Introduction to Deep Learning

A course by EuroCC Austria

Speaker: Simeon Harrison
Trainer at EuroCC Austria

EURO
AUSTRIA

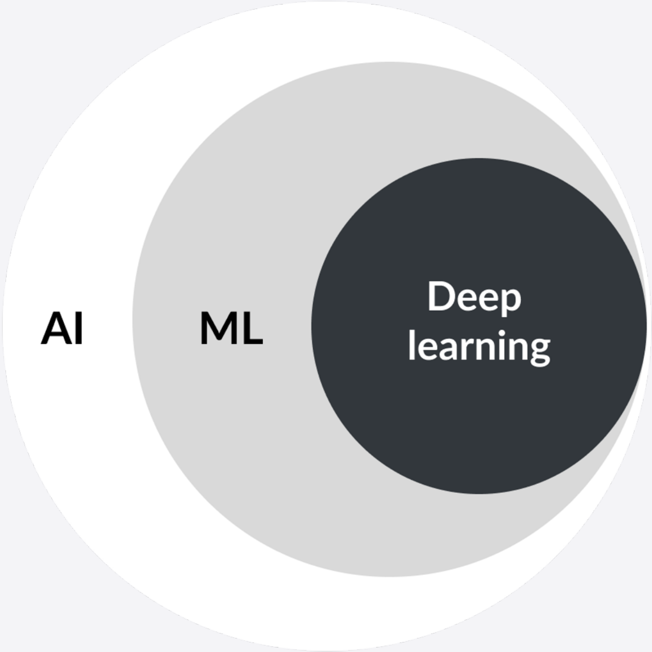
Deep Learning is a subset of Machine Learning

“Deep” does not mean a deeper understanding of the problem at hand. “Deep” stands for many successive layers of abstract representation

Representations are learned via models called artificial neural networks

Term “neural network” stems from neural biology. Models were inspired by our understanding of the brain

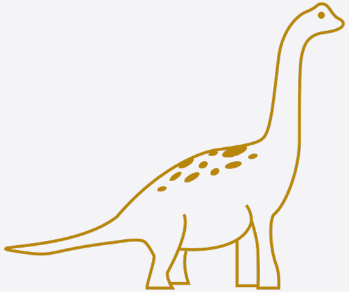
First coined in the 1940



Deep Learning is a multi-stage information distillation process. Information goes through successive filters and gets more and more purified

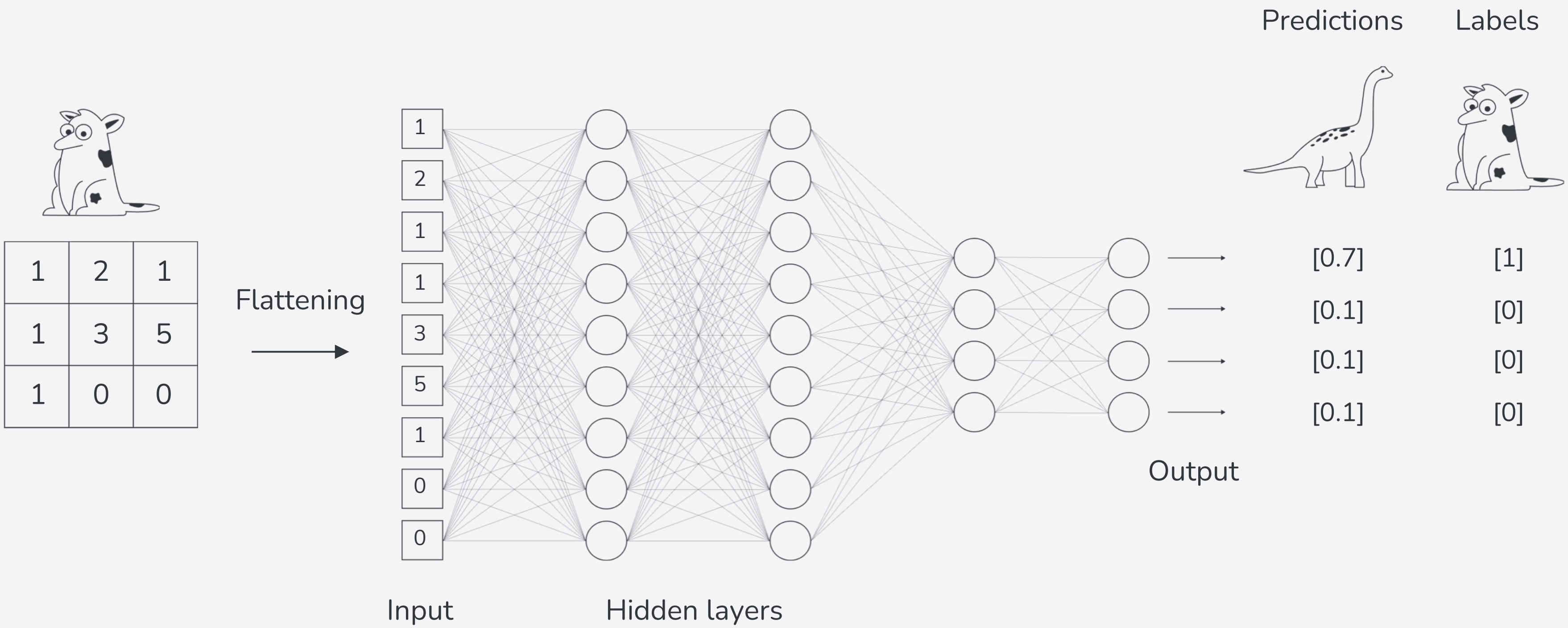
Learning happens by exposure to examples i.e. mapping inputs to targets

