Deep Learning Fundamentals

Mihai DOGARIU

www.mdogariu.aimultimedialab.ro

Summary

M1. Introduction

M2. The basic process of learning

M3. Terminology

M4. Practical considerations

M5. Supervised Deep Learning (general info)

M6. Classification

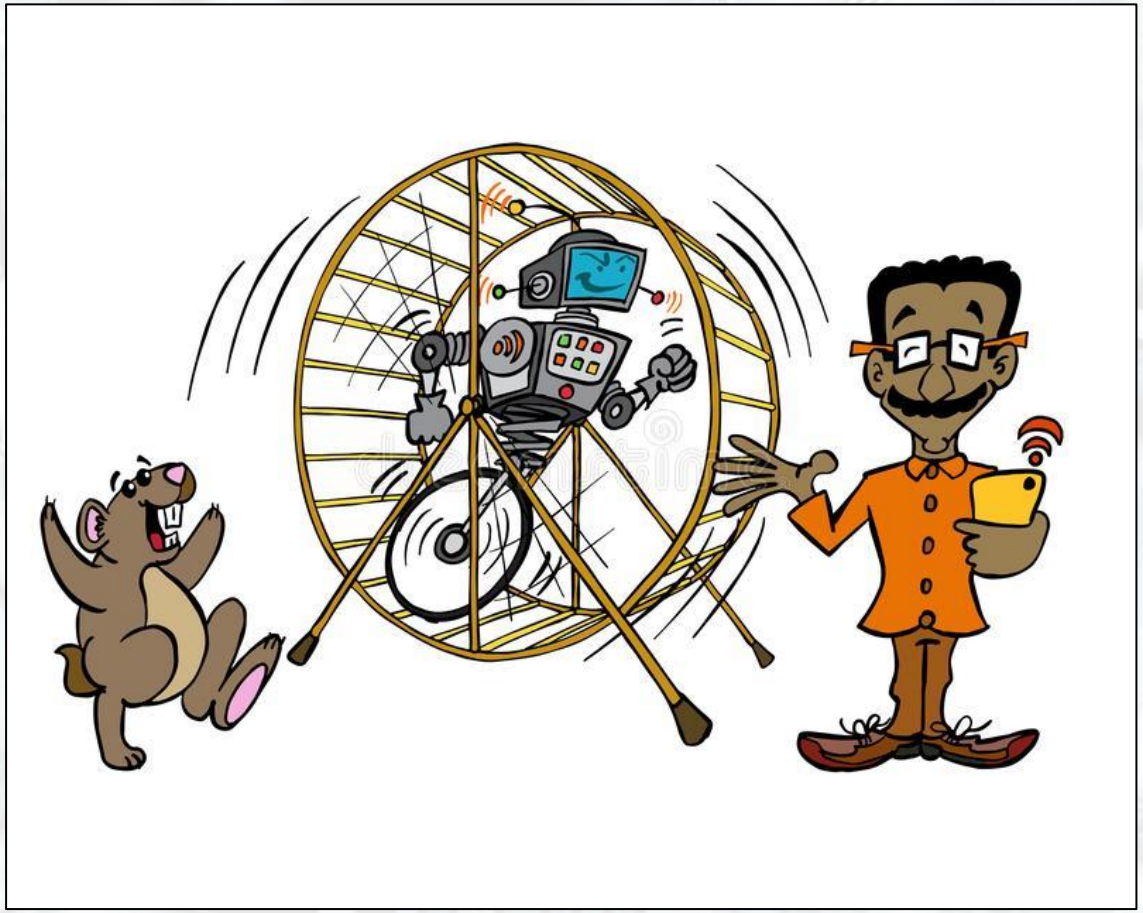
M7. Unsupervised Deep Learning (general info)

M1. Introduction

Image: www.dreamstime.com

Identifying the problems

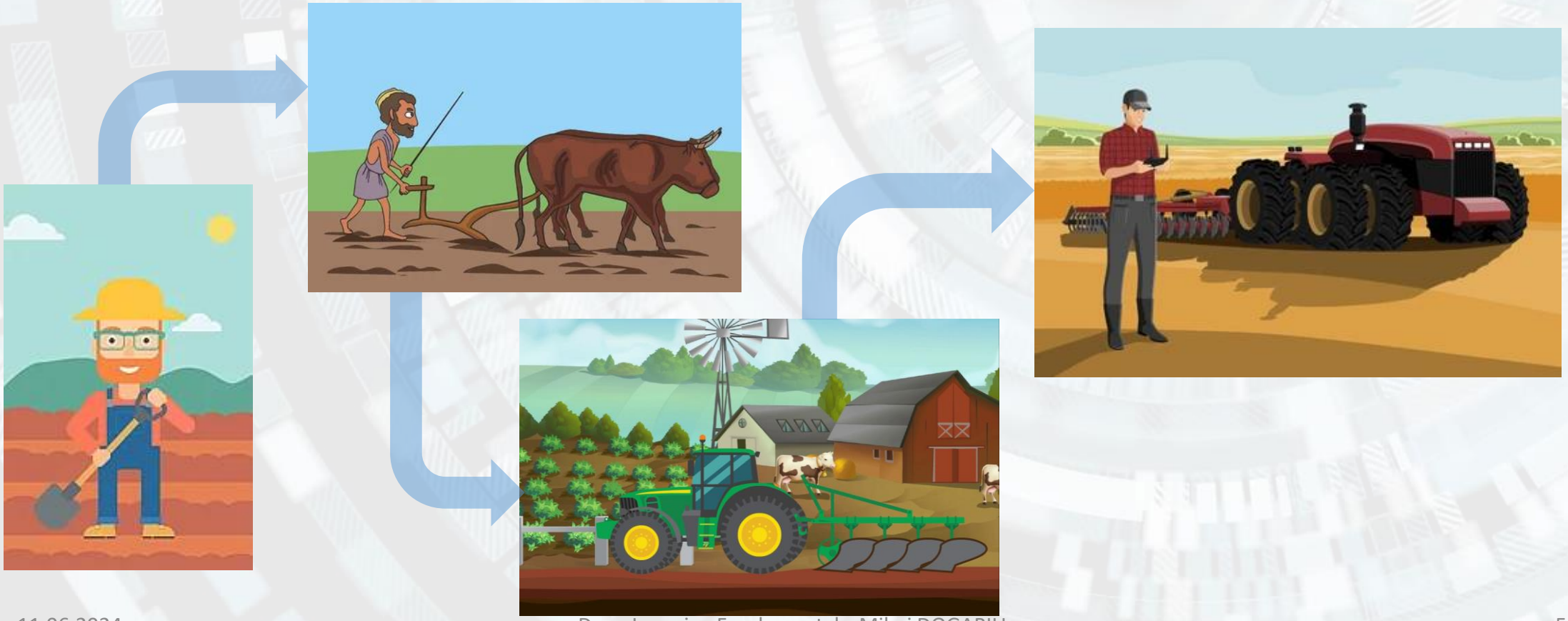
- Need for process automation (redundant ones, most of the time).
- Human capacity is limited and prone to erros, especially after longer time periods (approx. 6h).
- Increasing cost/efficiency ratio.



Images: www.canstockphoto.com, www.123rf.com, www.shutterstock.com

Solving the problems

- Externalizing/replacing human effort.



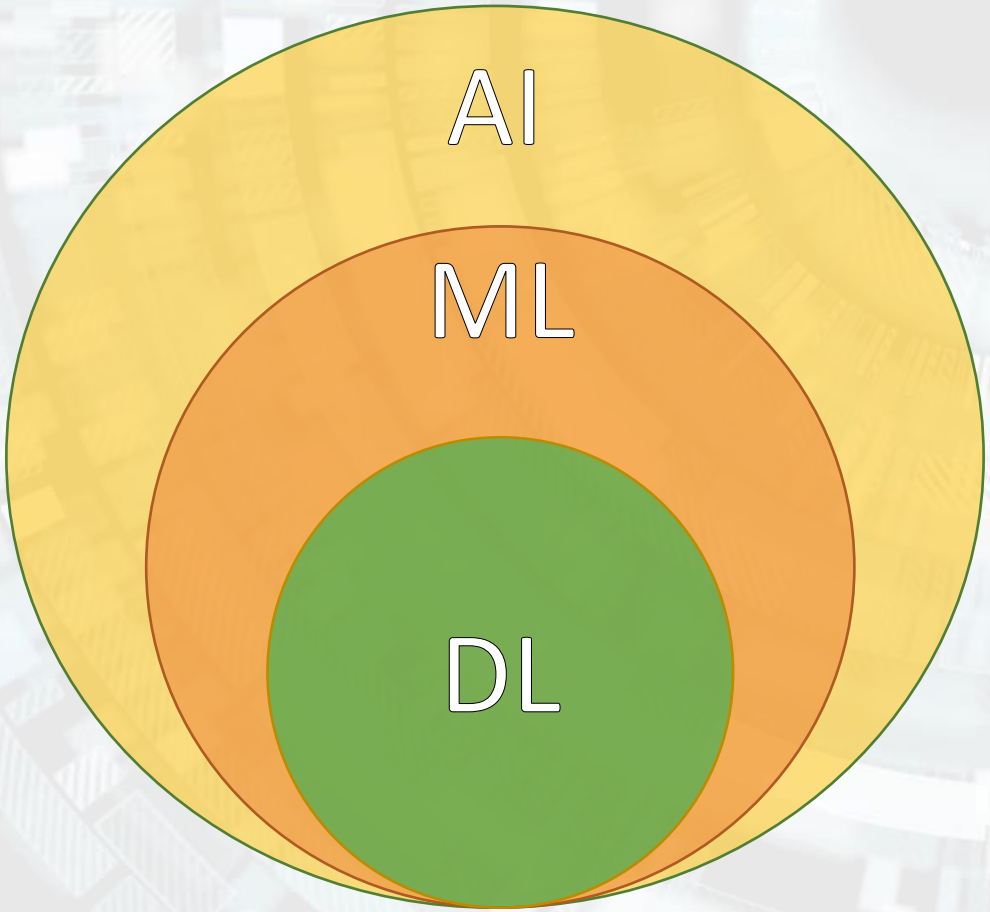
AI, ML, DL?

Artificial Intelligence (AI) = the ability of computer systems to perform tasks normally requiring human intelligence.

Machine Learning (ML) = AI subdomain in which systems are designed with the ability to learn based on previous examples.

Deep Learning (DL) = ML subdomain in which systems are designed based on the human neural network model.

AI, ML, DL?



The Deep Learning Boom

What lead to the current exponential advancement of deep learning?

1. Higher computing power (hardware) – GPU Let's test it out – unit #1
2. A lot more data => better results
3. Optimized frameworks (software): Tensorflow, PyTorch, Caffe, MXNet, DeepLearning4J etc.
4. Attention and financial influx from the industry: Facebook AI Research (FAIR), Google Deepmind, NVIDIA, OpenAI, Microsoft Research, AWS Deep Learning etc.

Image: Pinterest

The biologic neuron

- The fundamental cell of the nervous system

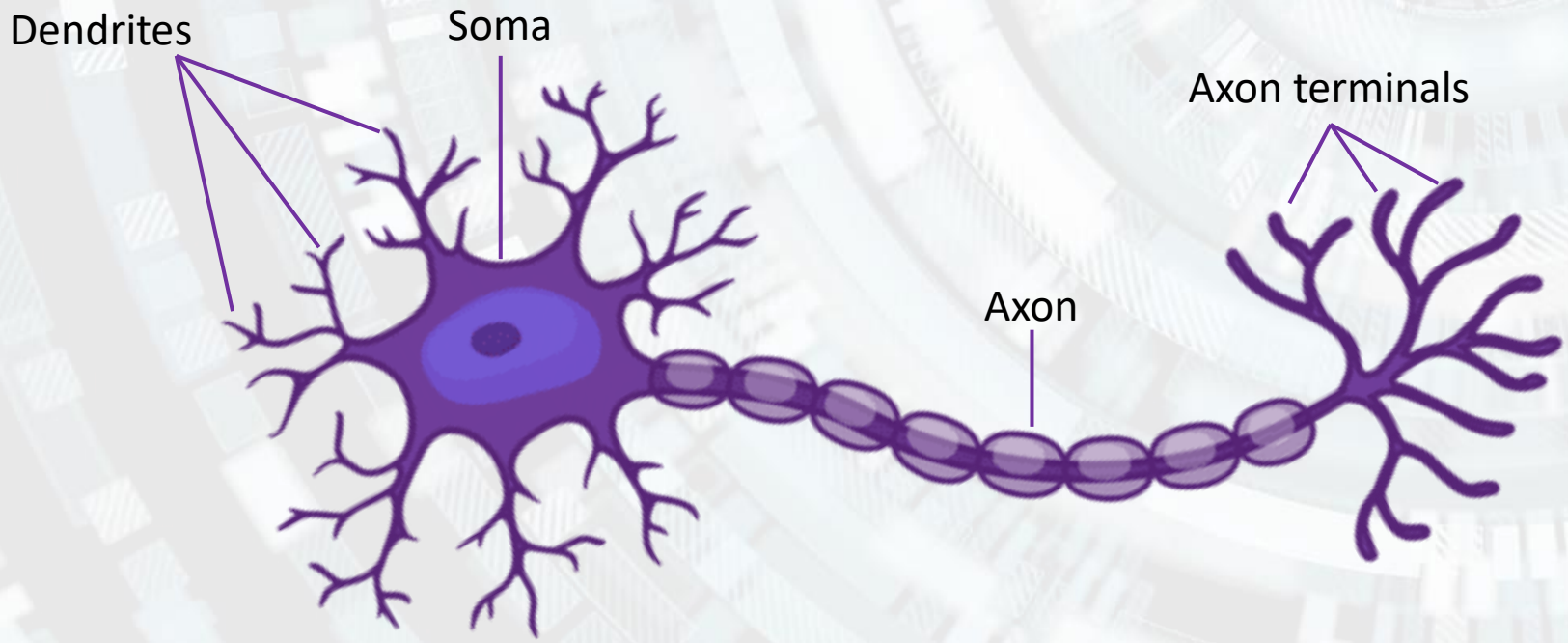
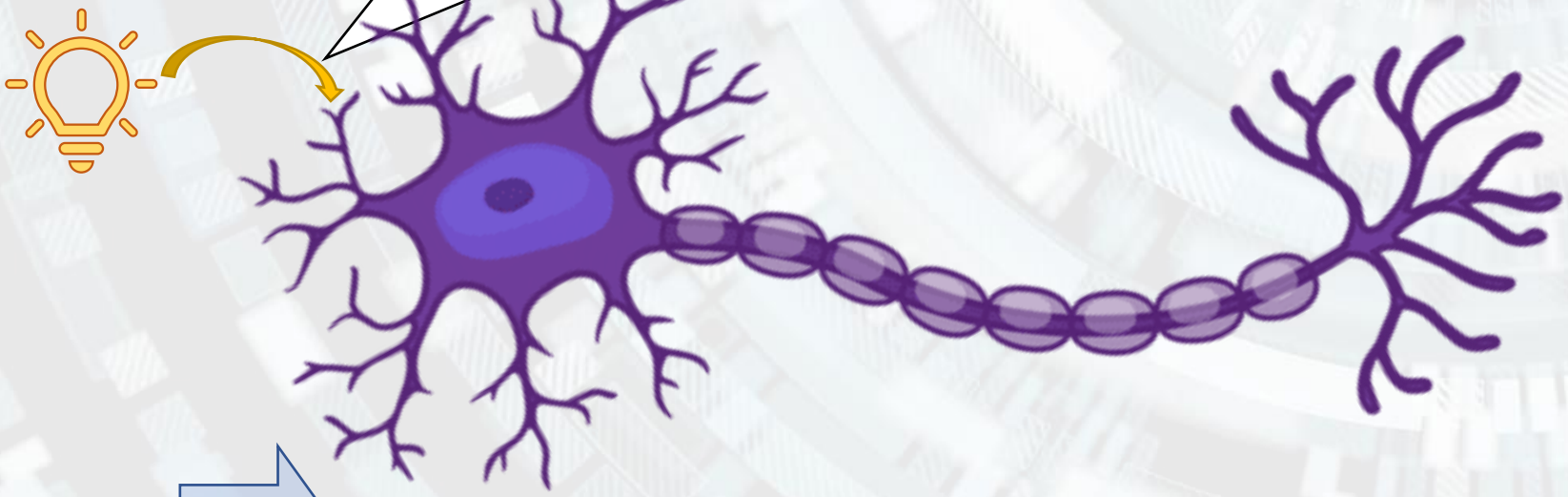


Image: Pinterest

The biologic neuron

Dendrites are the input gate for neurons. They are connected to and receive signals from other neurons (>1000). Each dendrite weights the signal that it receives in a different manner (inhibit or excite).



