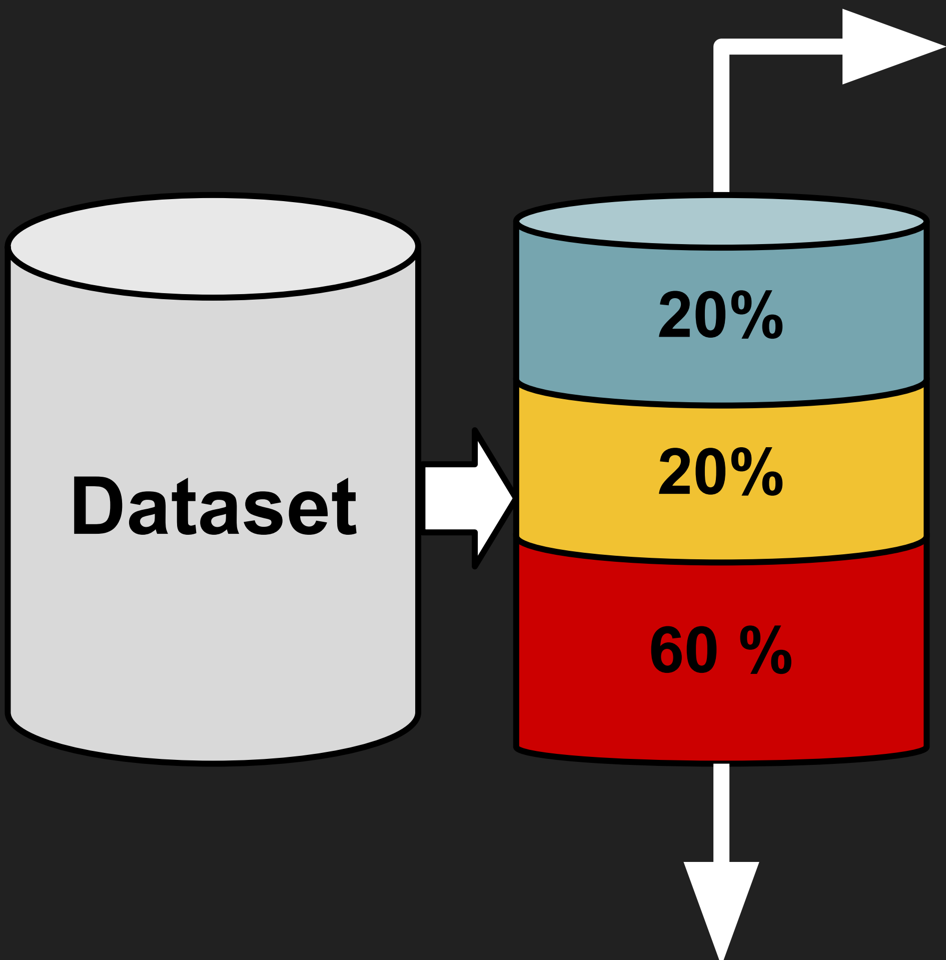
Regularisation: Early stopping



Regularisation: Early stopping



Train / Test / Validation



Test

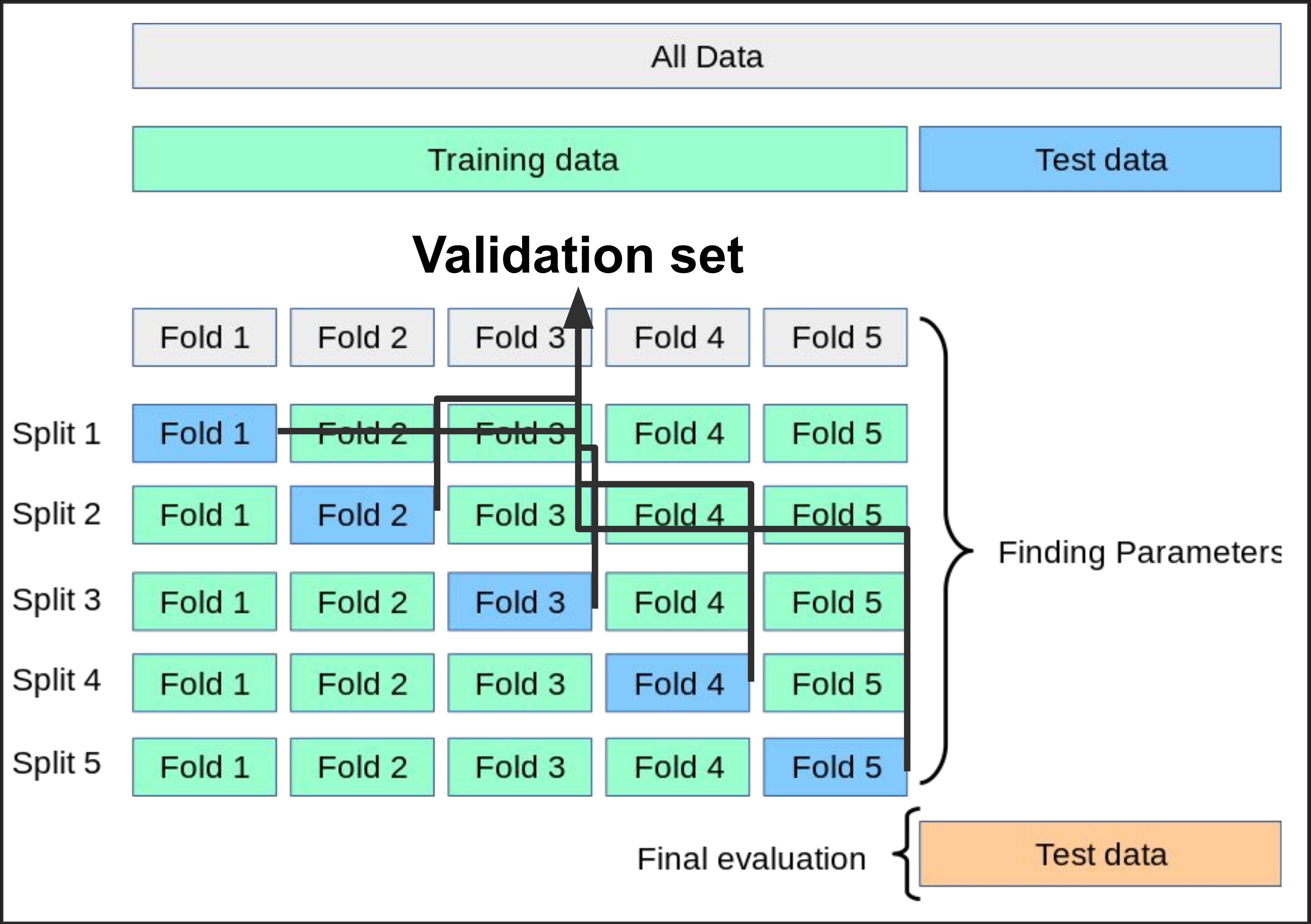
- Mesure model performances

Validation

- Choice of hyper-parameters (Ex: number of layers)

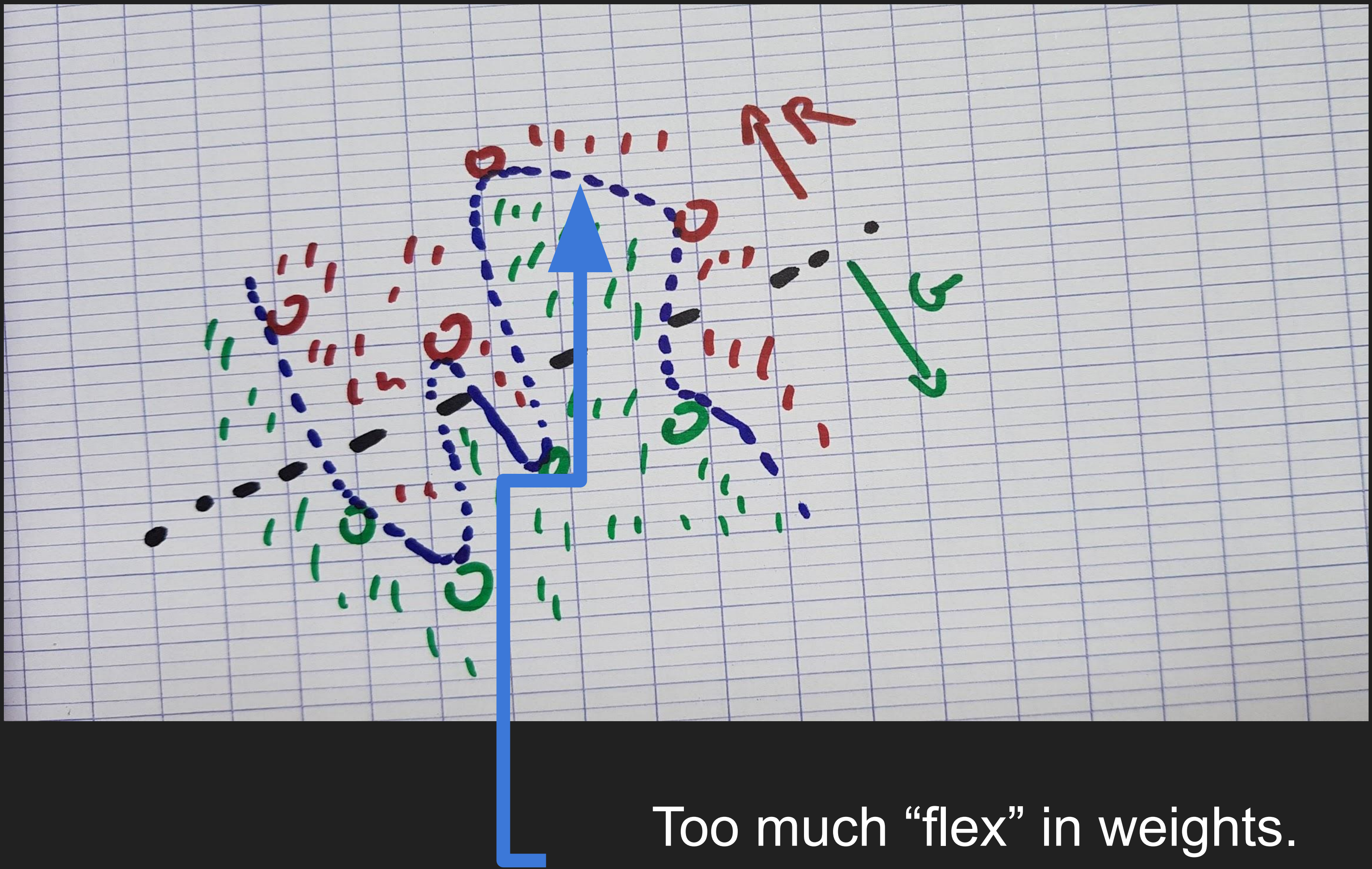
For training **only!**(train)

