Introduction to Deep Learning

Pierre Colombo

pierre.colombo@centralesupelec.fr

MICS
CentraleSupélec Université Paris-Saclay

Agenda - 3h Class

I. Theoretical Class

1. Introduction

2. Deep Learning Architecture

3. Performing the learning phase

4. From theory to practice

II. Deep Learning in action

Curriculum Vitae

Education & Diploma

2018. Diplôme D'Ingénieur, Supelec, *France*

MSc in Computer Science, Ecole Polytechnique Fédérale de Lausanne, *Switzerland*

2021. PhD in Computer Science, Telecom Paris, Institut Polytechnique de Paris, *France*

Title: Learning to represent and generate text using information measures

2022 . *Post-Doctoral Researcher*, CentraleSupelec, Laboratoire des Signaux et Systèmes, CNRS, Université Paris-Saclay, CentraleSupelec

2022 - Present. *Invited Member (external) of Comète Inria Team*, Laboratoire d'informatique de l'École polytechnique, Ecole polytechnique

Present Occupation

2022 - Present. Assistant Prof au MICS (CentraleSupelec)

Past Research Experiences

2016 (3 months). *Research Intern*, Procter & Gamble, ***Kronberg, Germany***

2016 - 2017 (6 months). *Lab Assistant*, EPFL Laboratory for Quantum Gases, ***Lausanne, Switzerland***

Developing various sensors algorithms (Python/C++) for a quantum experiment on fermi gas.

2017 (6 months). *Research Intern*, Swisscom Digital Lab, **Lausanne, Switzerland**

Research on NLP: Information Retrieval for a product oriented chatbot

2017 - 2018 (6 months). *Research Intern*, IBM Research, ***Zürich, Switzerland***

Research on deep learning for cancer treatment

2018 (6 months). *Lab Associate*, Disney Research, ***Los Angeles, USA***

Research on Natural Language Generation, 1 workshop paper, 1 paper accepted at NAACL, 1 US Patent

2018 - 2021 (36 months). *PhD Student (CIFRE)*, Telecom Paris (S2A) & IBM France, ***Paris, France***

11 first author papers published in A conferences (A*) , 4 papers accepted as oral presentation (top 5%), 1 best student paper award

What Is Deep Learning?

What Is Deep Learning?

Artificial Intelligence **The set of technics that can be used to mimic human behavior**

What Is Deep Learning?

Artificial Intelligence — **The set of technics that can be used to mimic human behavior**

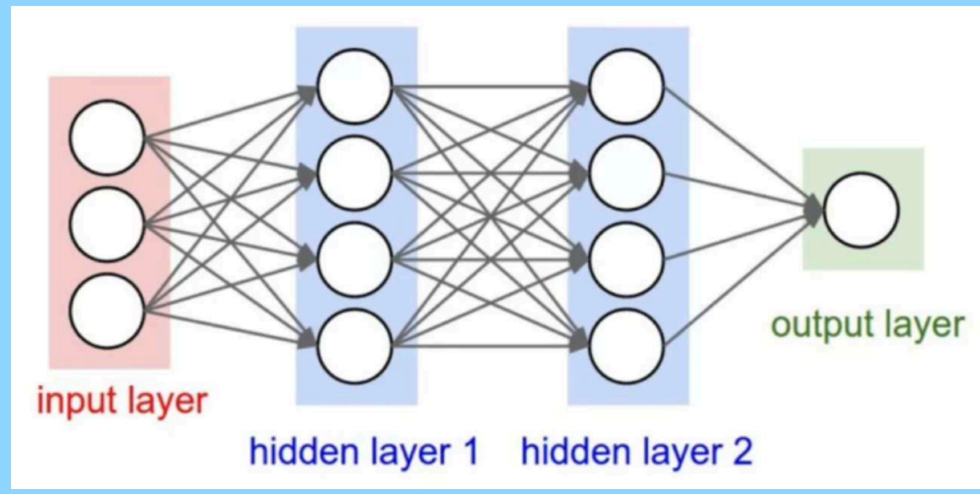
Machine Learning — **The set of technics that can « learn » from examples without being explicitly programmed**

What Is Deep Learning?

Artificial Intelligence — **The set of technics that can be used to mimic human behavior**

Machine Learning — **The set of technics that can « learn » from examples without being explicitly programmed**

Deep Learning — **The set of technics that can « learn » from examples using deep neural networks**



What are the goals of this class?

What are the goals of this class?

What to expect?

What are the goals of this class?

What to expect?

1. Have a high level view of how neural networks works

2. Getting familiar with general concepts such as overfitting, regularization,….

3. Code and train your first neural network

What are the goals of this class?

What to expect?

1. Have a **high level view** of how neural networks works

2. Getting **familiar with general concepts** such as overfitting, regularization,….

3. **Code and train** your first neural network

What not to expect?

What are the goals of this class?

What to expect?

1. Have a high level view of how neural networks works

2. Getting familiar with general concepts such as overfitting, regularization,….

3. Code and train your first neural network

What not to expect?

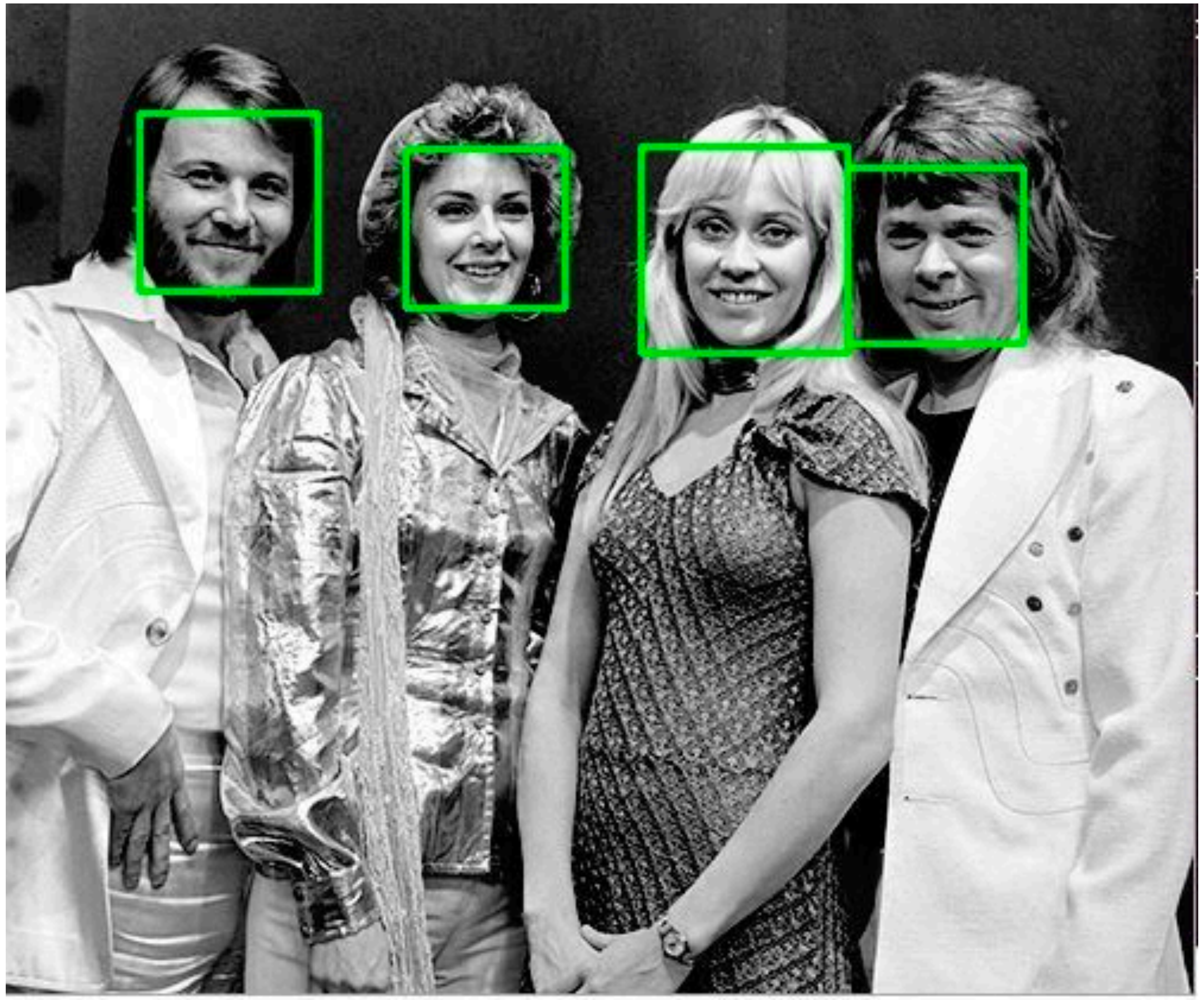
1. You will not become an expert in deep learning

