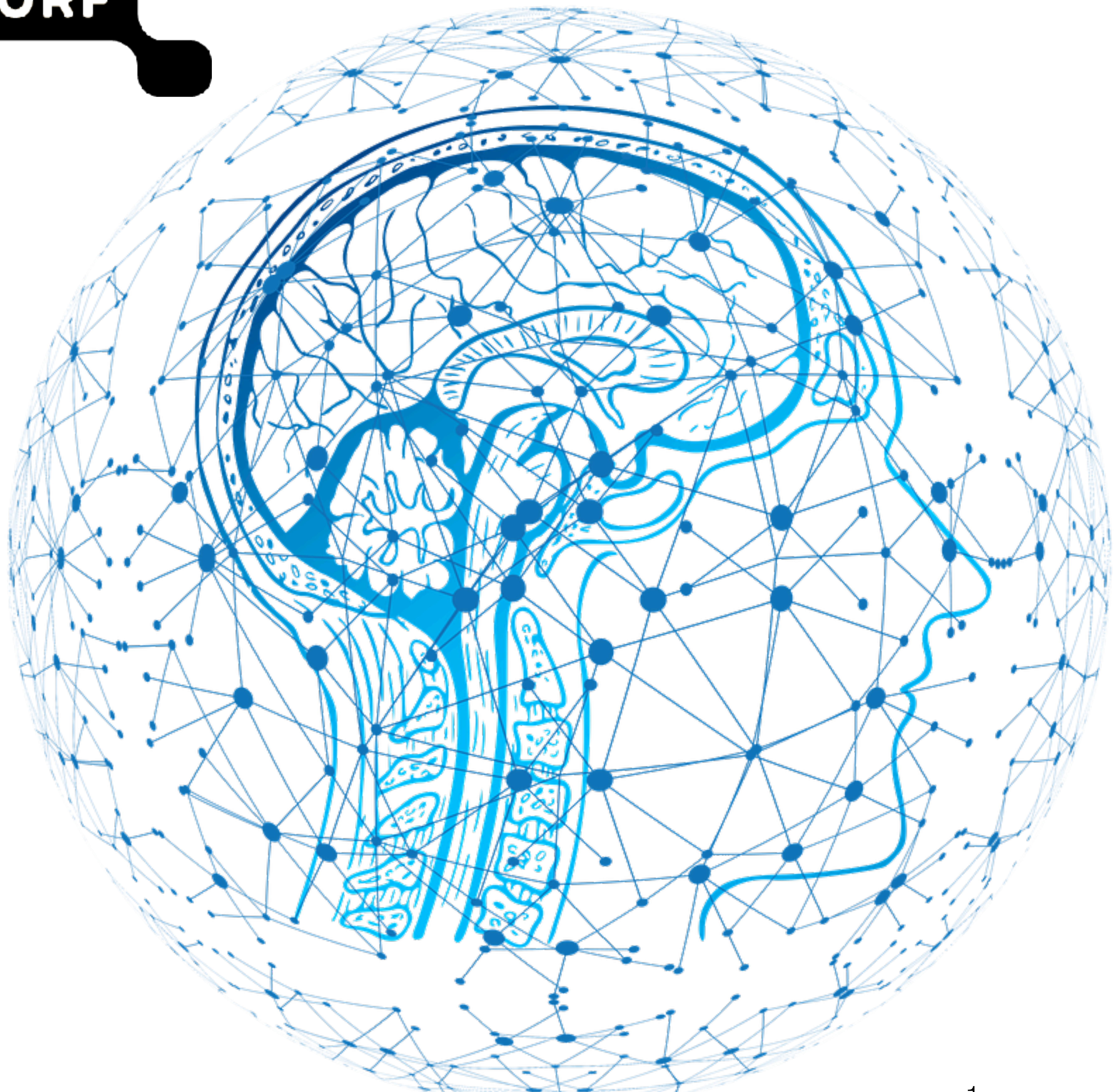


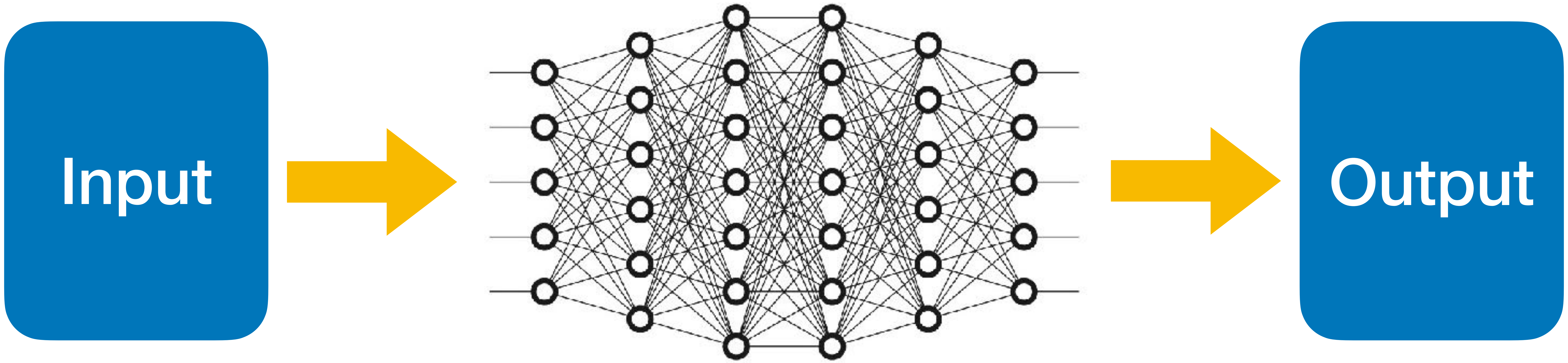
Introduction to Deep Learning

Multi-layer Perceptron

Maxwell Cai

maxwellcai.com

How does a neural network work?

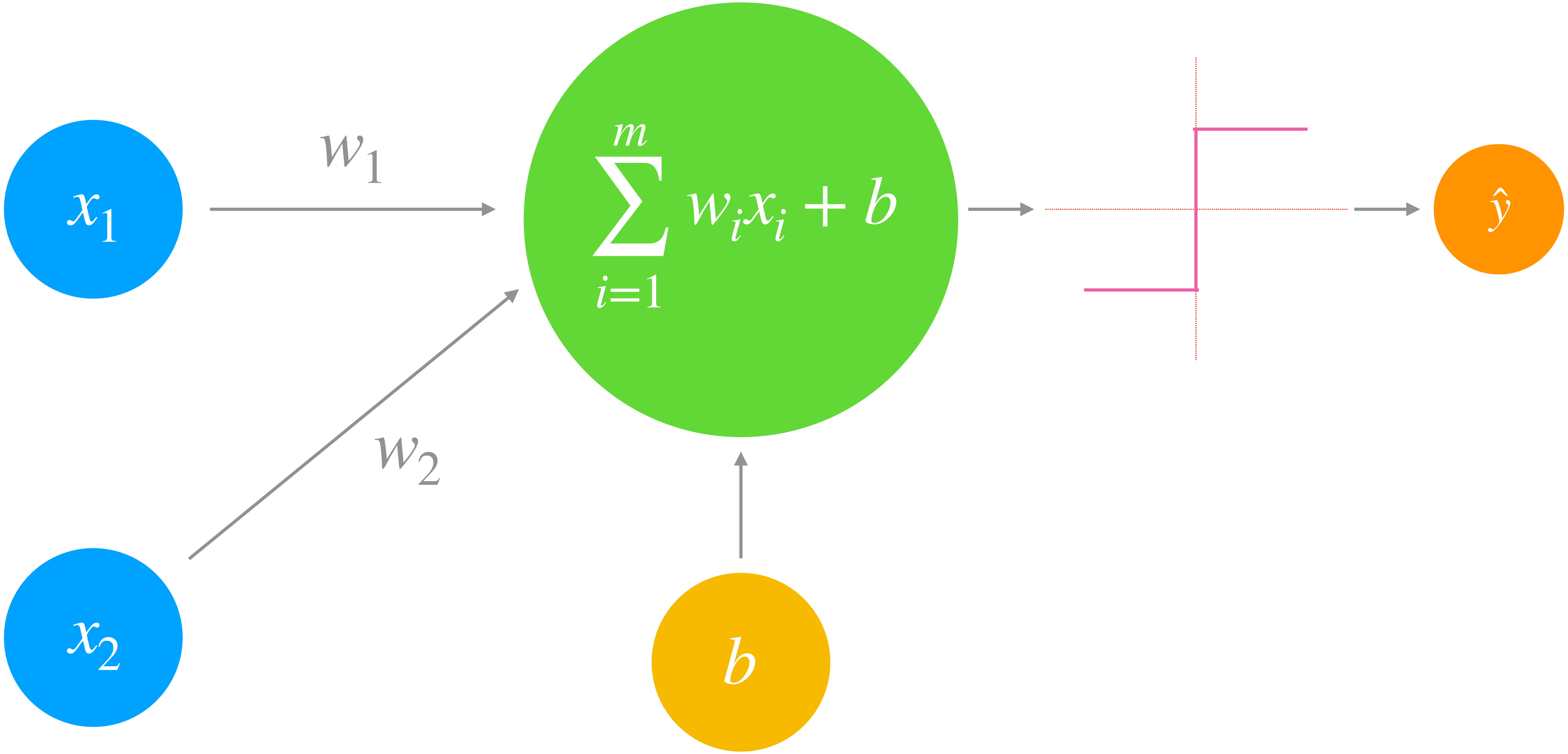


Input

Output

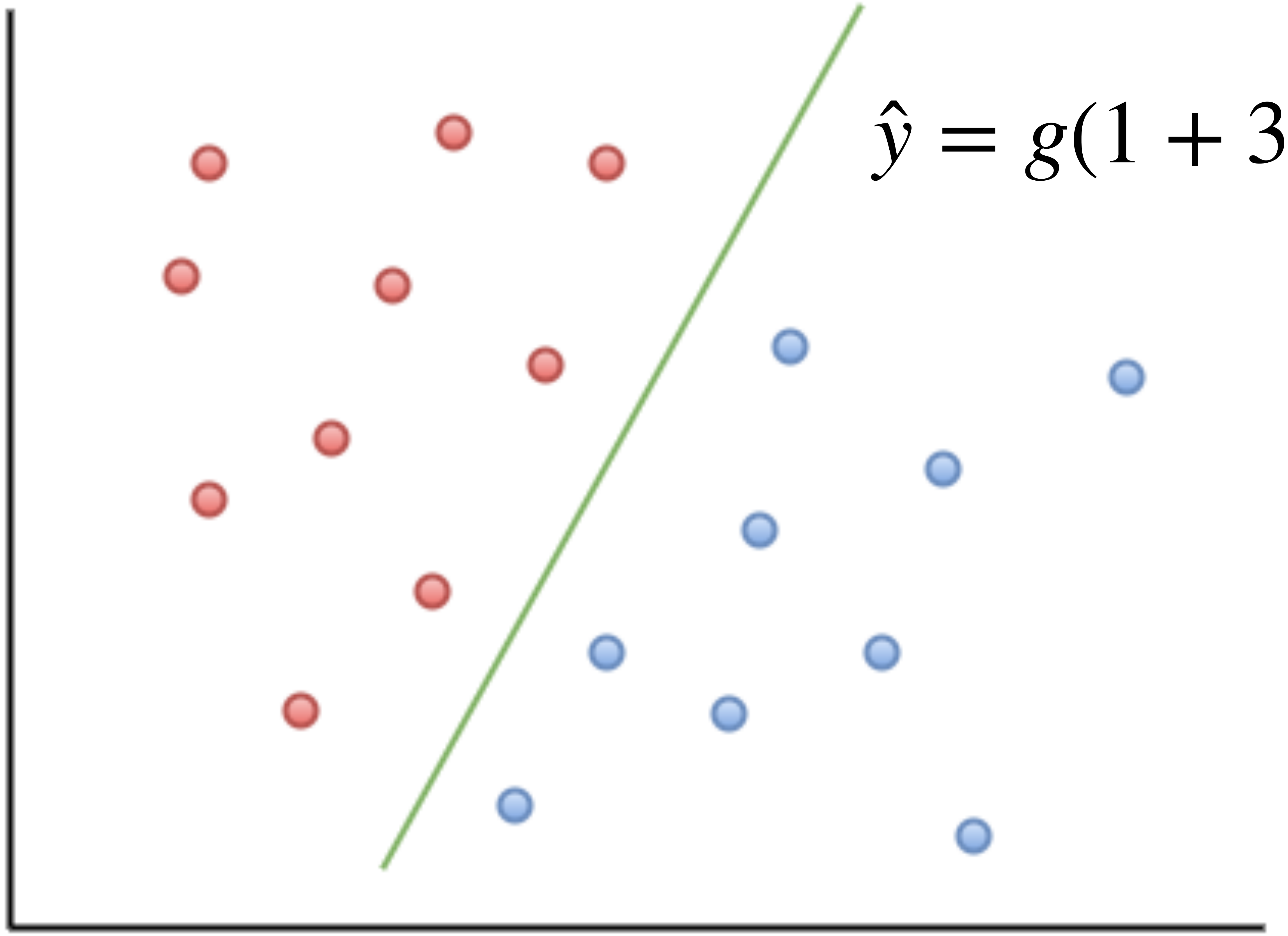
$$\hat{y} = f_{\mathrm{NN}}(x_1, x_2, \ldots, x_n)$$