Cross validation: no separate ‘validation’ set



Adapted from sklearn

Regularisation: limit network capacity



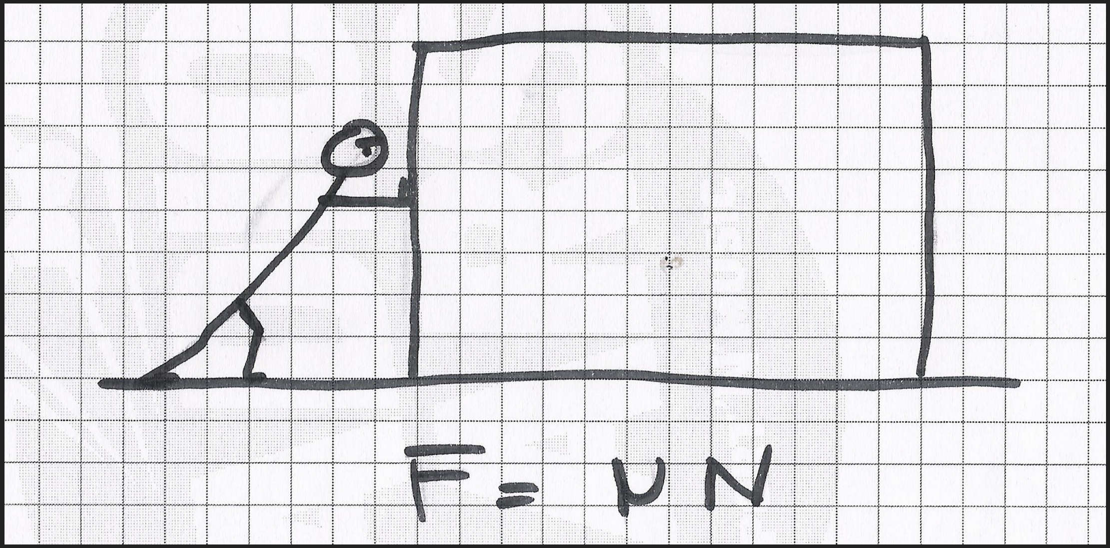
Too much “flex” in weights.
Weights are too high.

Preventing overfitting, Regularisation: L1

MeanSquaredError(Y, Z)

Loss_reg = Loss + **0.0001 * ||W||**

Coefficient



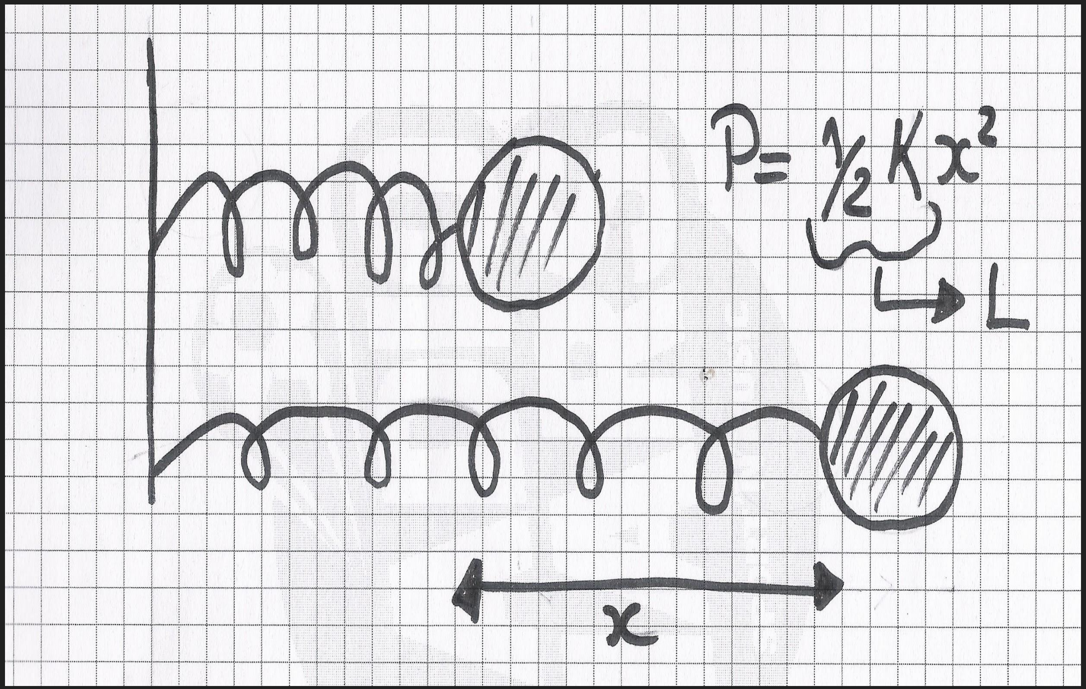
Weights -> 0

Preventing overfitting, Regularisation: L2

MeanSquaredError(Y, Z)