2. You (most likely) won't be able to develop your own deep learning model

3. You won't know the state-of-the-art technics to deploy neural networks

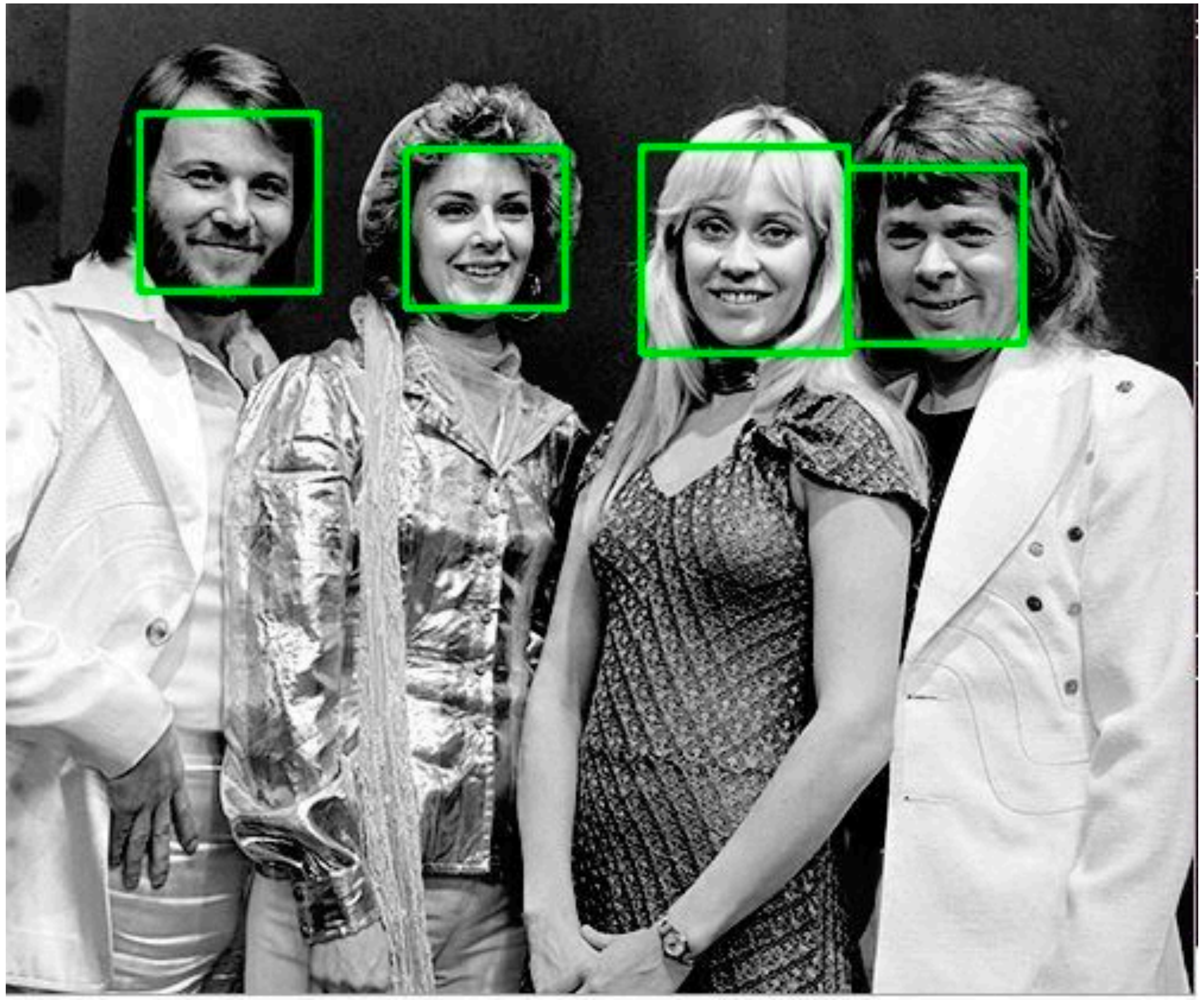
Facial Detection / Recognition

Facial Detection / Recognition



Face detection

Facial Detection / Recognition

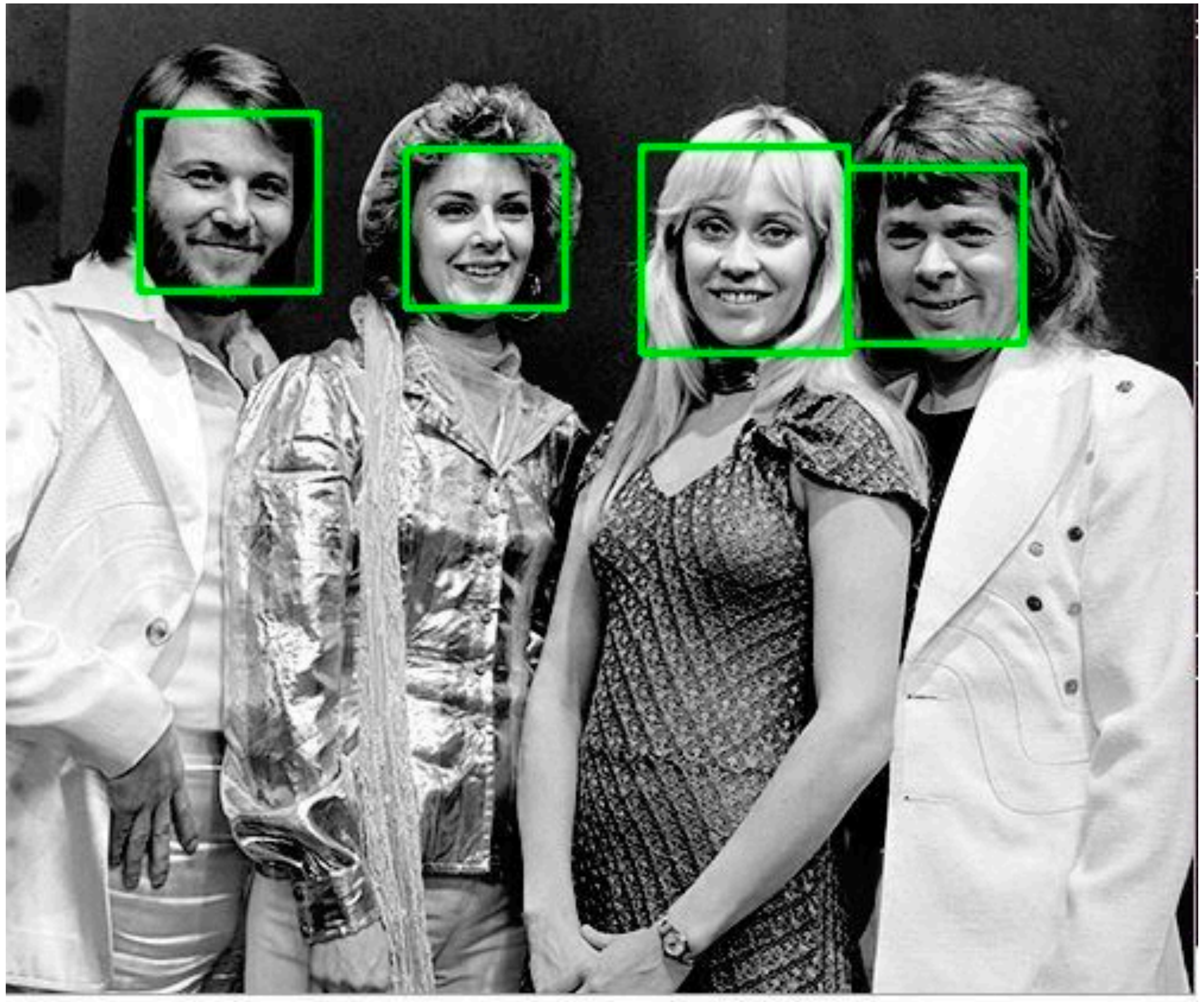


Face detection

Unlock your smarphone

Take pictures

Facial Detection / Recognition



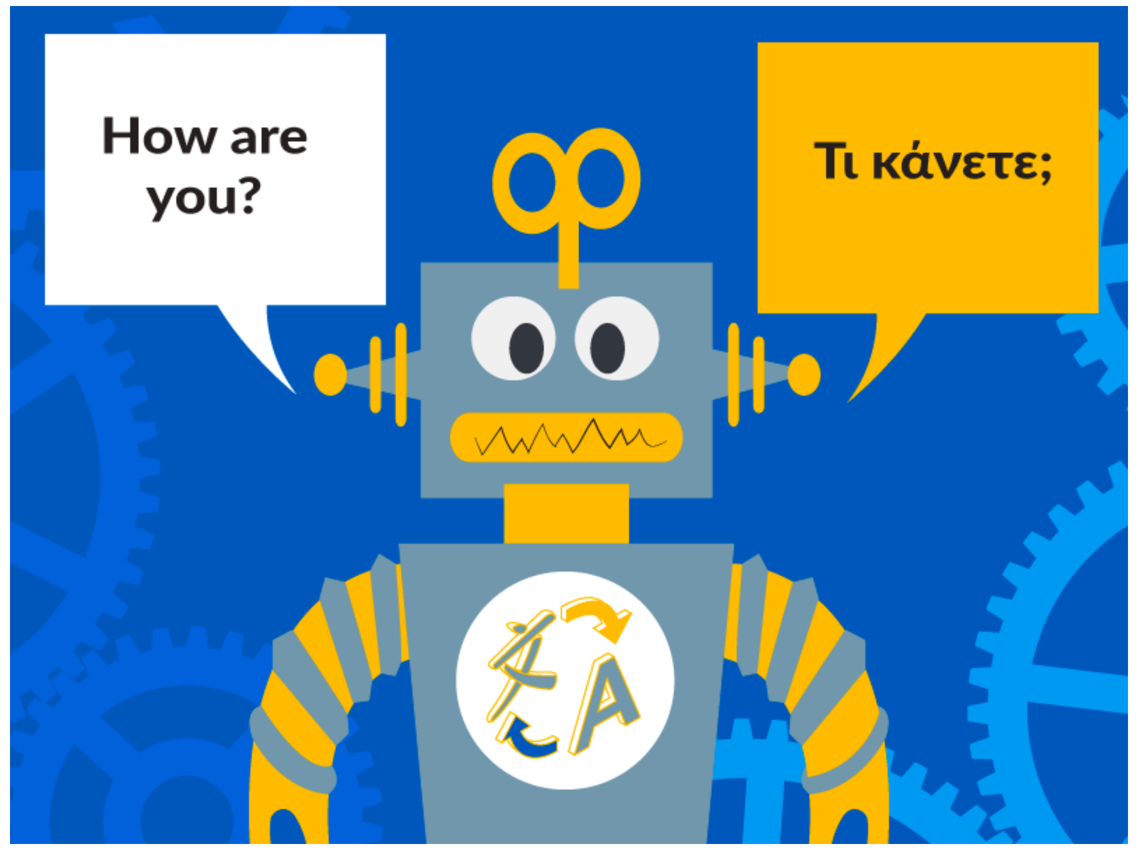
Face detection

Unlock your smarphone

Take pictures

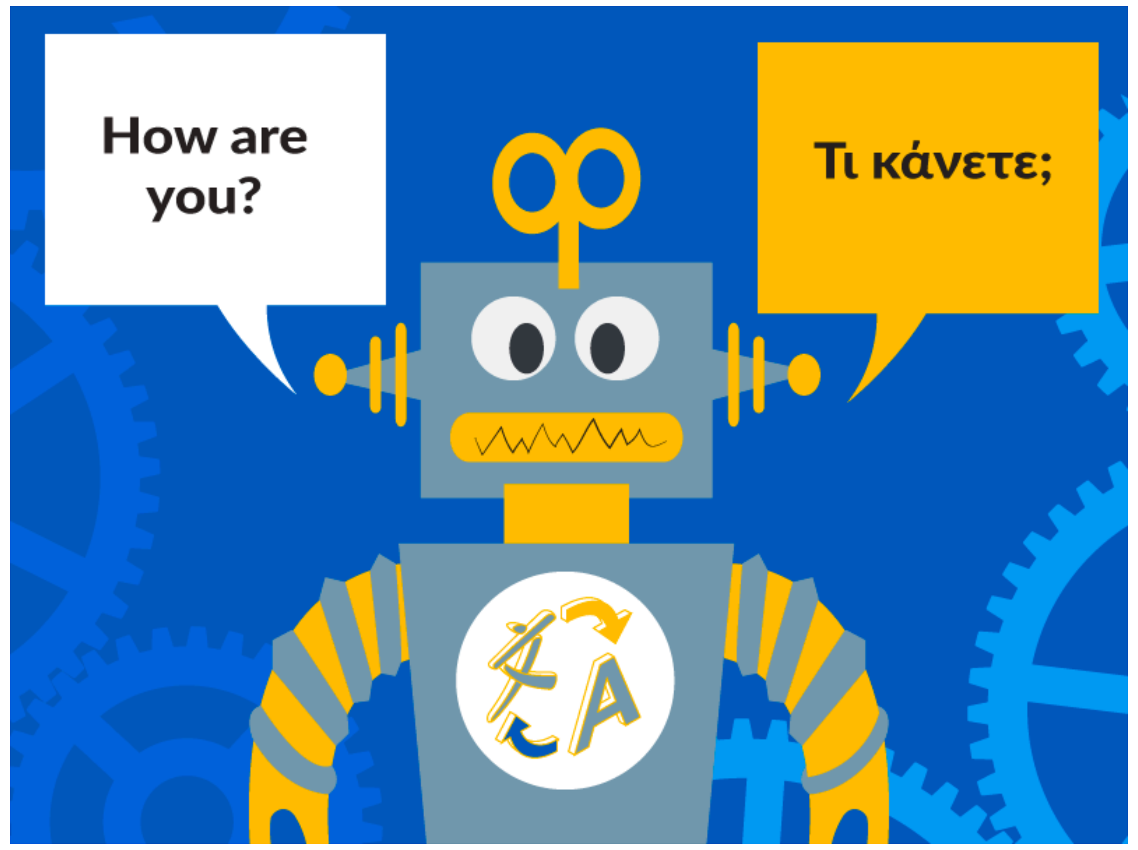
Airport

Translation / Information retrieval



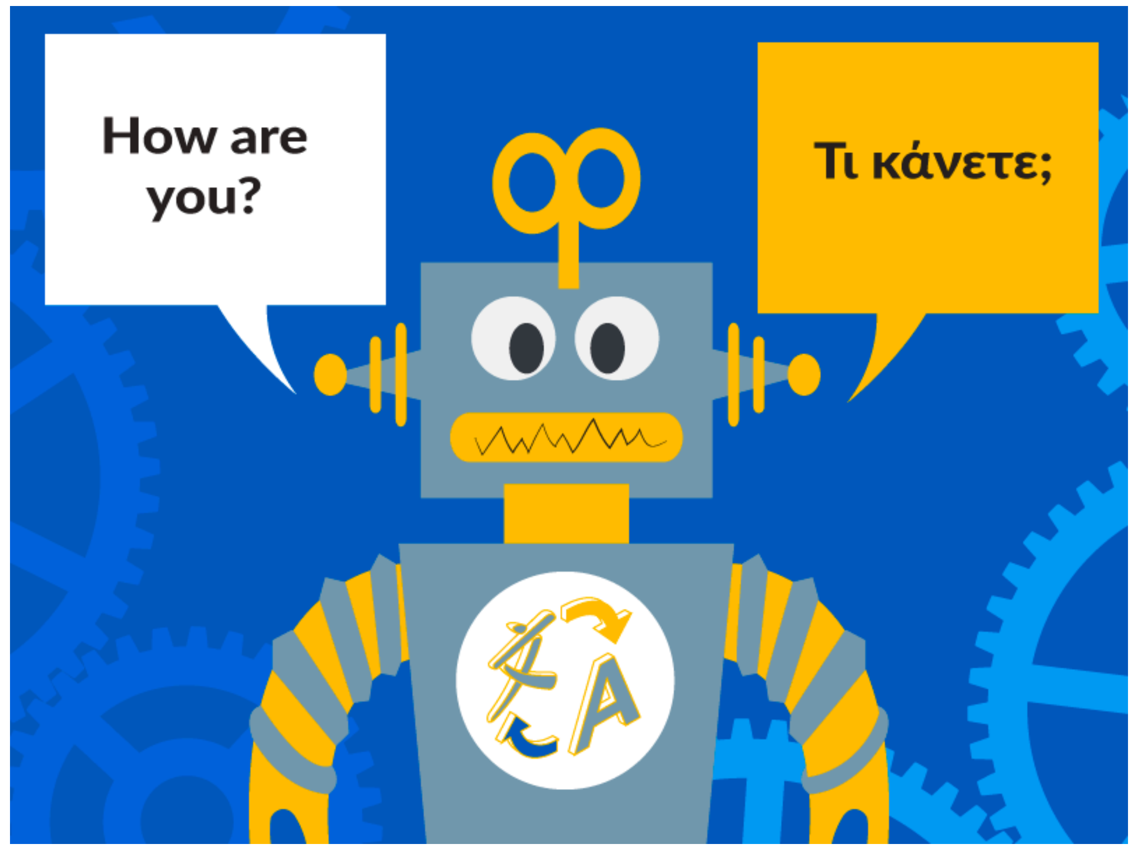
Translation

Translation / Information retrieval

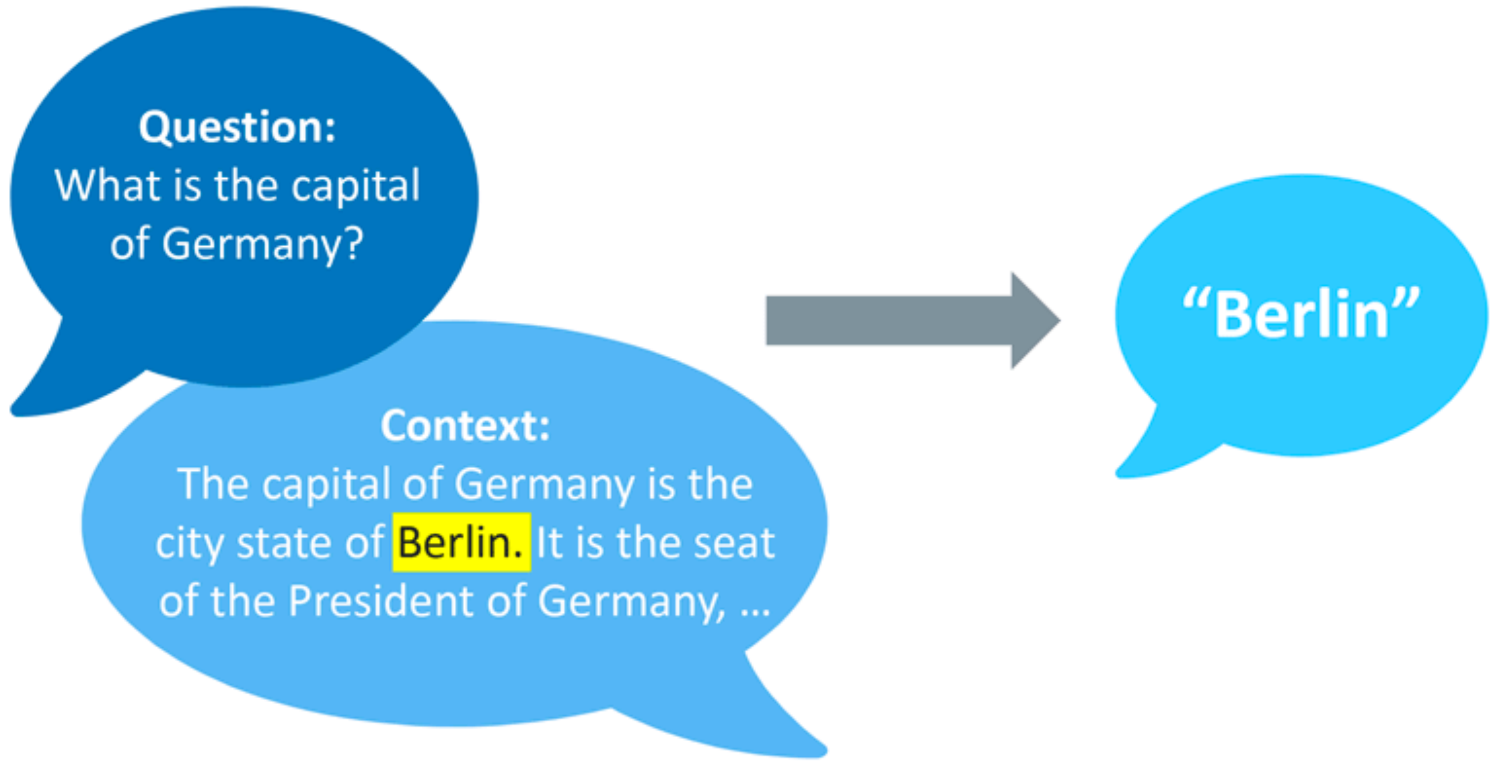


Translation

Information Retrieval



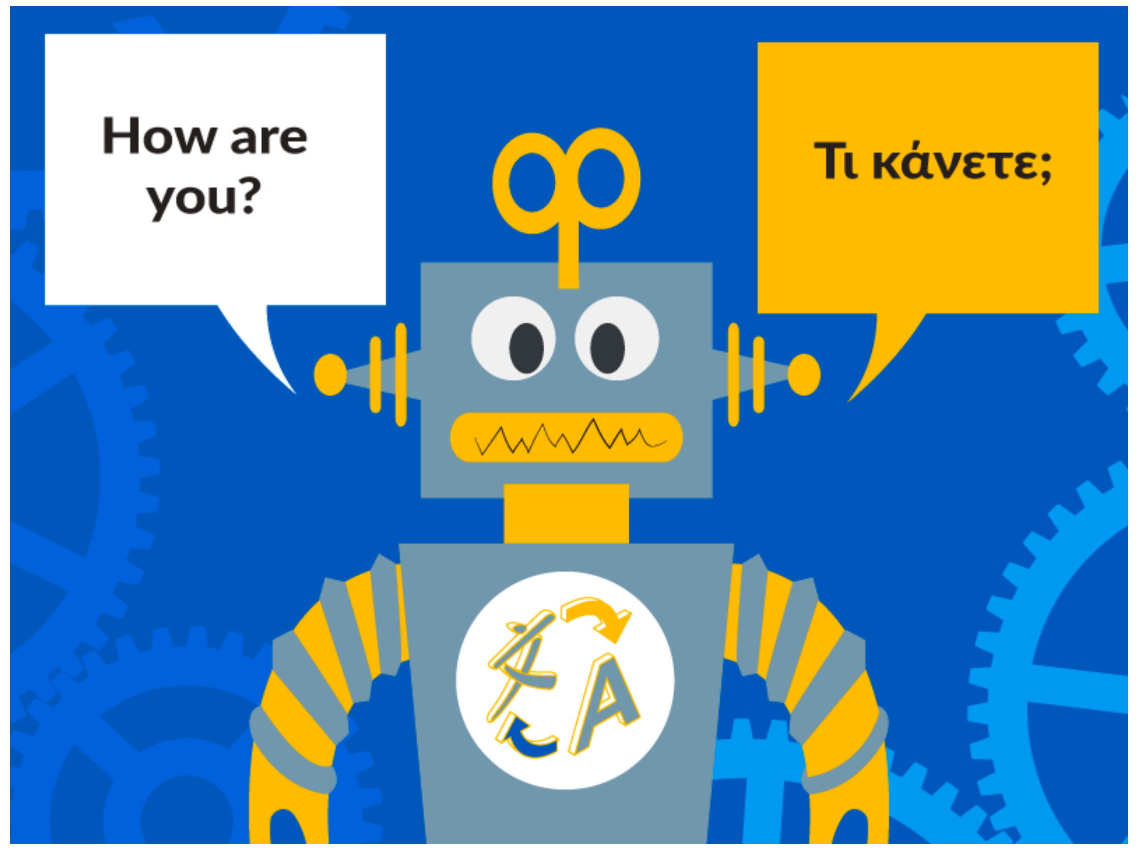
Translation



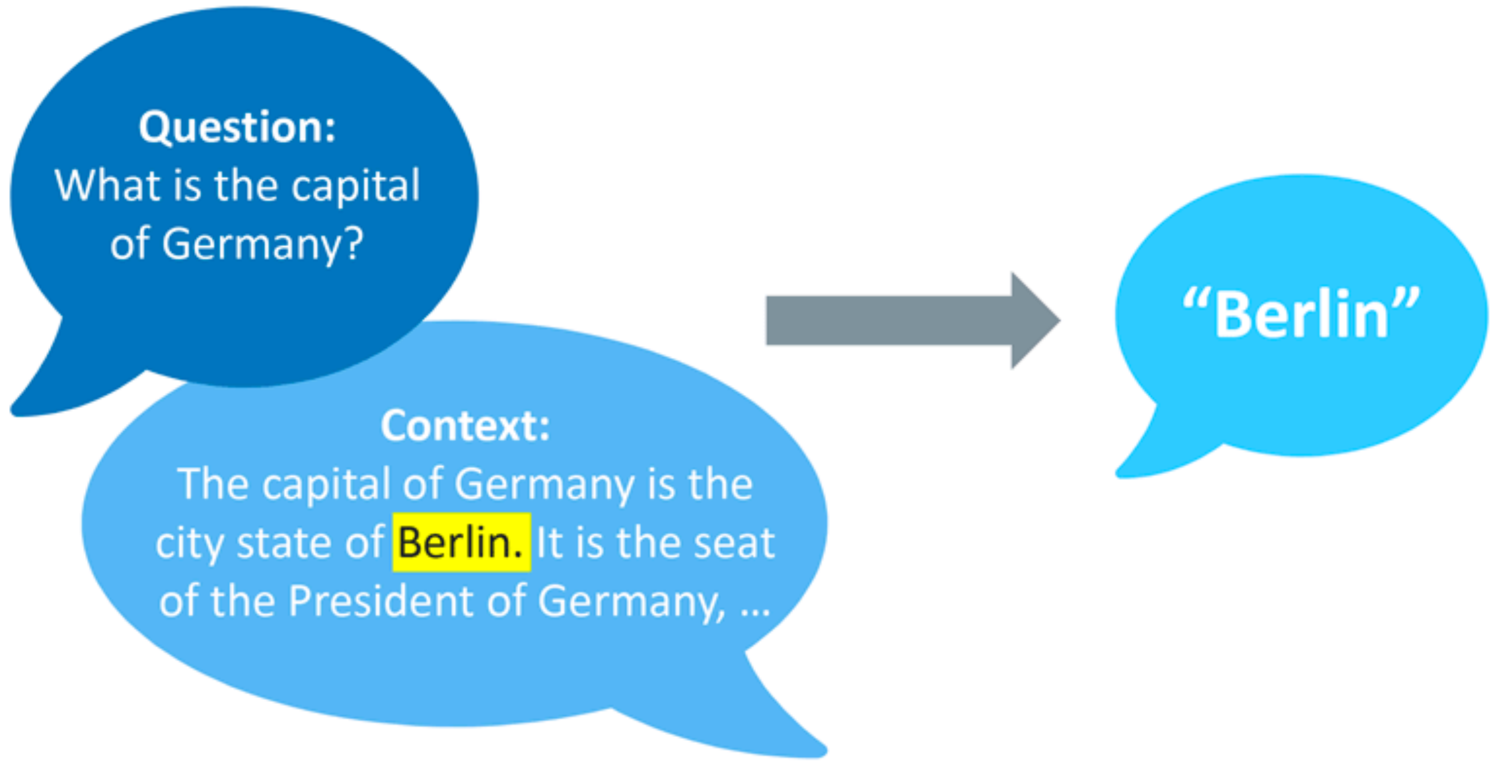
Question Answering

Information Retrieval

Translation / Information retrieval



Translation

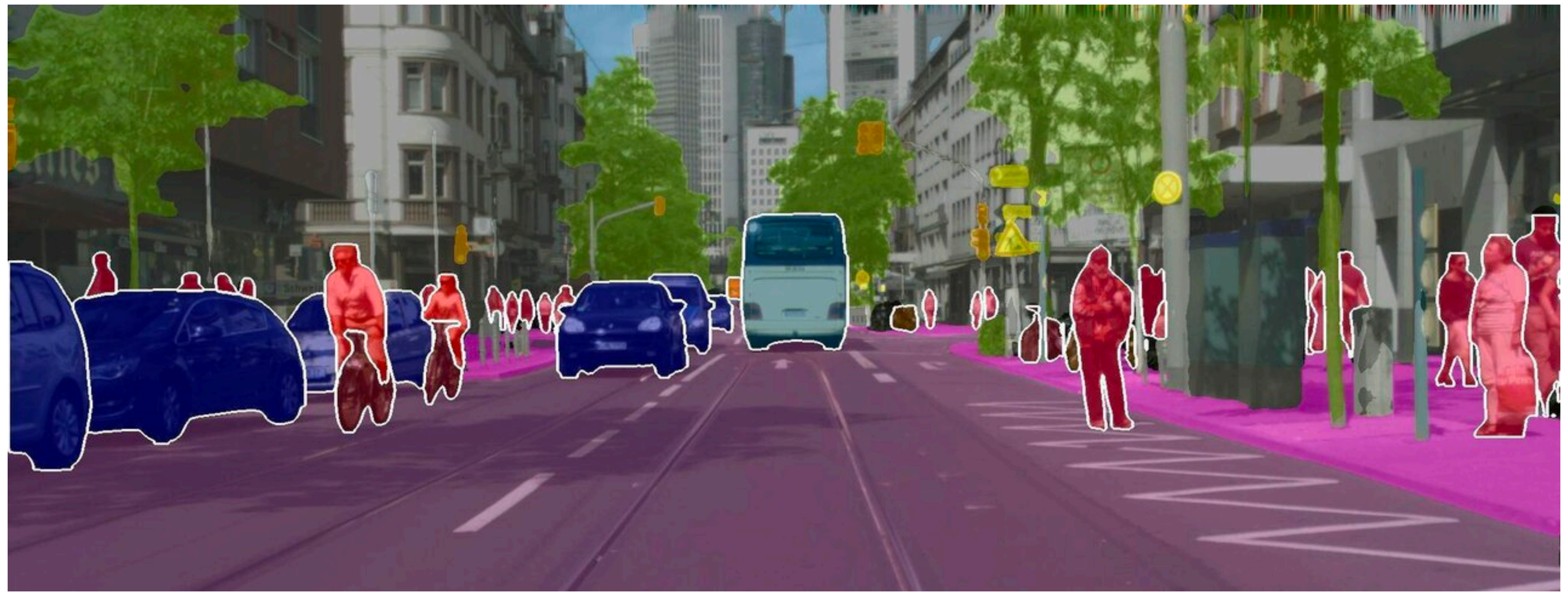


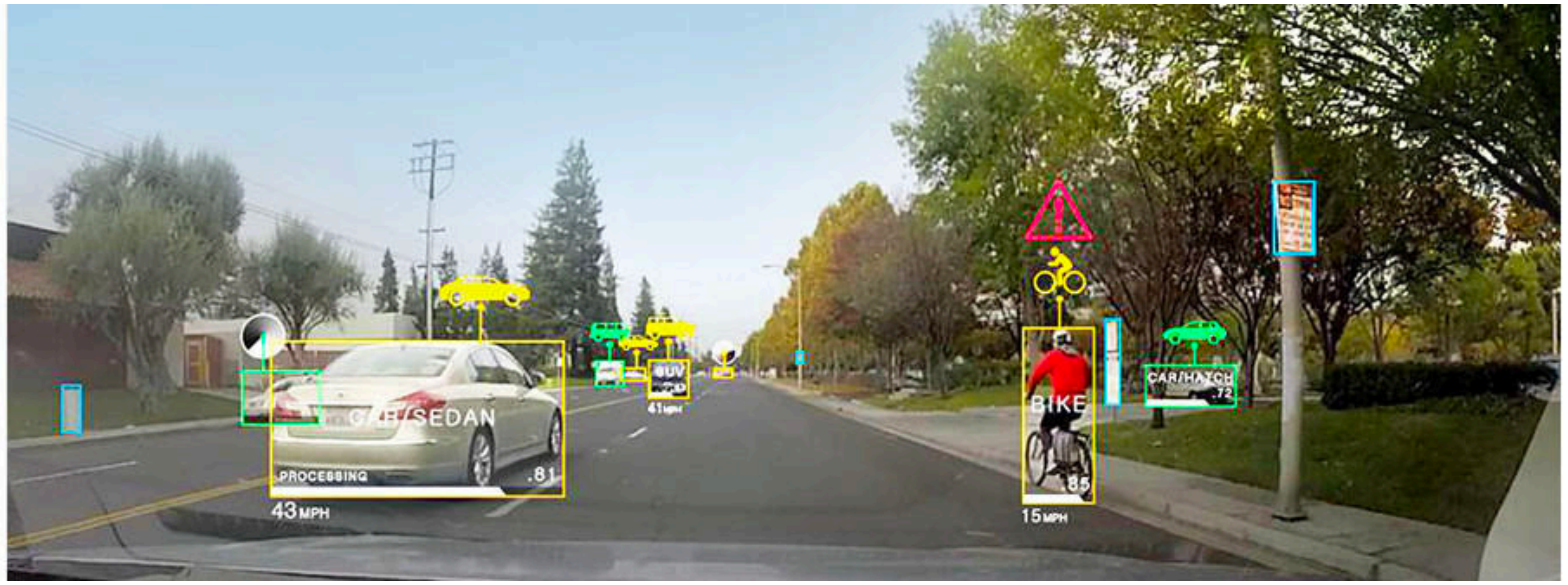
Question Answering

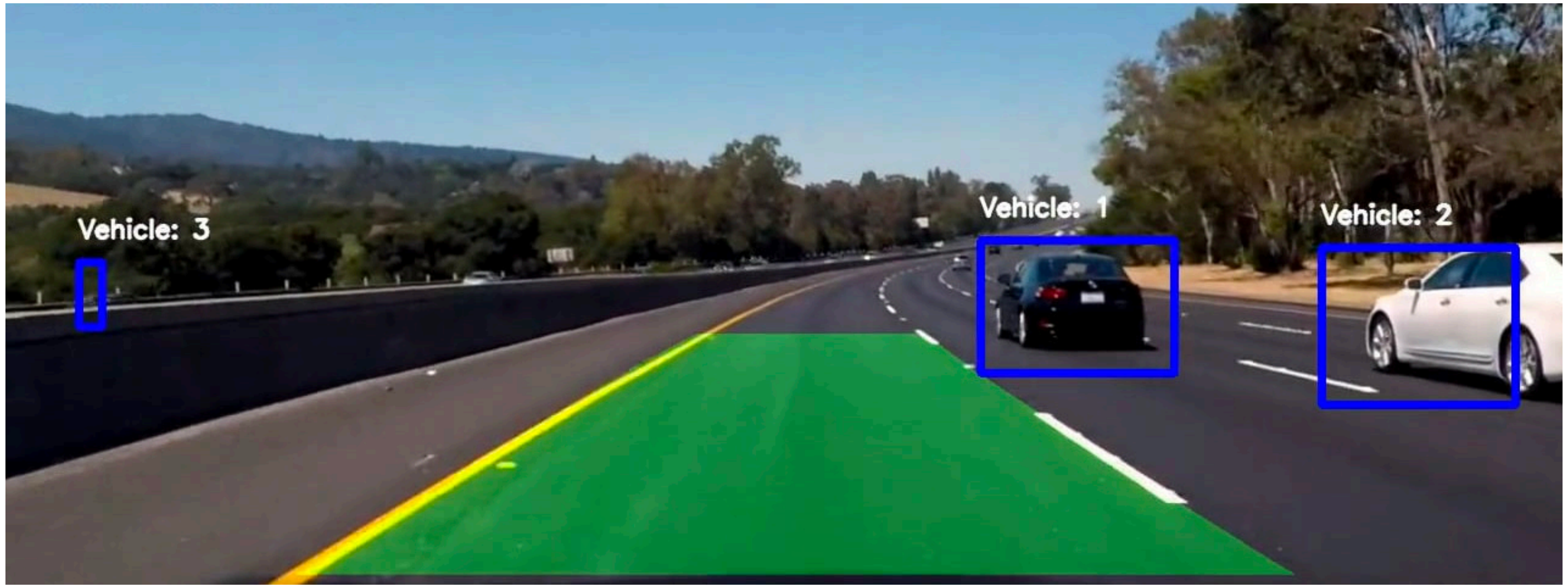
Information Retrieval

Social Media Analysis

Autonomous Driving

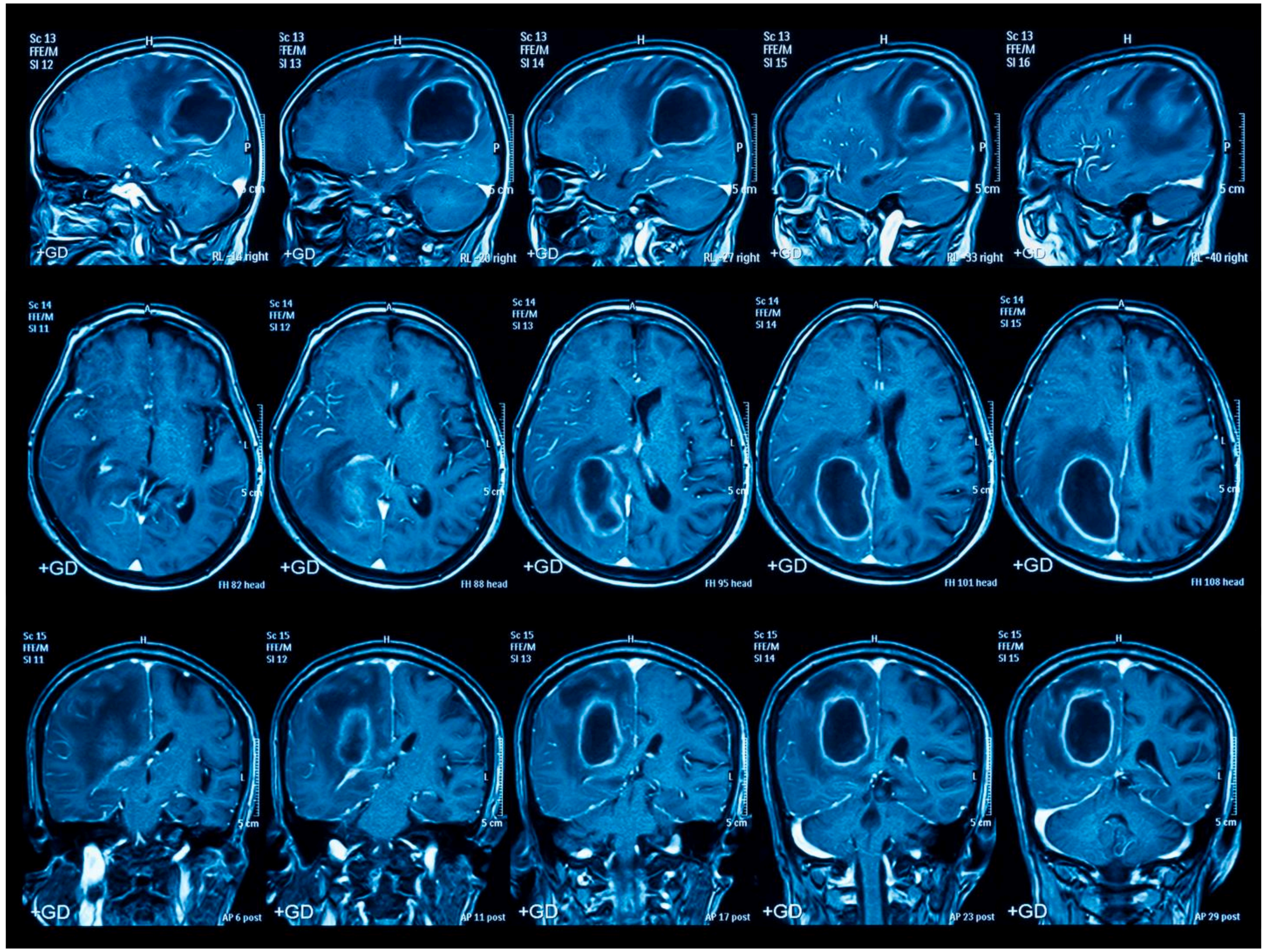



CAR/SEDAN
PROCESSING
.81
43 MPH
BIKE
15 MPH
CAR/HATCH

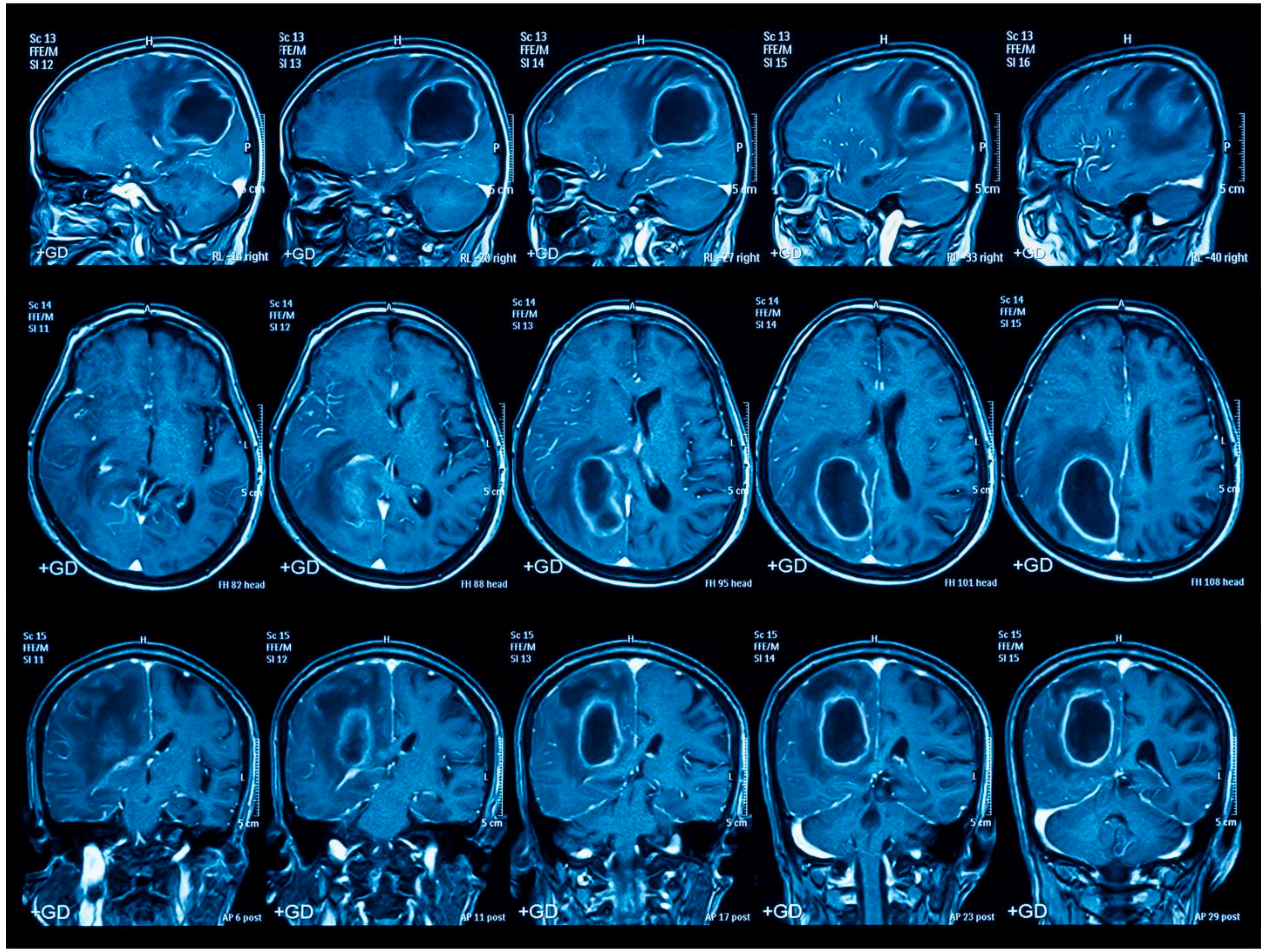

Vehicle: 3
Vehicle: 1
Vehicle: 2

Medical Application

Medical Application



Medical Imaging



Medical Imaging

Drug Discovery

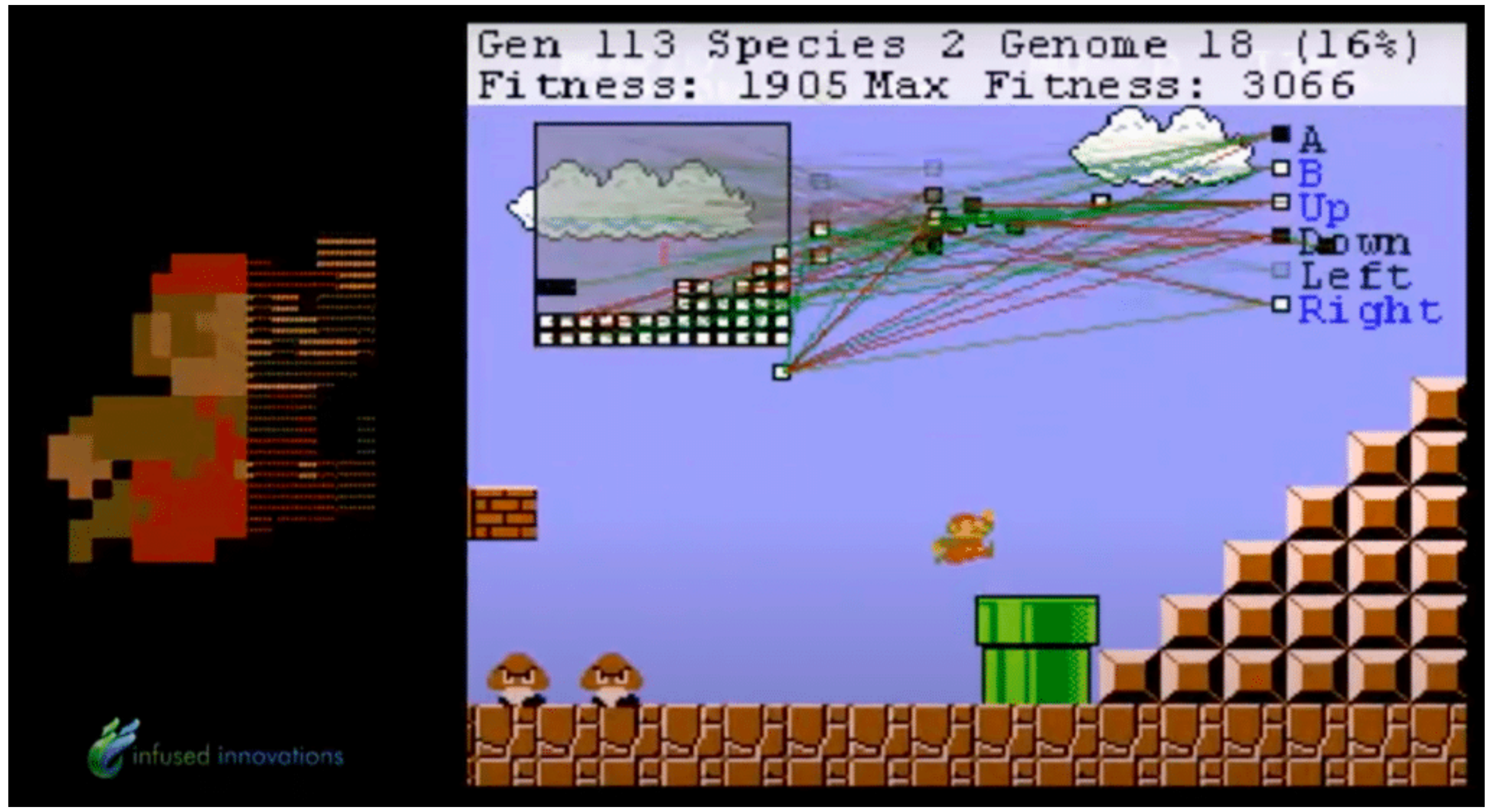
Other

Other

Sport Analysis

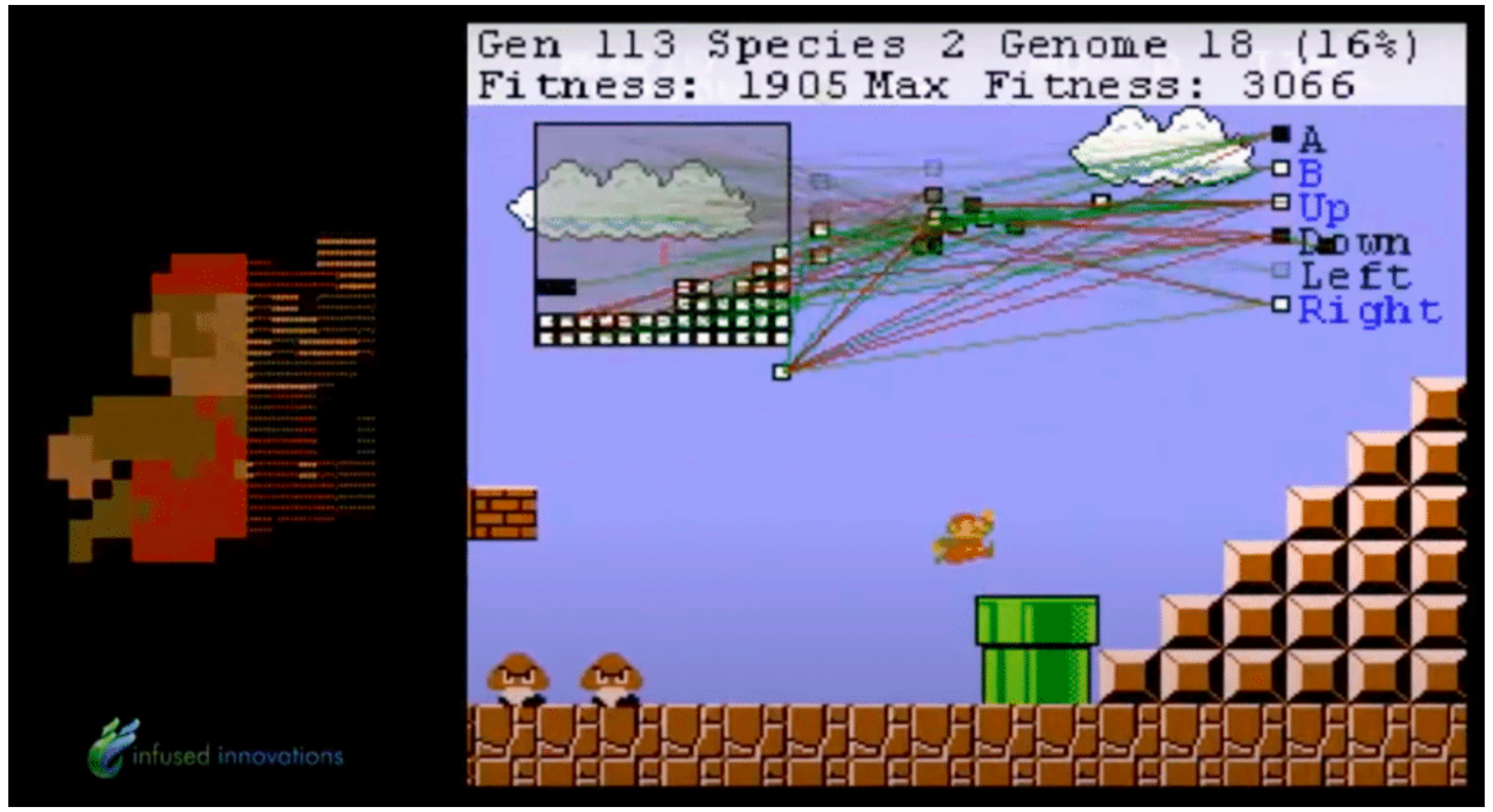
Other

Sport Analysis



Video Games

Sport Analysis

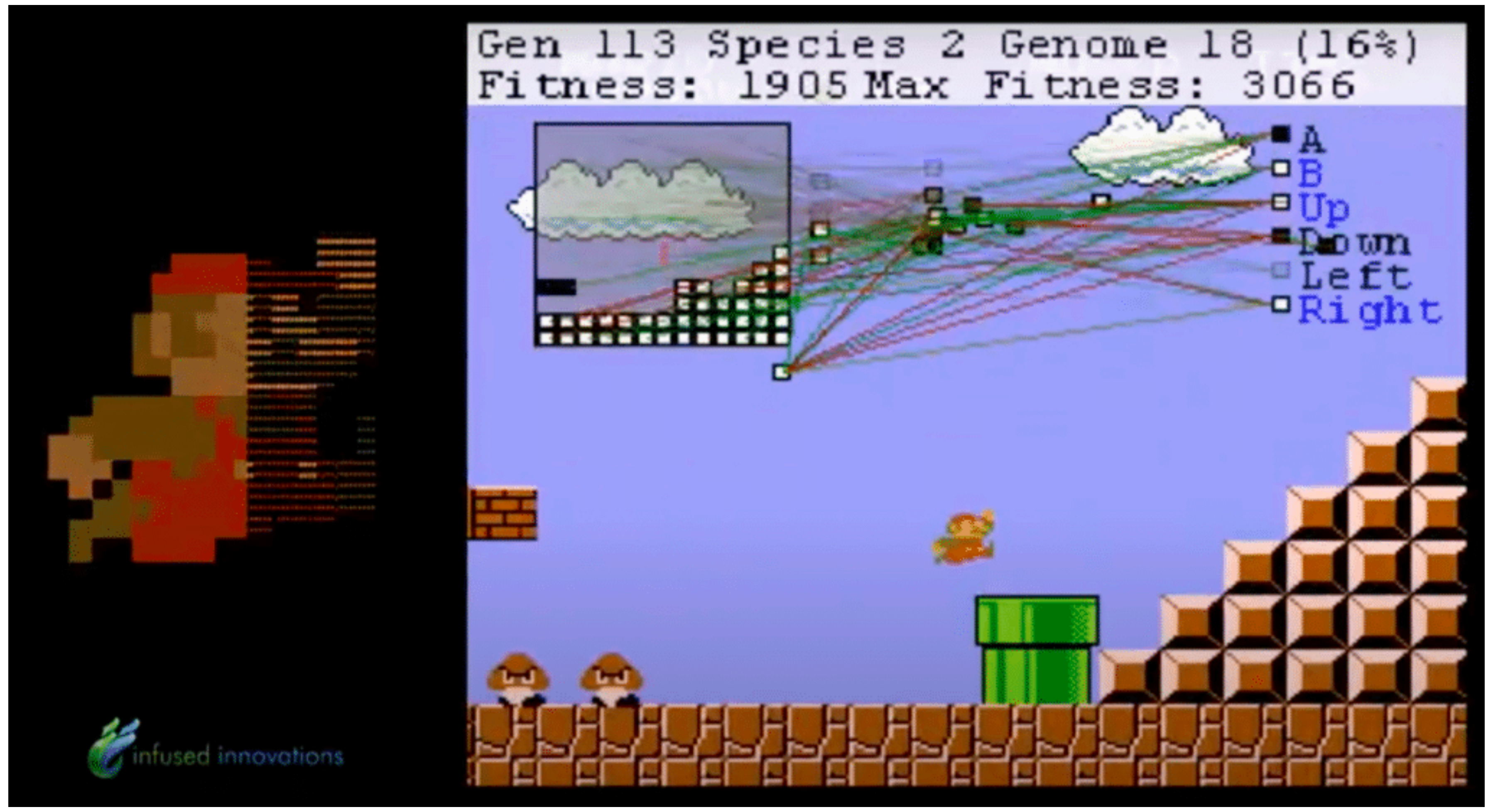


Video Games

$$
\begin{cases}
\dfrac{\mathrm{d}x}{\mathrm{d}t} = \sigma\,[(y(t) - x(t)] \\
\dfrac{\mathrm{d}y}{\mathrm{d}t} = \rho\,x(t) - y(t) - x(t)\,z(t) \\
\dfrac{\mathrm{d}z}{\mathrm{d}t} = x(t)\,y(t) - \beta\,z(t)
\end{cases}
$$

ODE

Sport Analysis



Video Games

$$\begin{cases} \frac{\mathrm{d}x}{\mathrm{d}t} = \sigma\left[(y(t) - x(t)\right] \\ \frac{\mathrm{d}y}{\mathrm{d}t} = \rho\, x(t) - y(t) - x(t)\, z(t) \\ \frac{\mathrm{d}z}{\mathrm{d}t} = x(t)\, y(t) - \beta\, z(t) \end{cases}$$

ODE

Chess

Why and when to use deep learning in practice ?

Why and when to use deep learning in practice ?

Deep learning rely on the concept of learning from examples.

Deep learning rely on the concept of learning from examples.

You need « a lot of examples »

Why and when to use deep learning in practice ?

Deep learning rely on the concept of learning from examples.

You need « a lot of examples »

How much in practice?

Why and when to use deep learning in practice ?

Deep learning rely on the concept of learning from examples.

You need « a lot of examples » **How much in practice?** **Open research question**

Why and when to use deep learning in practice ?

Deep learning rely on the concept of learning from examples.

You need « a lot of examples » How much in practice? Open research question

Why should you use deep learning?

Hand-crafted features are time consuming, not scalable and in practice sub-optimal

Deep Learning aims at automatically learning features directly from data!

Why and when to use deep learning in practice ?

Deep learning rely on the concept of learning from examples.

You need « a lot of examples » How much in practice? Open research question

Why should you use deep learning?

Hand-crafted features are time consuming, not scalable and in practice sub-optimal

Deep Learning aims at automatically learning features directly from data!

Pattern,
discriminative characteristics

Deep learning rely on the concept of learning from examples.

You need « a lot of examples » How much in practice? Open research question

Why should you use deep learning?

Hand-crafted features are time consuming, not scalable and in practice sub-optimal

Deep Learning aims at automatically learning features directly from data!

Pattern,
discriminative characteristics

MNIST

Why and when to use deep learning in practice ?

Deep learning rely on the concept of learning from examples.

You need « a lot of examples » How much in practice? Open research question

Why should you use deep learning?

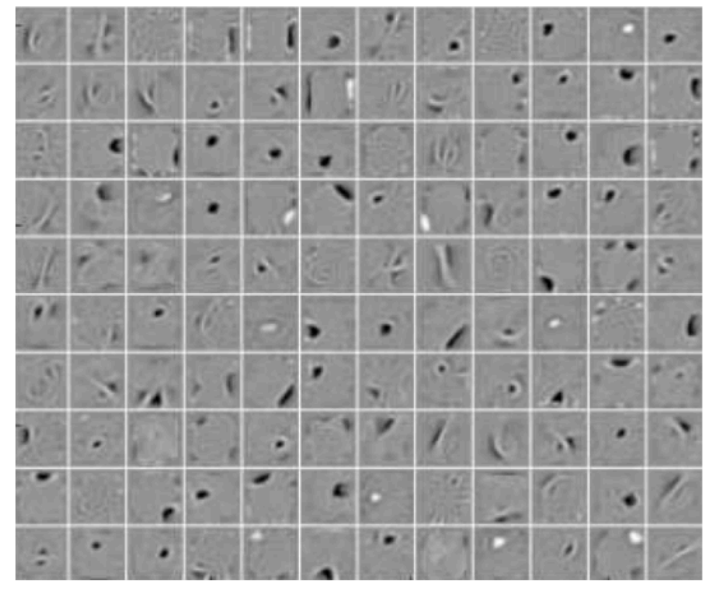
Hand-crafted features are time consuming, not scalable and in practice sub-optimal

Deep Learning aims at automatically learning $\boxed{\text{features}}$ directly from data!

Pattern,
discriminative characteristics

MNIST

Features Learnt by the NN



Deep learning rely on the concept of learning from examples.

You need « a lot of examples » How much in practice? Open research question

Why should you use deep learning?

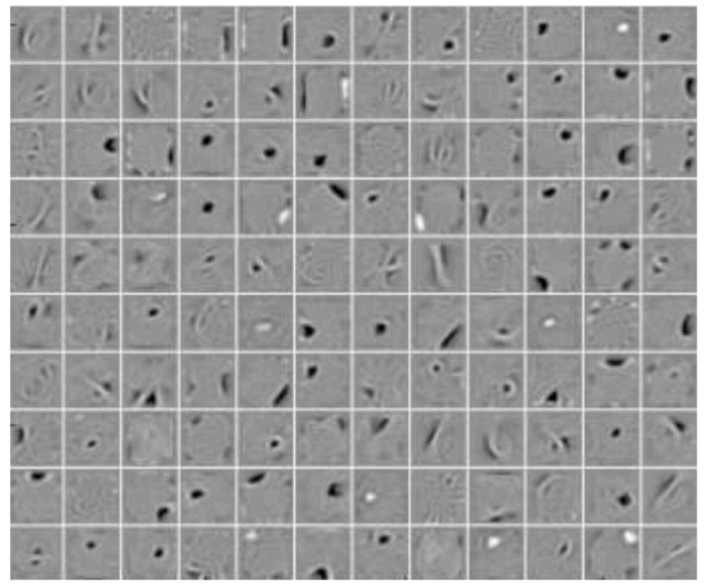
Hand-crafted features are time consuming, not scalable and in practice sub-optimal

Deep Learning aims at automatically learning features directly from data!

Pattern,
discriminative characteristics

MNIST

Features Learnt by the NN

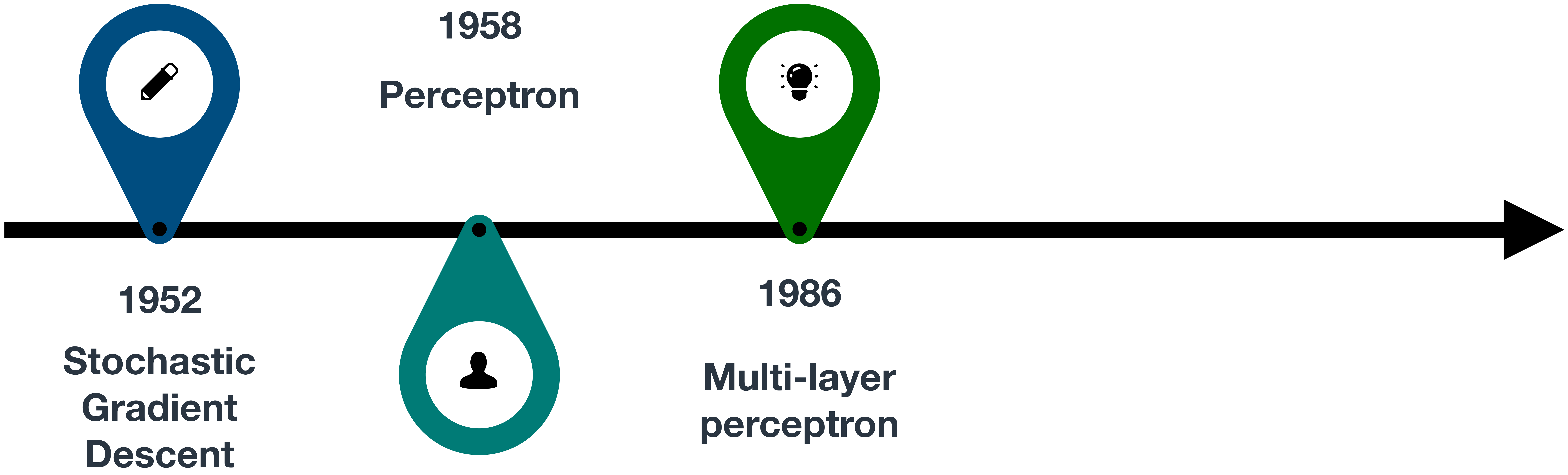


Theses features
are hard to guess !

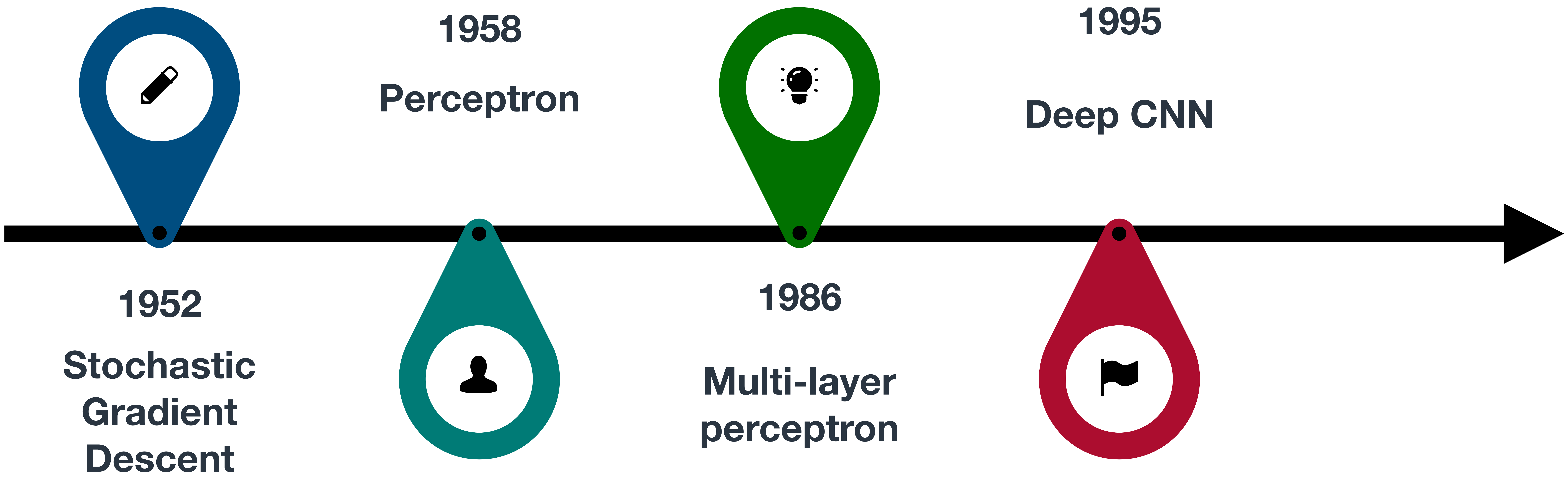
A brief history of deep learning

A brief history of deep learning

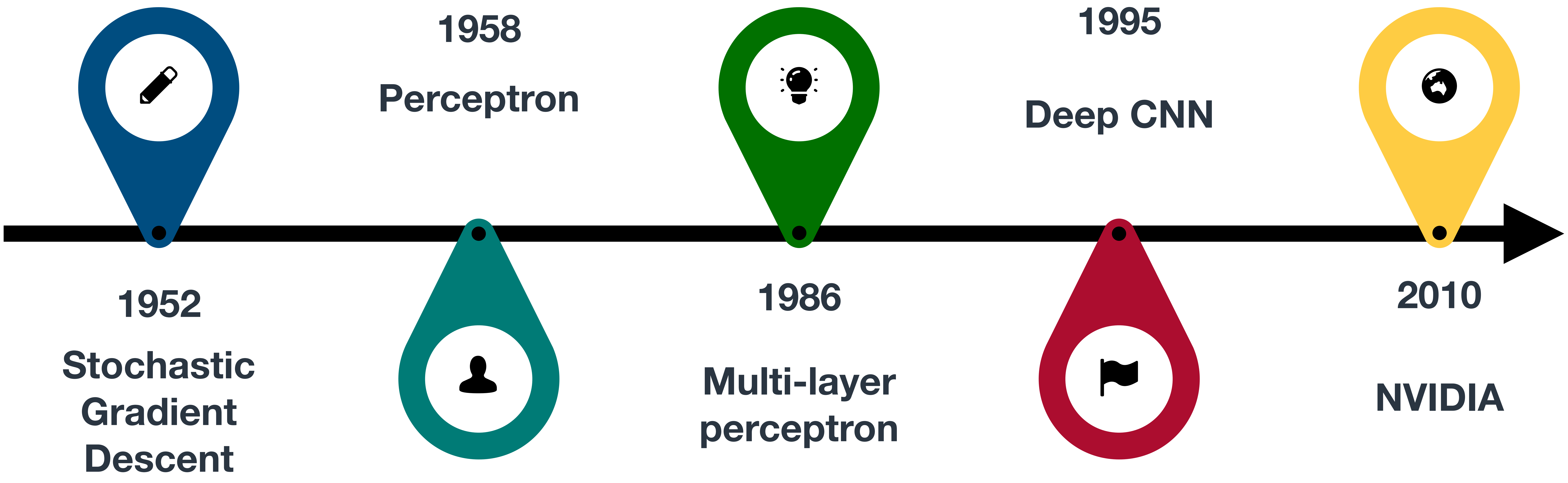
A brief history of deep learning



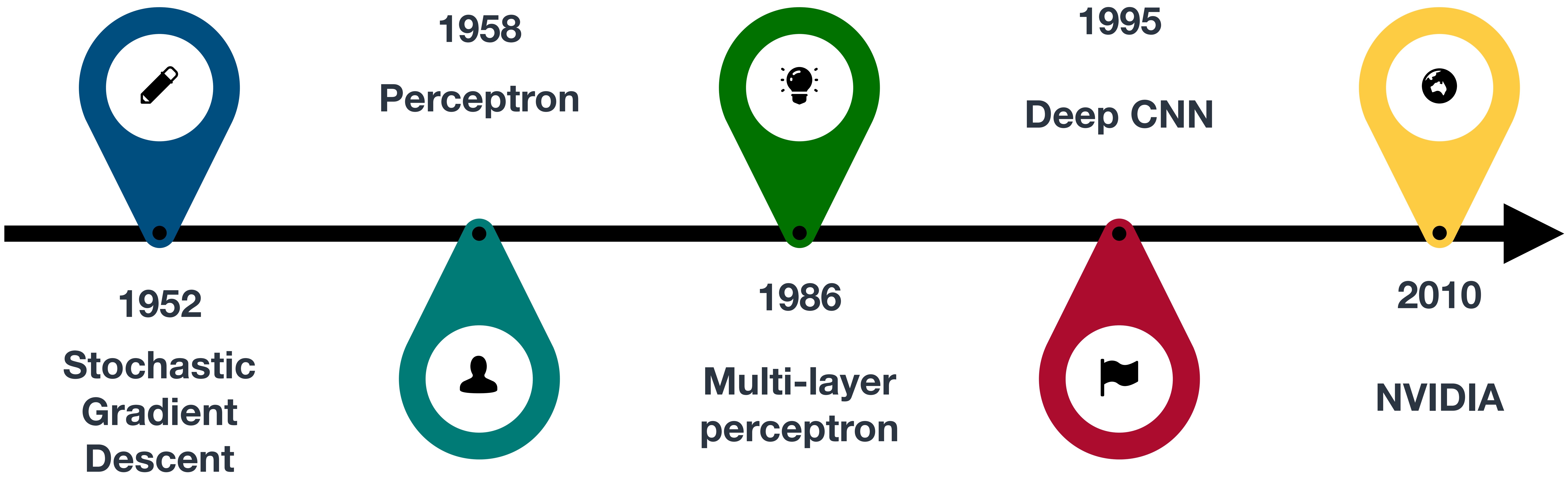
A brief history of deep learning



A brief history of deep learning



A brief history of deep learning



Increase of computing power and available data

Why Now ?

Deep learning date back decades, so why now?

With internet we have access to massive amount of data

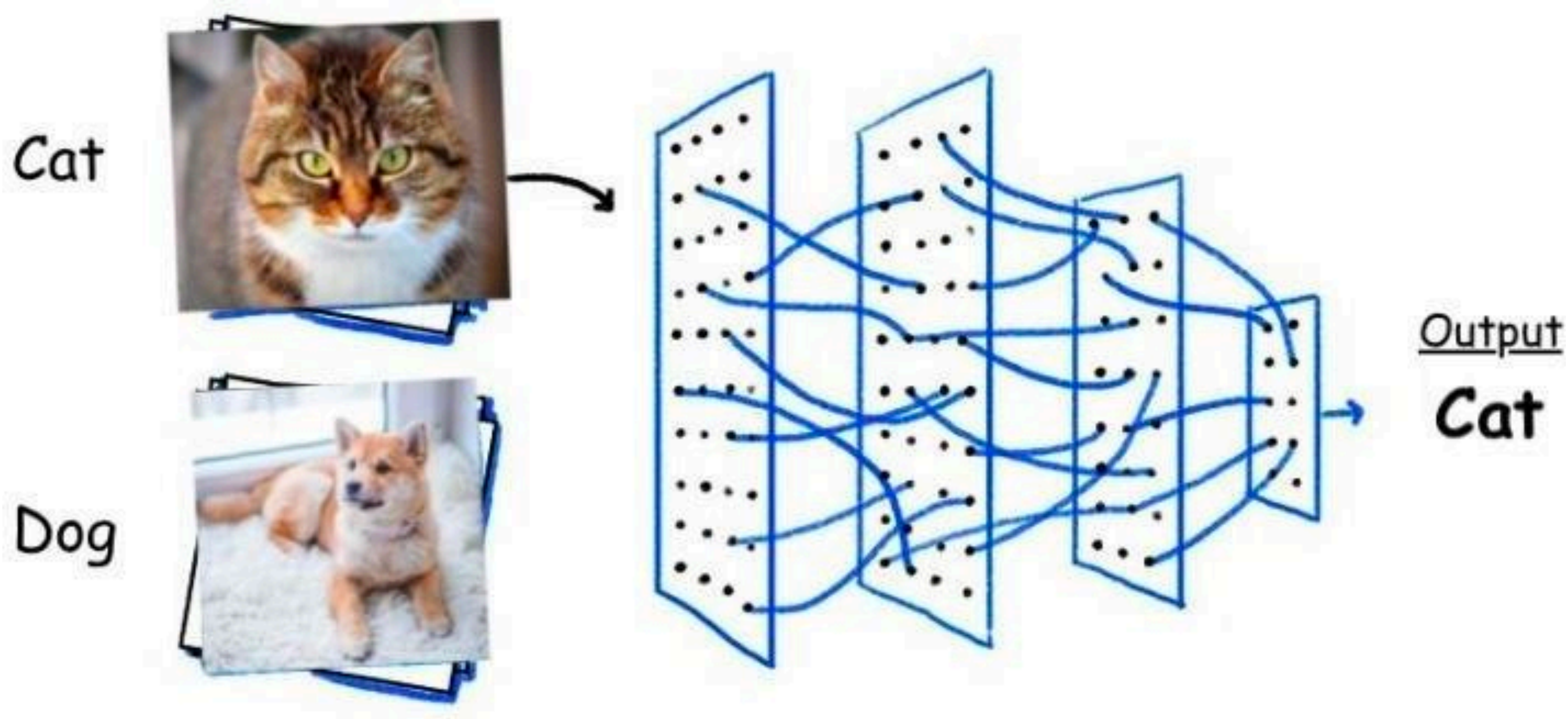
With GPUs we have access to massive amount of GPUs

Jeanzay etc..

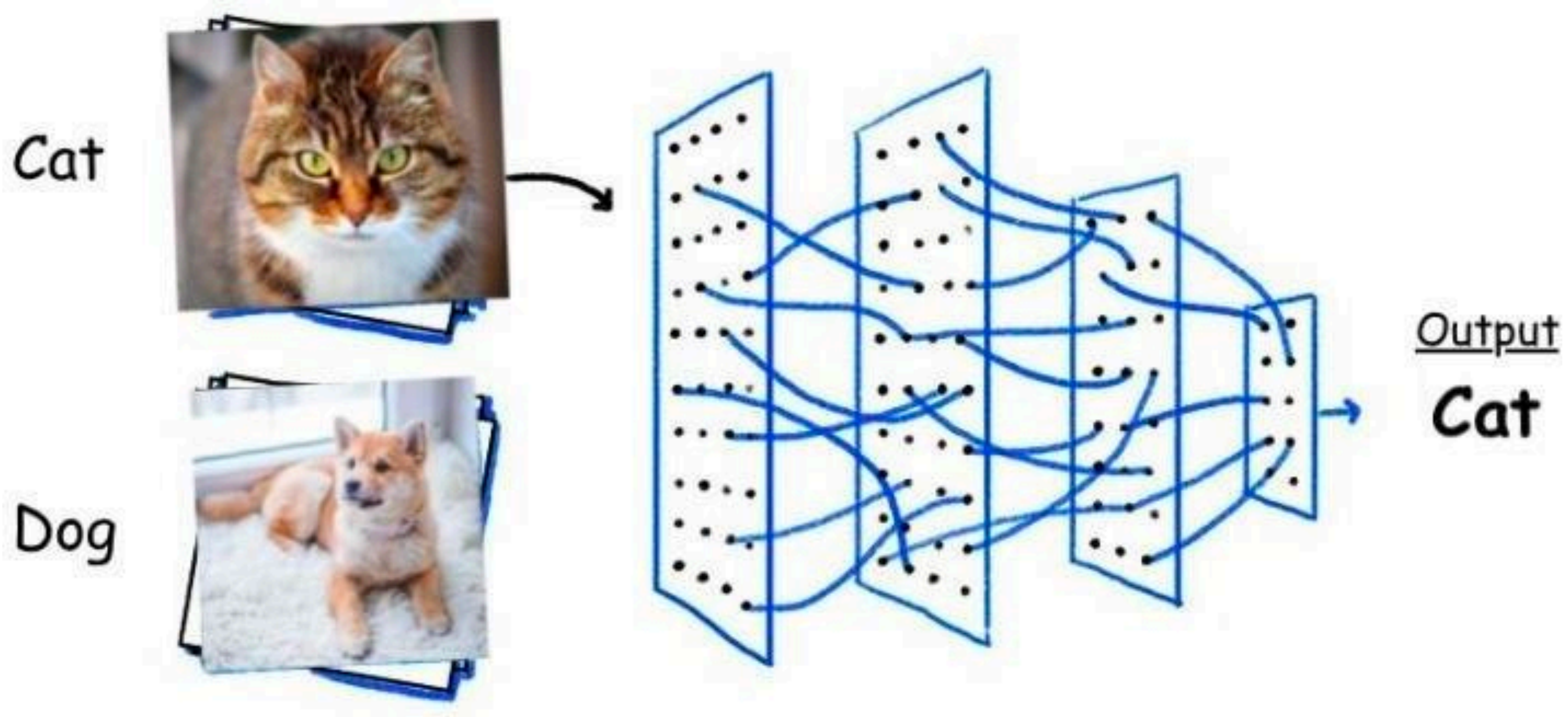
With the huge effort of Google, Facebook and others we have access to high level Software

Example of Task to solve with a neural network (I)

Example of Task to solve with a neural network (I)

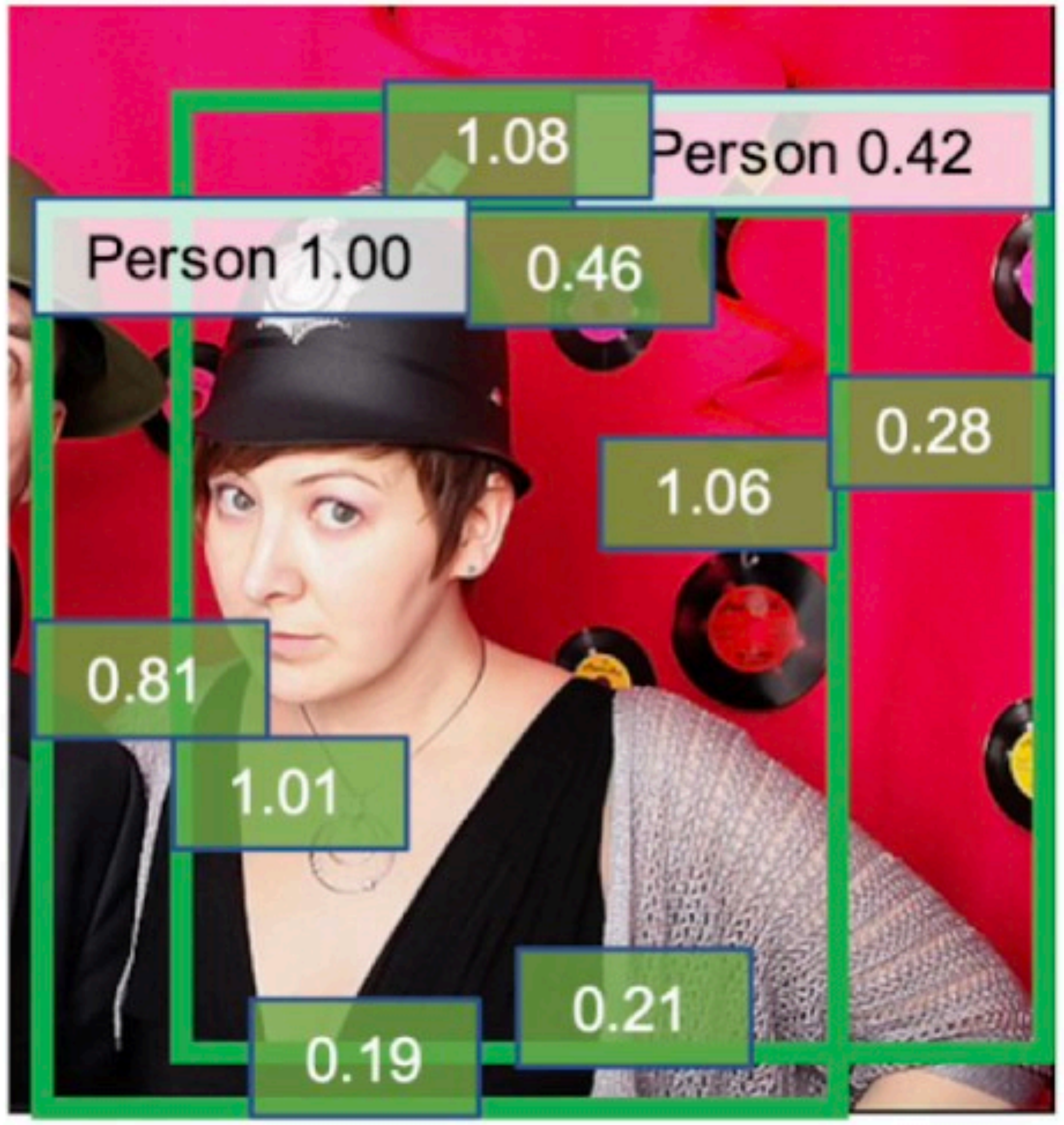


Example of Task to solve with a neural network (I)

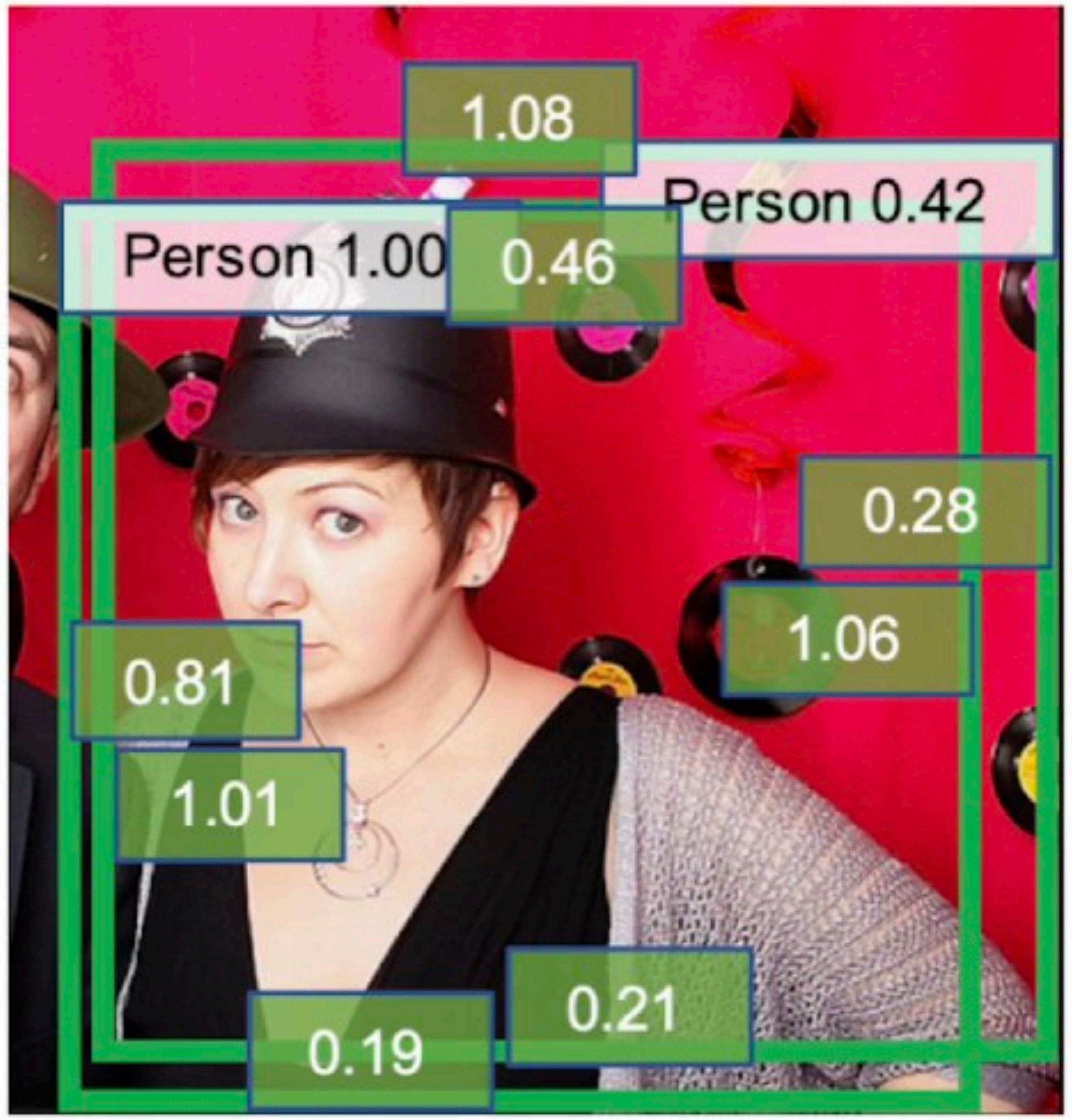


NN can do that without any image processing background

Example of Task to solve with a neural network (II)

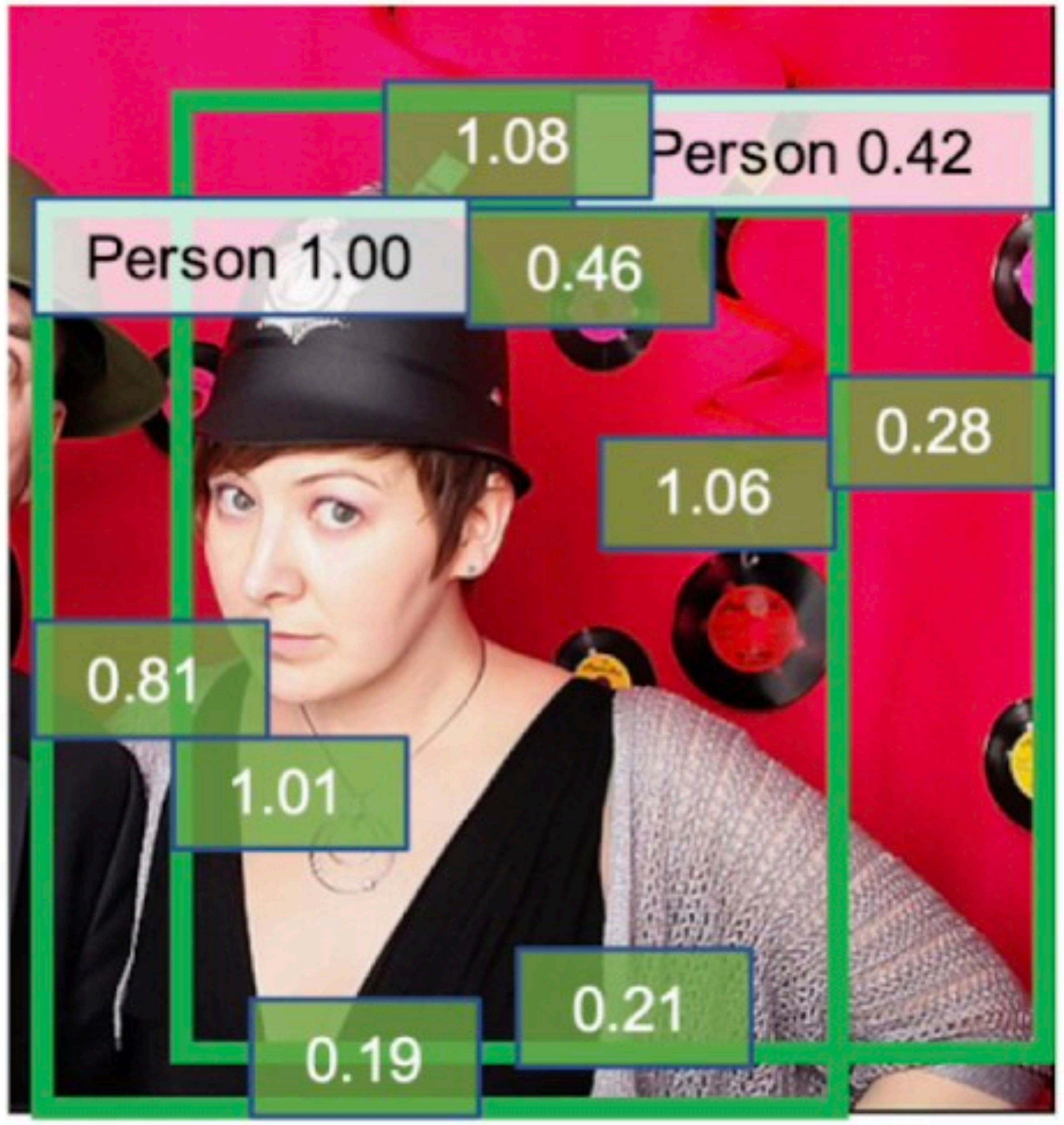


Before

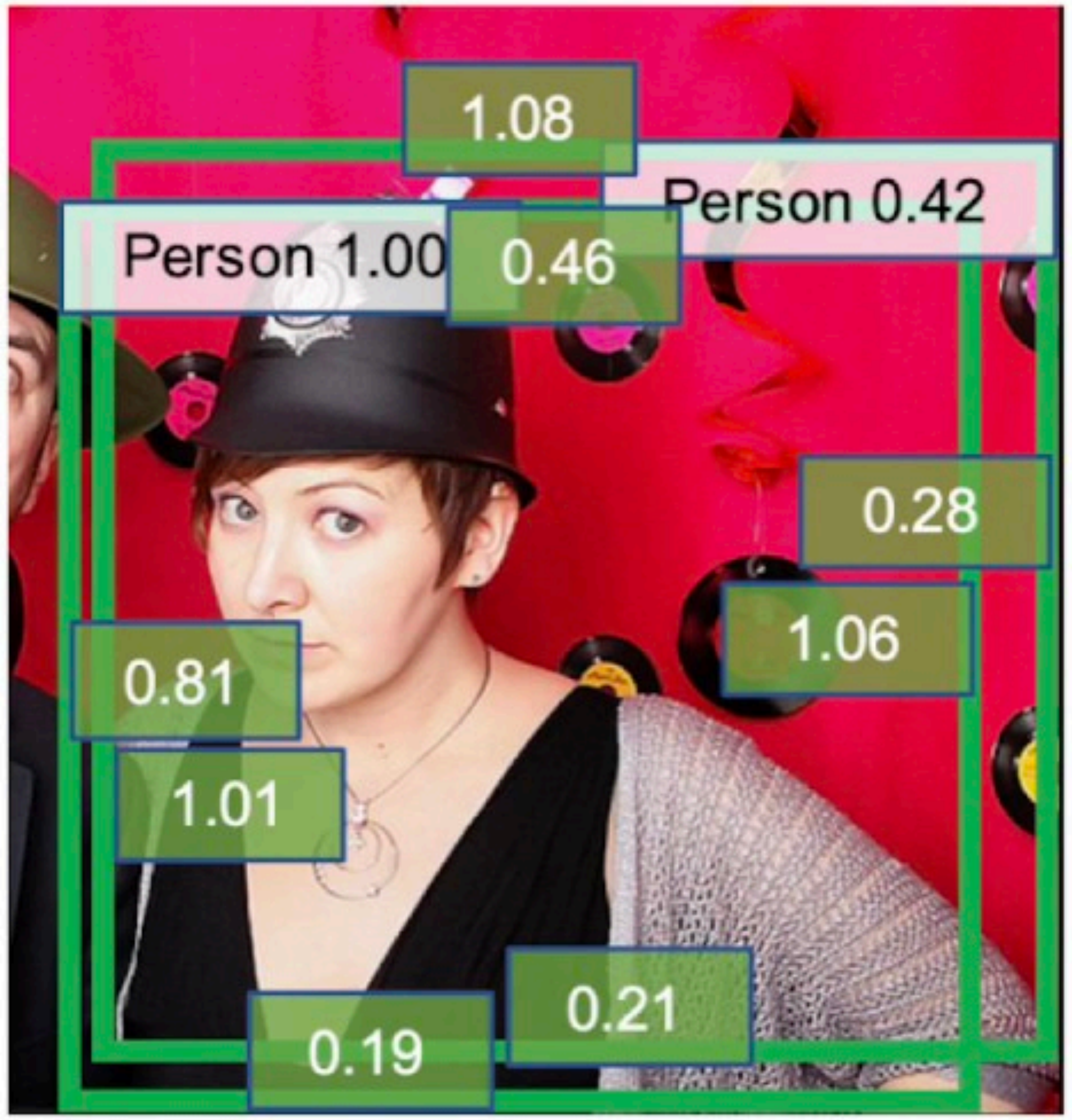


after

Example of Task to solve with a neural network (II)



Before



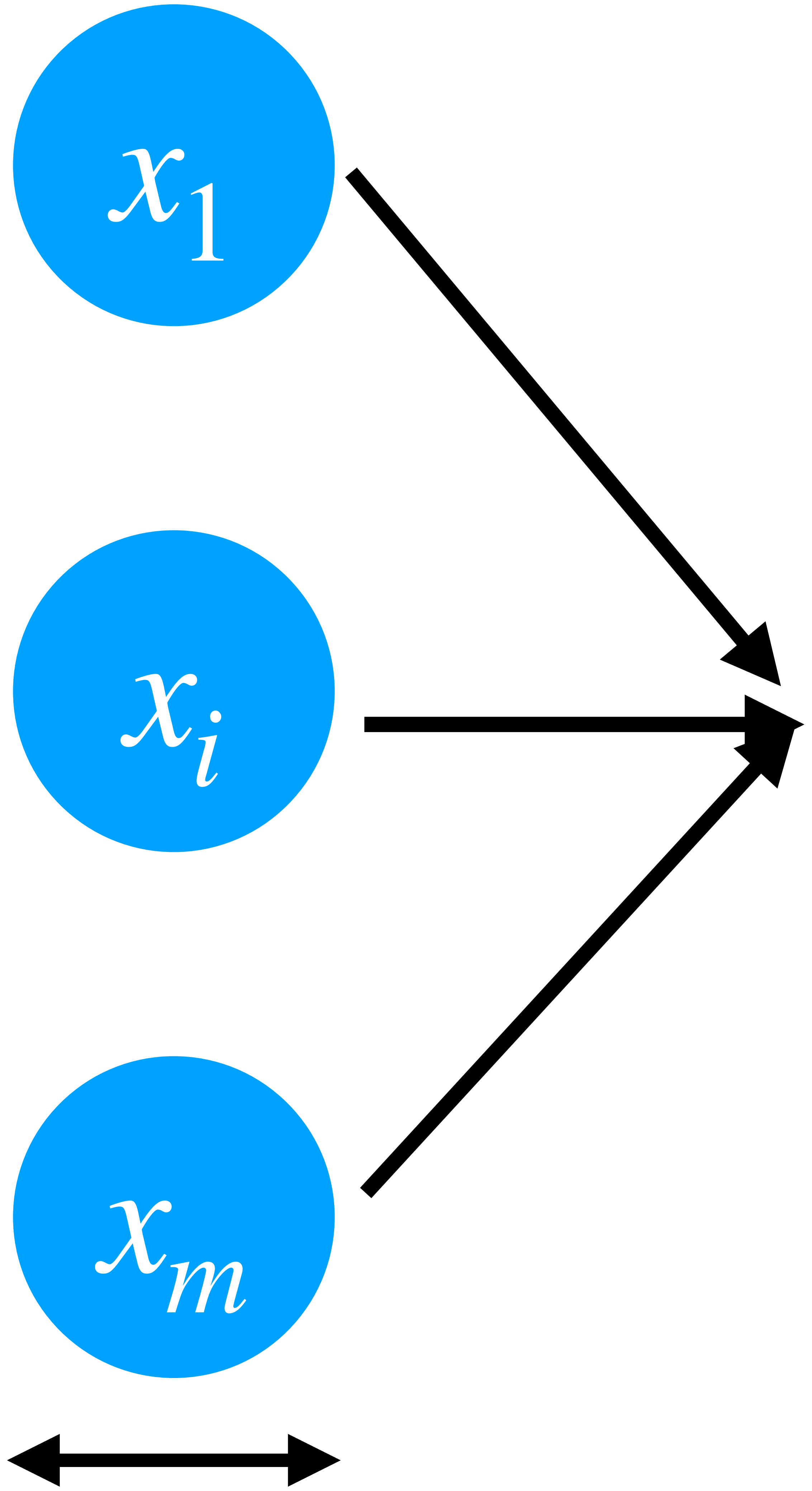
after

NN can do that without any image processing background

Design the architecture of the neural network

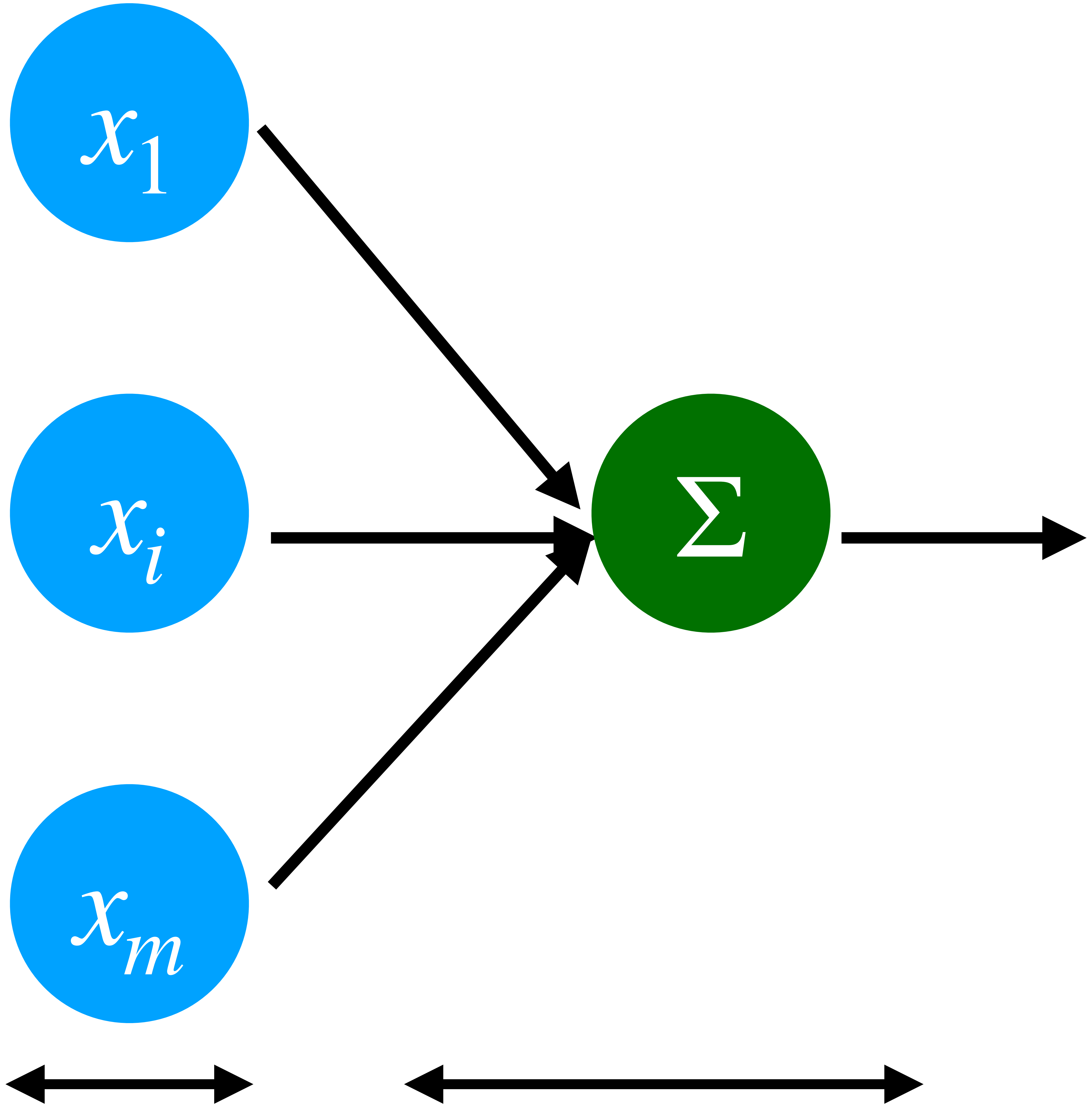
The Perceptron: the most basic neural network