Prediction

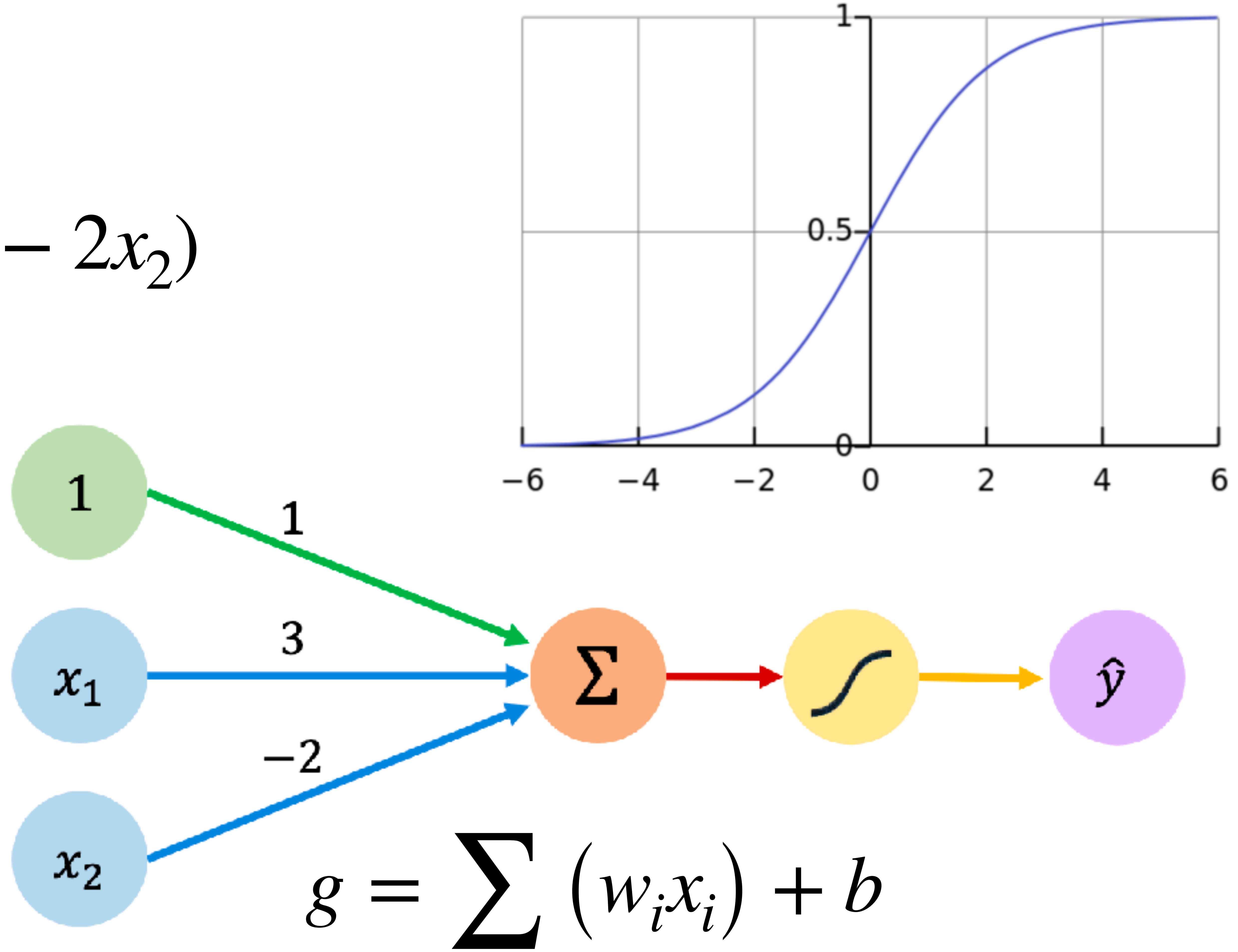


inputs weights summation + bias activation output

Neurons/Perceptrons & Activation Functions

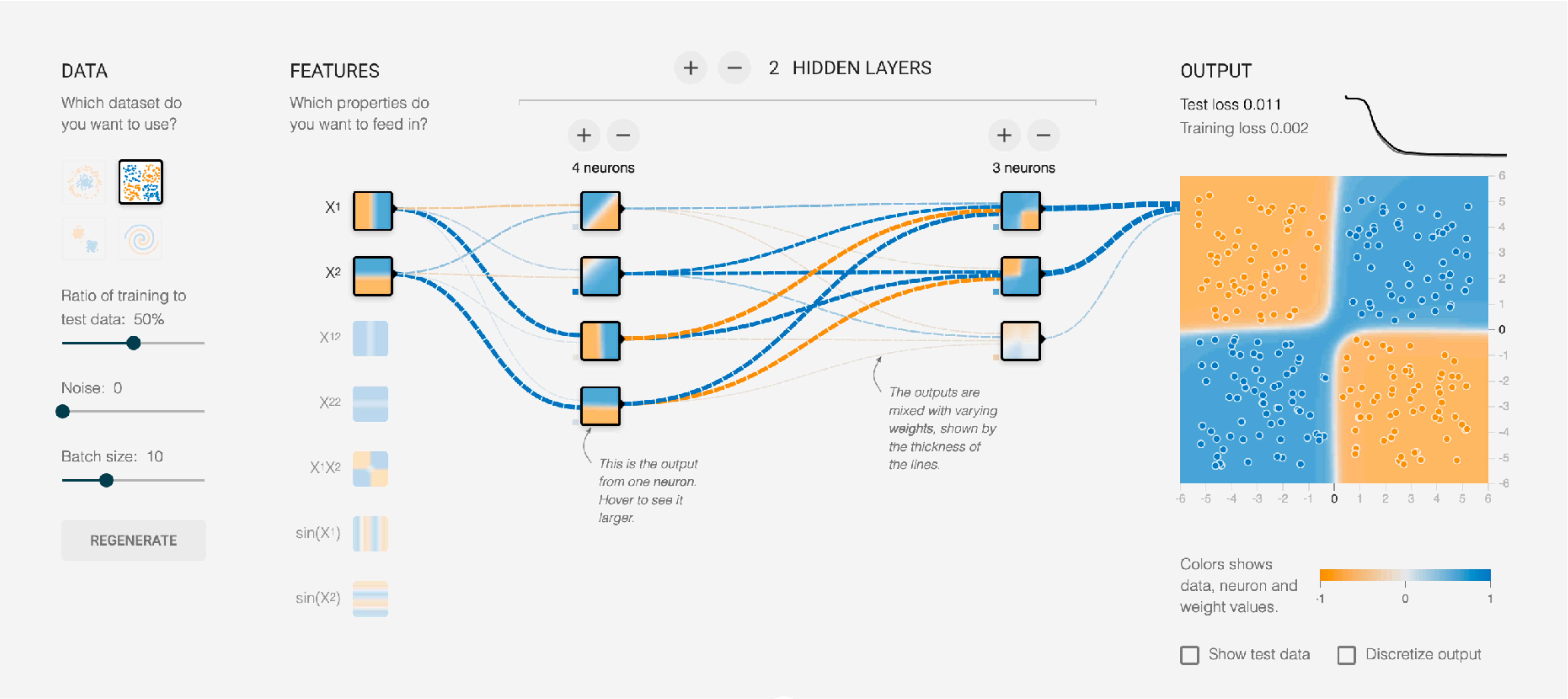


$$\hat{y} = g(1 + 3x_1 - 2x_2)$$



$$g = \sum \left(w_i x_i\right) + b$$

Live demo: Multi-layer Perceptrons



Which dataset do you want to use?

Which properties do you want to feed in?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

Test loss 0.011

Training loss 0.002

This is the output from one neuron. Hover to see it larger.

The outputs are mixed with varying weights, shown by the thickness of the lines.

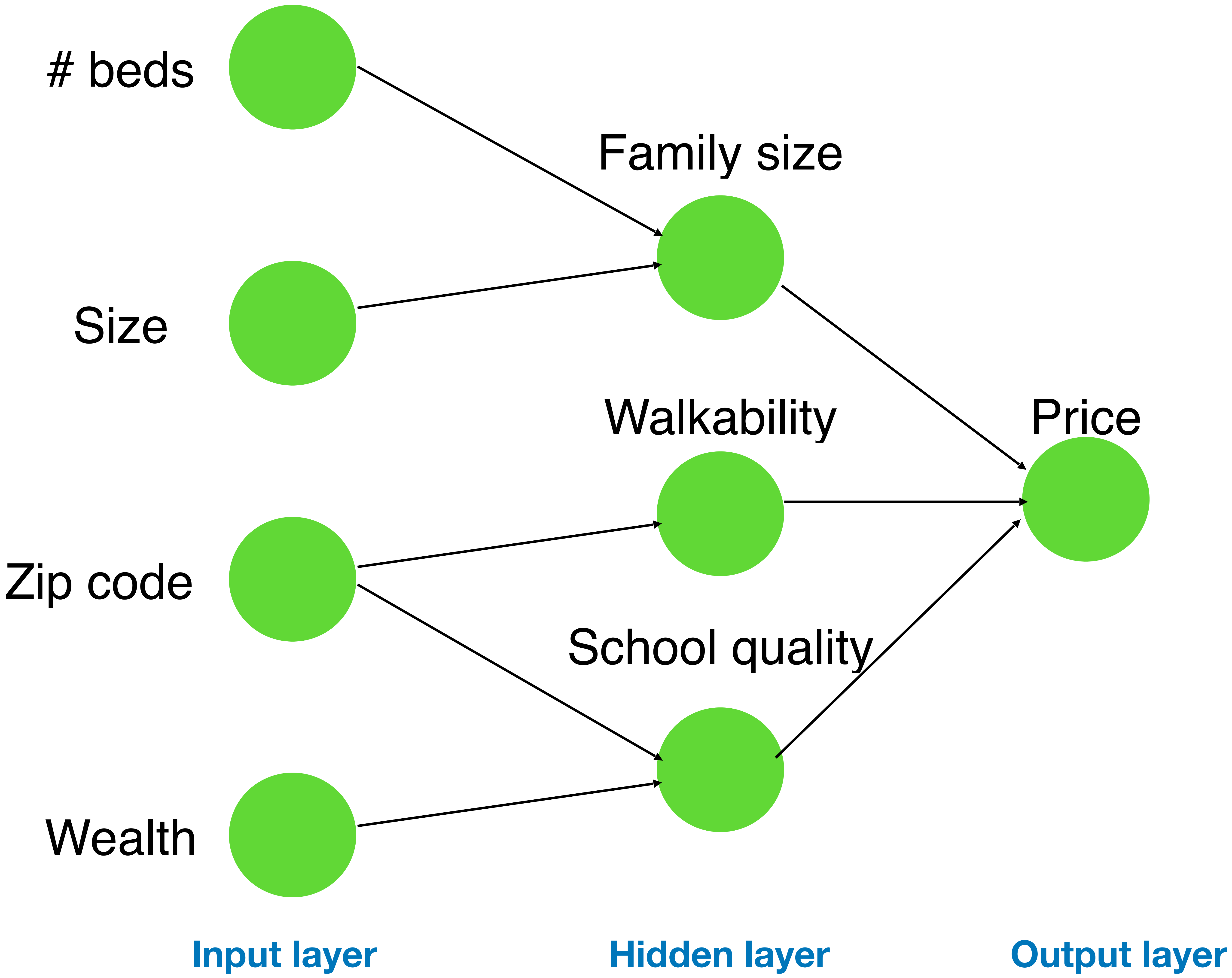
Colors shows data, neuron and weight values.

□ Show test data □ Discretize output

https://playground.tensorflow.org

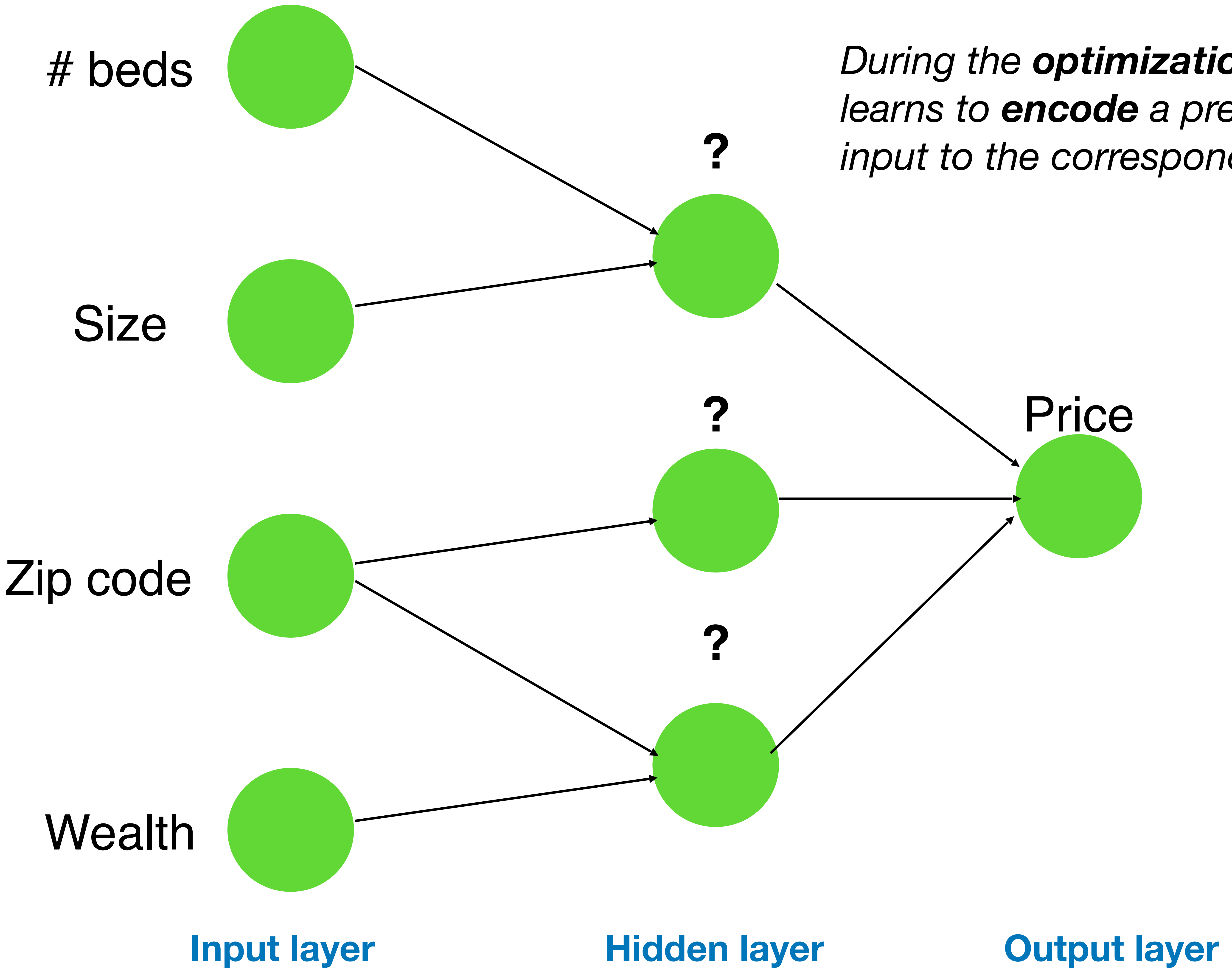
Why multiple layers?

Example: house price prediction model (designed by humans)



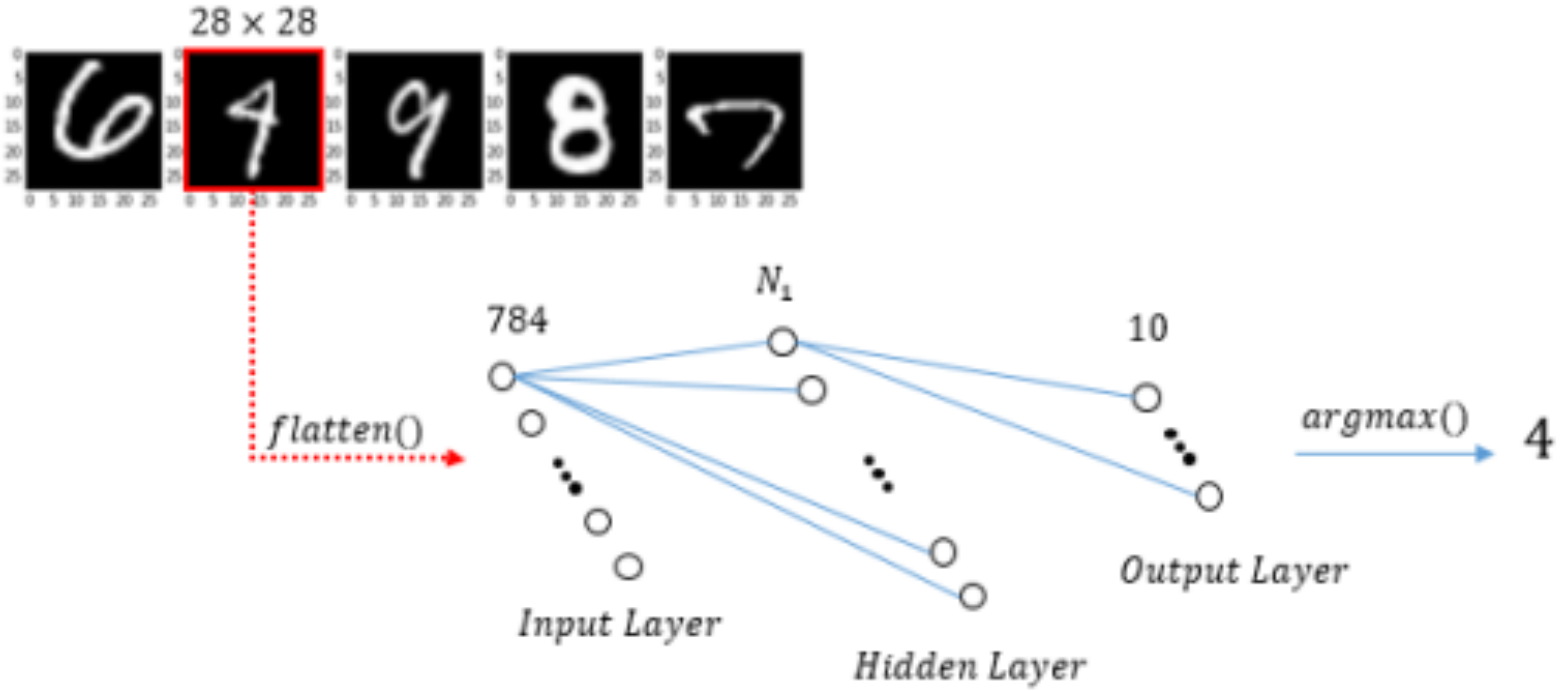
Why multiple layers?