Loss_reg = Loss + **0.00018 * ||W||^2**

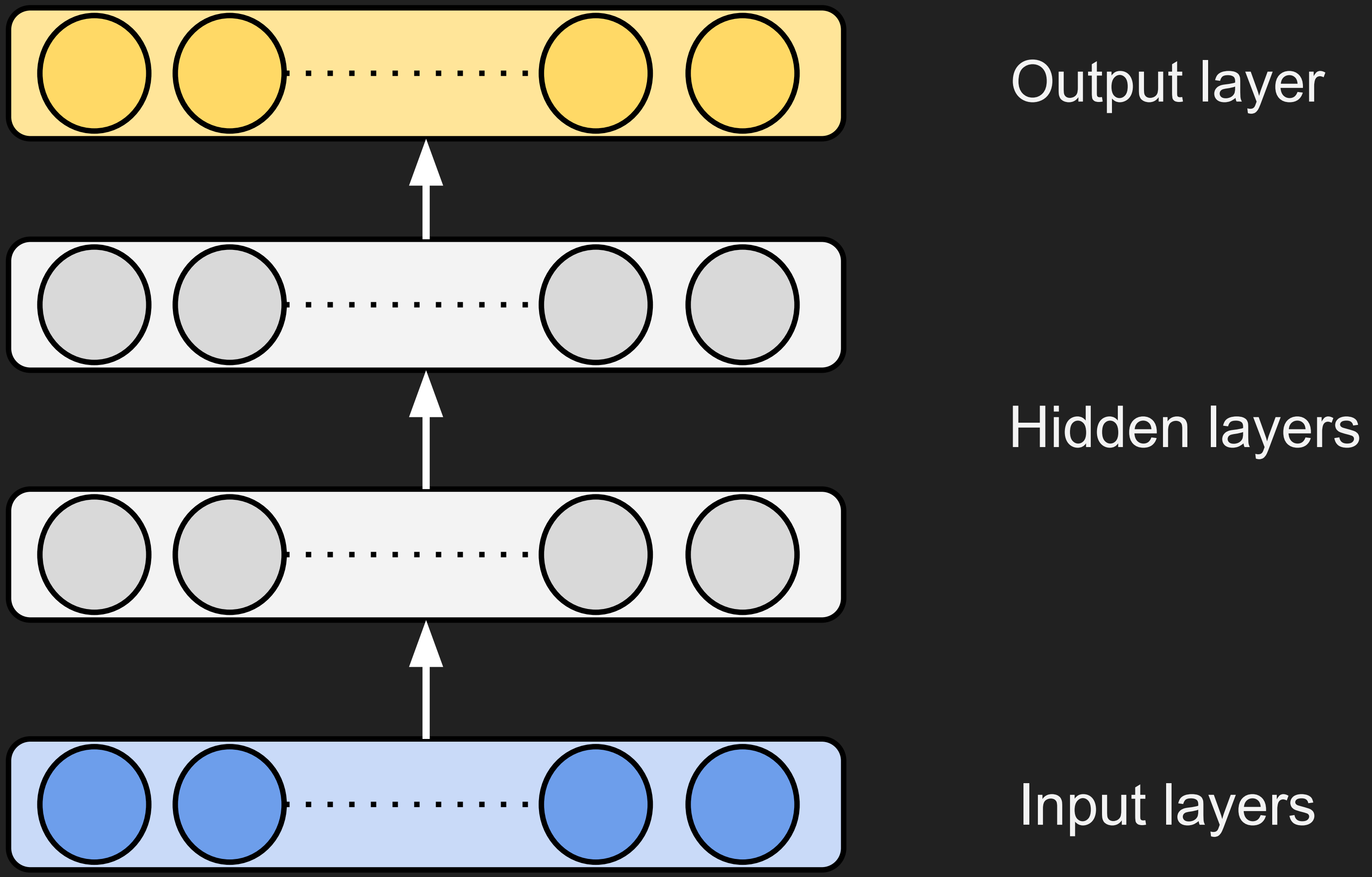
Coefficient



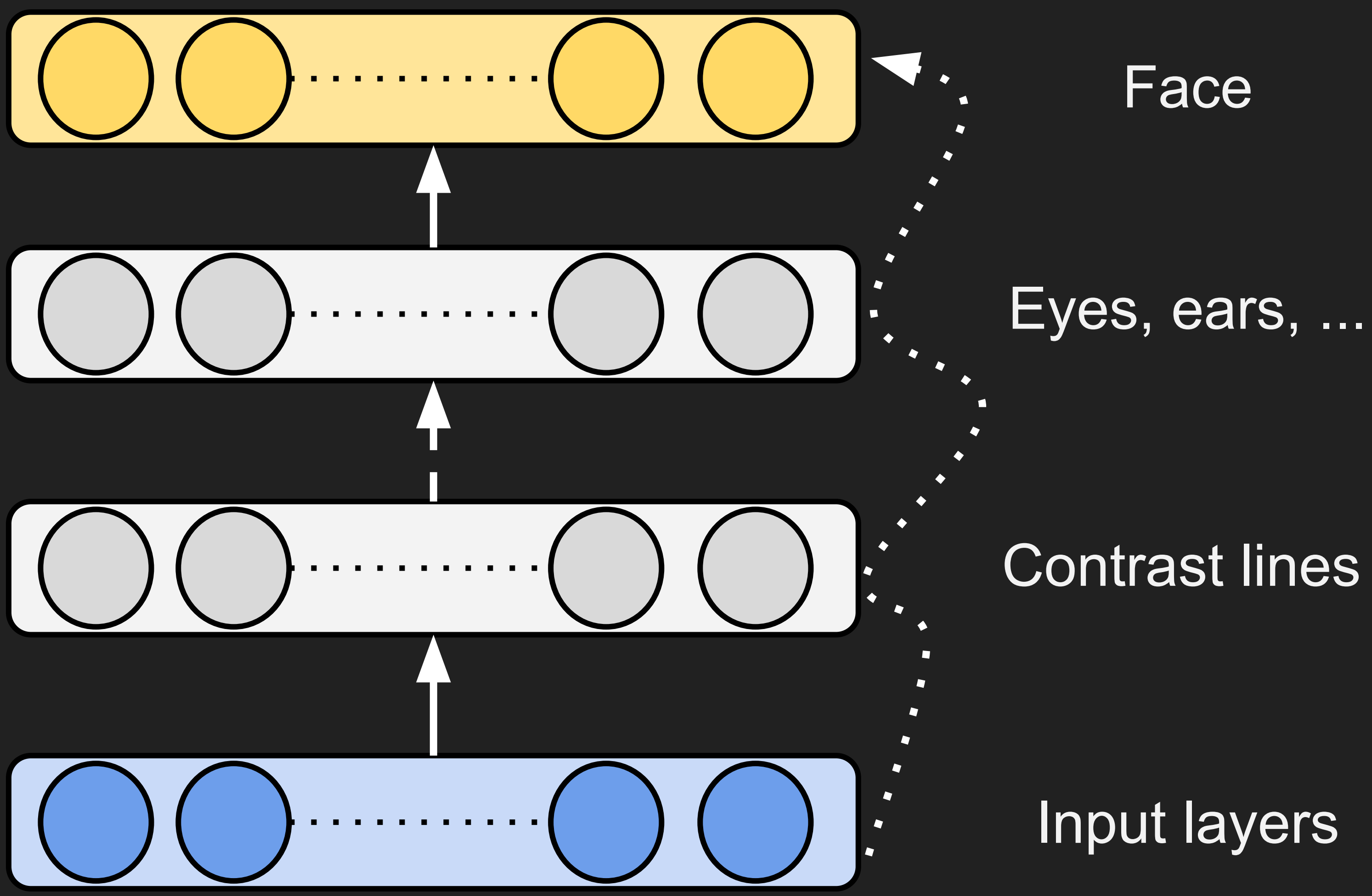
Weights -> small

Deeper networks

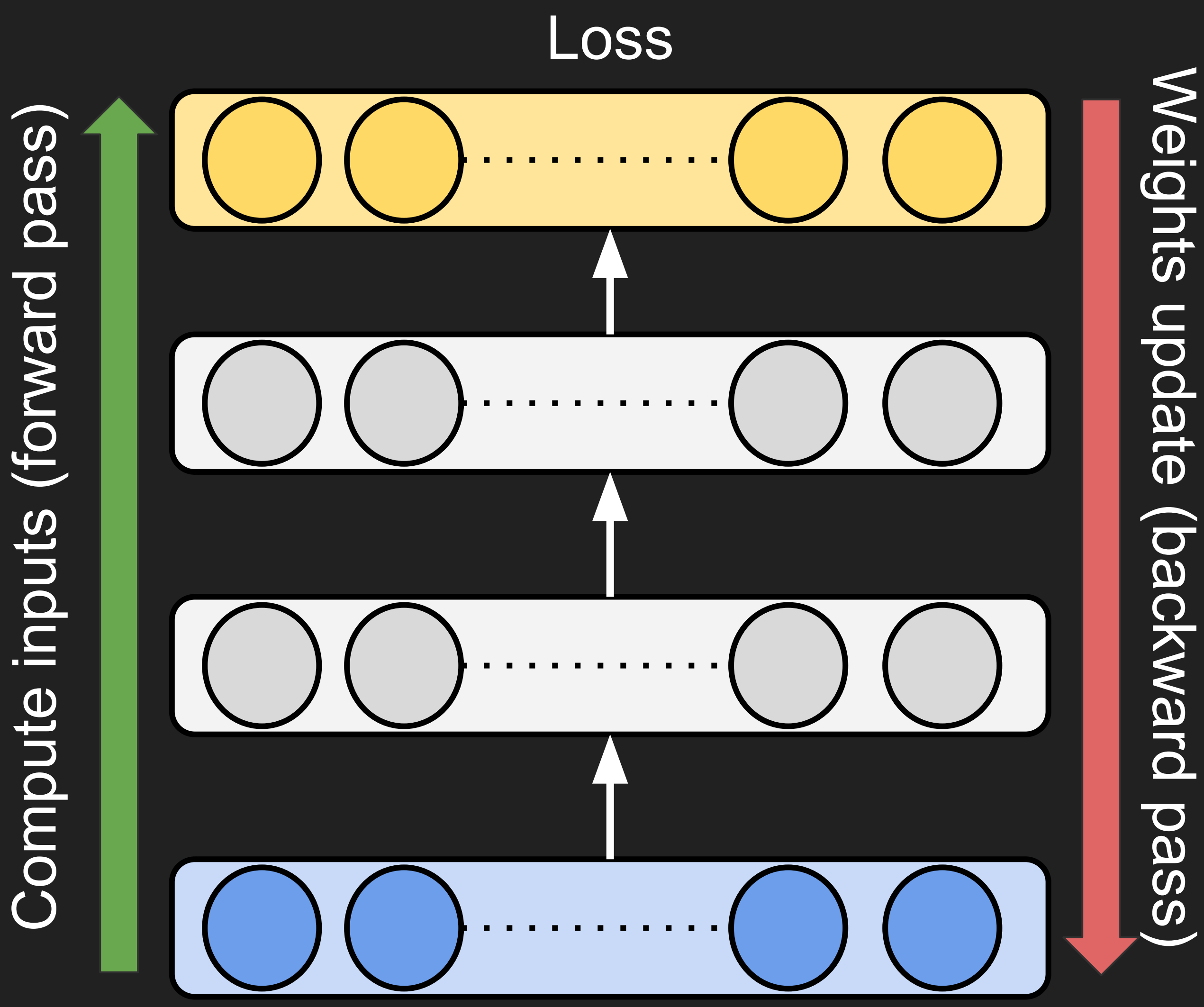
Deep neural networks: A lot of layers



Deep neural networks: Higher levels of abstraction

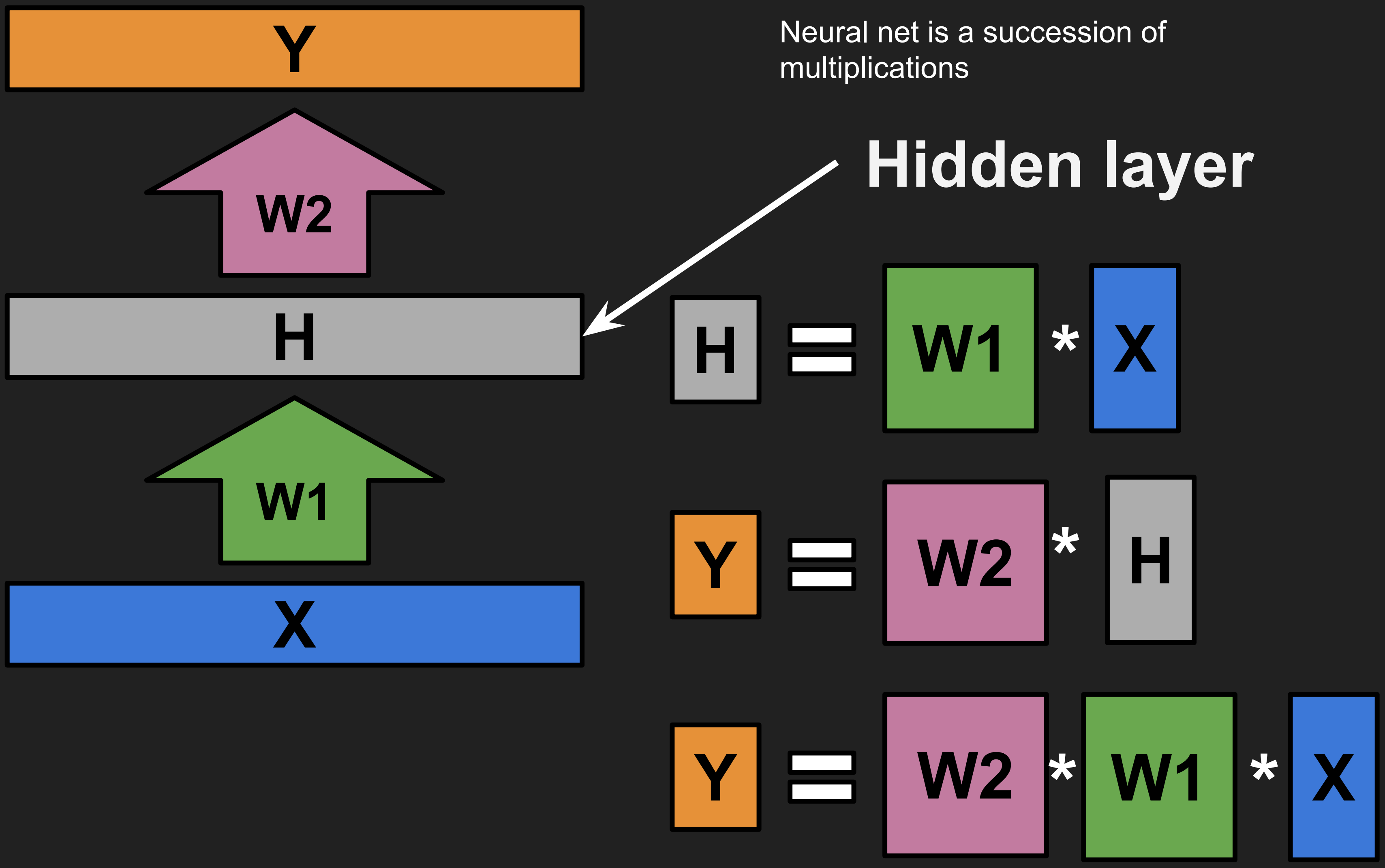


Gradient descent (chain rule)

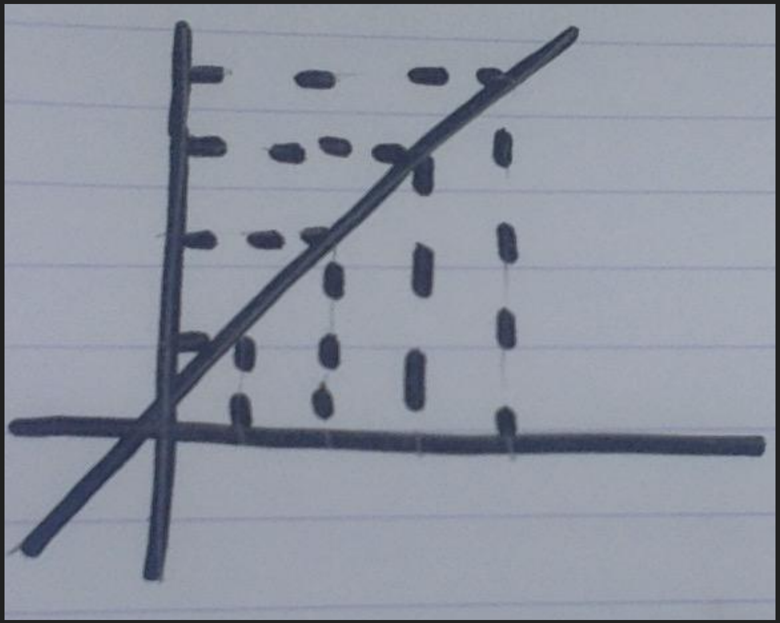