The Perceptron: the most basic neural network



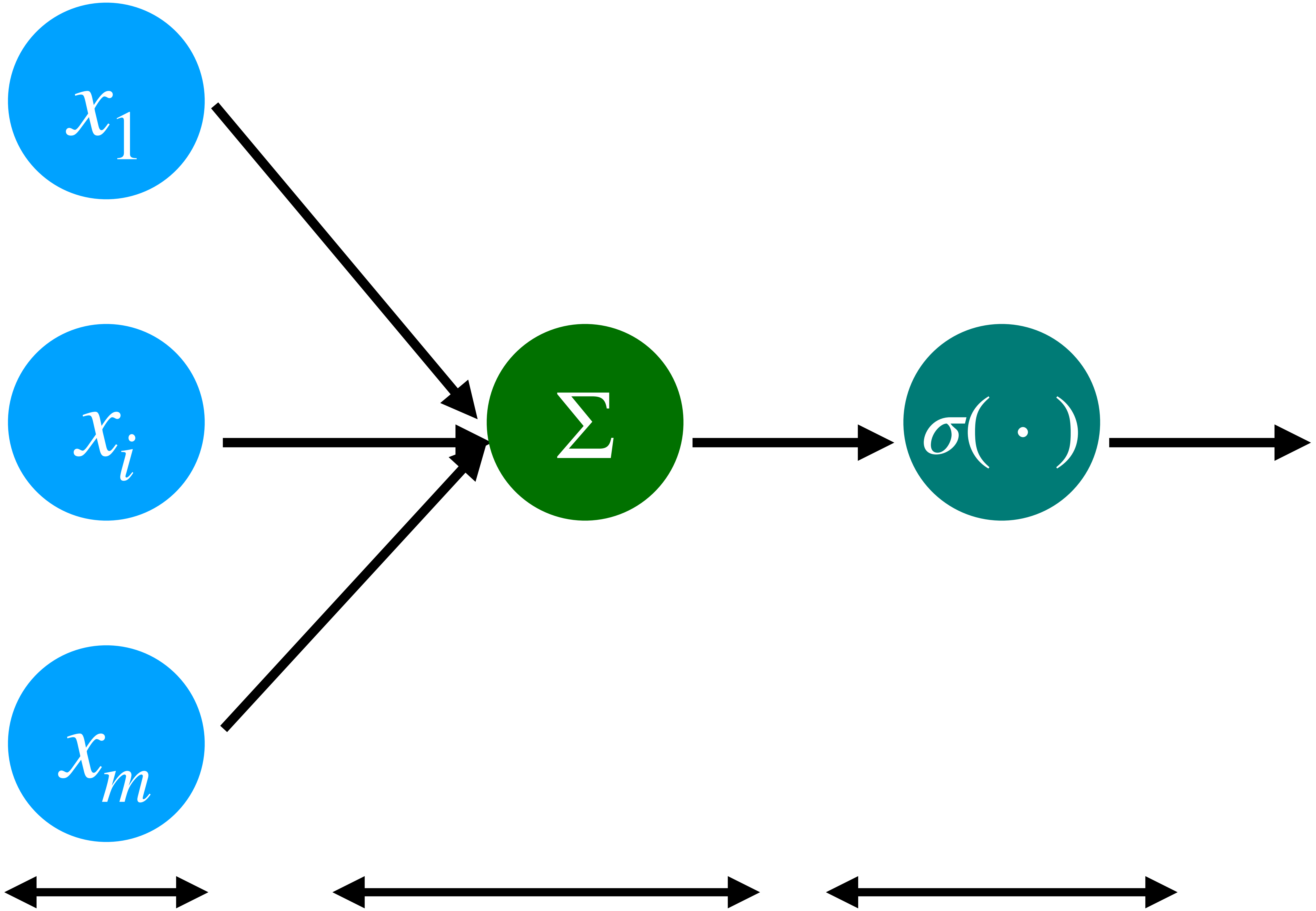
Input $\mathbf{x}$

The Perceptron: the most basic neural network



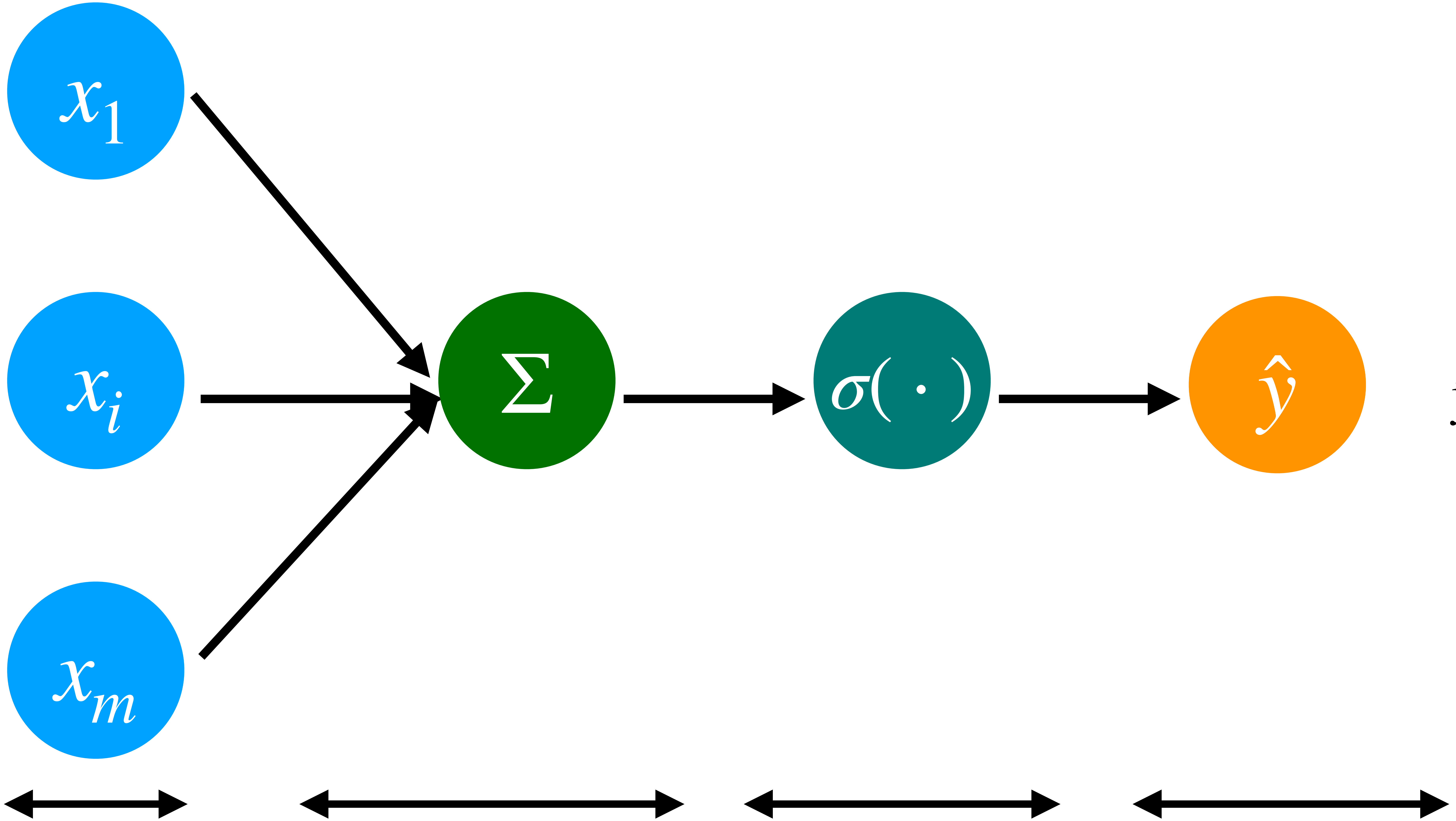
Input $\mathbf{x}$ **Weight combination**

The Perceptron: the most basic neural network



Input $\mathbf{x}$ **Weight combination** **Non - linearity**

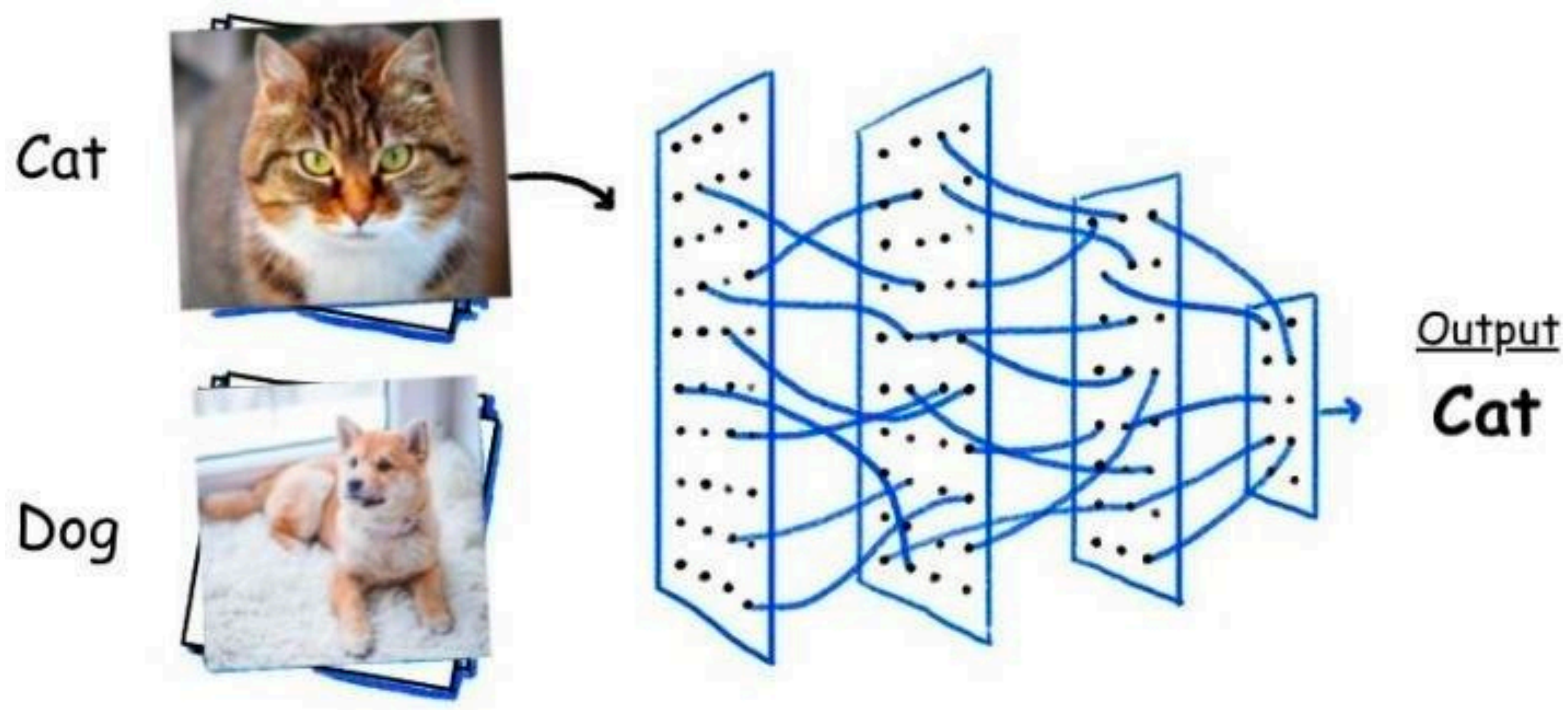
The Perceptron: the most basic neural network



$$\hat{y} = \sigma\left(\sum_{i=1}^{m} x_i \omega_i\right)$$

Input $\mathbf{x}$ | **Weight combination** | **Non - linearity** | **Output $y \in \mathscr{R}$**

Example of Inputs

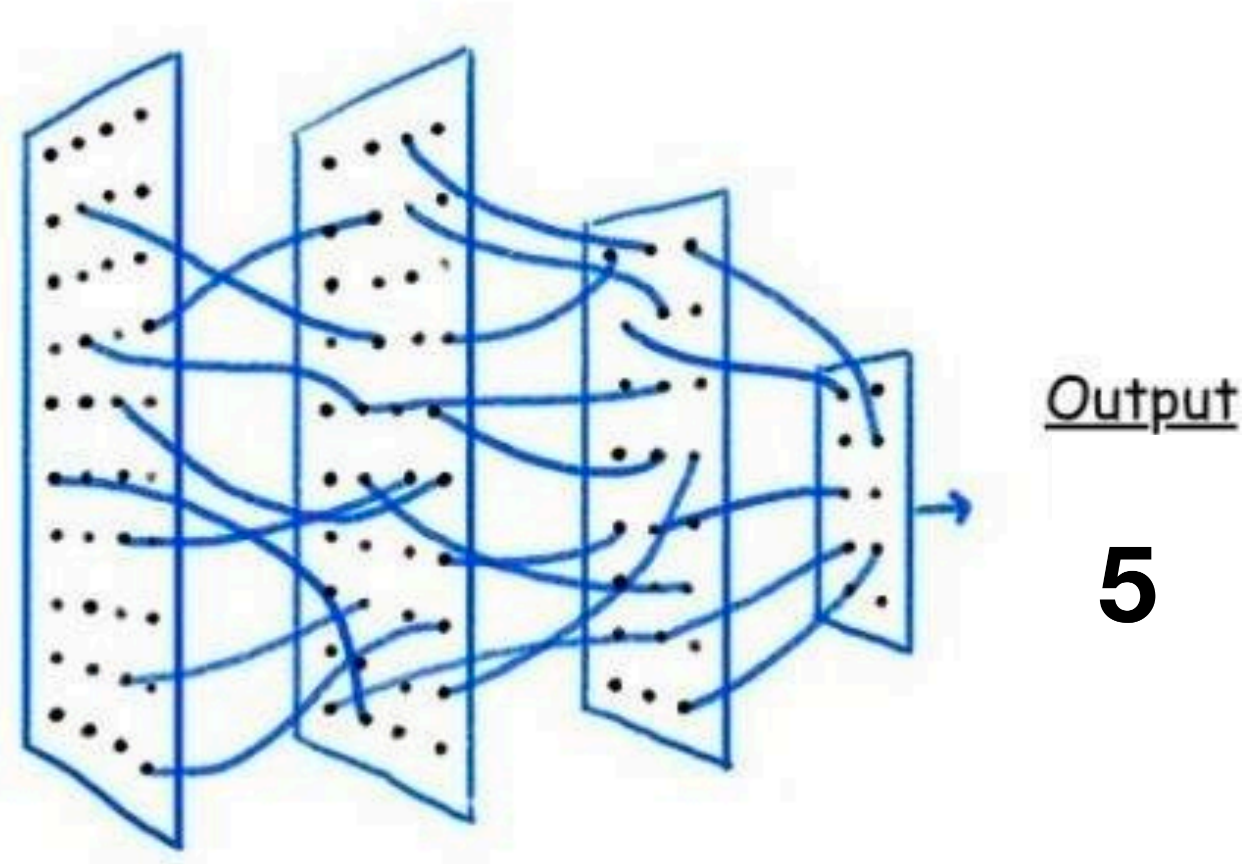


$x_1, \cdots, x_m$ corresponds to the pixels of the image.

$\hat{y}$ corresponds to the any scalar > 0 if the Input is an cat

corresponds to the any scalar < 0 if the Input is an dog

I really love my professor



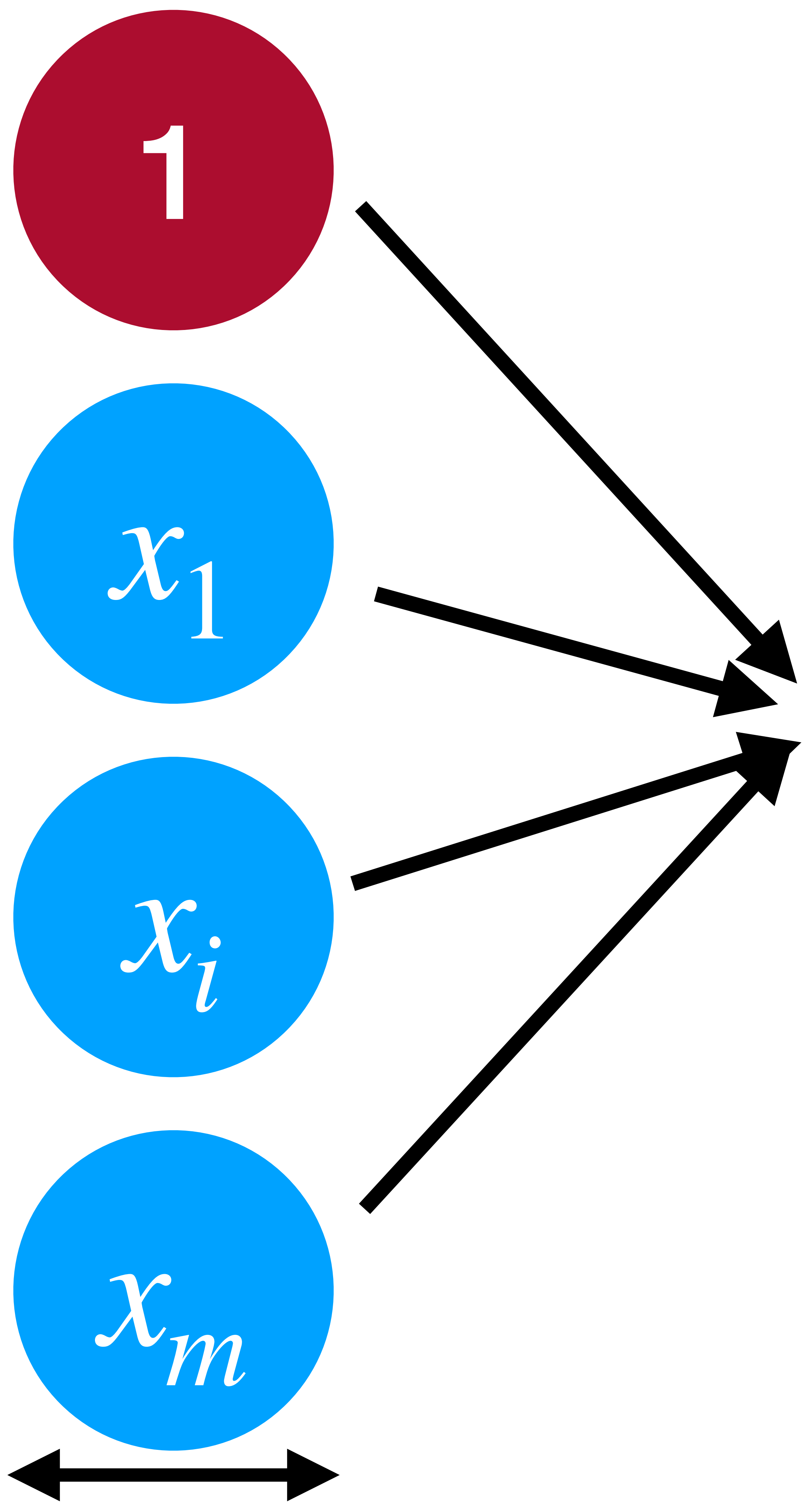
5

$x_1, \cdots, x_m$ corresponds to the index of the word in text

$\hat{y}$ corresponds to the any scalar between [0,5] and model the sentiment of the text.

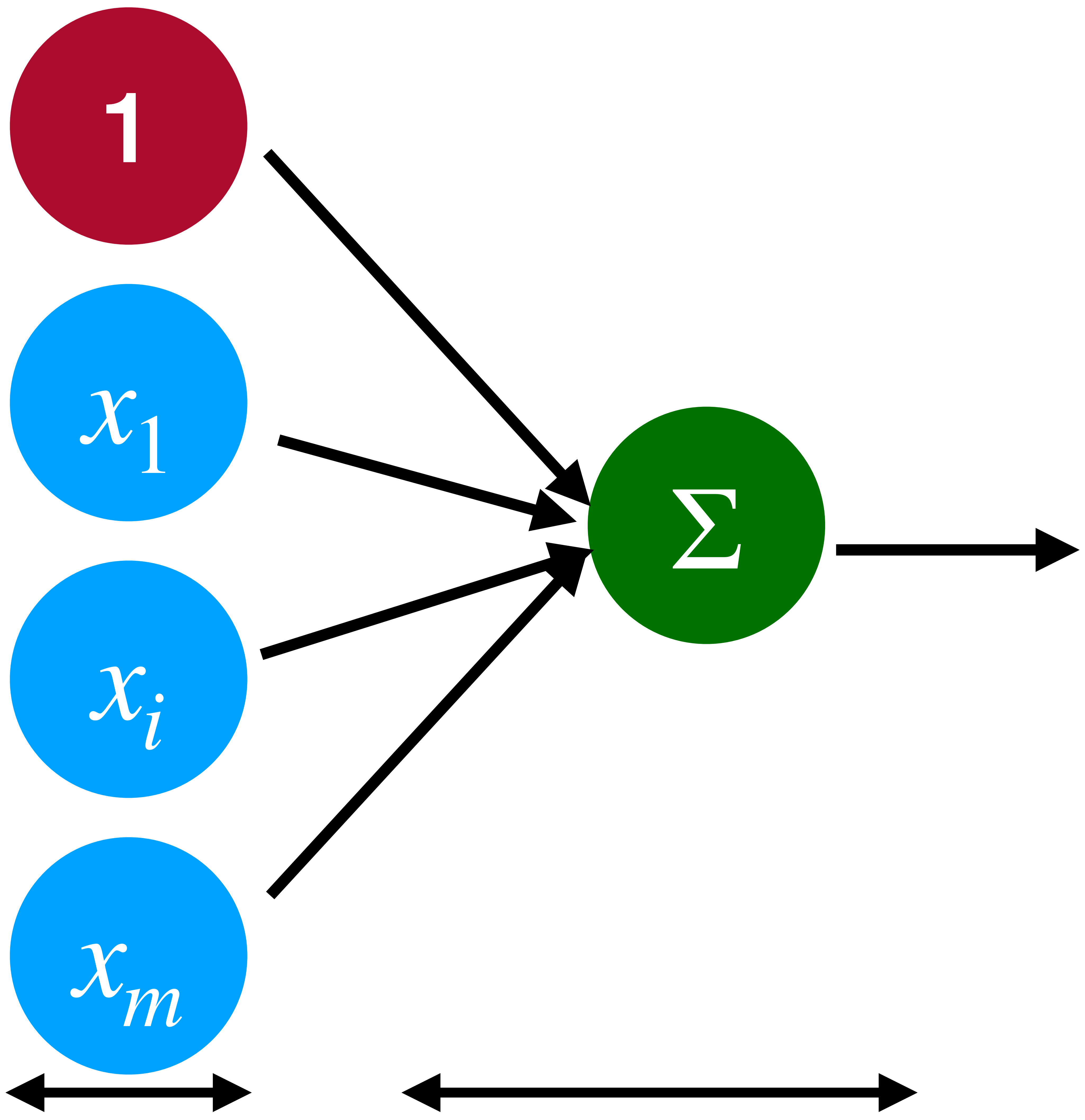
Do not forget the Biais for the forward propagation

Do not forget the Biais for the forward propagation



Input $\mathbf{x}$

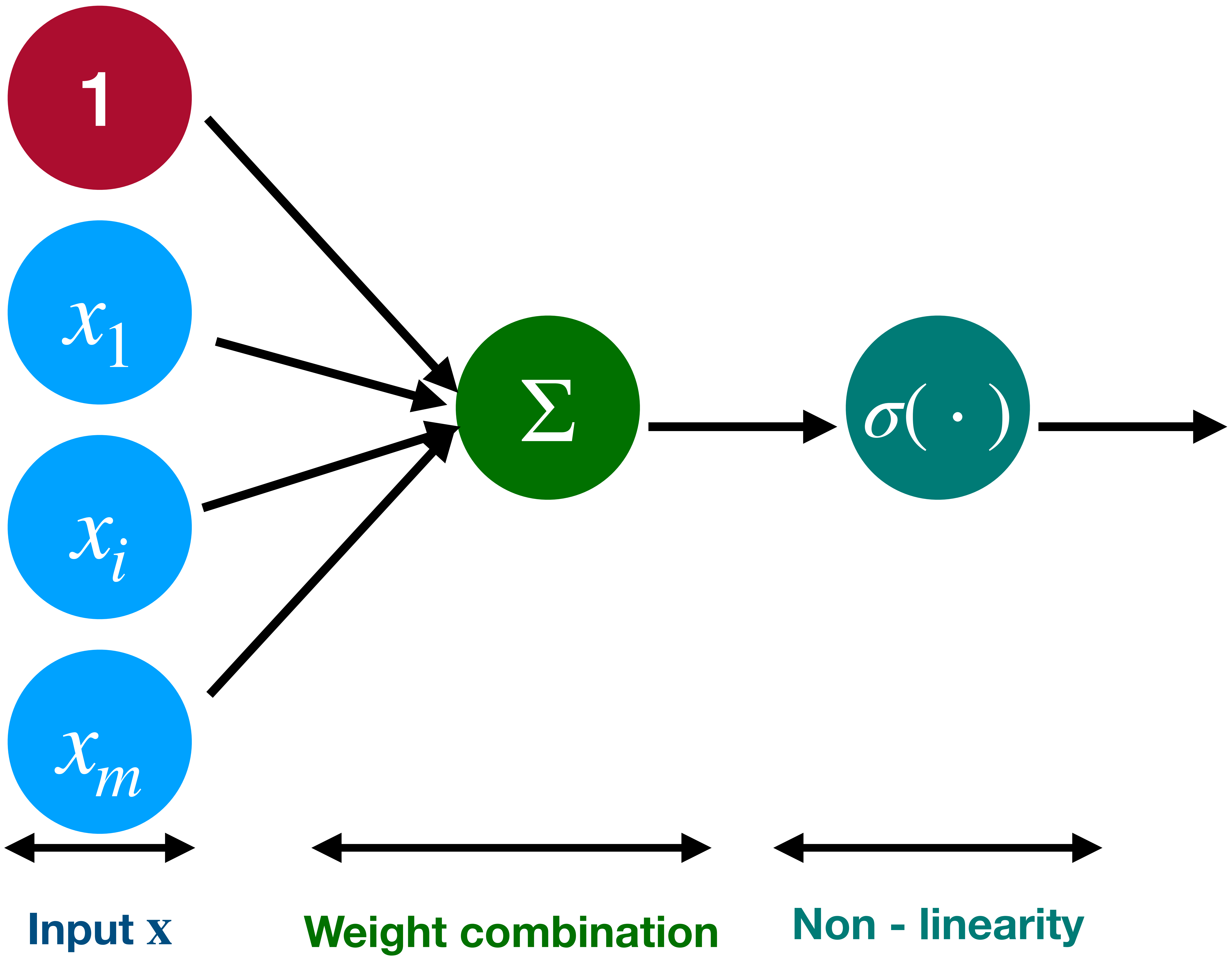
Do not forget the Biais for the forward propagation



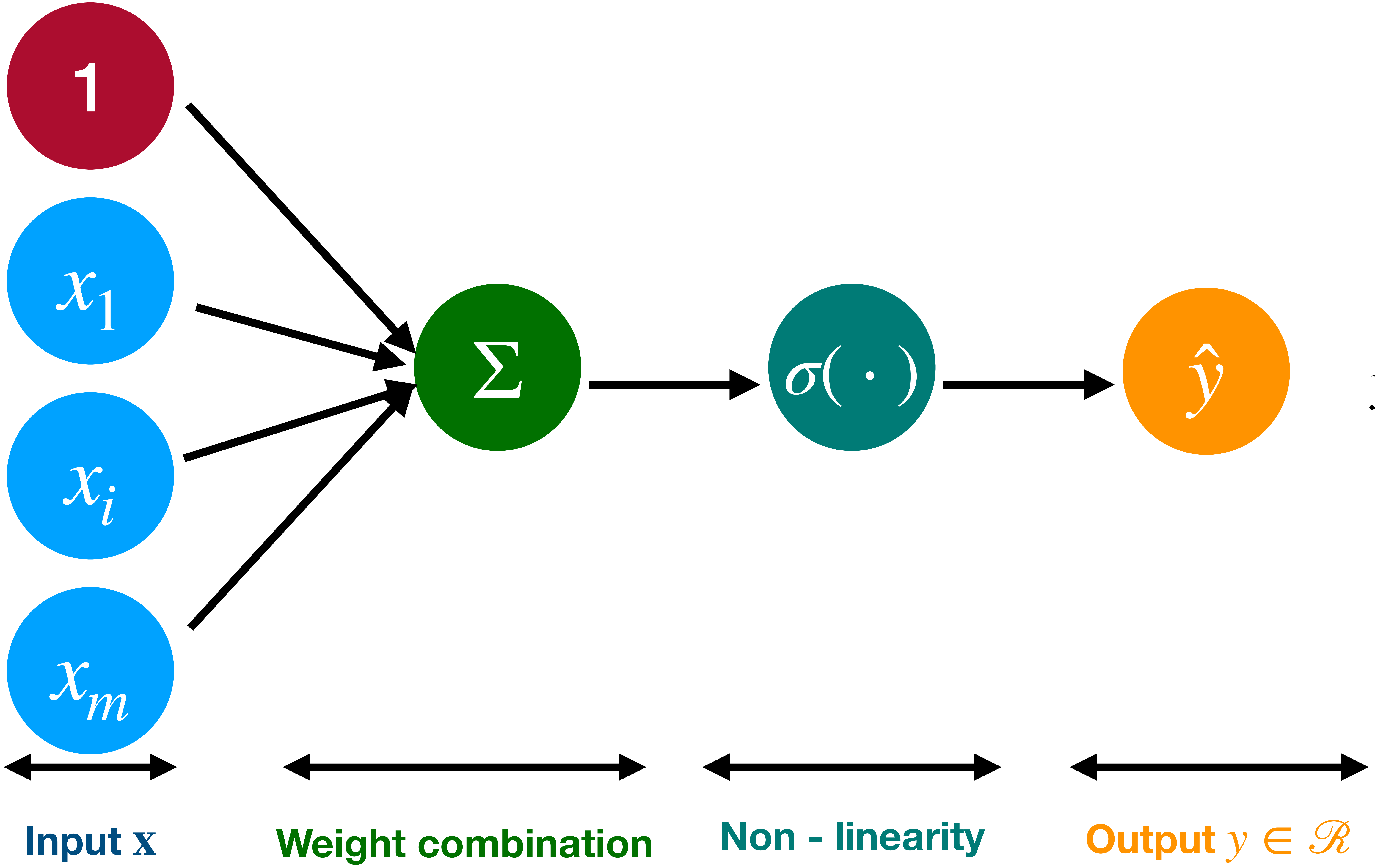
Input x

Weight combination

Do not forget the Biais for the forward propagation



Do not forget the Biais for the forward propagation



$$\hat{y} = \sigma\left(\sum_{i=0}^{m} x_i \omega_i\right)$$

Towards a Matrix Formulation

Towards a Matrix Formulation

Before

$$\hat{y} = \sigma\left(\sum_{i=0}^{m} x_i \omega_i\right)$$

Towards a Matrix Formulation

Before

$$\hat{y} = \sigma\left(\sum_{i=0}^{m} x_i \omega_i\right)$$

Matrix Formulation

$$\hat{y} = \sigma\left(\mathbf{X}^T \mathbf{W}\right)$$

Towards a Matrix Formulation

Before

$$\hat{y} = \sigma\left(\sum_{i=0}^{m} x_i \omega_i\right)$$

Matrix Formulation

$$\hat{y} = \sigma\left(\mathbf{X}^T \mathbf{W}\right)$$

Where $\mathbf{X} = [x_0, \cdots, x_m]$ $\mathbf{W} = [\omega_0, \cdots, \omega_m]$

Towards a Matrix Formulation

Before $$\hat{y} = \sigma\left(\sum_{i=0}^{m} x_i \omega_i\right)$$

Matrix Formulation $$\hat{y} = \sigma\left(\mathbf{X}^T\mathbf{W}\right) \longrightarrow$$ **Simple Dot Product !**

Where $$\mathbf{X} = [x_0, \cdots, x_m] \qquad \mathbf{W} = [\omega_0, \cdots, \omega_m]$$

Before

$$\hat{y} = \sigma\left(\sum_{i=0}^{m} x_i \omega_i\right)$$

Matrix Formulation

$$\hat{y} = \sigma\left(\mathbf{X}^T\mathbf{W}\right) \longrightarrow$$

Simple Dot Product !

Neural Network Weights

Where

$$\mathbf{X} = [x_0, \cdots, x_m]$$

$$\boxed{\mathbf{W} = [\omega_0, \cdots, \omega_m]}$$

Before $$\hat{y} = \sigma\left(\sum_{i=0}^{m} x_i \omega_i\right)$$

Matrix Formulation $$\hat{y} = \sigma\left(\mathbf{X}^T \mathbf{W}\right) \longrightarrow$$ **Simple Dot Product !**

Where $$\mathbf{X} = [x_0, \cdots, x_m]$$

Neural Network Weights

$$\boxed{\mathbf{W} = [\omega_0, \cdots, \omega_m]}$$

For an image of size H,W **we have X** $\in \mathscr{R}^{H \times W}$ **but** $\hat{y} \in \mathscr{R}$

Towards a Matrix Formulation

Before $$\hat{y} = \sigma\left(\sum_{i=0}^{m} x_i \omega_i\right)$$

Matrix Formulation $$\hat{y} = \sigma\left(\mathbf{X}^T \mathbf{W}\right) \longrightarrow$$ **Simple Dot Product !**

Neural Network Weights

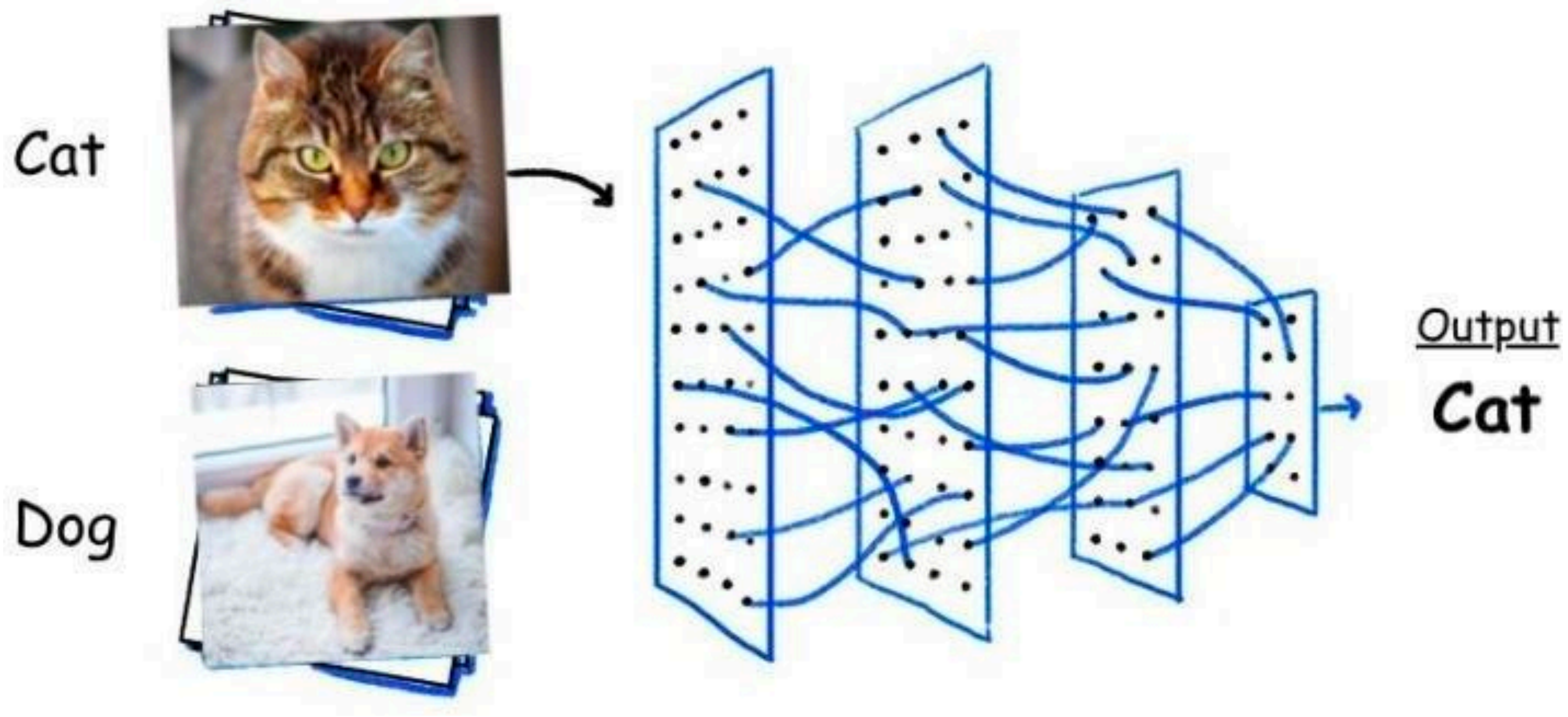
Where $$\mathbf{X} = [x_0, \cdots, x_m]$$ $$\boxed{\mathbf{W} = [\omega_0, \cdots, \omega_m]}$$

For an image of size H,W **we have X** $\in \mathcal{R}^{H \times W}$ **but** $\hat{y} \in \mathcal{R}$

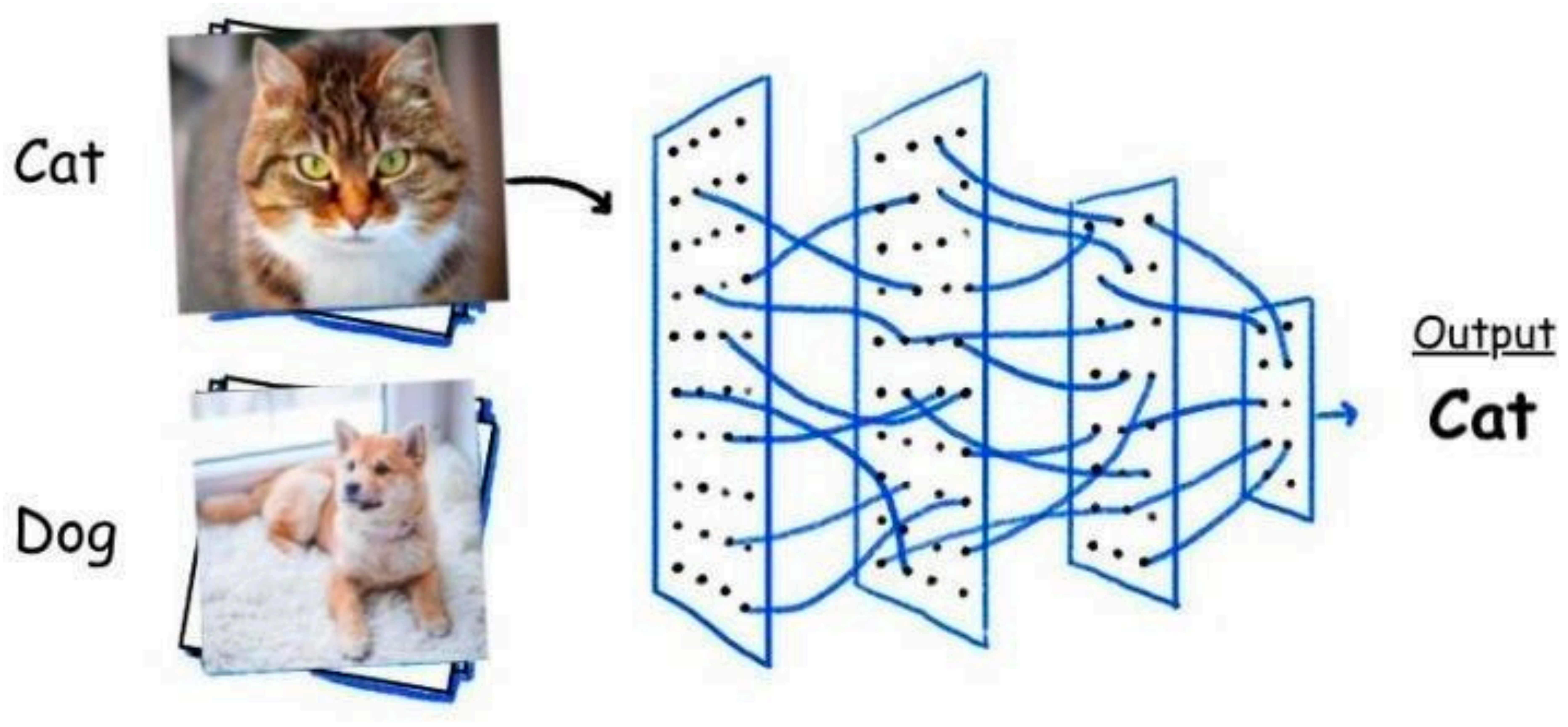
Be aware !!!!!! X $\in \mathcal{R}^d$ **but** $\hat{y} \in \mathcal{R}$

Let’s go back to cat and dog classification

Let’s go back to cat and dog classification



Let's go back to cat and dog classification



For an image of size H,W **we have** $\mathbf{X} \in \mathcal{R}^{H \times W}$ **but** $\hat{y} \in [0,1]$

Let’s sum up so far

Let’s sum up so far

1. The basic bloc to build a Neural Network

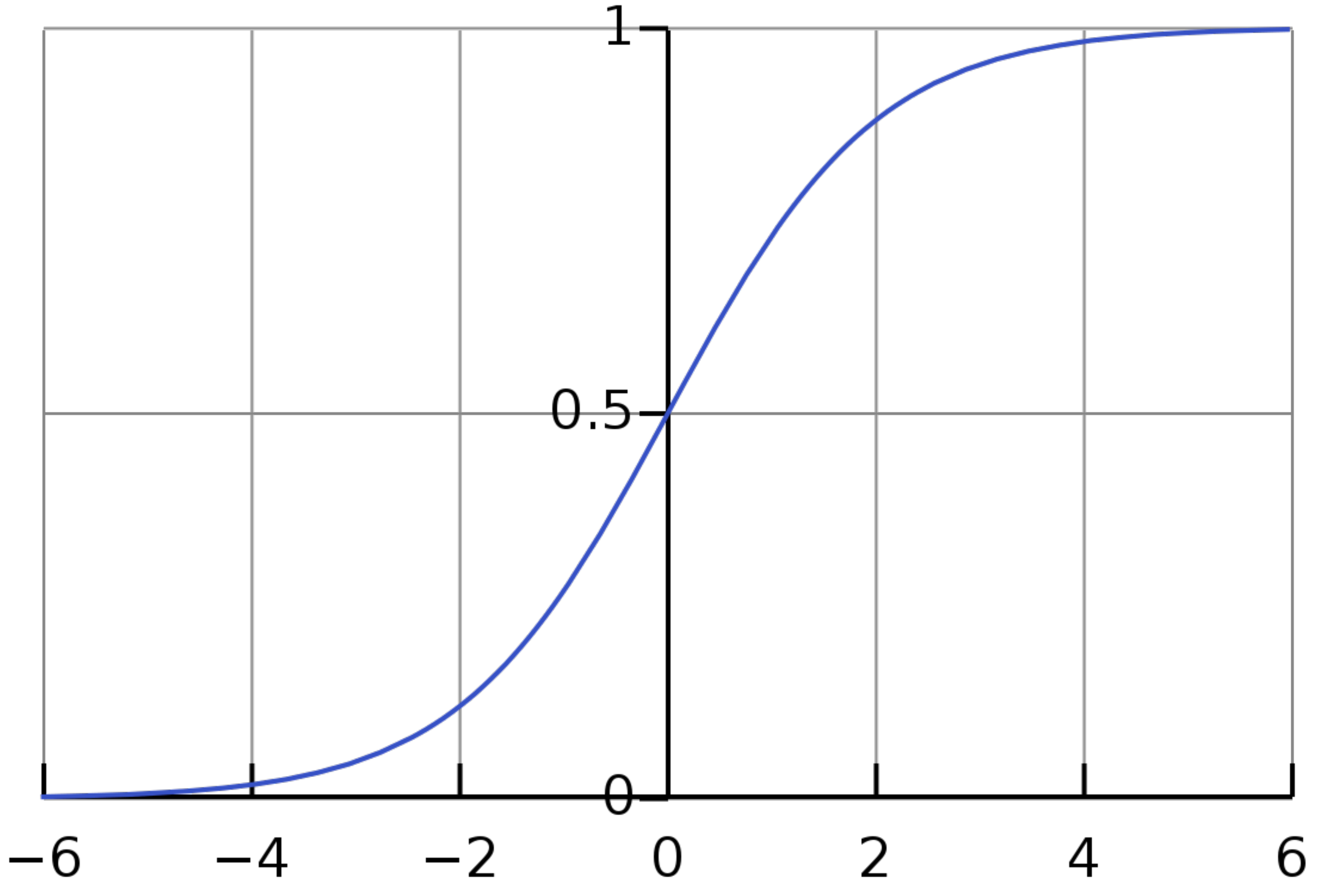
We have seen the forward propagation, i.e. from an input how to use the Neural Network

This operation is call forward propagation, this operation is used at inference time

What are the most common activation functions?

What are the most common activation functions?

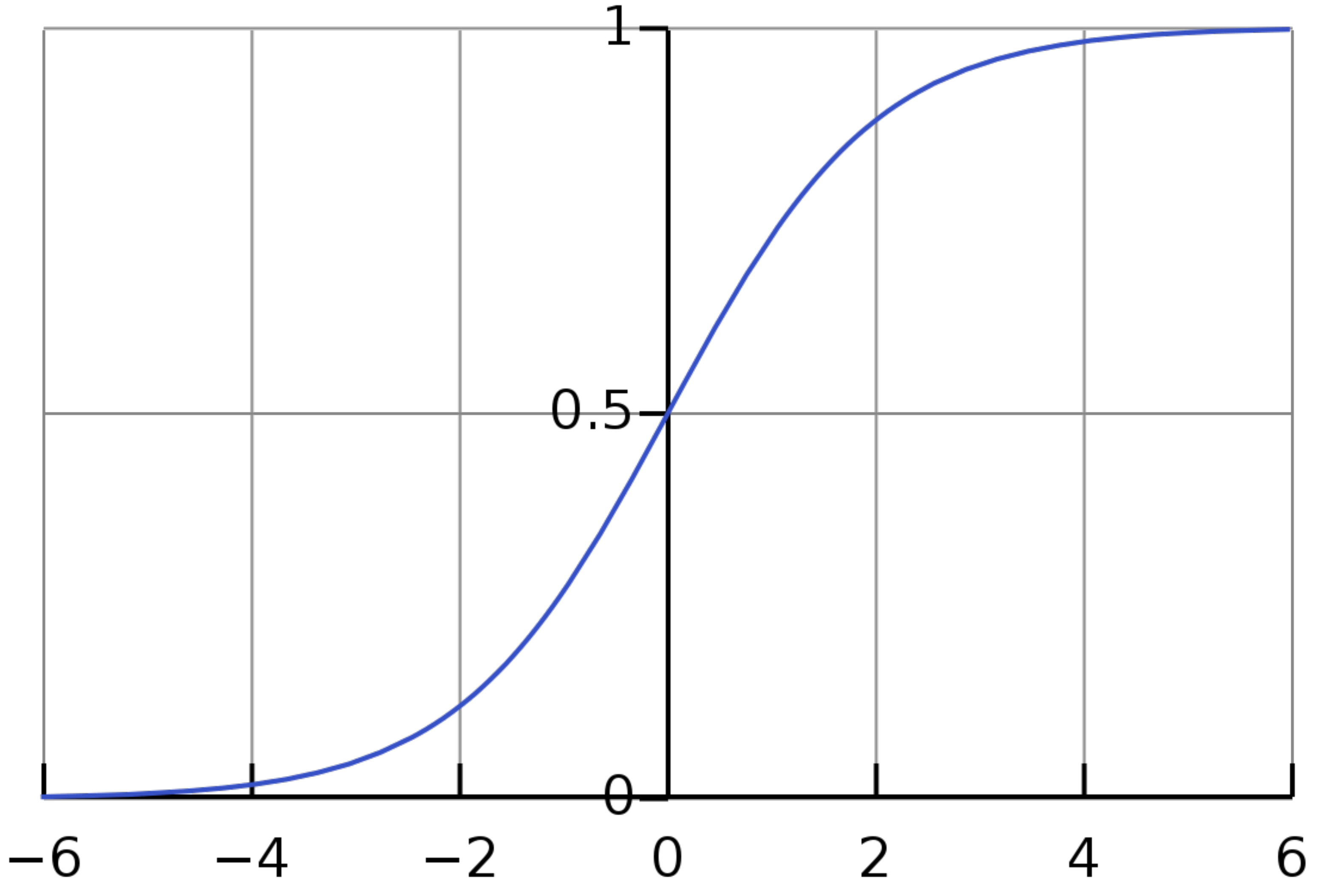
Sigmoid



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

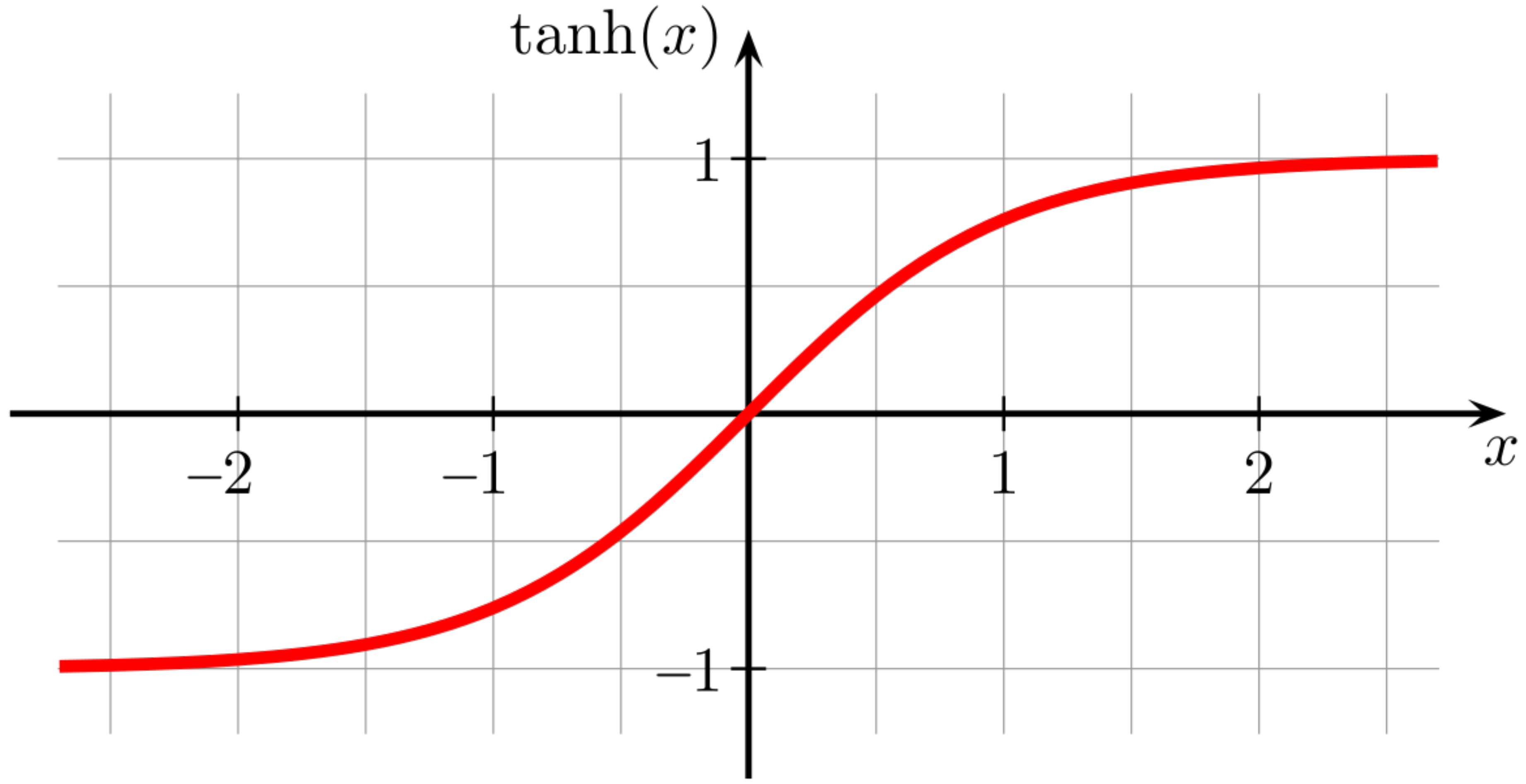
What are the most common activation functions?

Sigmoid



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

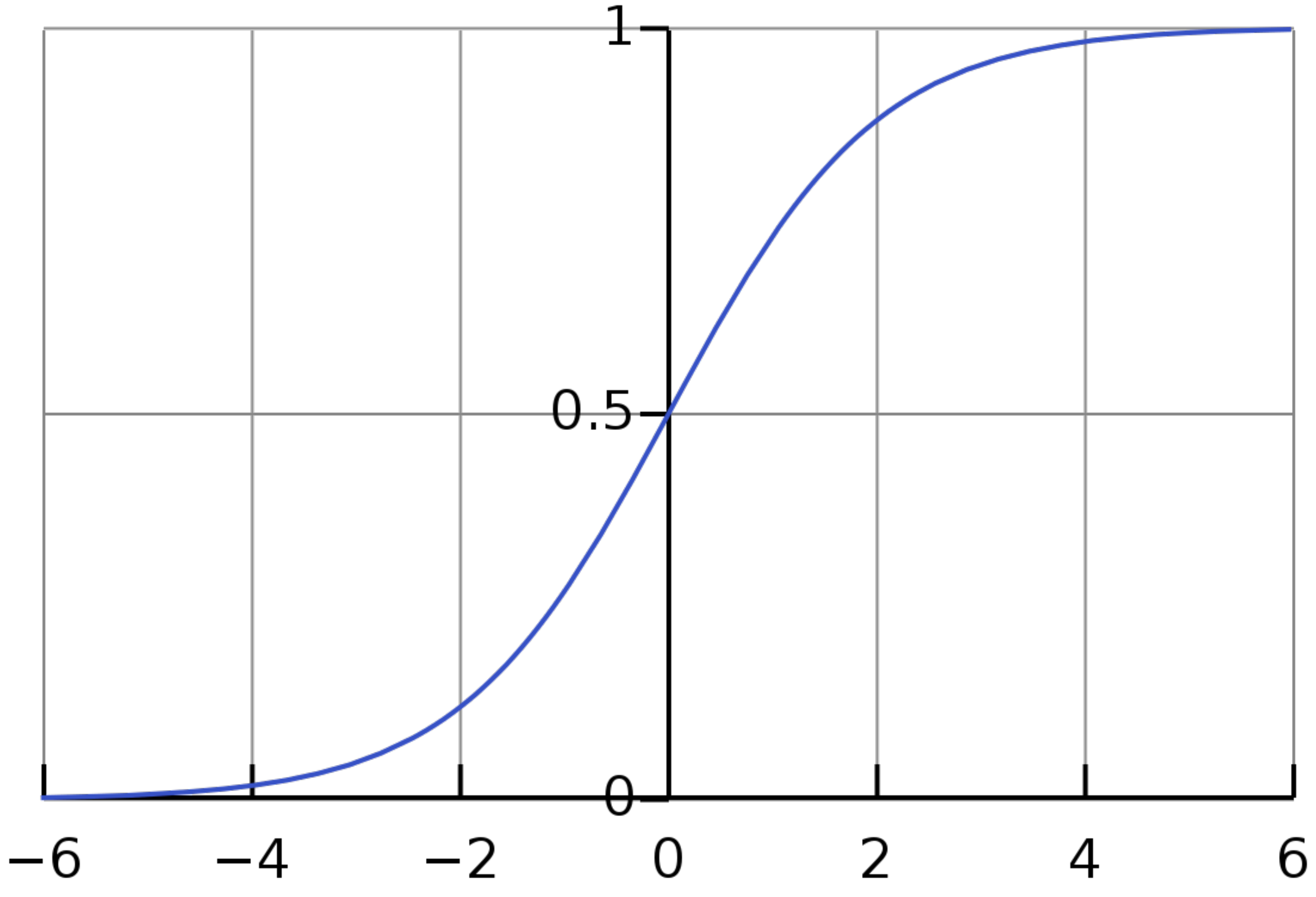
Tanh



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

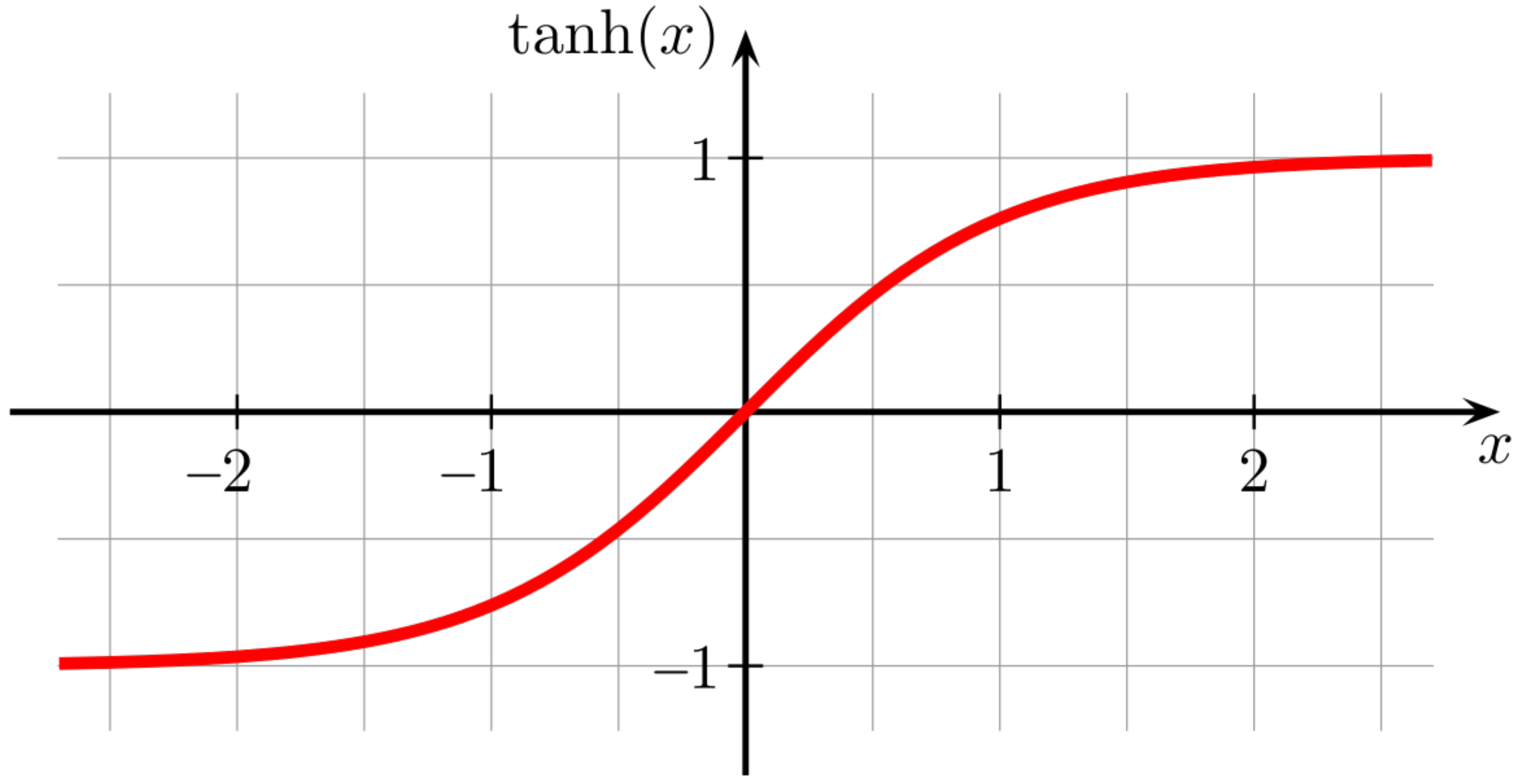
What are the most common activation functions?