Example: house price prediction model (designed by machines)



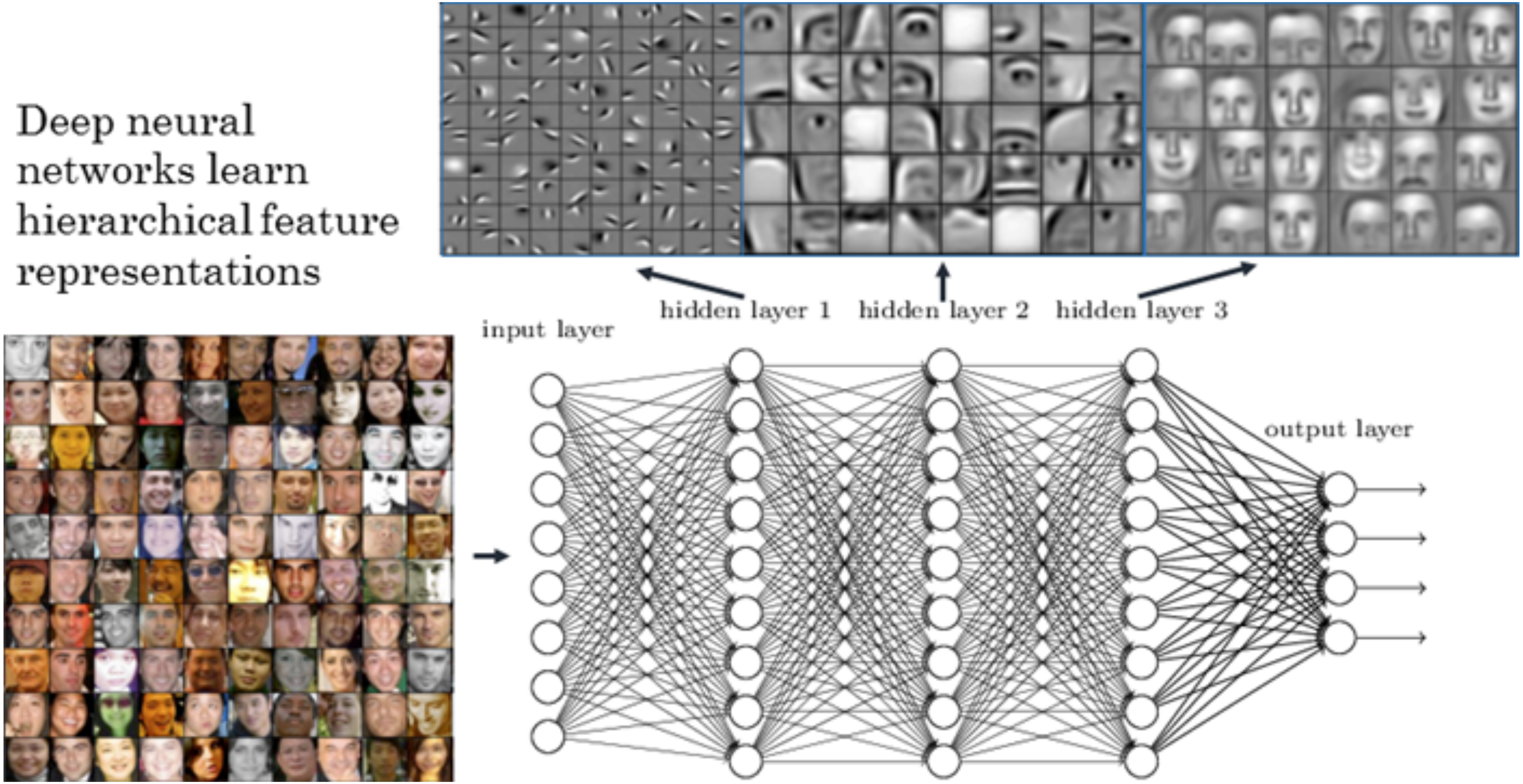
*During the **optimization** process, the machine learns to **encode** a presentation that **maps** the input to the corresponding output.*

Multi-Layer Perception



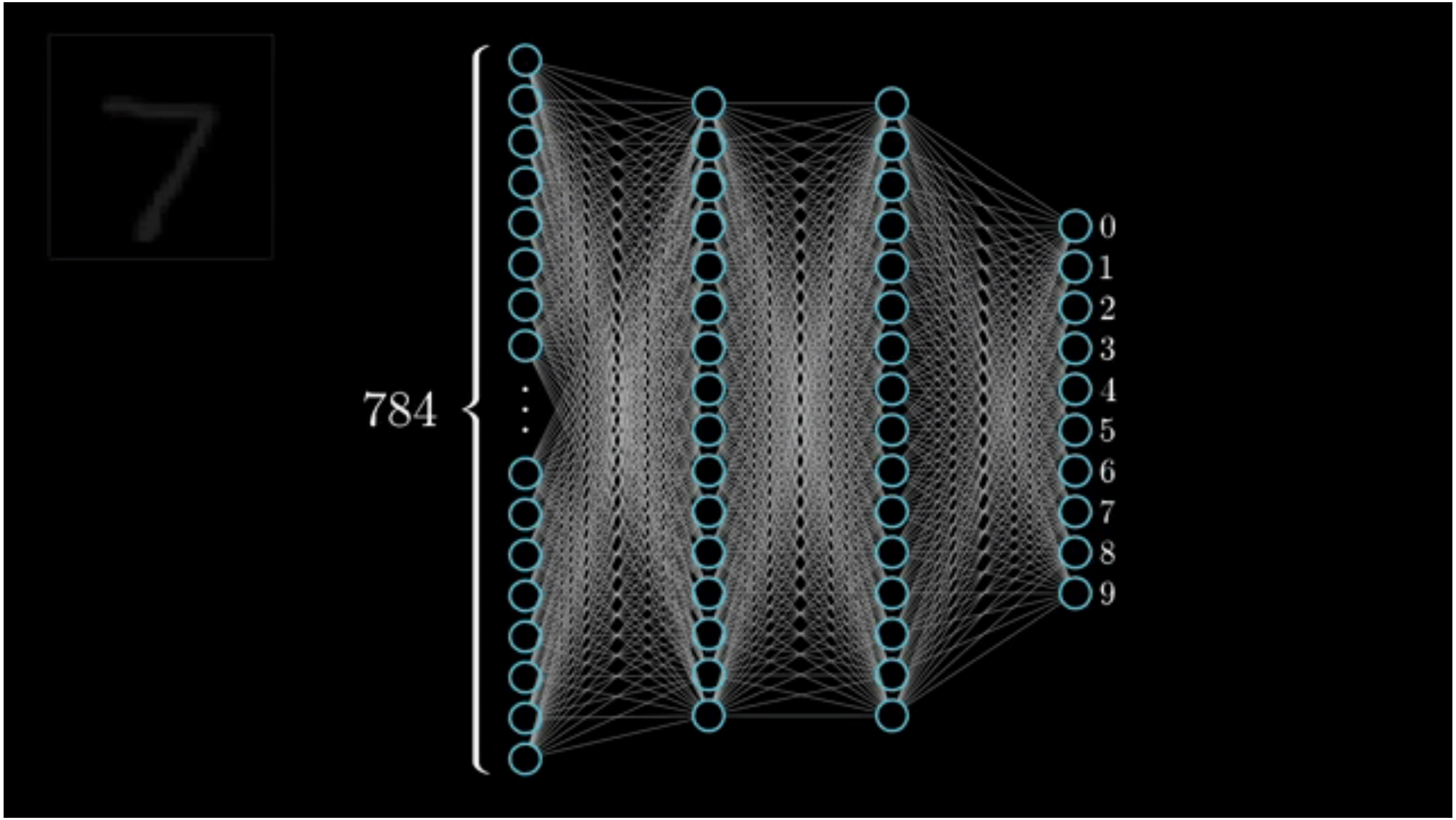
MNIST dataset
(**M**odified **N**ational **I**nstitute for **S**tandards and **T**echnology)

A DNN encodes the representation hierarchically



Deep neural networks learn hierarchical feature representations

Prediction: forward propagation

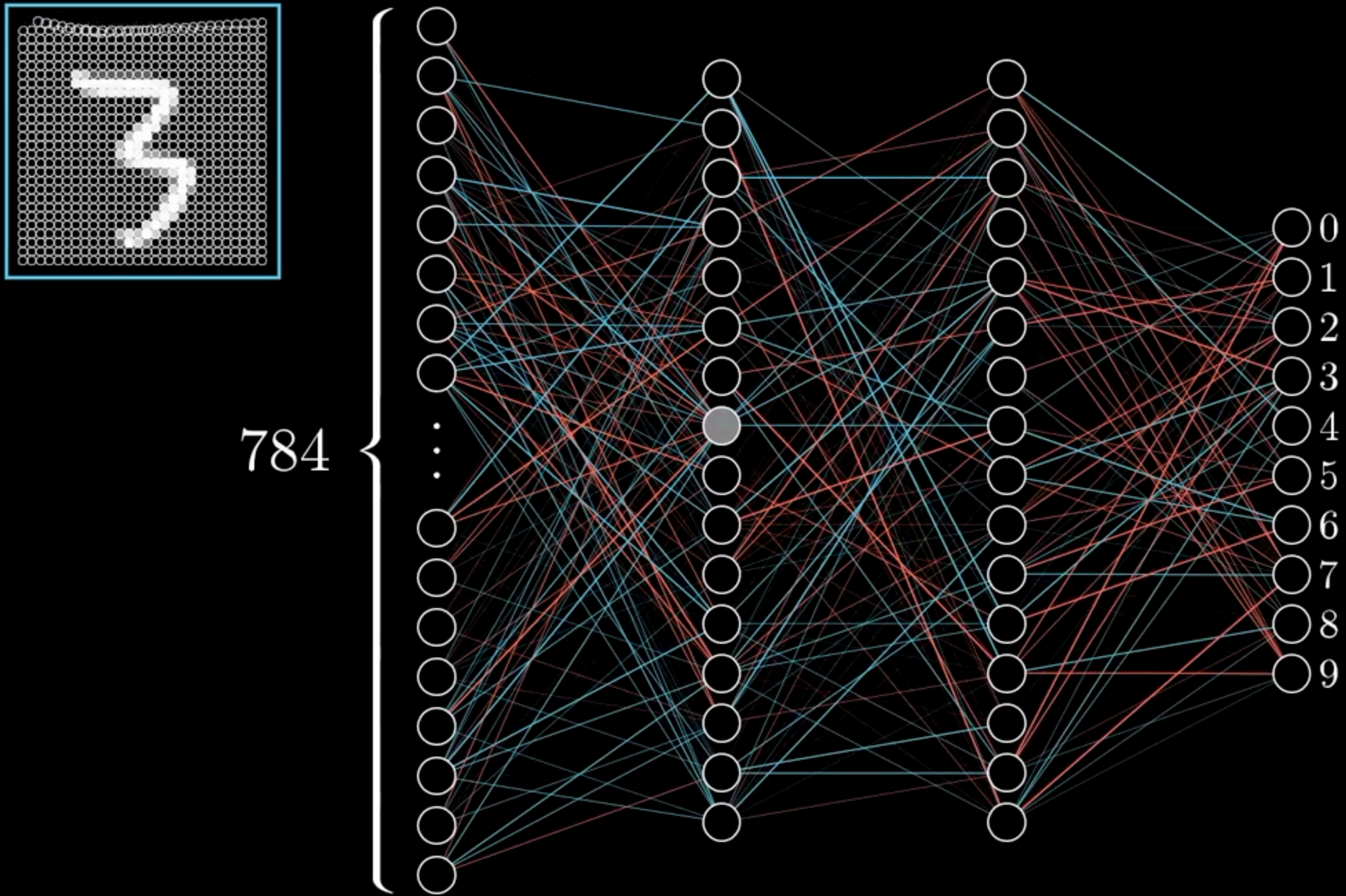


Source: 3blue1brown

What's the "cost" of this difference?
784
0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8
9
Utter trash