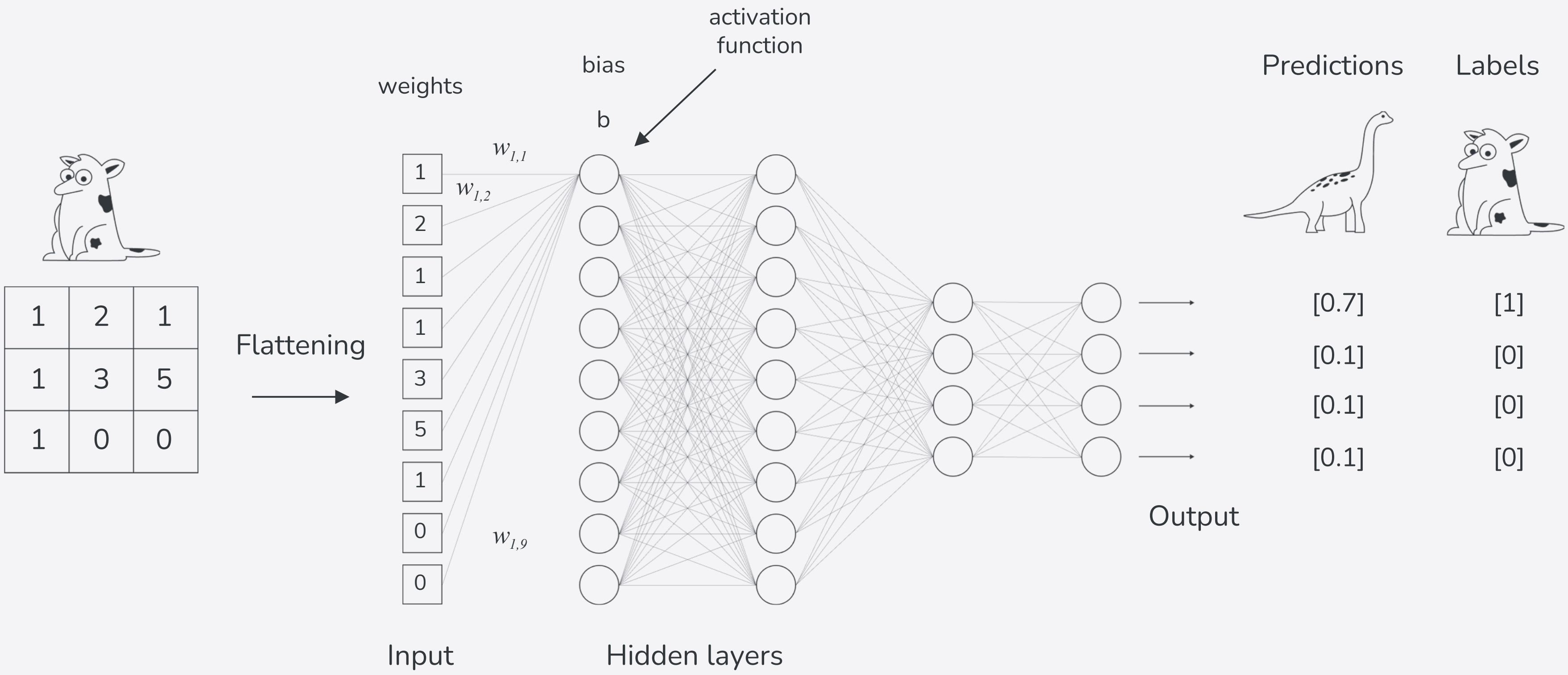
This is a dog



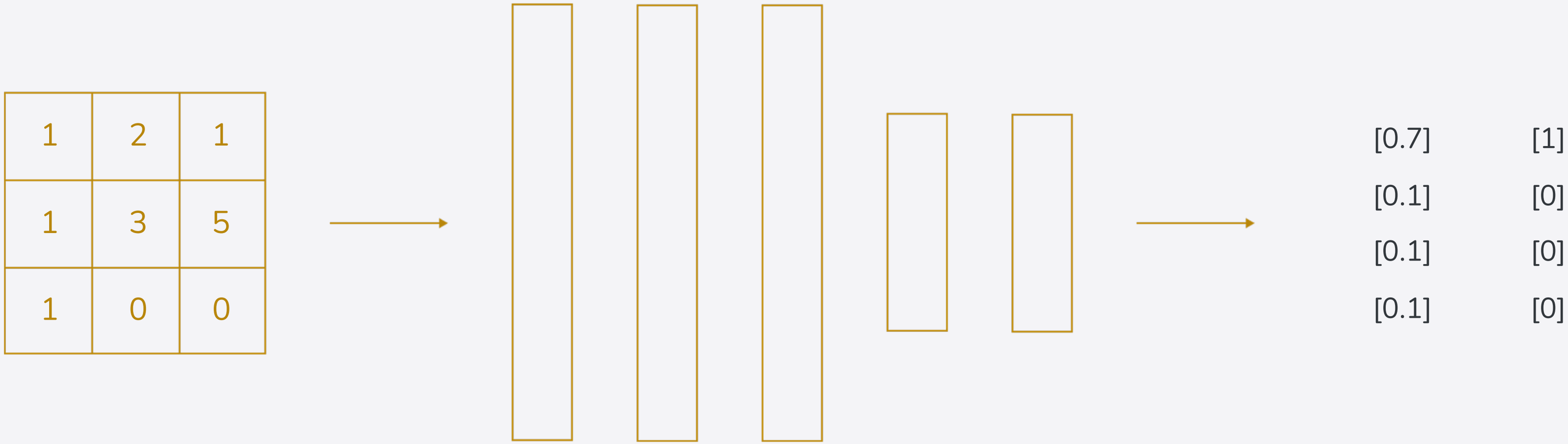
This is (most likely) not a dog

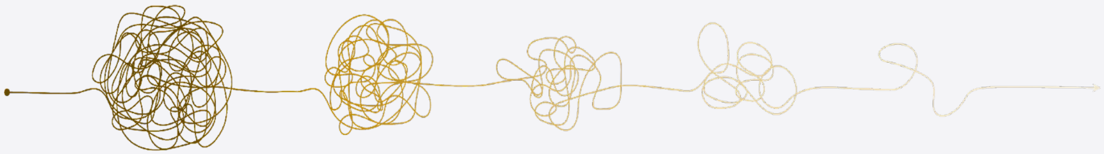
From Input to Output

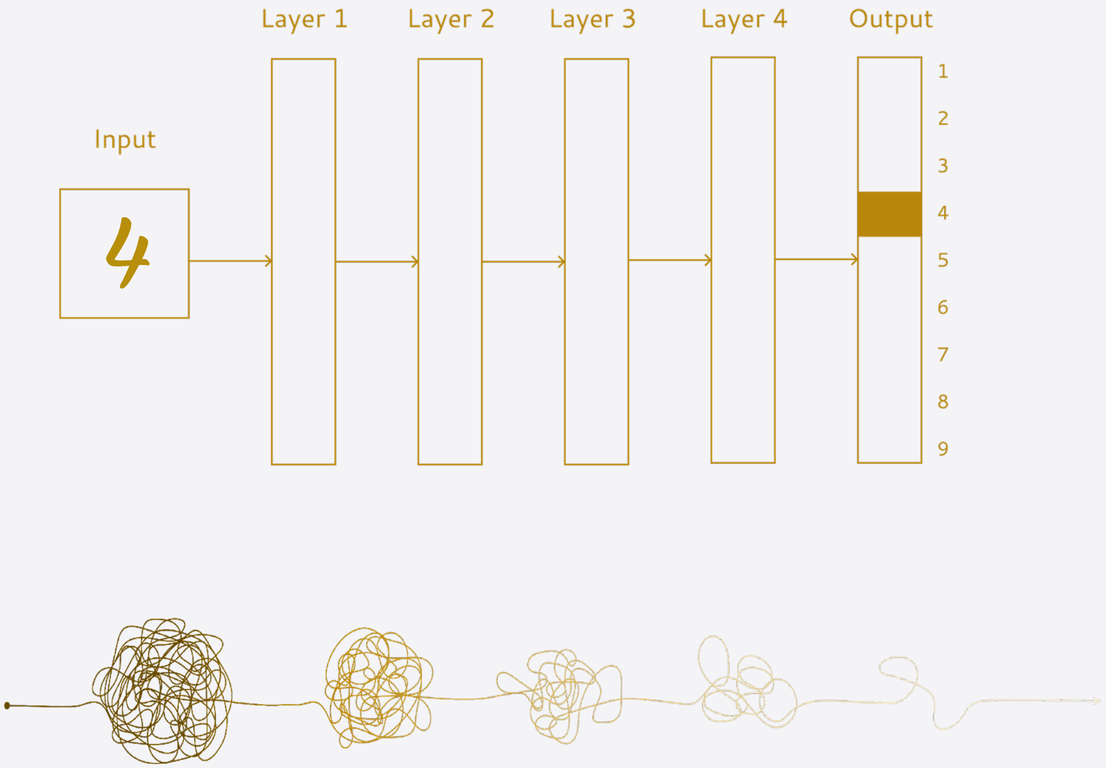


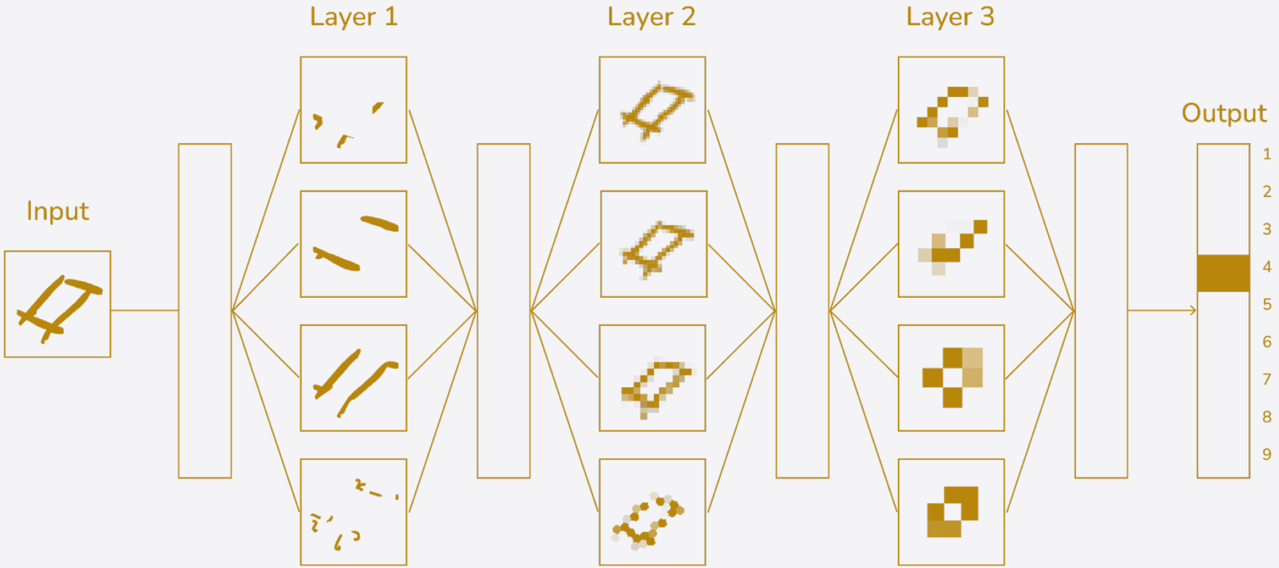


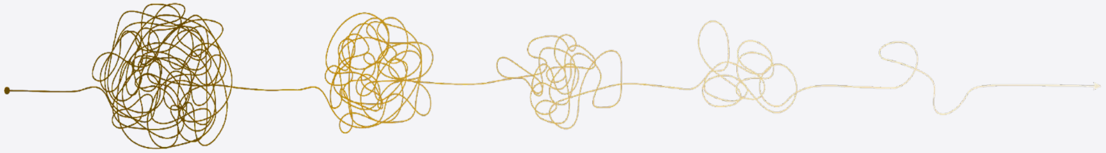
1
2
1
1
3
5
1
0
0
Flattening
weights
bias
b
activation
function
$w_{1,1}$
$w_{1,2}$
$w_{1,9}$
1
2
1
1
3
5
1
0
0
Input
Hidden layers
Output
Predictions
Labels
[0.7]
[0.1]
[0.1]
[0.1]
[1]
[0]
[0]
[0]