Sigmoid



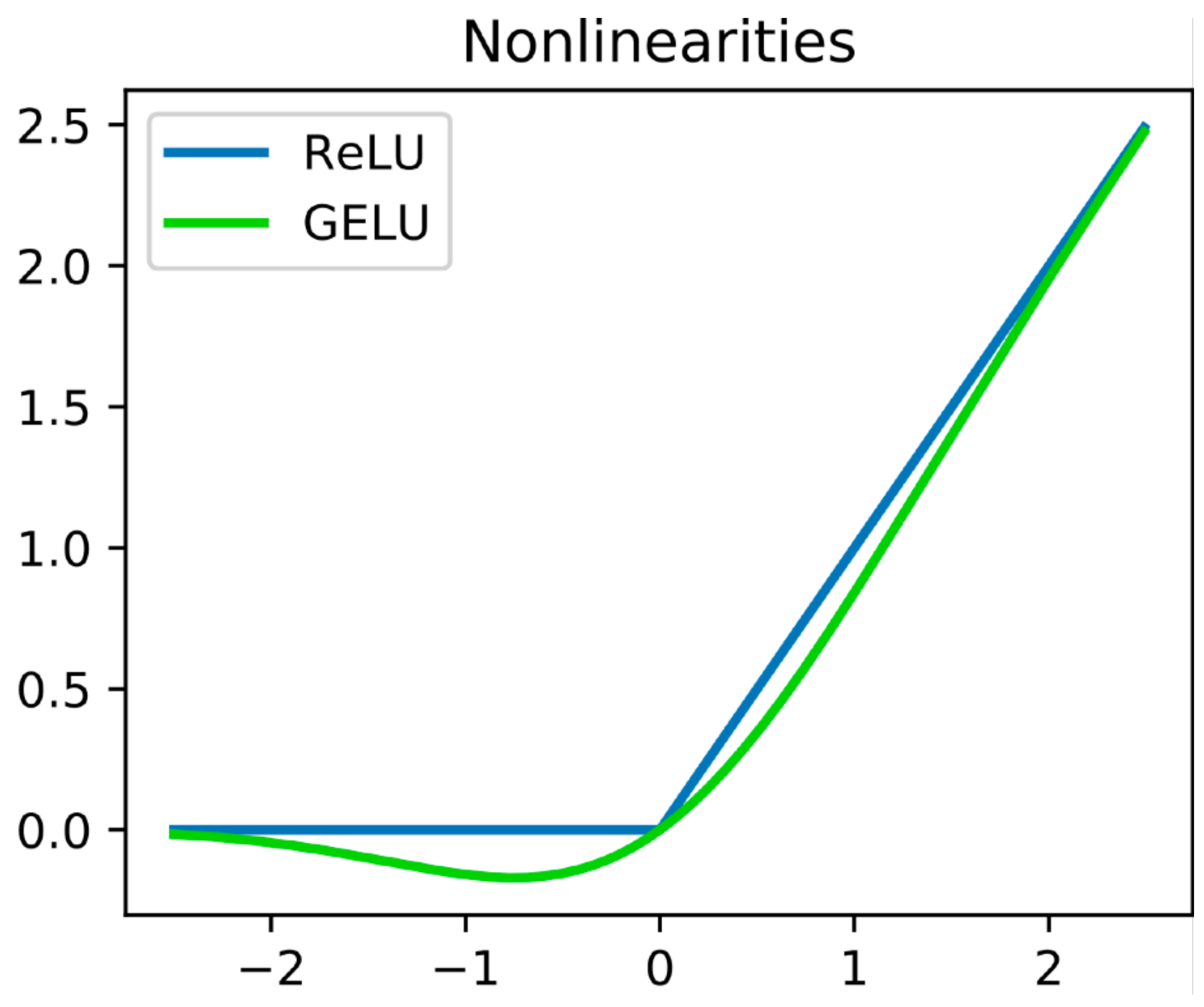
$$\sigma(x) = \frac{1}{1 + e^{x}}$$

Tanh



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

ReLU/GeLU

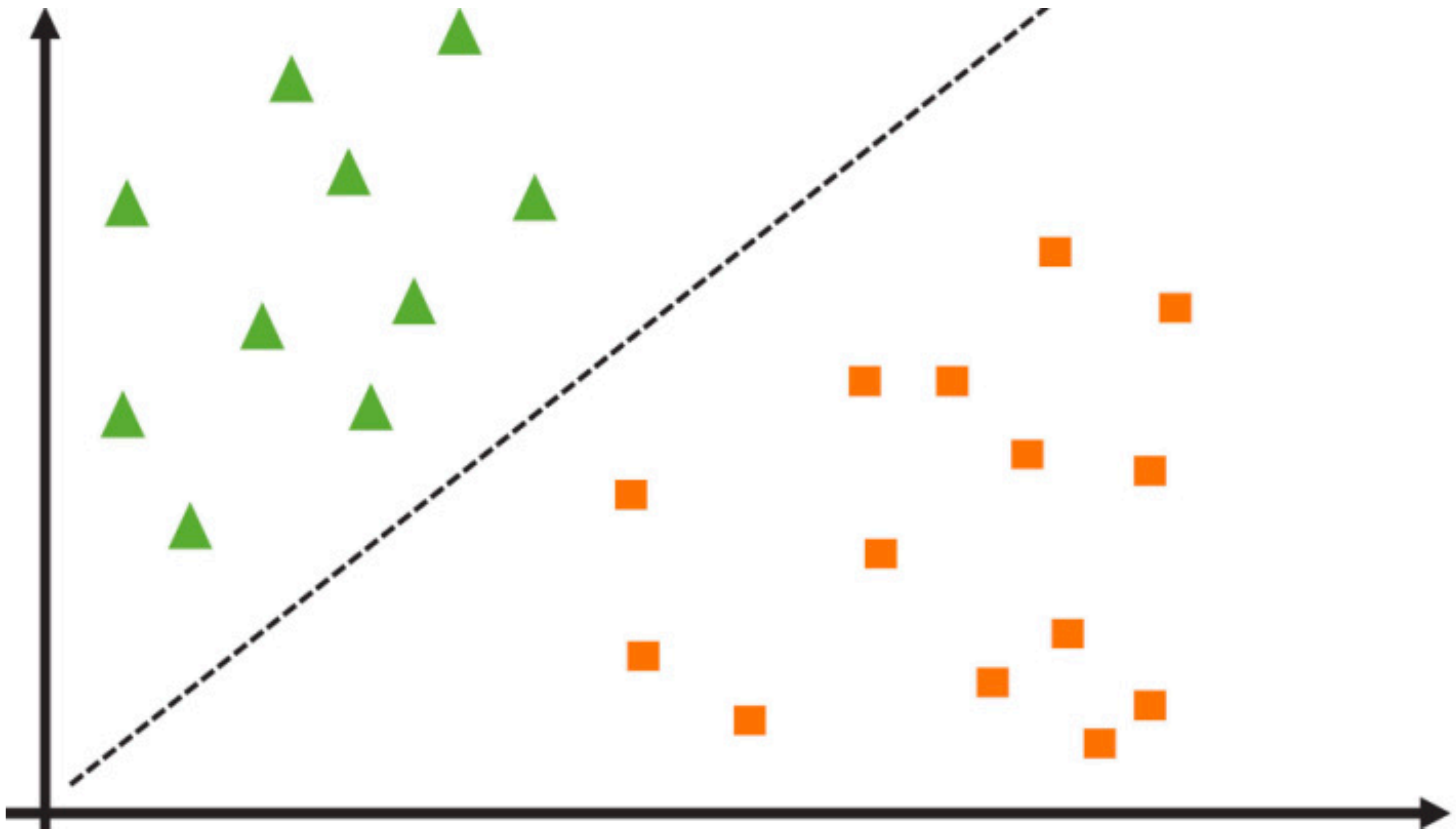


$$\sigma(x) = \max(0,x)$$

Why should I use activations functions?

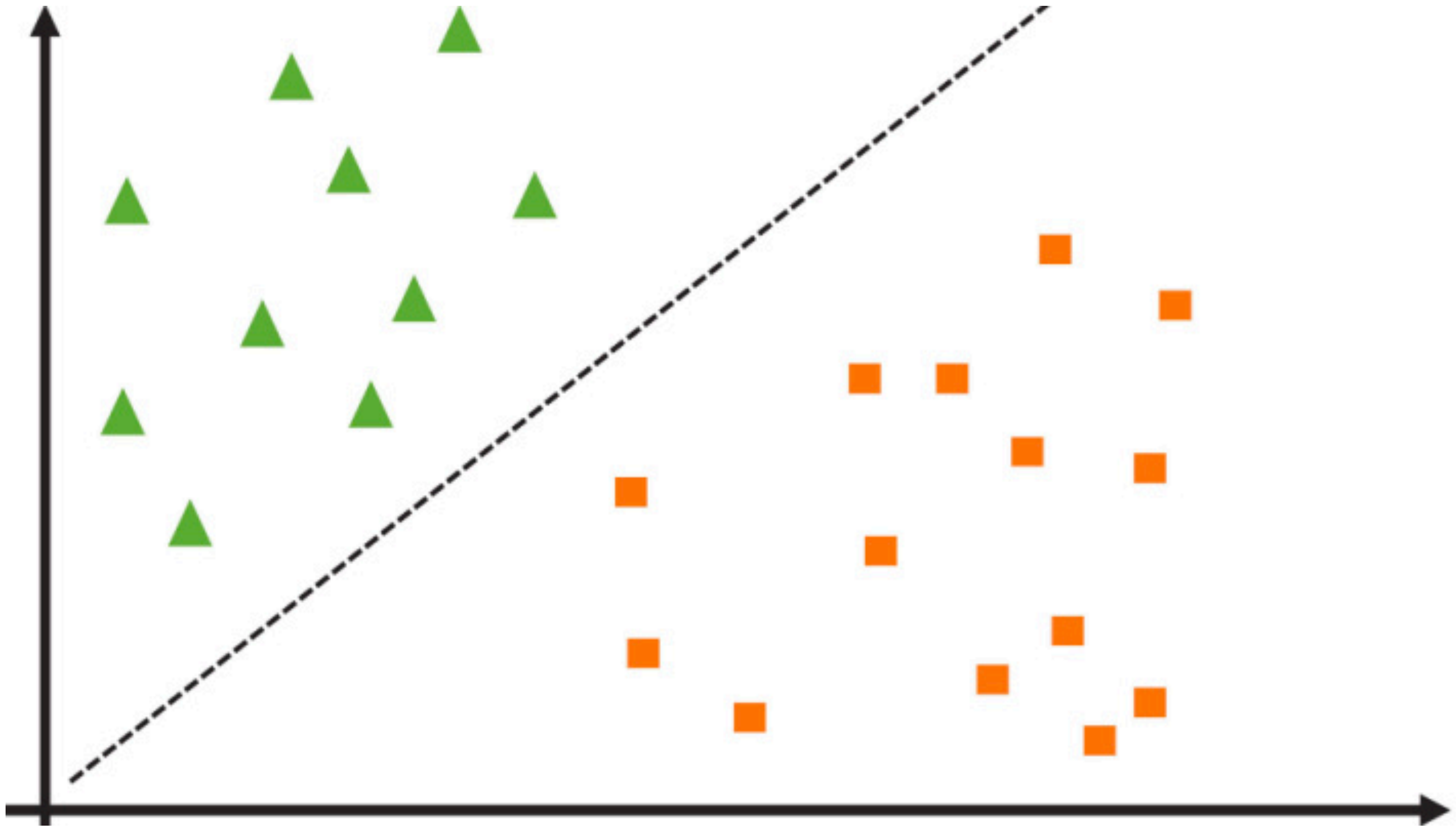
Why should I use activations functions?

Suppose your goal is to separate/ classify, the green triangles from the orange squares…..



Why should I use activations functions?

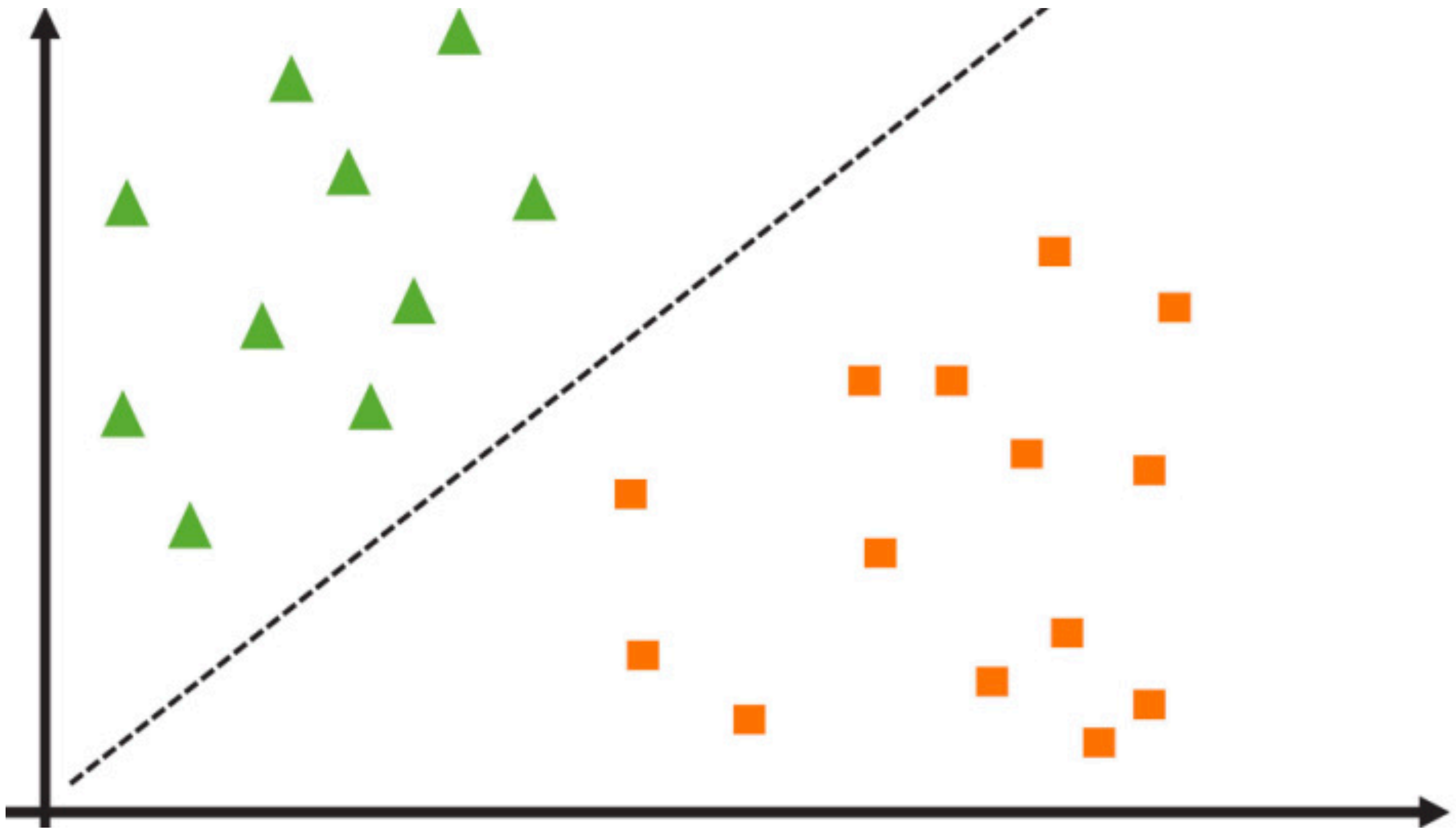
Suppose your goal is to separate/ classify, the green triangles from the orange squares…..



If you do not use linearity you decision boundary will an hyperplan

Why should I use activations functions?

Suppose your goal is to separate/classify, the green triangles from the orange squares…..



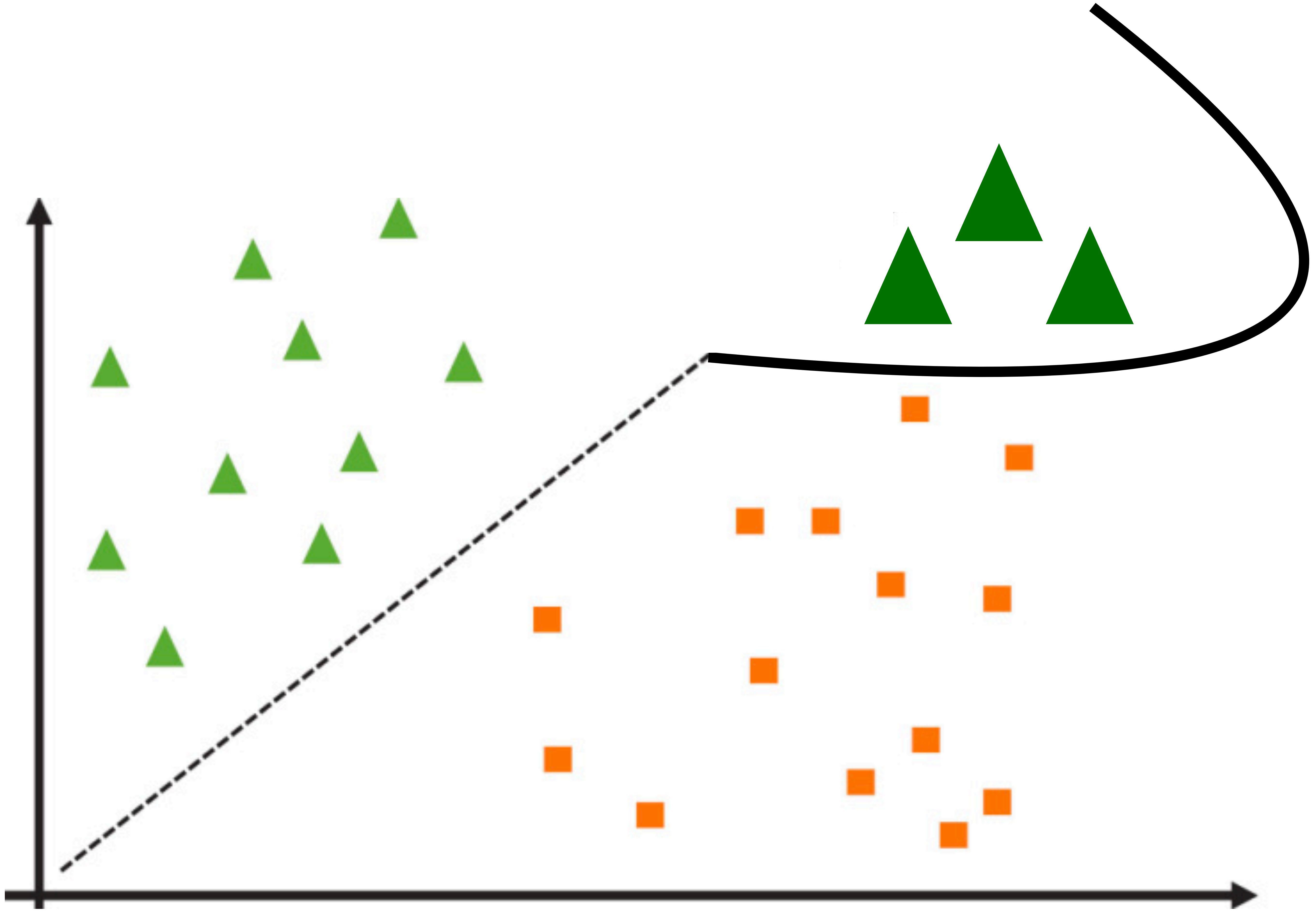
If you do not use linearity
you decision
boundary will an hyperplan

In this example 2D points represent the input (e.g. images/text)

Why should I use activations functions?

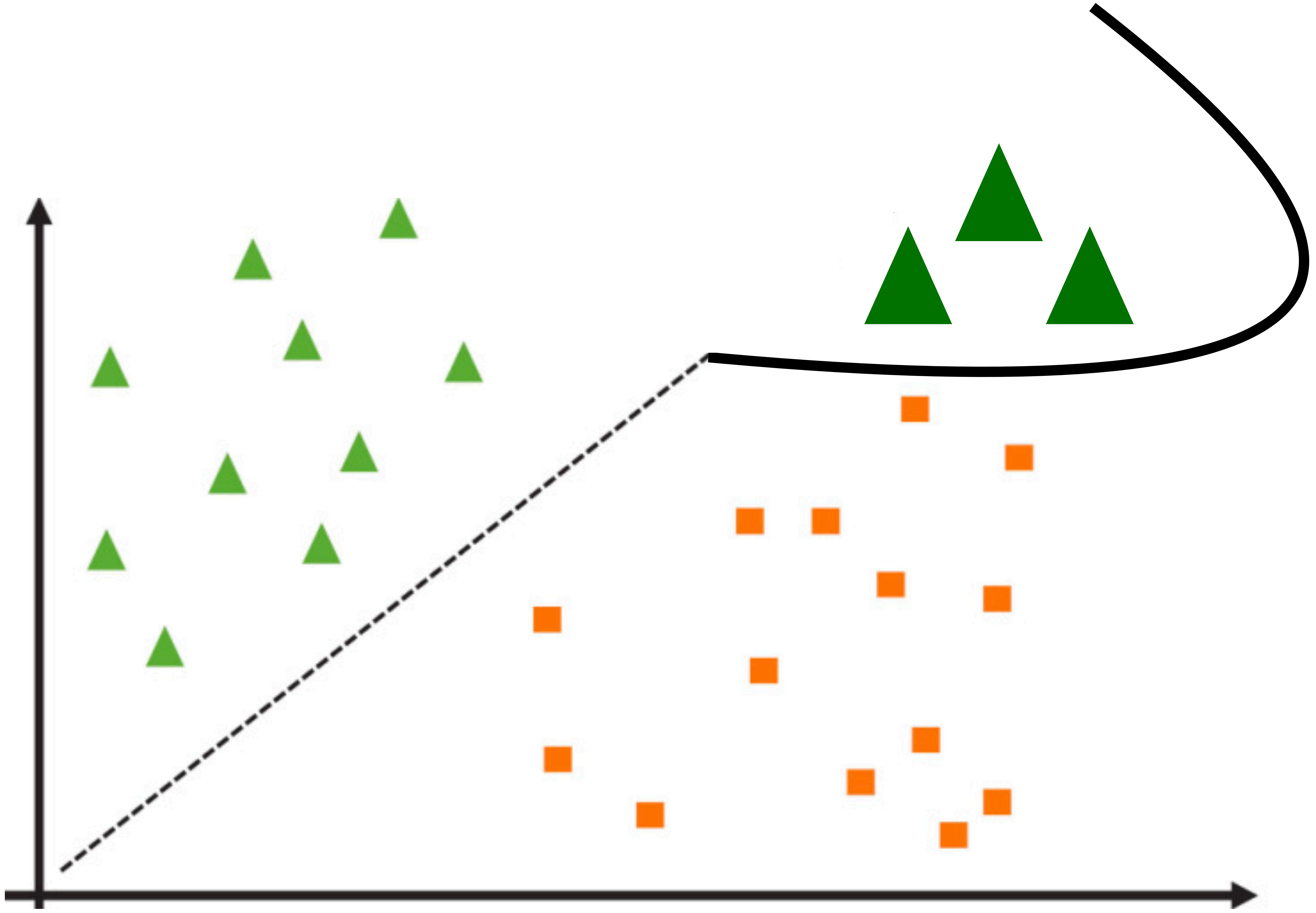
Why should I use activations functions?

Suppose your goal is to separate, the green triangles from the orange squares…..



Why should I use activations functions?

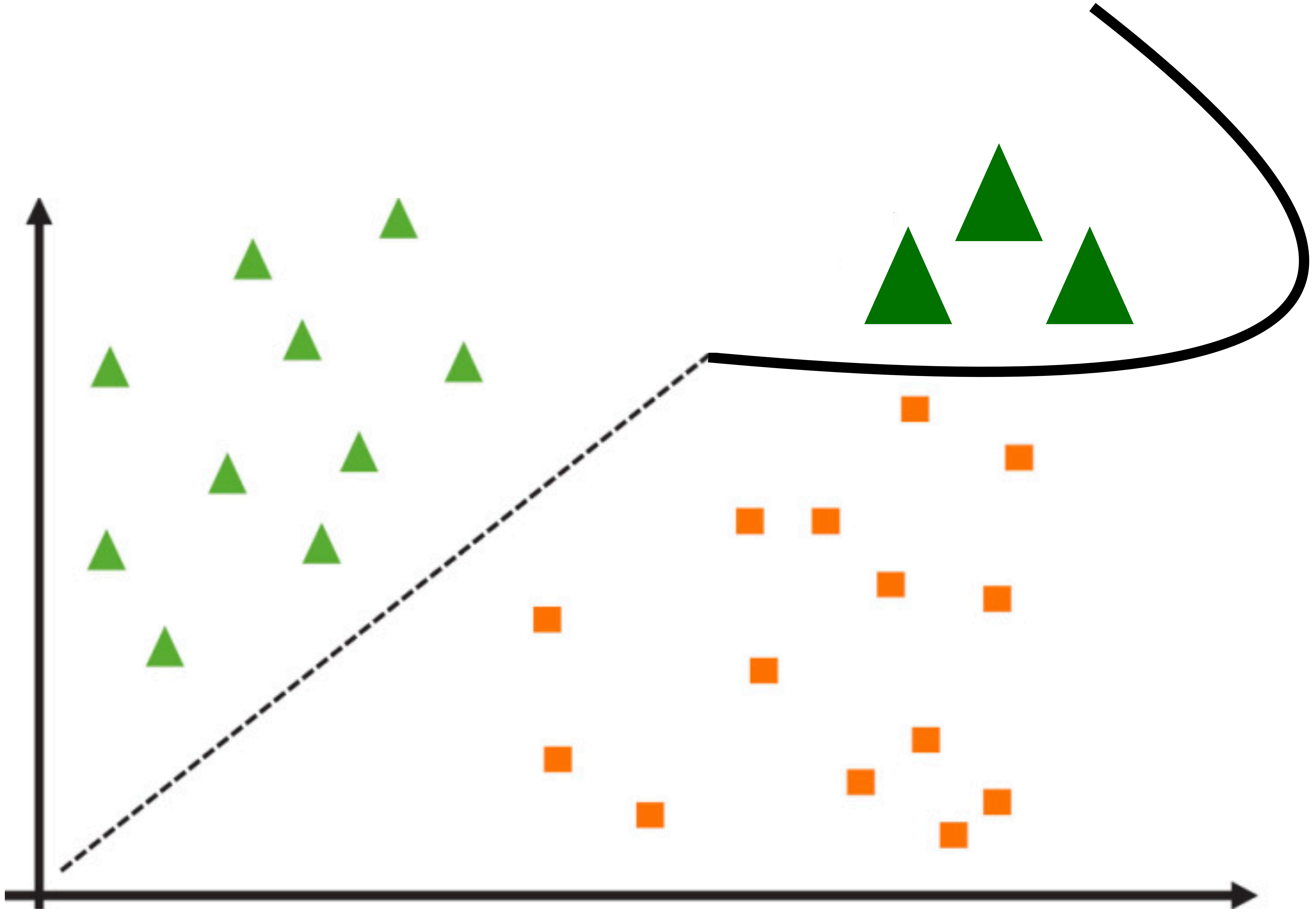
Suppose your goal is to separate, the green triangles from the orange squares…..



Thanks to the linearity you can have more « complex decision functions ».

Why should I use activations functions?

Suppose your goal is to separate, the green triangles from the orange squares…..



Thanks to the linearity you can have more « complex decision functions ».

See universal approximation Theorem.

Let’s sum up so far

Let’s sum up so far

1. The basic bloc to build a Neural Network

We have seen the forward propagation, i.e. from an input how to use the Neural Network

This operation is call forward propagation, this operation is used at inference time

Let's sum up so far

1. The basic bloc to build a Neural Network

We have seen the forward propagation, i.e. from an input how to use the Neural Network

This operation is call forward propagation, this operation is used at inference time

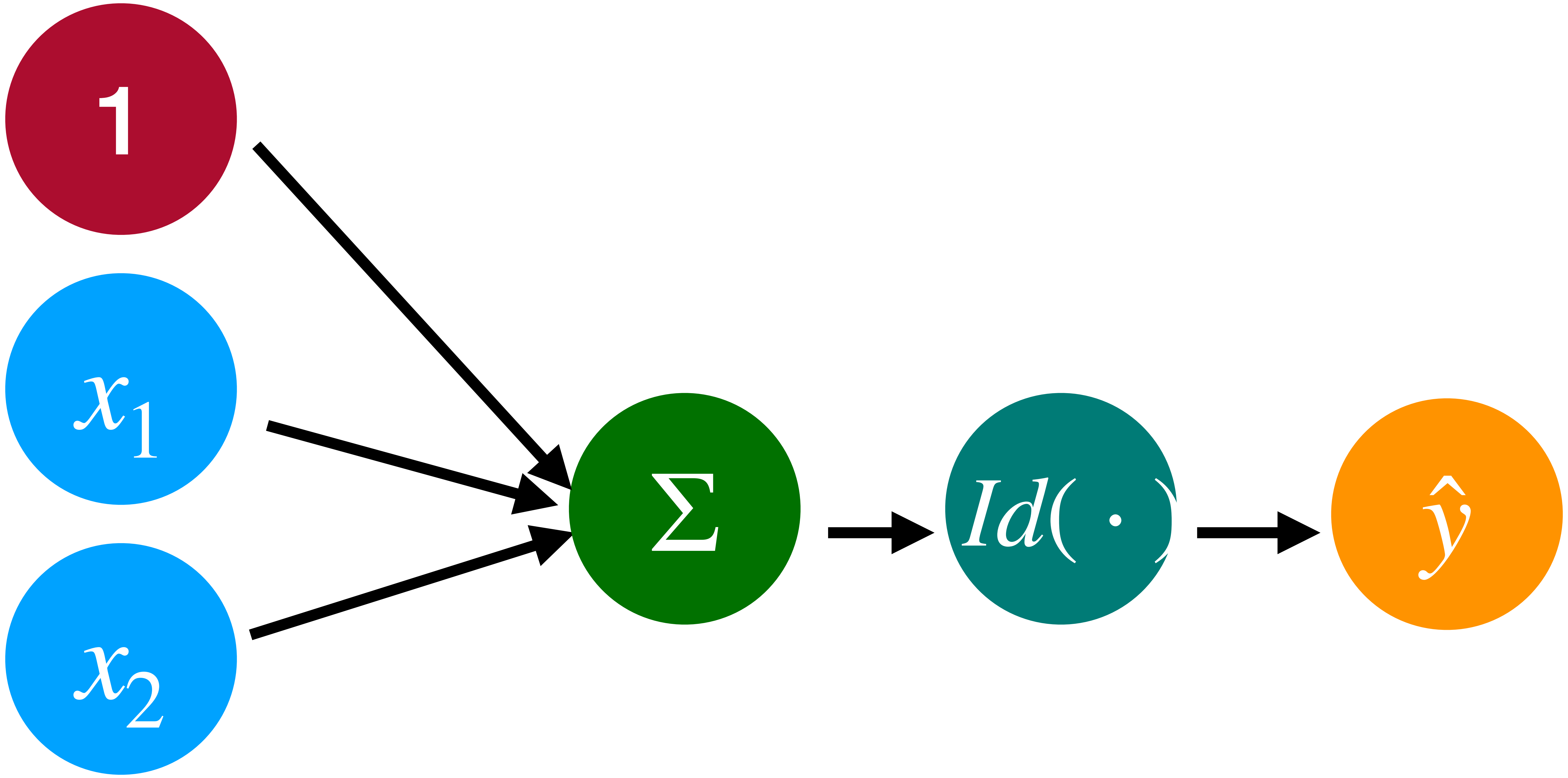
2. The role of activation functions

They complexity the neural network and allow to fit more complex data

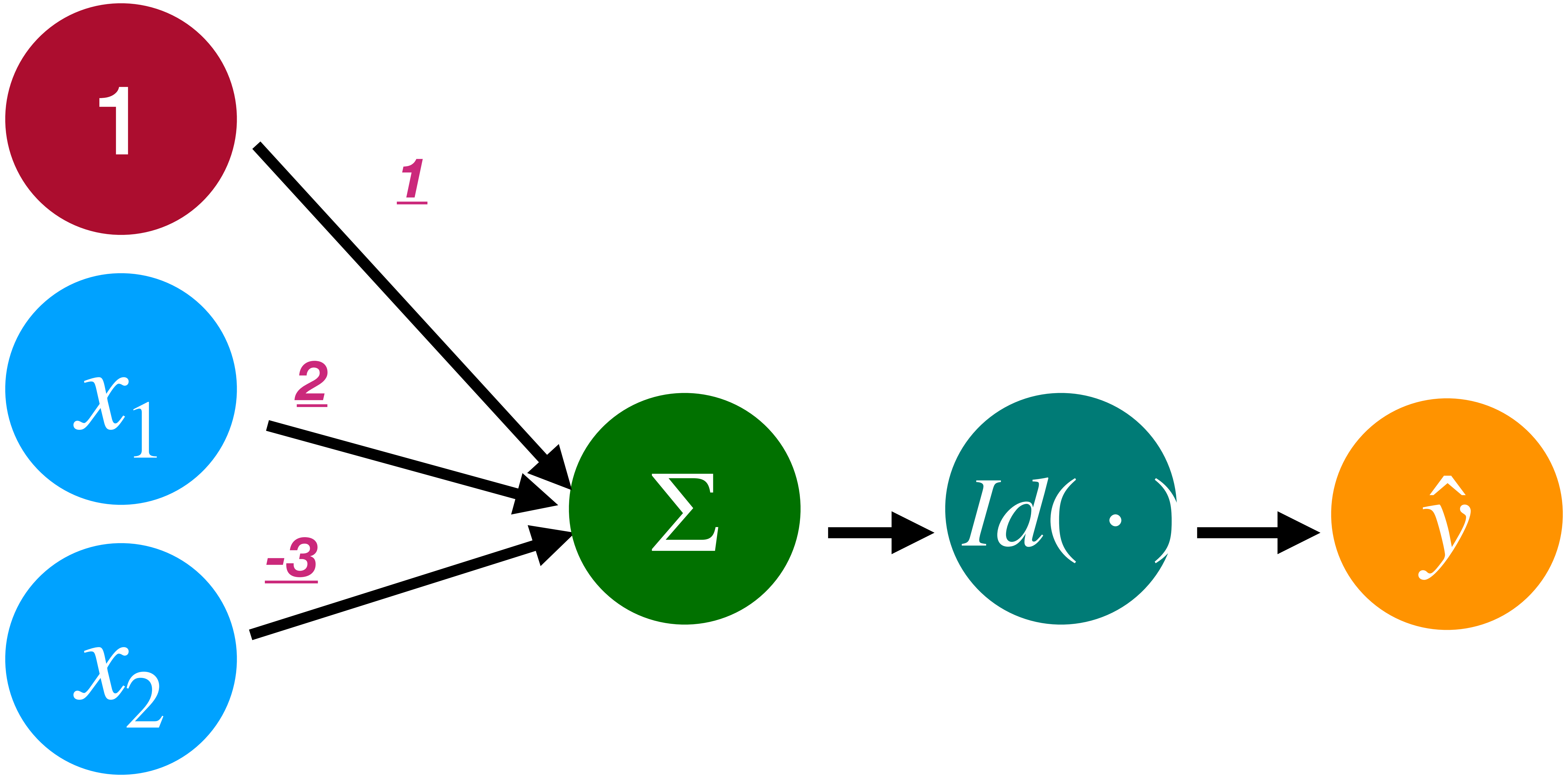
They are added at the end of each layer and there is a large variety of them

Our Perceptron to distinguish triangles and square

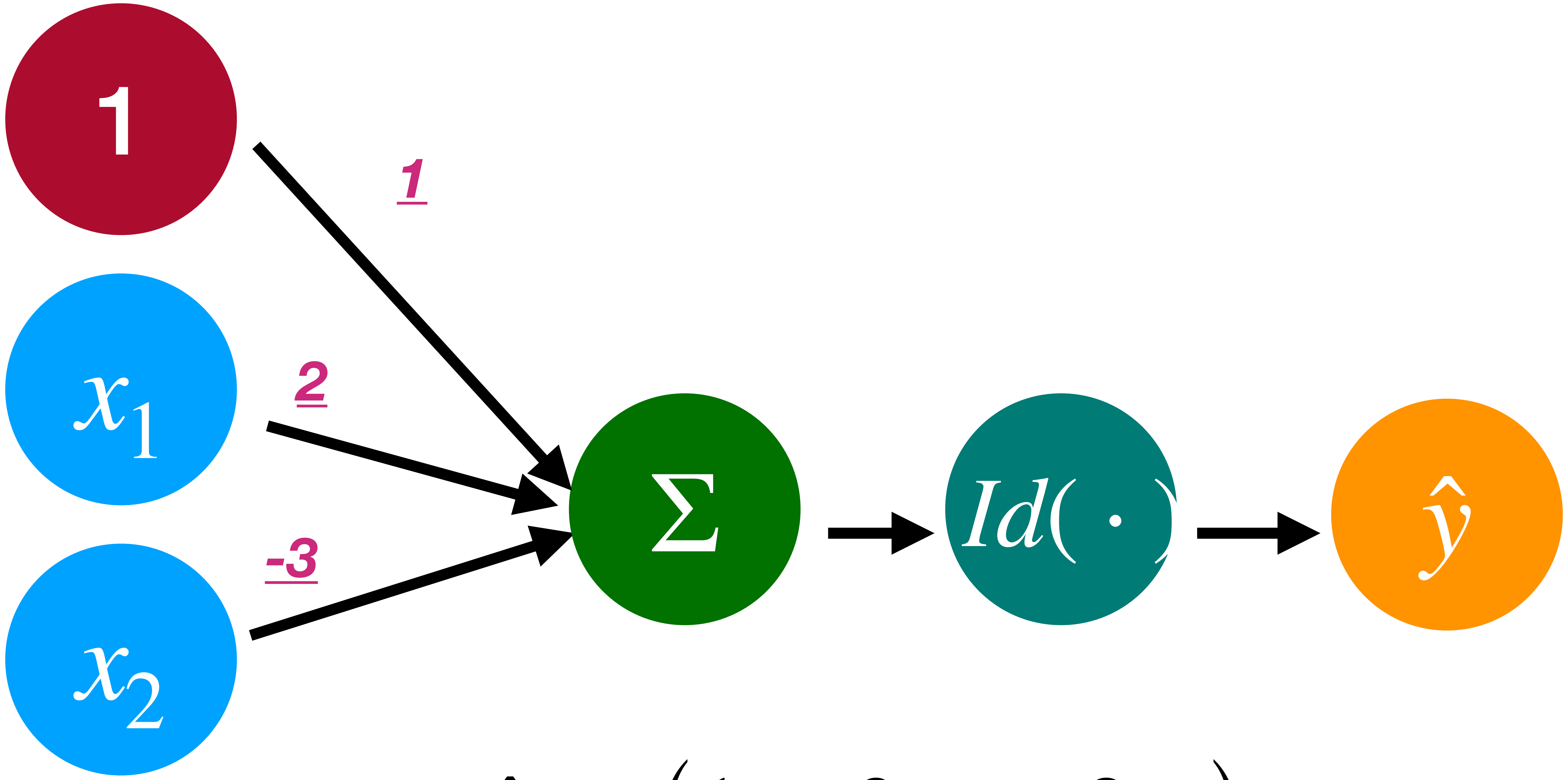
Our Perceptron to distinguish triangles and square



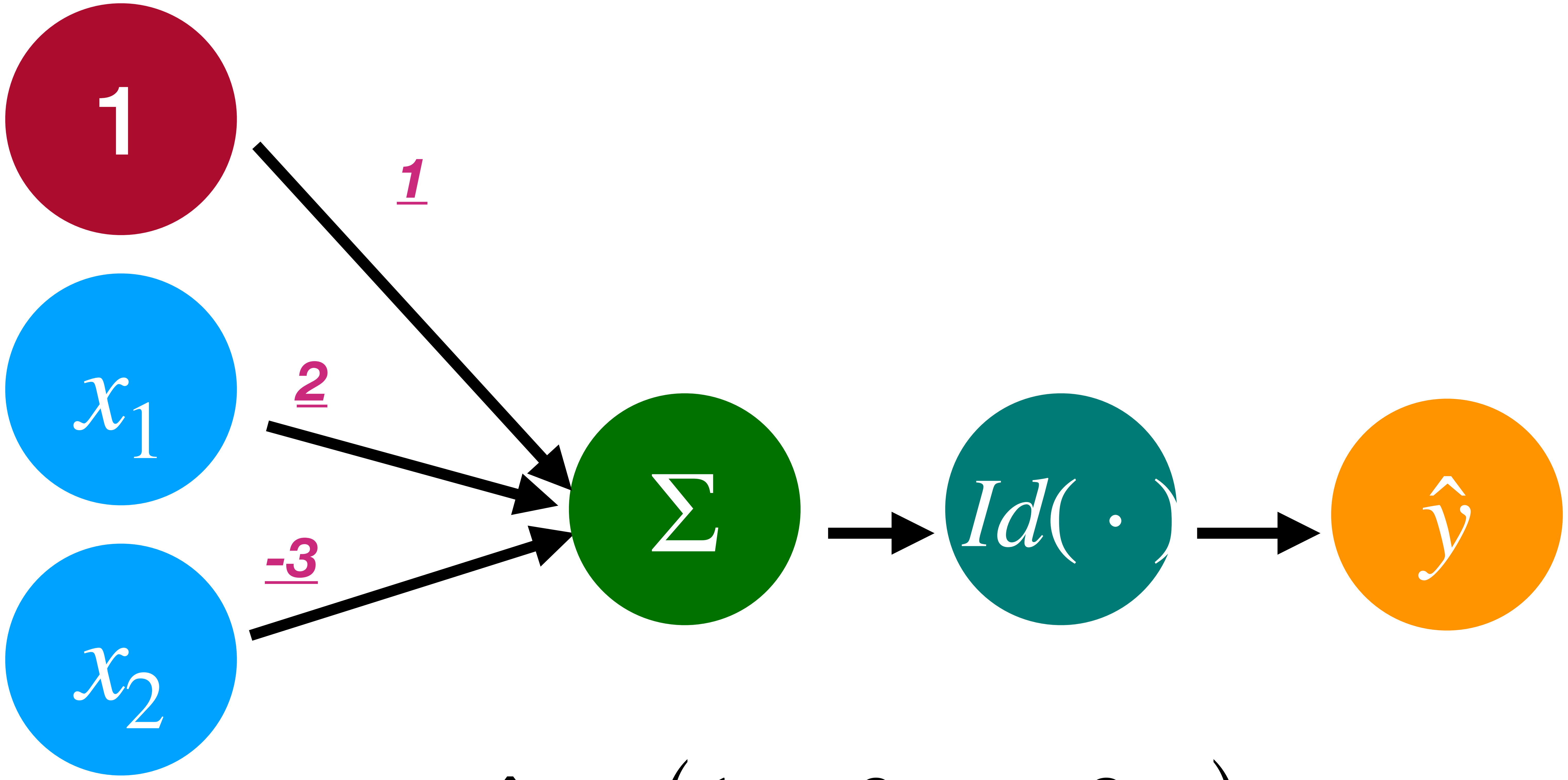
Our Perceptron to distinguish triangles and square



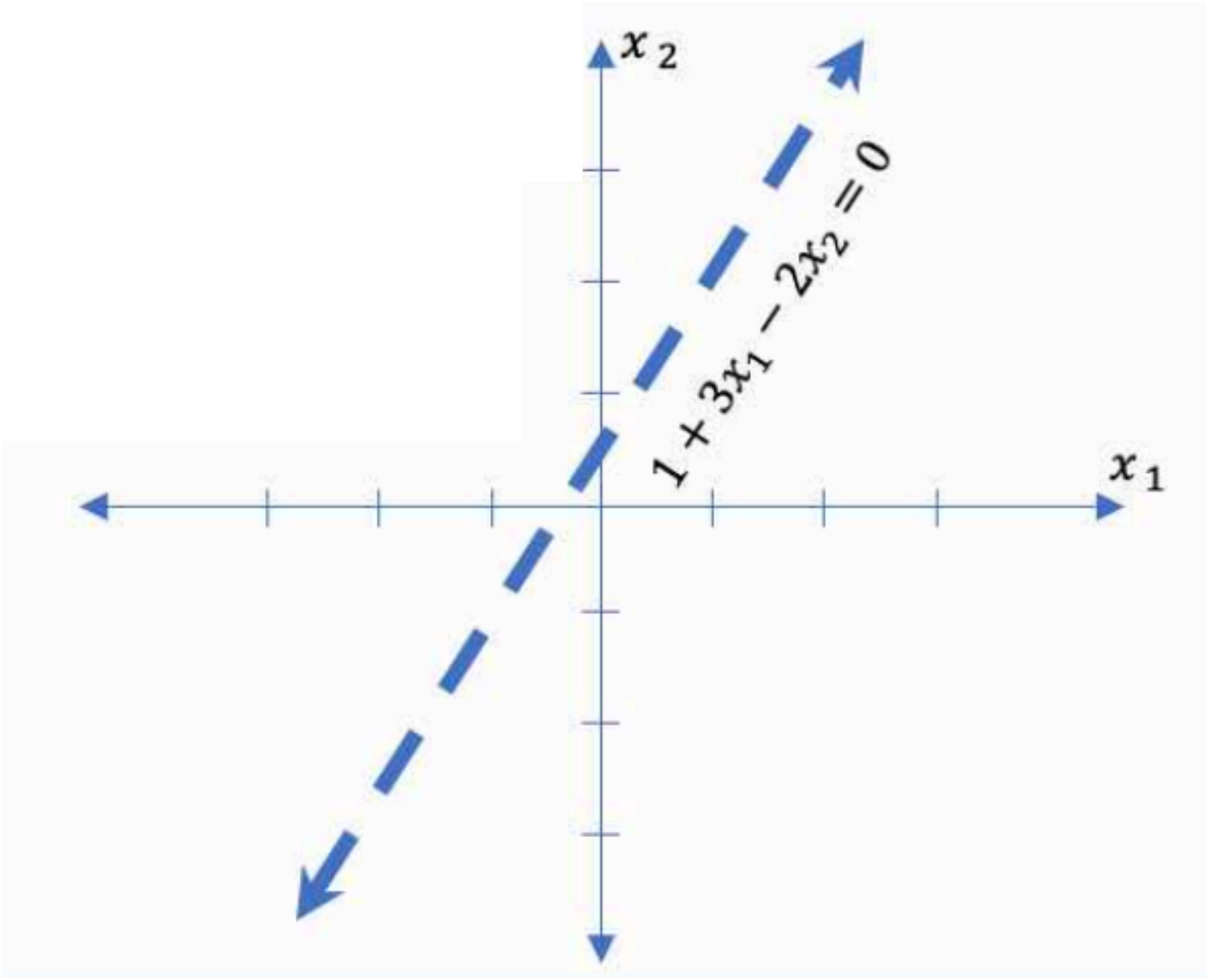
Our Perceptron to distinguish triangles and square



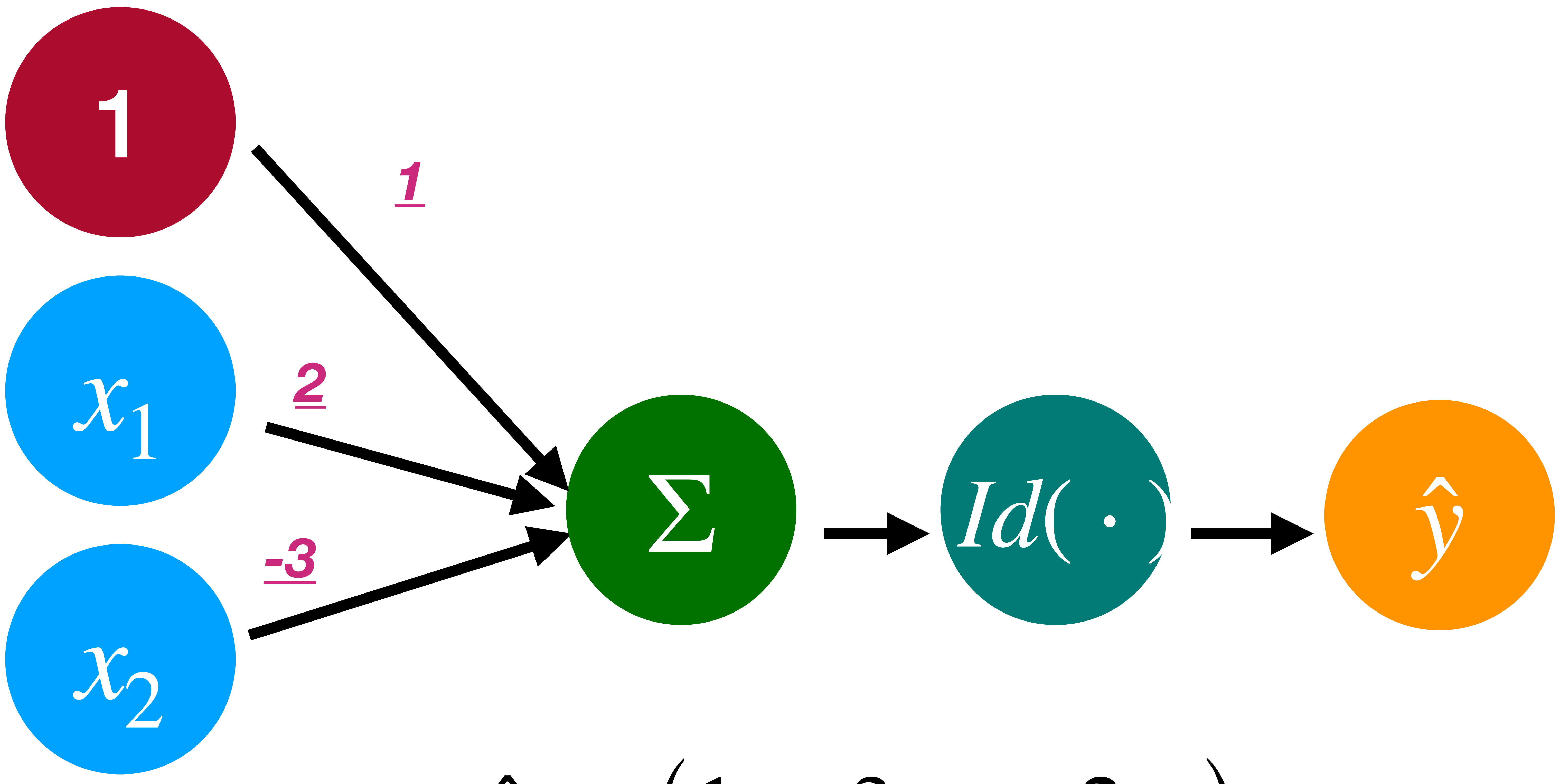
$$\hat{y} = \left(1 + 3x_1 - 2x_2\right)$$



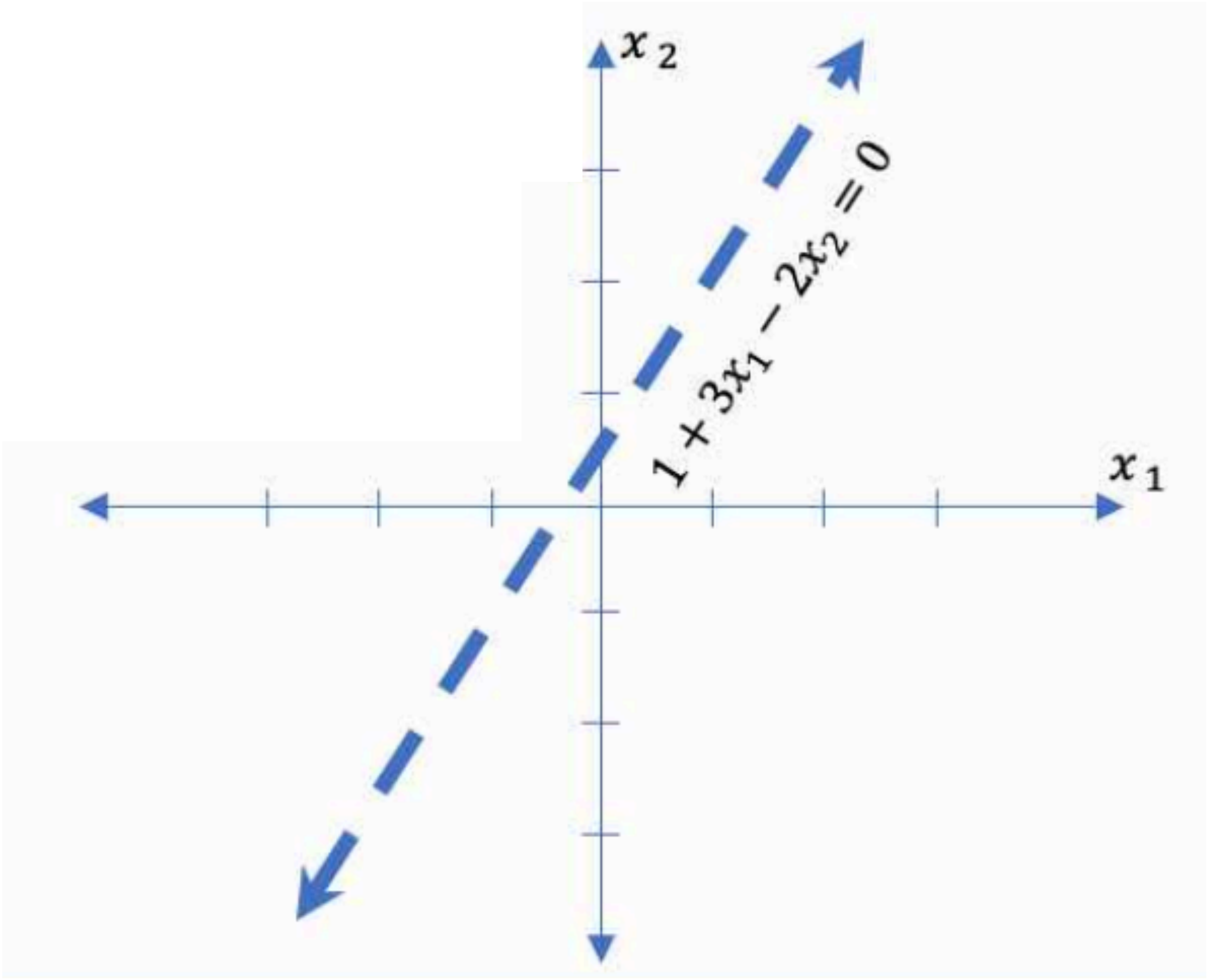
$$\hat{y} = \left(1 + 3x_1 - 2x_2\right)$$