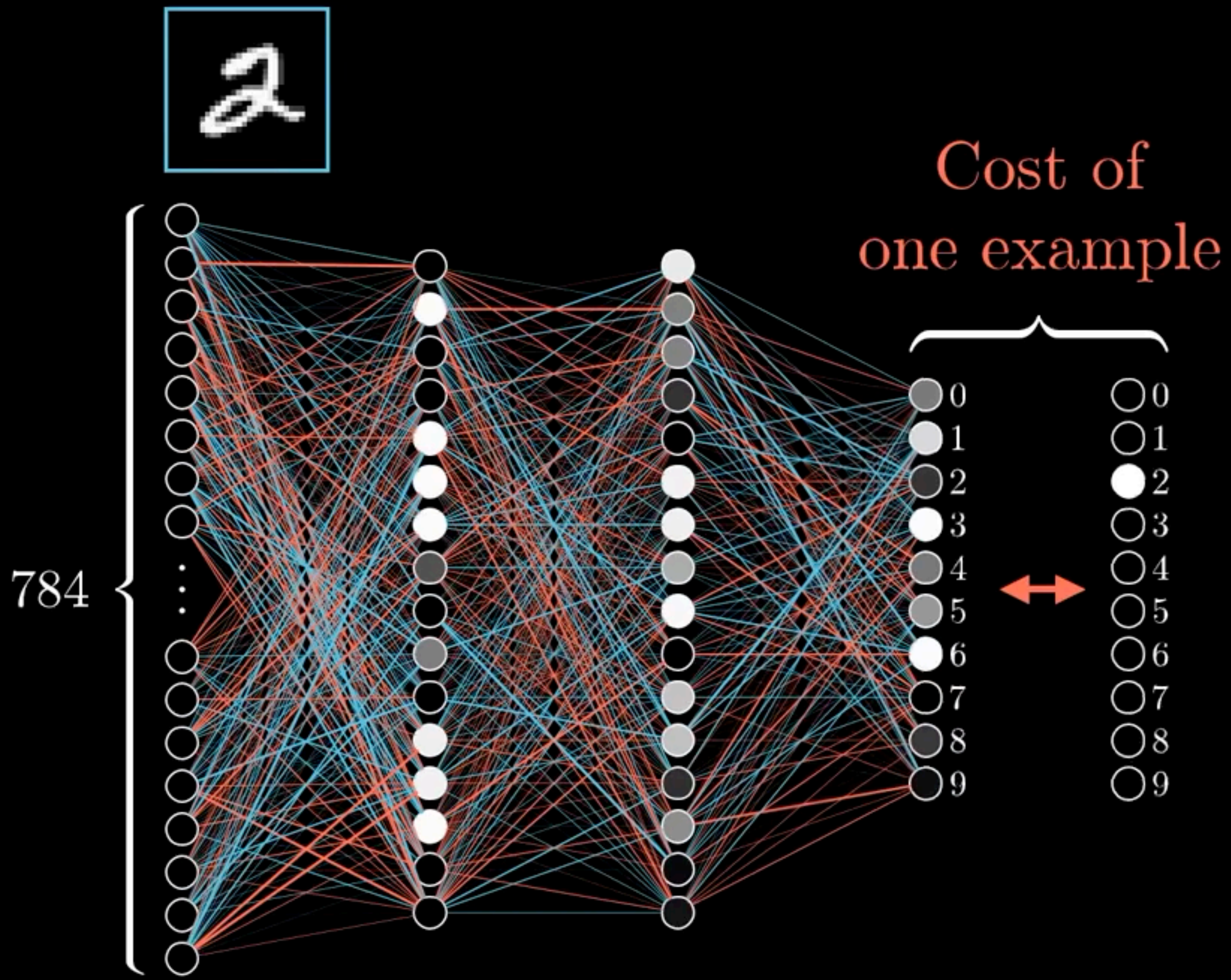
Training: backward propagation

Animation: 3blue1brown



Training: backward propagation

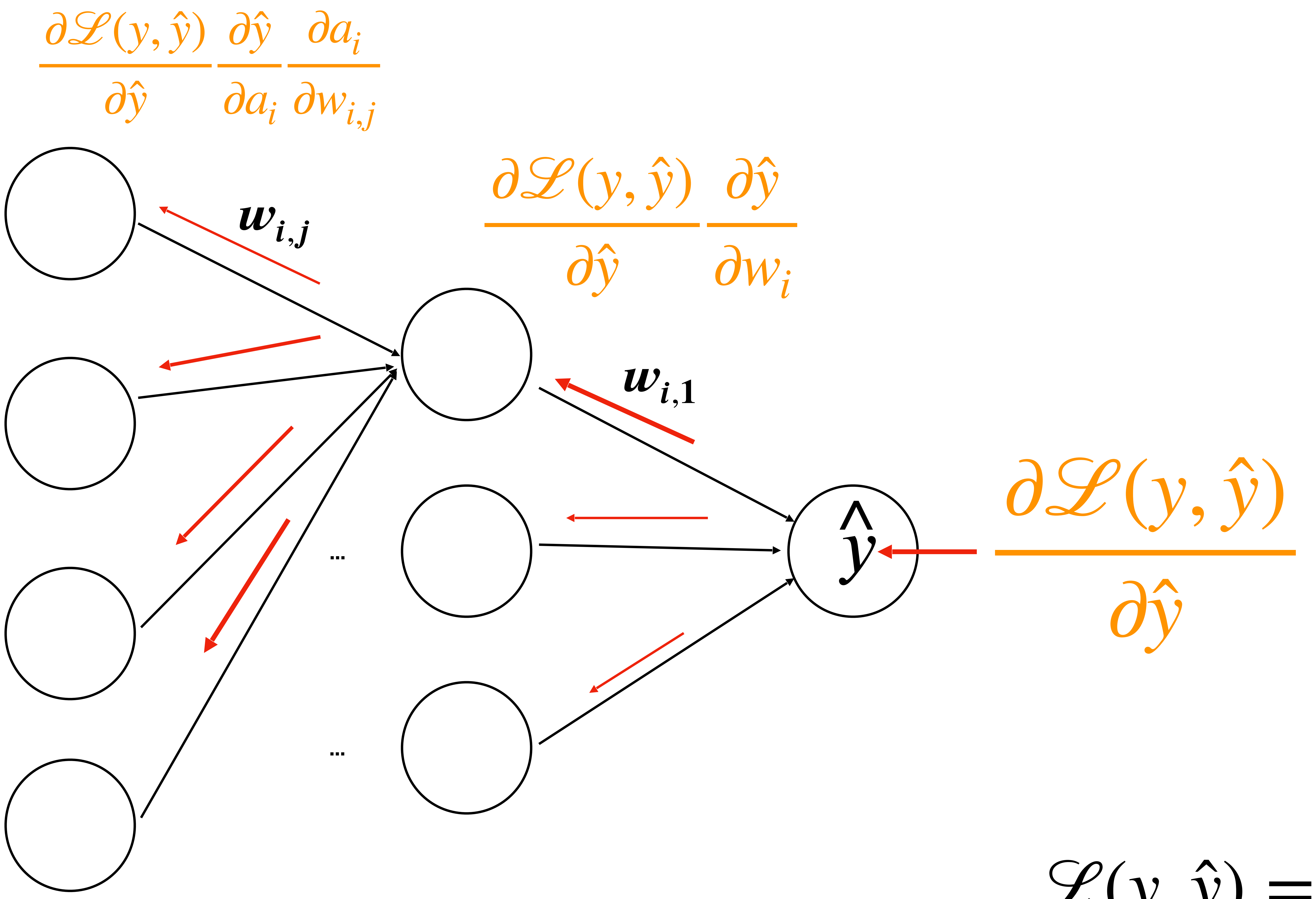
Animation: 3blue1brown



Training: backward propagation

Animation: 3blue1brown

Training: backward propagation



$\frac{\partial \mathscr{L}(y,\hat{y})}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_i} \frac{\partial a_i}{\partial w_{i,j}}$
$\boldsymbol{w}_{i,j}$
$\frac{\partial \mathscr{L}(y,\hat{y})}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w_i}$
$\boldsymbol{w}_{i,1}$
…
…
$\hat{y}$
$\frac{\partial \mathscr{L}(y,\hat{y})}{\partial \hat{y}}$

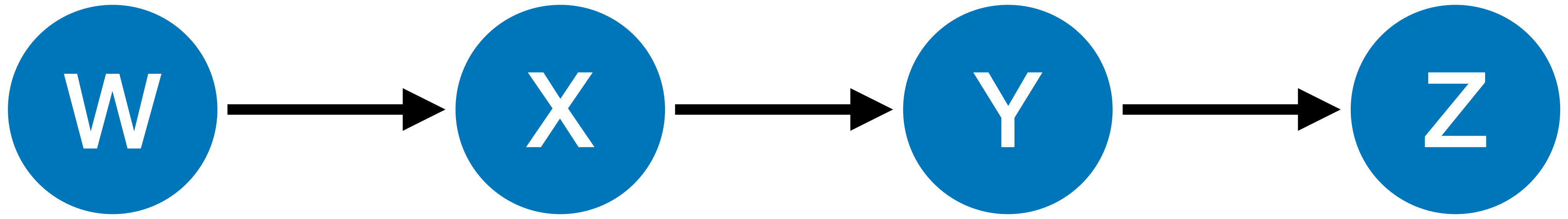
- In the context of DL we need to compute the gradient for each layer.
- We do this by applying the **chain rule** of derivatives.
- This algorithm is known as **backpropagation**.

$$\mathscr{L}(y,\hat{y}) = L(\mathbf{W}, b) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2$$

Neural Network: the deeper, the better?

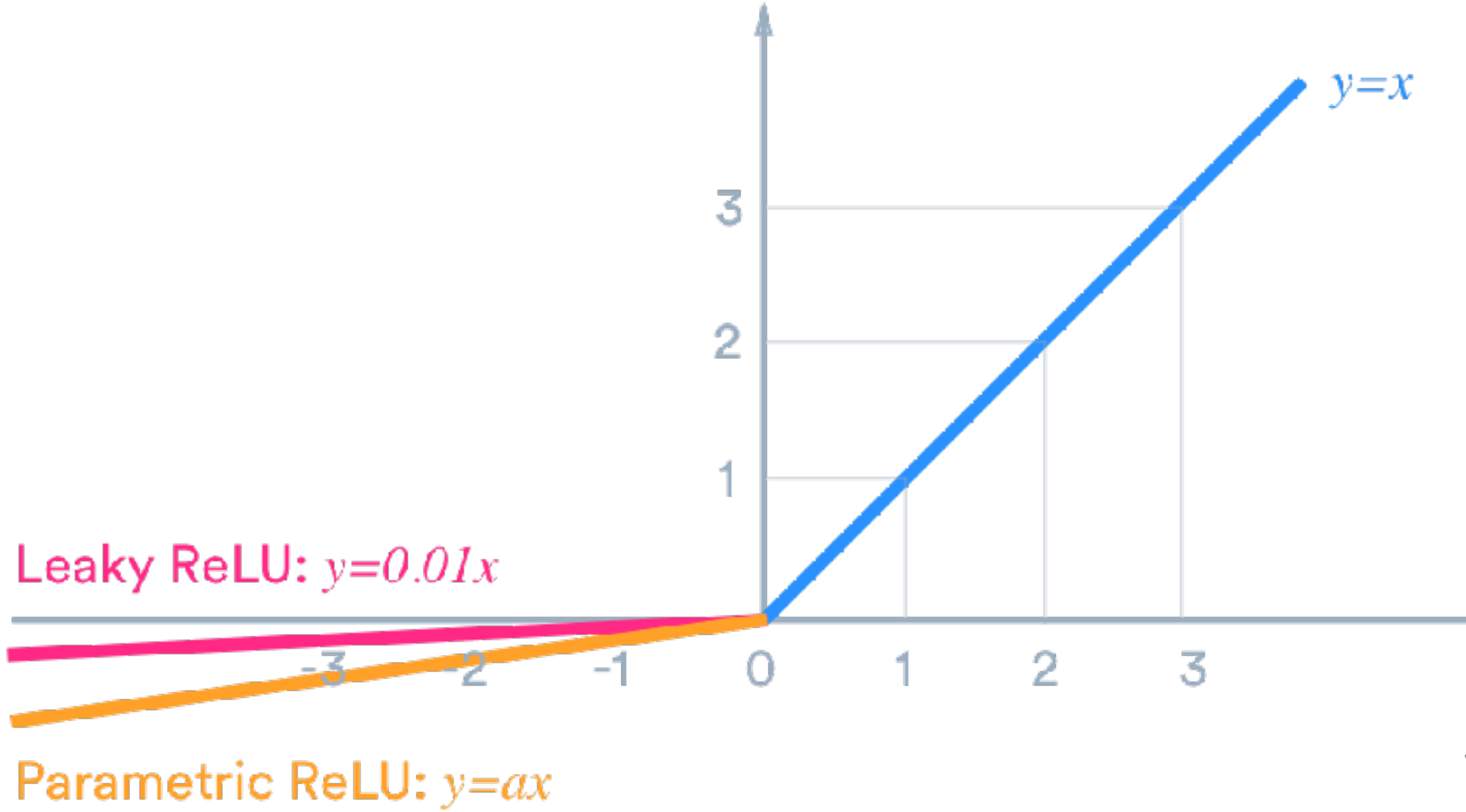
Not really.

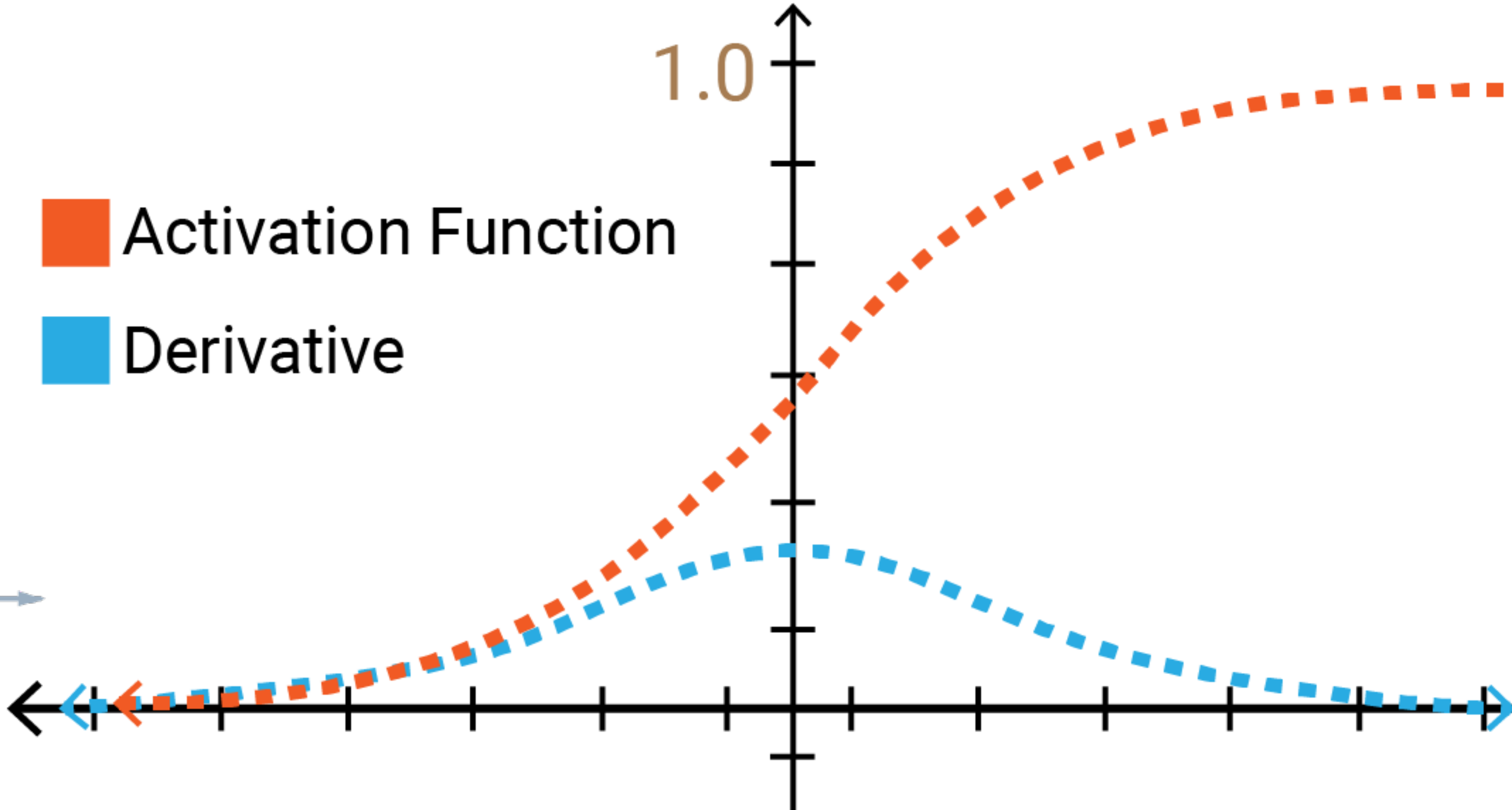
The vanishing gradient problem



Chain rule

$$\frac{\partial z}{\partial w} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\frac{\partial x}{\partial w}$$