1 2 1
1 3 5
1 0 0
[0.7] [1]
[0.1] [0]
[0.1] [0]
[0.1] [0]




Layer 1
Layer 2
Layer 3
Layer 4
Output
Input
4
1
2
3
4
5
6
7
8
9


Layer 1
Layer 2
Layer 3
Input
Output
1
2
3
4
5
6
7
8
9



Weights & Bias

Numerical representation of input needs to be “nudged” the right way to result in desired output

What a layer does to its input data is stored in its weights

Weights are also called parameters of a model

Learning means finding the best set of weights for each layer, so that the input is correctly mapped to the corresponding labels

Bias term is added to each data transformation

Bias term is not always needed

Hyperparameters

The actual parameters of the model are the weights & biases

Hyperparameters are set by the programmer to influence the learning outcome

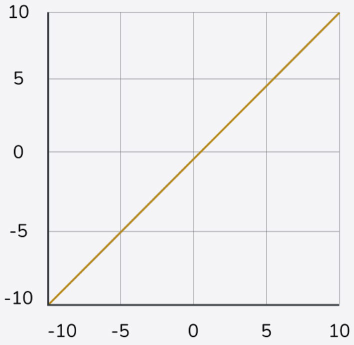
They include:

- Batch size
- Activation function
- Optimizer
- Learning rate
- Epochs

Activation functions

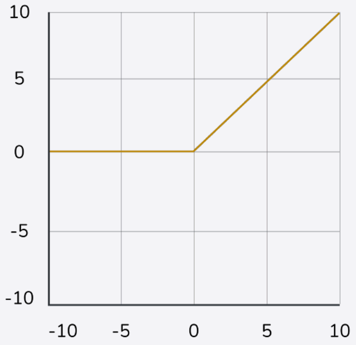
Linear

$$\hat{y} = wx + b$$



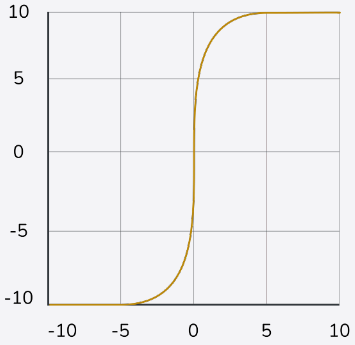
ReLu

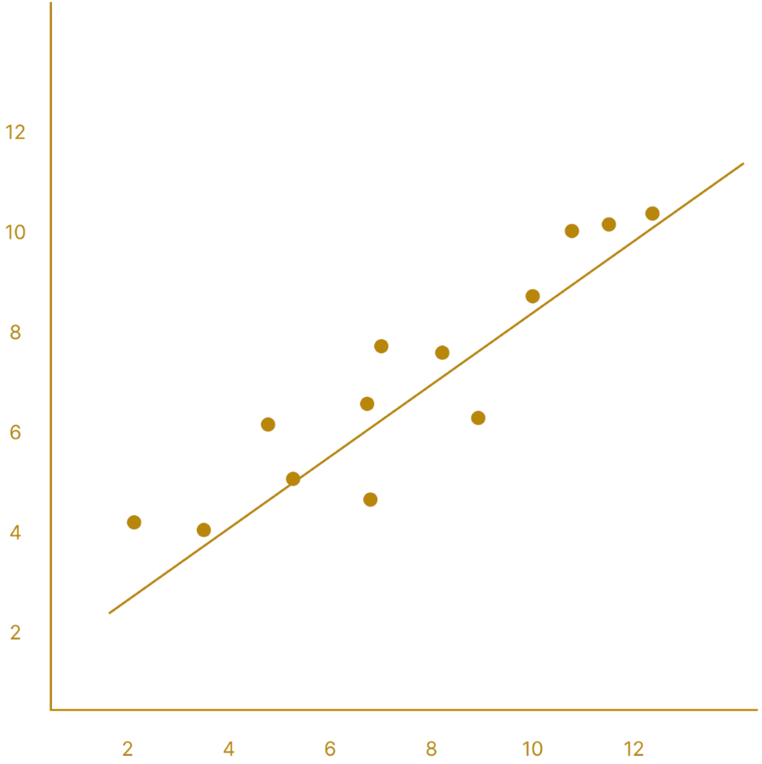
$$\hat{y} = \begin{cases} wx + b \; if \; wx + b > 0 \\ 0 \; otherwise \end{cases}$$



Sigmoid

$$\hat{y} = \frac{1}{1 + e^{-(wx+b)}}$$



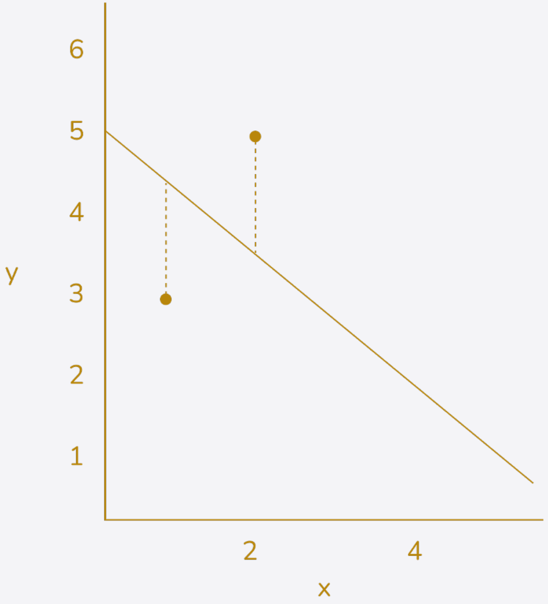


$$MSE = \frac{1}{m}\sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})^2$$

Loss function is some form of averaged difference between the predictions and the true values (e.g. mean squared error)