Our Perceptron to distinguish triangles and square



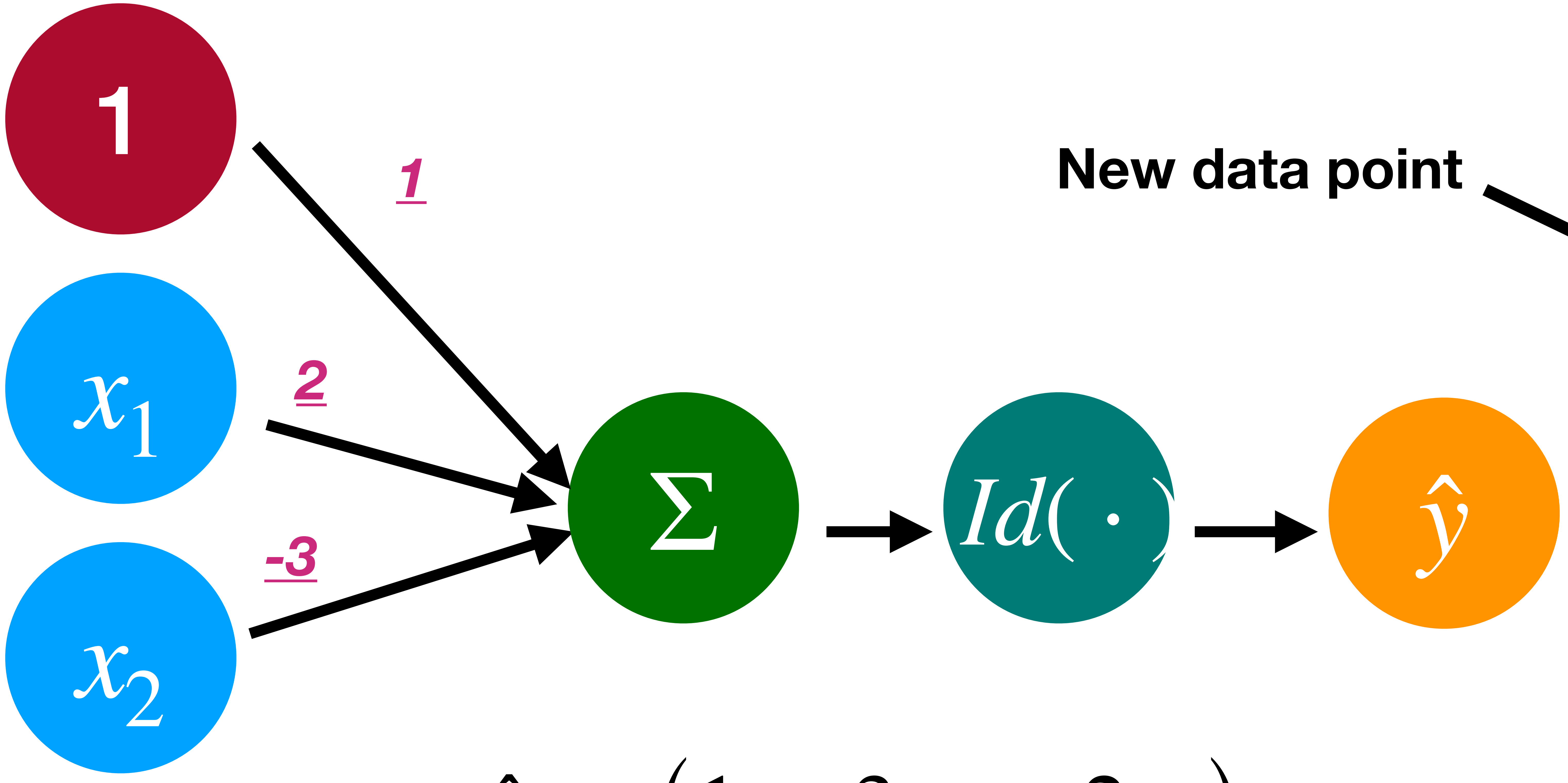
$$\hat{y} = \left(1 + 3x_1 - 2x_2\right)$$



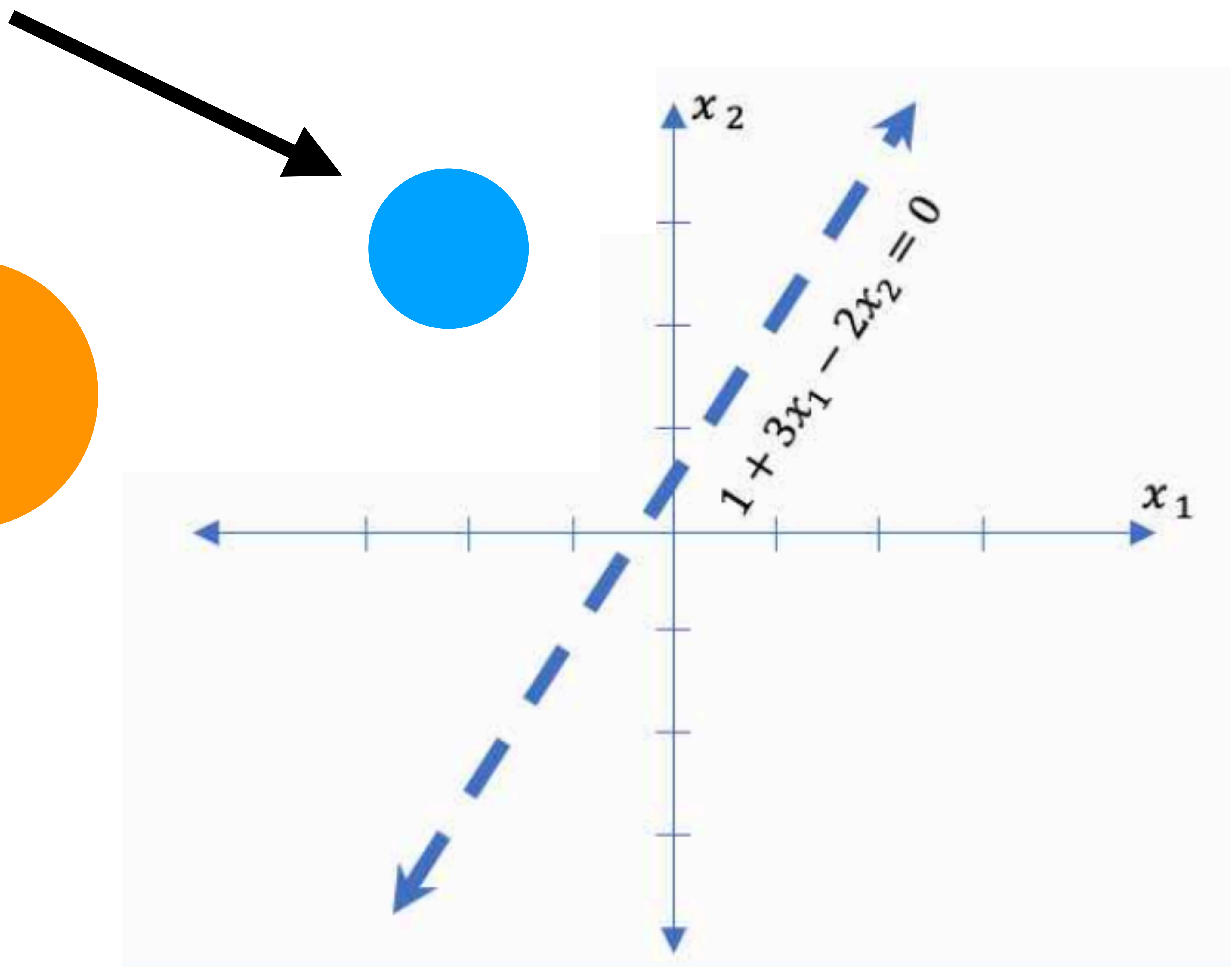
corresponds to the any scalar > 0 if the Input is an triangle

corresponds to the any scalar < 0 if the Input is an square

Our Perceptron to distinguish triangles and square



New data point

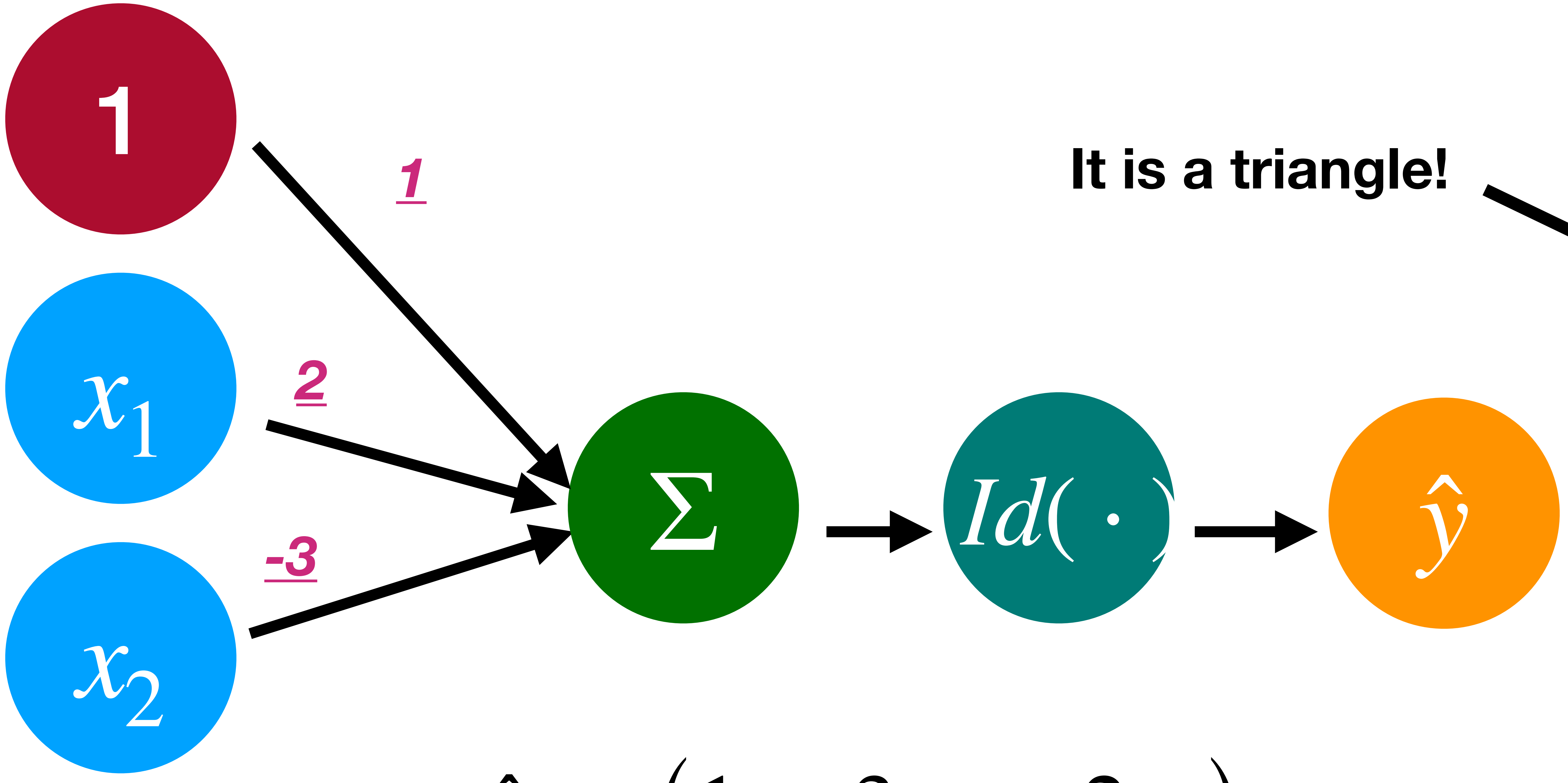


$$\hat{y} = \left(1 + 3x_1 - 2x_2\right)$$

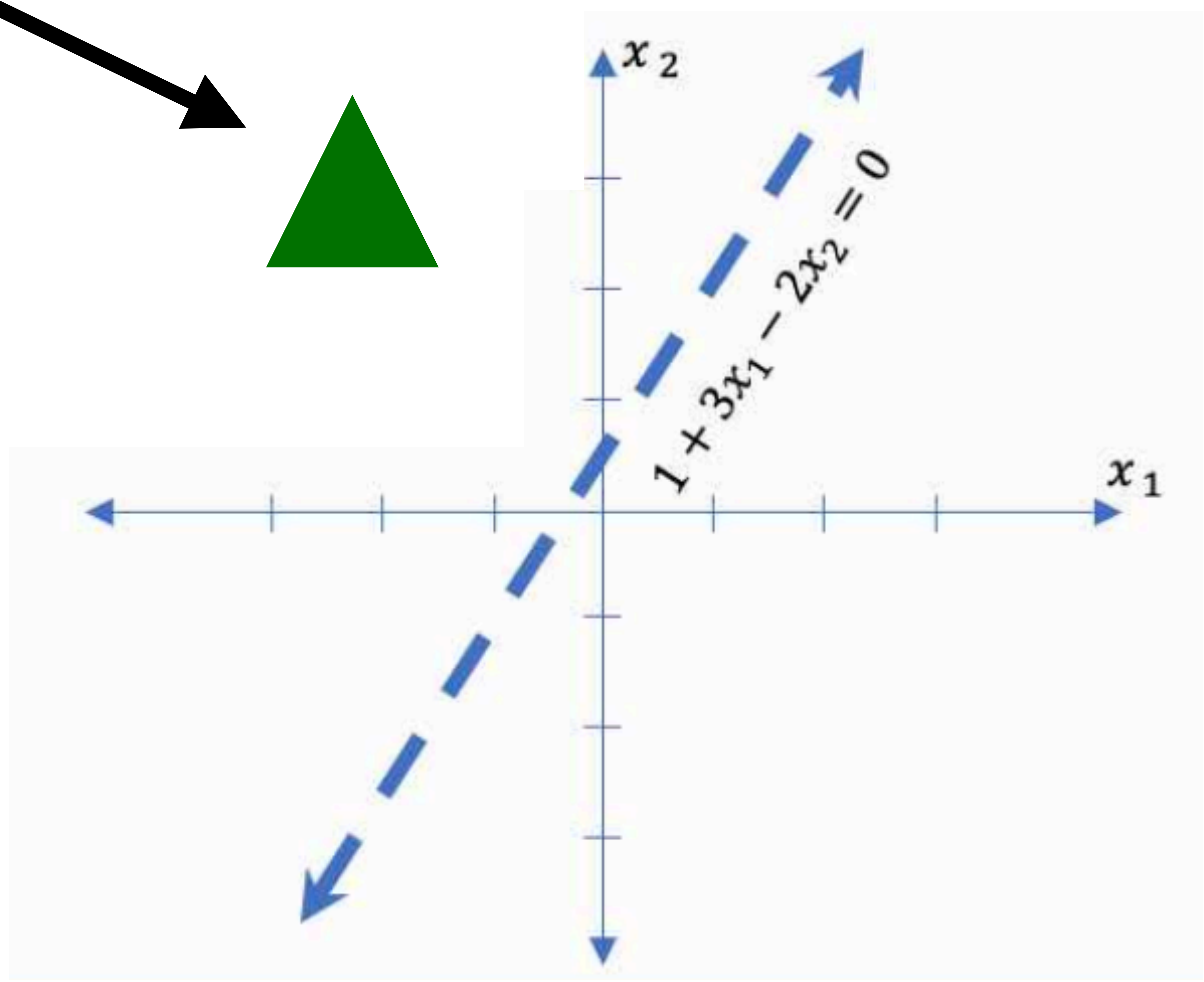
corresponds to the any scalar > 0 if the Input is an triangle

corresponds to the any scalar < 0 if the Input is an square

Our Perceptron to distinguish triangles and square



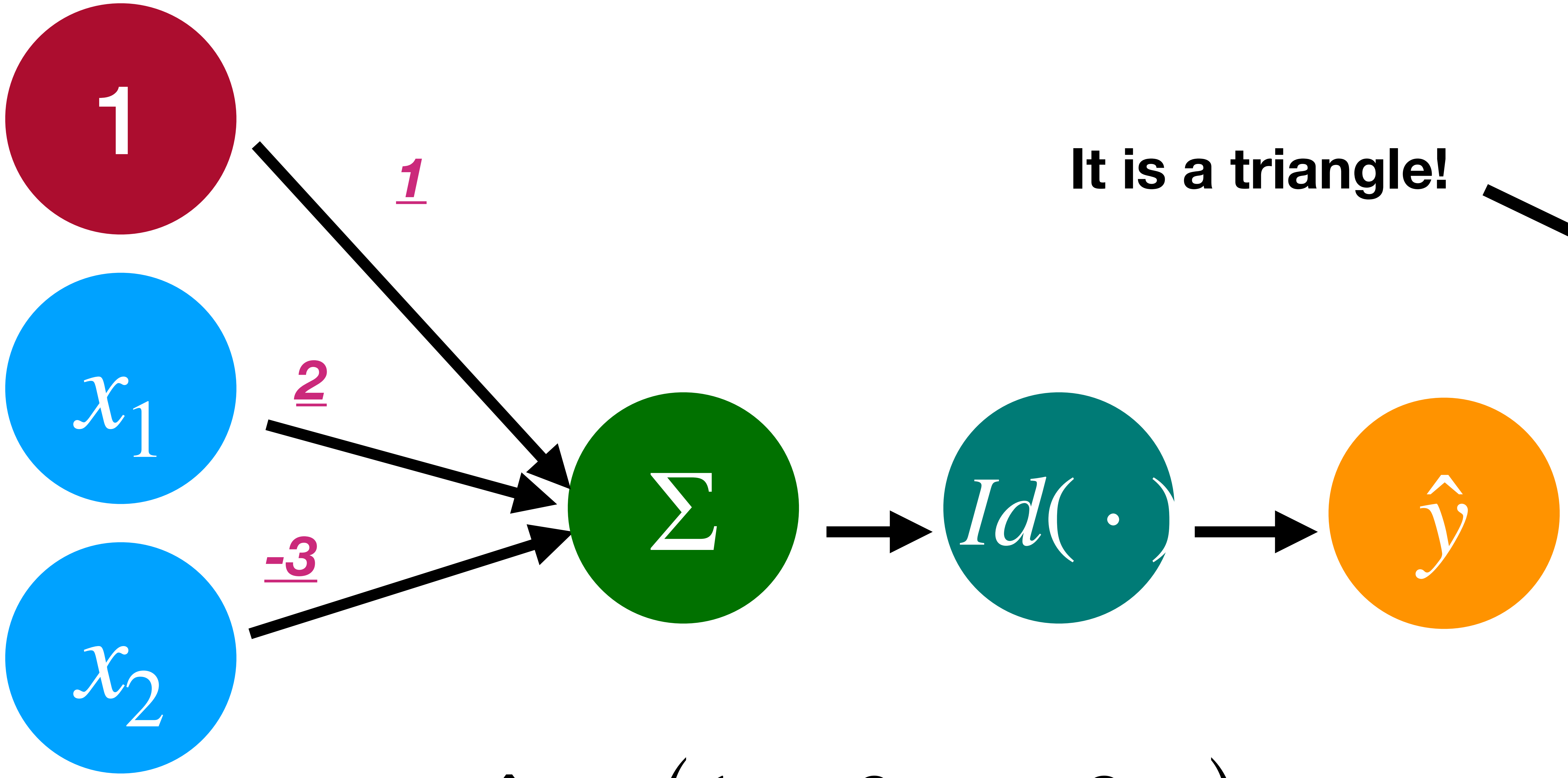
It is a triangle!



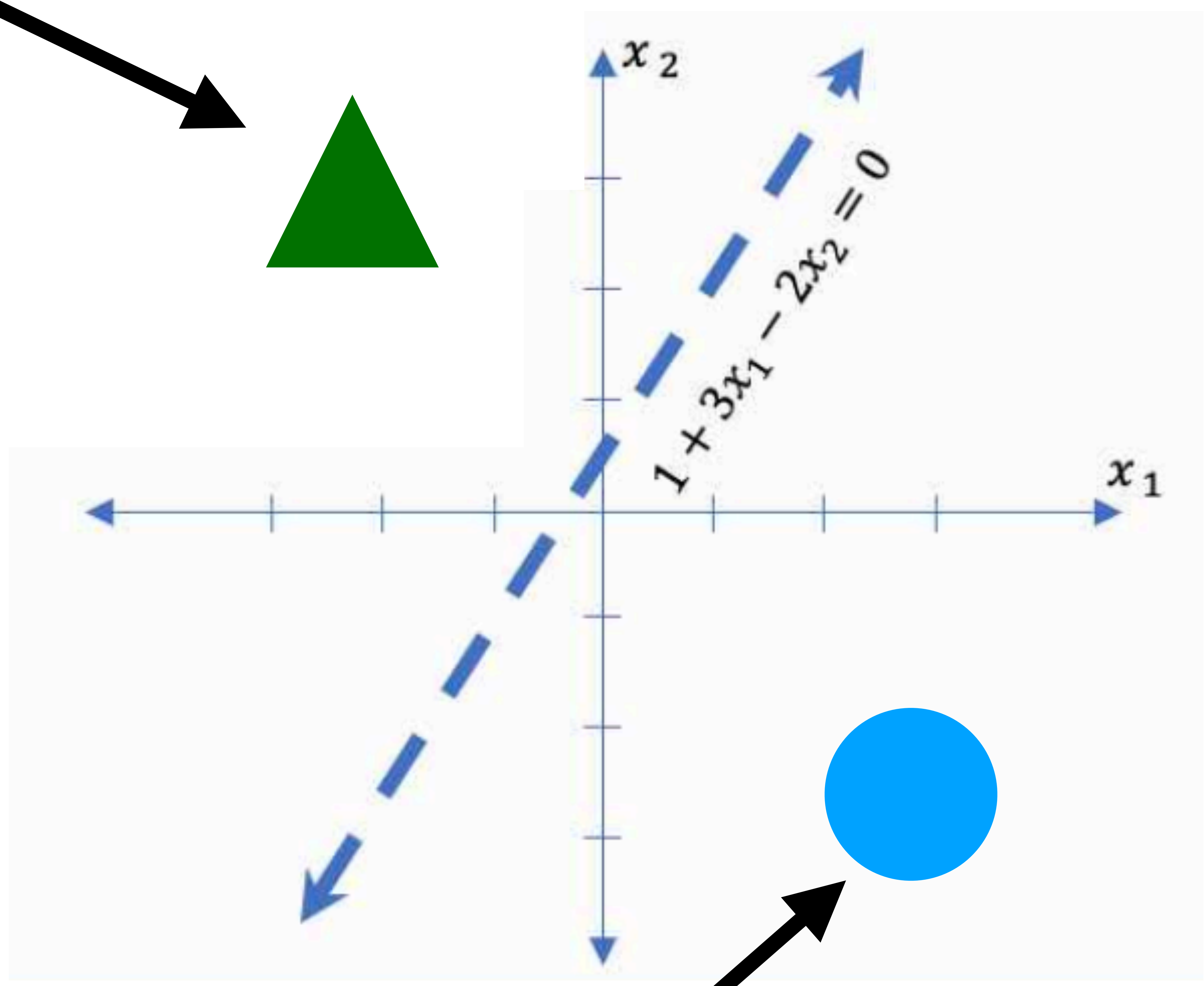
$$\hat{y} = \left(1 + 3x_1 - 2x_2\right)$$

corresponds to the any scalar > 0 if the Input is an triangle

corresponds to the any scalar < 0 if the Input is an square



It is a triangle!



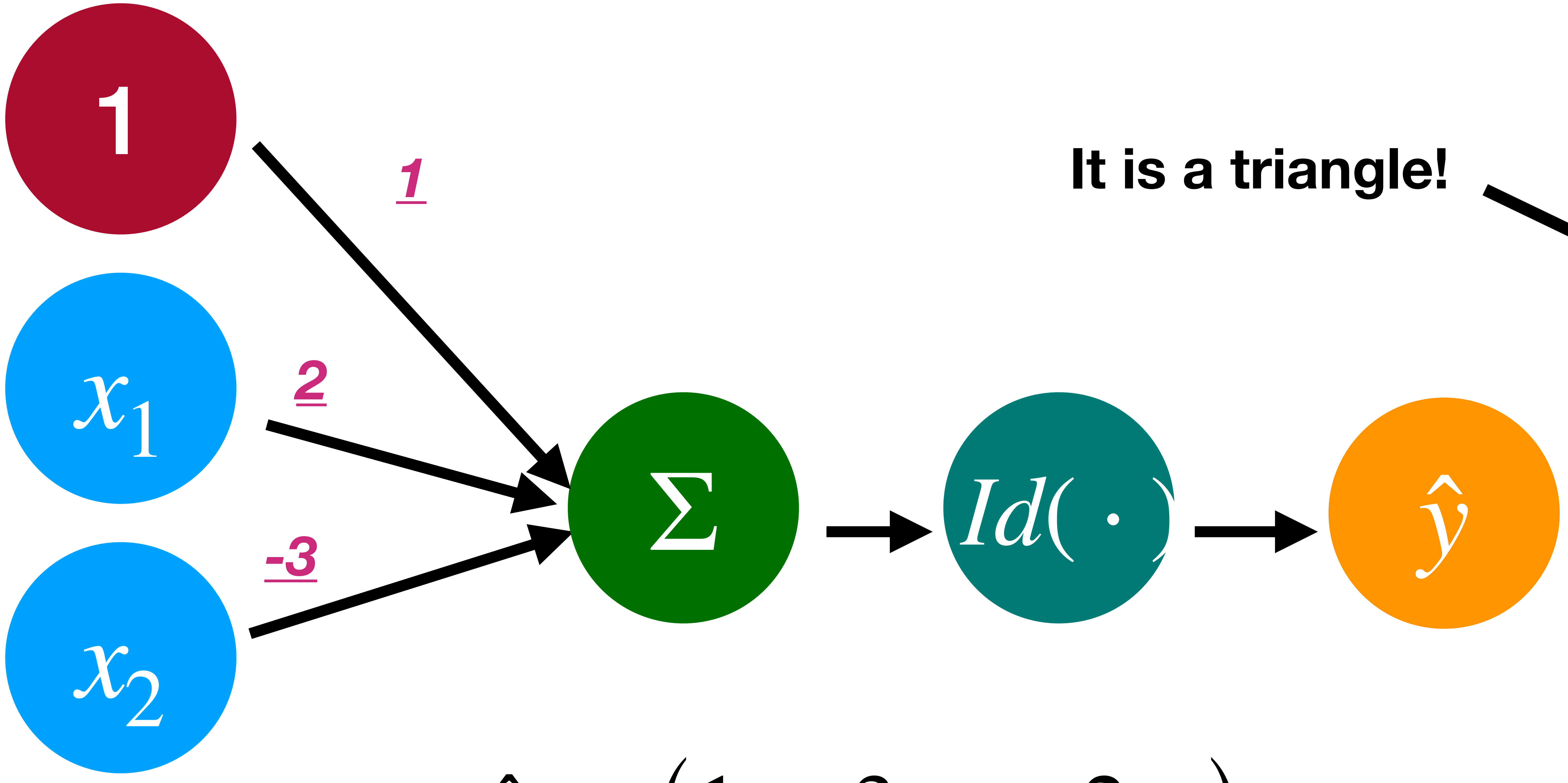
$$\hat{y} = \left(1 + 3x_1 - 2x_2\right)$$

New data point

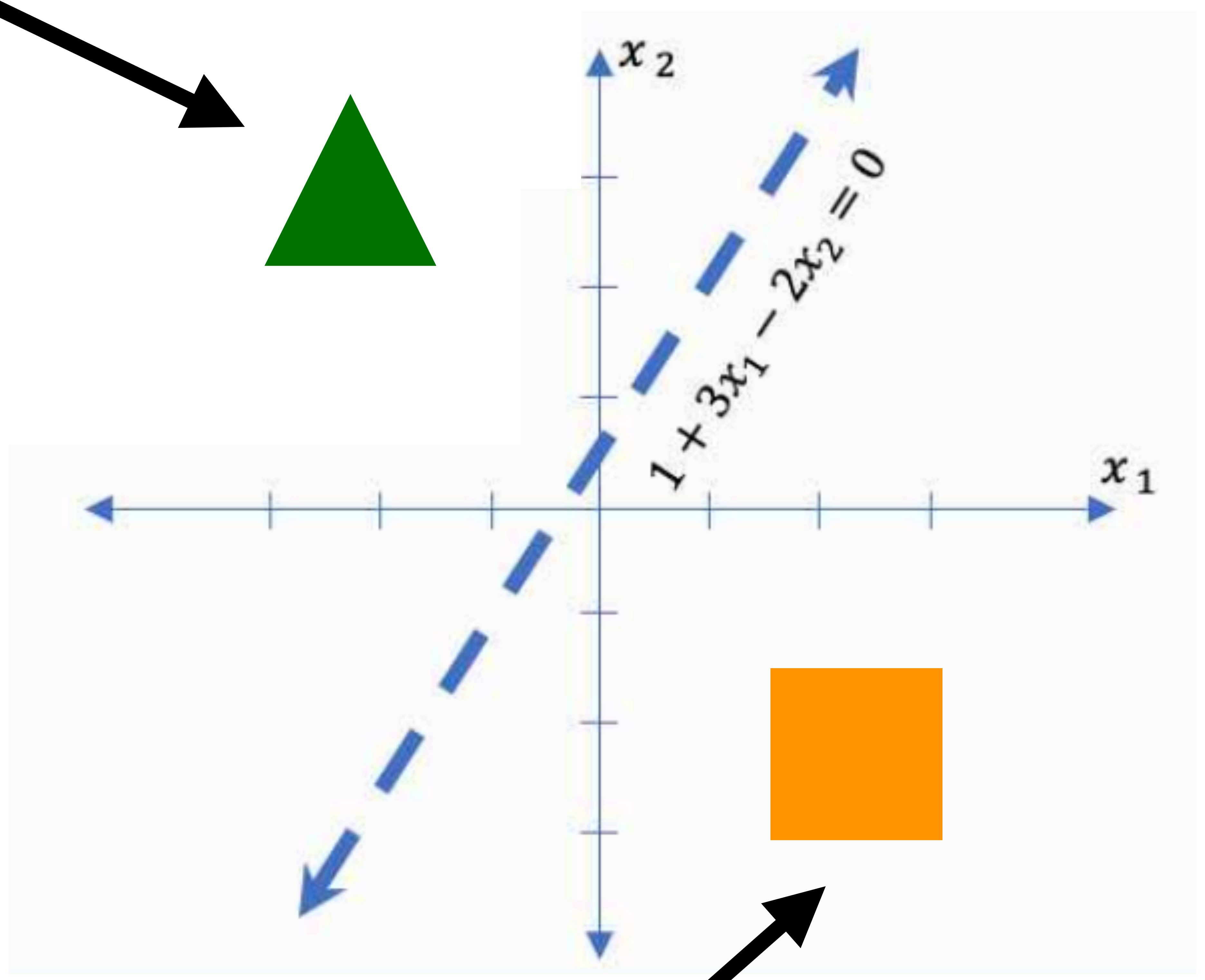
corresponds to the any scalar > 0 if the Input is an triangle

corresponds to the any scalar < 0 if the Input is an square

Our Perceptron to distinguish triangles and square



It is a triangle!



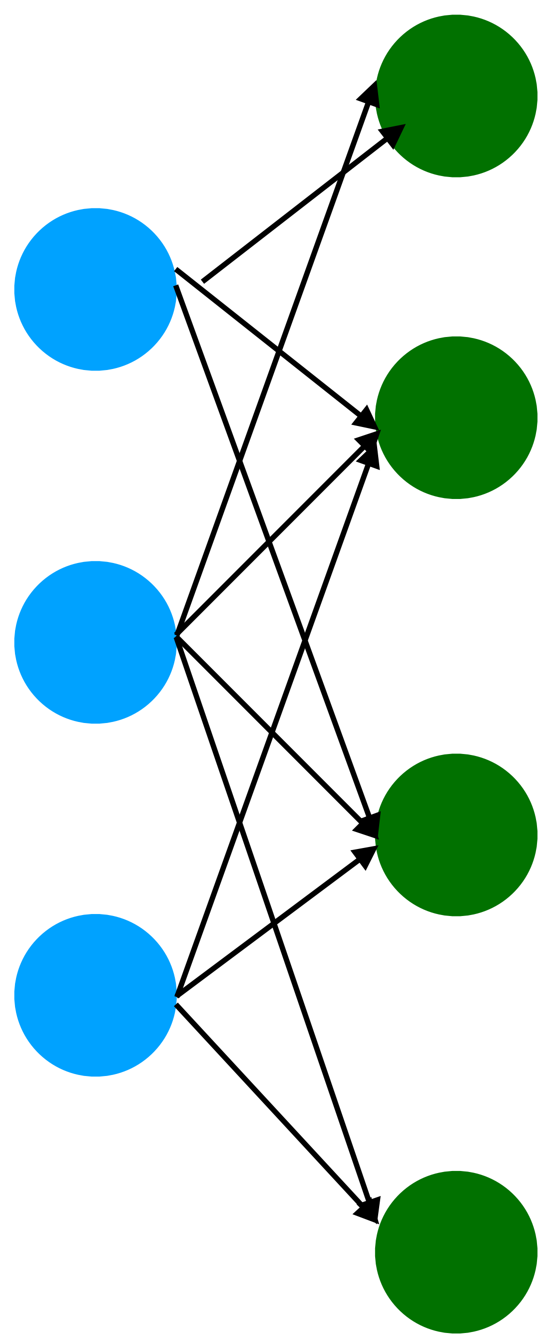
It is a square

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

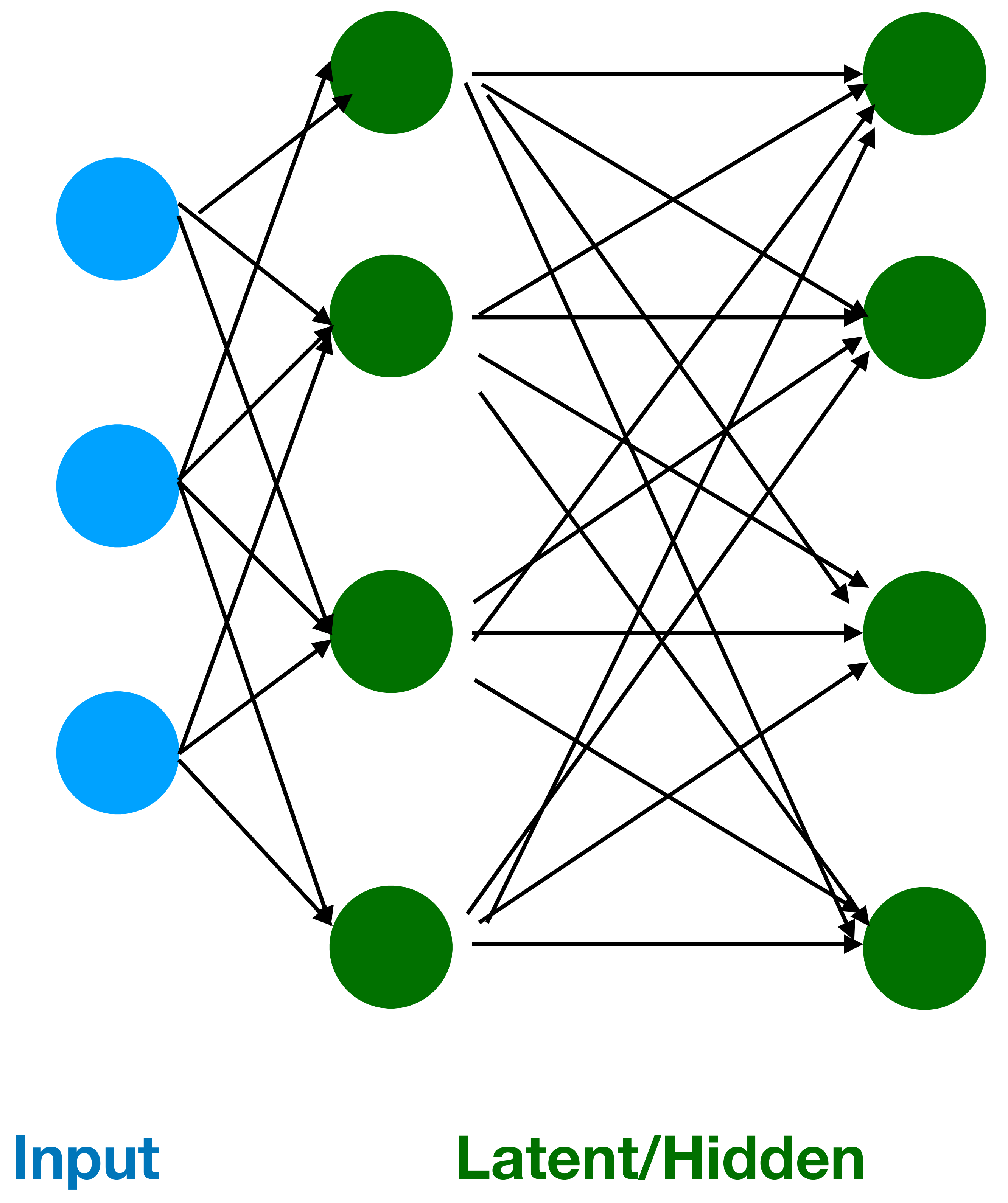
corresponds to the any scalar > 0 if the Input is an triangle

corresponds to the any scalar < 0 if the Input is an square

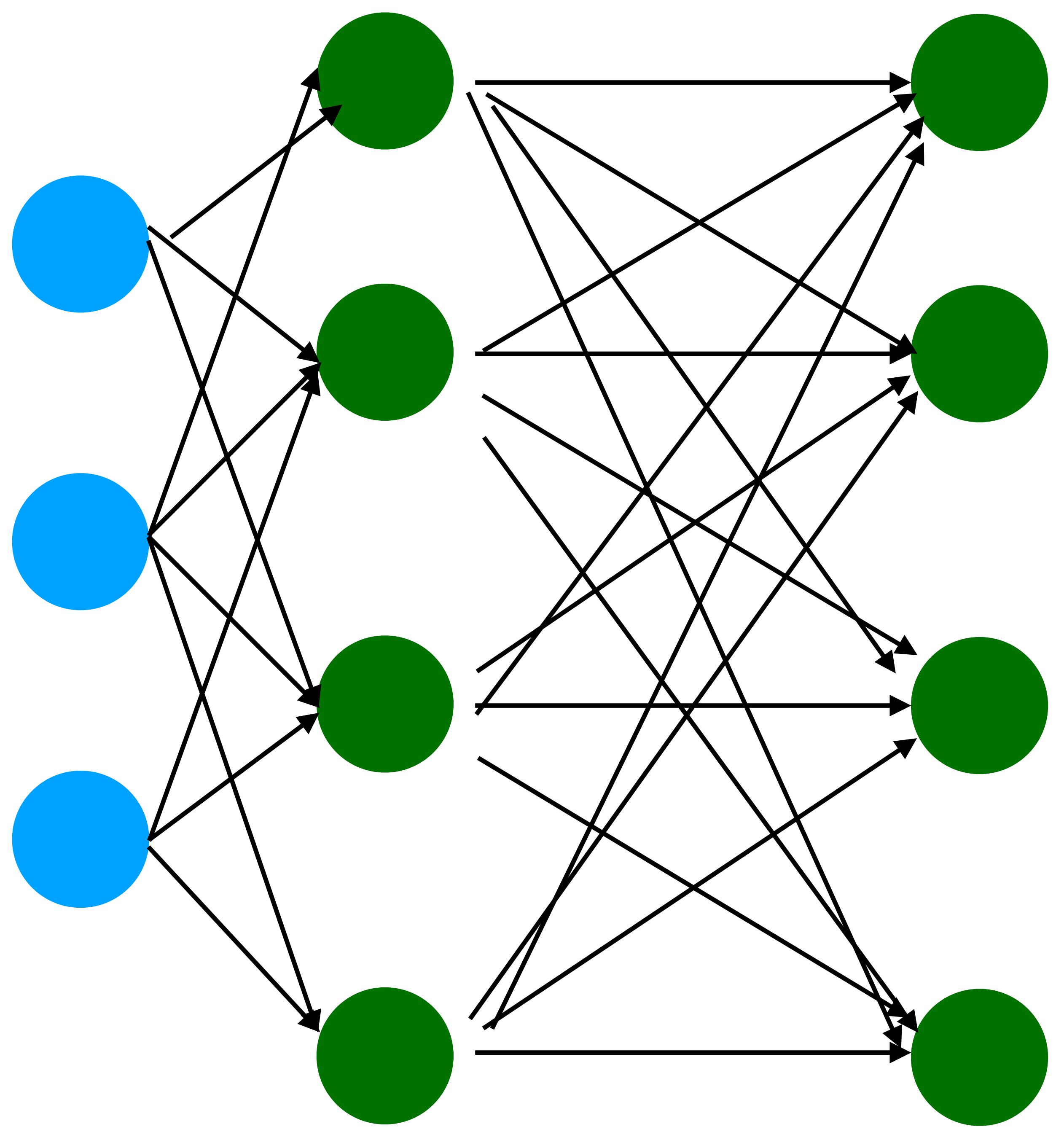
Towards more complex architectures


Input
Latent/Hidden

Towards more complex architectures



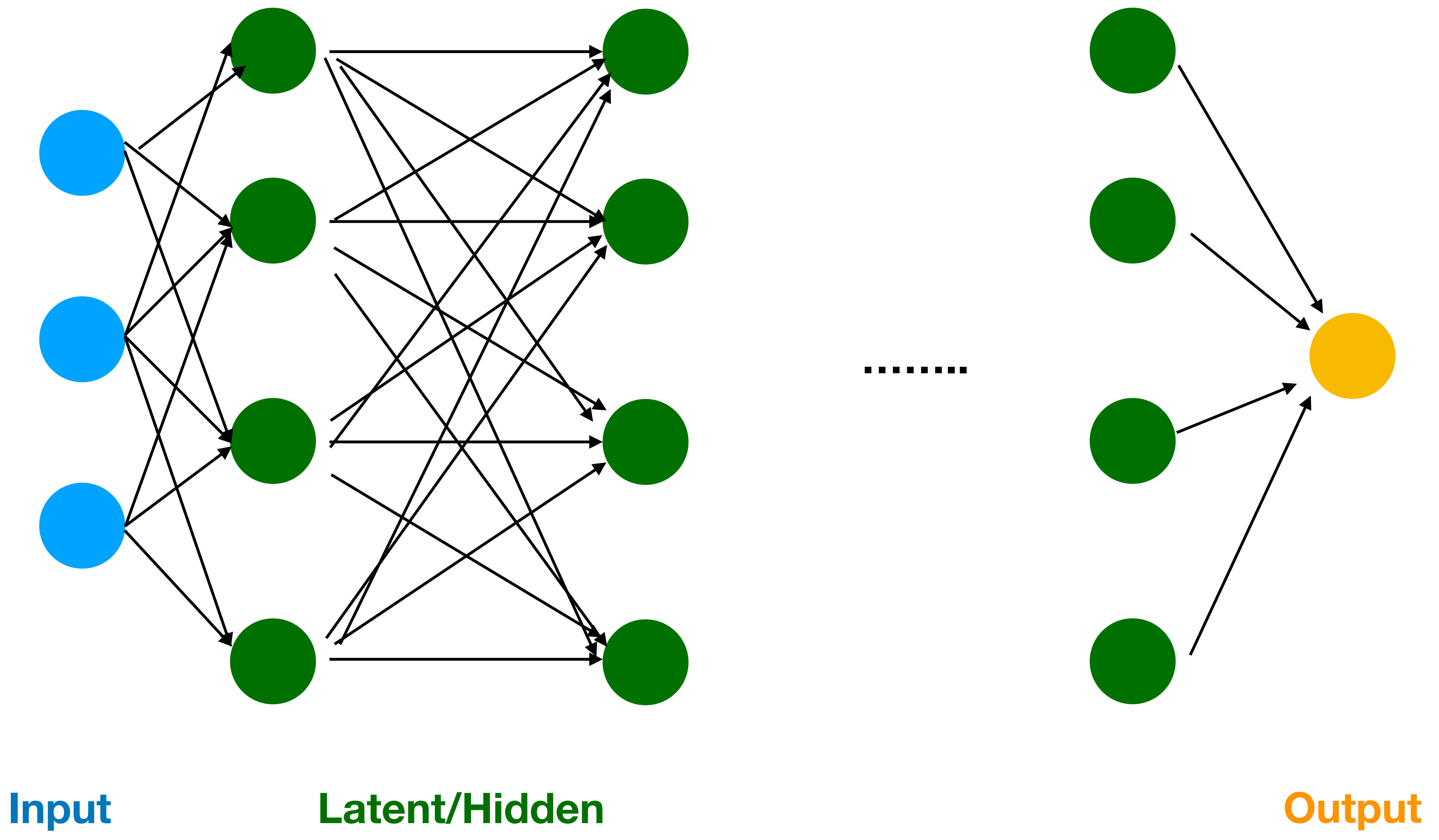
Towards more complex architectures



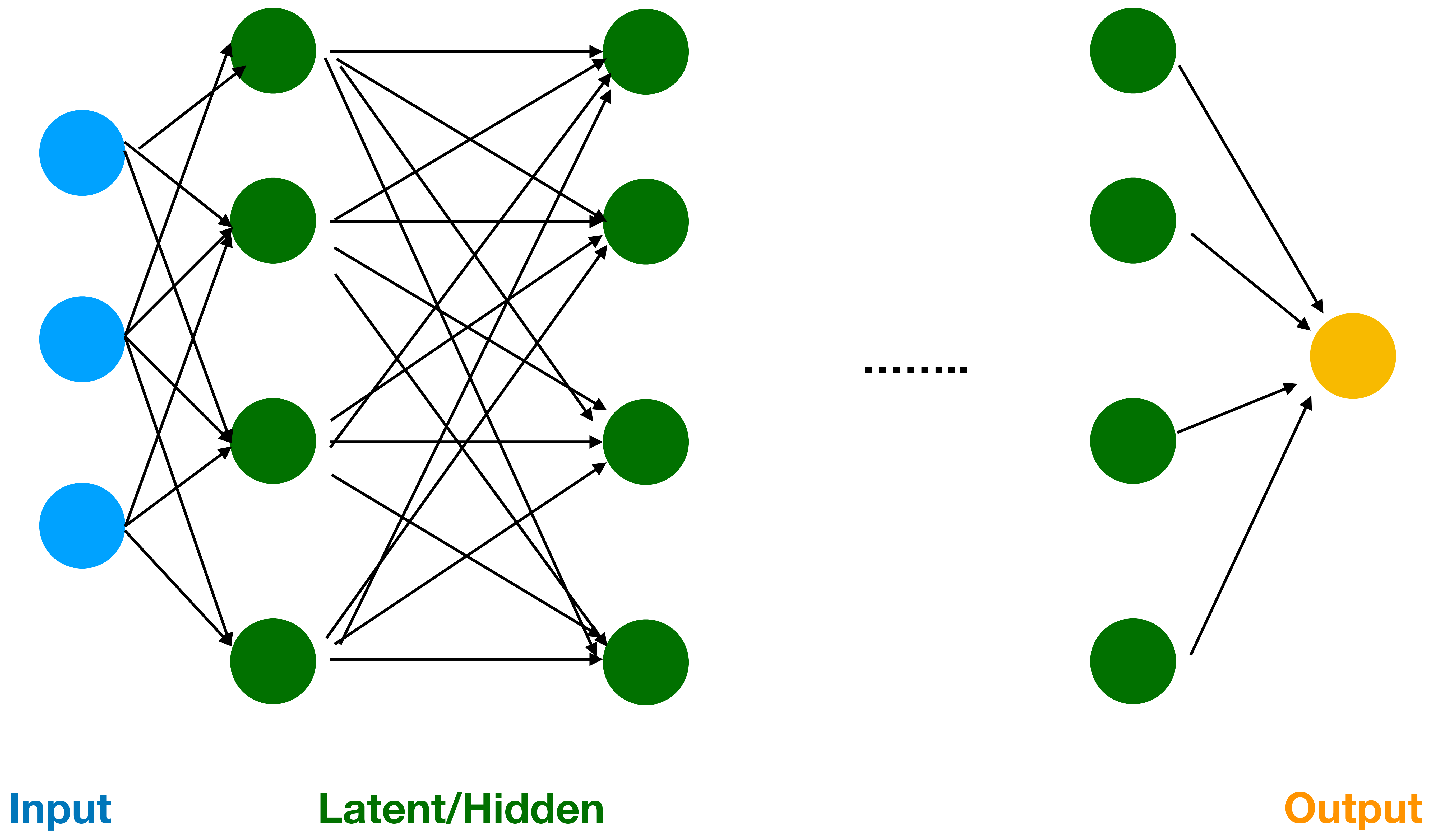
....

Input Latent/Hidden

Towards more complex architectures

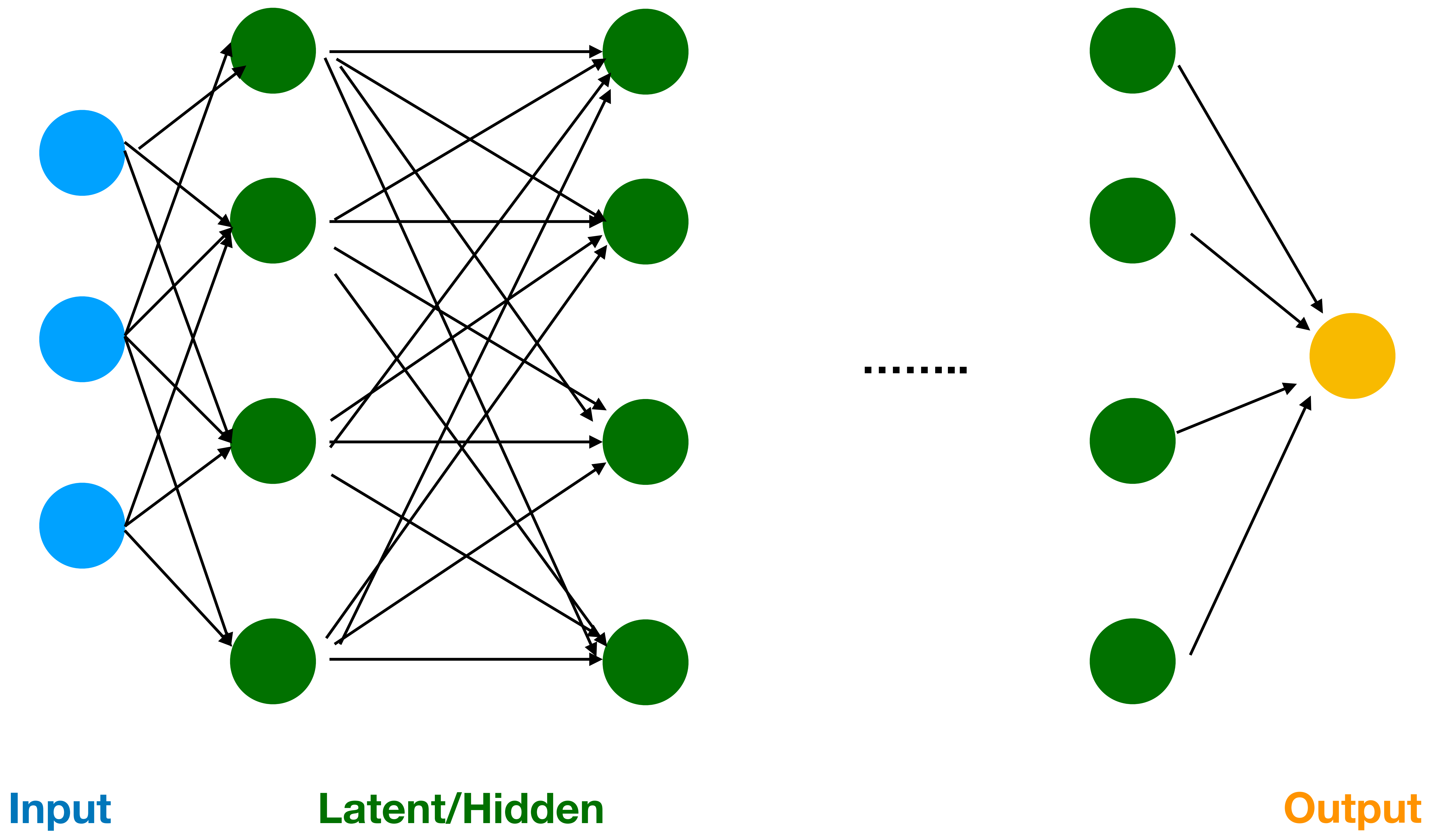


Towards more complex architectures



$$\hat{y} = \sigma\left(\cdots\sigma\left(W_1\sigma\left(\left(X^T W_0\right)^T\right)\right)\right)$$

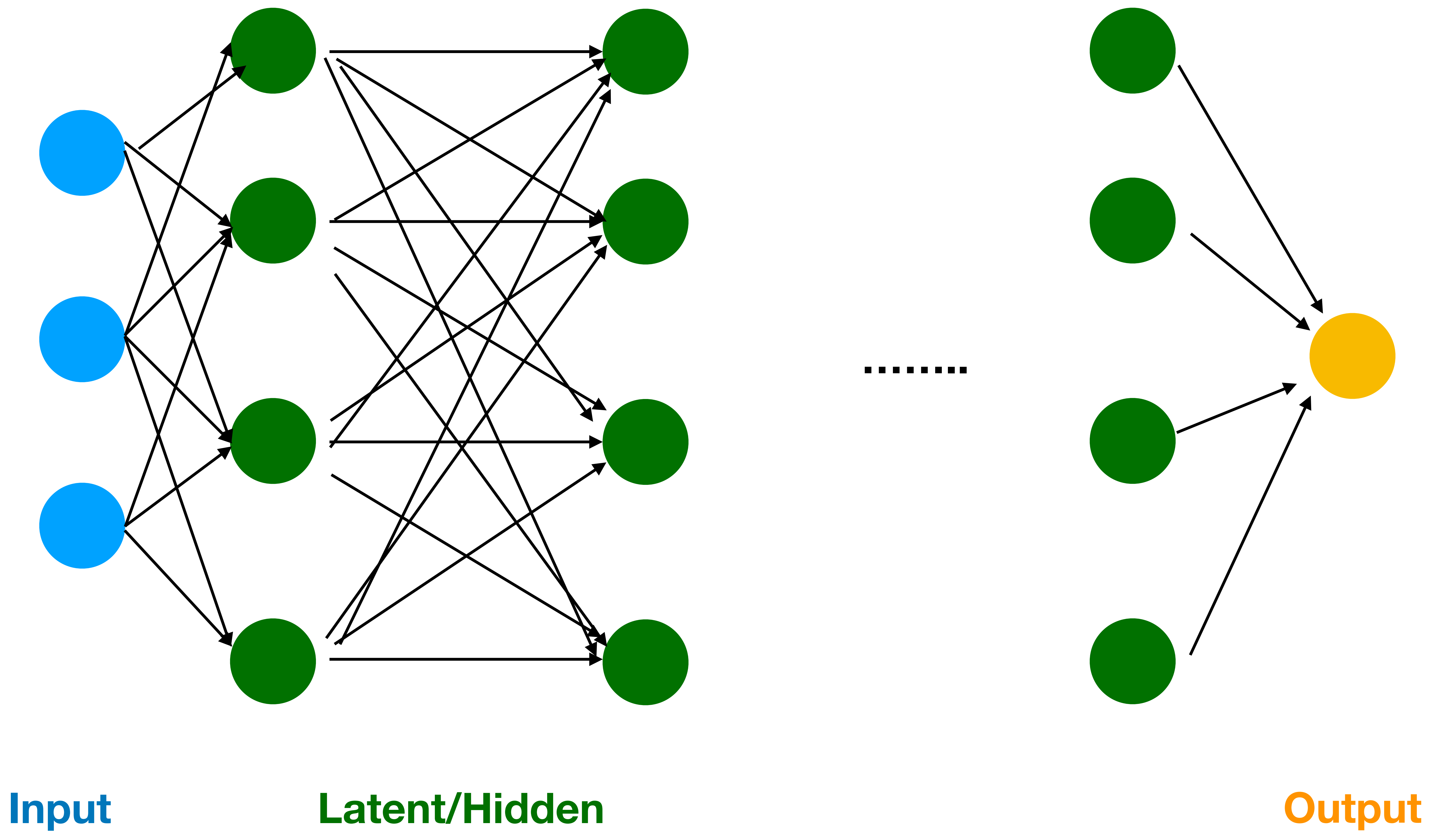
Towards more complex architectures



$$\hat{y} = \sigma\left(\cdots\sigma\left(W_1\sigma\left(X^T W_0\right)^T\right)\right)$$

Adding layers is equivalent to adding new matrix and activation functions !