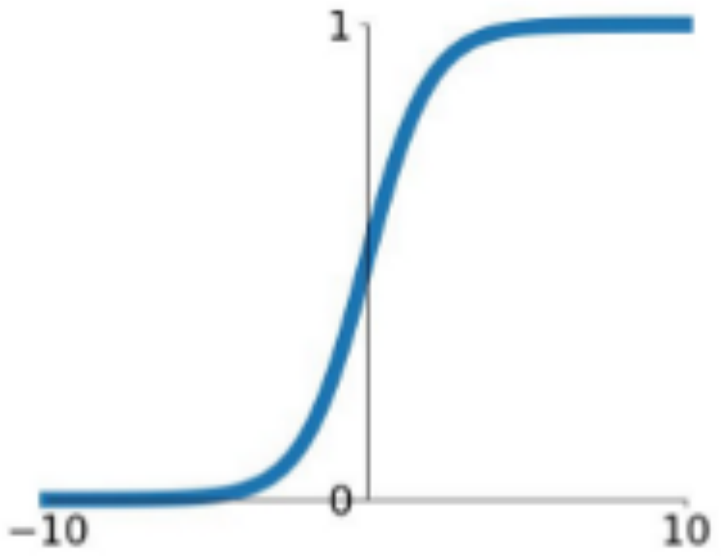




Activation Functions

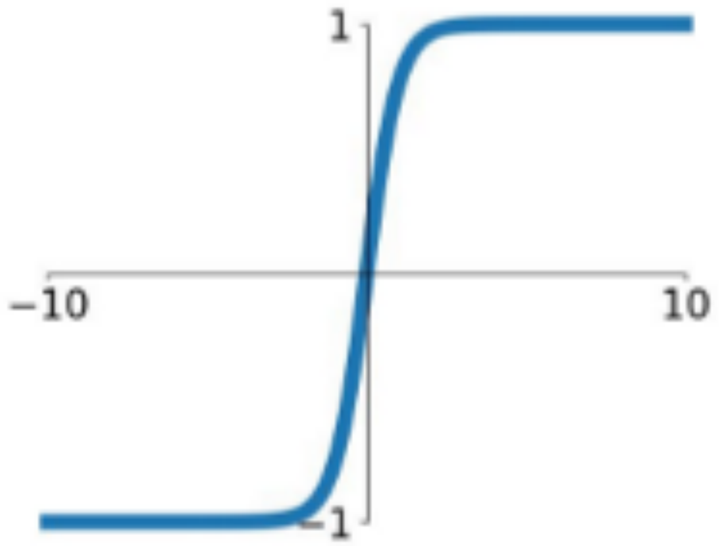
Sigmoid

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



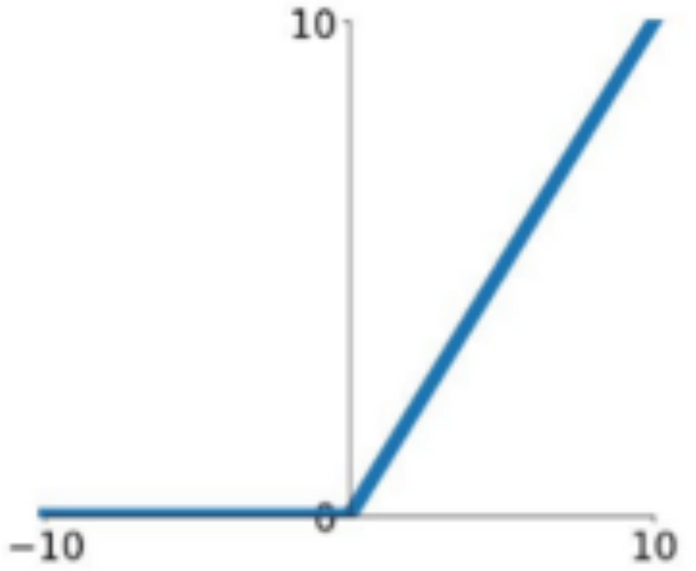
tanh

$\tanh(x)$



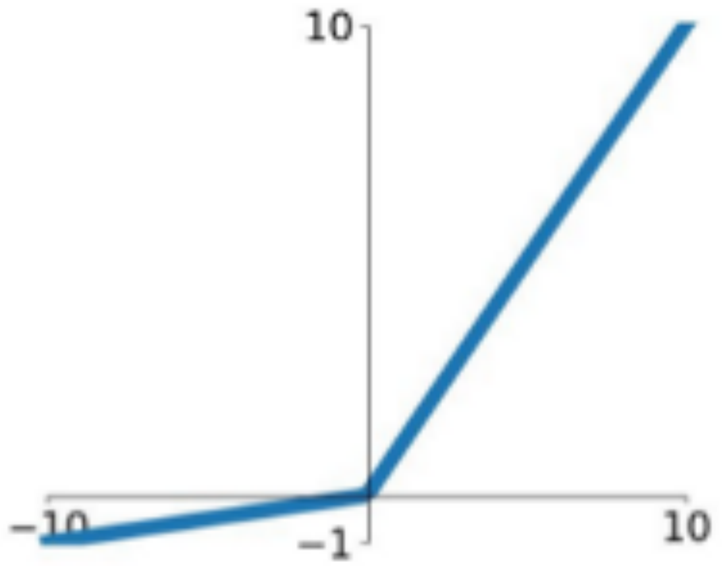
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

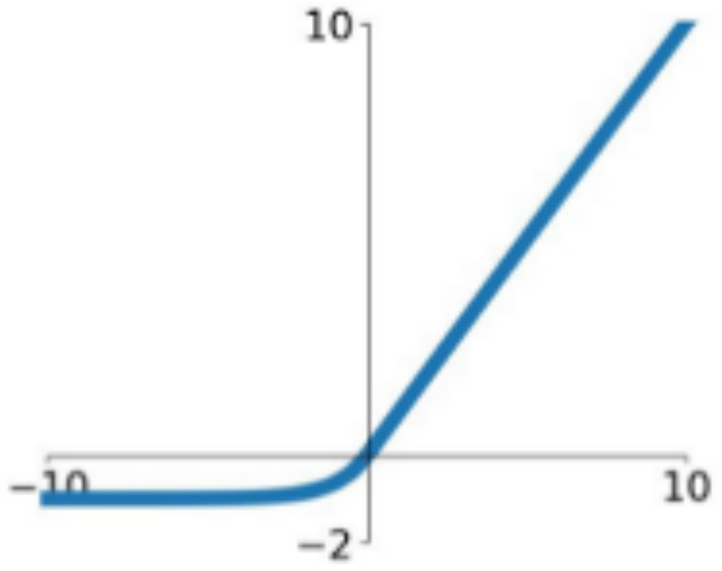


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

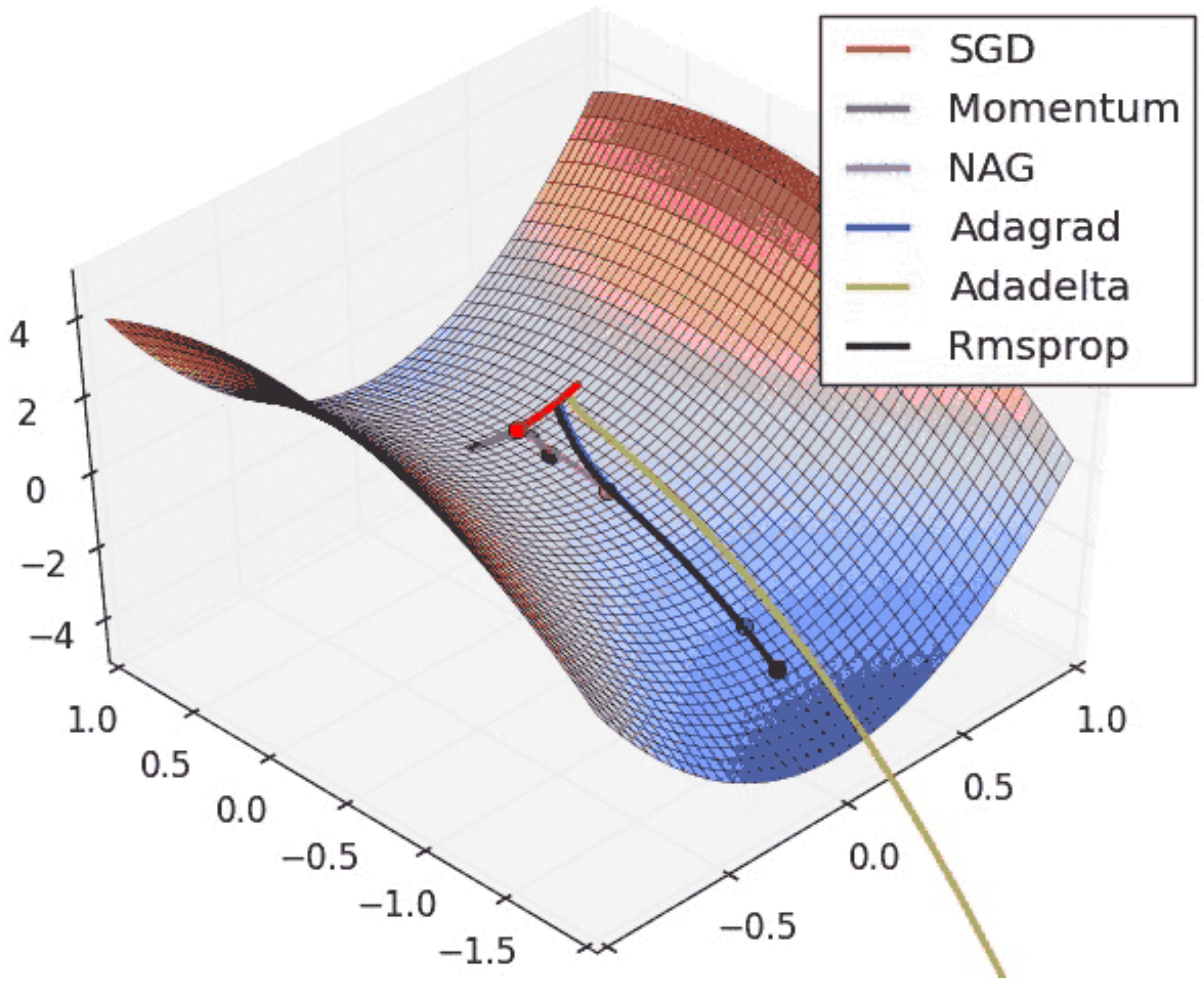
ELU

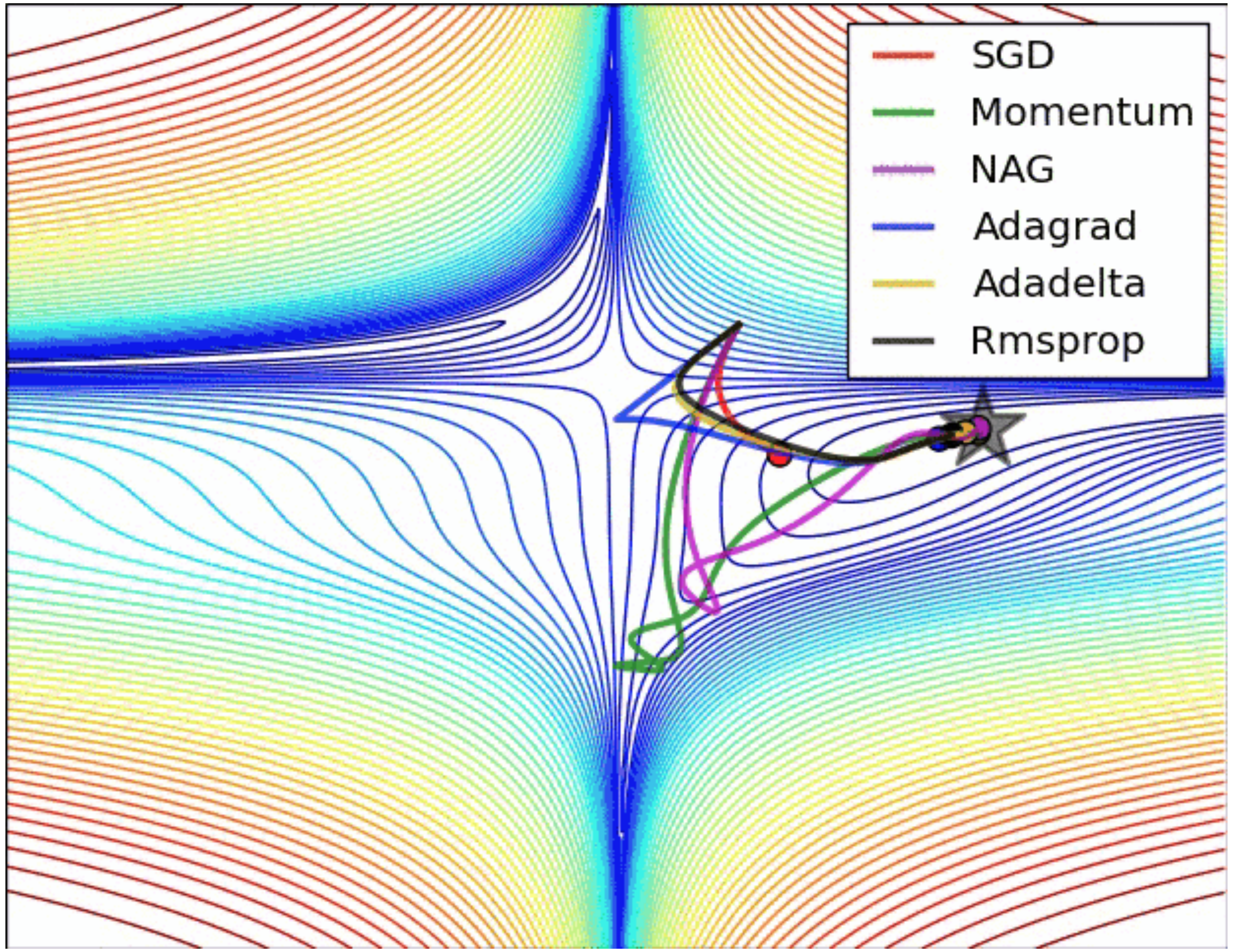
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



But one can design their own activation functions!