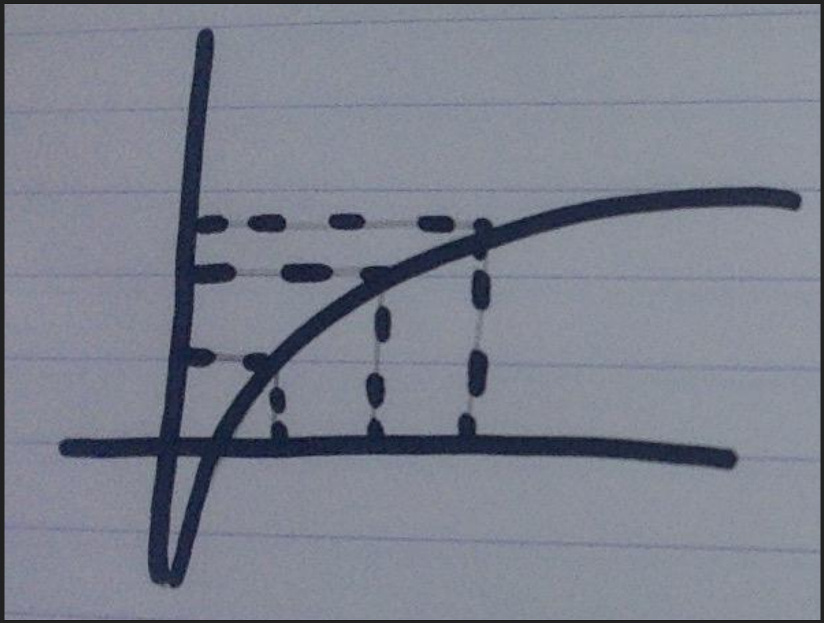
How networks are represented

Neural net is a succession of multiplications



Non-linearities

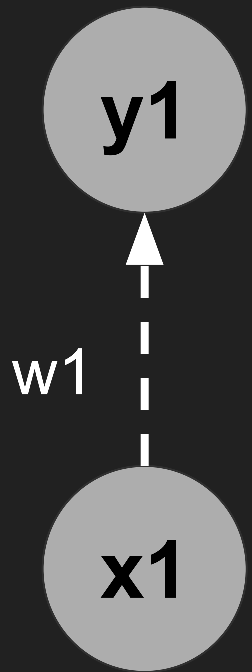




- Y = ax
- Scaling, rotations
- Conserve relationships between points

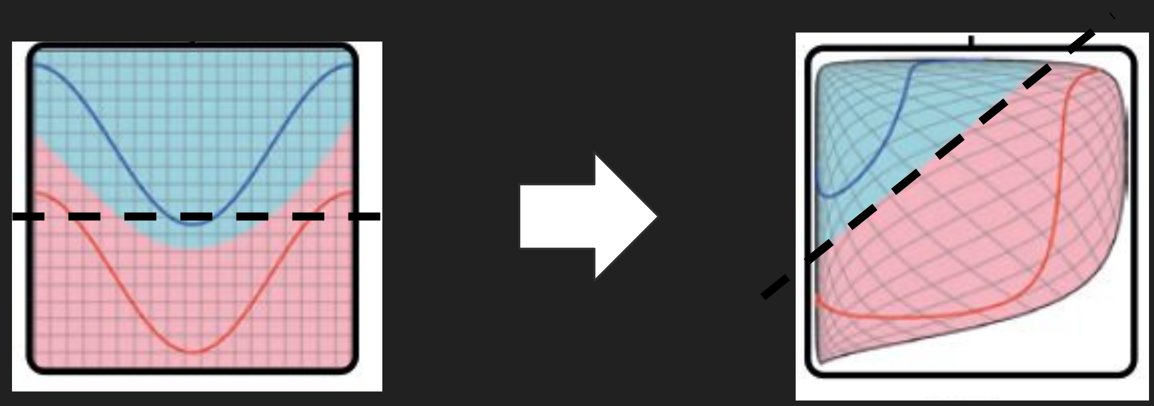
- Change relationships between points
- Space is “bended”
- More flexibility

Why use non-linearities a.k.a activation functions?