Information flow

Image: Pinterest

The biologic neuron

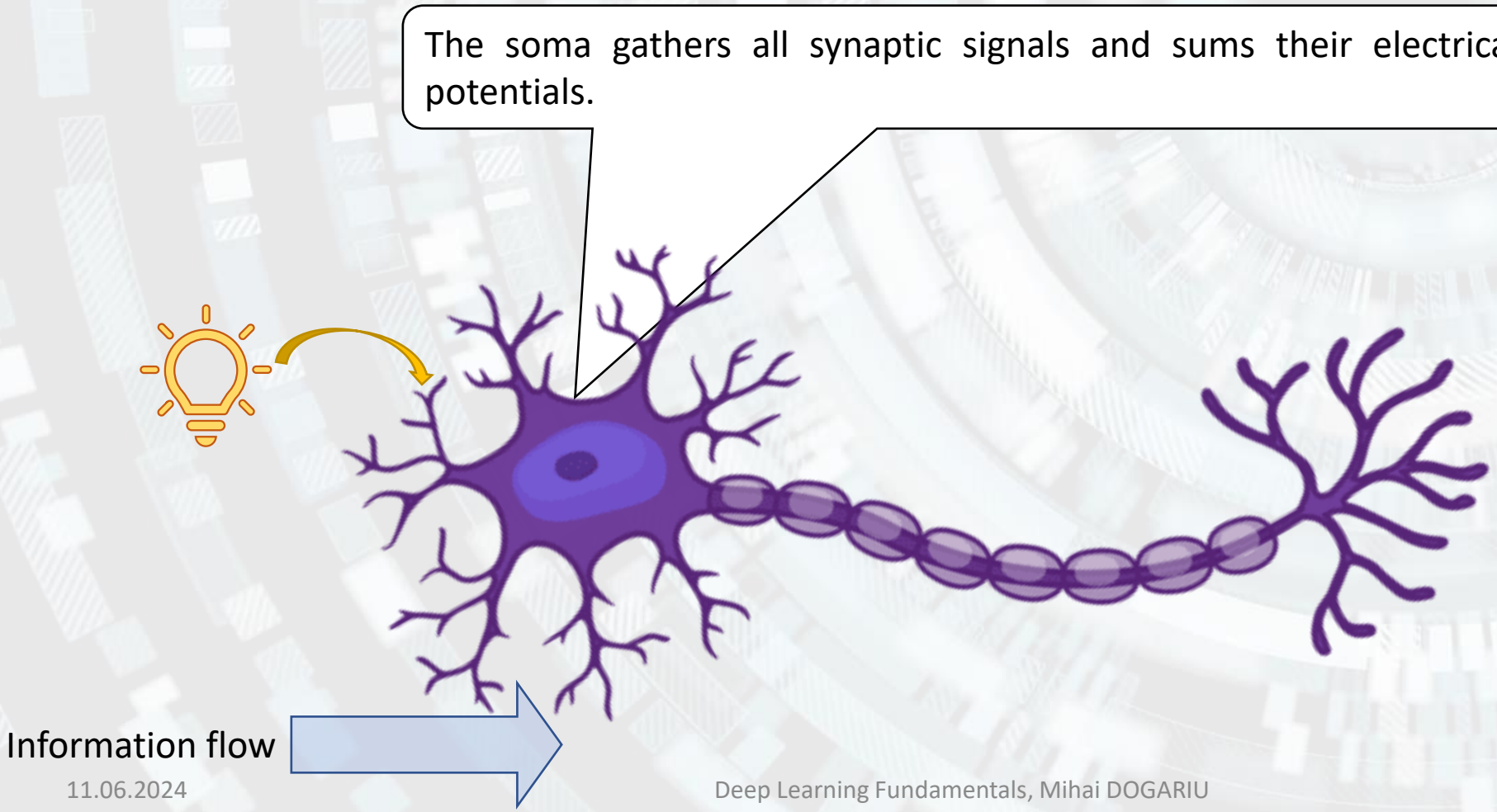
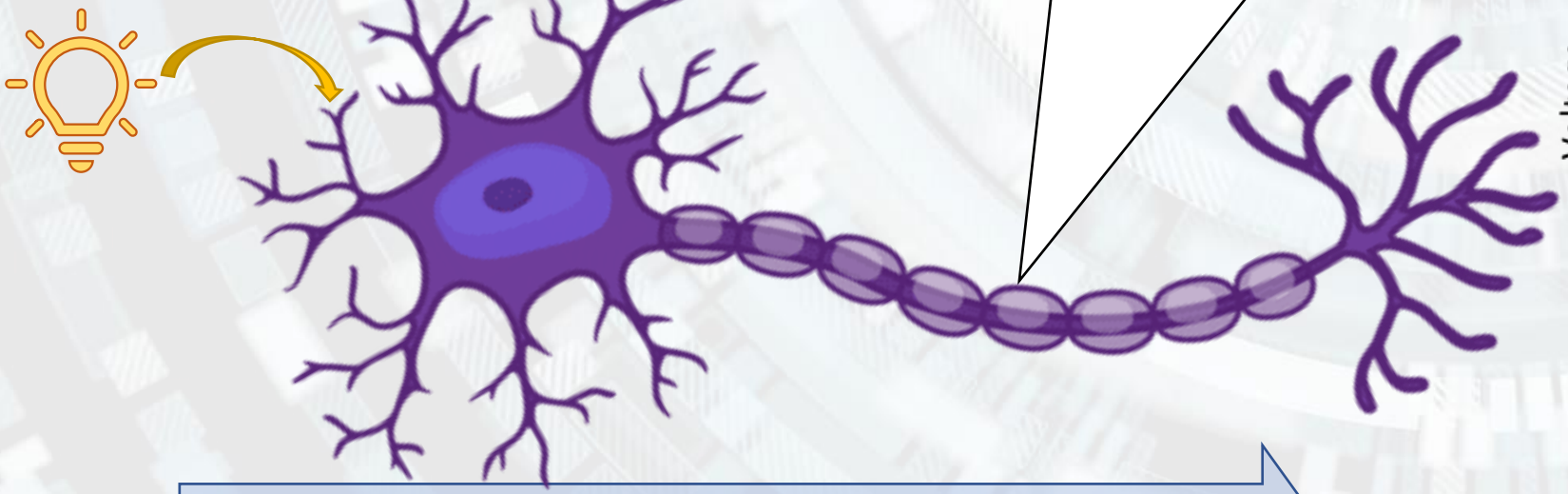
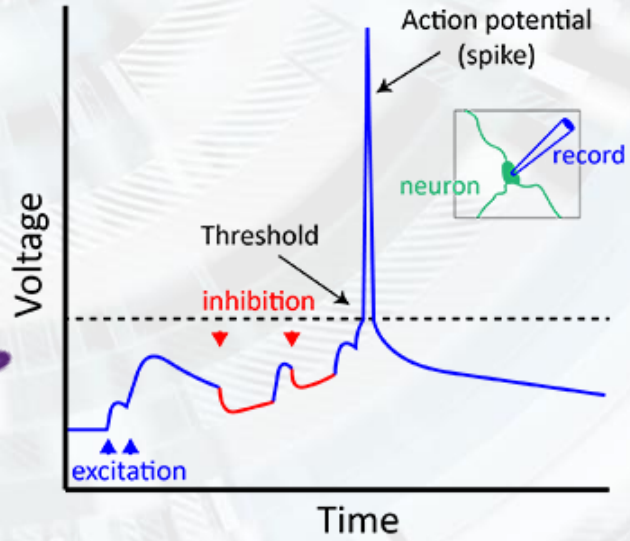


Image: Pinterest

The biologic neuron

If the electrical charge from the soma exceeds a given threshold (bias), then the neuron fires and the signal is transmitted onwards.



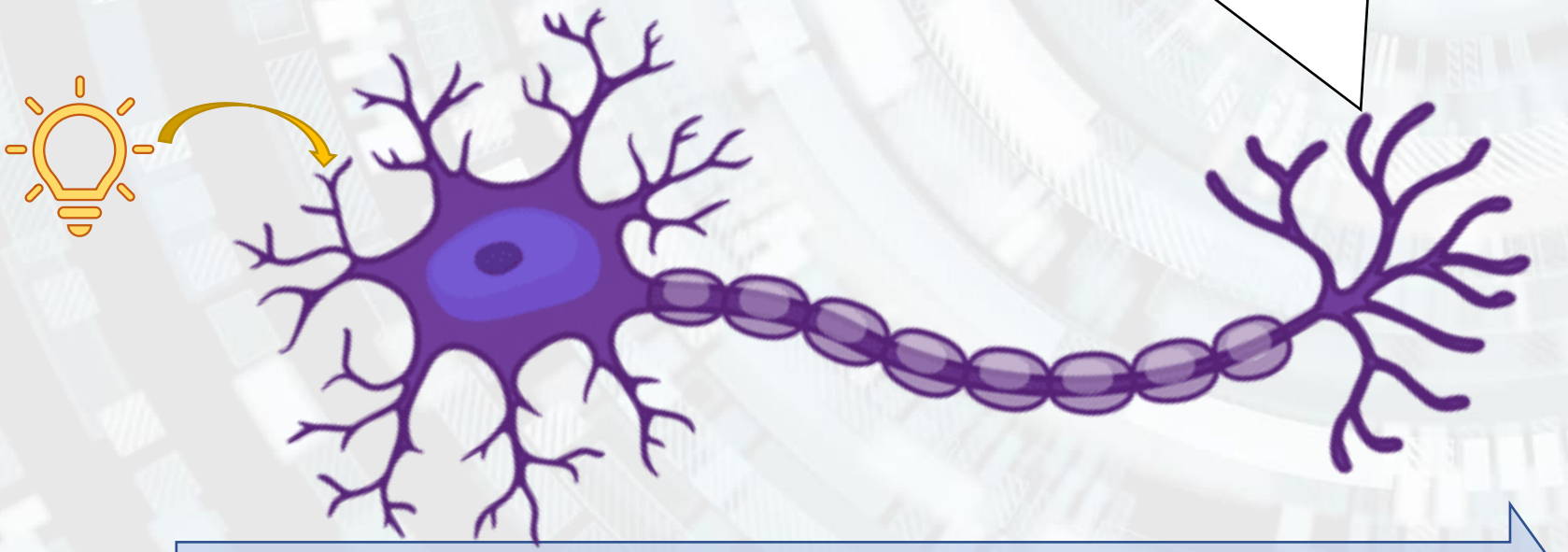


Information flow

Image: Pinterest

The biologic neuron

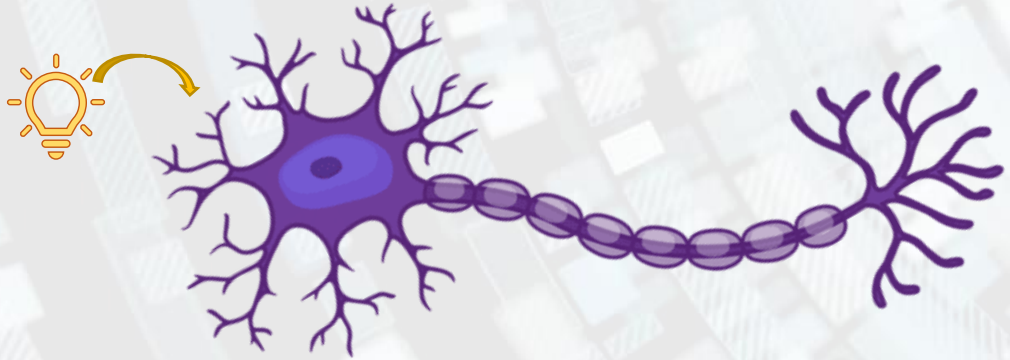
If the neuron was activated, its signal is sent to all other neurons connected to its axon terminals.



Information flow

The artificial neuron

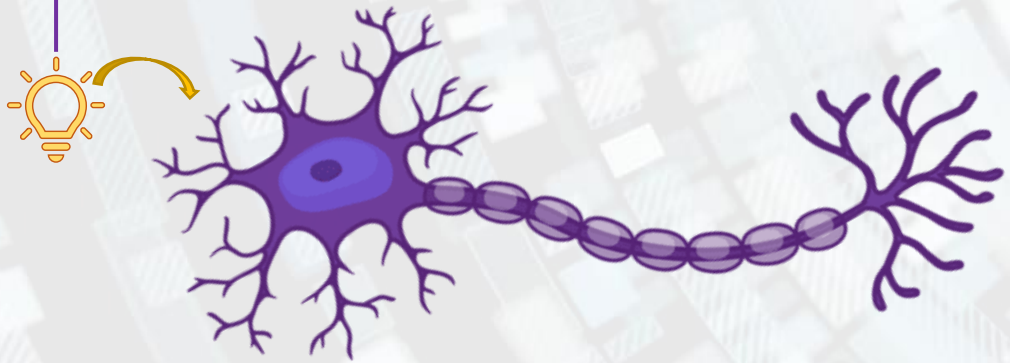
Represents a mathematical model of the biologic neuron.