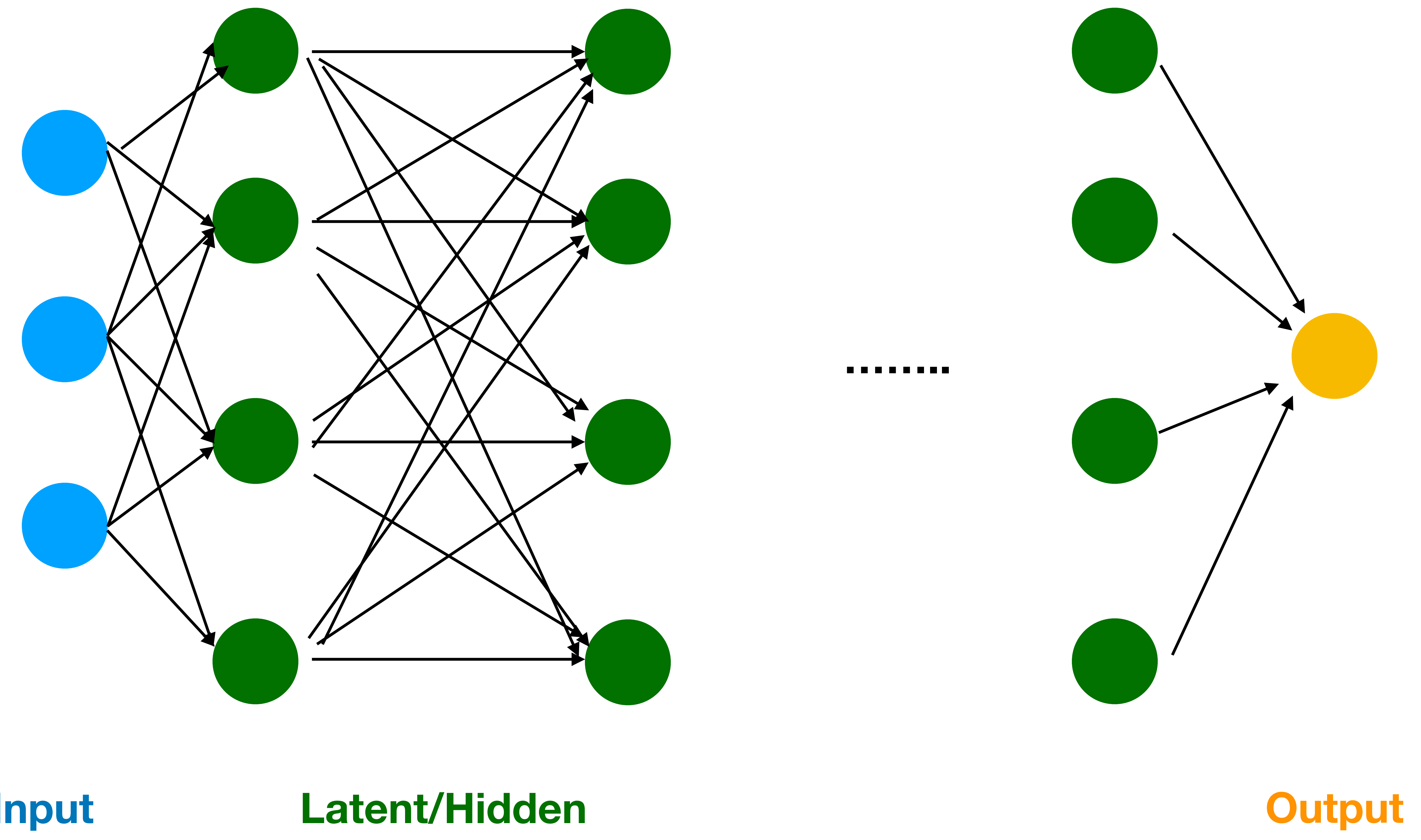
Towards more complex architectures



$$\hat{y} = \sigma\left(\cdots\sigma\left(W_1\sigma\left(\left(X^T W_0\right)^T\right)\right)\right)$$

Adding layers is equivalent to adding new matrix and activation functions !

How do I choose the network architecture?



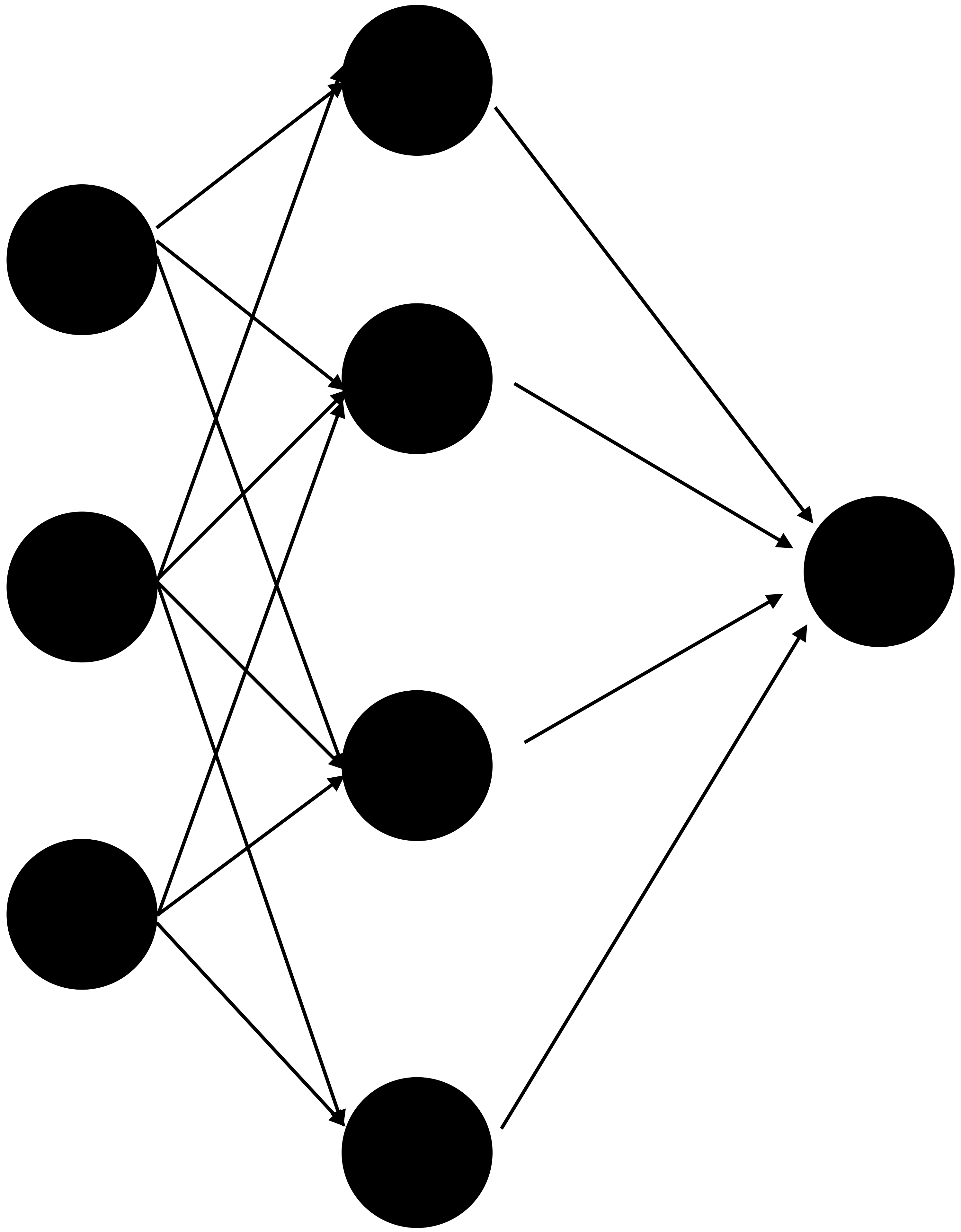
$$\hat{y} = \sigma\left(\cdots\sigma\left(W_1\sigma\left(\left(X^T W_0\right)^T\right)\right)\right)$$

Adding layers is equivalent to adding new matrix and activation functions !

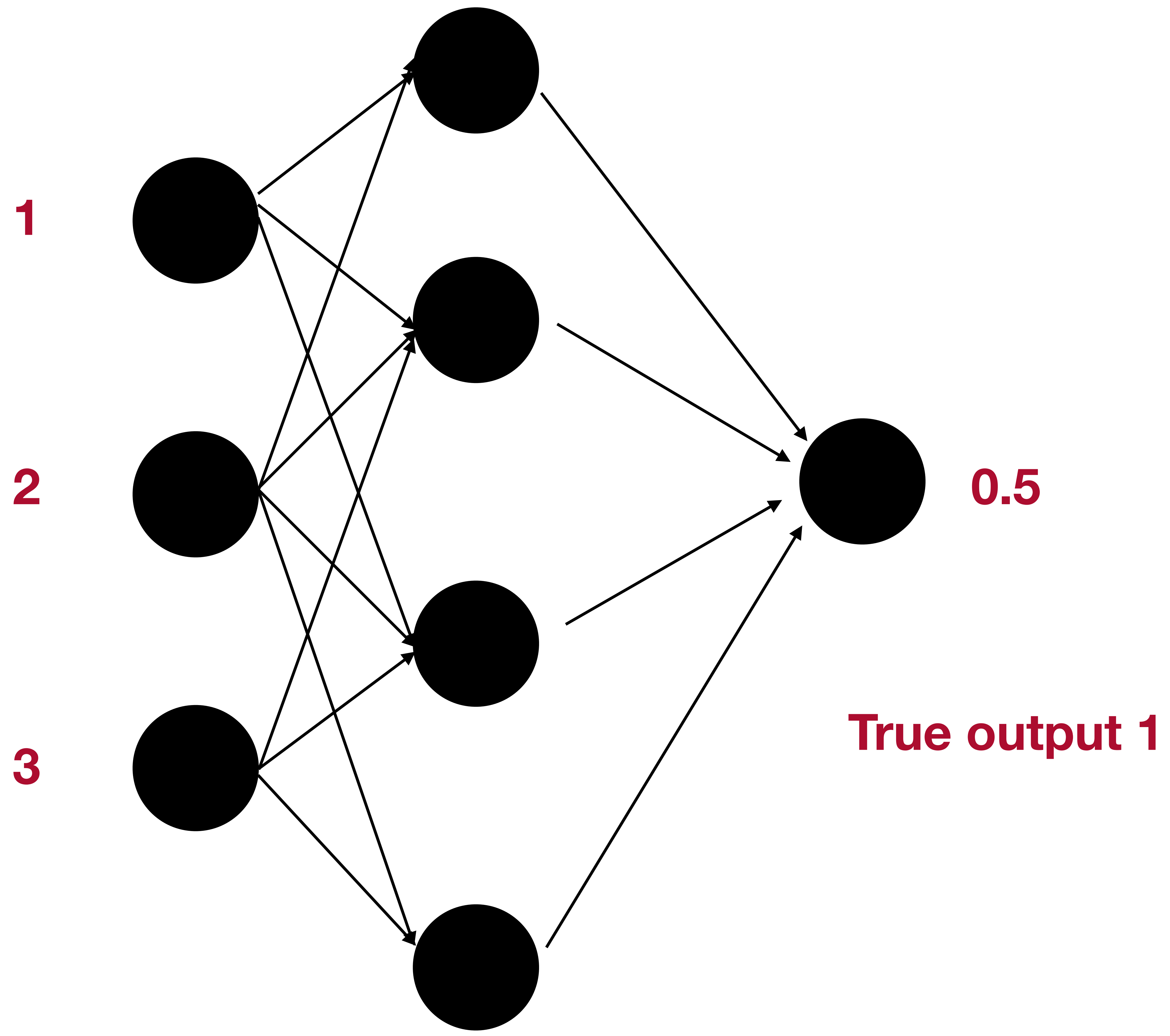
How do I choose the network architecture? → **Research papers!**

Another example with a one layer NN

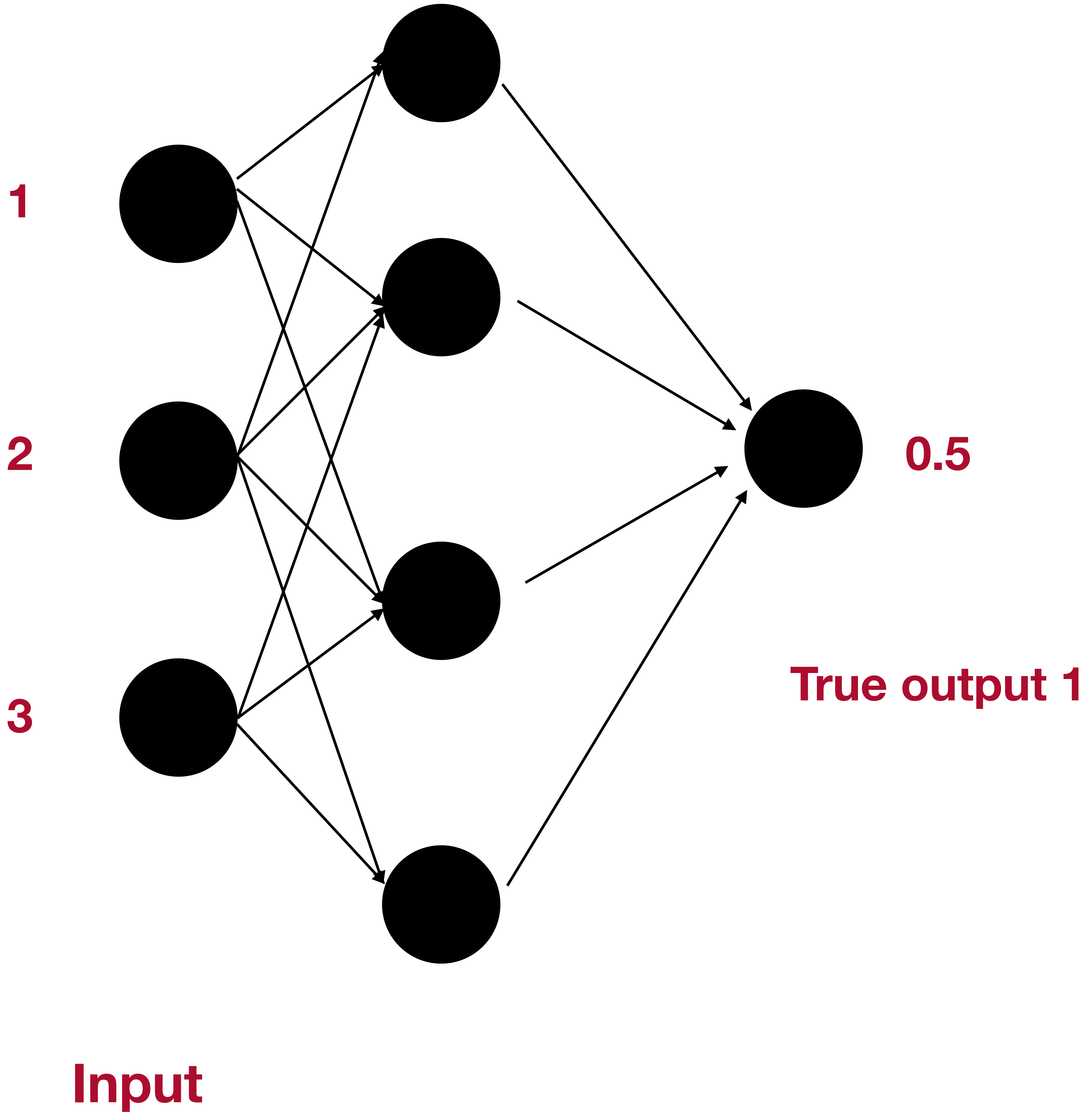
Another example with a one layer NN



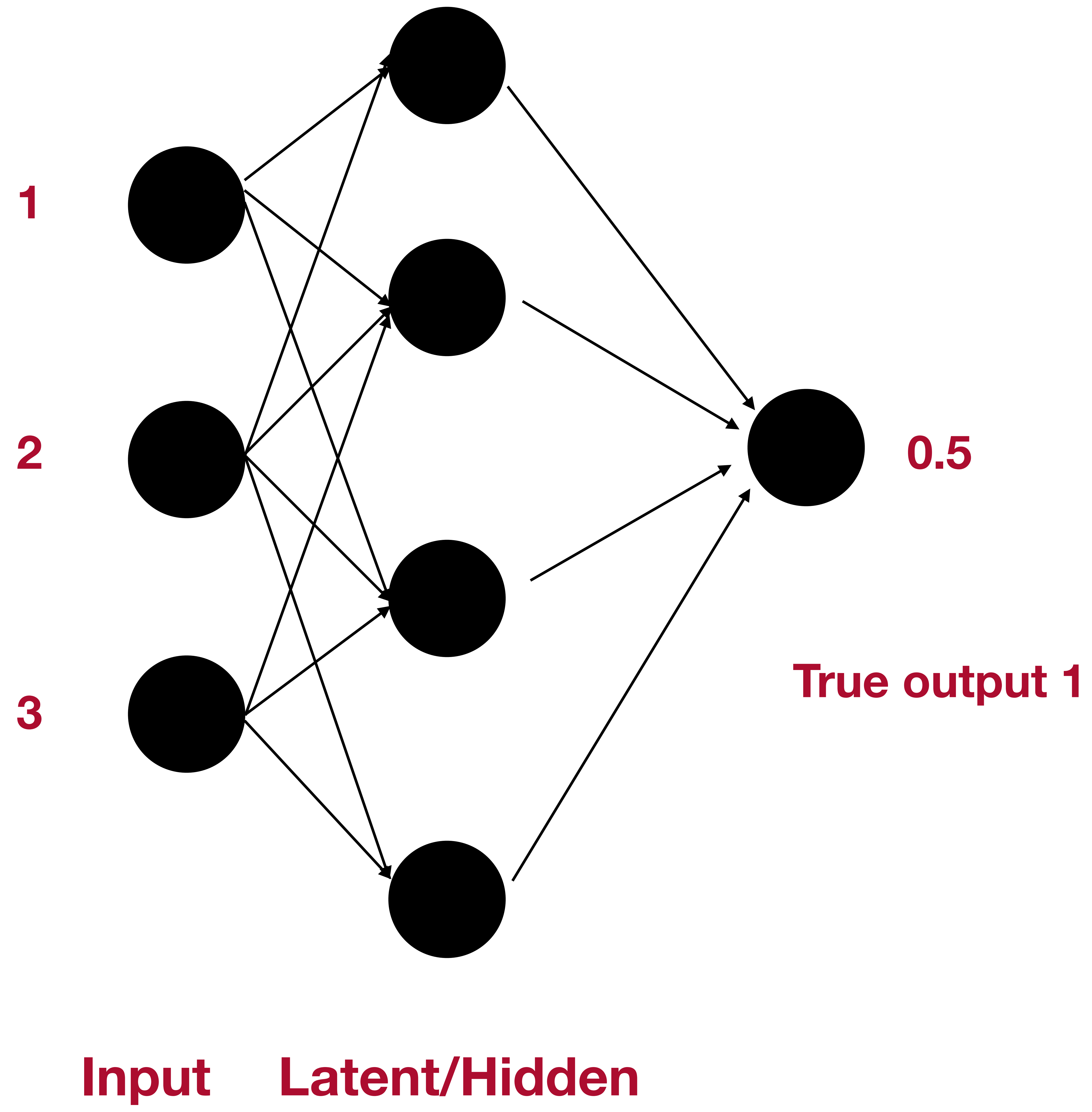
Another example with a one layer NN



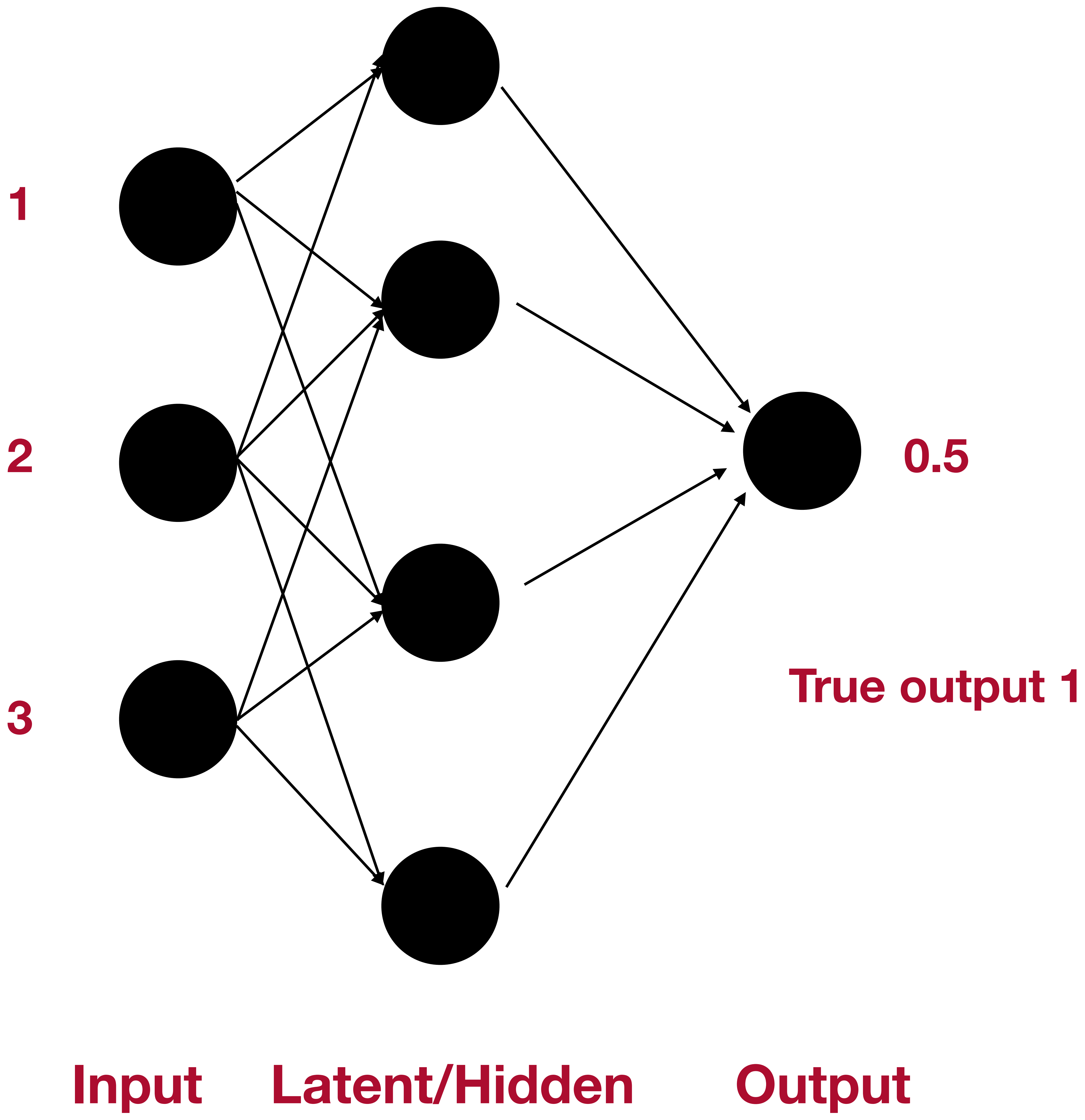
Another example with a one layer NN



Another example with a one layer NN

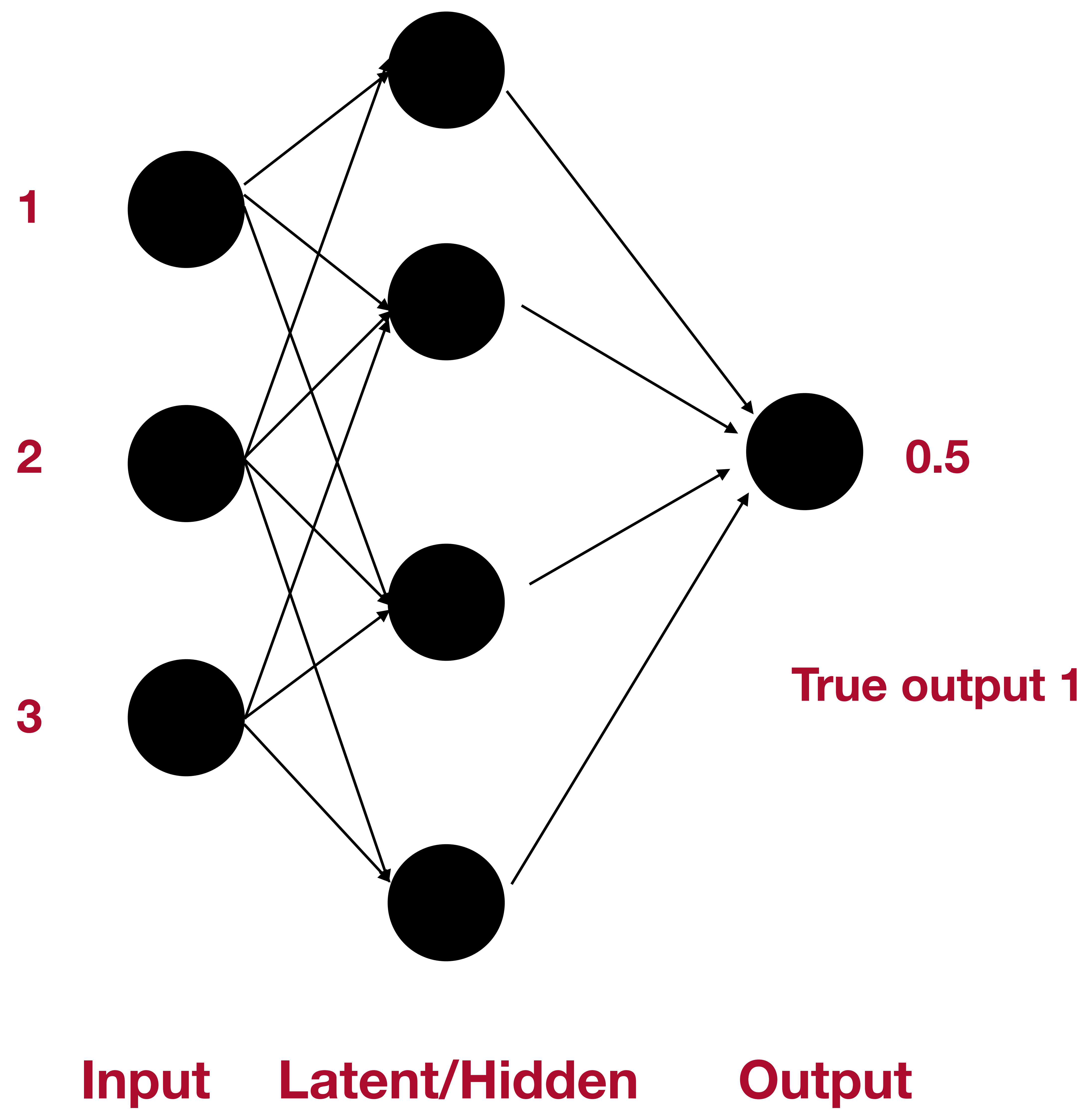


Another example with a one layer NN



Suppose that I give as input $\mathbf{X} = [1,2,3]$
and obtain as output $\hat{y} = 0.5$

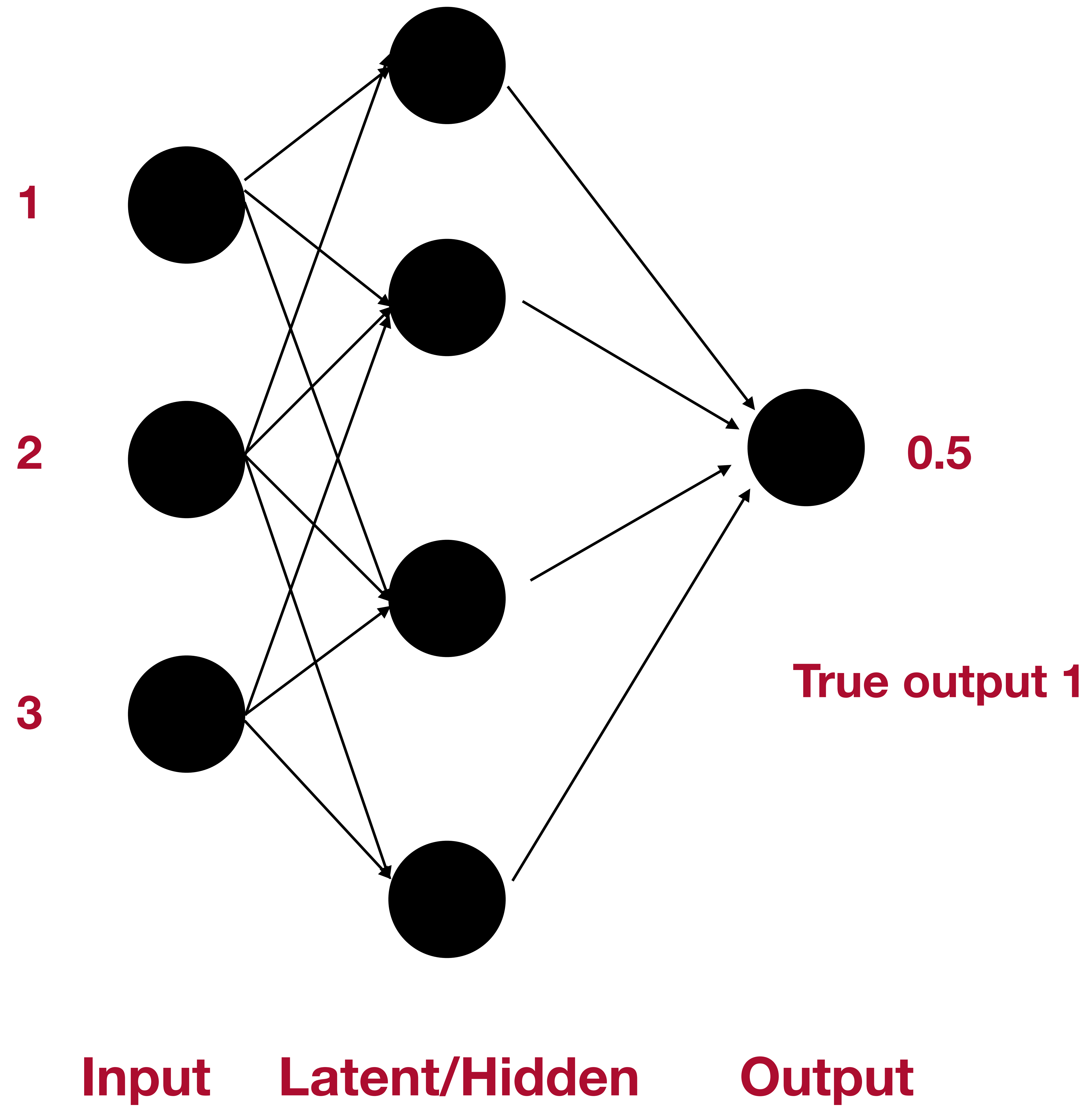
Another example with a one layer NN



Suppose that I give as input $\mathbf{X} = [1,2,3]$
and obtain as output $\hat{y} = 0.5$

How to measure the quality of the prediction?

Another example with a one layer NN

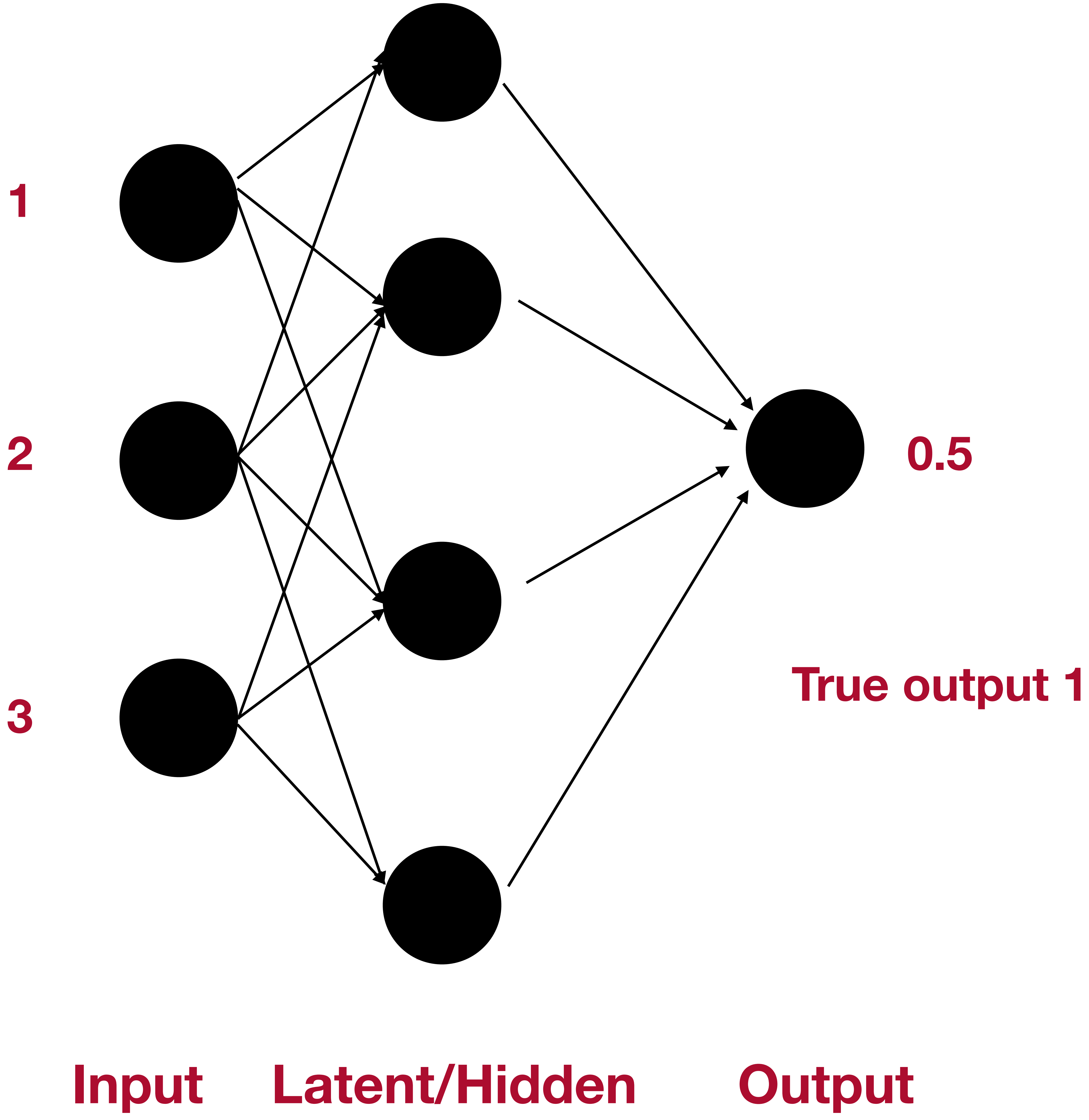


Suppose that I give as input $\mathbf{X} = [1,2,3]$
and obtain as output $\hat{y} = 0.5$

How to measure the quality of the prediction?

The loss of our network
will measure the cost of Incorrect predictions

Another example with a one layer NN



Suppose that I give as input $\mathbf{X} = [1,2,3]$
and obtain as output $\hat{y} = 0.5$

How to measure the quality of the prediction?

The loss of our network
will measure the cost of Incorrect predictions

$$\mathscr{L}(\mathbf{X}, \mathbf{Y}, \mathbf{W}, \sigma)$$

A glance at existing loss functions !

A glance at existing loss functions !

In what follows the Neural Network will be denoted as f_θ

A glance at existing loss functions !

In what follows the Neural Network will be denoted as f_θ

$\theta = [W_1, \cdots, W_k]$ **the set of weight matrices**

A glance at existing loss functions !

In what follows the Neural Network will be denoted as f_θ

$$\theta = [W_1, \cdots, W_k]$$ **the set of weight matrices**

each W_i **corresponds to the weights of the layer i**

A glance at existing loss functions !

f_θ **is obtained via composition of W_i and the activation functions**

In what follows the Neural Network will be denoted as f_θ

$$\theta = [W_1, \cdots, W_k]$$ **the set of weight matrices**

each W_i corresponds to the weights of the layer i

Empirical Losses

Empirical Losses

Neural Networks learn from data but so far we assumed no data

Empirical Losses

Neural Networks learn from data but so far we assumed no data

We have access to a dataset $\mathscr{D} = \{\mathbf{X}_i, y_i\}_0^N$

Empirical Losses

Neural Networks learn from data but so far we assumed no data

We have access to a dataset $\mathscr{D} = \{\mathbf{X}_i, y_i\}_0^N$

each y_i corresponds to label (dog/cat)

Empirical Losses

Neural Networks learn from data but so far we assumed no data

We have access to a dataset $\mathscr{D} = \{\mathbf{X}_i, y_i\}_0^N$

each y_i corresponds to label (dog/cat)

each $\mathbf{X}_i$ corresponds to an input sample (image)

Empirical Losses

Neural Networks learn from data but so far we assumed no data

We have access to a dataset $\mathscr{D} = \{\mathbf{X}_i, y_i\}_0^N$

each y_i corresponds to label (dog/cat)

each $\mathbf{X}_i$ corresponds to an input sample (image)

The empirical loss $J(\theta)$ **measures the total loss over our entire dataset**

Empirical Losses

Neural Networks learn from data but so far we assumed no data

We have access to a dataset $\mathscr{D} = \{\mathbf{X}_i, y_i\}_0^N$

each y_i corresponds to label (dog/cat)

each $\mathbf{X}_i$ corresponds to an input sample (image)

The empirical loss $J(\theta)$ measures the total loss over our entire dataset

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \mathscr{L}\left(f_\theta\left(\mathbf{X}_i\right), y_i\right)$$

Empirical Losses

Neural Networks learn from data but so far we assumed no data

We have access to a dataset $\mathscr{D} = \{\mathbf{X}_i, y_i\}_0^N$

each y_i corresponds to label (dog/cat)

each $\mathbf{X}_i$ corresponds to an input sample (image)

The empirical loss $J(\theta)$ measures the total loss over our entire dataset

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\mathscr{L}\left(\boxed{f_\theta\left(\mathbf{X}_i\right)}, y_i\right)$$

Prediction $\hat{y}_i$

Empirical Losses

Neural Networks learn from data but so far we assumed no data

We have access to a dataset $\mathscr{D} = \{\mathbf{X}_i, y_i\}_0^N$

each y_i corresponds to label (dog/cat)

each $\mathbf{X}_i$ corresponds to an input sample (image)

The empirical loss $J(\theta)$ measures the total loss over our entire dataset

Cost function

Empirical Risk

Objective function

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} \mathscr{L}\left(\boxed{f_\theta\left(\mathbf{X}_i\right)}, y_i\right)$$

Prediction $\hat{y}_i$

Empirical Losses

Neural Networks learn from data but so far we assumed no data

We have access to a dataset $\mathscr{D} = \{\mathbf{X}_i, y_i\}_0^N$

each y_i corresponds to label (dog/cat)

each $\mathbf{X}_i$ corresponds to an input sample (image)

The empirical loss $J(\theta)$ measures the total loss over our entire dataset

Cost function

Empirical Risk

Objective function

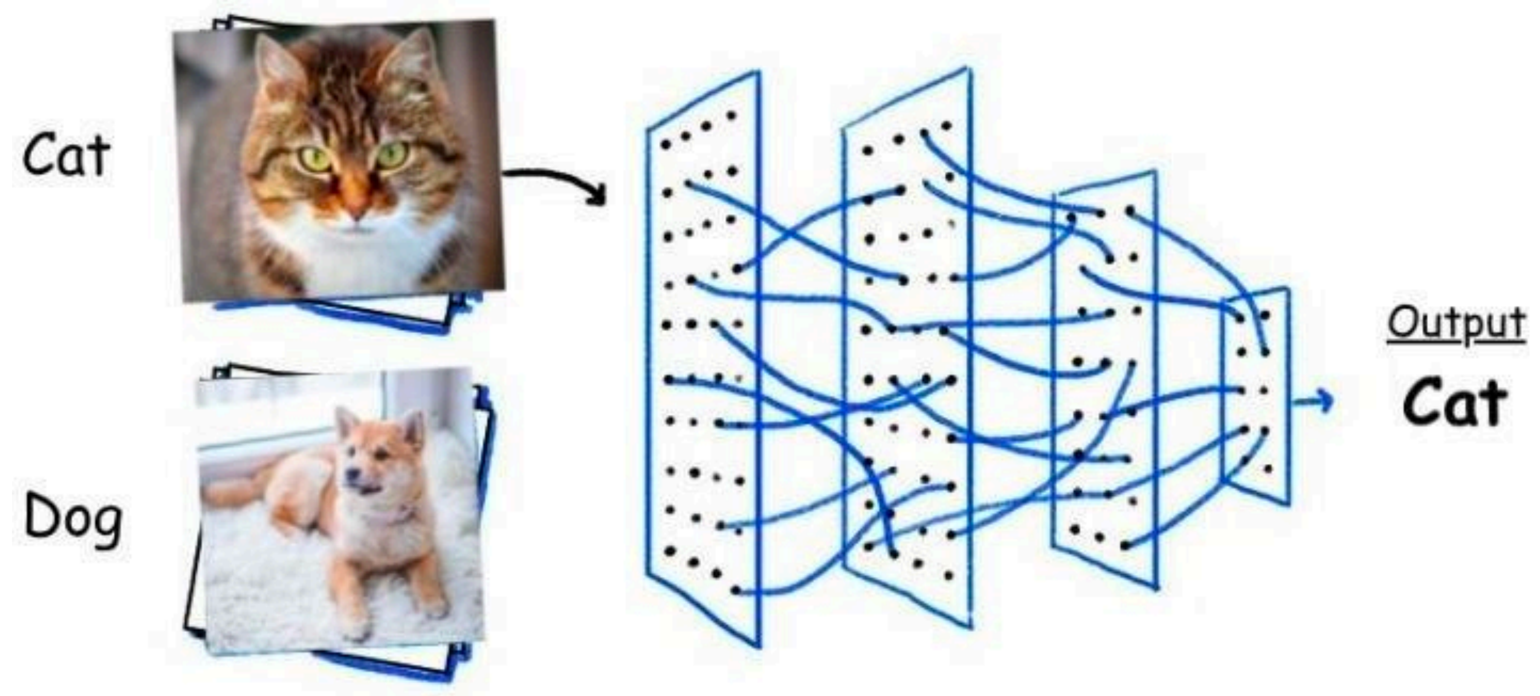
$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \mathscr{L}\left(\boxed{f_\theta(\mathbf{X}_i)}, y_i\right)$$

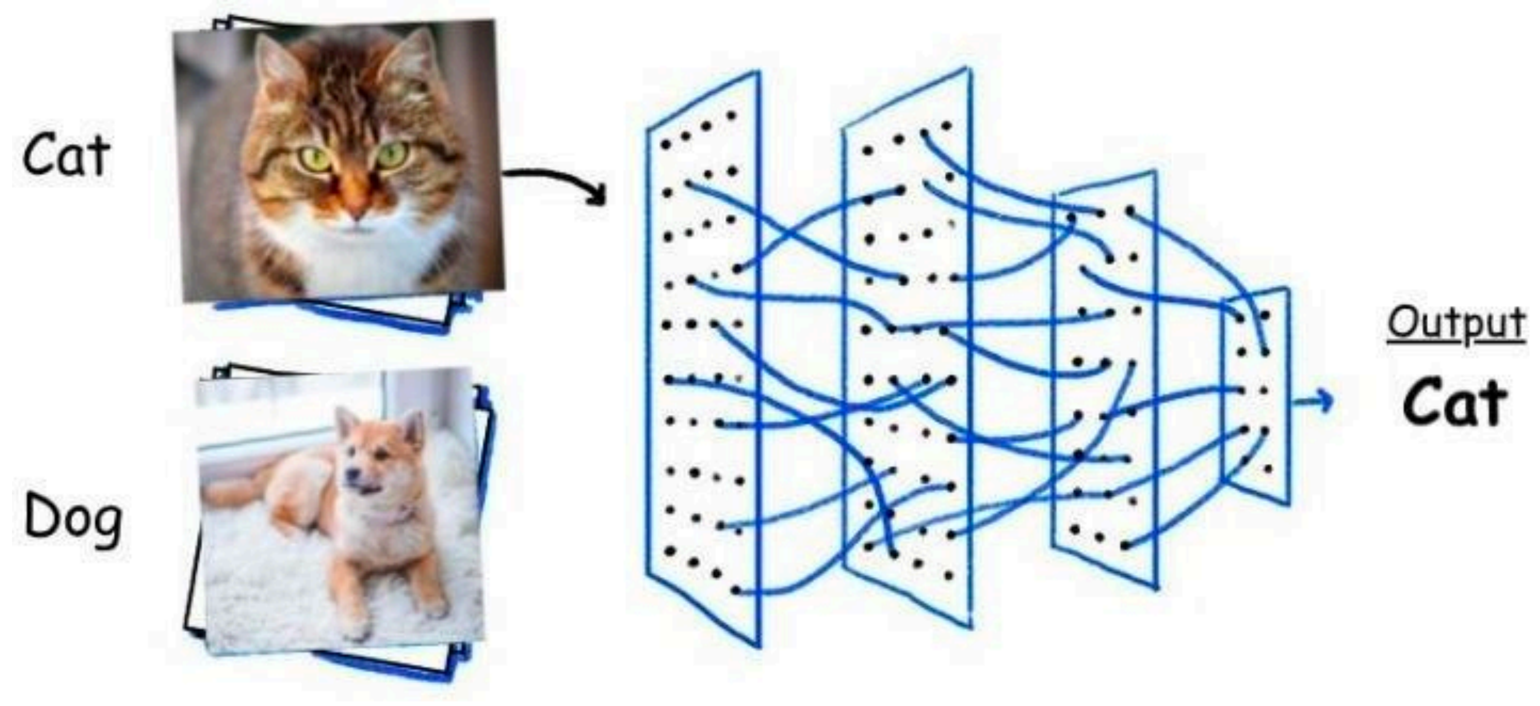
Prediction $\hat{y}_i$

Each type of problem induces a different loss function (e.g. Classification, Regression)

Classification via Cross Entropy Losses

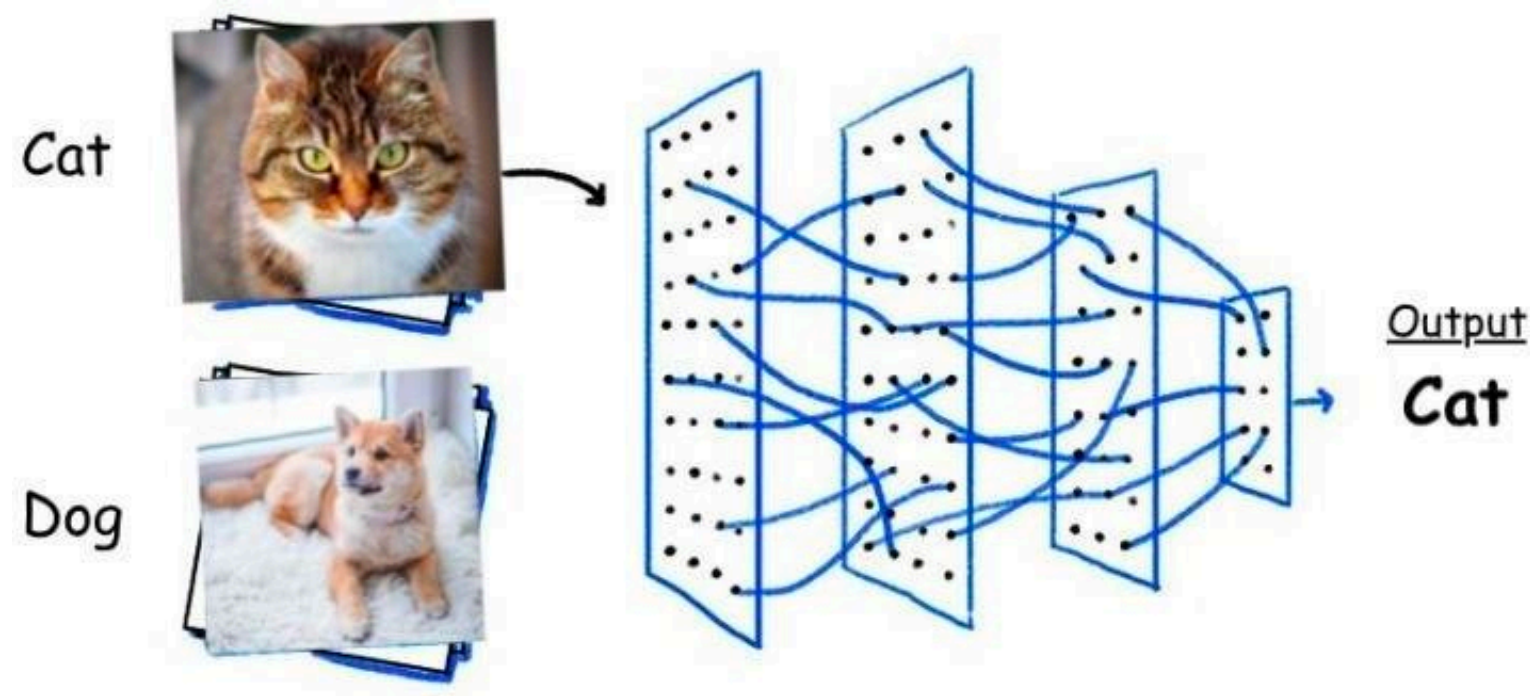
Classification via Cross Entropy Losses





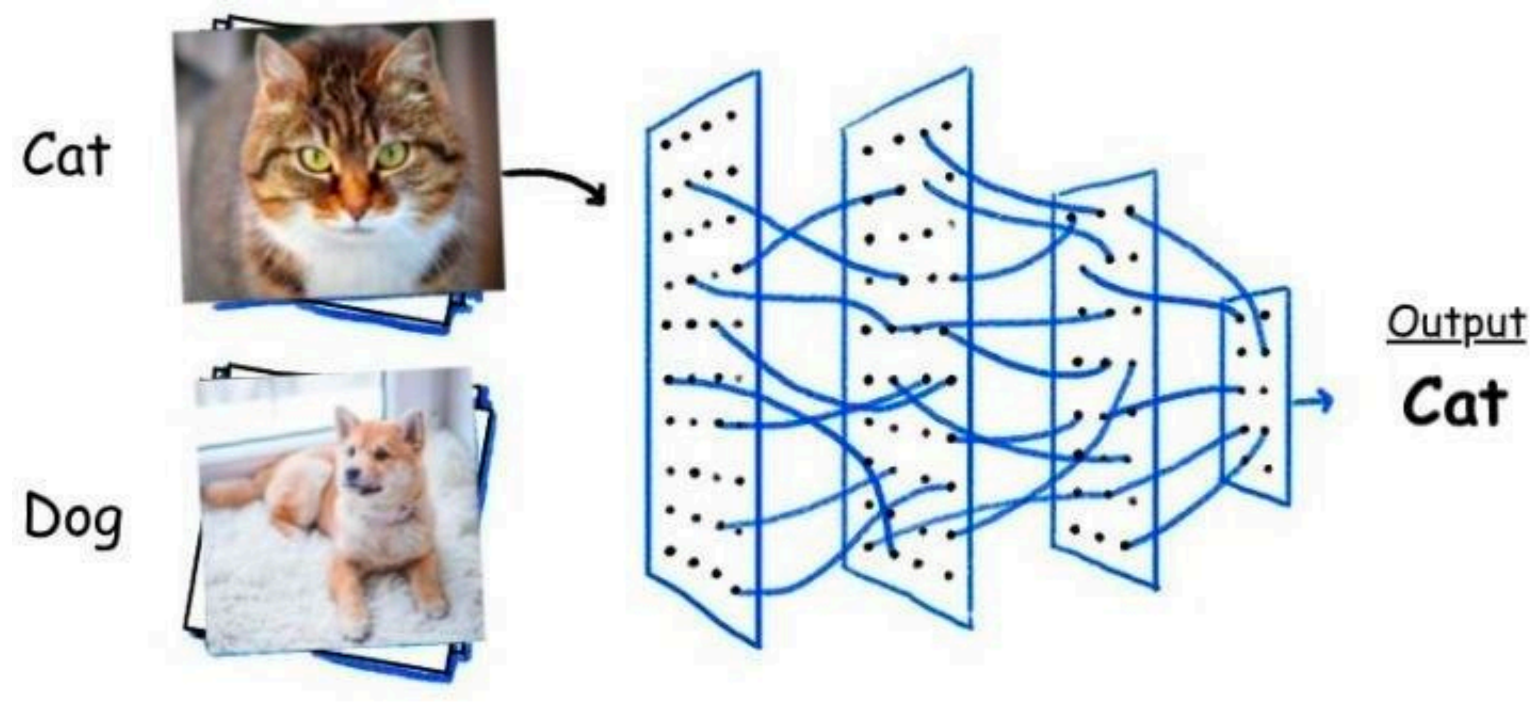
We want to classify an input picture as a cat or a dog