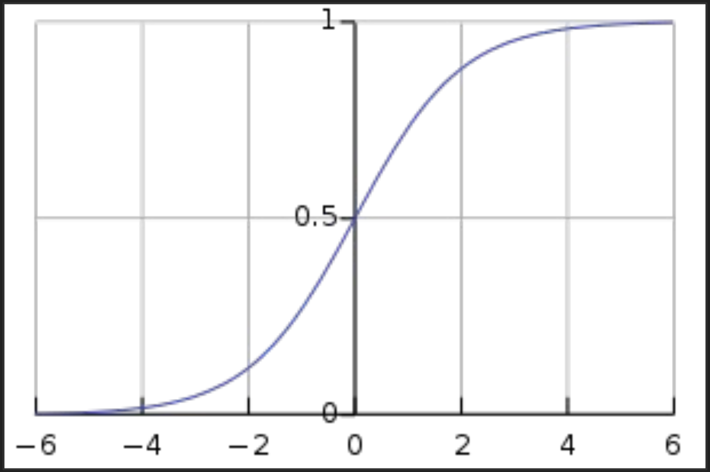


y1 = F(w1 * x1)

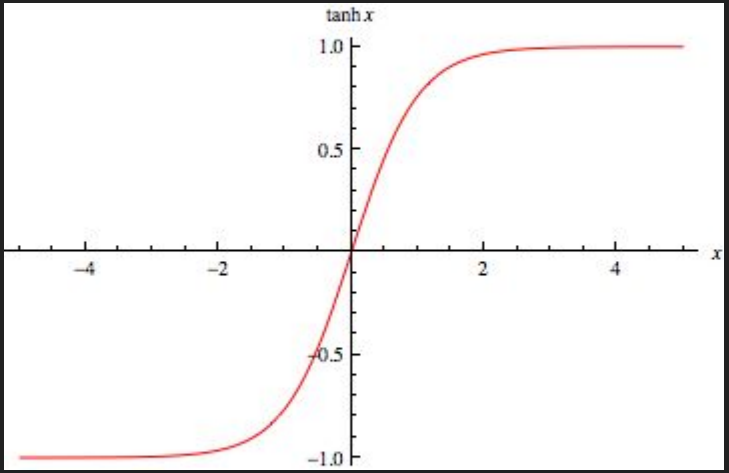
- F: Activation function
- F: non-linearity



Common, non-linearities



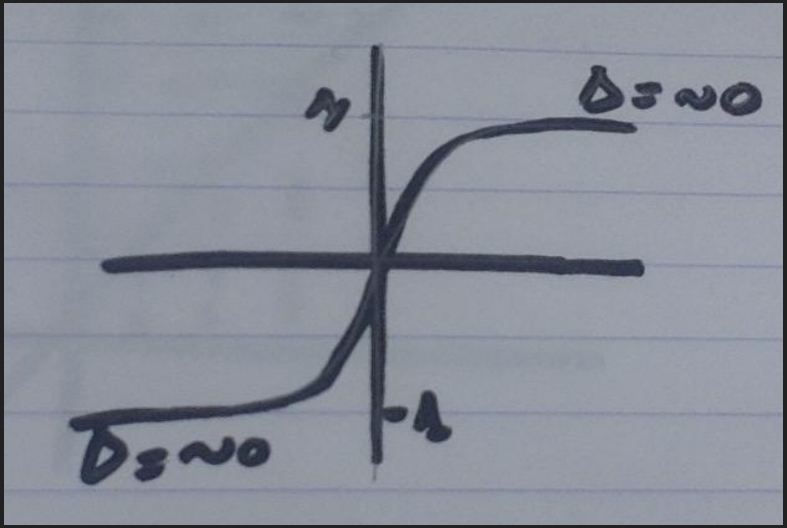
Sigmoid

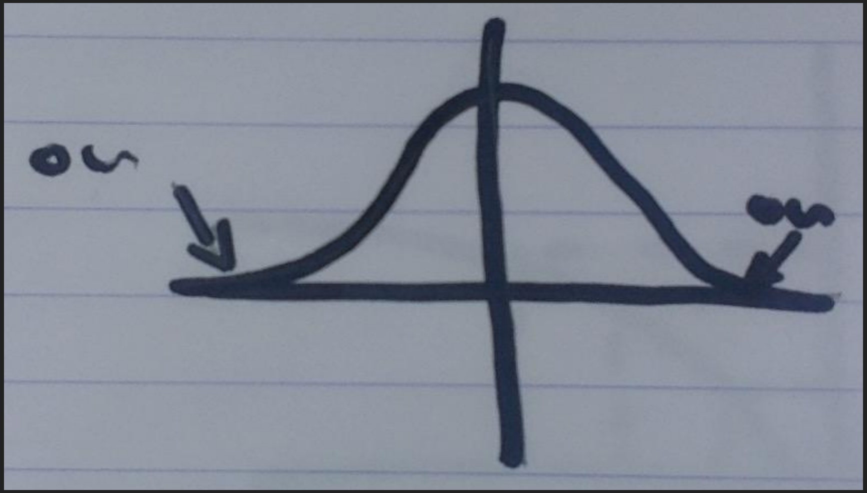


Tanh

y1 = tanh(w1 * x1)