The artificial neuron

The biologic neuron

Input



The artificial neuron

x_1

x_2

x_3

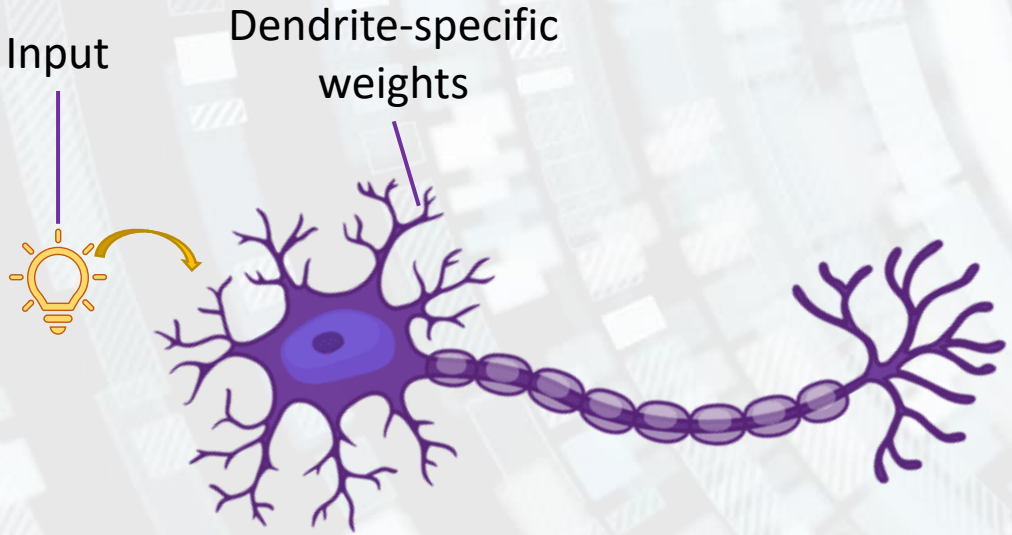
x_4

$\vdots$

x_n

The artificial neuron

The biologic neuron



The artificial neuron

$x_1 \times w_1$

$x_2 \times w_2$

$x_3 \times w_3$

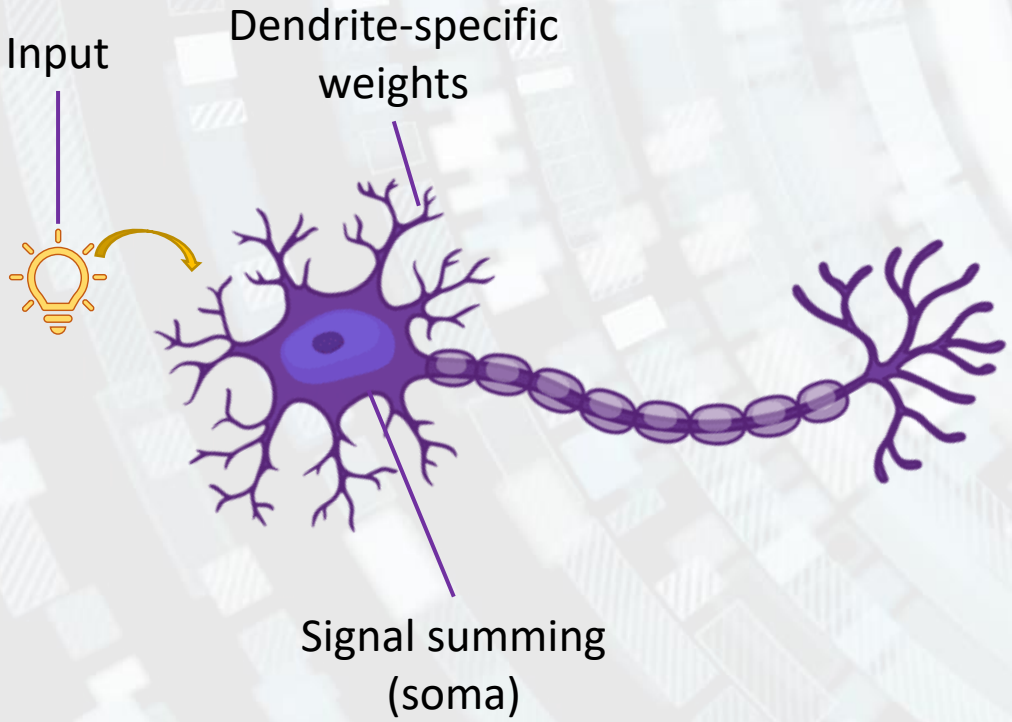
$x_4 \times w_4$

$\vdots \quad \vdots$

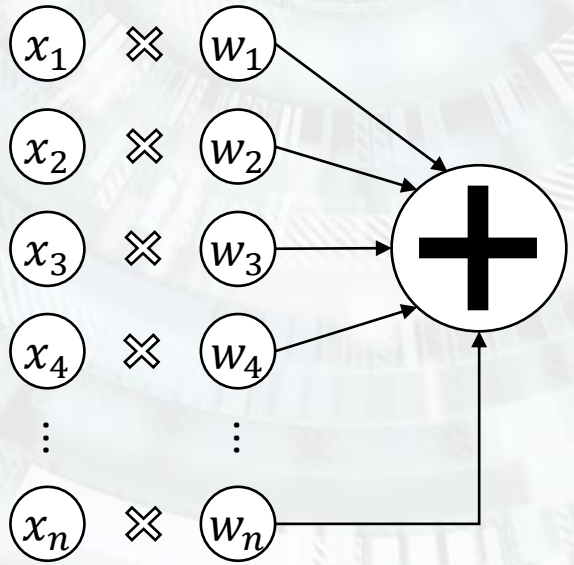
$x_N \times n$

The artificial neuron

The biologic neuron

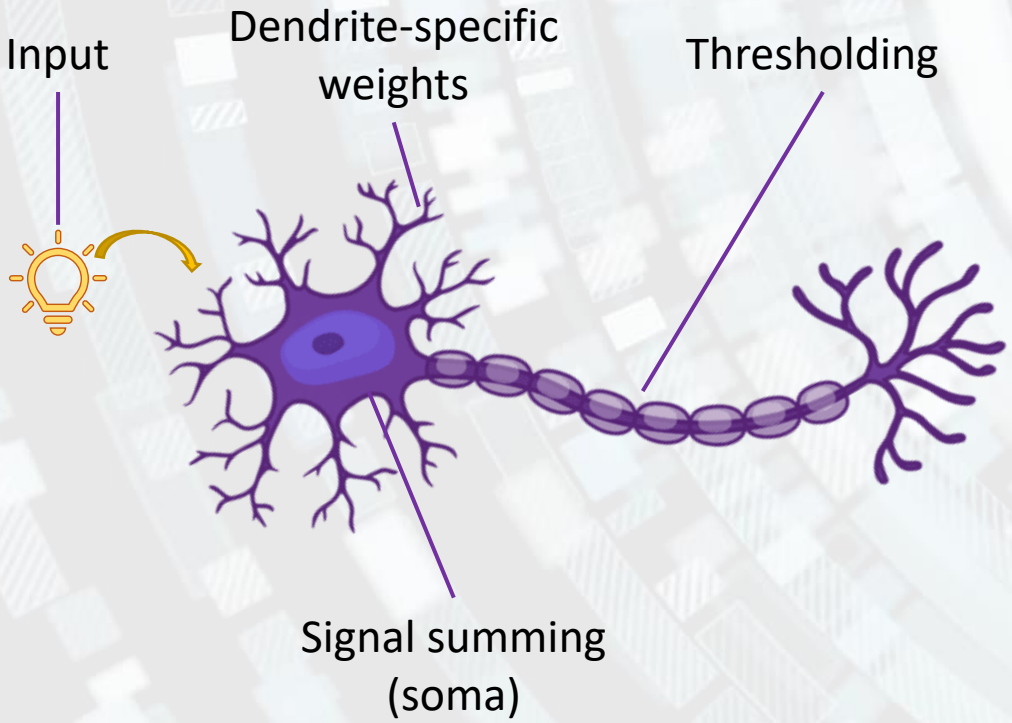


The artificial neuron

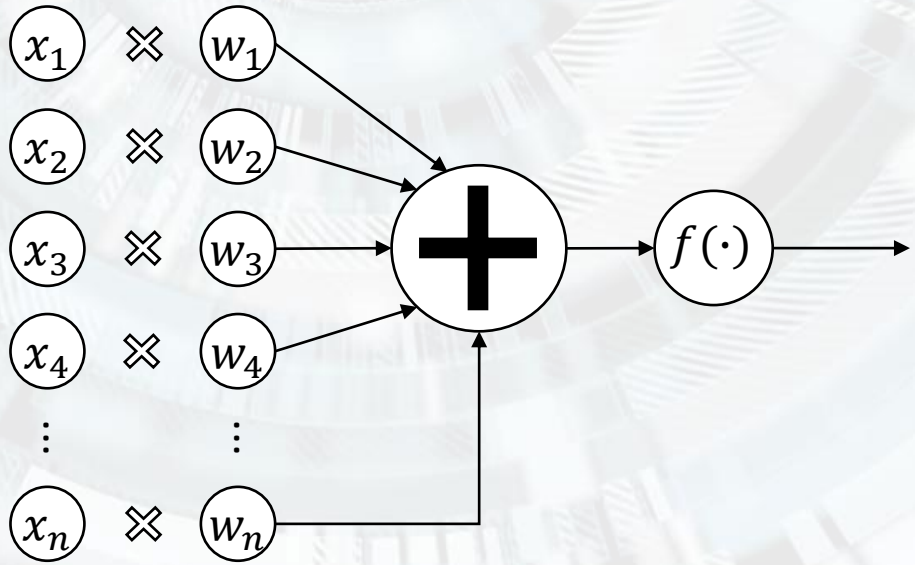


The artificial neuron

The biologic neuron

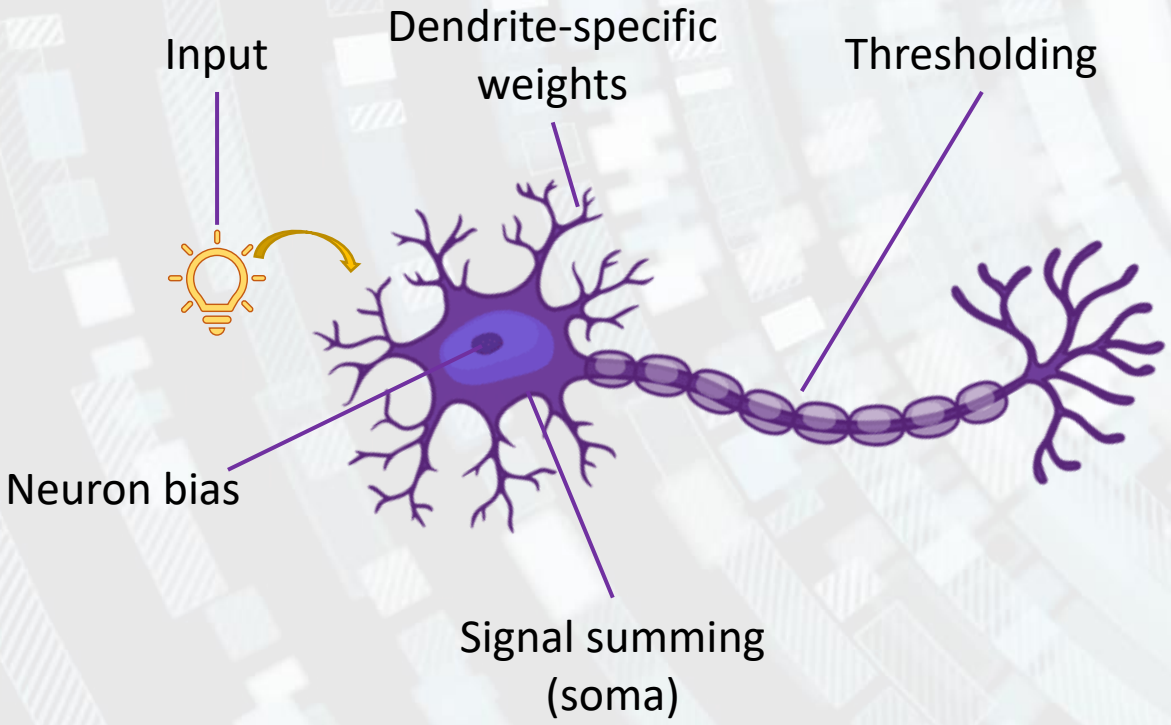


The artificial neuron

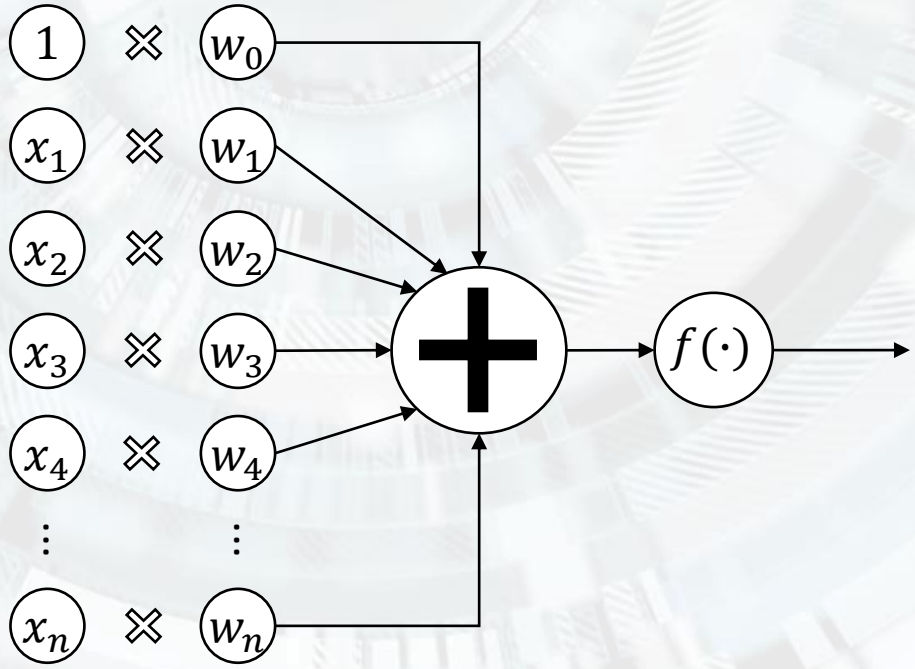


The artificial neuron

The biologic neuron



The artificial neuron



The artificial neuron

The mathematical model:

$$y = f\left(\sum_{i=0}^{n} w_i x_i\right) = f\left(\sum_{i=1}^{n} w_i x_i + b\right) = f(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

w_i – weights; $w_0 \rightarrow b$ - bias
x_i – input vector; $x_0 = 1$
n – input vector dimension
$f(\cdot)$ – activation function
y – neuron output.

The artificial neuron – graphical representation

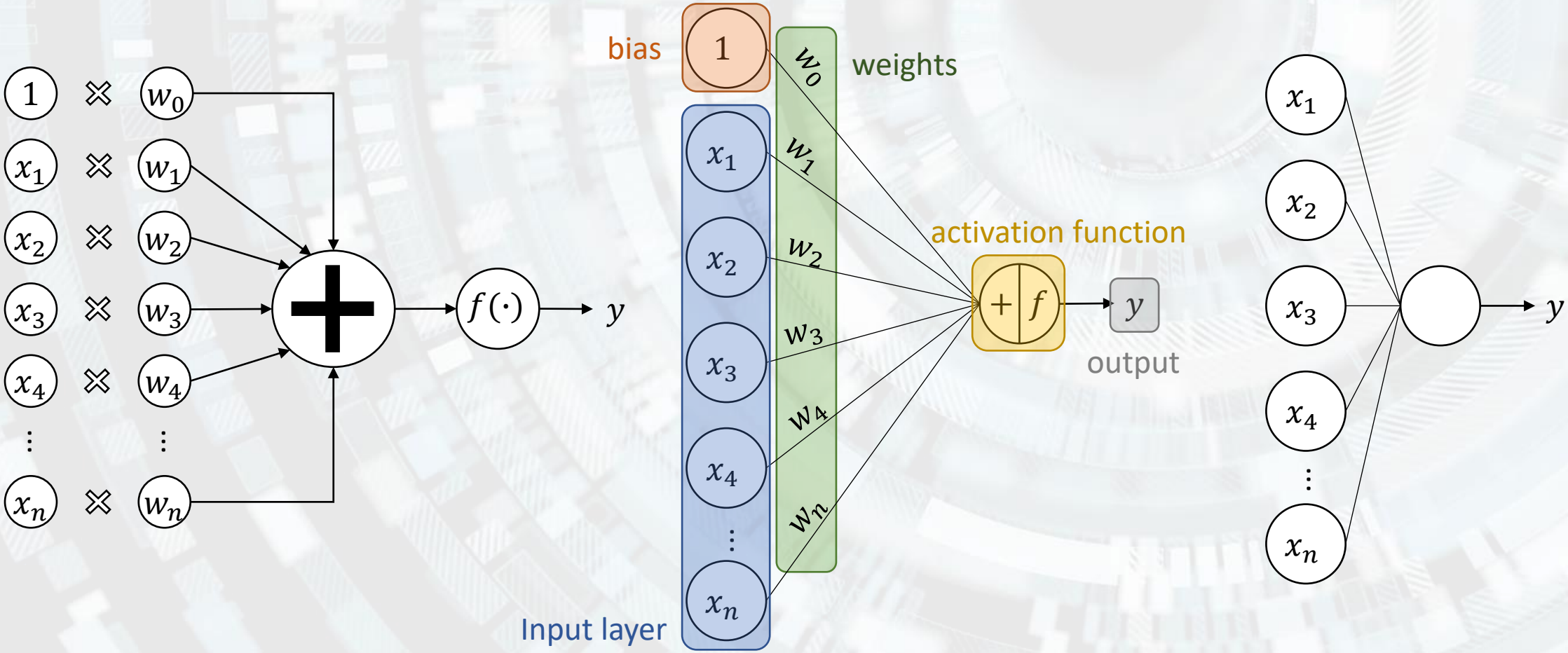
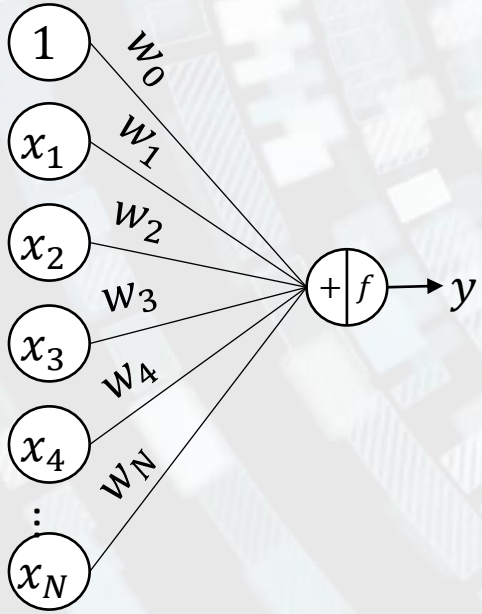


Image: veryicon

Learning of an artificial neuron

Learning = iterative process through which all parameters belonging to the neuron are learned such that the model correctly classifies the input data.



What are the fixed parameters?
- input
- output
- transfer function

What are the adjustable parameters?
- weights
- bias

Image: veryicon

Learning of an artificial neuron

Algorithm:

1. initialize weights with a low enough value.
2. for each input-output pair $(\boldsymbol{x}_j, \hat{y}_j)$ from the training set:
   2.1. compute the output of the system:

$$y_j(\mathrm{t}) = f\left(\sum_{i=0}^{n} w_i x_{j,i}\right)$$

   2.2. update the weights, $\forall i, 0 \leq i \leq n$:

$$w_i(\mathrm{t}+1) = w_i(t) + \alpha * \left(\hat{y}_j - y_j(\mathrm{t})\right) * x_{j,i}$$

3. repeat step 2 until the iteration error is lower than a given threshold or until sufficient iterations have been run.

Let's test it out – unit #2

Image: Pinterest

Linear separable space

A bidimensional space consisting of 2 classes of points is linearly separable if there exists at least one straight line separating the space into its 2 distinct classes.

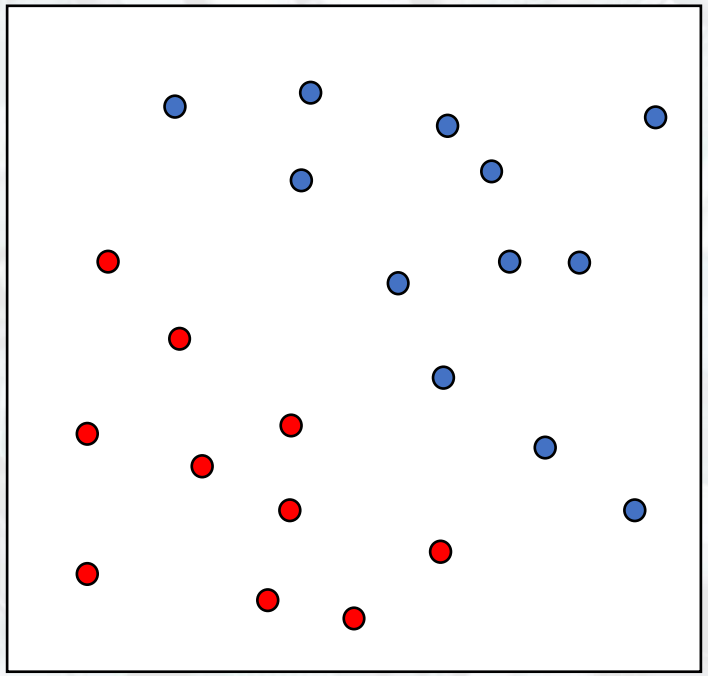


Image: Pinterest

Linear separable space

A bidimensional space consisting of 2 classes of points is linearly separable if there exists at least one straight line separating the space into its 2 distinct classes.

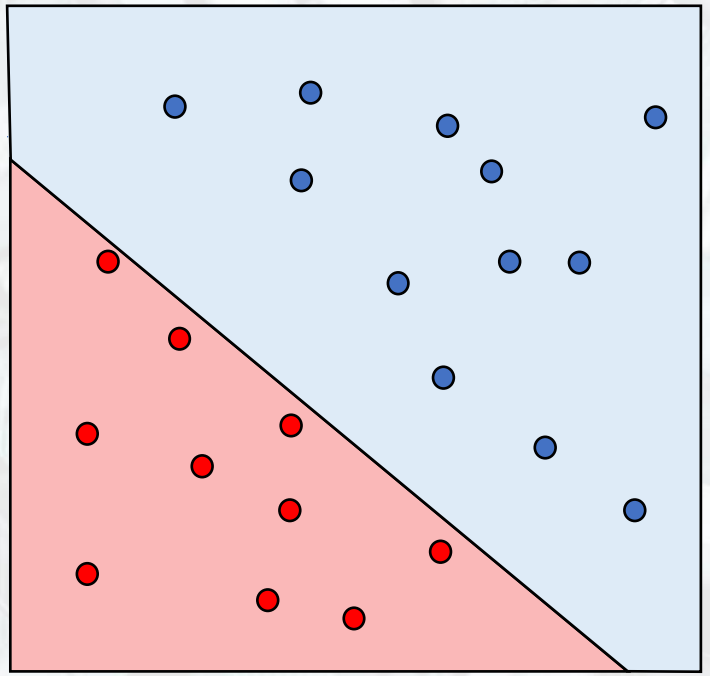


Image: Pinterest

Linear separable space

A bidimensional space consisting of 2 classes of points is linearly separable if there exists at least one straight line separating the space into its 2 distinct classes.

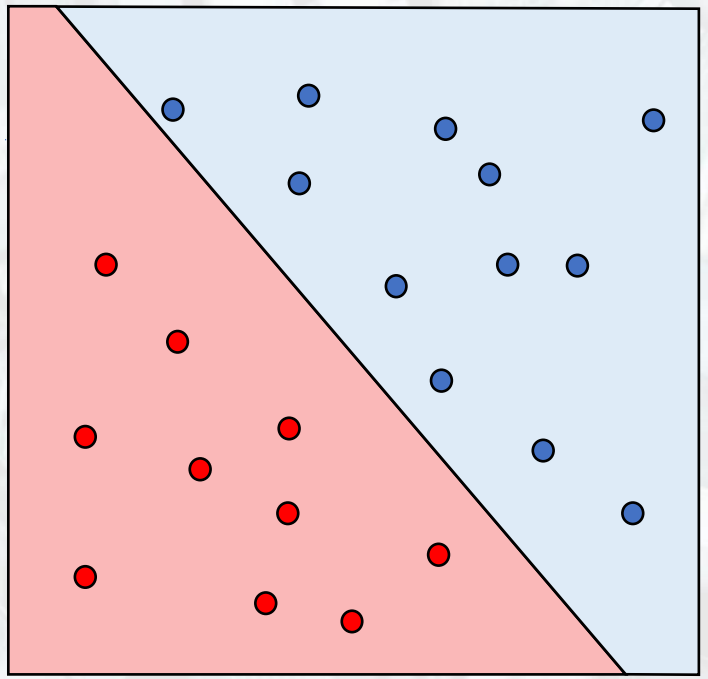


Image: Pinterest

Linear separable space

A bidimensional space consisting of 2 classes of points is linearly separable if there exists at least one straight line separating the space into its 2 distinct classes.

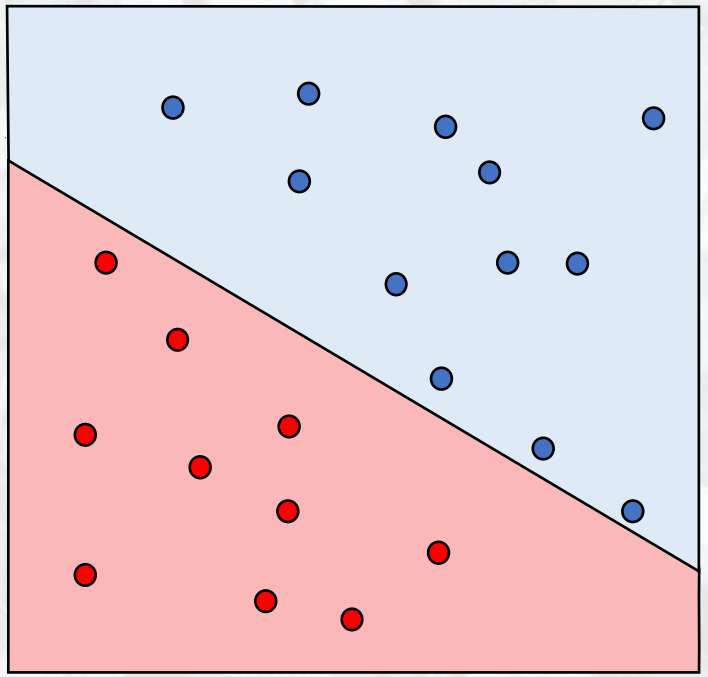
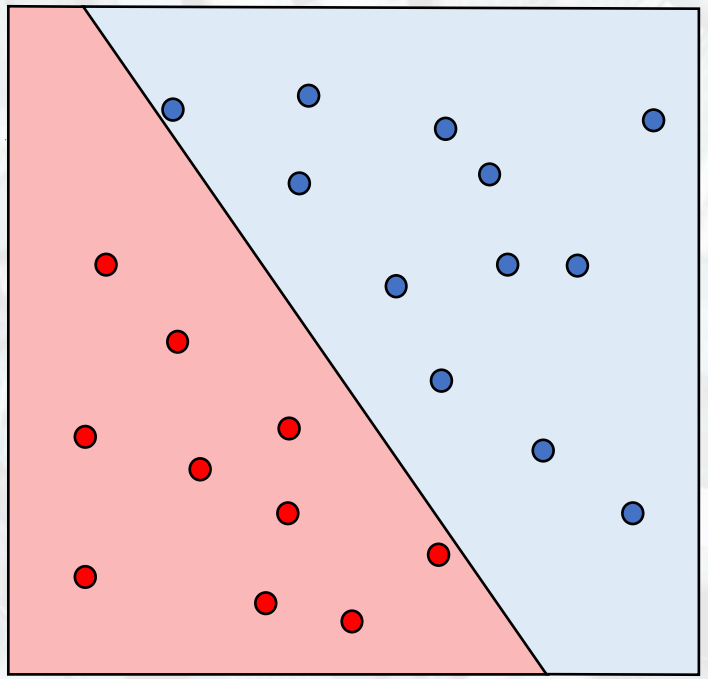


Image: Pinterest

Linear separable space

A bidimensional space consisting of 2 classes of points is linearly separable if there exists at least one straight line separating the space into its 2 distinct classes.



Non-linear separable space

What if the space is not linearly separable (e.g. XOR function)?

1. Combine several linear functions in cascade?
2. Apply non-linearities?
3. Combine several non-linear functions in cascade?