(Root) Mean Square Error

$\hat{y}$	err^2
4	1
3	4
MSE=	2.5
RMSE=	1.6



$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2$$

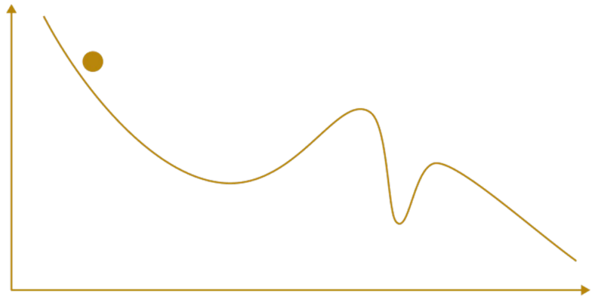
$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2}$$

(Root) Mean Square Error

Optimisation through some form of gradient descent

Iterative process in small steps (learning rate) in the direction of the negative gradient

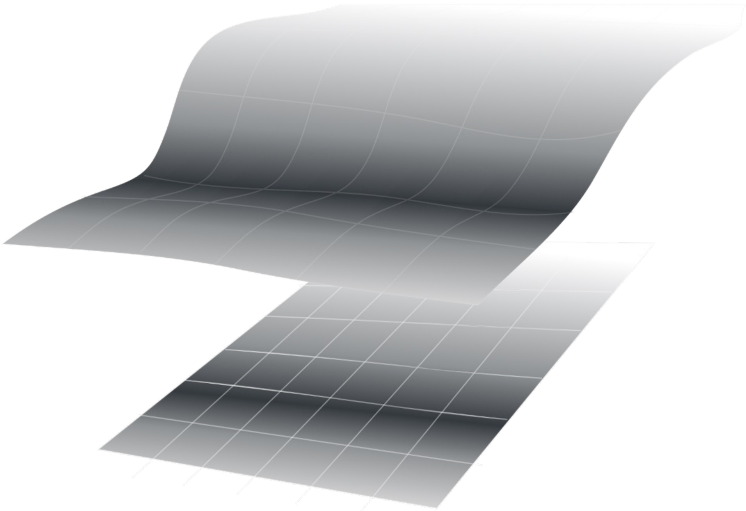
Goal: Find global minimum of loss function

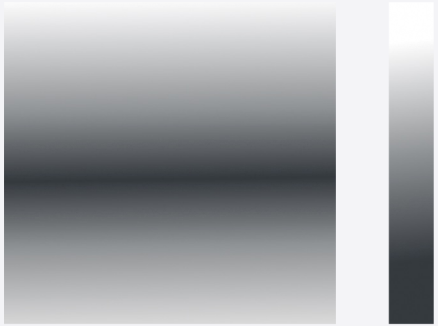


$$\nabla MSE(\theta) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} \\ \frac{\partial}{\partial \theta_1} \\ \vdots \\ \frac{\partial}{\partial \theta_n} \end{pmatrix} MSE(\theta)$$

$$\theta^{(next\ step)} = \theta - \lambda \nabla_{\theta} MSE(\theta)$$

Gradient Descent





Optimisers

There are many optimizers out there, but the most common ones are:

- Stochastic Gradient Descent
- Adam
- RMSprop

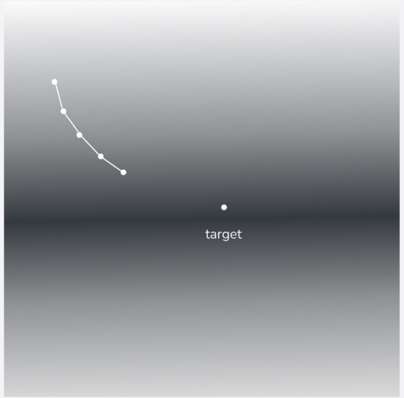
They all use some form of gradient descent. In addition, some use the adaptive learning rates and momentum