Non-linearities: Saturation





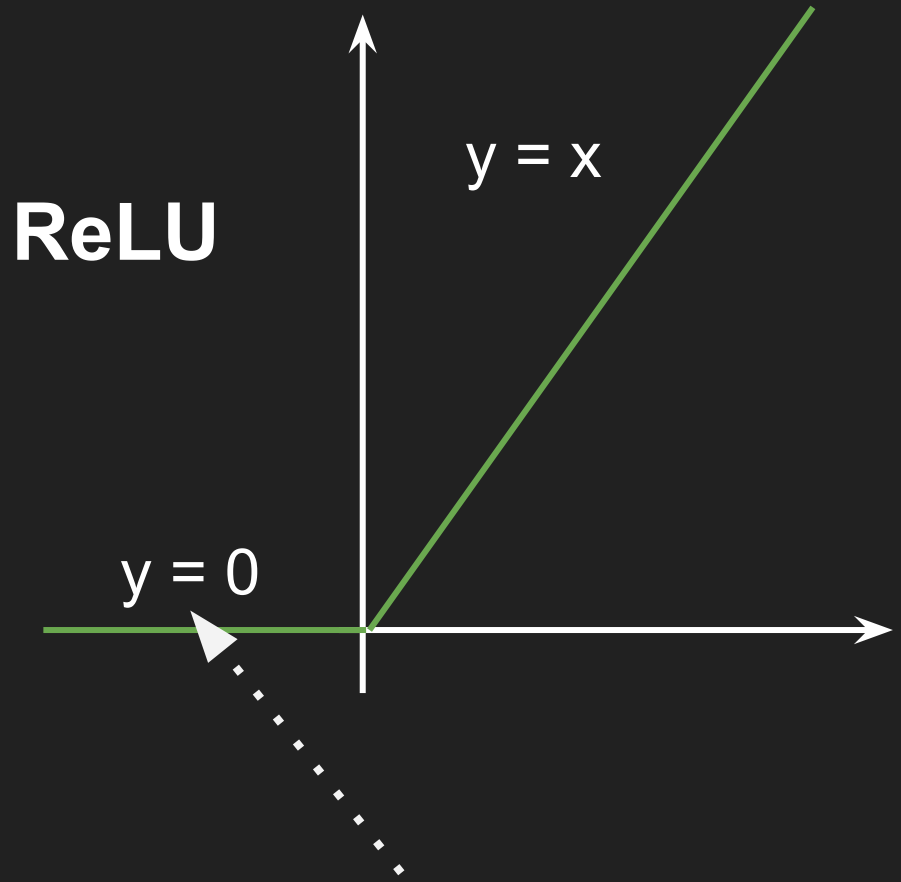
Y close to target

Neuron activation close to 1

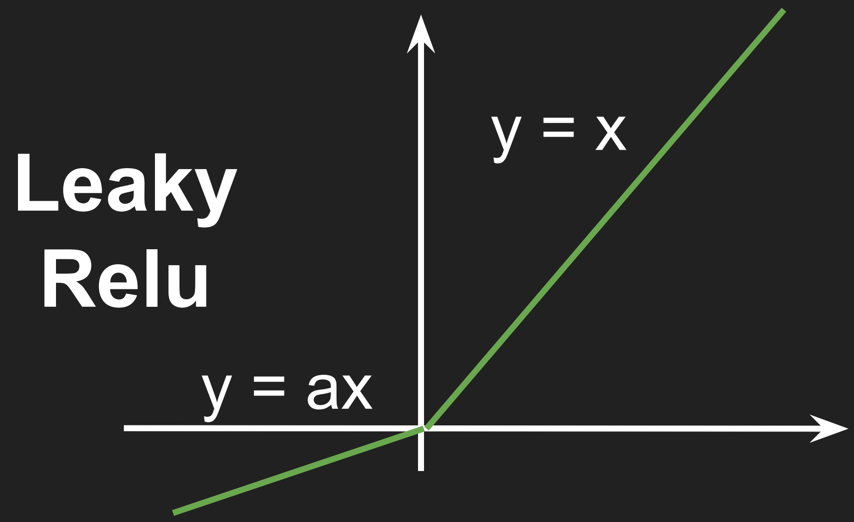
Derivative close to 0

Little update of parameter

ReLU: Rectified Linear Unit



- Fast to compute
- Non-saturating



- Death of the neuron

Loss functions for classification

True probability for class x

<u>1 for x, 0 for all others</u>

Network's output for class x

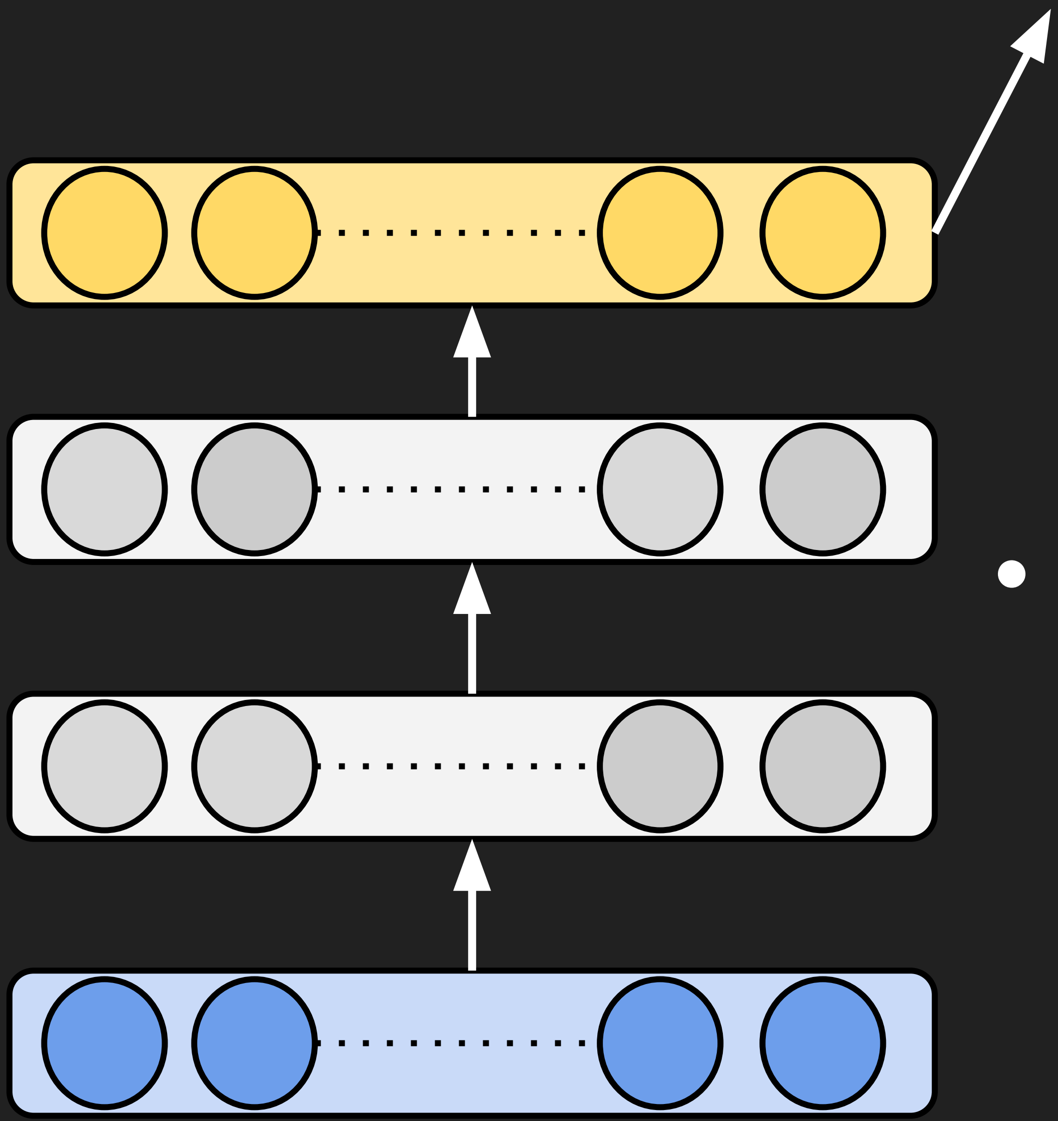
$$H(p,q) = -\sum_{x \in \mathcal{X}} p(x) \log q(x)$$

Everything cancels out!

$$H(p, q) = -\log(q(x))$$

Negative log-likelihood

Multi-class classification



- Activation: Softmax

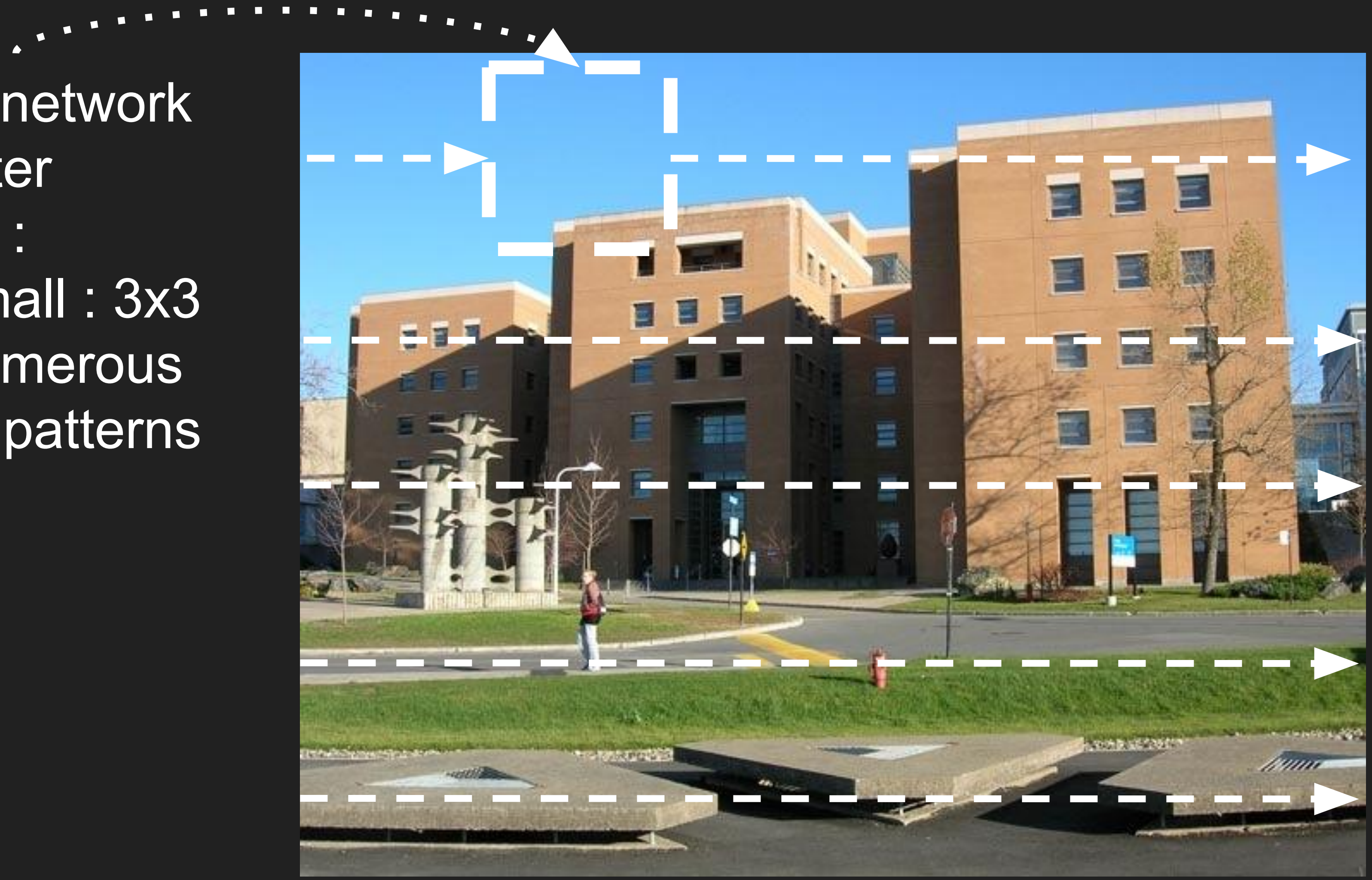
$$P(y=j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^\mathsf{T}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\mathsf{T}\mathbf{w}_k}}$$