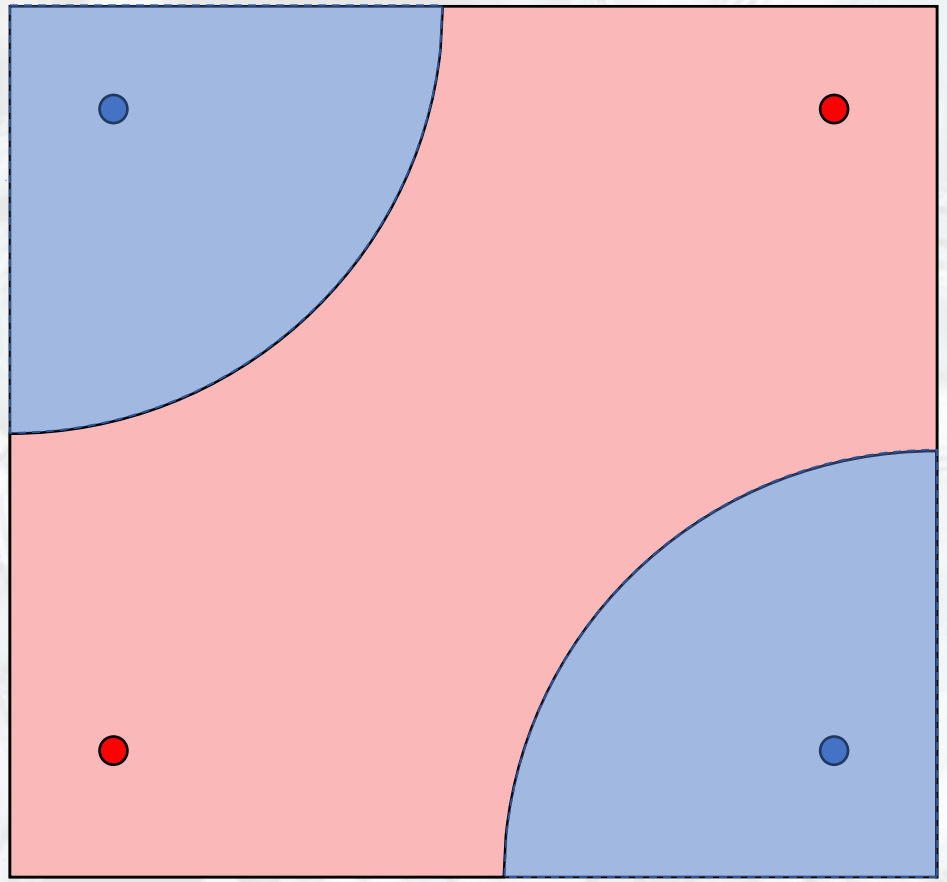
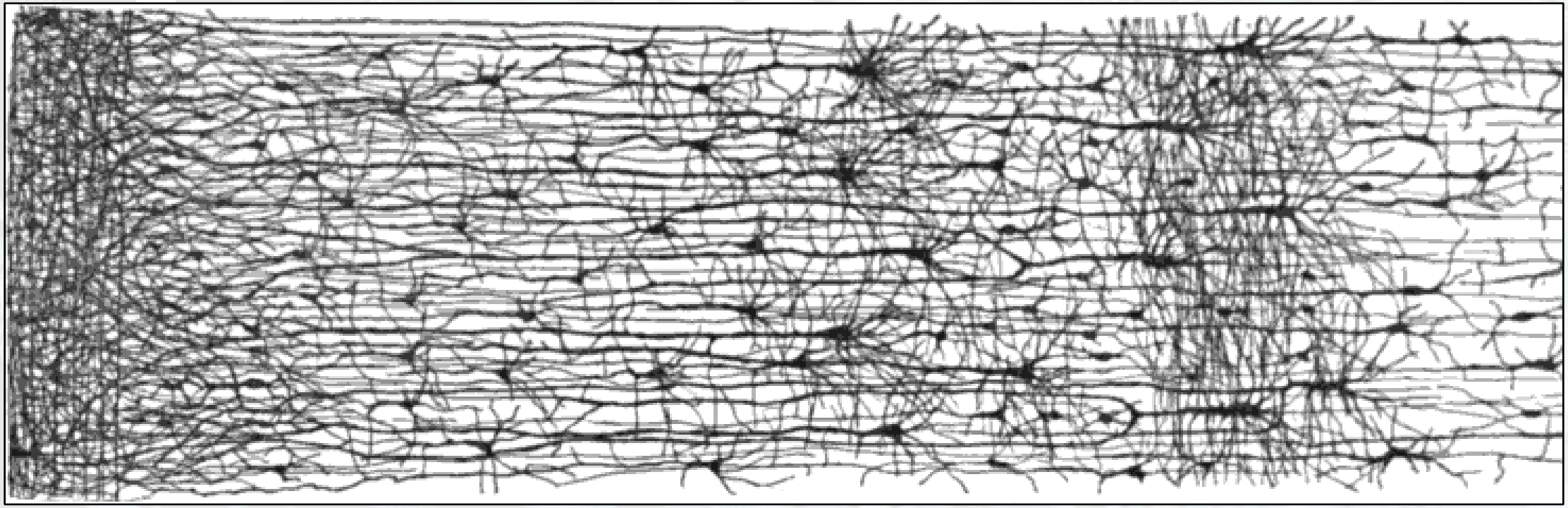


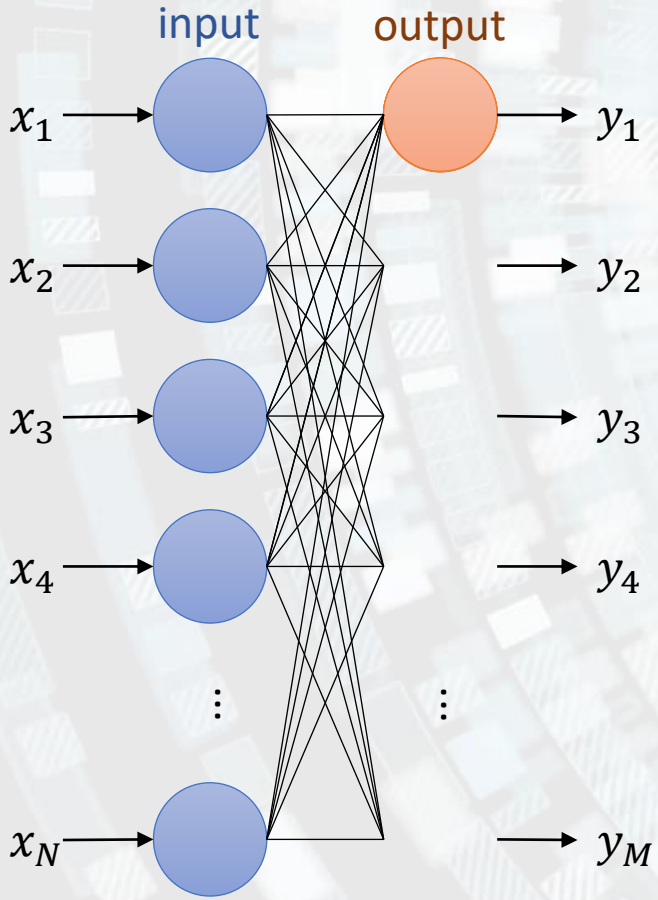
Image: Santiago Ramón y Cajal

The neuron network

The nervous system is formed of an entire network of neurons. Total number is estimated to be 86.1 ± 8.1 bn.



Single-layer neuron network

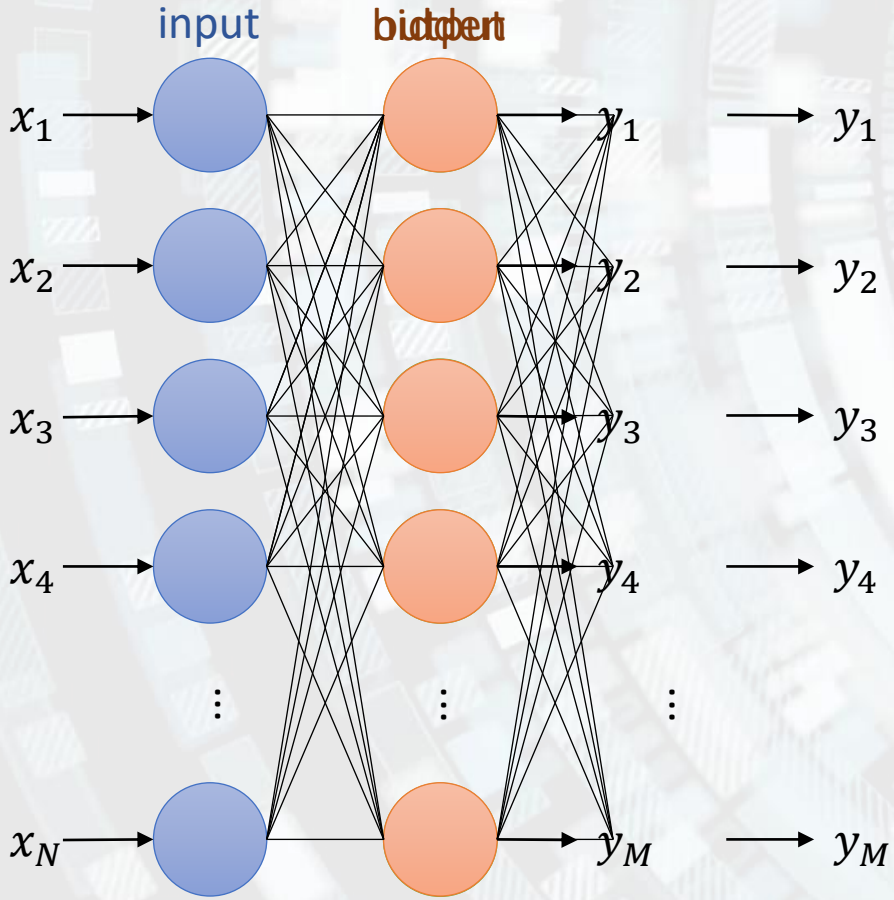


N can be (and usually is) different from M

Given a thresholding function, a single neuron can classify information in only 2 classes. Moreover, this is a "hard" decision.

A partial solution: a network formed of several neurons, placed in parallel, at the same level.
+ can be adapted to multi-class problems;
- still can't model non-linear separable spaces;
- every output is binary

Multi-layer neuron network



+ a multi-layer neuron network can classify more complex data, e.g. XOR function.
- every output is binary

N can be (and usually is) different from M

M2. The basic process of learning

The learning process

Machine learning = we say that a system „learns” from experience E regarding a set of tasks T and a performance measure P, if the performance in solving the tasks T, measured by P, grows with the experience E.

Goal: finding a function that associates an input dataset with their correct output, in the best way possible (smallest error; ideally – 0).

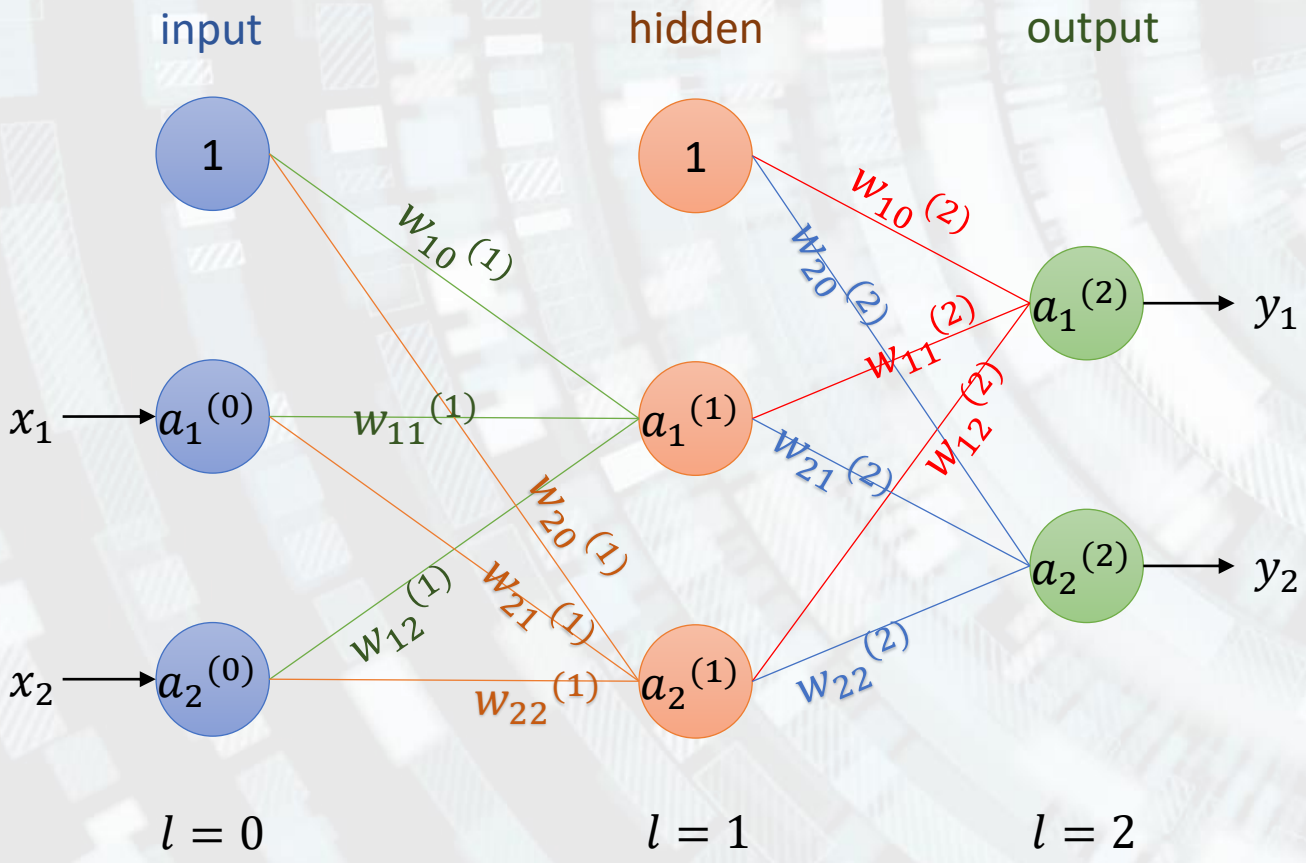
Consists of 2 steps:

1. Forward propagation;
2. Backpropagation.

Forward propagation

- **Goal**: obtaining the output $y = M(x)$;
- M = transfer function of the neural network, obtained by composition off all transfer functions associated to the network's layers.
- It involves passing the neural network from the input to the output.
- It breaks down to computing the activations for each neuron inside the network, from the lower layers towards the higher ones.

Forward propagation

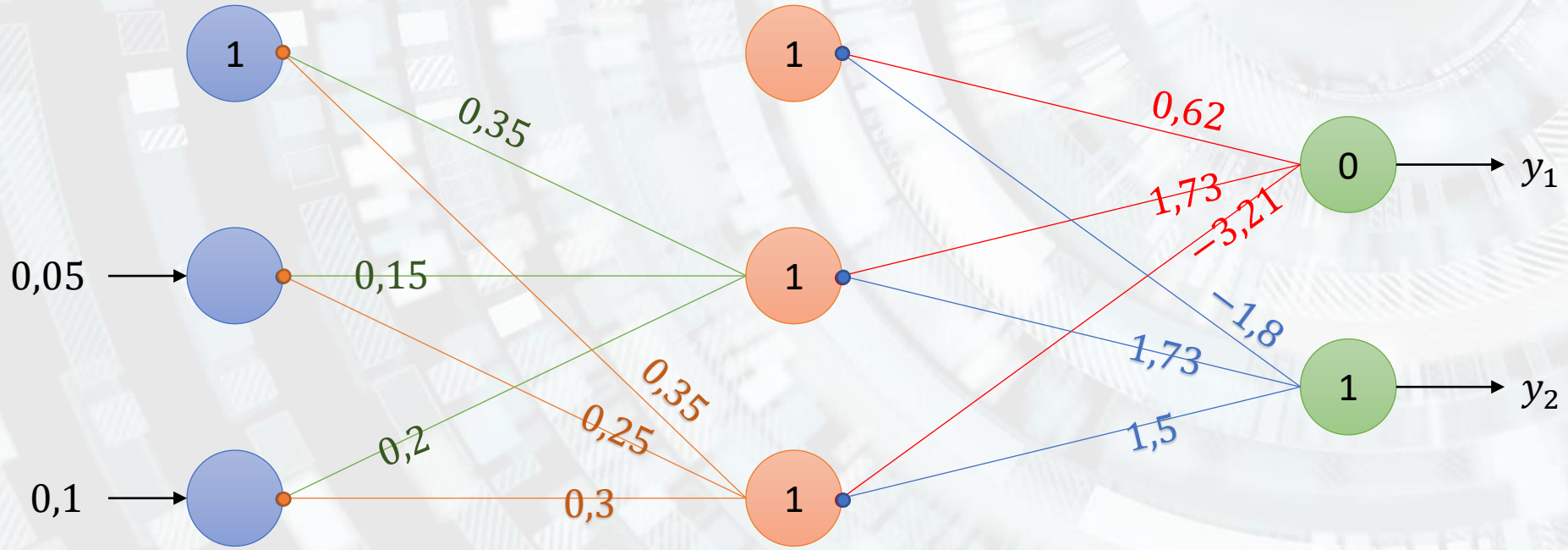


$$a_i^{(l)} = f\left(\sum_{j=0}^{n} w_{i,j}^{(l)} a_i^{(l-1)}\right)$$

$$a_0^{(l)} = 1$$

$$f(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases}$$

Forward propagation – numerical example



Backpropagation

- **Goal**: learning the internal representation of a neural network. It involves computing all weights (and biases) which bring the model's output tu a form as close as possible, ideally identical, to the desired/real output (groundtruth).

- How do we measure what it means to be „as close as possible"?
  - A: by using a cost function, e.g. mean square error.

- What rule do we use to adjust the weights?
  - A: gradient descent.