Optimizers





Source: medium.com/analytics-vidhya

Common practice for loss functions

Regression

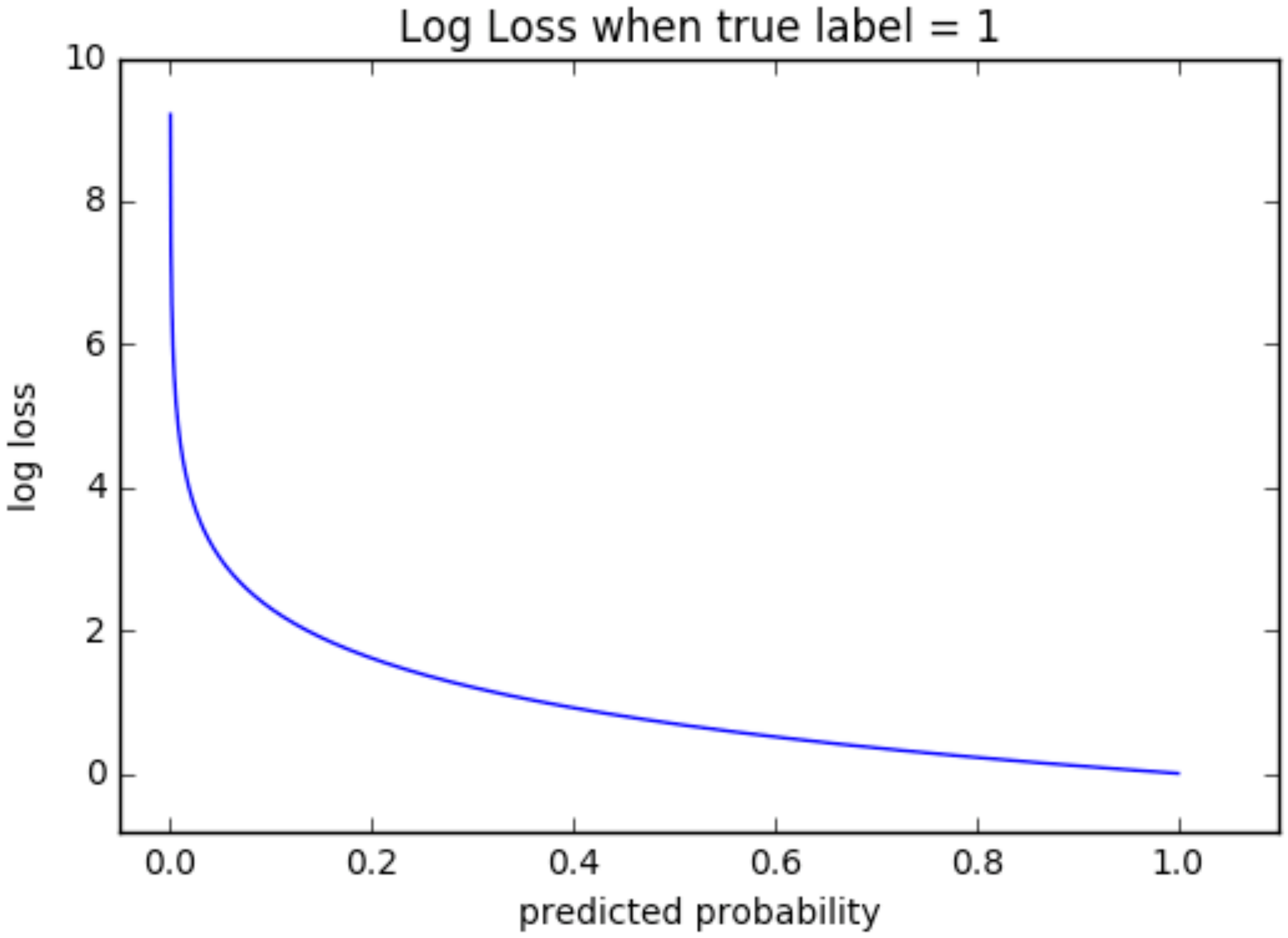
- Mean squared error
- Mean squared logarithmic error
- Mean absolute error

Binary Classification

- Binary cross-entropy
- Hinge loss
- Squared hinge loss

Multi-Class Classification

- Multi-class cross-entropy
- Sparse multi-class cross-entropy
- Kullback-Leibler divergence



Cross-entropy loss outputs a log probability

DL frameworks

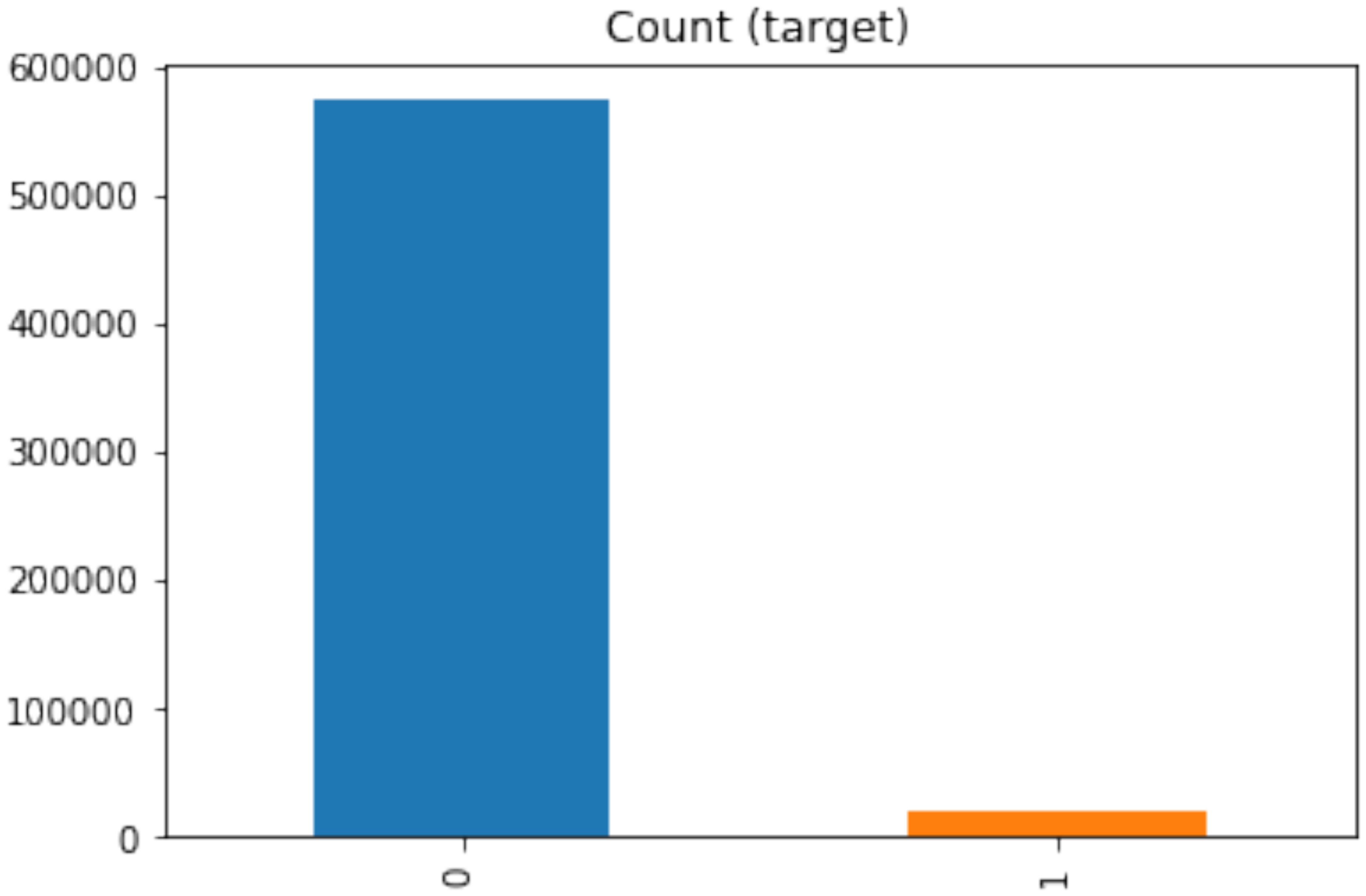
In DL, you need to

- Define neurons and layers
- Define loss function
- Calculate losses
- Calculate gradient (multivariate calculus)
- Backward propagation
- Update weights

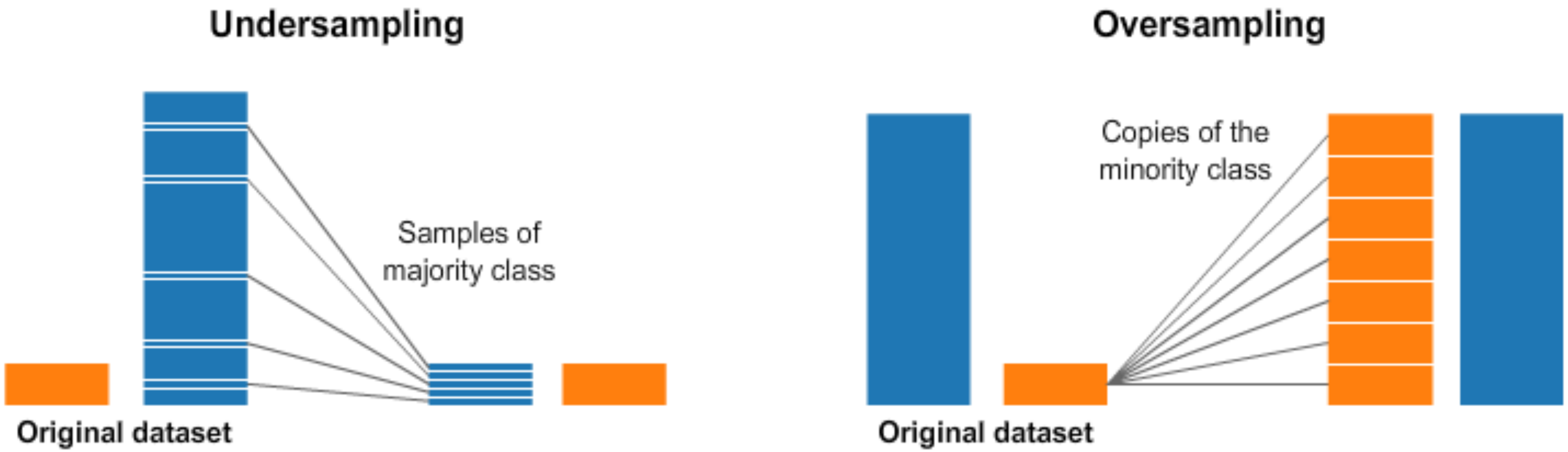
- Many frameworks exist; **TensorFlow**, **CNTK**, **Torch**, **Keras**, **Theano**, **Caffe**, ...
- We will use **TensorFlow/Keras**
- Keras used to call TensorFlow as a *backend*, but is now fully integrated in TensorFlow.

Model Evaluation

Balanced/Imbalanced training set



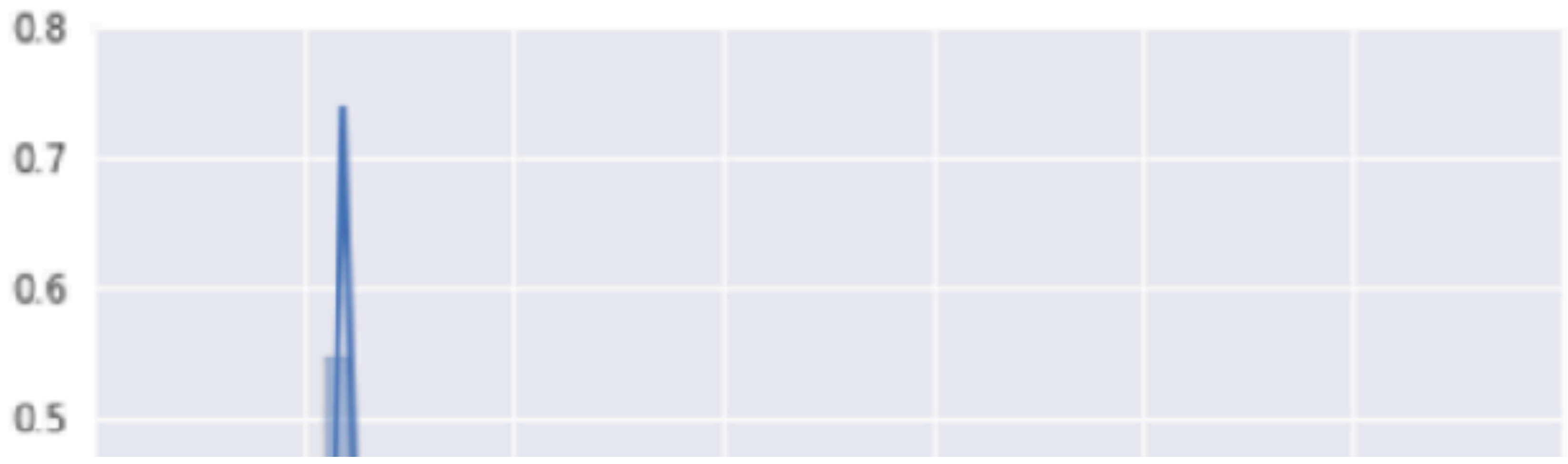
Pay attention to your data: they may fool your model.