- Loss
  - One right answer
    - Negative Log Likelihood
  - More than one
    - Cross entropy

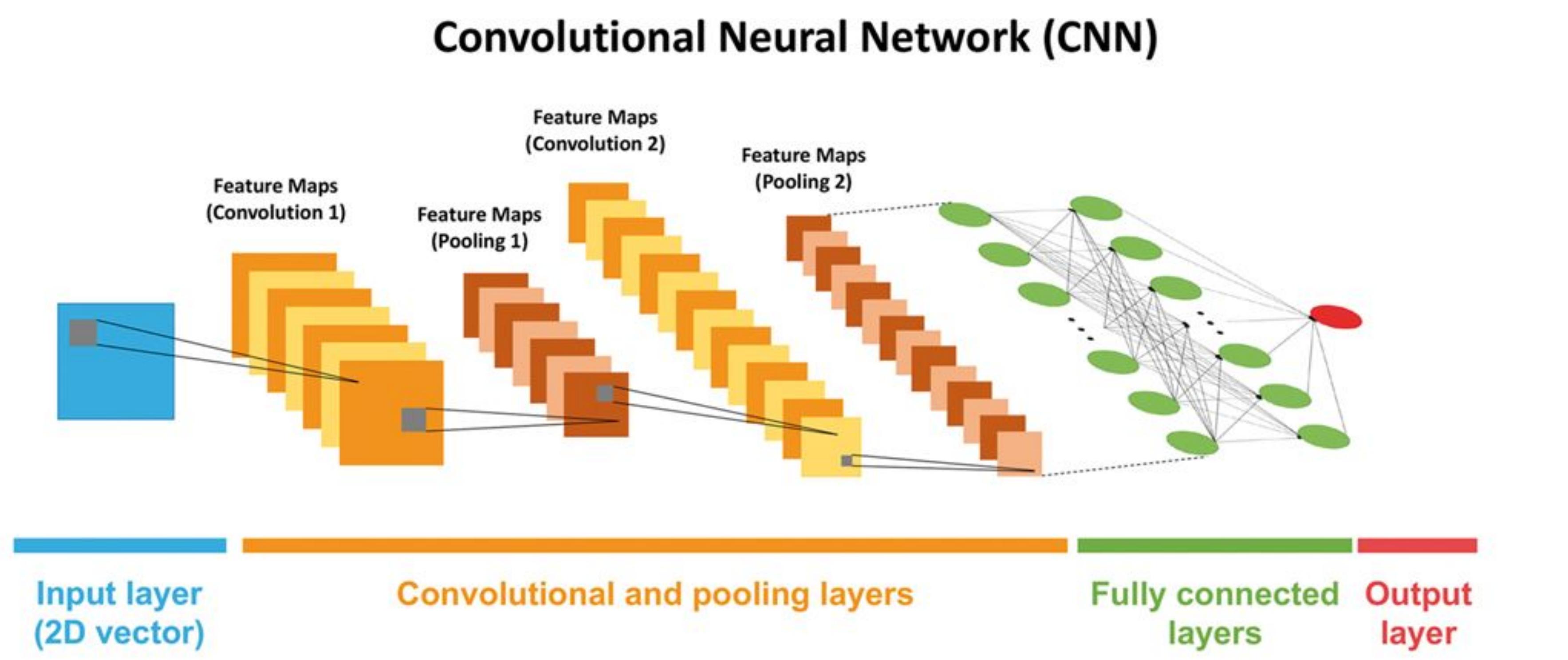
Convolutional Neural Networks

Convolutional Neural Networks

- Small network
  - Filter
- Filters :
  - Small : 3x3
  - Numerous
- Learn patterns

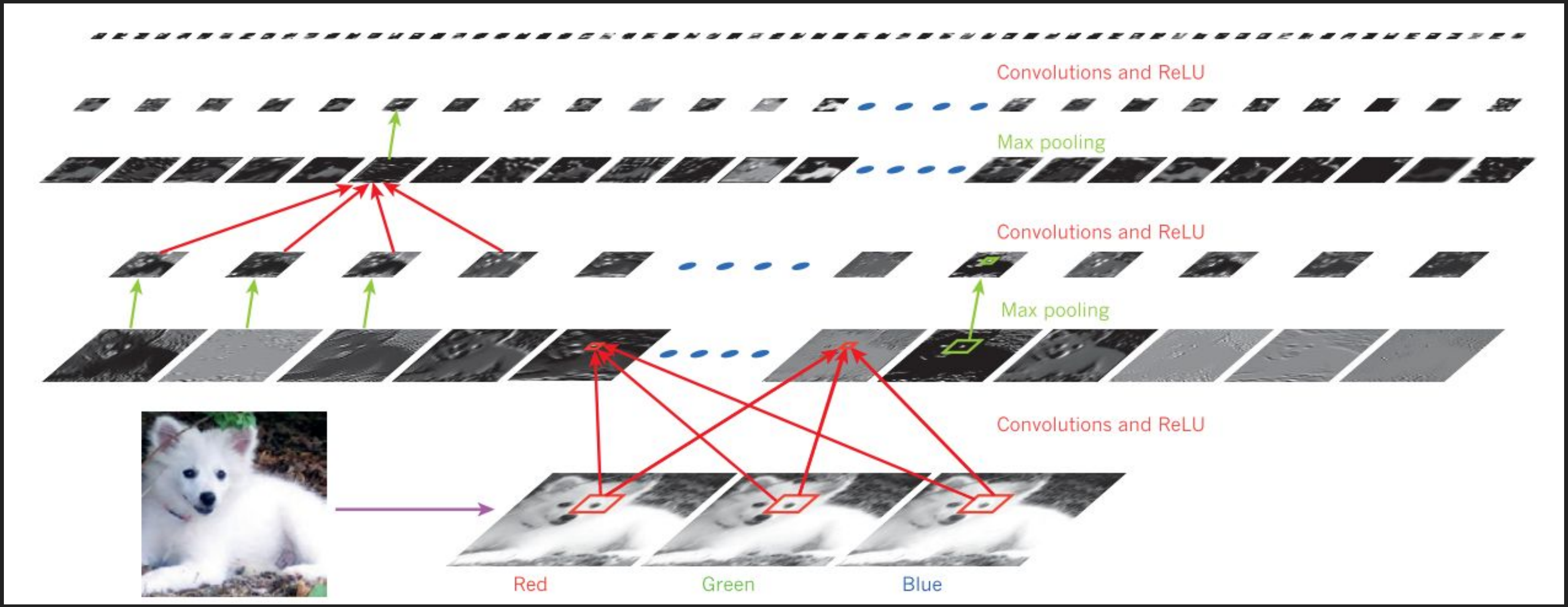


Convolutional Neural Networks



Sureyya Rifaioglu, et al, Brief in Bioinformatics, 2018 https://doi.org/10.1093/bib/bby061