Image: AdobeStock

Gradient descent – intuition

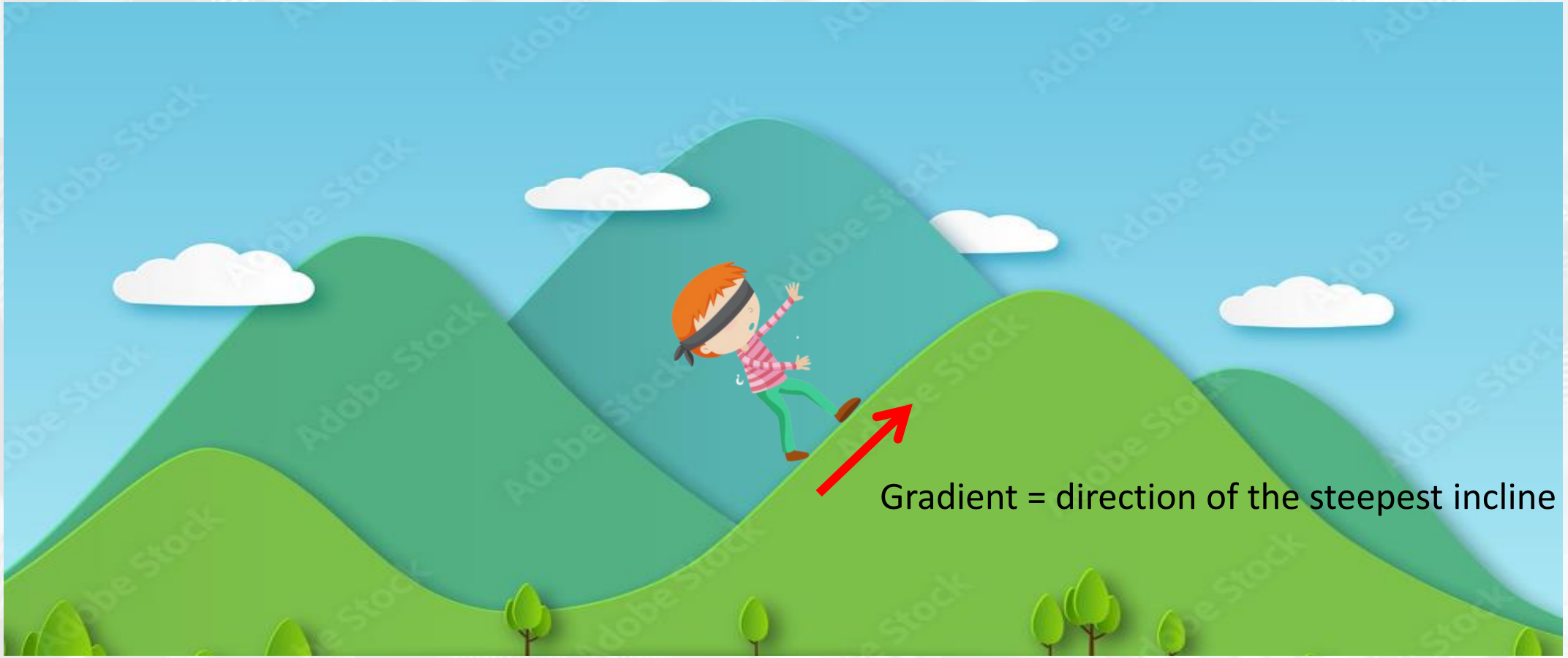


Image : AdobeStock

Gradient descent – intuition

Image : AdobeStock

Gradient descent – intuition

Image : AdobeStock

Gradient descent – intuition

Image : AdobeStock

Gradient descent – intuition

Image : AdobeStock

Gradient descent – intuition

Gradient descent

- Method to compute the gradient for a neural network's parameters.

$$y = f(\boldsymbol{w} \cdot \boldsymbol{x} + b) = f(\boldsymbol{w}^T \boldsymbol{x} + b)$$

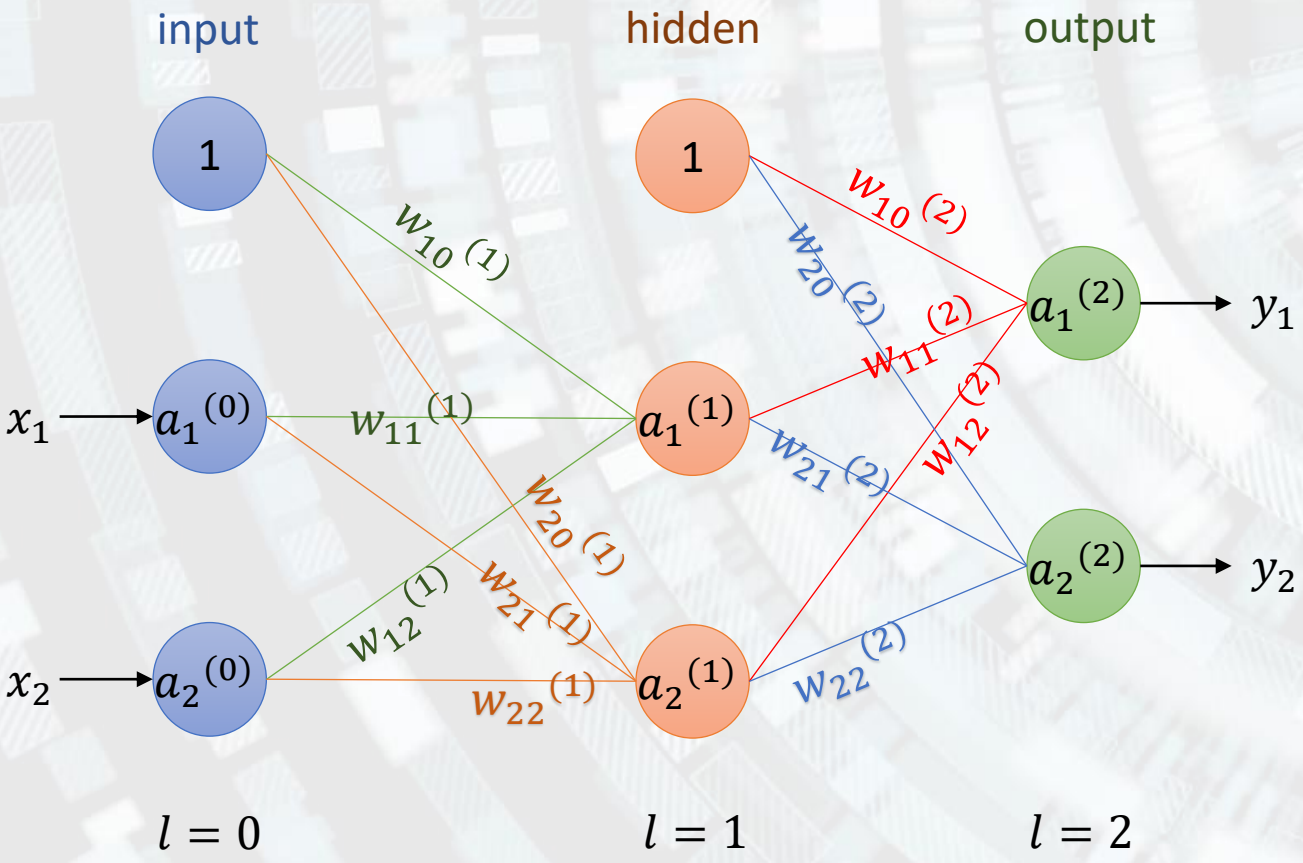
Funcție de pierdere (se calculează pe o singură pereche de eșantioane)

$$\mathcal{L}(y, \hat{y}) \overset{e.g.}{\Longrightarrow} \mathcal{L}(y, \hat{y}) = (y - \hat{y})^2$$

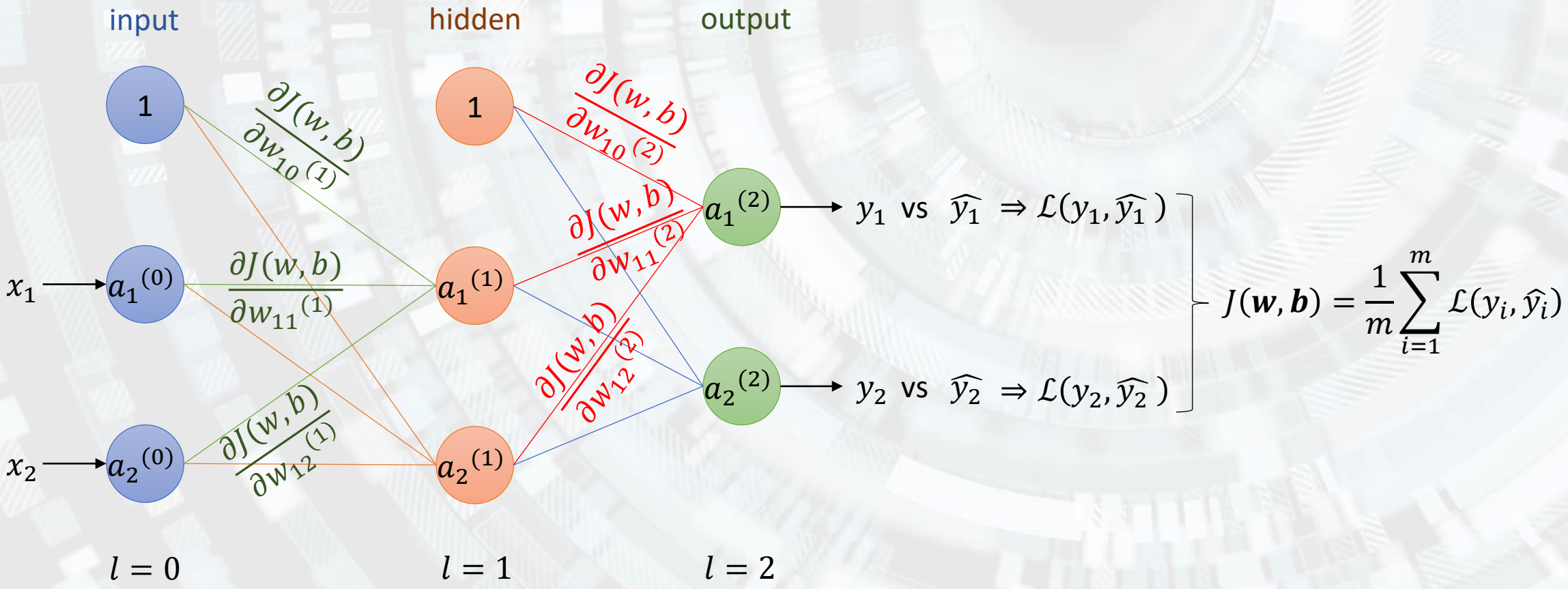
Funcție de cost (se calculează pe un set de perechi de eșantioane)

$$J(\boldsymbol{w}, \boldsymbol{b}) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(y_i, \hat{y}_i)$$

Gradient descent



Gradient descent



Gradient descent

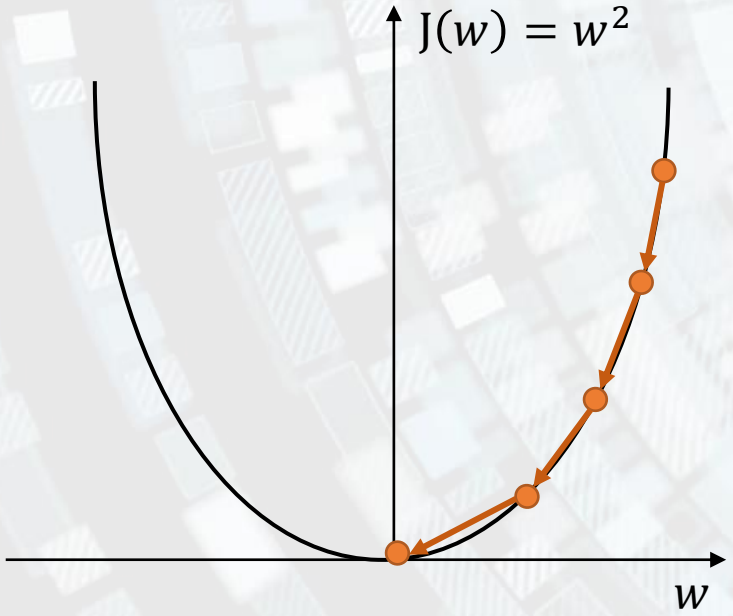
- Weights update follows this rule:

$$w^{(t+1)} = w^{(t)} - \alpha \frac{\partial J(w)}{\partial w^{(t)}}$$

Simplified writing: $$w := w - \alpha \frac{\partial J(w)}{\partial w}$$

Gradient descent

- Graphical example: consider the simplified case in which the cost function depends on a single variable and we choose MSE.

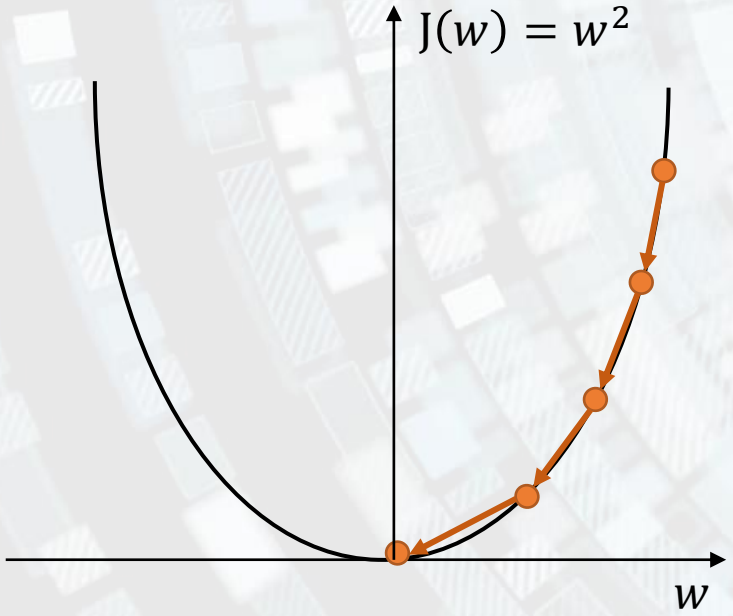


$$w := w - \alpha \frac{\partial J(w)}{\partial w}$$

Ideally, we find w s.t. $J(w) = 0$, but, in practice, this almost never happens.

Gradient descent

- Graphical example: consider the simplified case in which the cost function depends on a single variable and we choose MSE.

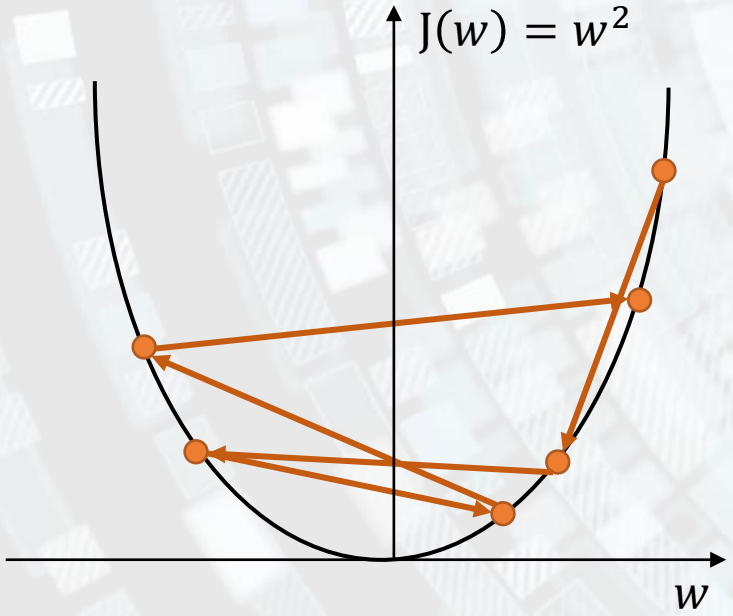


$$w := w - \alpha \frac{\partial J(w)}{\partial w}$$

- $\frac{\partial J(w)}{\partial w}$ indicates the direction of the steepest increase
- α indicates the amplitude of the modification = learning rate

Gradient descent – learning rate impact

➢Same example as before, but with a **high learning rate**.

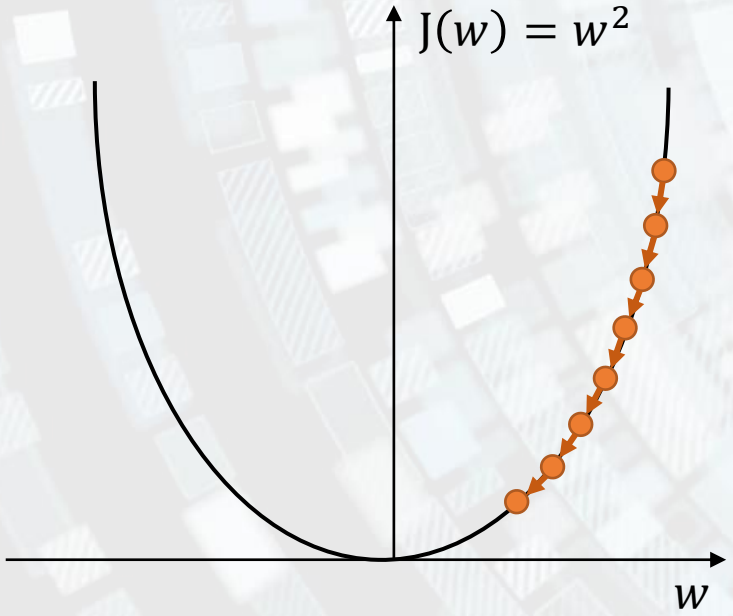


$$w := w - \alpha \frac{\partial J(w)}{\partial w}$$

- The algorithm may miss the optimum, because each iteration brings a high displacement.
+ Due to the large steps taken, the distance to optimum is travelled faster => faster training.

Gradient descent – learning rate impact

➢Same example as before, but with a **low learning rate**.

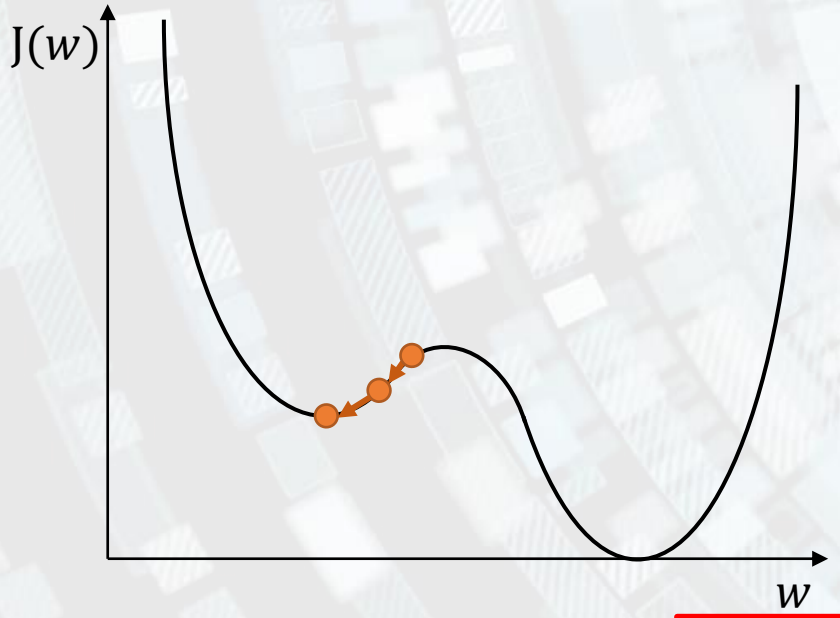


$$w := w - \alpha \frac{\partial J(w)}{\partial w}$$

- The algorithm performs small updates => slower training.
+ Low probability to miss the optimum dure to the small steps.
Why not always use small learning rate?

Gradient descent – learning rate impact

- Cost function that is not convex (local minimum is not necessarily global minimum), low learning rate.



$$w := w - \alpha \frac{\partial J(w)}{\partial w}$$

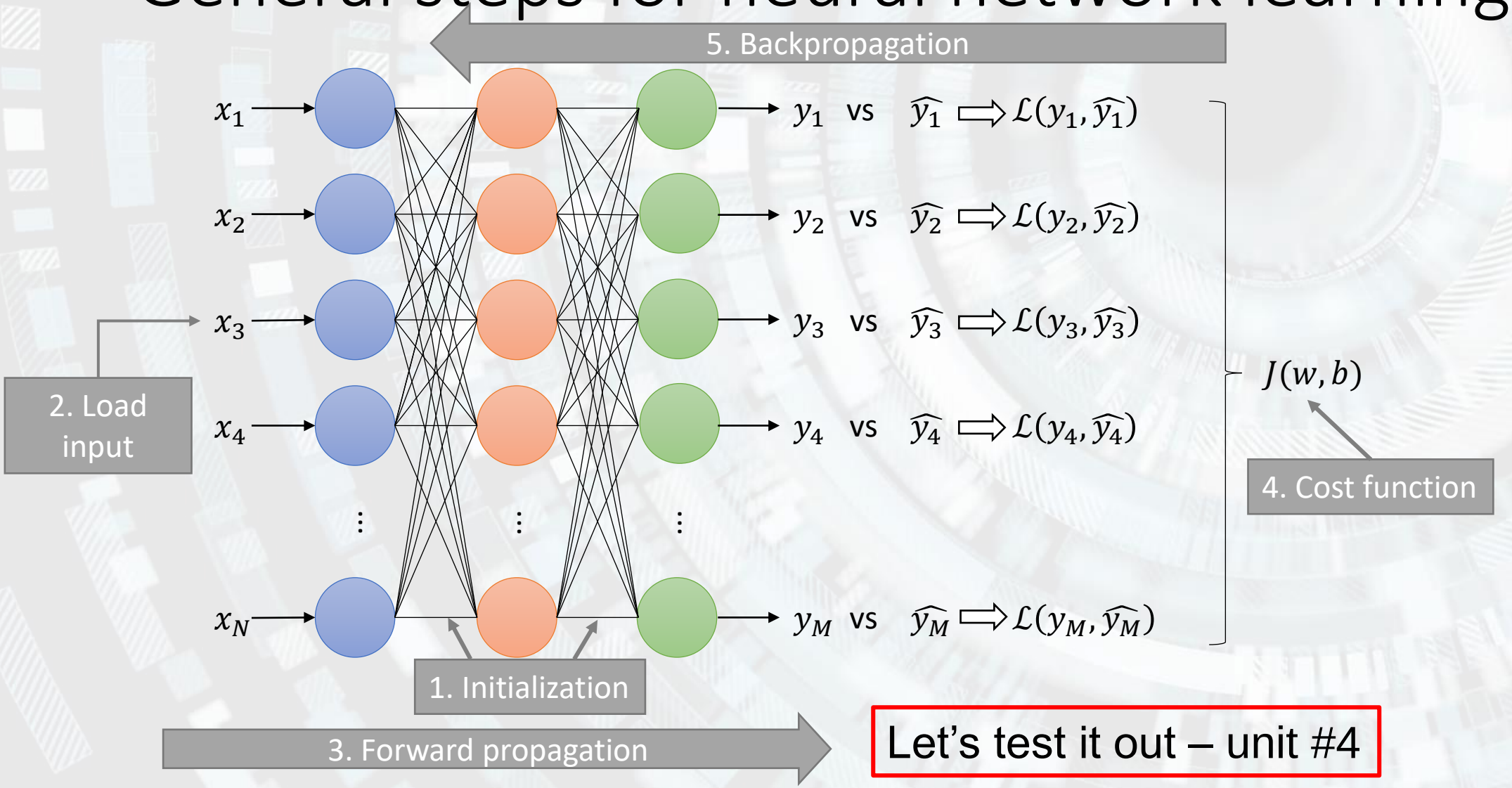
- There is a possibility that the algorithm gets stuck in a local minimum (saddle point)

Let's test it out – unit #3

General steps for neural network learning

1. Initialize network parameters
2. Load input data
3. Forward propagation
4. Compute cost function by comparing output data with groundtruth
5. Backpropagation + parameters update
6. Repeat steps 3-5 until convergence.