Classification via Cross Entropy Losses



We want to $\boxed{\text{classify}}$ an input picture as a cat or a dog

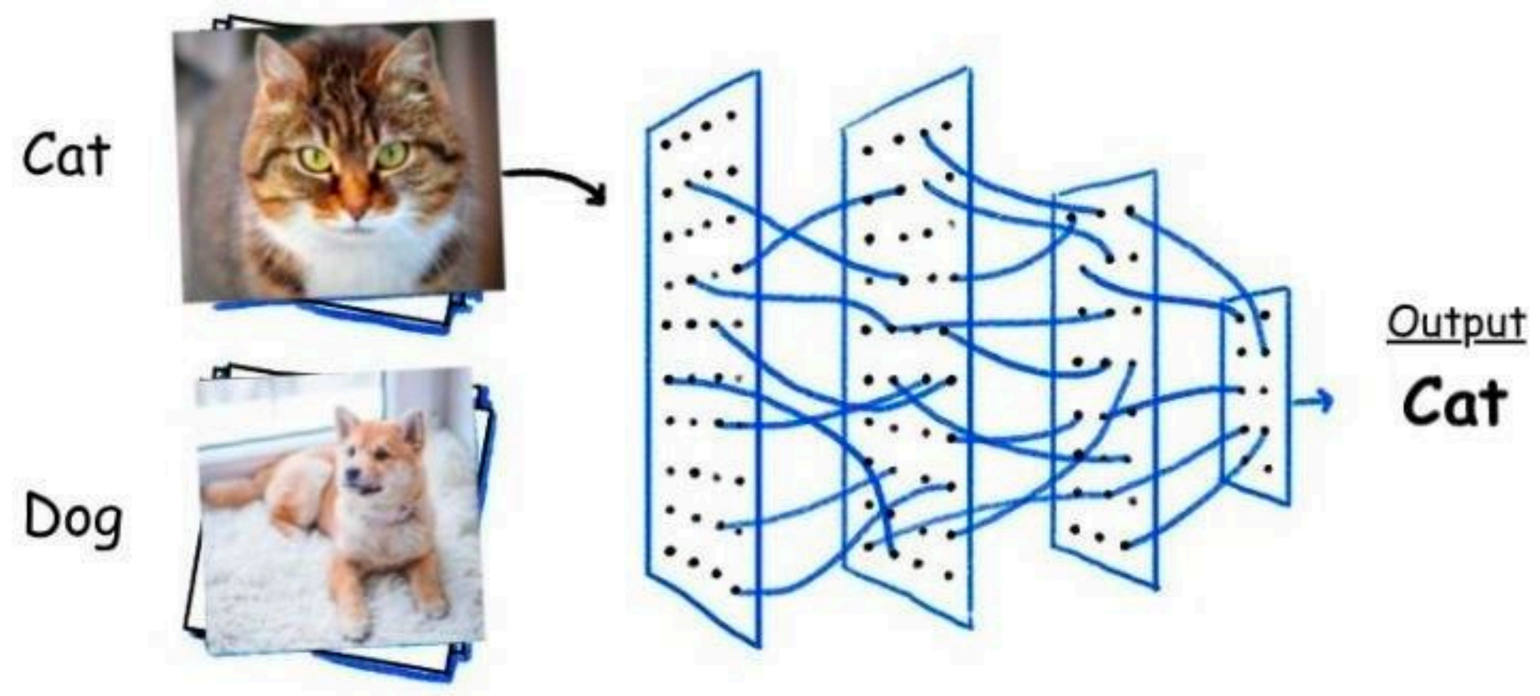
Classification via Cross Entropy Losses



We want to $\boxed{\text{classify}}$ an input picture as a cat or a dog

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \mathscr{L}\left(f_\theta\left(\mathbf{X}^i\right), y^i\right) \longrightarrow$$

Classification via Cross Entropy Losses



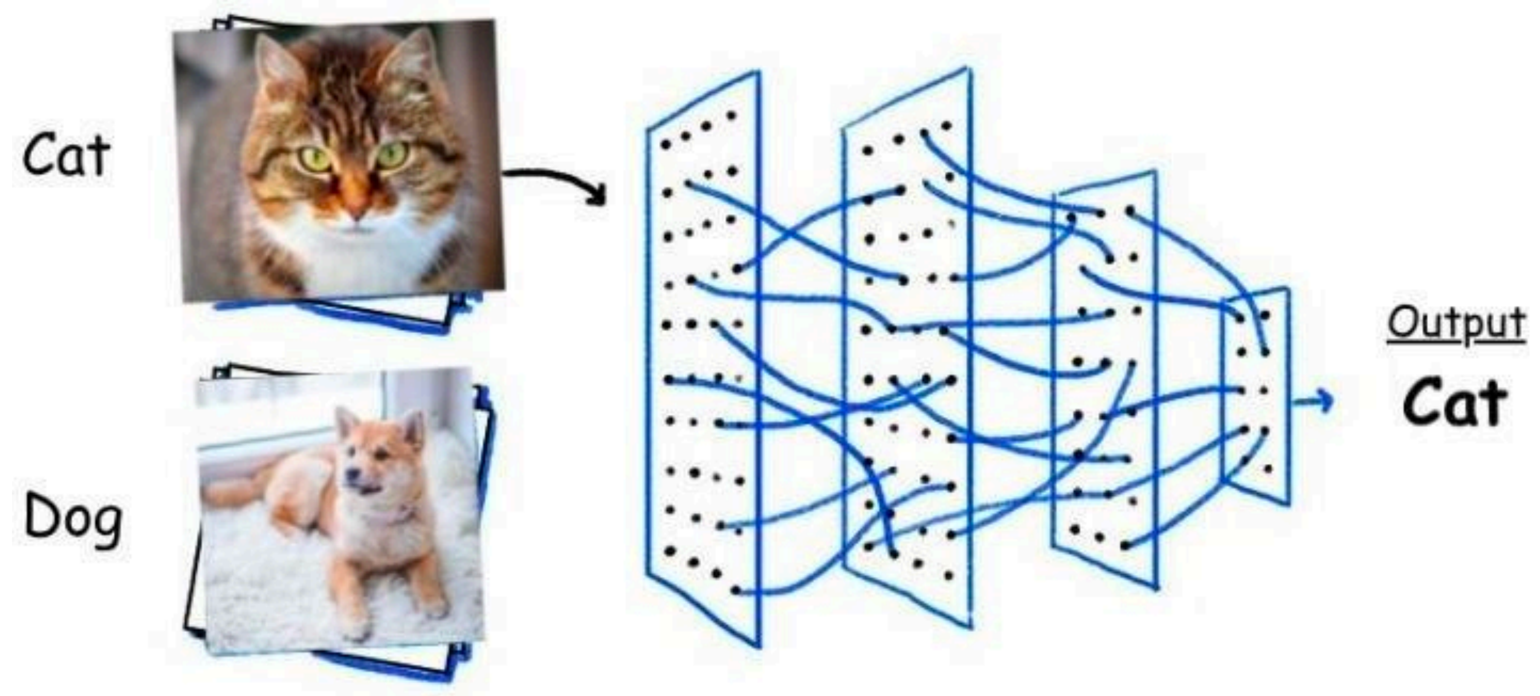
We want to $\boxed{\text{classify}}$ an input picture as a cat or a dog

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \mathscr{L}\left(f_\theta\left(\mathbf{X}^i\right), y^i\right)$$

$\longrightarrow$

We will rely on the binary cross entropy

The cross-entropy loss can be used with a classification model



We want to $\boxed{\text{classify}}$ an input picture as a cat or a dog

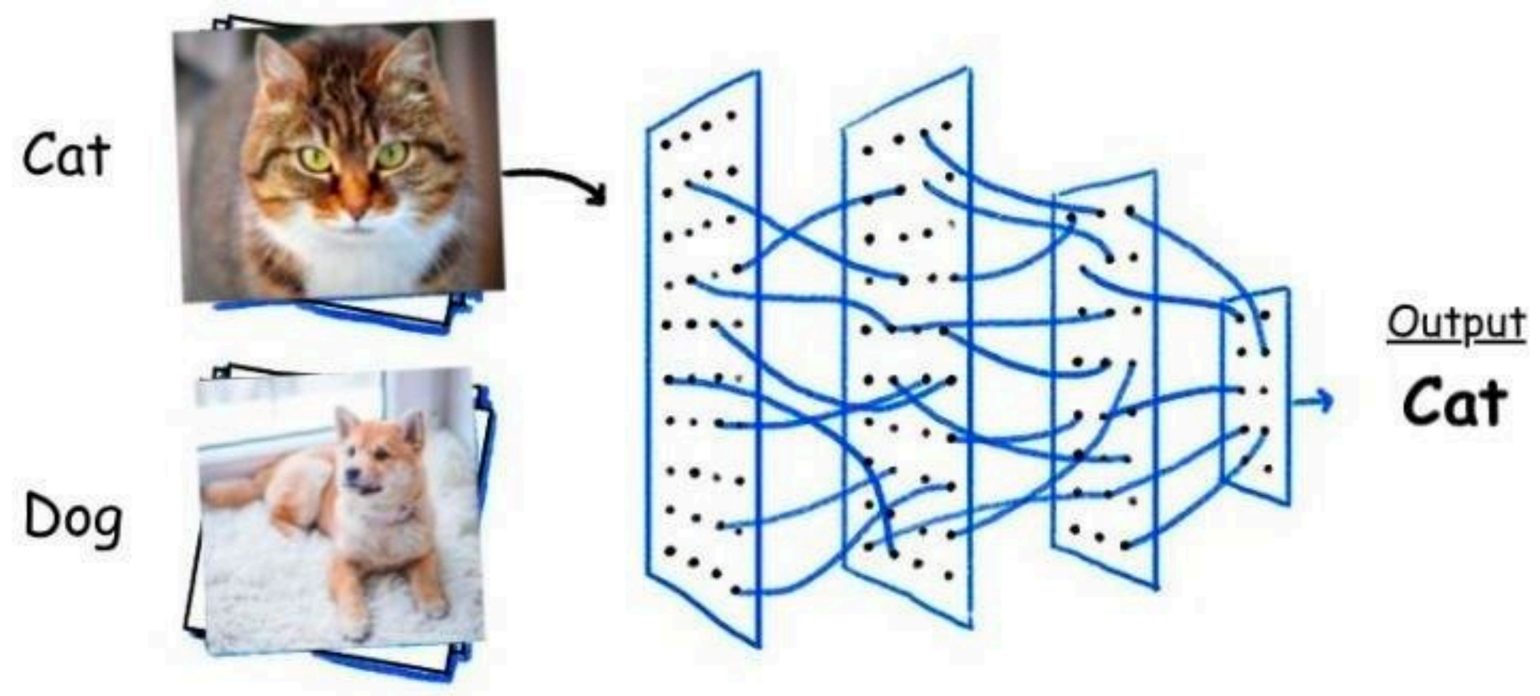
$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} \mathscr{L}\left(f_\theta\left(\mathbf{X}^i\right), y^i\right)$$

We will rely on the binary cross entropy

The cross-entropy loss can be used with a classification model

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} y_i \times \log\left(f_\theta\left(\mathbf{X}^i\right)\right) + (1-y_i) \times \log\left(1 - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right)$$

Classification via Cross Entropy Losses



We want to $\boxed{\text{classify}}$ an input picture as a cat or a dog

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \mathscr{L}\left(f_\theta\left(\mathbf{X}^i\right), y^i\right)$$

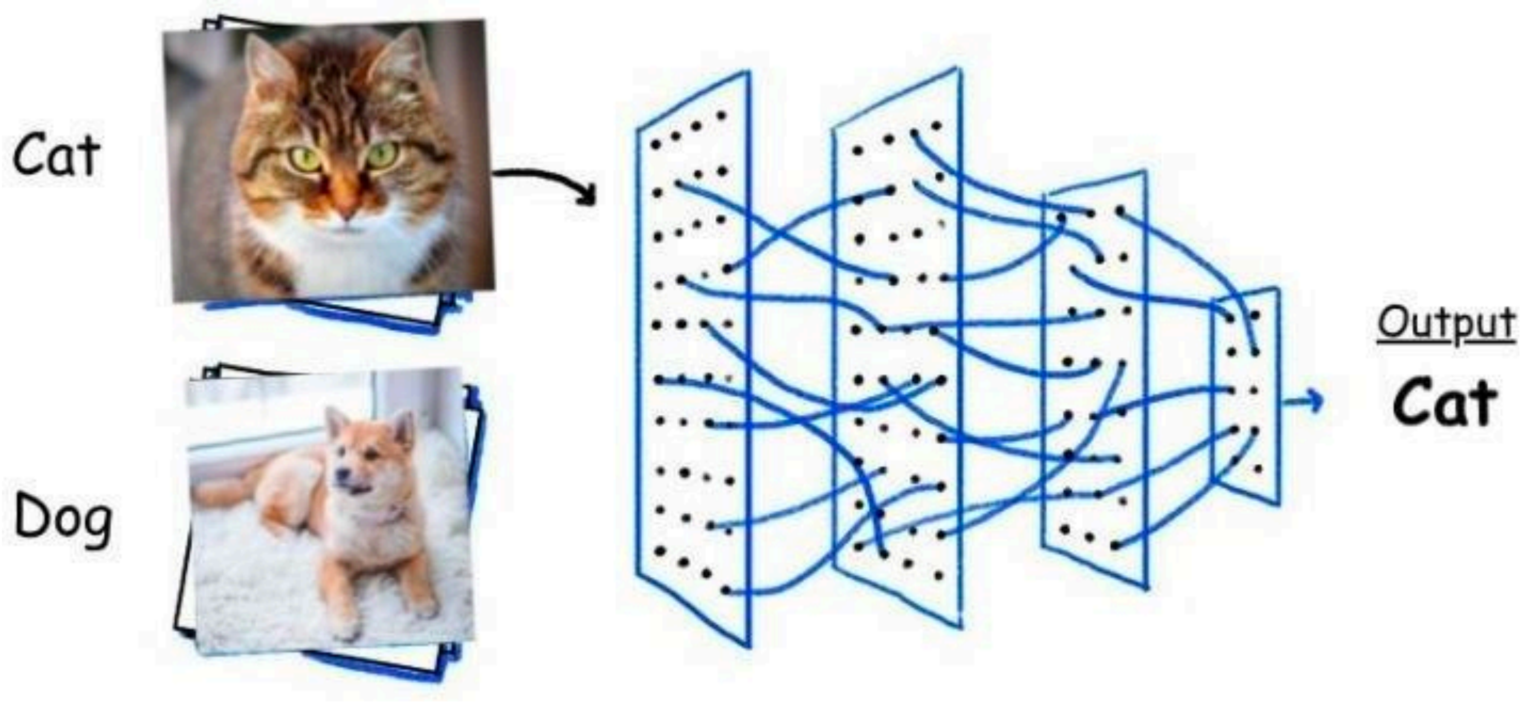
We will rely on the binary cross entropy

The cross-entropy loss can be used with a classification model

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} y_i \times \log\left(f_\theta\left(\mathbf{X}^i\right)\right) + (1 - \boxed{y_i)} \times \log\left(1 - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right)$$

True label

Classification via Cross Entropy Losses



We want to $\boxed{\text{classify}}$ an input picture as a cat or a dog

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \mathscr{L}\left(f_\theta\left(\mathbf{X}^i\right), y^i\right) \longrightarrow$$

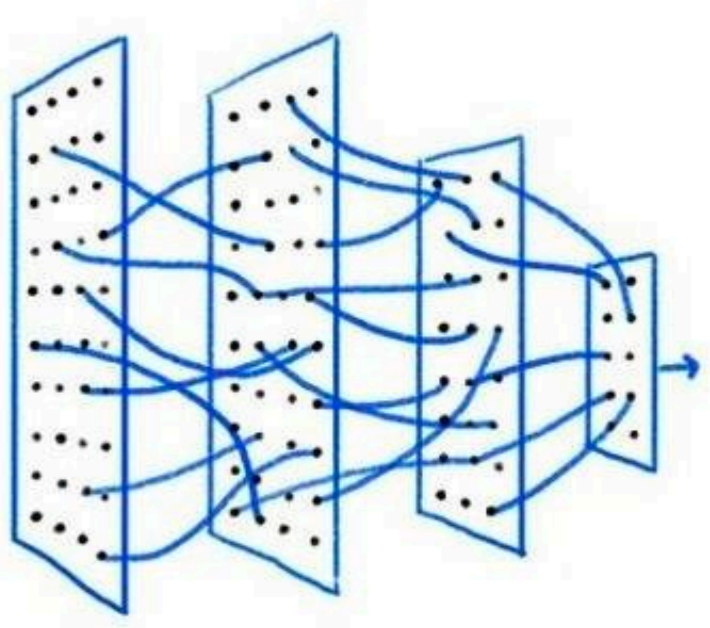
We will rely on the binary cross entropy

The cross-entropy loss can be used with a classification model

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} y_i \times \log \underbrace{\boxed{\left(f_\theta\left(\mathbf{X}^i\right)\right)}}_{\text{Prediction } \hat{y}_i} + (1 - \underbrace{\boxed{y_i}}_{\text{True label}}) \times \log\left(1 - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right)$$

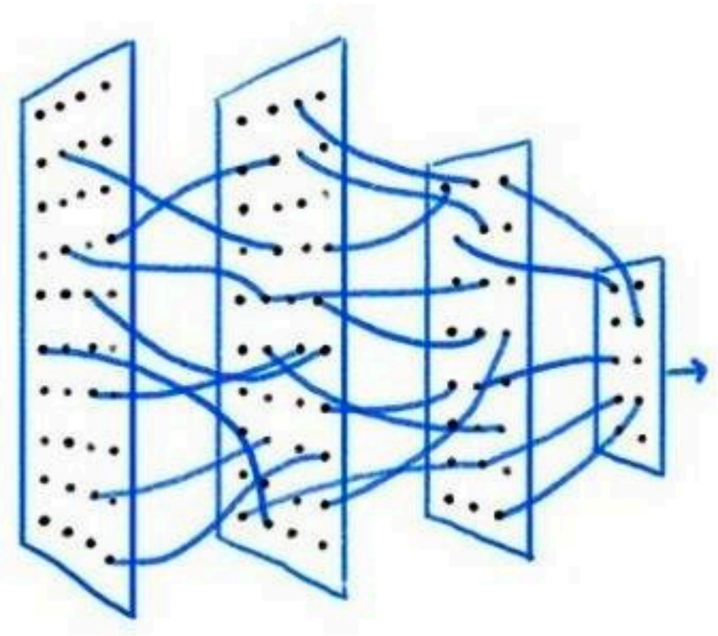
Regression via Mean Square Error or Mean Absolute Error

Regression via Mean Square Error or Mean Absolute Error



Output

Speed

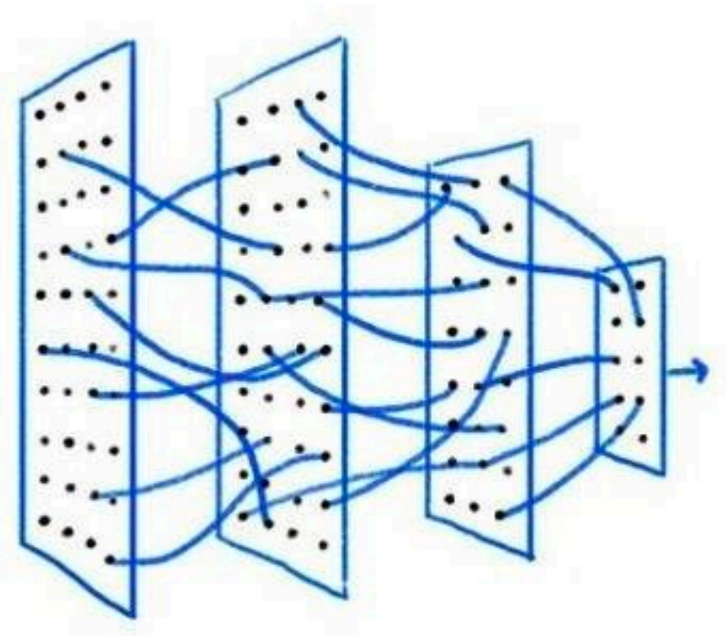


Output

Speed

We want to predict the maximum speed of the car

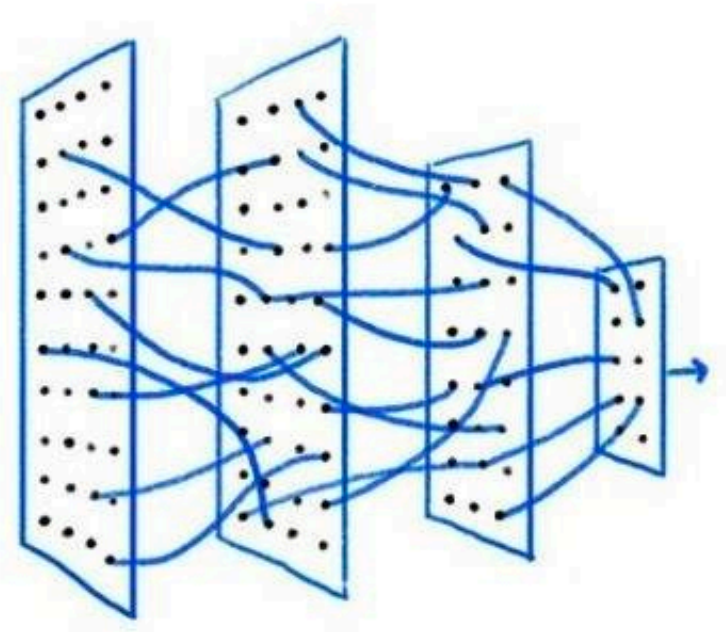
Regression via Mean Square Error or Mean Absolute Error



Real value!

We want to predict the maximum speed of the car

Regression via Mean Square Error or Mean Absolute Error



Output

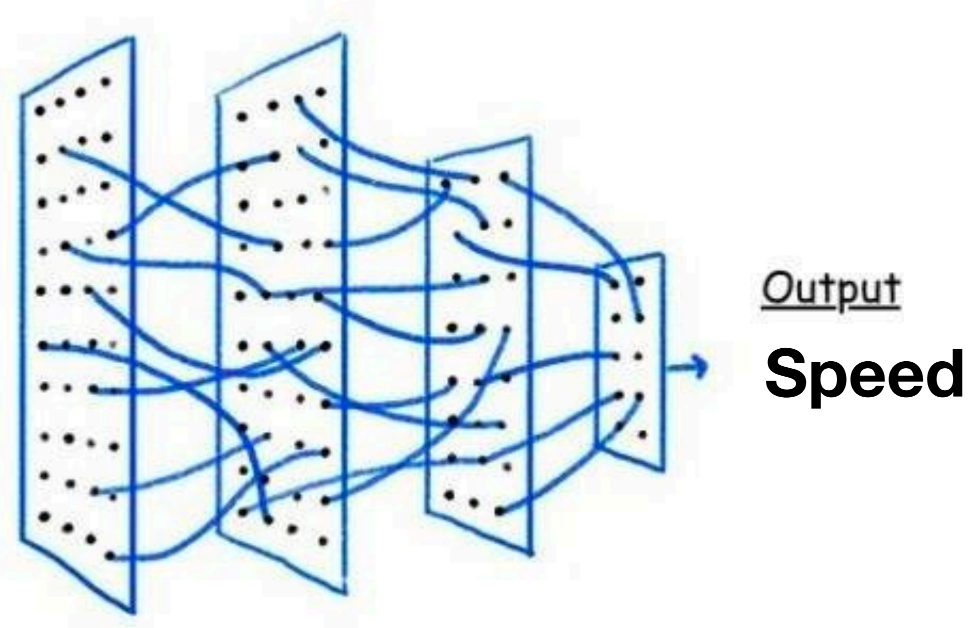
Speed

Real value!

We want to predict the $\boxed{\text{maximum speed of the car}}$

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \mathscr{L}\left(f_\theta\left(\mathbf{X}^i\right), y^i \right)$$

Regression via Mean Square Error or Mean Absolute Error



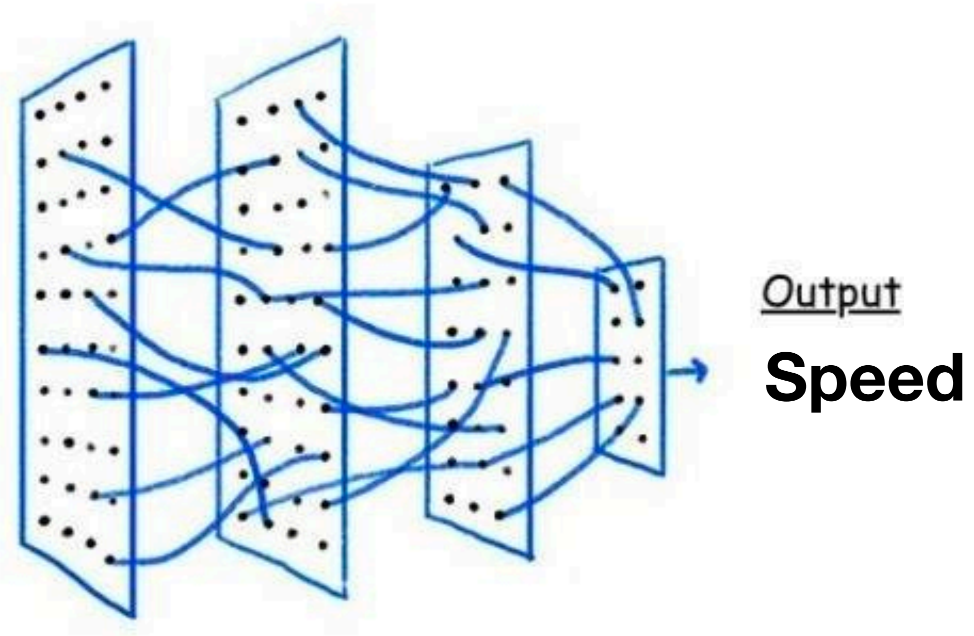
Real value!

We want to predict the $\boxed{\text{maximum speed of the car}}$

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \mathscr{L}\left(f_\theta\left(\mathbf{X}^i\right), y^i\right) \longrightarrow$$

We will rely on the mean square error

The MAE or MSE loss can be used with a regression model



Real value!

We want to predict the $\boxed{\text{maximum speed of the car}}$

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \mathscr{L}\left(f_\theta\left(\mathbf{X}^i\right), y^i\right)$$

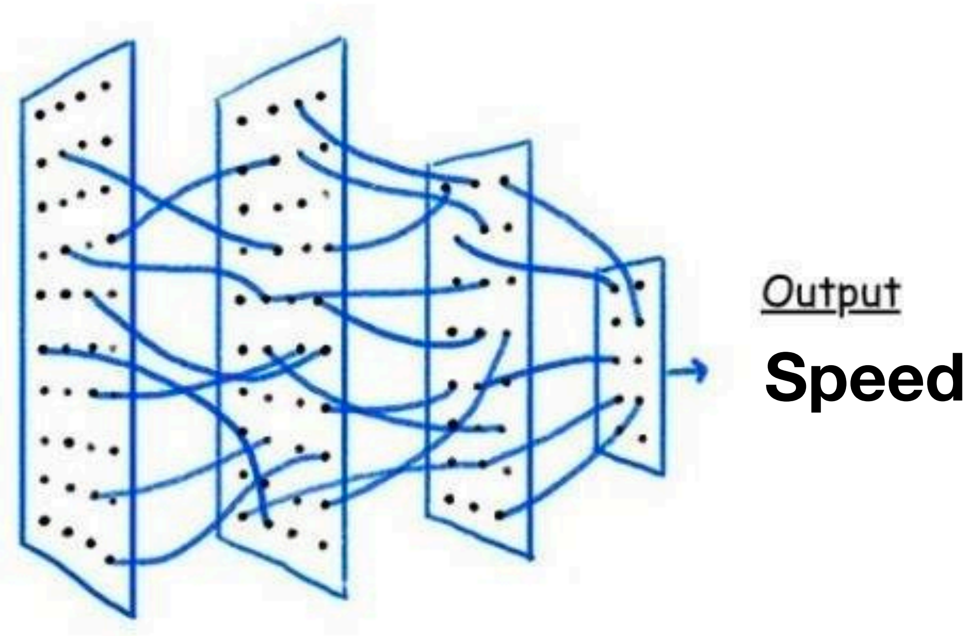
We will rely on the mean square error

The **MAE or MSE loss** can be used with a regression model

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right)^2$$

Mean Square Error

Regression via Mean Square Error or Mean Absolute Error



Real value!

We want to predict the $\boxed{\text{maximum speed of the car}}$

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} \mathscr{L}\left(f_\theta\left(\mathbf{X}^i\right), y^i\right)$$

We will rely on the mean square error

The MAE or MSE loss can be used with a regression model

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} \left(y_i - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right)^2$$

Mean Square Error

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} \left|y_i - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right|$$

Mean Absolute Error

Let’s sum up so far

Let's sum up so far

1. The basic bloc to build a Neural Network

We have seen the forward propagation, i.e. from an input how to use the Neural Network

This operation is call forward propagation, this operation is used at inference time

2. The role of activation functions

They complexity the neural network and allow to fit more complex data

They are added at the end of each layer and there is a large variety of them

Let’s sum up so far

1. The basic bloc to build a Neural Network

We have seen the forward propagation, i.e. from an input how to use the Neural Network

This operation is call forward propagation, this operation is used at inference time

2. The role of activation functions

They complexity the neural network and allow to fit more complex data

They are added at the end of each layer and there is a large variety of them

3. The cost function

It allows to measure how « far » is the neural network from the reality

How to perform the « learning phase »?

How to optimize the weights (learning algorithm)

How to optimize the weights (learning algorithm)

We want to find the network's weight that achieve the lowest cost/loss

We want to find the network's weight that achieve the lowest cost/loss

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \; J(\theta)$$

How to optimize the weights (learning algorithm)

We want to find the network's weight that achieve the lowest cost/loss

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \quad J(\theta)$$

Remember: our loss is a function of the network weights $\theta = [W_1, \cdots, W_k]$

How to optimize the weights (learning algorithm)

We want to find the network's weight that achieve the lowest cost/loss

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \quad J(\theta)$$

Remember: our loss is a function of the network weights $\theta = [W_1, \cdots, W_k]$

We will use gradient descent to find the best weights

Loss Optimization via Gradient Descent

Loss Optimization via Gradient Descent

Requirements: **Gradient Descent can be done if we can compute the derivative of** $J(\theta)$

Loss Optimization via Gradient Descent

Requirements: **Gradient Descent can be done if we can compute the derivative of** $J(\theta)$

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right)^2 \qquad J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right| \qquad J(\theta) = \frac{1}{N}\sum_{i=1}^{N} y_i \times \log\left(f_\theta\left(\mathbf{X}^i\right)\right) + (1 - y_i) \times \log\left(1 - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right)$$

Loss Optimization via Gradient Descent

Requirements: **Gradient Descent can be done if we can compute the derivative of** $J(\theta)$

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right)^2 \qquad J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right| \qquad J(\theta) = \frac{1}{N}\sum_{i=1}^{N} y_i \times \log\left(f_\theta\left(\mathbf{X}^i\right)\right) + (1 - y_i) \times \log\left(1 - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right)$$

Goal: **Find the best neural network weights**

Loss Optimization via Gradient Descent

Requirements: **Gradient Descent can be done if we can compute the derivative of** $J(\theta)$

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right)^2 \qquad J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right| \qquad J(\theta) = \frac{1}{N}\sum_{i=1}^{N} y_i \times \log\left(f_\theta\left(\mathbf{X}^i\right)\right) + (1 - y_i) \times \log\left(1 - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right)$$

Goal: **Find the best neural network weights**

1. **Initialize** θ

Loss Optimization via Gradient Descent

Requirements: **Gradient Descent can be done if we can compute the derivative of** $J(\theta)$

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right)^2 \qquad J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right| \qquad J(\theta) = \frac{1}{N}\sum_{i=1}^{N} y_i \times \log\left(f_\theta\left(\mathbf{X}^i\right)\right) + (1 - y_i) \times \log\left(1 - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right)$$

Goal: **Find the best neural network weights**

1. Initialize θ

Usually sampled from a Gaussian distribution

Loss Optimization via Gradient Descent

Requirements: **Gradient Descent can be done if we can compute the derivative of** $J(\theta)$

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right)^2 \qquad J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right| \qquad J(\theta) = \frac{1}{N}\sum_{i=1}^{N} y_i \times \log\left(f_\theta\left(\mathbf{X}^i\right)\right) + (1 - y_i) \times \log\left(1 - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right)$$

Goal: **Find the best neural network weights**

1. **Initialize** θ

Usually sampled from a Gaussian distribution

2. **While it has not converged**

Loss Optimization via Gradient Descent

Requirements: **Gradient Descent can be done if we can compute the derivative of** $J(\theta)$

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right)^2 \qquad J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right| \qquad J(\theta) = \frac{1}{N}\sum_{i=1}^{N} y_i \times \log\left(f_\theta\left(\mathbf{X}^i\right)\right) + (1 - y_i) \times \log\left(1 - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right)$$

Goal: **Find the best neural network weights**

1. Initialize θ — **Usually sampled from a Gaussian distribution**

2. While it has not converged

Compute the gradient $\nabla_\theta J(\theta)$ — **This operation is called back-propagation**

Loss Optimization via Gradient Descent

Requirements: Gradient Descent can be done if we can compute the derivative of $J(\theta)$

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right)^2 \qquad J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right| \qquad J(\theta) = \frac{1}{N}\sum_{i=1}^{N} y_i \times \log\left(f_\theta\left(\mathbf{X}^i\right)\right) + (1 - y_i) \times \log\left(1 - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right)$$

Goal: Find the best neural network weights

1. Initialize θ — Usually sampled from a Gaussian distribution

2. While it has not converged

   Compute the gradient $\nabla_\theta J(\theta)$ — This operation is called back-propagation

   Update the weights $\theta = \theta - \lambda \nabla_\theta J(\theta)$ — This operation is call gradient update

Loss Optimization via Gradient Descent

Requirements: Gradient Descent can be done if we can compute the derivative of $J(\theta)$

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right)^2 \qquad J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right| \qquad J(\theta) = \frac{1}{N}\sum_{i=1}^{N} y_i \times \log\left(f_\theta\left(\mathbf{X}^i\right)\right) + (1-y_i)\times\log\left(1-\left(f_\theta\left(\mathbf{X}^i\right)\right)\right)$$

Goal: Find the best neural network weights

1. Initialize θ — Usually sampled from a Gaussian distribution

2. While it has not converged

   Compute the gradient $\nabla_\theta J(\theta)$ — This operation is called back-propagation

   Update the weights $\theta = \theta - \boxed{\lambda}\nabla_\theta J(\theta)$ — This operation is call gradient update

λ is the learning rate

Loss Optimization via Gradient Descent

Requirements: Gradient Descent can be done if we can compute the derivative of $J(\theta)$

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right)^2 \qquad J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right| \qquad J(\theta) = \frac{1}{N}\sum_{i=1}^{N} y_i \times \log\left(f_\theta\left(\mathbf{X}^i\right)\right) + (1 - y_i) \times \log\left(1 - \left(f_\theta\left(\mathbf{X}^i\right)\right)\right)$$

Goal: Find the best neural network weights

1. Initialize θ — Usually sampled from a Gaussian distribution

2. While it has not converged

   Compute the gradient $\nabla_\theta J(\theta)$ — This operation is called back-propagation

   λ is the learning rate

   Update the weights $\theta = \theta - \boxed{\lambda}\nabla_\theta J(\theta)$ — This operation is call gradient update

3. Return the final weights θ — θ should be closed to $\theta*$

Some remarks

Some remarks

About Backpropagation

This step compute the gradient of the error w.r.t to the weights

This step is compute intensive. This step is usually done on GPU

For most of the Neural Network you won't be able to write the close form solution.

Some remarks

About Backpropagation

This step compute the gradient of the error w.r.t to the weights

This step is compute intensive. This step is usually done on GPU

For most of the Neural Network you won't be able to write the close form solution.

About the optimization algorithm

Finding a good learning rate λ is crucial in practice.

There exists many alternative to Gradient Descent (e.g. SGD, Adam, RMSProp)

Some remarks

About Backpropagation

This step compute the gradient of the error w.r.t to the weights

This step is compute intensive. This step is usually done on GPU

For most of the Neural Network you won't be able to write the close form solution.

About the optimization algorithm

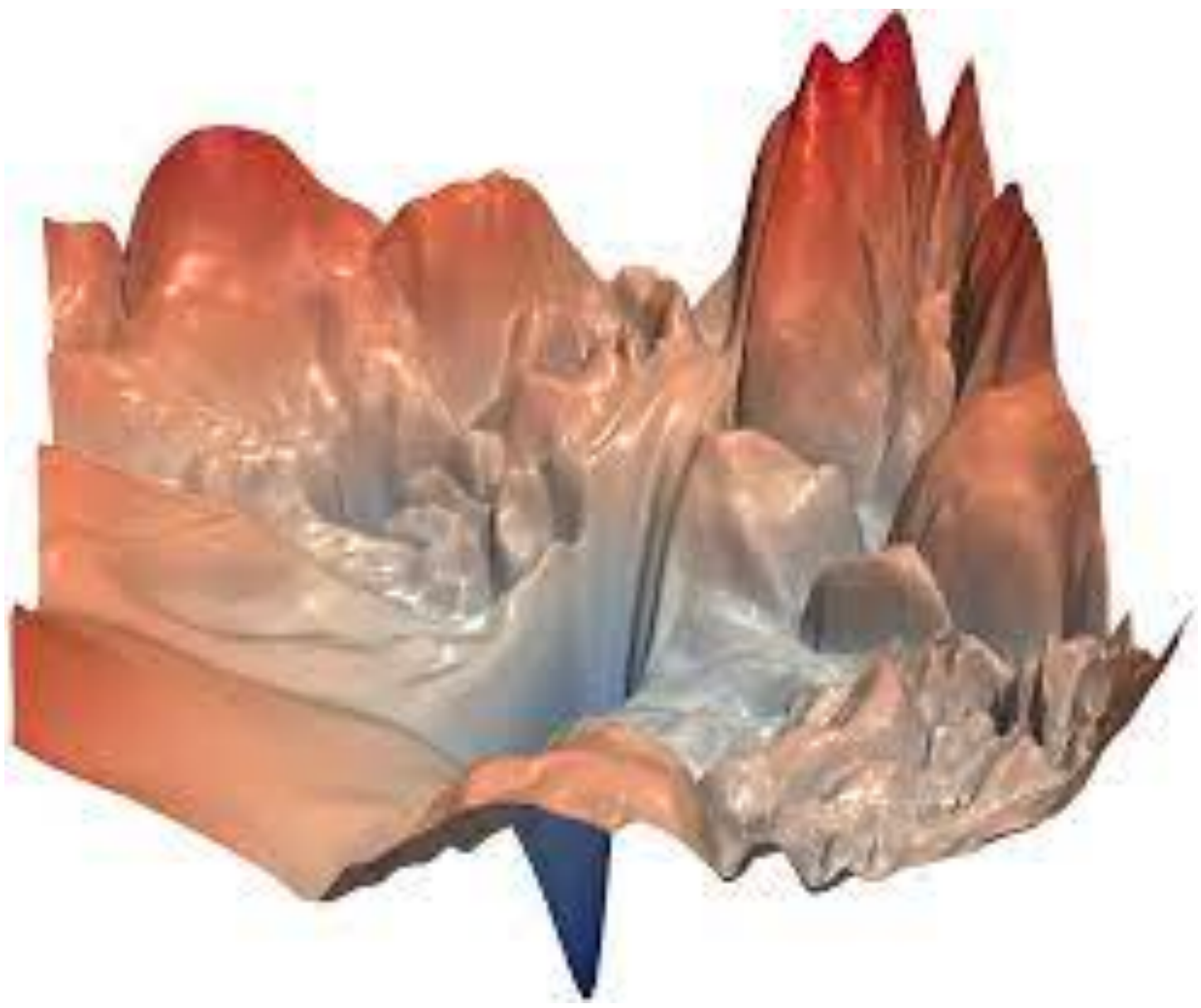
Finding a good learning rate λ is crucial in practice.

There exists many alternative to Gradient Descent (e.g. SGD, Adam, RMSProp)

Speed, speed, speed……

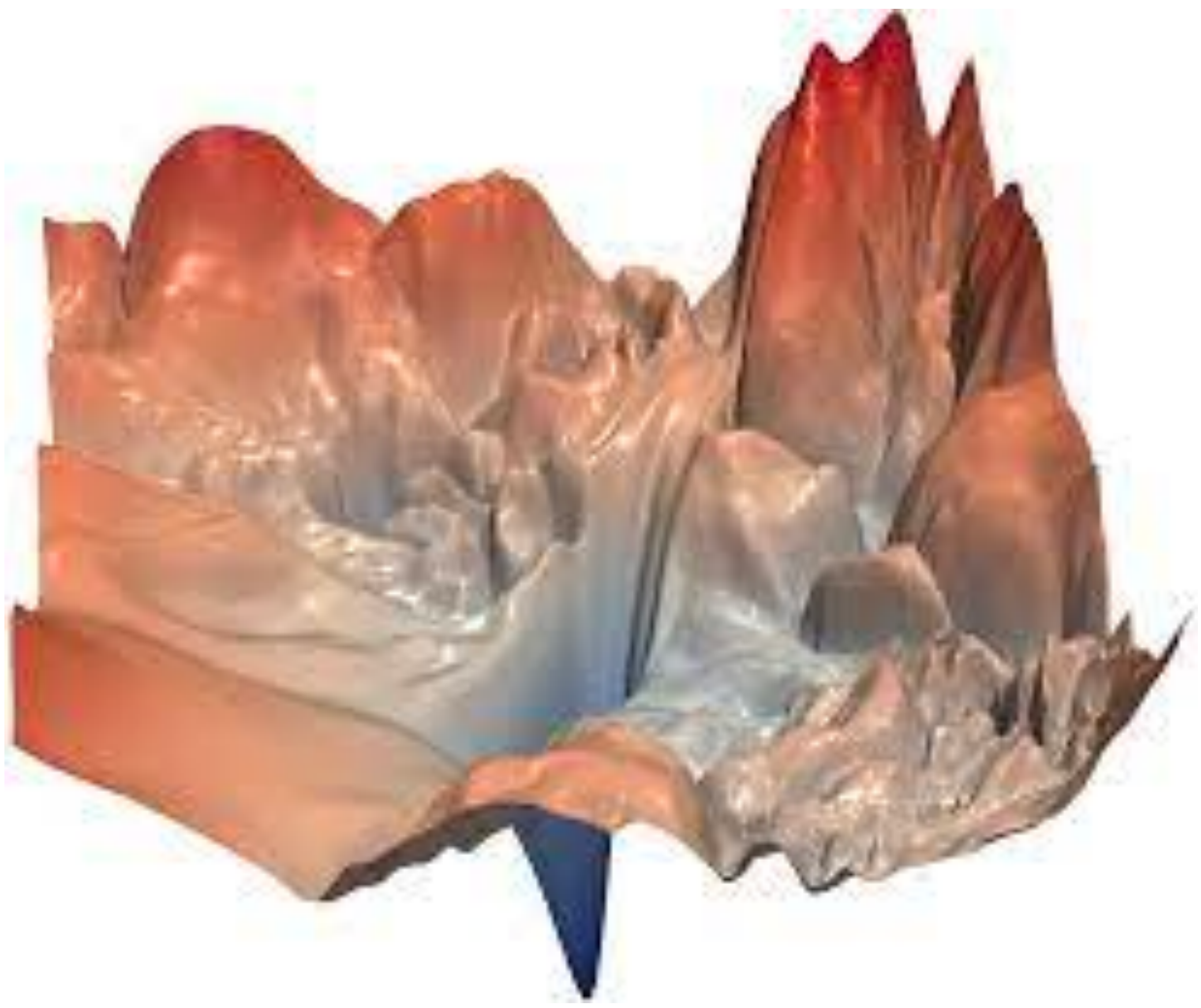
The difficulty to train neural networks

The difficulty to train neural networks



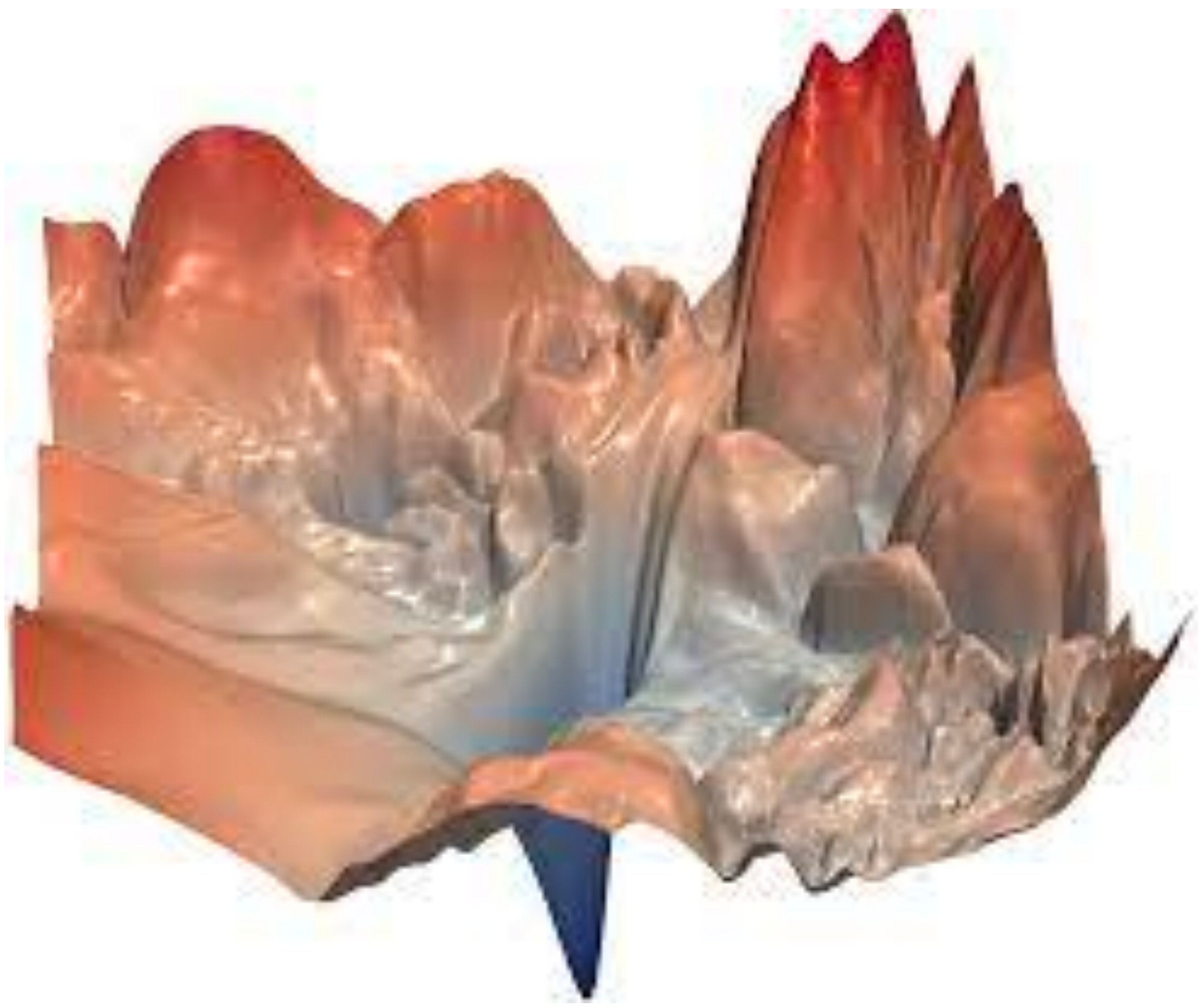
Example of loss landscape of NN

The difficulty to train neural networks



Example of loss landscape of NN

What do you notice? →

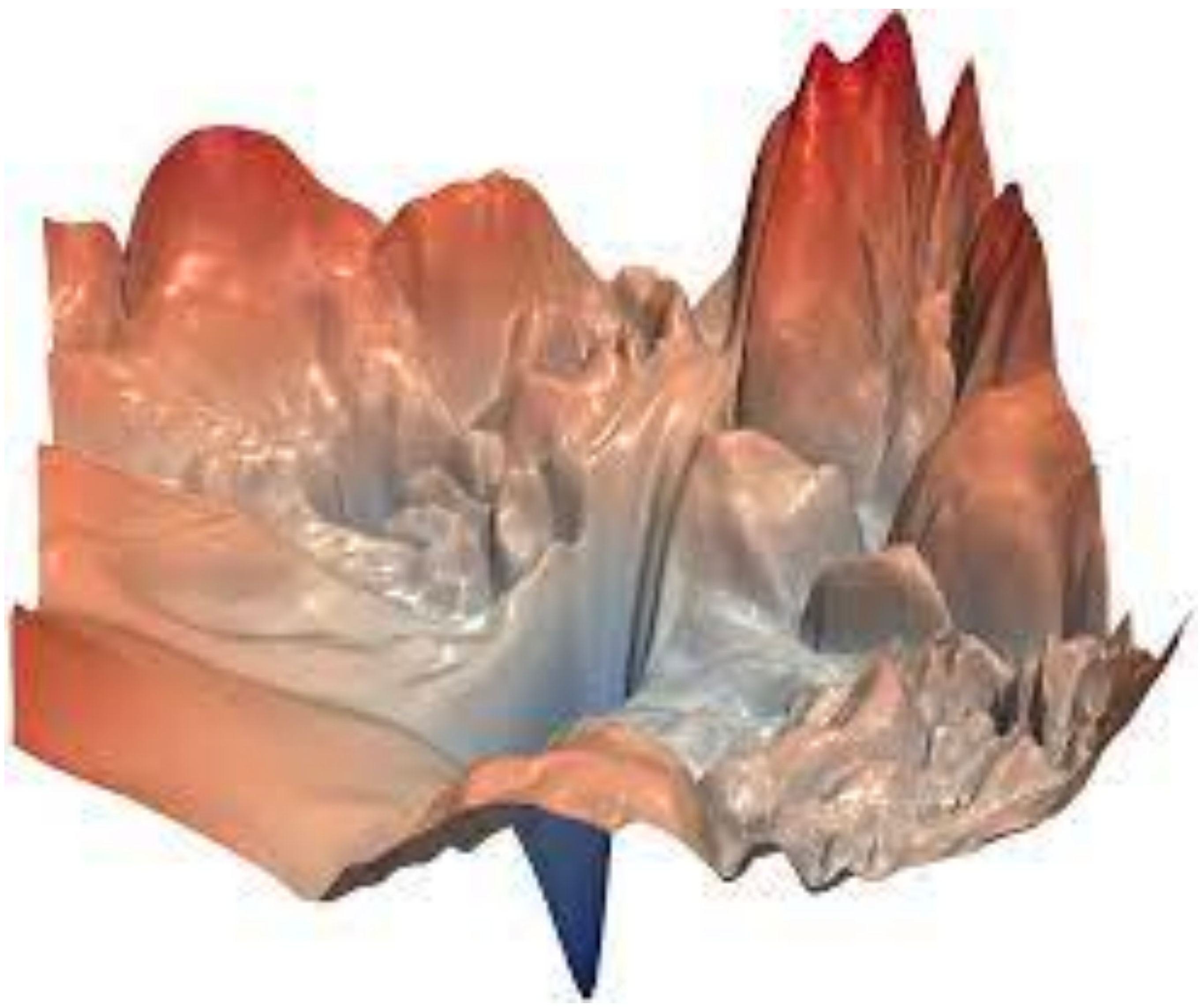


Example of loss landscape of NN

What do you notice?

This is not convex at all !

The difficulty to train neural networks



Example of loss landscape of NN

What do you notice?

This is not convex at all !

Gradient descent can be stuck in local optima

Initialization will matter a lot!

Setting the learning rate will also matter!

How to deal with it? Some recipe

How to deal with it? Some recipe

Learning rates a now adaptive

« adapt them to the loss landscape »

Can be made larger or small depending on the problem

How to deal with it? Some recipe

Learning rates a now adaptive

« adapt them to the loss landscape »

Can be made larger or small depending on the problem

Use Mini-batches for speed and stability

« Instead of computing the gradient on the whole dataset use a subset »

Smoother convergence + allows for larger learning rate

How to deal with it? Some recipe

Learning rates a now adaptive

« adapt them to the loss landscape »

Can be made larger or small depending on the problem

Use Mini-batches for speed and stability

« Instead of computing the gradient on the whole dataset use a subset »

Smoother convergence + allows for larger learning rate

Change the neural network architecture

Add more data

How to deal with it? Some recipe

Learning rates a now adaptive

« adapt them to the loss landscape »

Can be made larger or small depending on the problem

Use Mini-batches for speed and stability

« Instead of computing the gradient on the whole dataset use a subset »

Smoother convergence + allows for larger learning rate

Change the neural network architecture

Add more data

Those will always work!