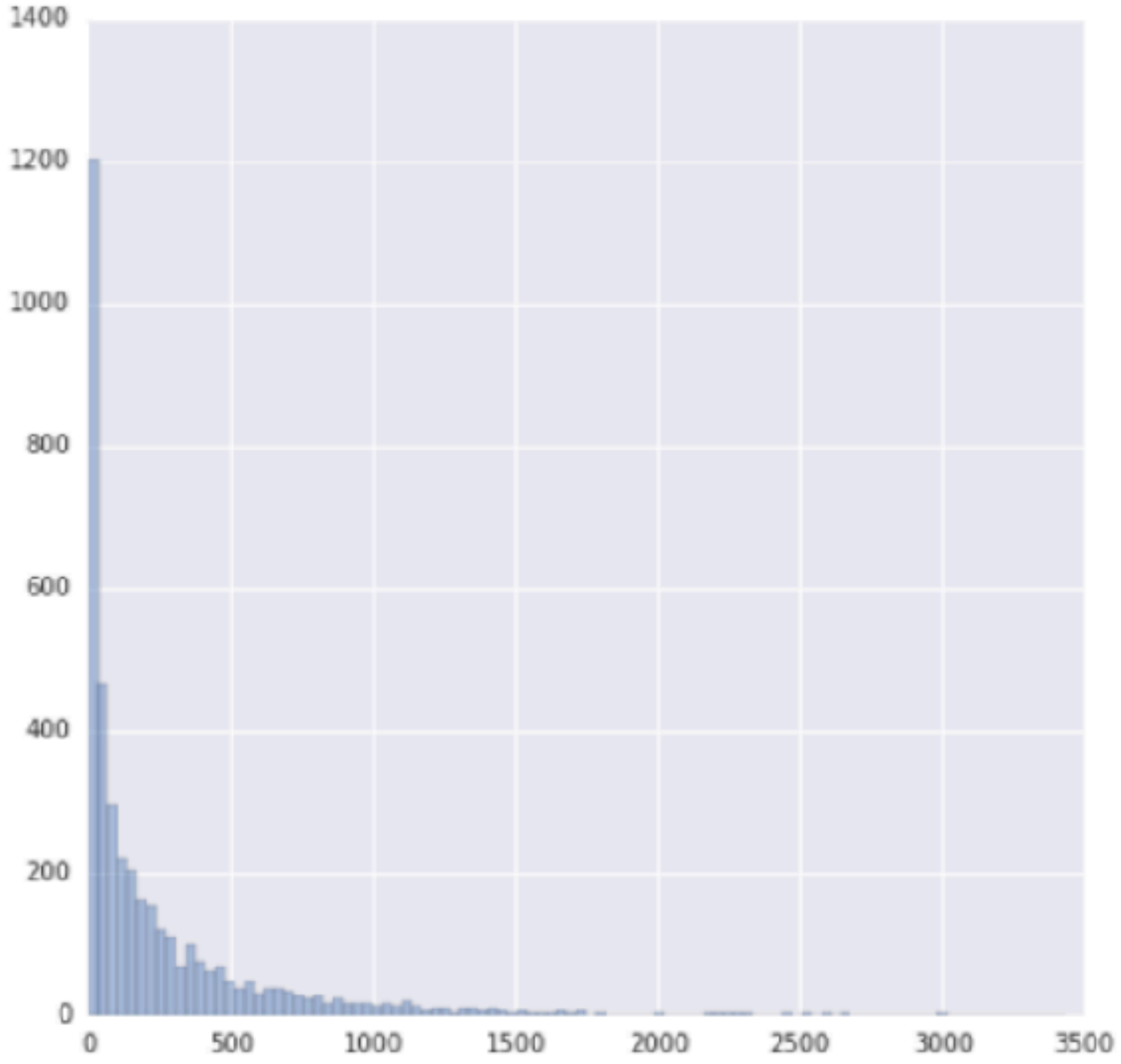
Source: kaggle.com

Data Normalization

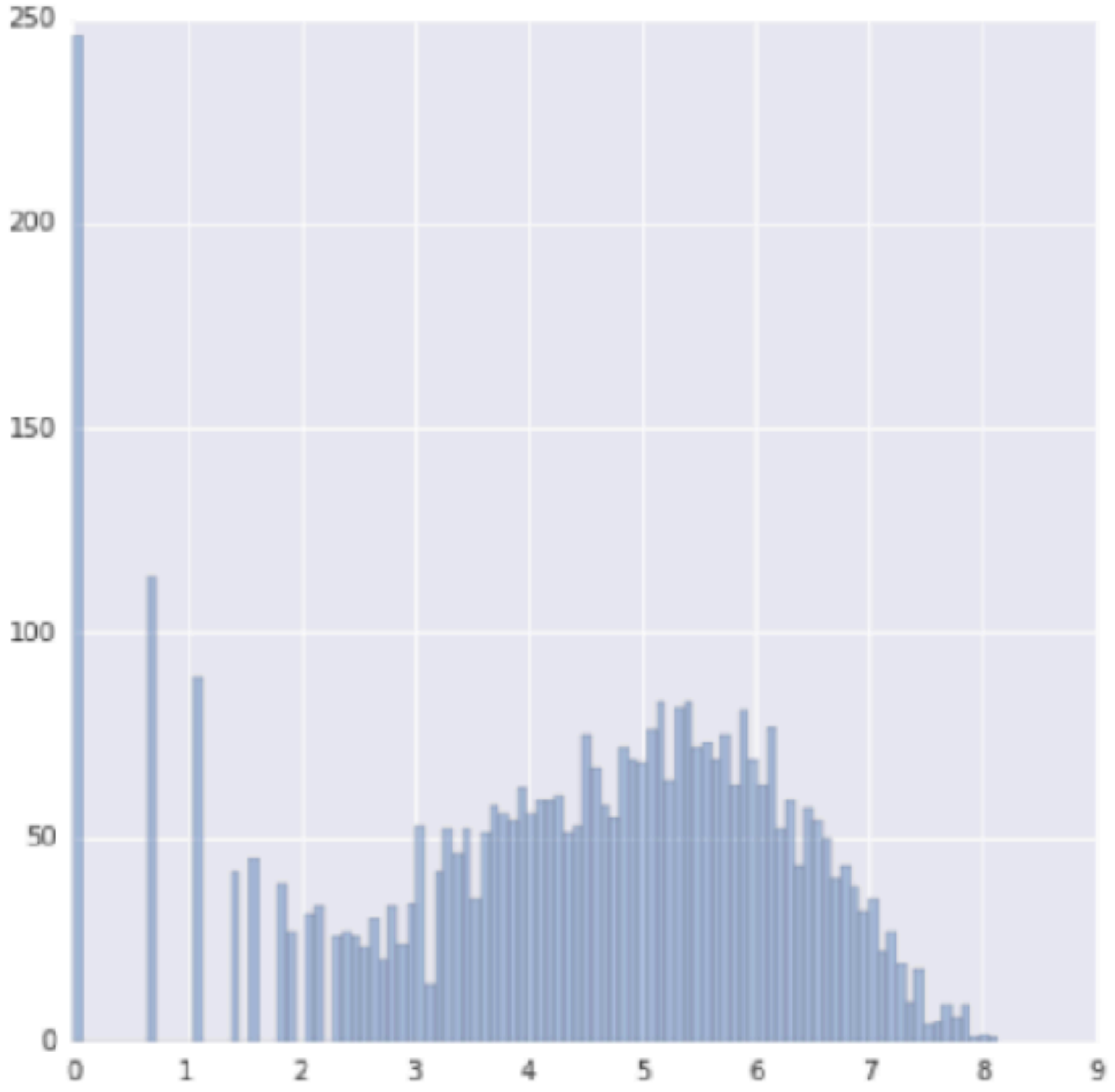
A process to transform the input **data** in a **well-behaved** form.

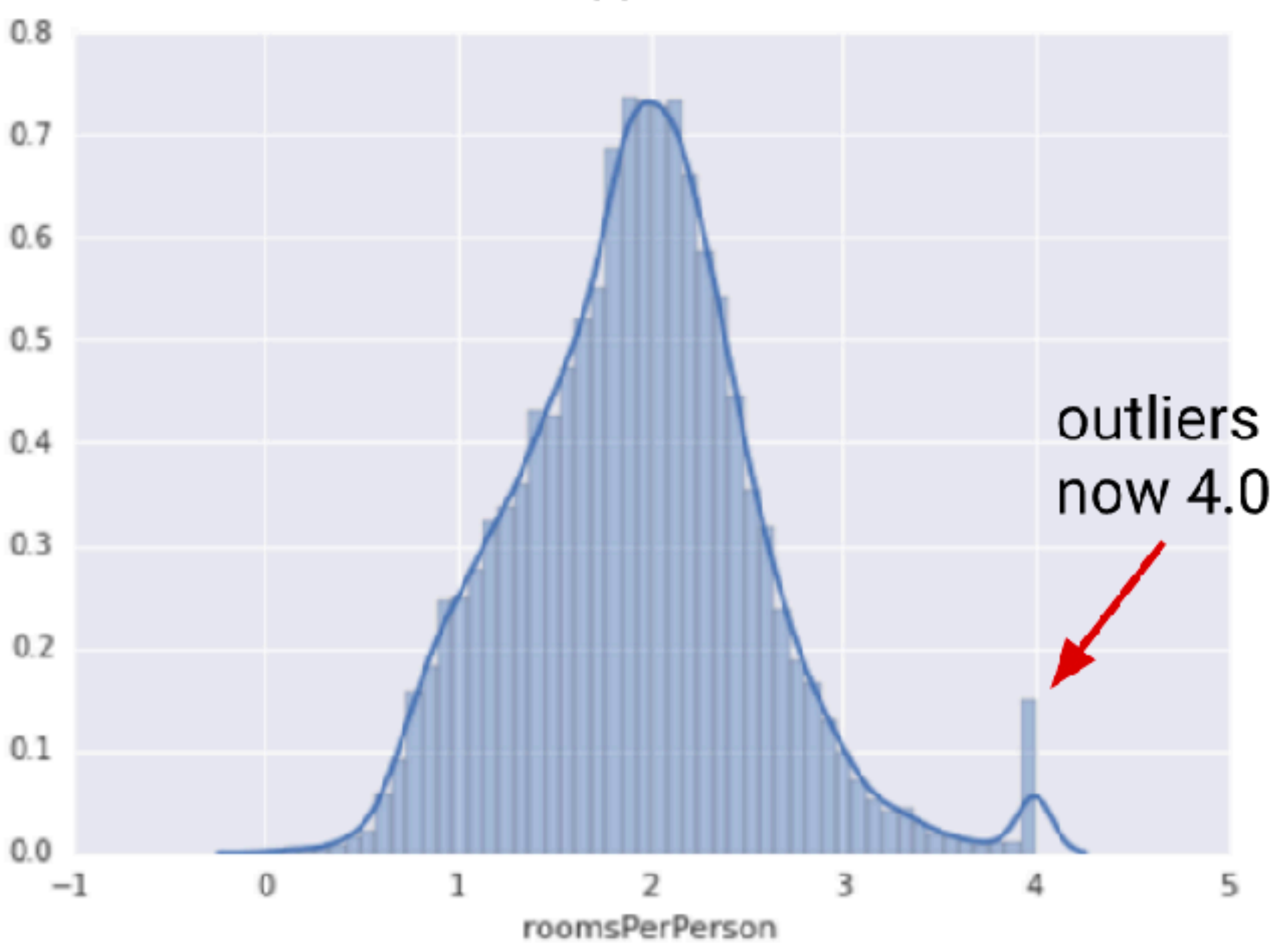


Ratings per movie



Log ratings per movie



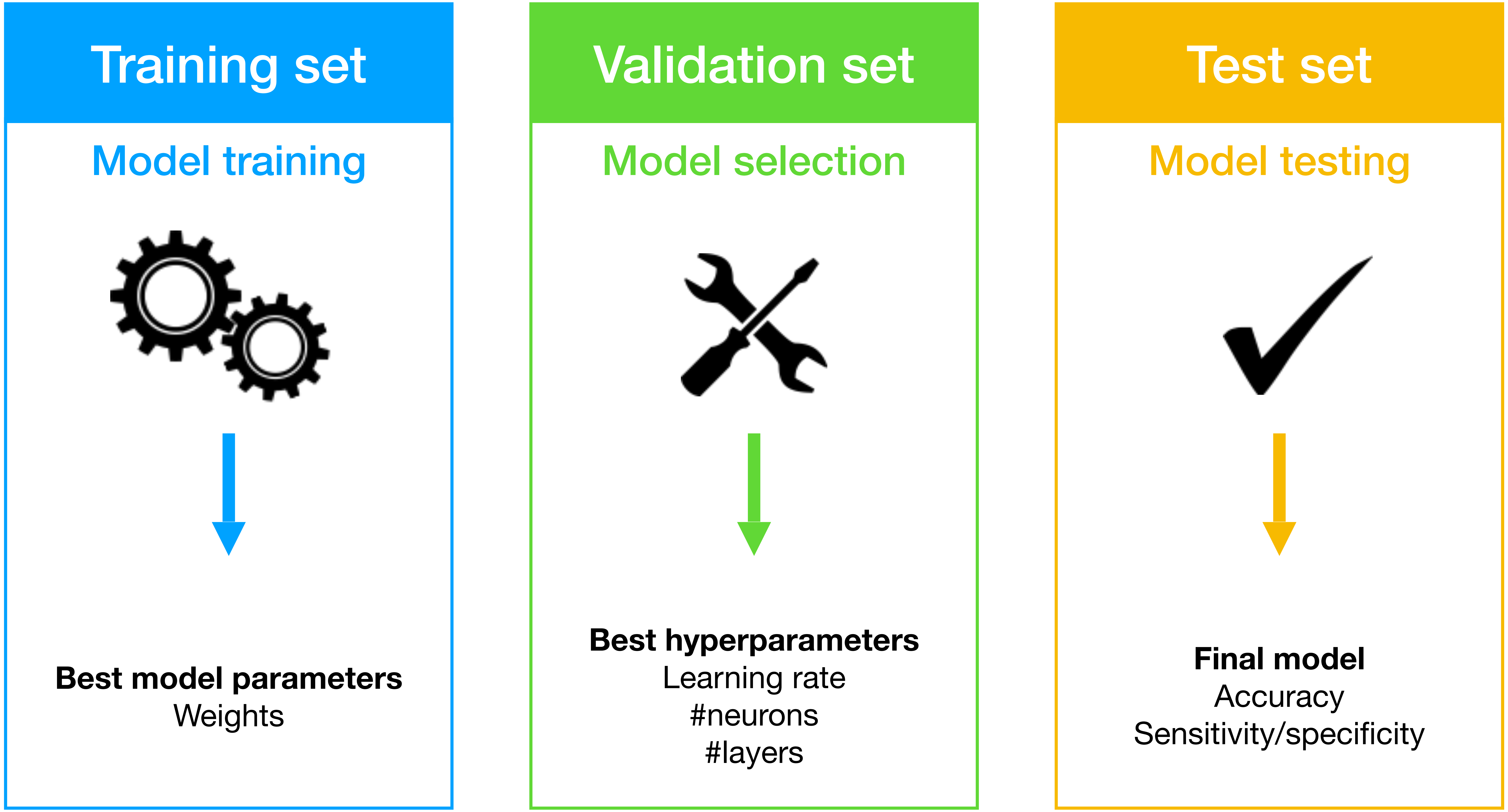


outliers are now 4.0

Further reading: sklearn scalers

Source: developers.google.com

Dataset splitting



Training set

Model training

Best model parameters
Weights

70%

Validation set

Model selection

Best hyperparameters
Learning rate
#neurons
#layers

20%

Test set

Model testing

Final model
Accuracy
Sensitivity/specificity

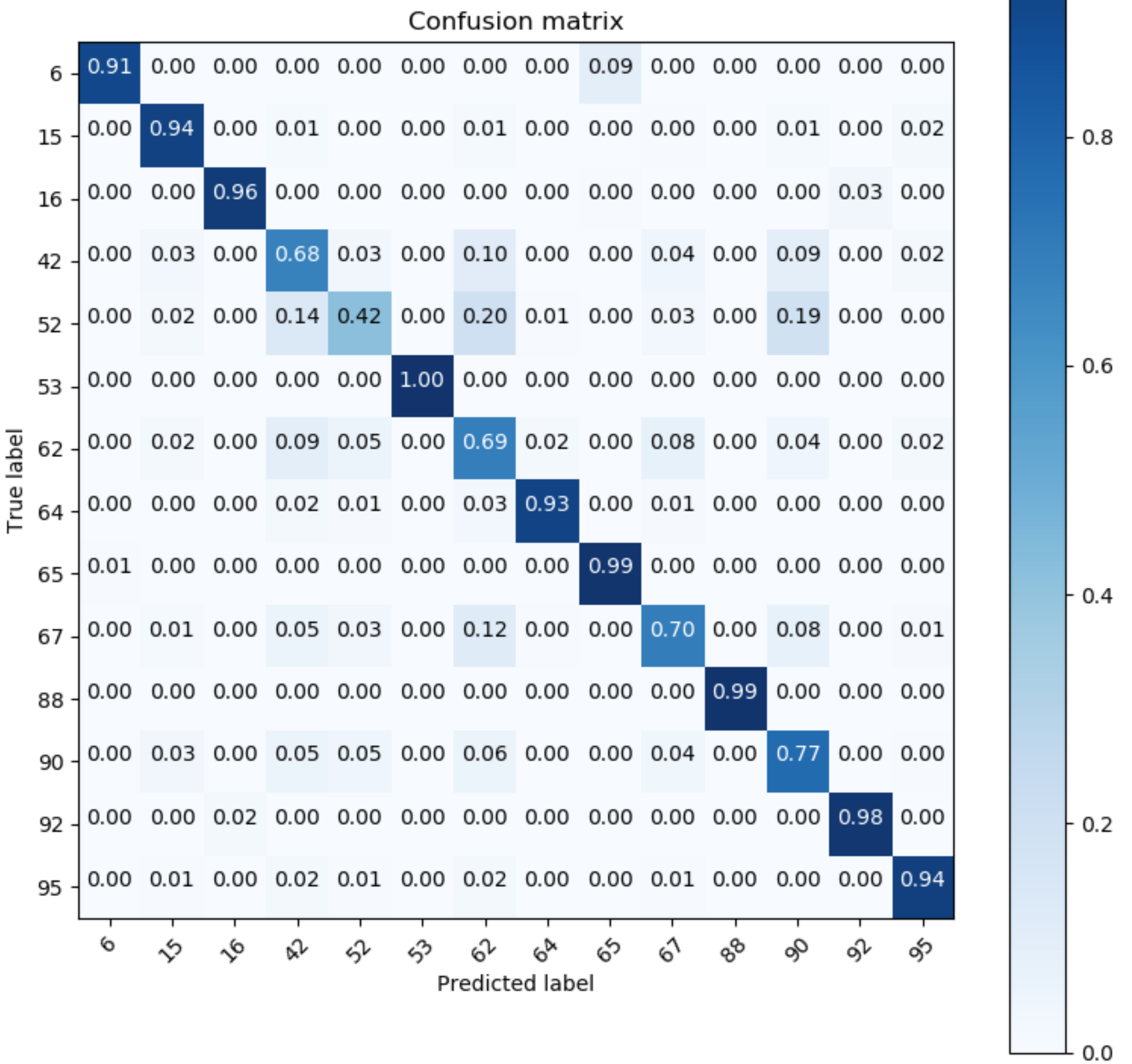
10%

Confusion Matrix

		Prediction outcome		
		p	n	total
actual value	p′	True Positive	False Negative	P′
	n′	False Positive	True Negative	N′
	total	P	N	