General steps for neural network learning



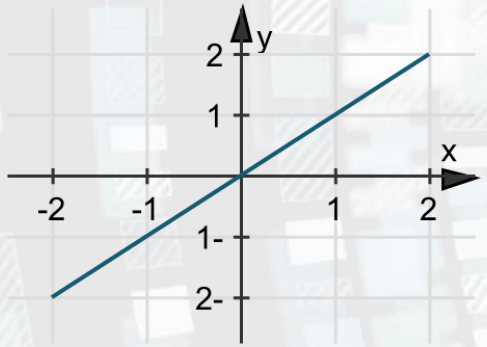
Let's test it out – unit #4

M3. Terminology

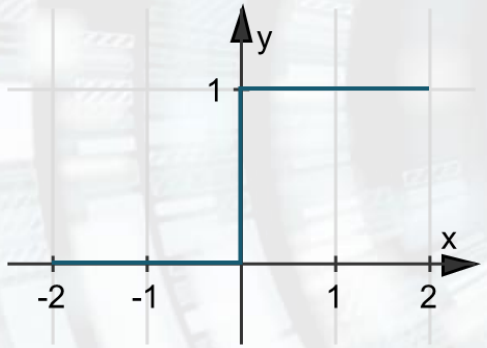
Terminology

1. Activation functions
2. Cost functions
3. Learning rate
4. Parameters vs hyper-parameters
5. Datasets
6. Optimizers
7. Regularizers
8. Popular layers

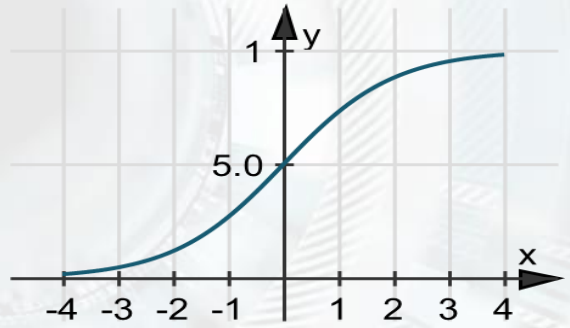
1. Activation functions



a) linear

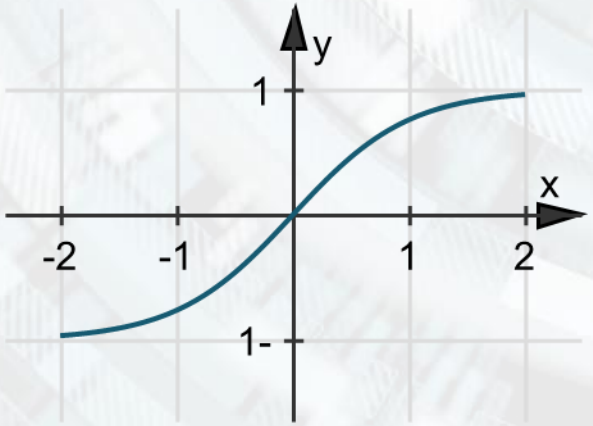


b) Heaviside



c) sigmoid

d) softplus



e) tanh

f) PReLU

2. Cost functions

Loss function = function computed in a **single point** which measures the penalty of taking a decision (distance between predicted and groundtruth value).

Type of

Cost function = loss function computed at the **entire dataset level**. Is more general than the loss function.

Part of

Objective function = function which needs to be optimized, either by minimizing (cost function), or by maximizing.

2. Cost functions – examples

Loss functions:

$$\mathcal{L}(y_i, \hat{y_i}) = (y_i - \hat{y_i})^2$$
$$\mathcal{L}(y_i) = \max(0, 1 - t_i y_i), t \in \{-1, 1\}, y = \boldsymbol{wx} + b$$

Cost functions:

$$MSE = \left[\frac{1}{N}\right] L2 = J(y_i, \hat{y_i}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2$$

$$L1 = J(y_i, \hat{y_i}) = \left[\frac{1}{N}\right]\sum_{i=1}^{N}|y_i - \hat{y_i}|$$

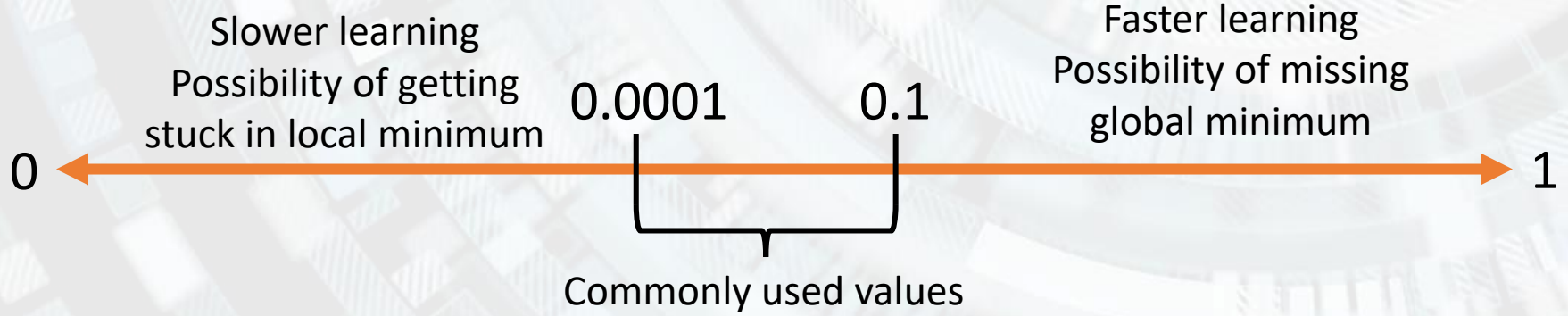
$$CE = -\sum_{c=1}^{M} \mathbb{1}_{[y \equiv \hat{y}]} \log(p(y \equiv \hat{y}))$$

Objective functions:

$$w = \arg\max_{w} \sum_{i=1}^{M} \log p_{model}(\hat{y_i} | x_i, w)$$

3. Learning rate

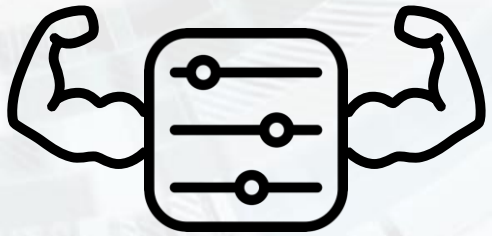
Learning rate = adjustable parameter which determines the step size when advancing towards minimizing the cost function, at each iteration. It represents the „speed” with which an algorithm learns.



4. Parameters vs hiper-parameters

Parameters = adjustable variables representing the internal state of the model. They are deducted from the training data through learning. Example: model weights.

Hyper-parameters = configurable variables, external to the model, influencing the learning process. They are set by the user. Example: learning rate, number of epochs, batch size etc.



5. Datasets

Dataset = a group of elements with common properties. It represents the „experience” that an algorithm makes use of when learning a certain task (according to the definition).

$$D = \{((x_i, y_i)|T), 1 \leq i \leq M\}$$

input output task dimension

5. Datasets

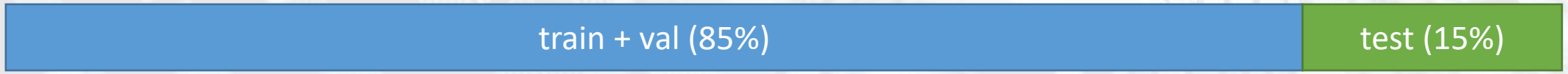
- They usually contain examples of input-output data. Their purpose is to help the model find an association law between input and output.

- A good dataset contains sufficient examples (10 x no. param.) and is uniformely distributed among its classes:
  - for regression: 10 examples/prediction variable
  - for classification: 1000 examples/class

5. Datasets – splits

- During model training, the training is done on a subset of the datasets, called „train”. Hyper-parameter optimization is done by looking at the results obtained when testing the model on another subset, called „val” ⇔ we optimize w.r.t. „val”.

- After finding the best set of hyper-parameters, the model is trained again on the „train+val” subset and its generalizing ability is checked by testing it on the „test” subset, unseen up until that point by the model. The reported performance is the one obtained on „test”.



5. Datasets – browsing

Iteration = the set of processes done between two successive weight updates during training.

Batch = group of instances from the dataset which can be run by the model at the same time, in parallel.

Batch size = number of instances inside a batch.

Epoch = a term which signals a complete pass of the entire dataset through the learning algorithm.

6. Optimizers

Optimizer = algorithm used for optimizing (minimizing or maximizing) an objective function.

- (Batch, Stochastic, Mini-batch) Gradient Descent
- Momentum
- Learning rate scheduling
- Adagrad
- RMSProp
- Adam

6. Optimizers

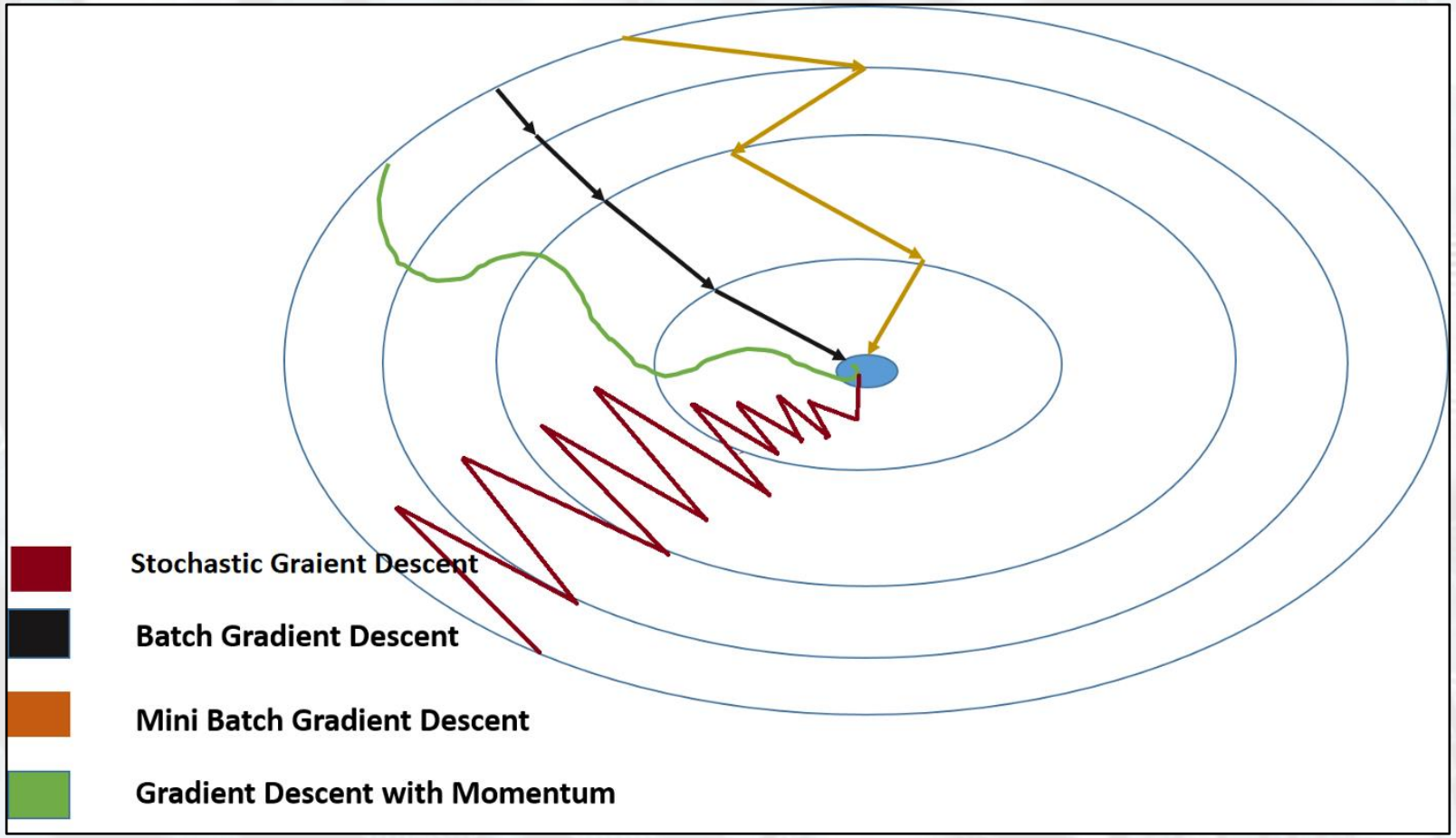
- Batch gradient descent – loads the entire dataset, runs forward propagation and updates the weights accordingly => best model approximation, requires storing the dataset in memory.

- Stochastic gradient descent – loads one sample, runs forward propagation, updates the weights, repeats for next sample => best sample approximation, poor model approximation, loads a single sample at a time.

- Mini-batch gradient descent – loads a batch of samples, runs forward propagation, updates the weights, repeats for next batch => best compromise between memory and model approximation.

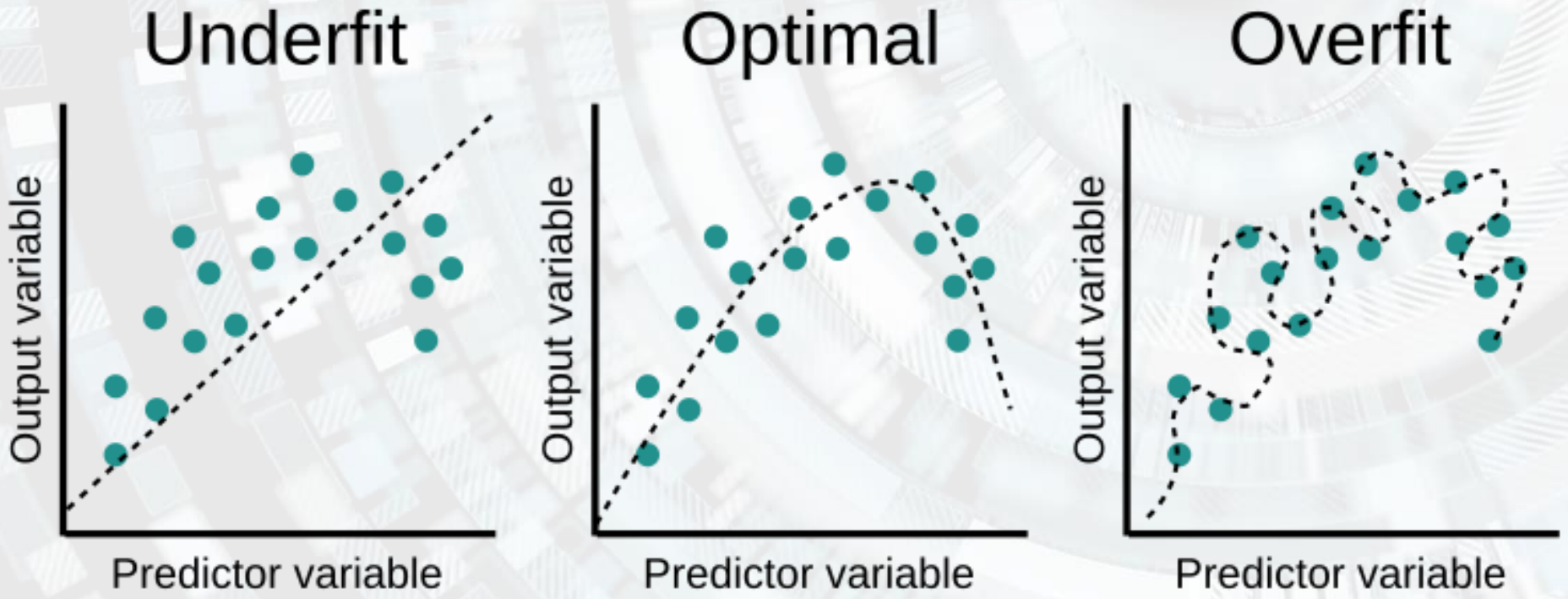
Image: Bouzouitina Hamdi

6. Optimizers – comparison



Sursă imagine: educative.io

7. Regularization

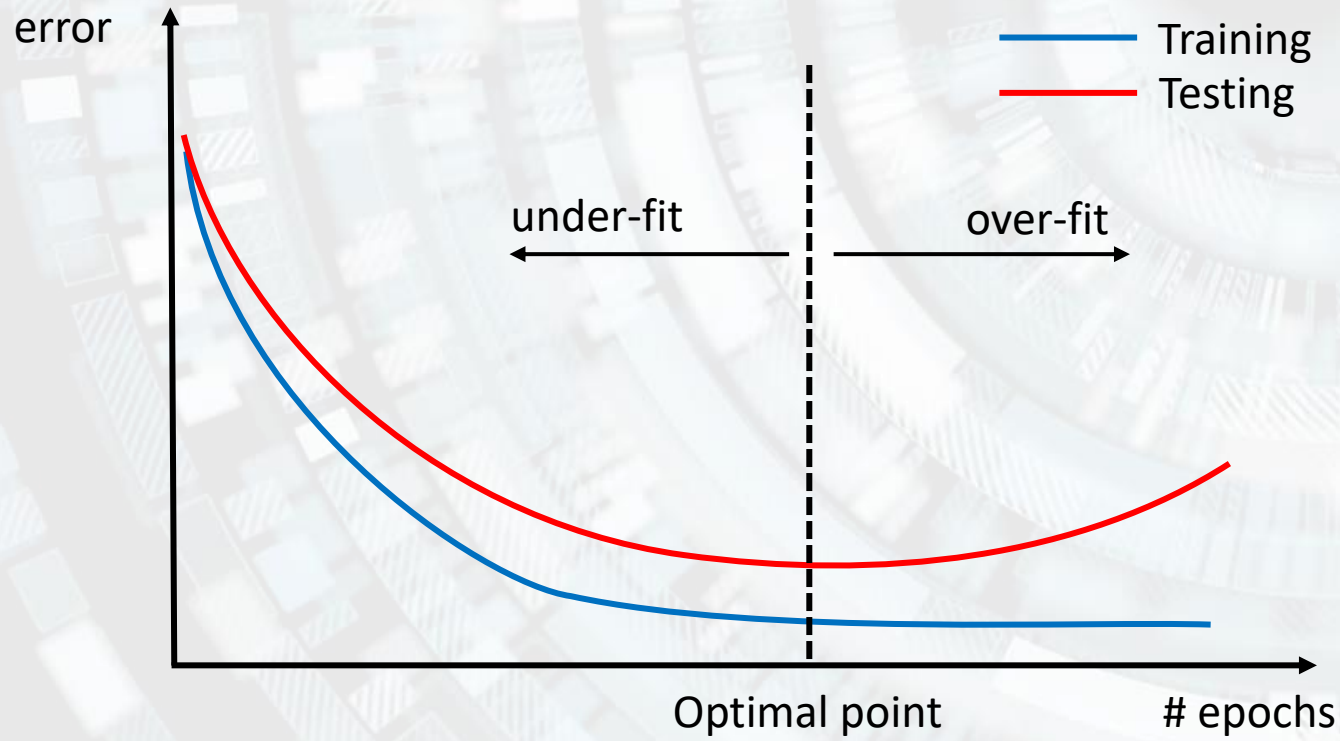


7. Regularization

- Model learning has 2 phases:
  1. Training the model on the training dataset – optimization;
  2. Testing the predictive capacity on the (unseen) test dataset - generalization.