Choosing the right optimizer is very much a trial and error decision

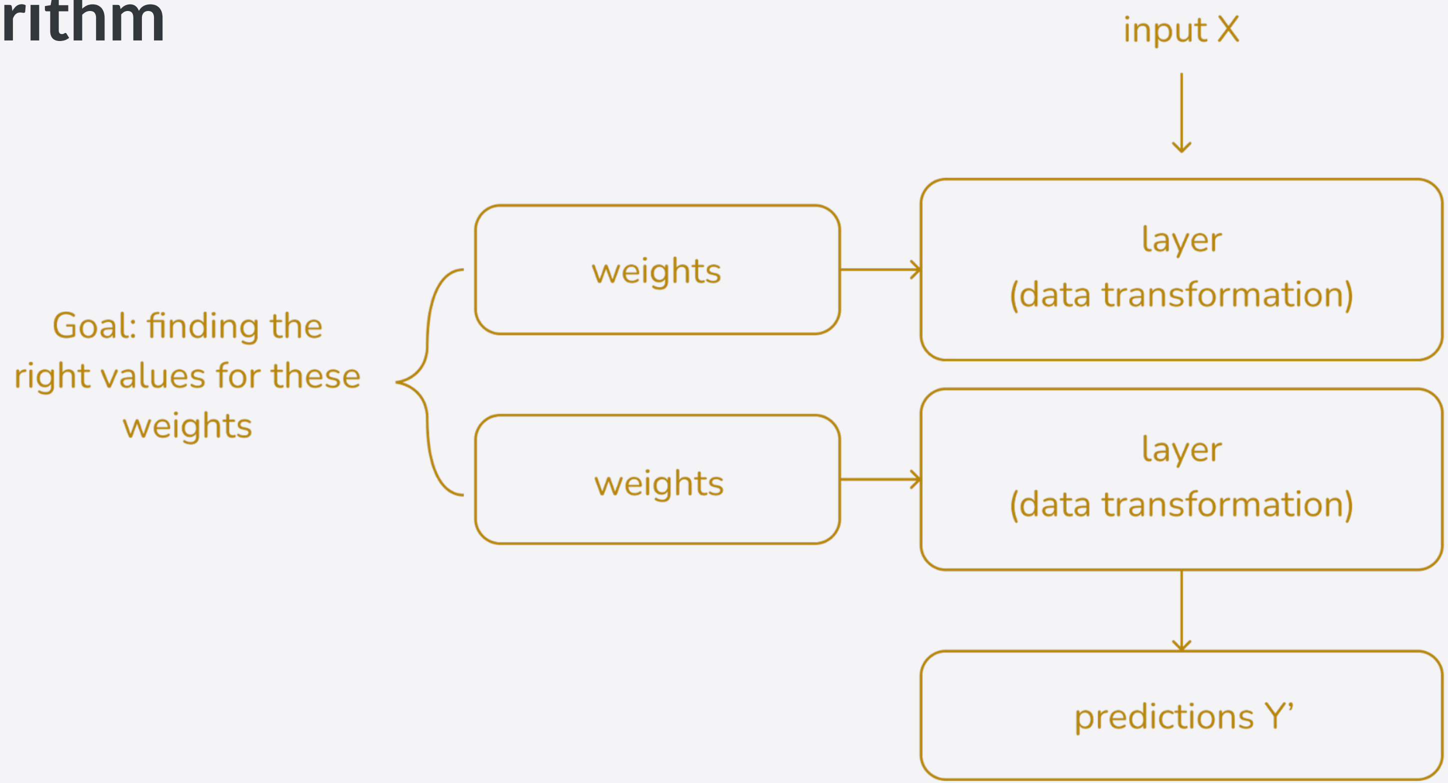
It is advisable to use the default settings of an optimizer (to start with)

Optimisers

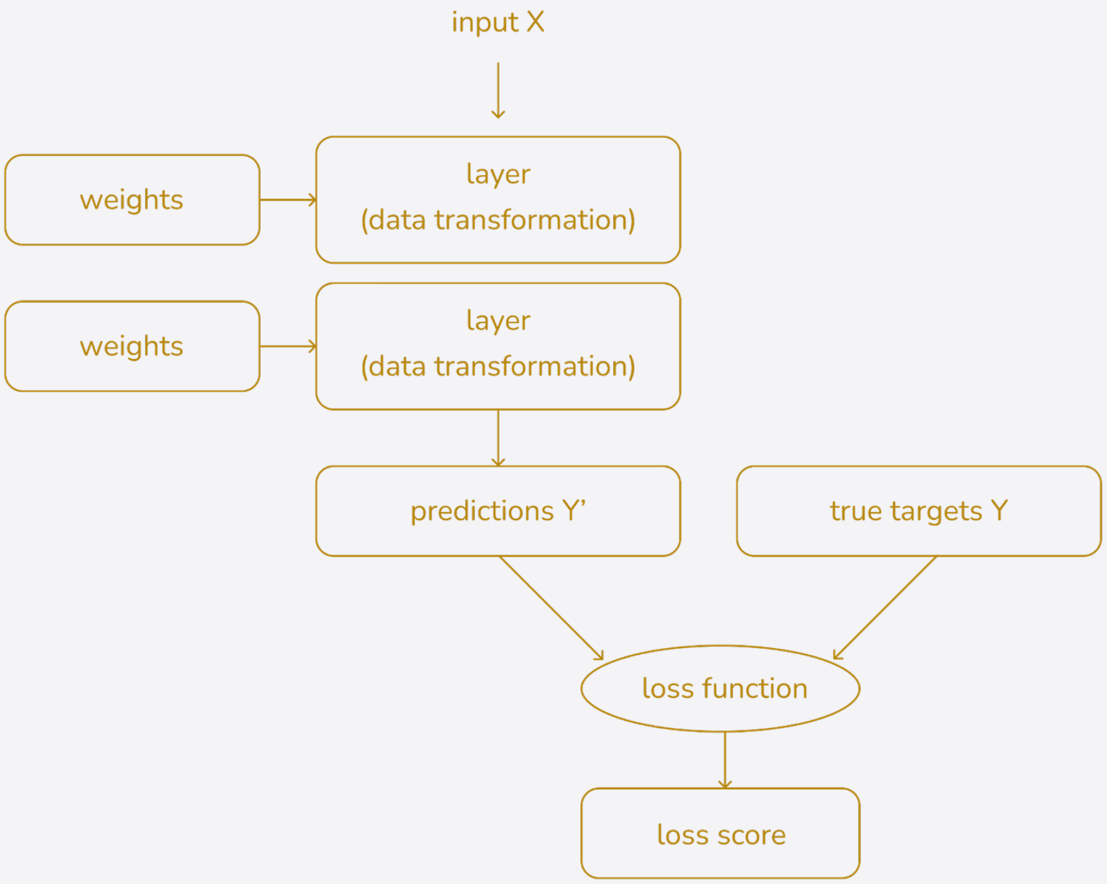
Gradient	Which direction loss decreases the most
λ:The learning rate	How far to travel
Epoch	A model update with full dataset
Batch	A sample of the full dataset
Step	An update to the weight parameters