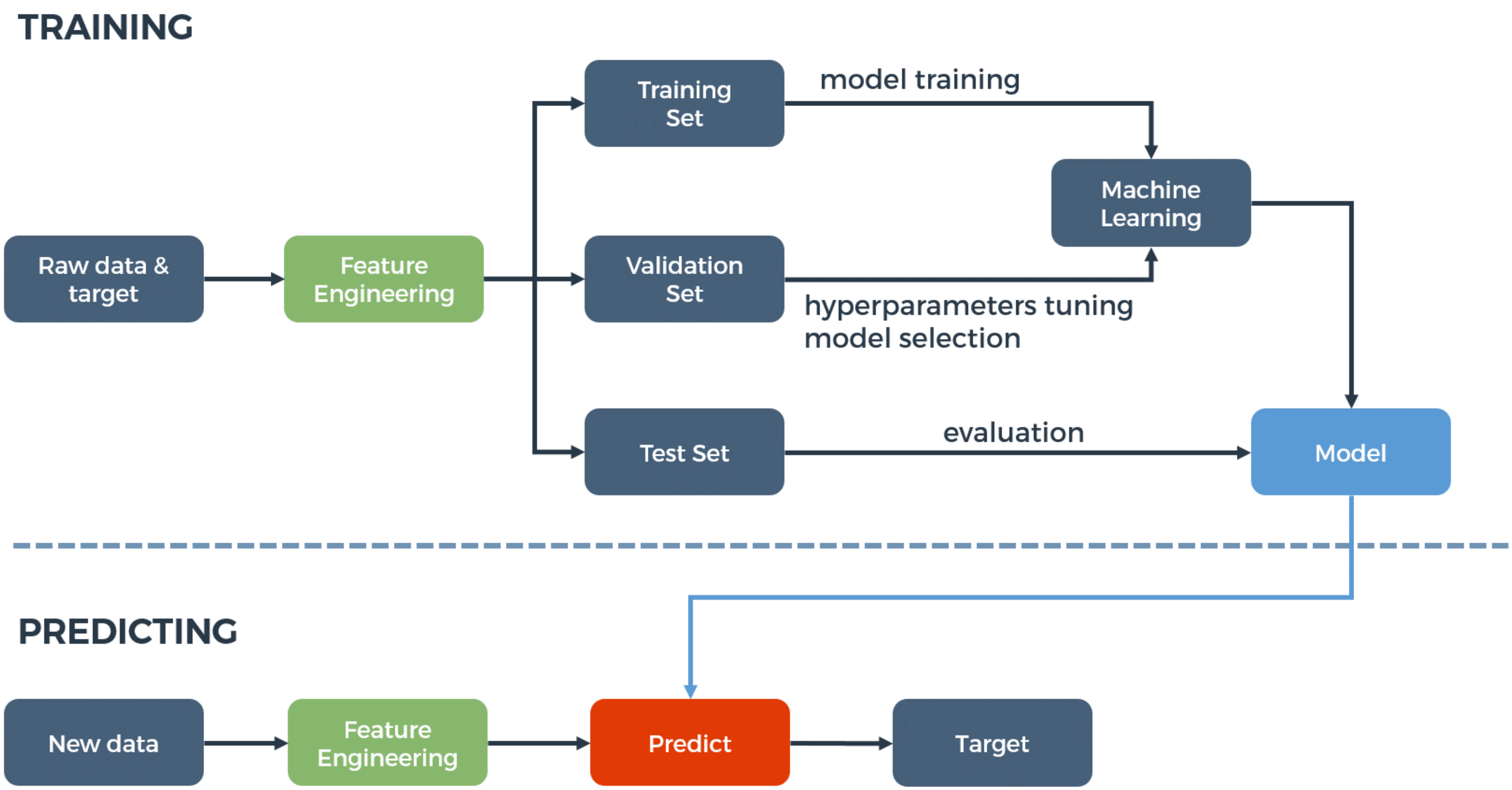
Accuracy = (TP + TN) / (TP + FN + FP + TN)
Precision (p) = TP / (TP + FP)
Recall (r) = TP / (TP + FN)

$$F_1 = \frac{2}{r^{-1} + p^{-1}}$$



	6	15	16	42	52	53	62	64	65	67	88	90	92	95
6	0.91	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.00	0.00	0.00	0.00	0.00
15	0.00	0.94	0.00	0.01	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.02
16	0.00	0.00	0.96	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00
42	0.00	0.03	0.00	0.68	0.03	0.00	0.10	0.00	0.00	0.04	0.00	0.09	0.00	0.02
52	0.00	0.02	0.00	0.14	0.42	0.00	0.20	0.01	0.00	0.03	0.00	0.19	0.00	0.00
53	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
62	0.00	0.02	0.00	0.09	0.05	0.00	0.69	0.02	0.00	0.08	0.00	0.04	0.00	0.02
64	0.00	0.00	0.00	0.02	0.01	0.00	0.03	0.93	0.00	0.01	0.00	0.00	0.00	0.00
65	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00
67	0.00	0.01	0.00	0.05	0.03	0.00	0.12	0.00	0.00	0.70	0.00	0.08	0.00	0.01
88	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00
90	0.00	0.03	0.00	0.05	0.05	0.00	0.06	0.00	0.00	0.04	0.00	0.77	0.00	0.00
92	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.98	0.00
95	0.00	0.01	0.00	0.02	0.01	0.00	0.02	0.00	0.00	0.01	0.00	0.00	0.00	0.94

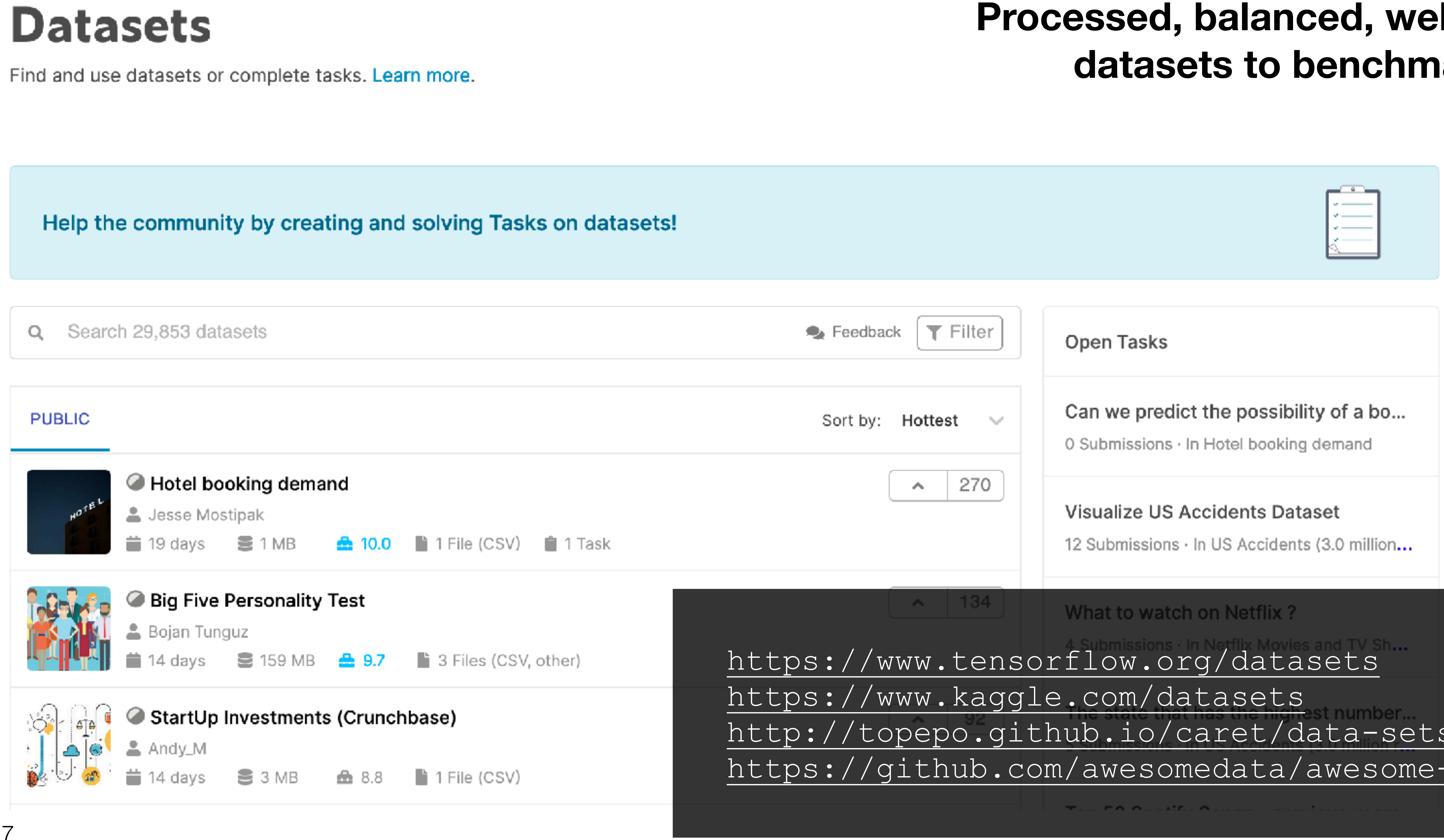
General Workflow of ML/DL



Source: hackernoon.com

Open Datasets

Processed, balanced, well-behaved, labeled datasets to benchmark your networks!



Datasets

Find and use datasets or complete tasks. Learn more.

Help the community by creating and solving Tasks on datasets!

Hotel booking demand
Jesse Mostipak
19 days · 1 MB · 10.0 · 1 File (CSV) · 1 Task · 270

Big Five Personality Test
Bojan Tunguz
14 days · 159 MB · 9.7 · 3 Files (CSV, other) · 134

StartUp Investments (Crunchbase)
Andy_M
14 days · 3 MB · 8.8 · 1 File (CSV) · 92

Open Tasks

Can we predict the possibility of a bo...
0 Submissions · In Hotel booking demand

Visualize US Accidents Dataset
12 Submissions · In US Accidents (3.0 million...

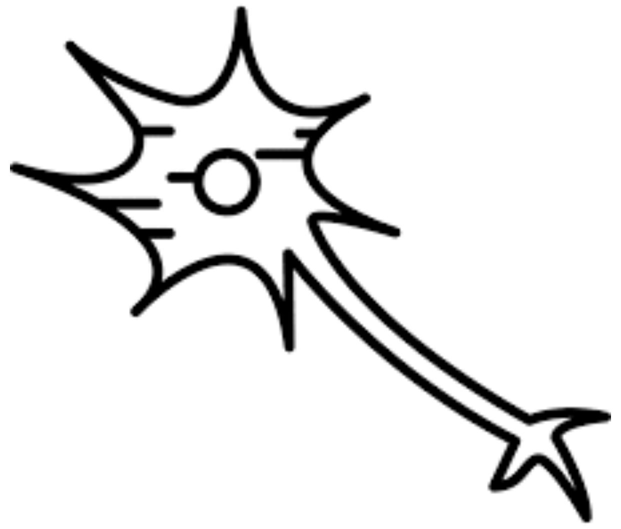
What to watch on Netflix ?
4 Submissions · In Netflix Movies and TV Sh...

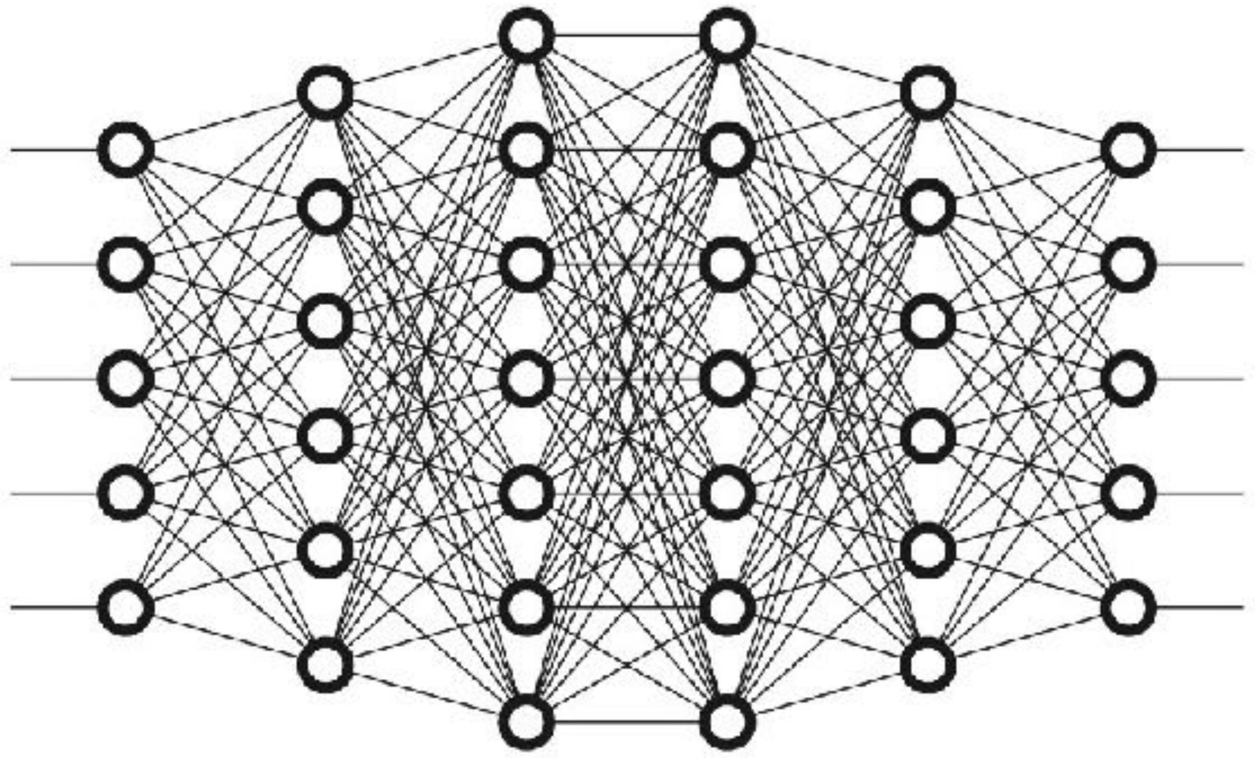
The state that has the highest number...

https://www.tensorflow.org/datasets
https://www.kaggle.com/datasets
http://topepo.github.io/caret/data-sets.html
https://github.com/awesomedata/awesome-public-datasets

Take-home messages

In a neuron:
… the main job is to calculate a **weighted average**
… the **decision** is made through the **activation** function





In a neural network:
… losses are calculated using the loss function
… losses are calculated by **comparing** the truths and the prediction
… **predictions** are made through **forward** propagation
… weights are **updated** through the **backward** propagation process
… **optimizers** are used to decide the weights updating **strategies**

In a deep learning workflow:
… the heavy lifting is mostly done by **DL frameworks**
… open datasets are crucial for benchmarking and bootstrapping DNNs