- A high performant model is characterized by 2 aspects:
  1. Low training error
  2. Small difference between training and testing errors.

7. Regularization



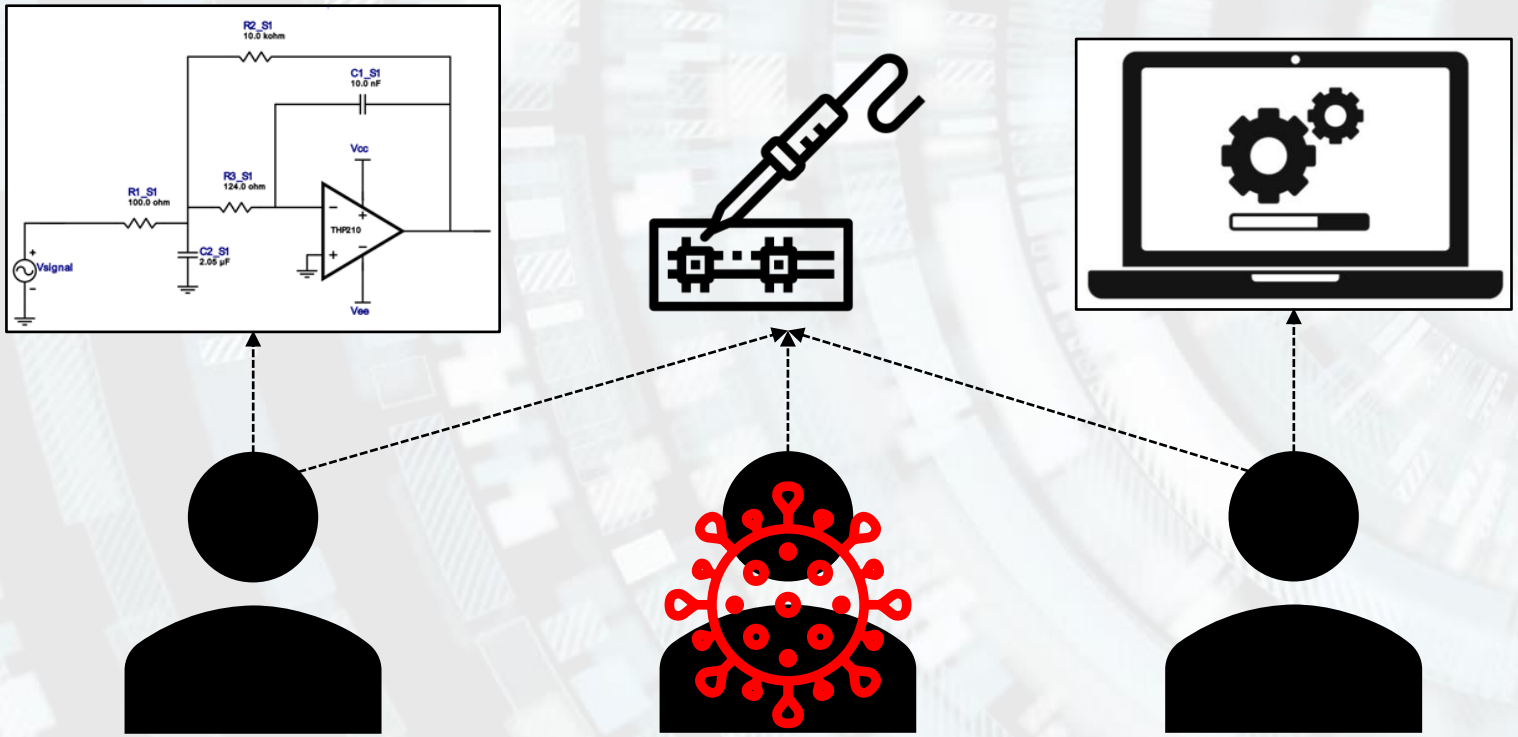
7. Regularization

Regularization = modifying the learning algorithm such that it increases the generalization capacity and prevents overfitting.

- Types of regularization:
  1. L1, L2;
  2. Dropout;
  3. Early stopping;
  4. Dataset augmentation.

Images:hardwarebee, flaticon, vectorstock

7. Regularization – dropout



Scenario – building a robot:

- Team of 3 students;
- Each students specializes in a different field;
- One of the students cannot work at the project anymore;
- The other 2 students must take over his task;
- The knowledge of the 2 remaining students increases.

8. Popular layers

Layer = fundamental topological component composed of nodes, which is part of a neural network. Layers take information from previous layers and transmit it to the next ones. Chaining several layers forms a neural network.

- Input
- Dense
- Convolutional
- Pooling
- Recurrent
- Activations
- Dropout
- Batch normalization
- Output

8. Popular layers

- Input layer:
  - First layer of any network;
  - We load the input data in this layer;
  - Can have more dimensions: $num \times width \times height \times channel$.

- Dense / fully-connected layer:
  - Each input neuron is connected to each output neuron;
  - One of the most commonly used layers, especially at the end of a network;
  - Described in the multi-layer network.

8. Popular layers

- Convolutional layer:
  - Performs „convolution” between an input signal and a set of features, named kernel/filter;
  - Traditionally, it is applied on 2D data, but it also has support for 1D and 3D;
  - It is the main layer for most computer vision applications;
  - Its purpose is to extract relevan features and reduce input dimension;
  - Special cases: dilated convolutions, 1×1 convolutions.
  - Convolutions are invariant to translations

8. Popular layers – convolutional layer

padding=1

0	0	0	0	0	0	0
0	2	1	0	1	1	0
0	1	1	0	2	0	0
0	0	0	1	1	1	0
0	1	2	2	0	2	0
0	2	0	1	2	1	0
0	0	0	0	0	0	0

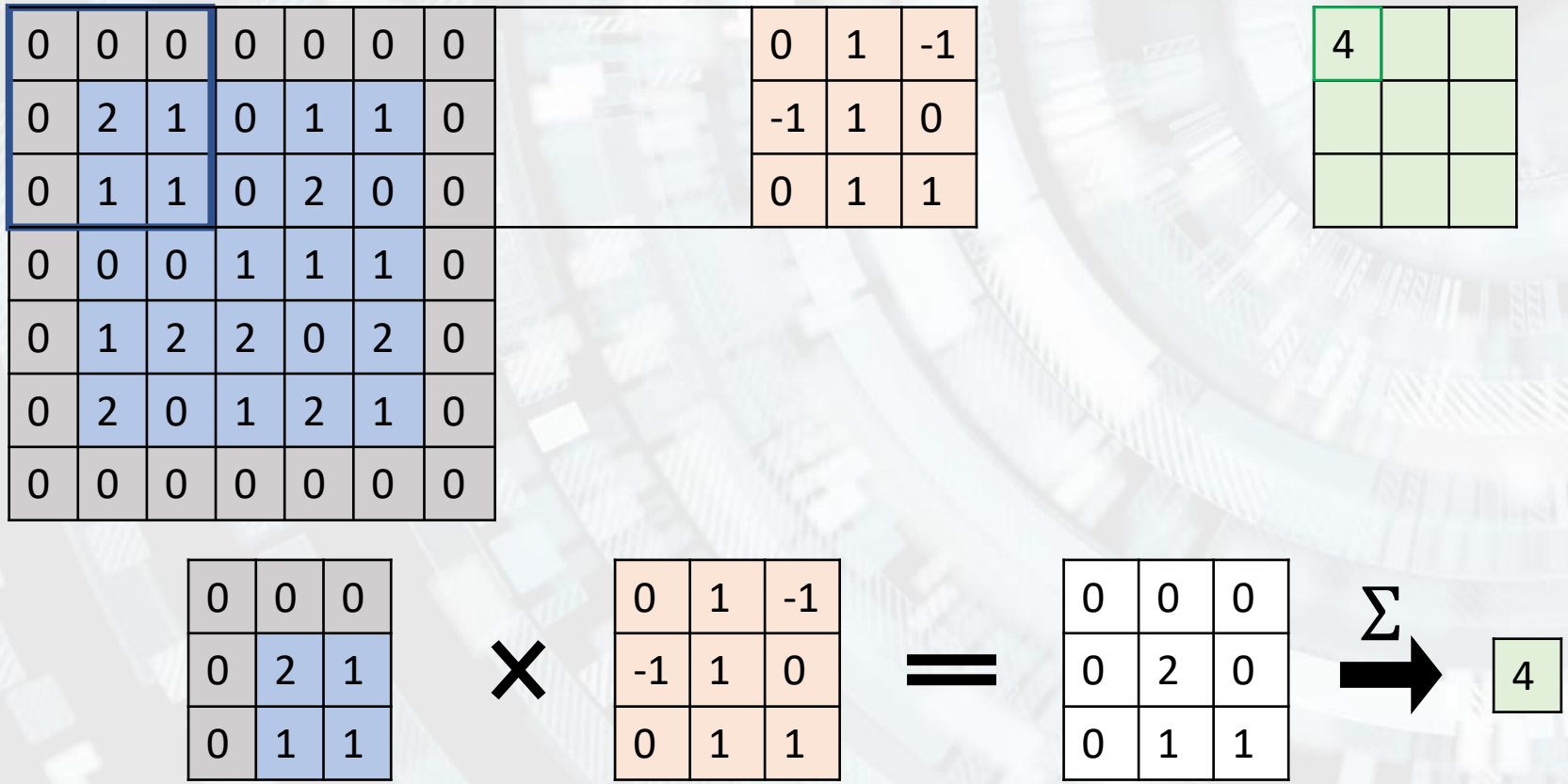
Image features
(image)

0	1	-1
-1	1	0
0	1	1

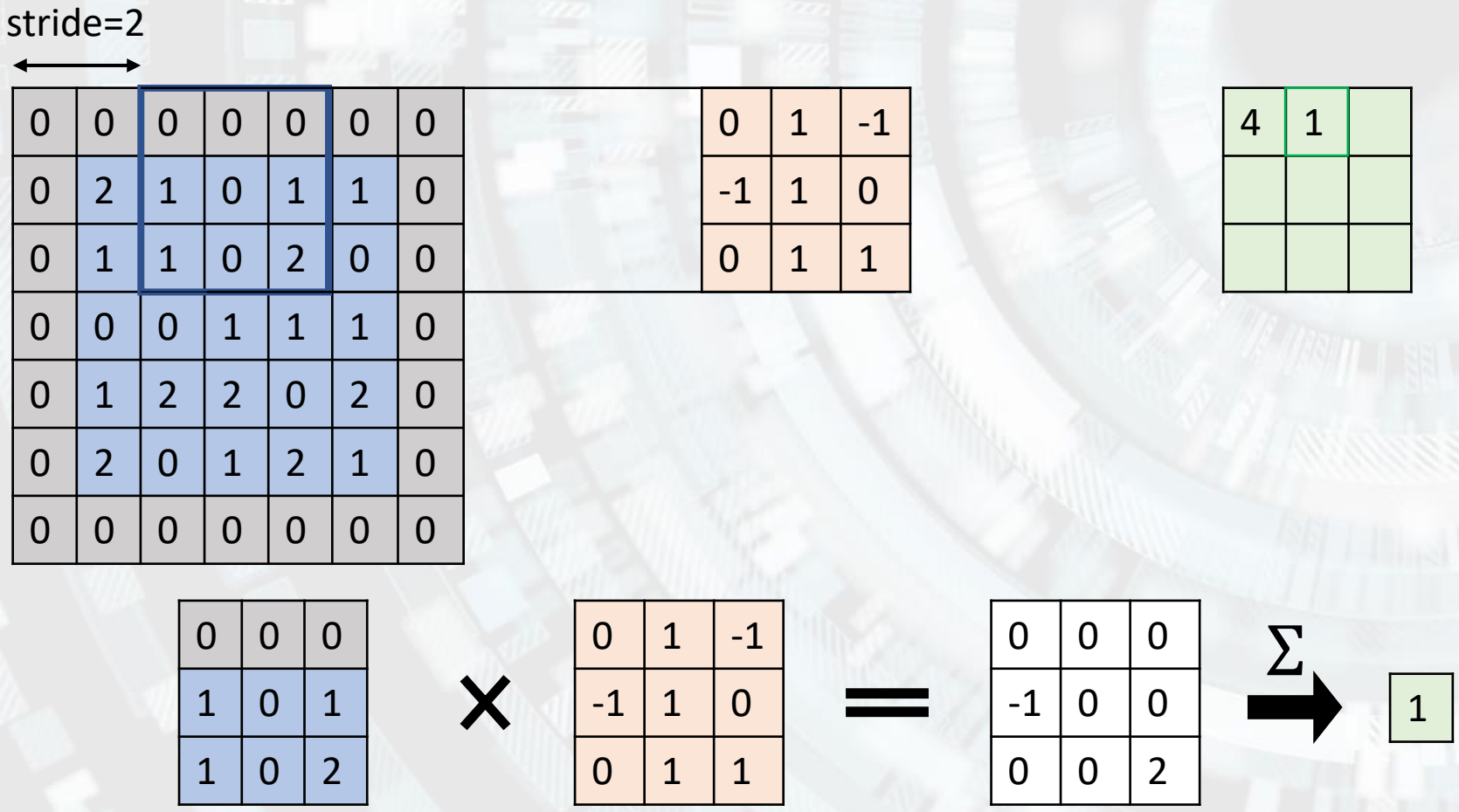
Filter
(kernel)