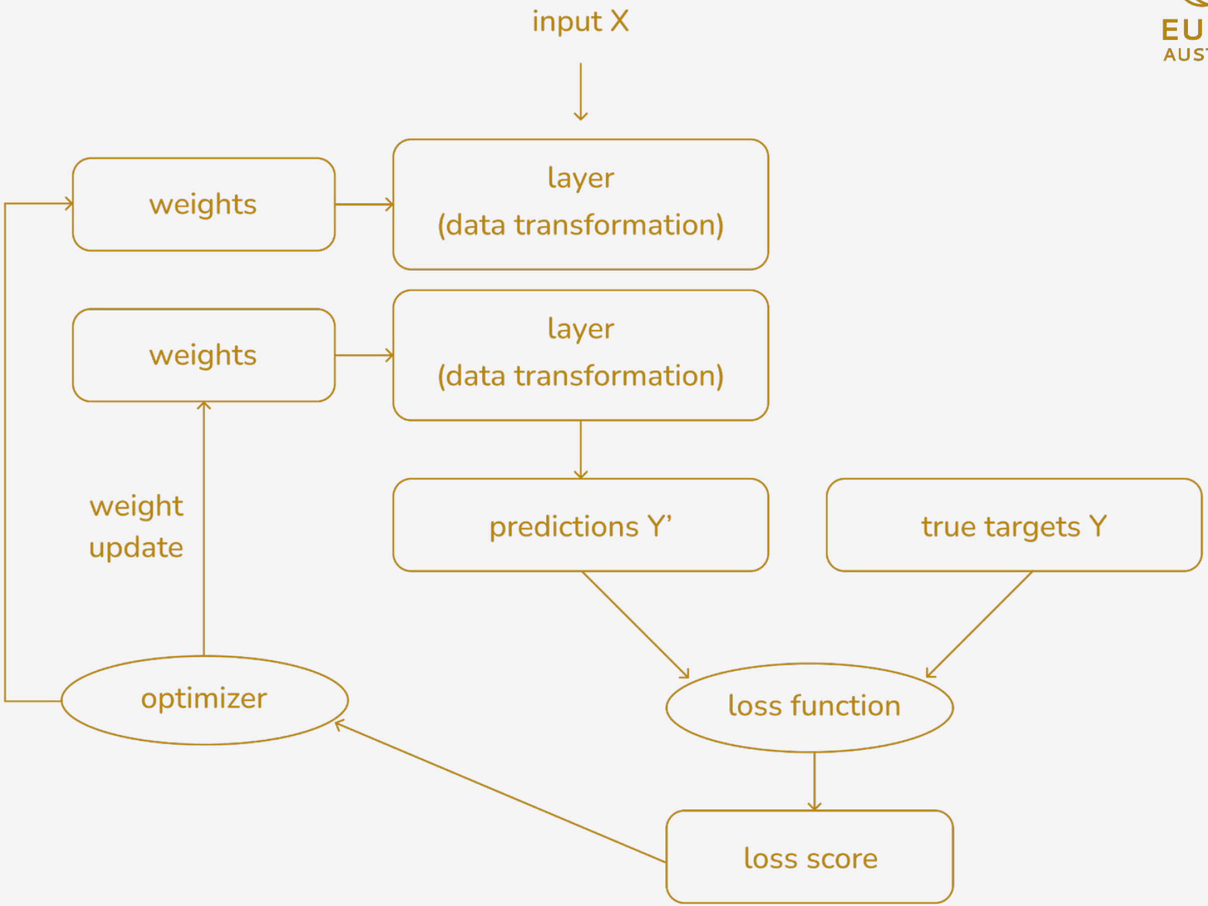
Training algorithm



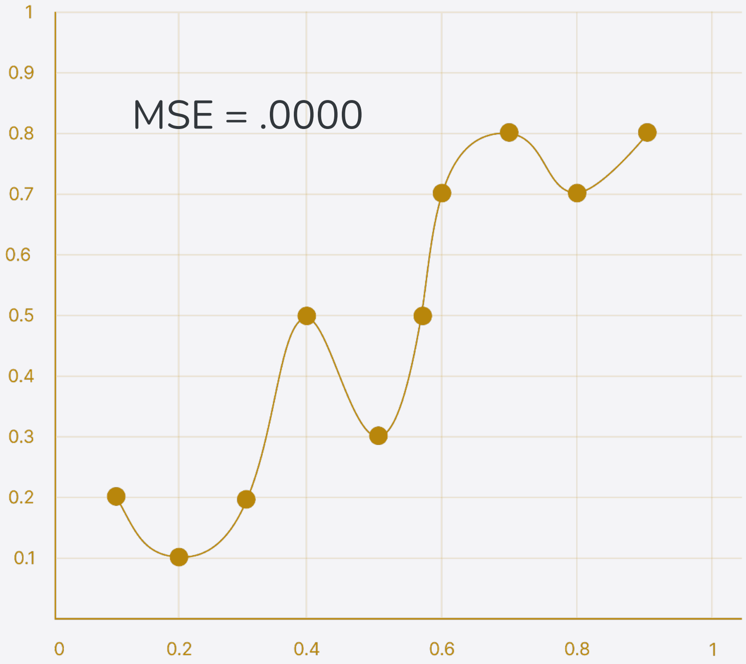
Training algorithm

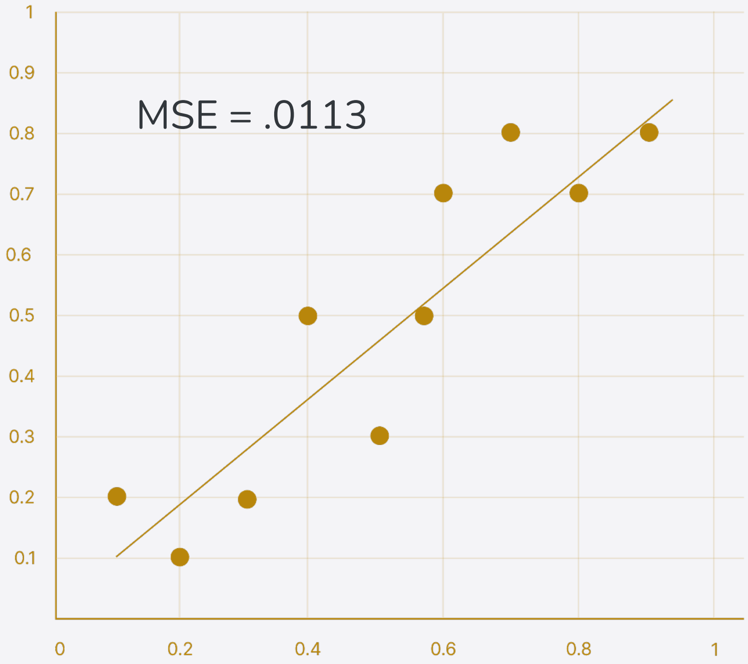


Training algorithm

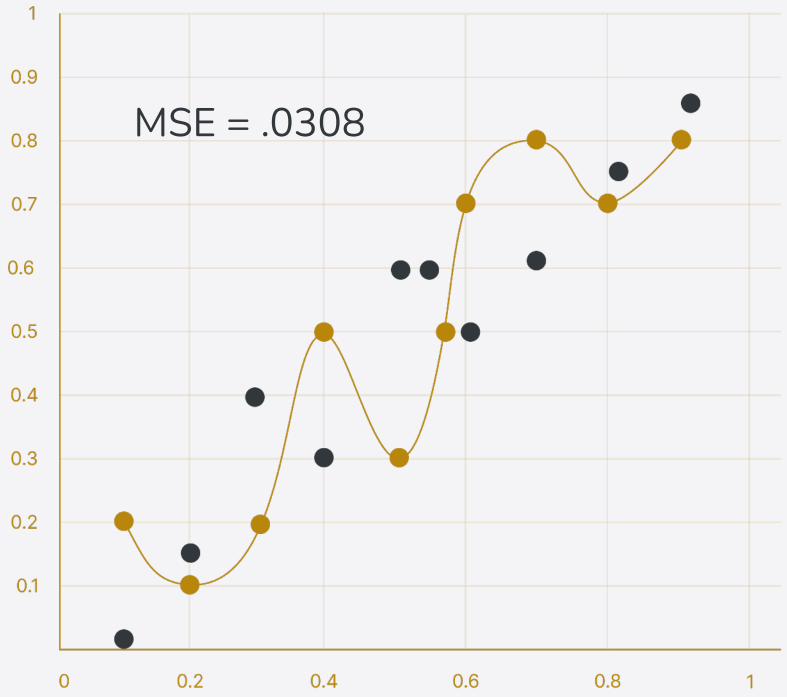


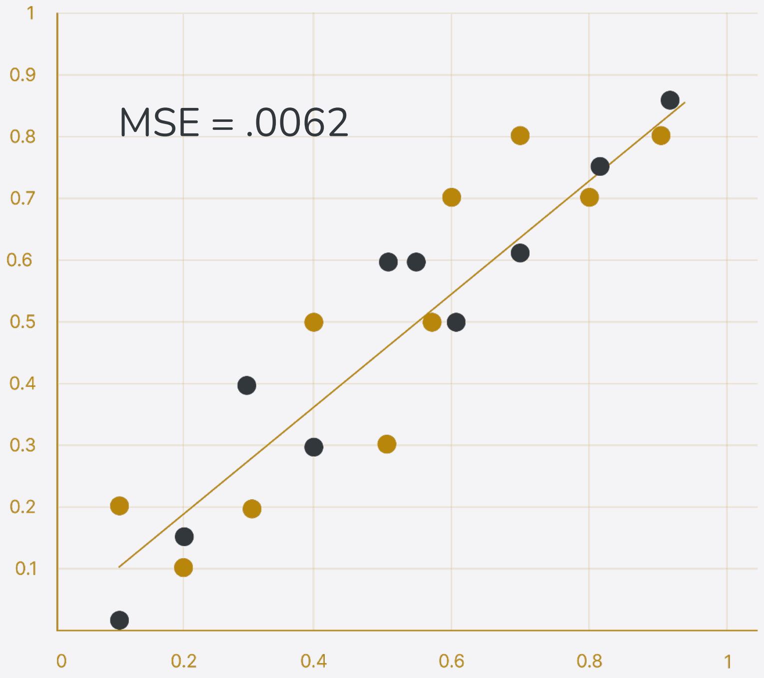
Which trendline is better?





Which trendline is better?





Training data

Core dataset for the model to learn on

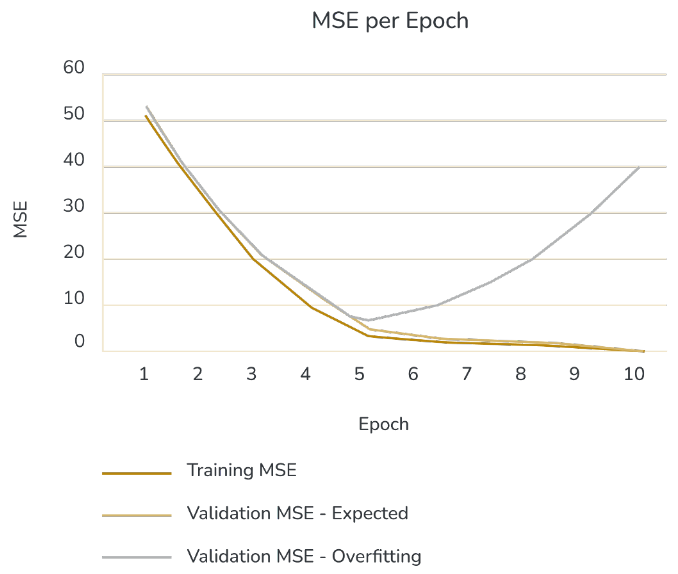
Validation data

New data for model to see if it truly understands

Test data

When model performs well on training data, but not the validation data (evidence of memorization)

Ideally, the accuracy and loss should be similar between both datasets



THANK YOU

EURO
AUSTRIA

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101101903. The JU receives support from the Digital Europe Programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia

STAY IN TOUCH

EuroCC Austria

@eurocc_austria

eurocc-austria.at