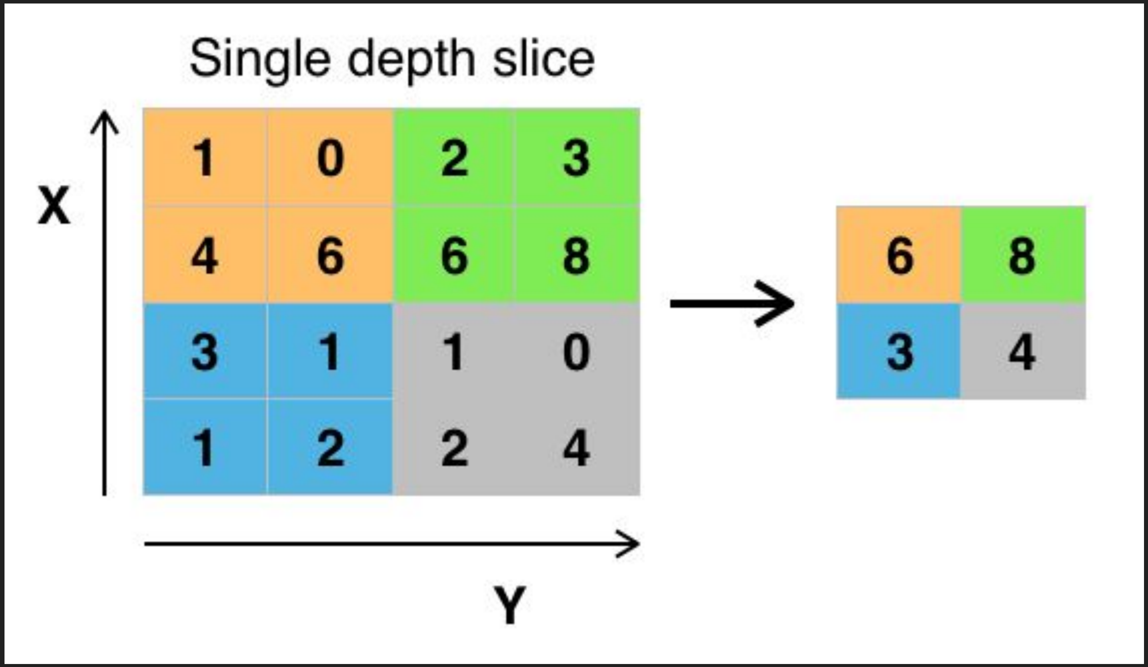
Convolutional Neural Networks



LeCun, Bengio, Hinton. Nature 2015

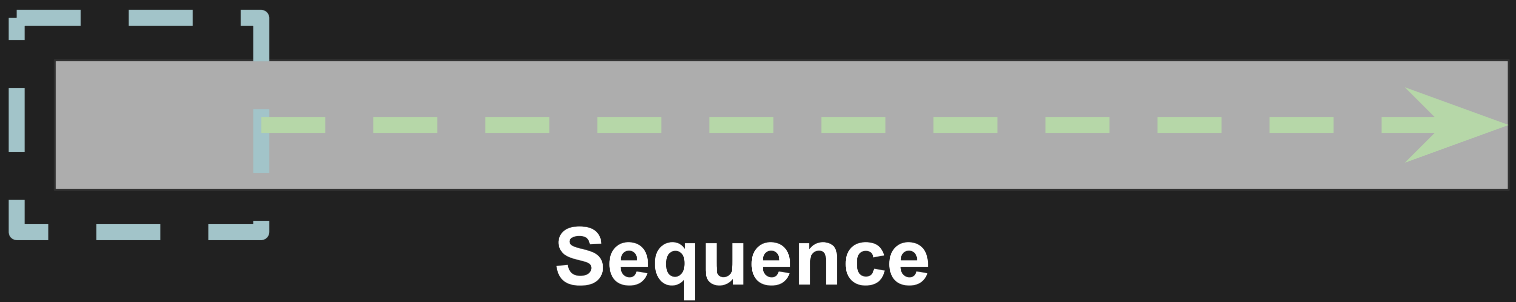
Sub-sampling, ex: max-pooling

Aphex34, wikipedia



- Reduce number of parameters
- Resistant to small translations

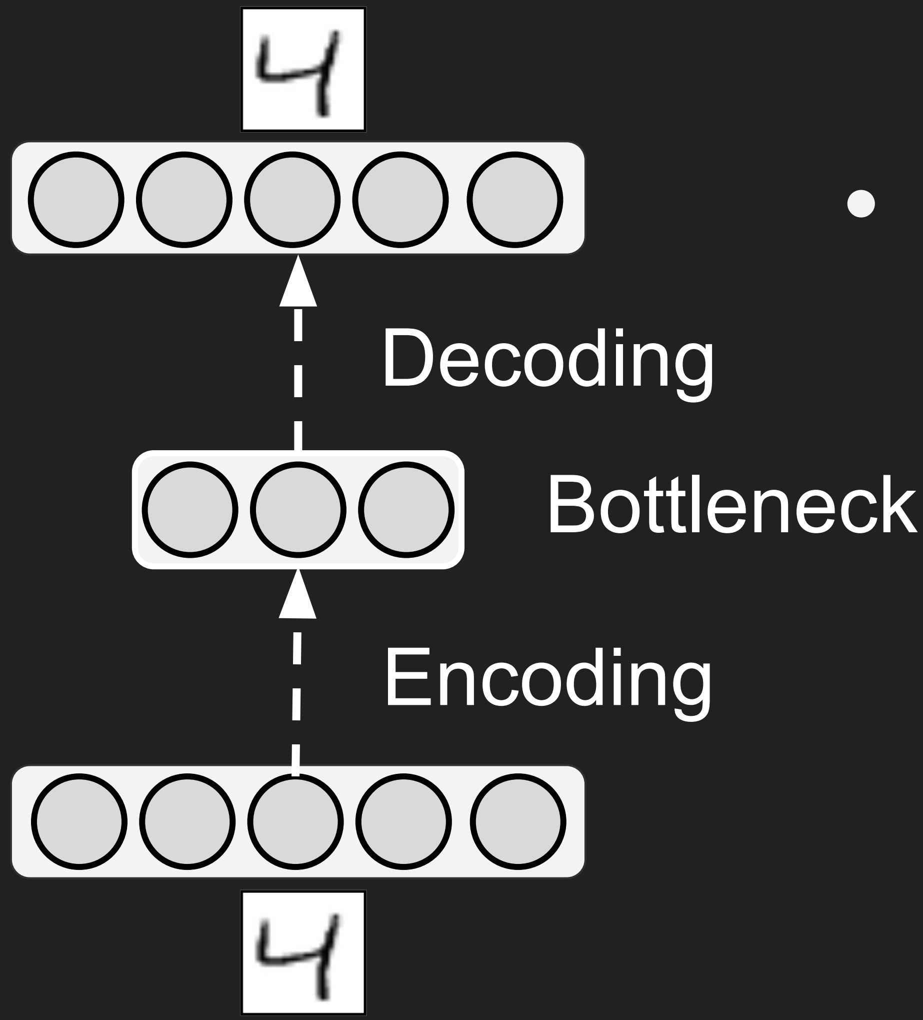
1D Convolutions



Find patterns in sequences!

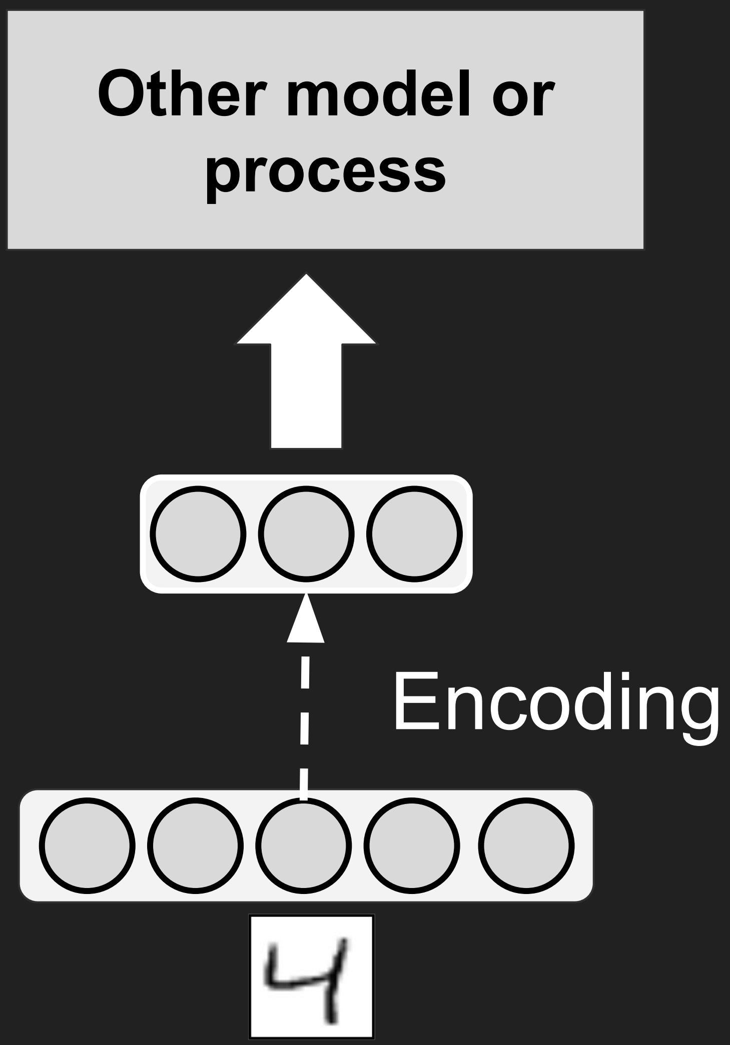
Autoencoders

Dimensionality reduction : Autoencoders



- Bottleneck neurons "summarise" inputs in lower dimensionality

Autoencoders



- Reduce input size conserving information
- More “meaningful” input
- Improve generalisation
- Reduce overfitting

- Visualization

.FIN.

Tools

Entraînement

Outils : Calculs numériques

SciPy

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

NumPy

Outils : Visualisation

Altair

ggplot2

Visdom

Seaborn

Outils : bases de données

ArangoDB

MySQL

mongoDB

PostgreSQL

Outils : Apprentissage

PyTorch

Keras

scikit learn

MARIANA