Trăsături de ieșire

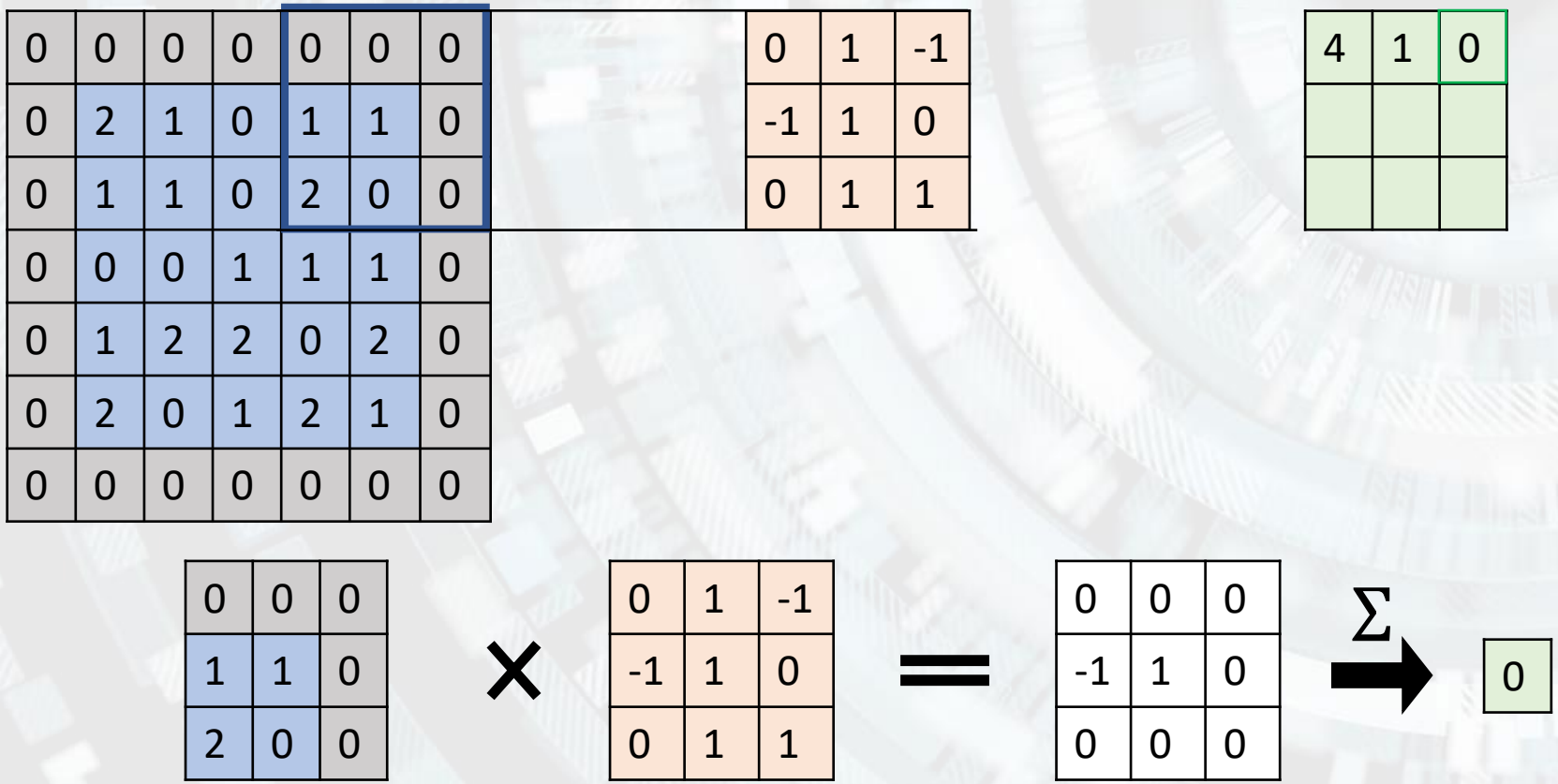
8. Popular layers – convolutional layer



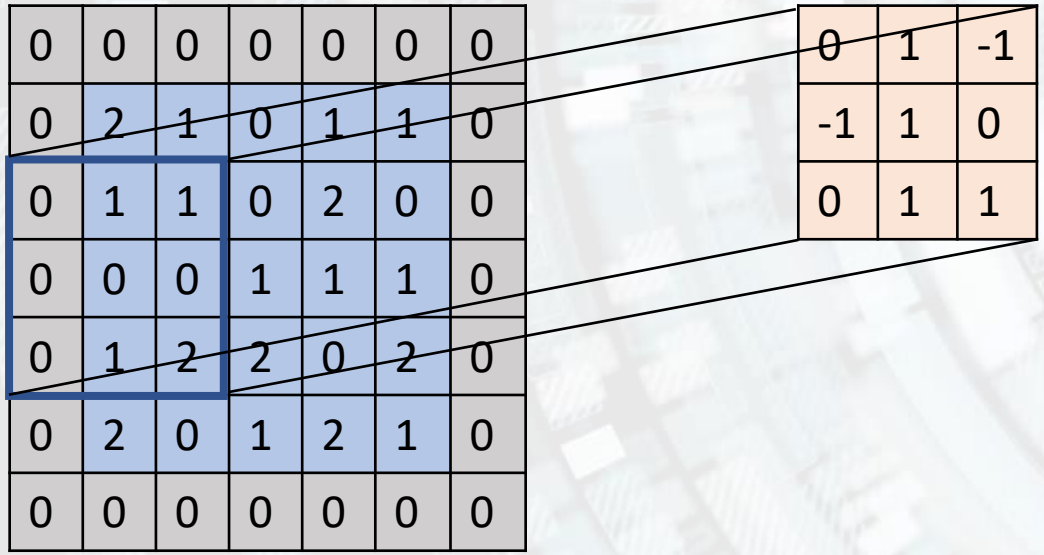
8. Popular layers – convolutional layer

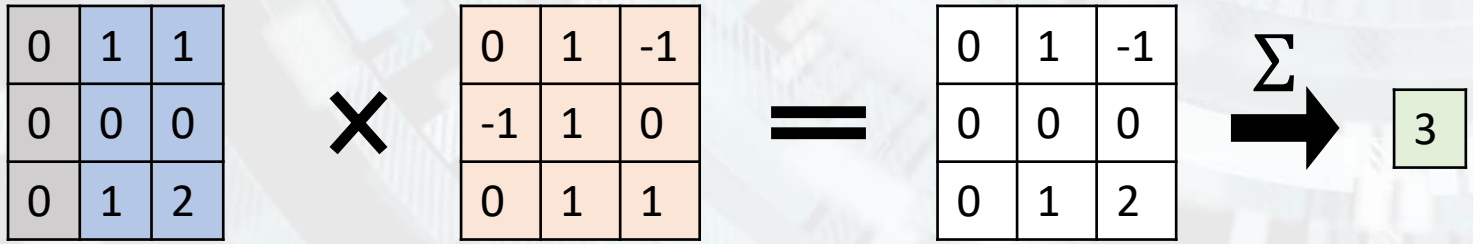


8. Popular layers – convolutional layer

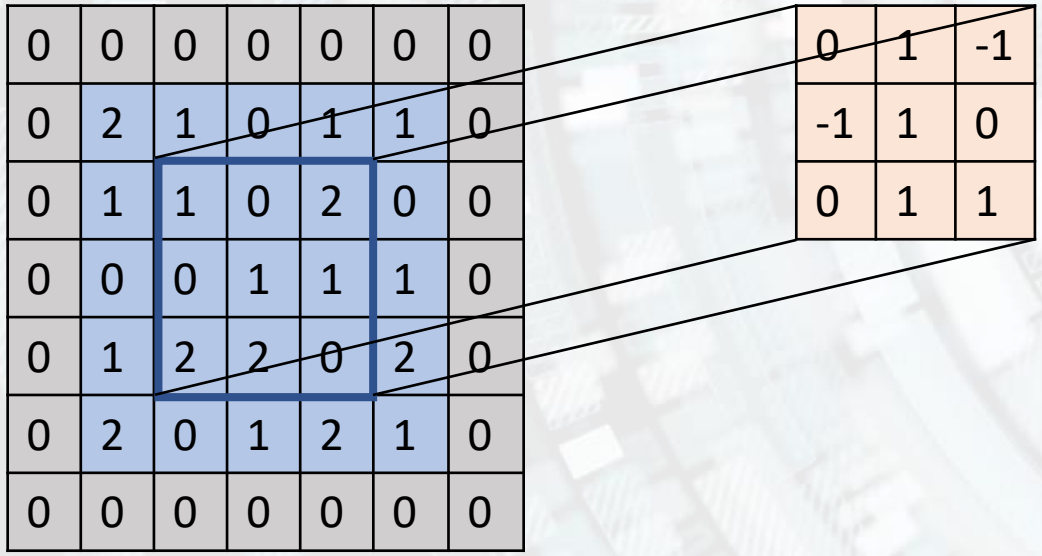


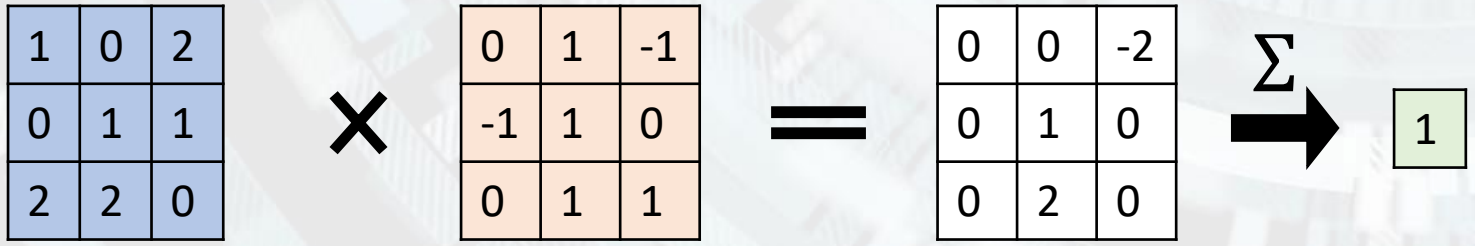
8. Popular layers – convolutional layer



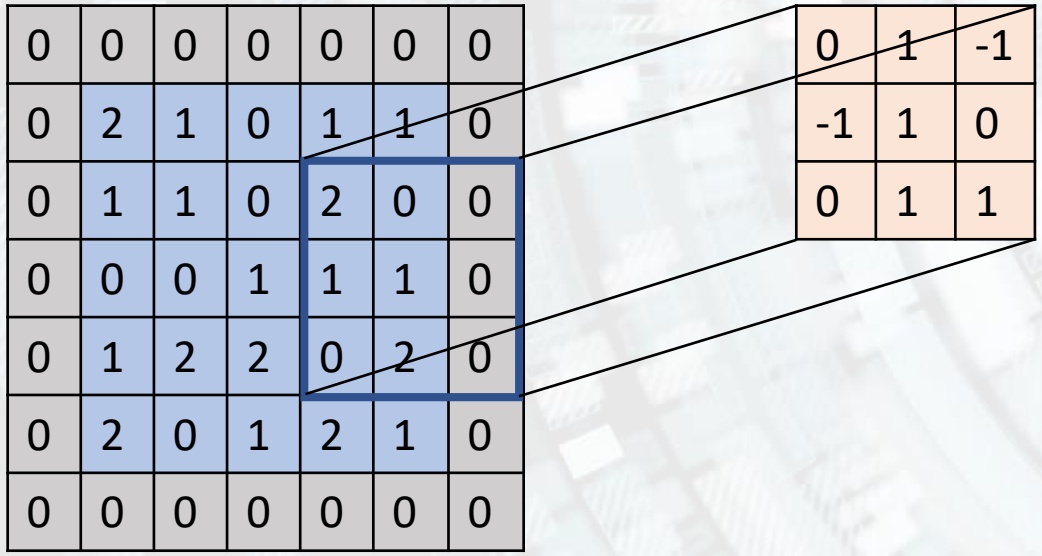


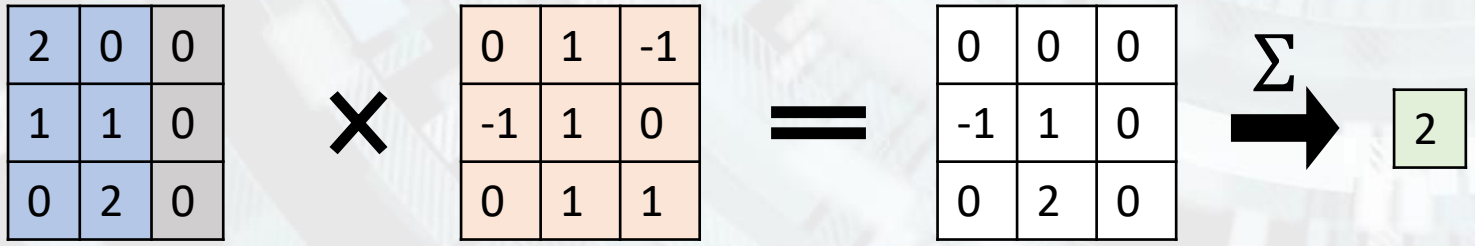
8. Popular layers – convolutional layer



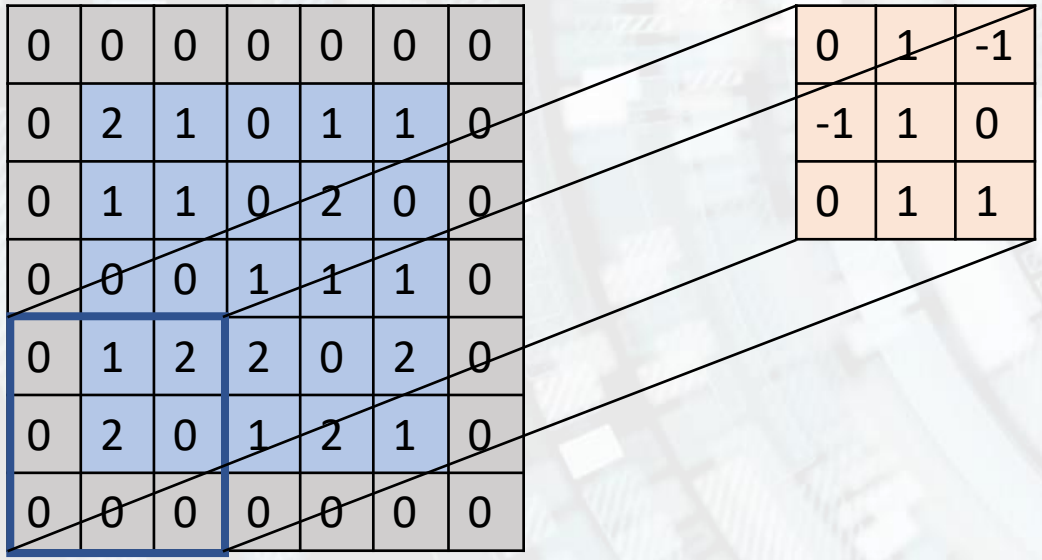


8. Popular layers – convolutional layer





8. Popular layers – convolutional layer



4	1	0
3	1	2
1		

0	1	2
0	2	0
0	0	0

×

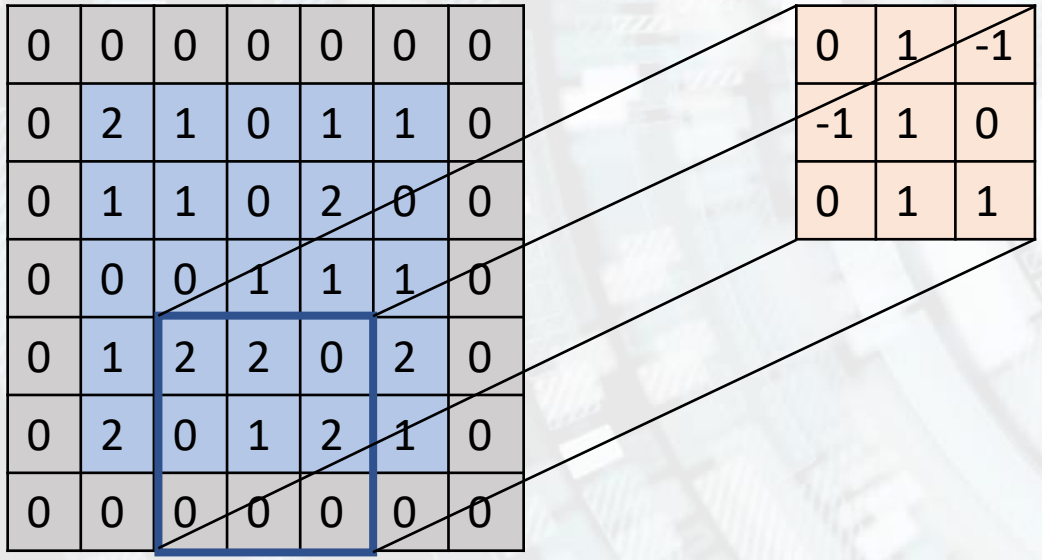
0	1	-1
-1	1	0
0	1	1

=

0	1	-2
0	2	0
0	0	0

Σ → 1

8. Popular layers – convolutional layer



4	1	0
3	1	2
1	3	

2	2	0
0	1	2
0	0	0

×

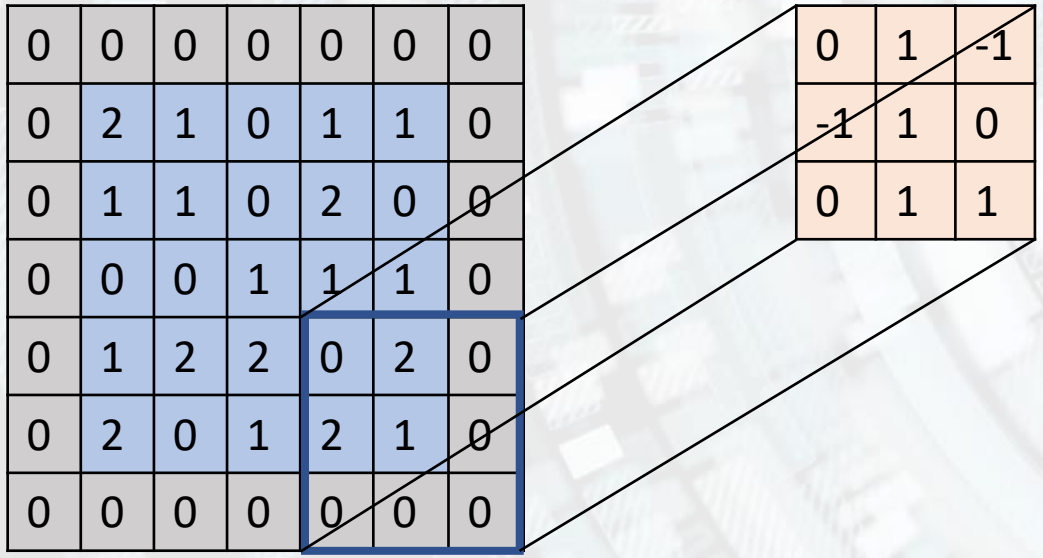
0	1	-1
-1	1	0
0	1	1

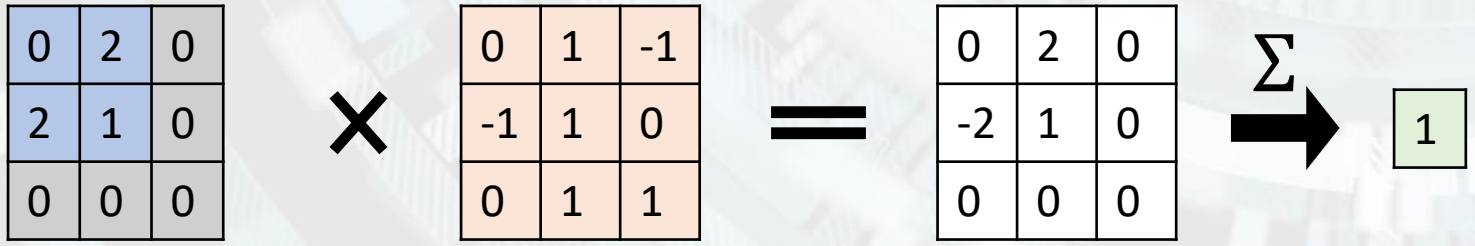
=

0	2	0
0	1	0
0	0	0

Σ → 3

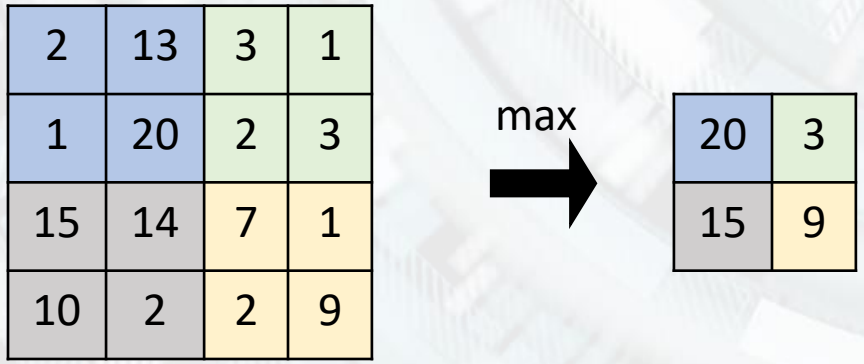
8. Popular layers – convolutional layer





8. Popular layers

- Pooling layer
  - Can be max/average pooling – similar concept.
  - Hyper-parameters similar to conv layer: filter size - F, stride - S, padding - P



M4. Practical considerations

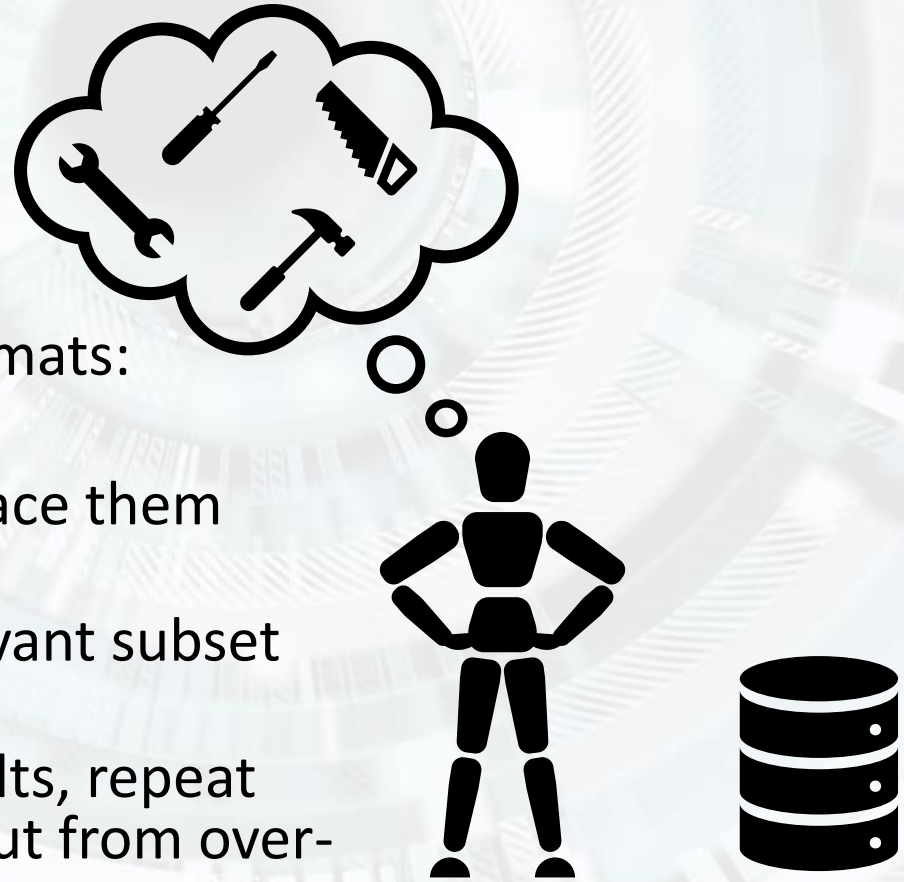
Steps in designing a neural network model

1. Choosing data
   - Number of examples in the dataset:
     - General: 10x no. of model parameters
     - Regression: 10 examples / prediction variable
     - Classification: 1000 examples / class
   - Sample quality – labeling done by experts, reduced noise
   - Dataset resources:
     - https://archive.ics.uci.edu/ml/index.php
     - https://www.kaggle.com/datasets
     - https://paperswithcode.com/datasets
     - https://datasetsearch.research.google.com/

Steps in designing a neural network model

2. Data pre-processing
   - Dataset split: train (70%), val (15%), test (15%)
   - All subsets contain similar data
   - Data formatting, switching between various formats:

   .csv <-> .pkl <-> .json <-> database;
   - Missing data (Null, NaN) – what should we replace them with?
   - Dataset is too large – use just a statistically relevant subset for training
   - Class imbalance – different weights for the results, repeat input from under-represented class, reduce input from over-represented class, etc.
   - Normalization – bring features in the same range of values



Steps in designing a neural network model

3. Model training
   - Initialize weights
   - Load data
   - Forward propagation
   - Compute cost function
   - Backpropagation
   - Update weights
   - Rinse & repeat

Steps in designing a neural network model

4. Model validation
   - Test the model on the validation subset to test its generalizing ability
   - This validation is not run at each iteration, but at a multiple of it (N=30, 100 etc.).

5. Model optimization
   - Hyperparameter tuning: adjusting model hyperparameters
   - Solve over-fitting with regularization methods: L1, L2, dropout, early stopping, dataset augmentation, etc.
   - Pre-train + fine-tune
   - Feature transfer
   - At the end of the optimization process we run the model one last time, on the test set. The metrics obtained on this subset are the officially reported ones.

Let's test it out – unit #5