Let's sum up so far

Let's sum up so far

1. The basic bloc to build a Neural Network

We have seen the forward propagation, i.e. from an input how to use the Neural Network

This operation is call forward propagation, this operation is used at inference time

2. The role of activation functions

They complexity the neural network and allow to fit more complex data

They are added at the end of each layer and there is a large variety of them

3. The cost function

It allows to measure how « far » is the neural network from the reality

Let’s sum up so far

1. The basic bloc to build a Neural Network

We have seen the forward propagation, i.e. from an input how to use the Neural Network

This operation is call forward propagation, this operation is used at inference time

2. The role of activation functions

They complexity the neural network and allow to fit more complex data

They are added at the end of each layer and there is a large variety of them

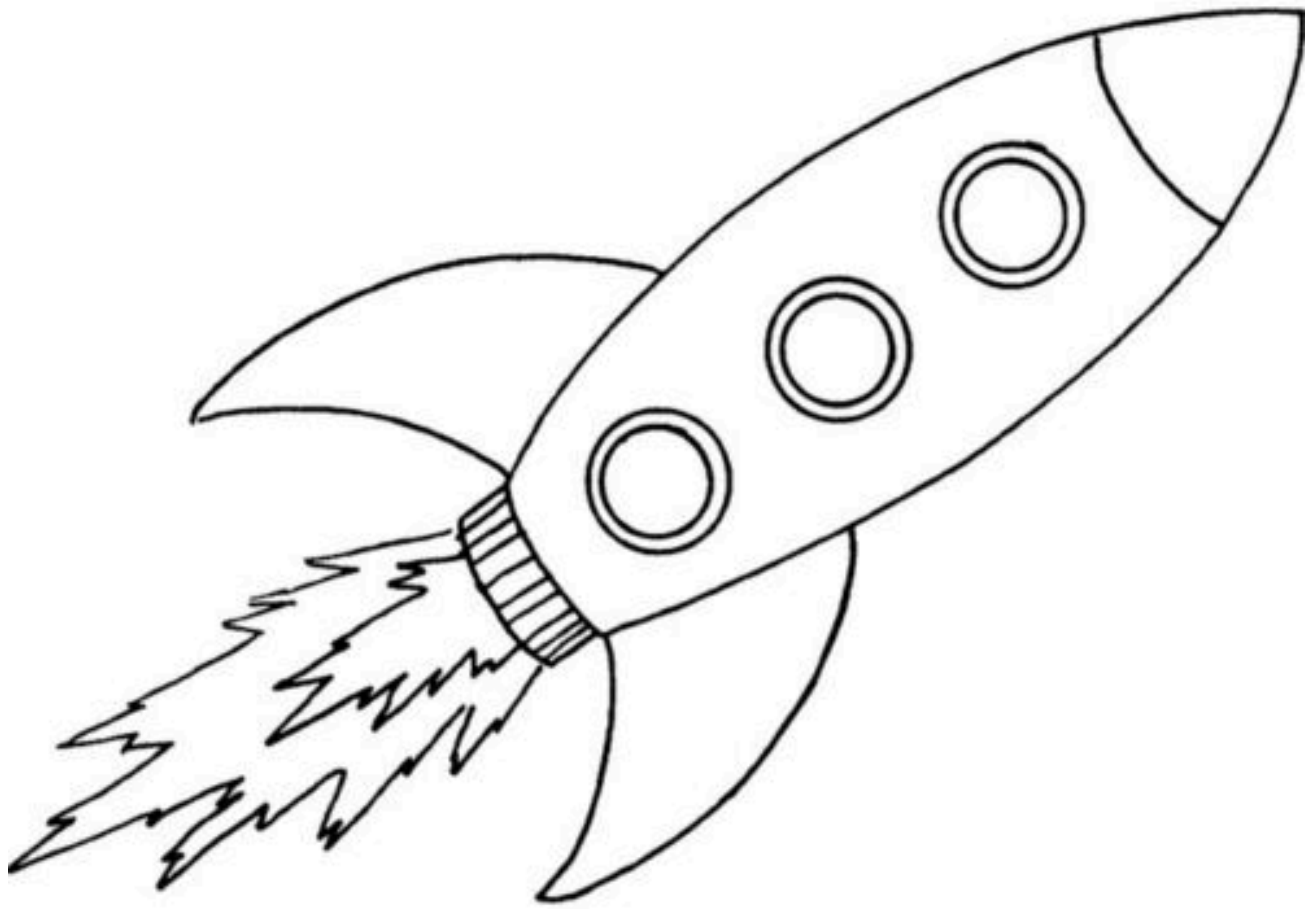
3. The cost function

It allows to measure how « far » is the neural network from the reality

4. The learning phase

The Gradient descent algorithm that allows to update the weights of the networks

From theory to practice



Splitting Data Into Training/Validation/Test Set

Splitting Data Into Training/Validation/Test Set

We have access to a dataset

Splitting Data Into Training/Validation/Test Set

We have access to a dataset

$$\mathscr{D} = \{\mathbf{X}_i, y_i\}_0^N$$

We have access to a dataset

$$\mathscr{D} = \{\mathbf{X}_i, y_i\}_0^N$$

$\xrightarrow{80\%}$

Training Set

Use for the learning algorithm

Splitting Data Into Training/Validation/Test Set

We have access to a dataset

$$\mathscr{D} = \{\mathbf{X}_i, y_i\}_0^N$$

80% | 10%

Training Set | **Validation Set**

Use for the learning algorithm | Select best hyper-parameter

Splitting Data Into Training/Validation/Test Set

We have access to a dataset

$$\mathscr{D} = \{\mathbf{X}_i, y_i\}_0^N$$

80%	10%	10%
Training Set	**Validation Set**	**Testing Set**
Use for the learning algorithm	Select best hyper-parameter	Measure the performance

Splitting Data Into Training/Validation/Test Set

We have access to a dataset

$$\mathscr{D} = \{\mathbf{X}_i, y_i\}_0^N$$

80%	10%	10%
Training Set	Validation Set	Testing Set
Use for the learning algorithm	Select best hyper-parameter	Measure the performance

Best practices:

Splitting Data Into Training/Validation/Test Set

We have access to a dataset

$$\mathscr{D} = \{\mathbf{X}_i, y_i\}_0^N$$

80%	10%	10%
Training Set	**Validation Set**	**Testing Set**
Use for the learning algorithm	**Select best hyper-parameter**	**Measure the performance**

Best practices: **1. ensure that your results are independent of the split**

Splitting Data Into Training/Validation/Test Set

We have access to a dataset

$$\mathscr{D} = \{\mathbf{X}_i, y_i\}_0^N$$

80%	10%	10%
Training Set	**Validation Set**	**Testing Set**
Use for the learning algorithm	Select best hyper-parameter	Measure the performance

Best practices:

1. ensure that your results are independent of the split

2. make sur that the proportion of the labels in the split are uniform

Splitting Data Into Training/Validation/Test Set

We have access to a dataset

$$\mathscr{D} = \{\mathbf{X}_i, y_i\}_0^N$$

80% → **Training Set**: Use for the learning algorithm

10% → **Validation Set**: Select best hyper-parameter

10% → **Testing Set**: Measure the performance

Measure performance across different seeds

Best practices:

1. ensure that your results are independent of the split

2. make sur that the proportion of the labels in the split are uniform

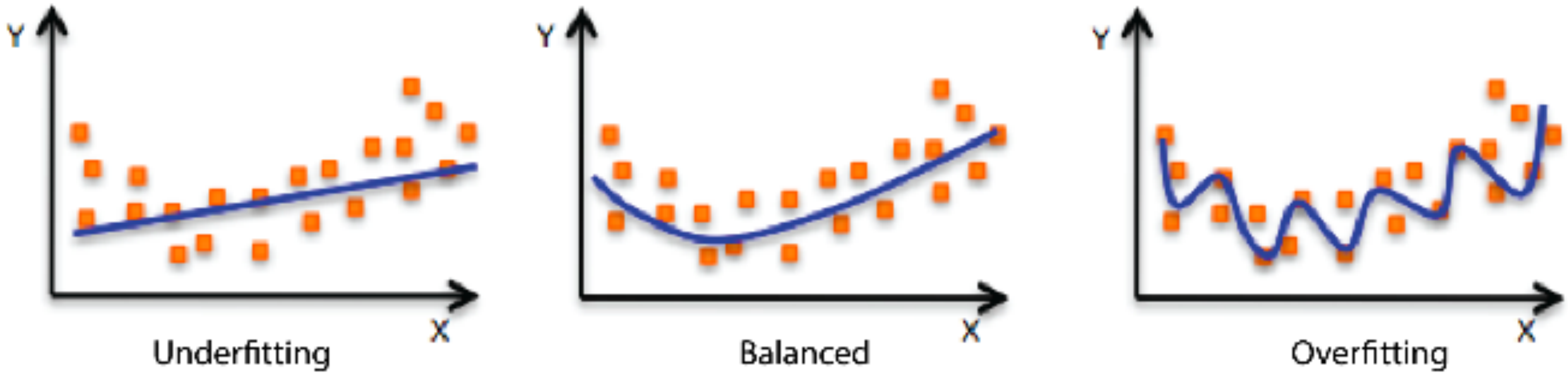
Overfitting vs Underfitting

Overfitting vs Underfitting

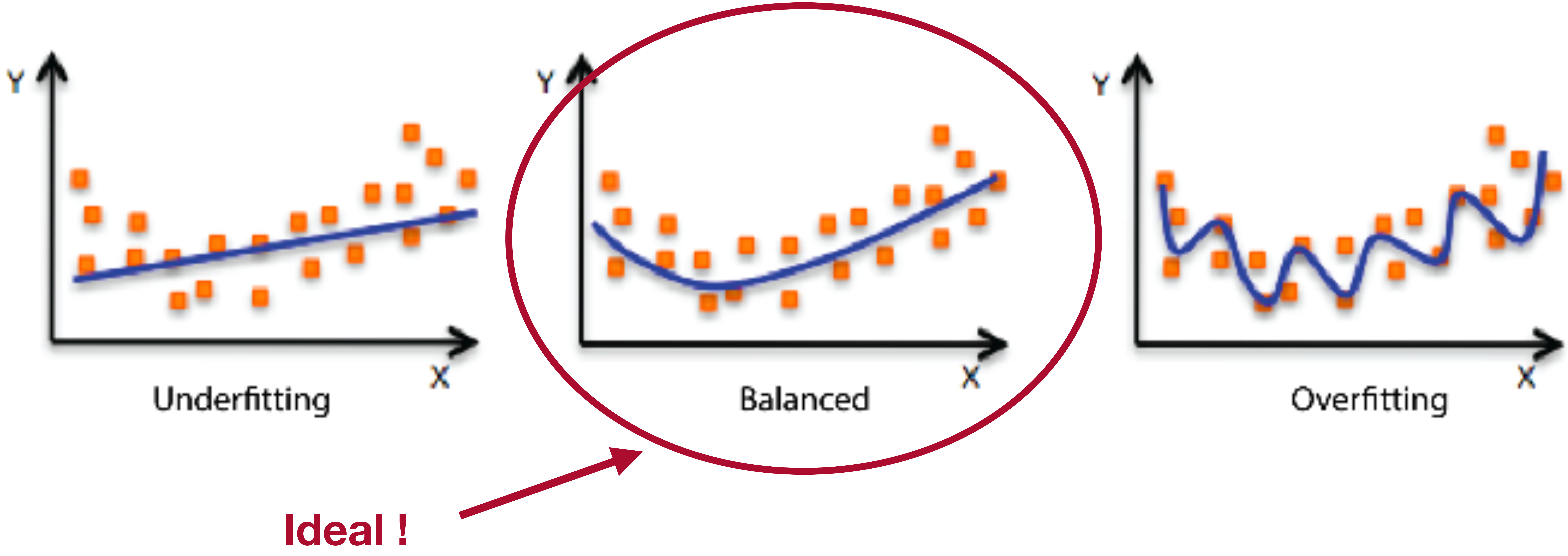
A gentle reminder from basic Machine Learning Classes

Overfitting vs Underfitting

A gentle reminder from basic Machine Learning Classes

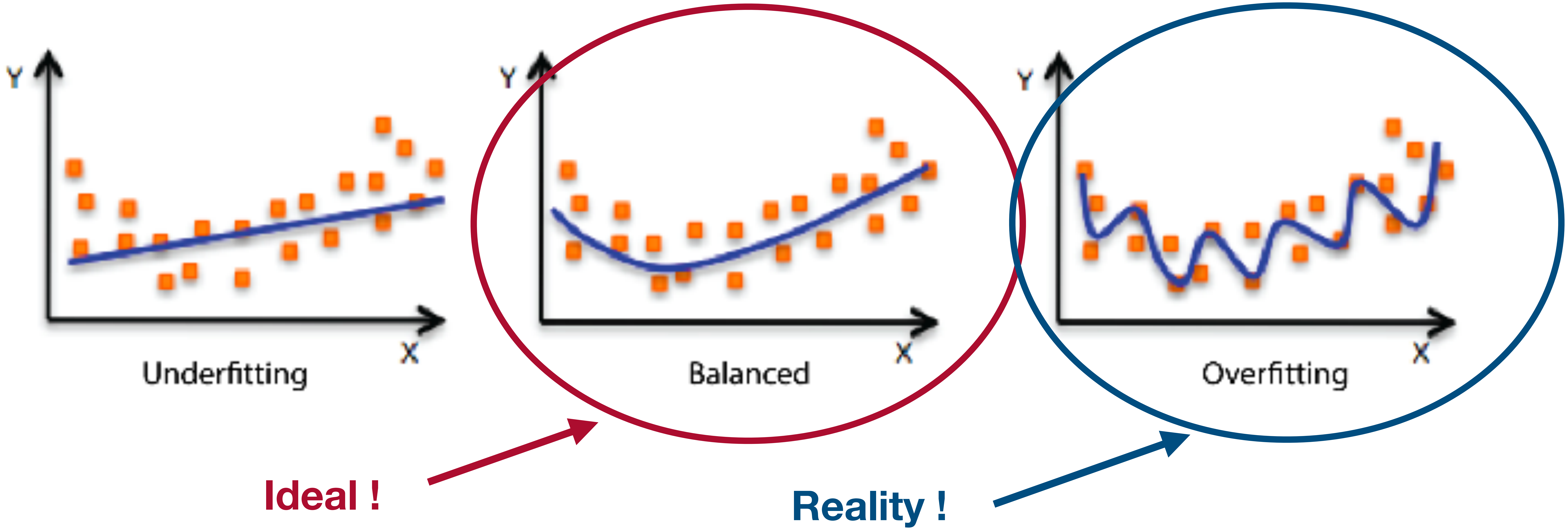


A gentle reminder from basic Machine Learning Classes



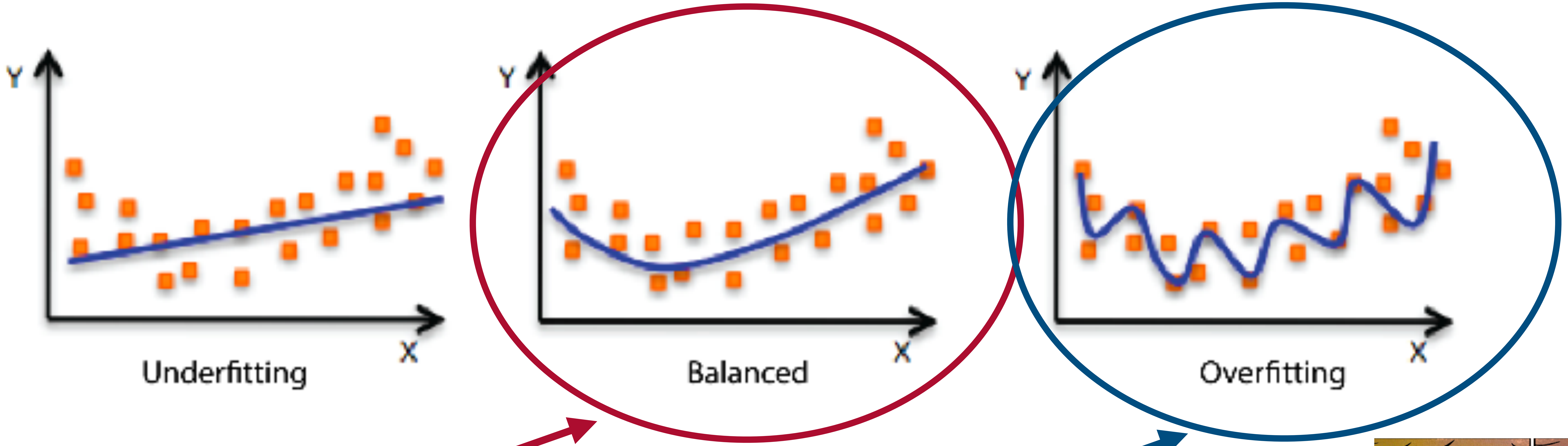
Overfitting vs Underfitting

A gentle reminder from basic Machine Learning Classes



Overfitting vs Underfitting

A gentle reminder from basic Machine Learning Classes

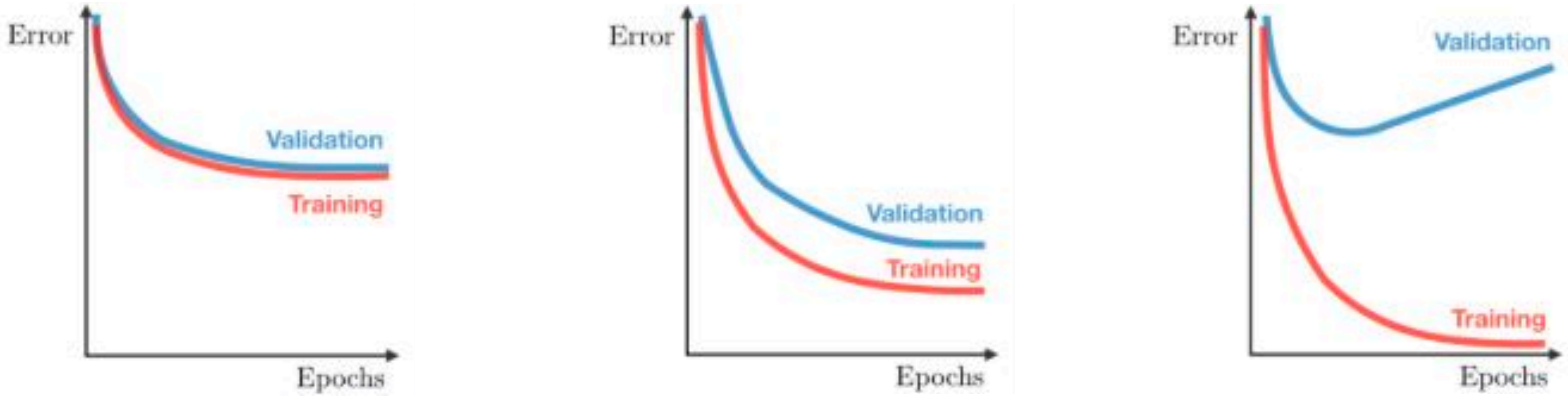


Ideal !

Reality !

Quiz: who is who ?

Which situation consist of overfitting/underfitting/ just right ?



In practice, when the validation loss starts to increase you are overfitting

When to stop training? **when the validation loss has converged or starts to increase**

Fighting Overfitting with regularization

Fighting Overfitting with regularization

Fighting Overfitting with regularization

Regularization: the set of technics to prevent or fight overfitting

Fighting Overfitting with regularization

Regularization: the set of technics to prevent or fight overfitting

It improves the generalization capability of your model (i.e. better results)

Regularization: the set of technics to prevent or fight overfitting

It improves the generalization capability of your model (i.e. better results)

Technique 1: Dropout

Fighting Overfitting with regularization

Regularization: the set of technics to prevent or fight overfitting

It improves the generalization capability of your model (i.e. better results)

Technique 1: Dropout

During training randomly set some activation to 0

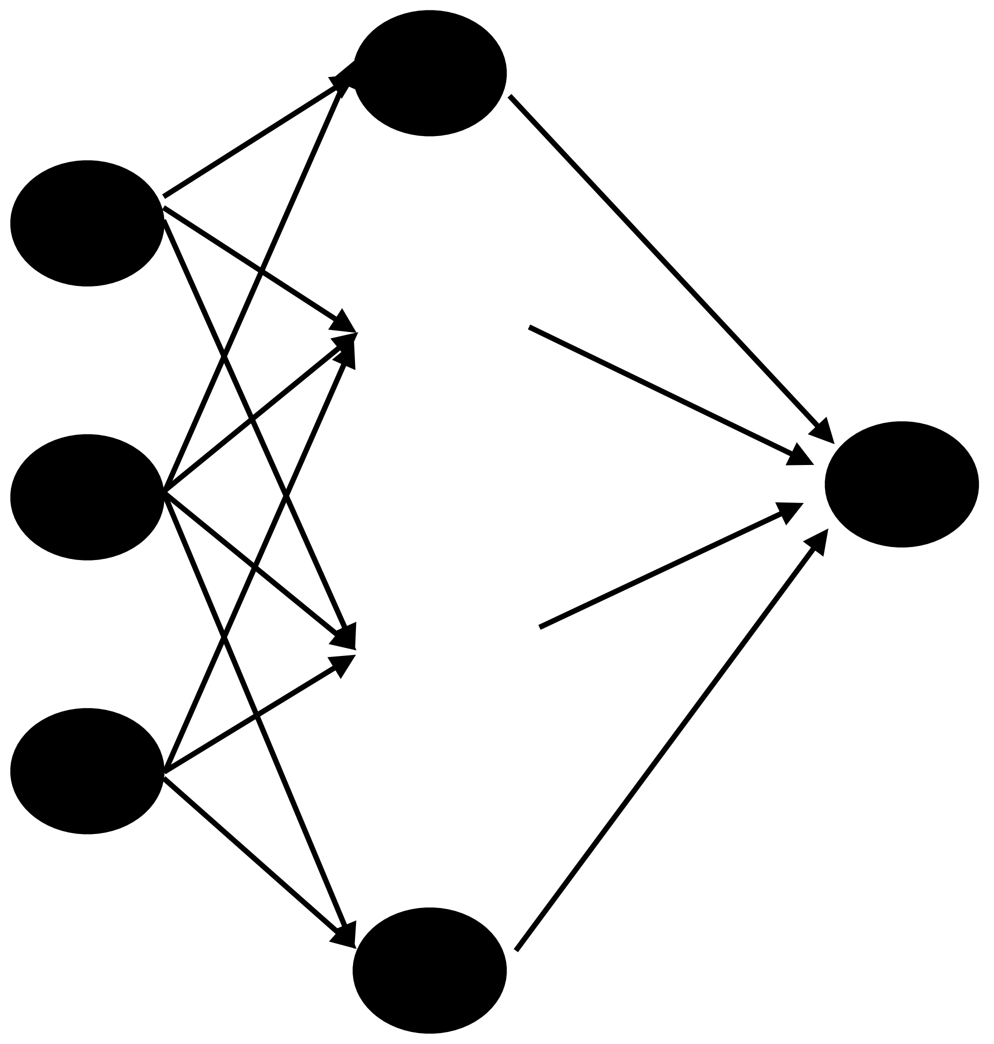
Regularization: the set of technics to prevent or fight overfitting

It improves the generalization capability of your model (i.e. better results)

Technique 1: Dropout

During training randomly set some activation to 0

Iter 1



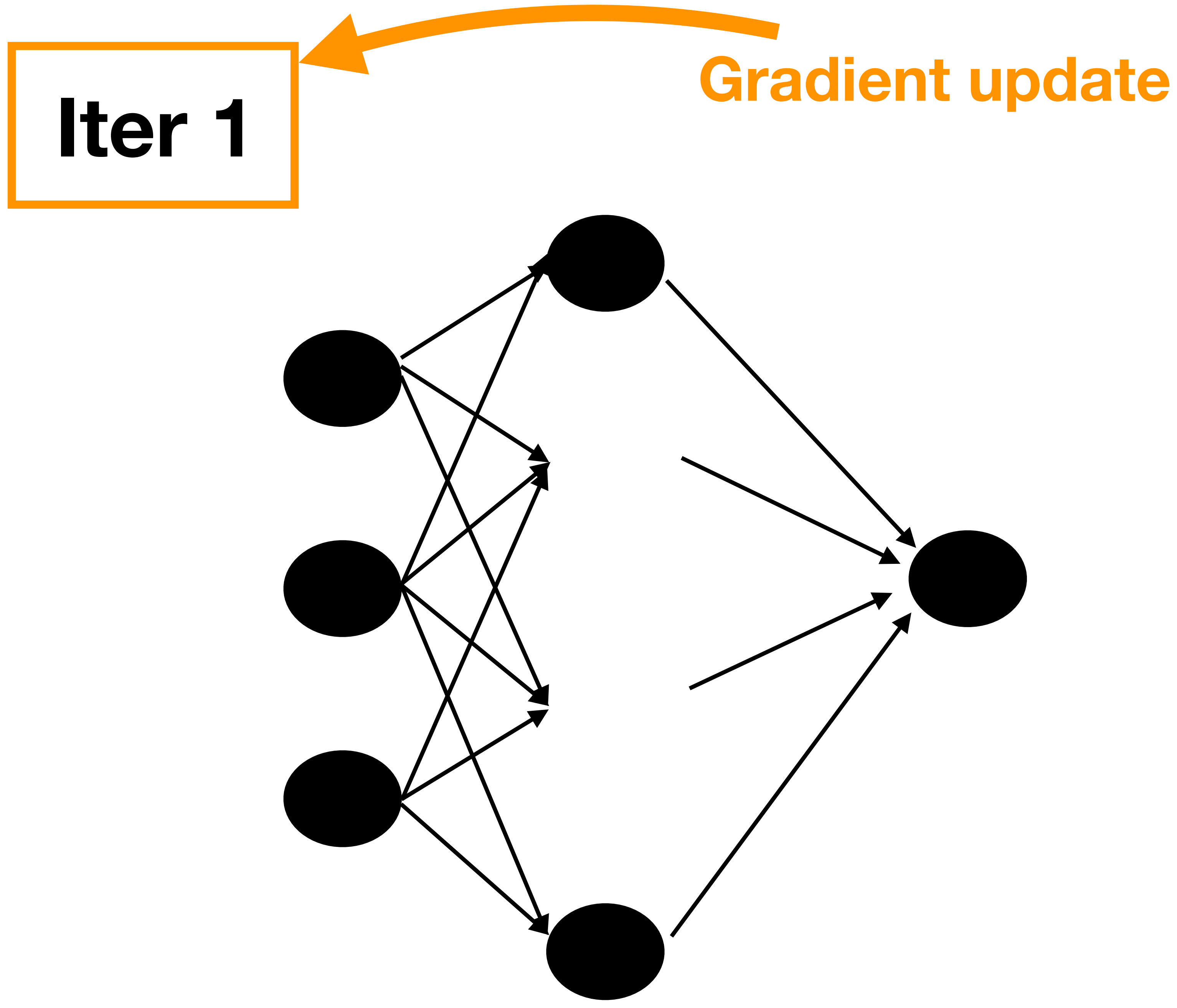
Fighting Overfitting with regularization

Regularization: the set of technics to prevent or fight overfitting

It improves the generalization capability of your model (i.e. better results)

Technique 1: Dropout

During training randomly set some activation to 0

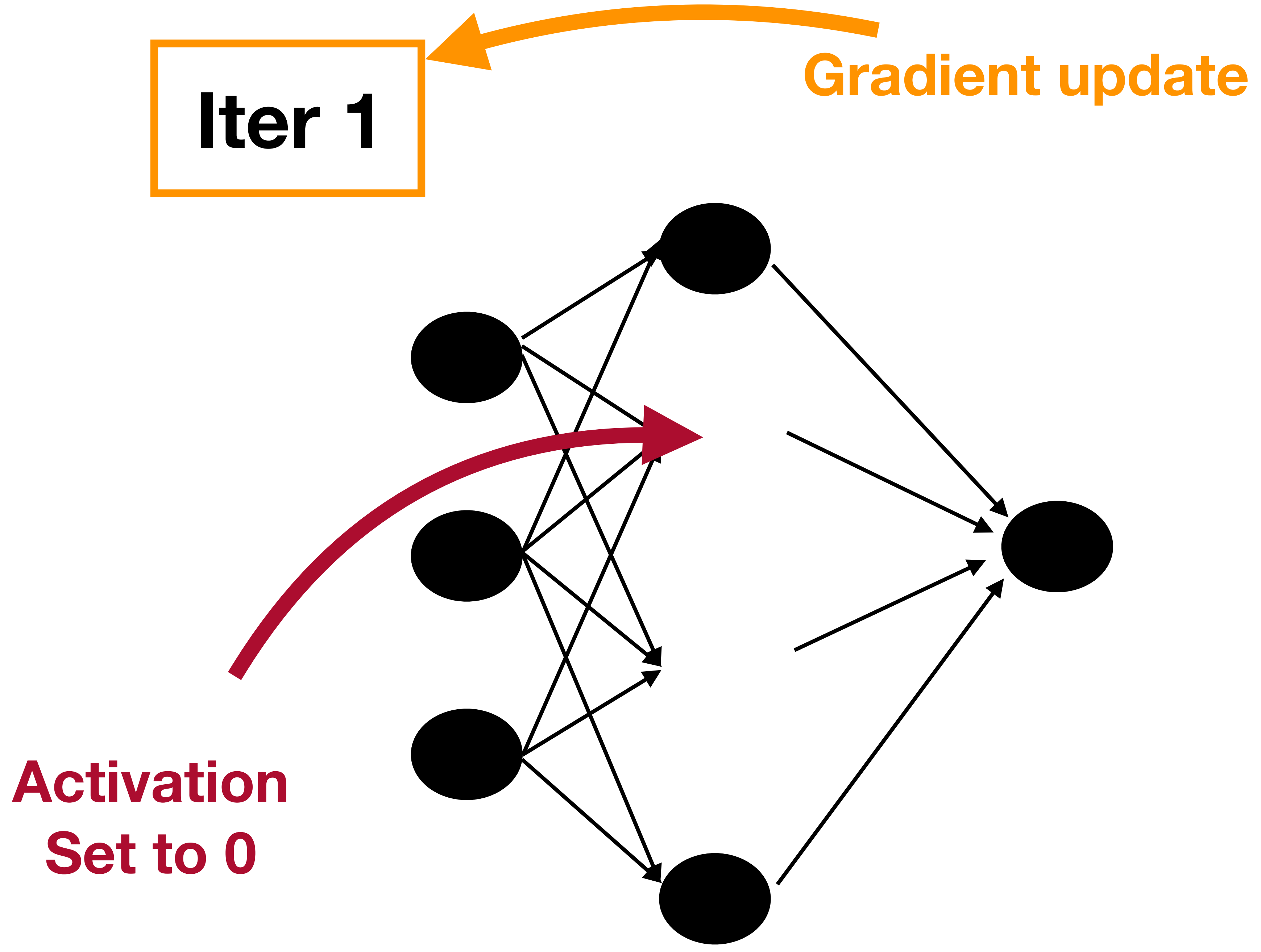


Regularization: the set of technics to prevent or fight overfitting

It improves the generalization capability of your model (i.e. better results)

Technique 1: Dropout

During training randomly set some activation to 0



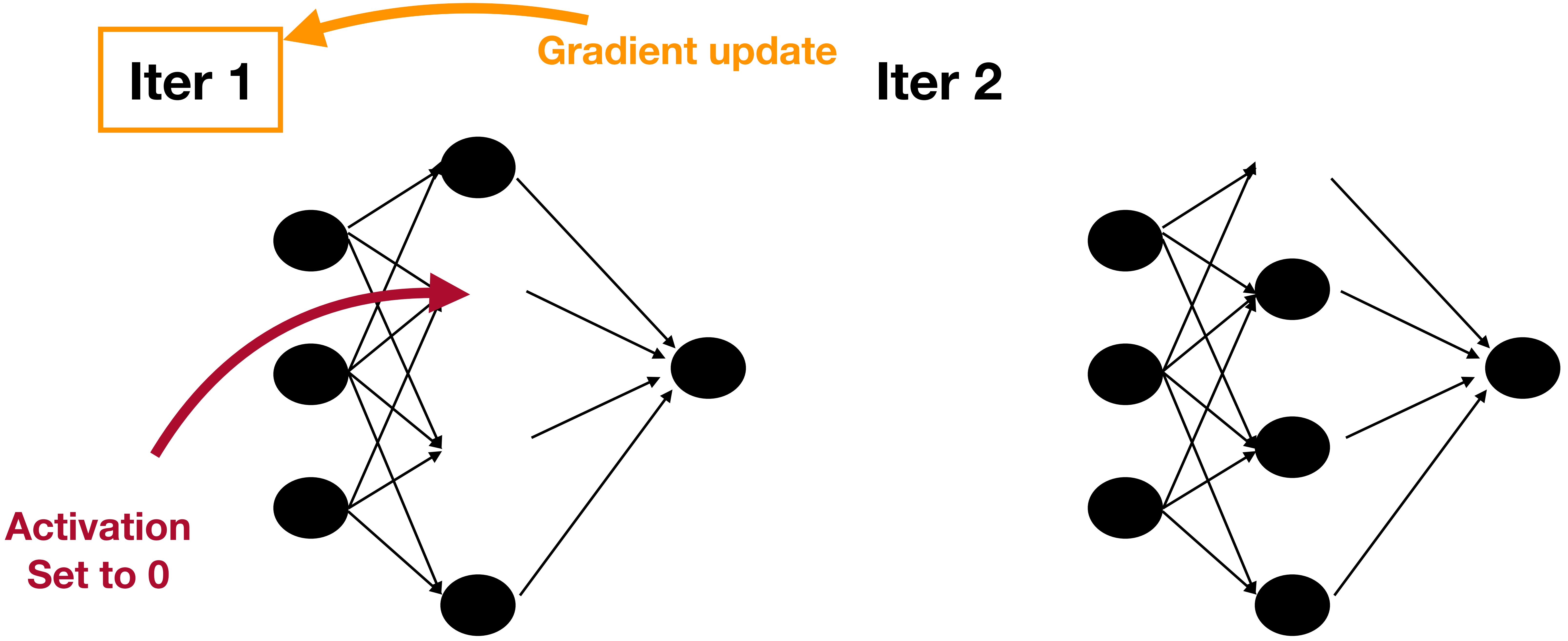
Fighting Overfitting with regularization

Regularization: the set of technics to prevent or fight overfitting

It improves the generalization capability of your model (i.e. better results)

Technique 1: Dropout

During training randomly set some activation to 0



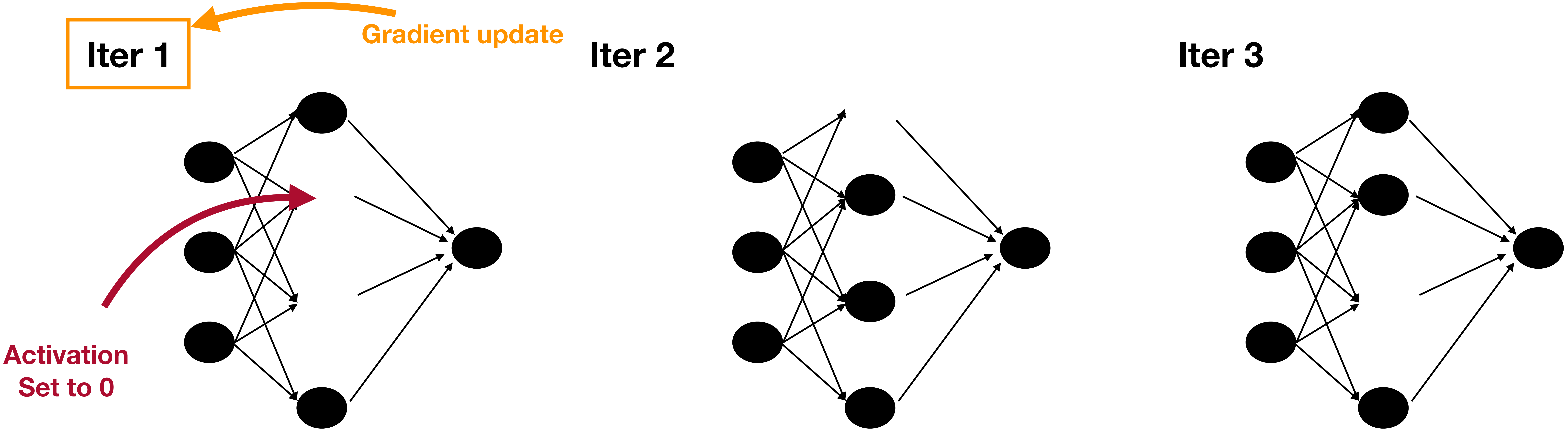
Fighting Overfitting with regularization

Regularization: the set of technics to prevent or fight overfitting

It improves the generalization capability of your model (i.e. better results)

Technique 1: Dropout

During training randomly set some activation to 0



Fighting Overfitting with regularization

Regularization: the set of technics to prevent or fight overfitting

It improves the generalization capability of your model (i.e. better results)

Regularization: the set of technics to prevent or fight overfitting

It improves the generalization capability of your model (i.e. better results)

Technique 2: Early Stopping

Fighting Overfitting with regularization

Regularization: the set of technics to prevent or fight overfitting

It improves the generalization capability of your model (i.e. better results)

Technique 2: Early Stopping

During training randomly set some activation to 0

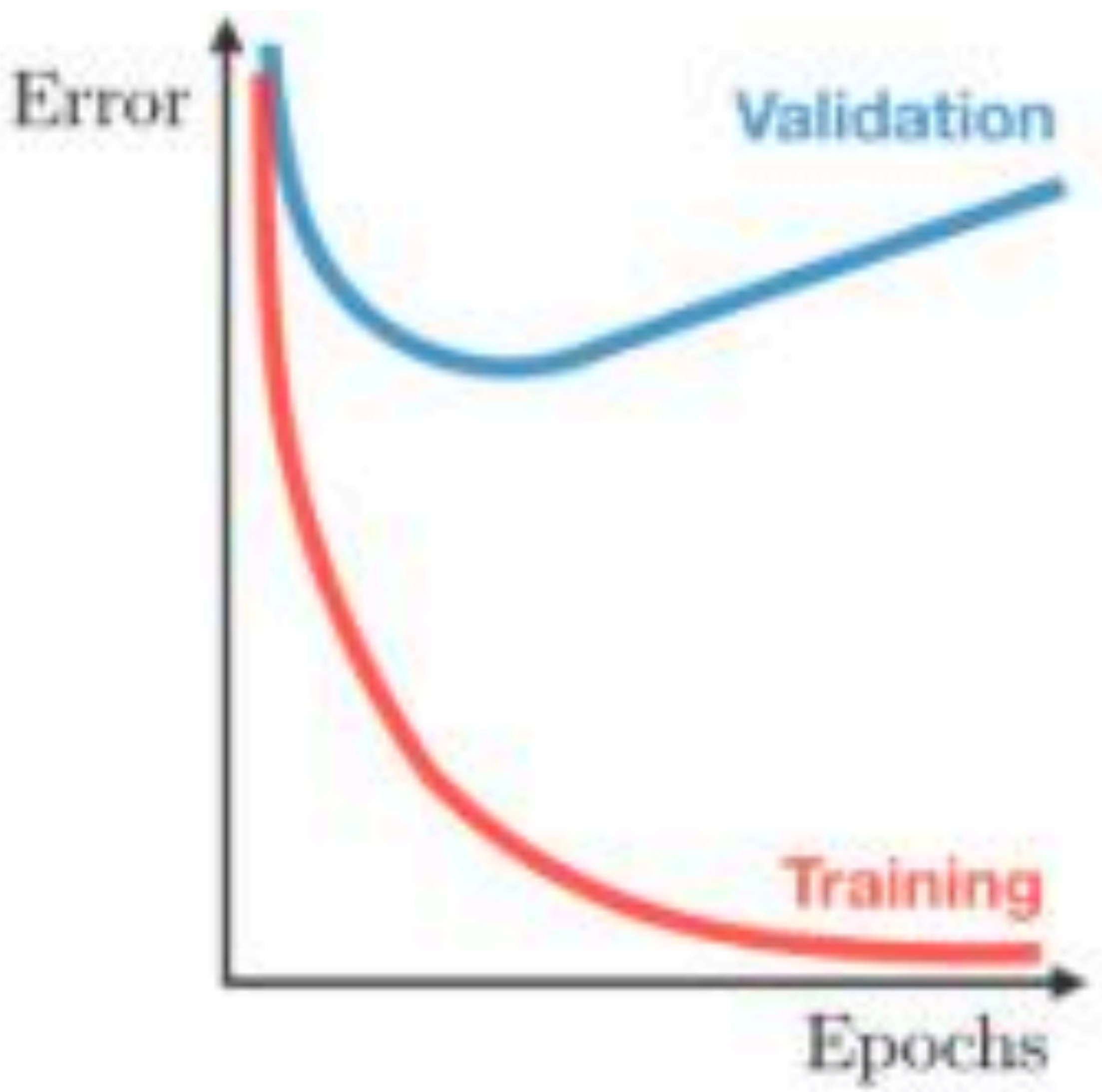
Fighting Overfitting with regularization

Regularization: the set of technics to prevent or fight overfitting

It improves the generalization capability of your model (i.e. better results)

Technique 2: Early Stopping

During training randomly set some activation to 0



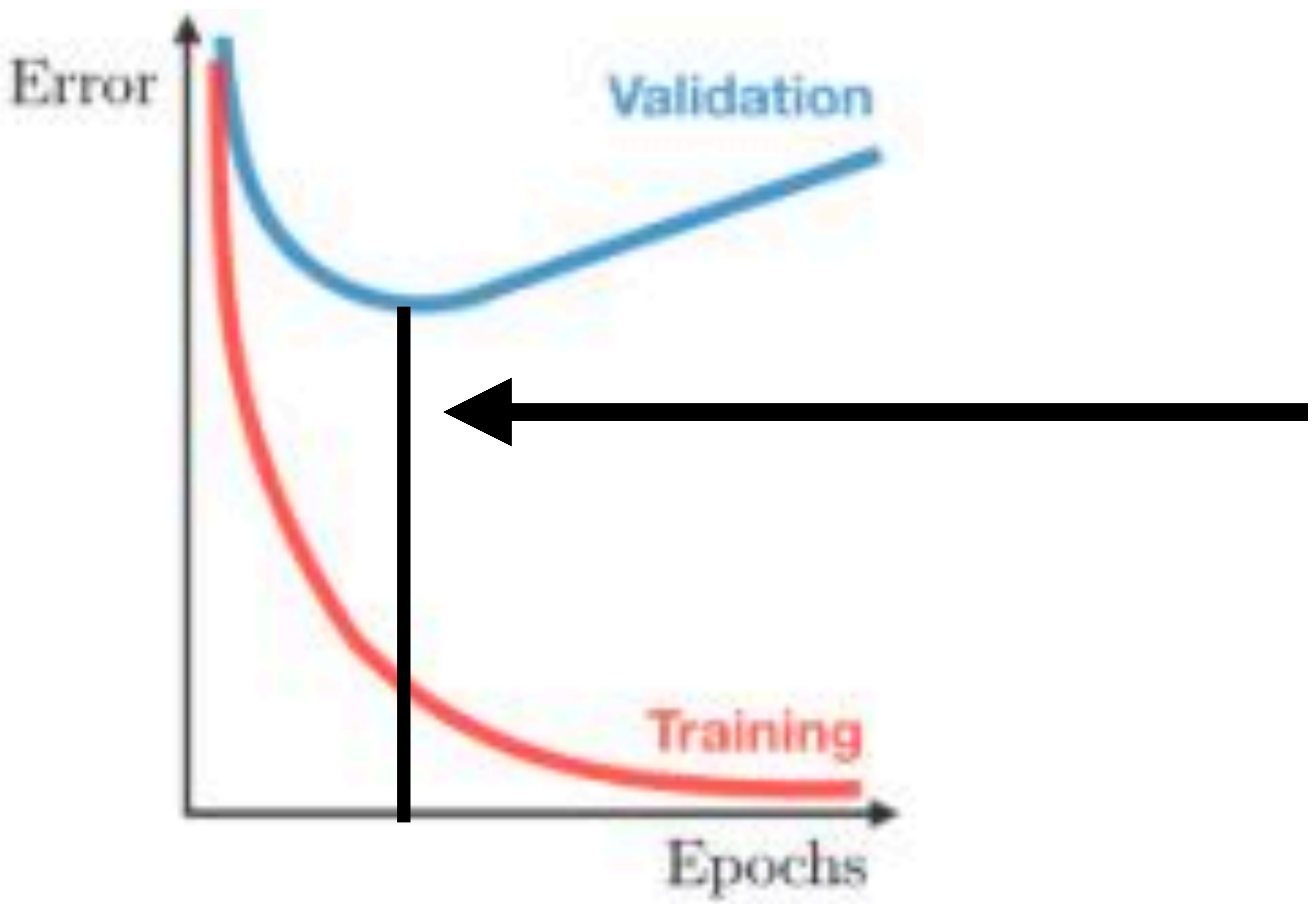
Fighting Overfitting with regularization

Regularization: the set of technics to prevent or fight overfitting

It improves the generalization capability of your model (i.e. better results)

Technique 2: Early Stopping

During training randomly set some activation to 0



Select this final model

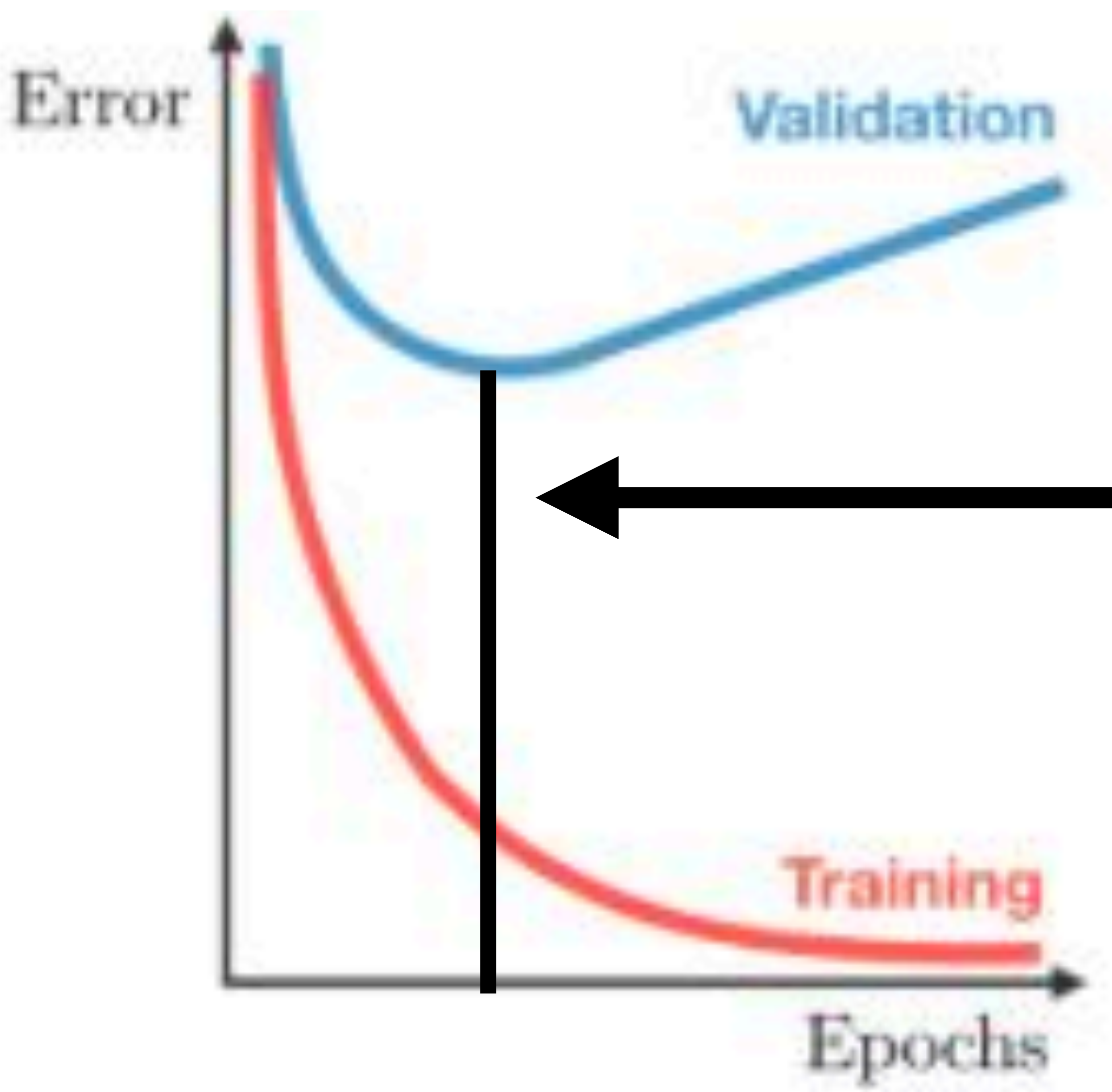
Fighting Overfitting with regularization

Regularization: the set of technics to prevent or fight overfitting

It improves the generalization capability of your model (i.e. better results)

Technique 2: Early Stopping

During training randomly set some activation to 0



Select this final model

This is the reason you need a validation set!

Let's sum up!

Procedure to use deep learning

Procedure to use deep learning

1. Get your **data clean**

Procedure to use deep learning

1. Get your **data clean**

2. **Split** your data into train/val/test split

Procedure to use deep learning

1. Get your data clean

2. Split your data into train/val/test split

3. Define your architecture/loss function/optimization algorithm

4. Train and tune your algorithm by monitoring training/validation loss

Procedure to use deep learning

1. Get your **data clean**

2. **Split** your data into train/val/test split

3. Define your **architecture/loss function/optimization** algorithm

4. Train and tune your algorithm by monitoring training/validation loss

5. Test your model on test set

Procedure to use deep learning

1. Get your data clean

2. Split your data into train/val/test split

3. Define your architecture/loss function/optimization algorithm

4. Train and tune your algorithm by monitoring training/validation loss

5. Test your model on test set

6. Deploy your model in production

Procedure to use deep learning

1. Get your data clean

2. Split your data into train/val/test split

Training Loop

3. Define your architecture/loss function/optimization algorithm

4. Train and tune your algorithm by monitoring training/validation loss

5. Test your model on test set

6. Deploy your model in production

Procedure to use deep learning

1. Get your data clean

80% of the job is done here!

2. Split your data into train/val/test split

Training Loop

3. Define your architecture/loss function/optimization algorithm

4. Train and tune your algorithm by monitoring training/validation loss

5. Test your model on test set

6. Deploy your model in production

Putting Everything Together: Training

Data **Deep Learning Architecture**

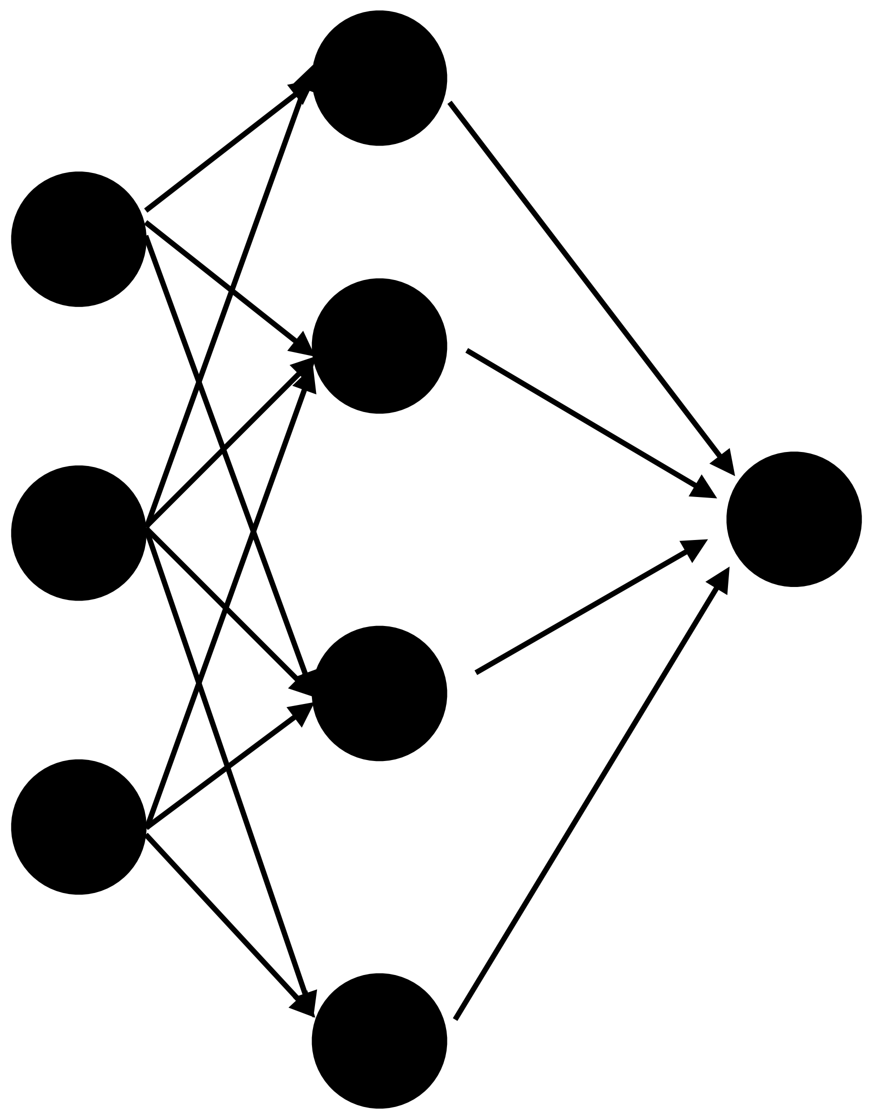
Putting Everything Together: Training

Data

Deep Learning Architecture

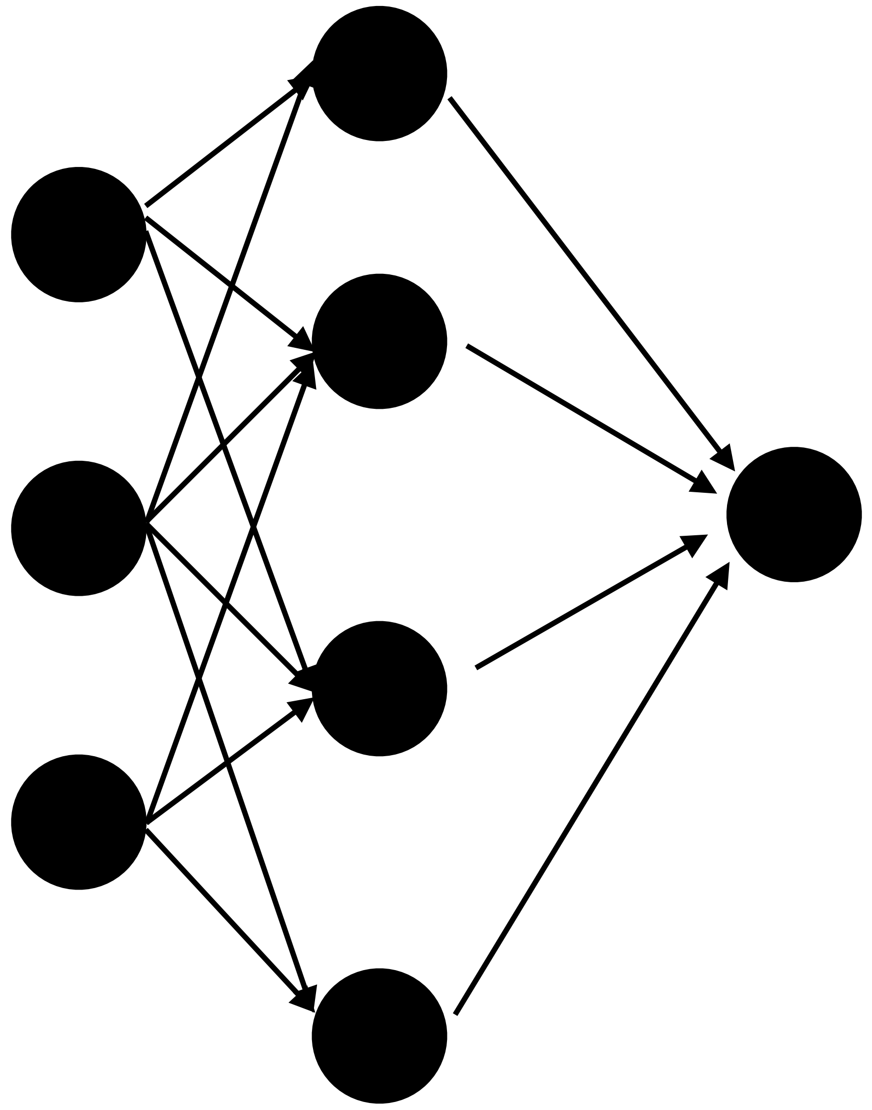
Data

Deep Learning Architecture



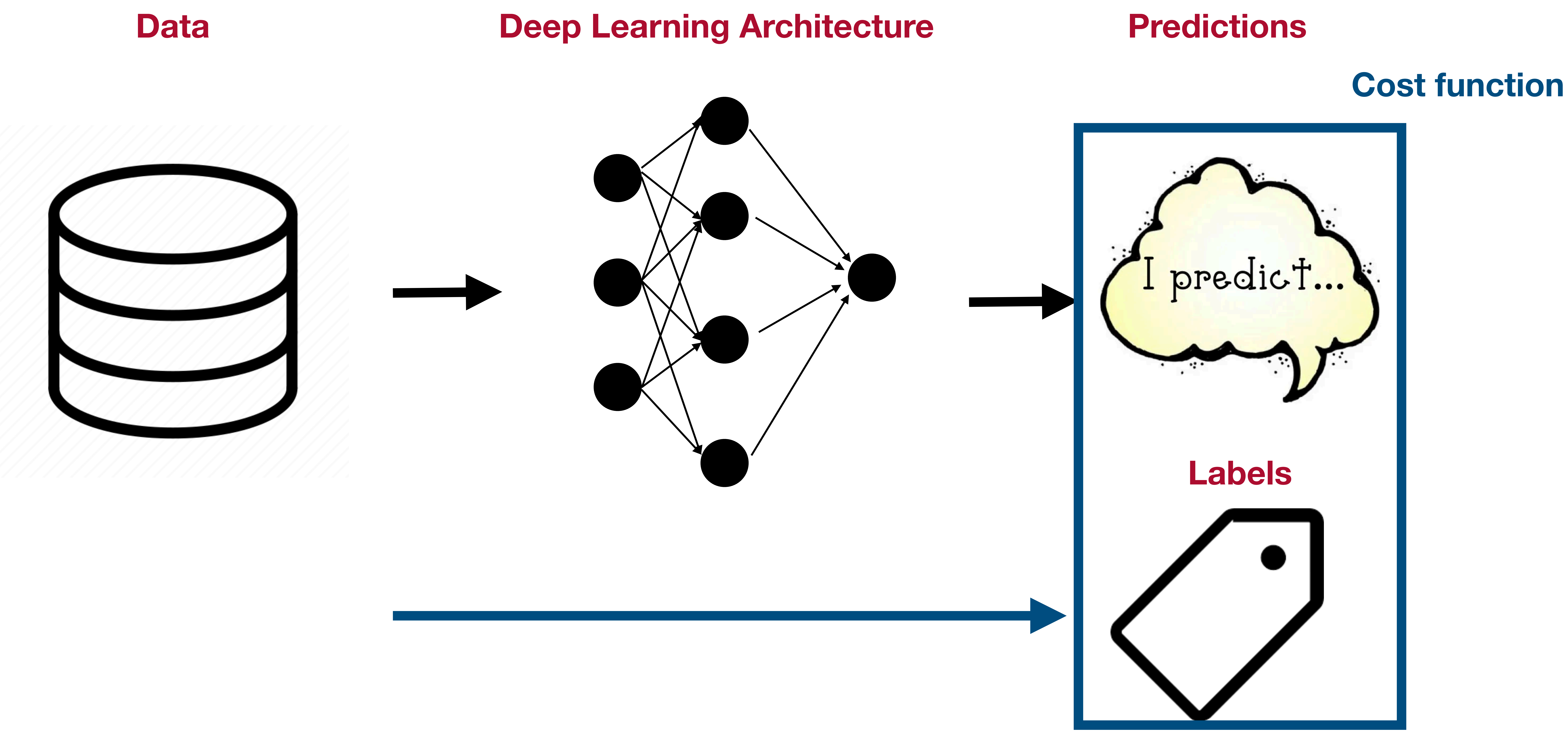
Data

Deep Learning Architecture



Predictions

Putting Everything Together: Training



Putting Everything Together: Training

Putting Everything Together: Training

Putting Everything Together: Inference (deployment mode)

Data **Deep Learning Architecture**

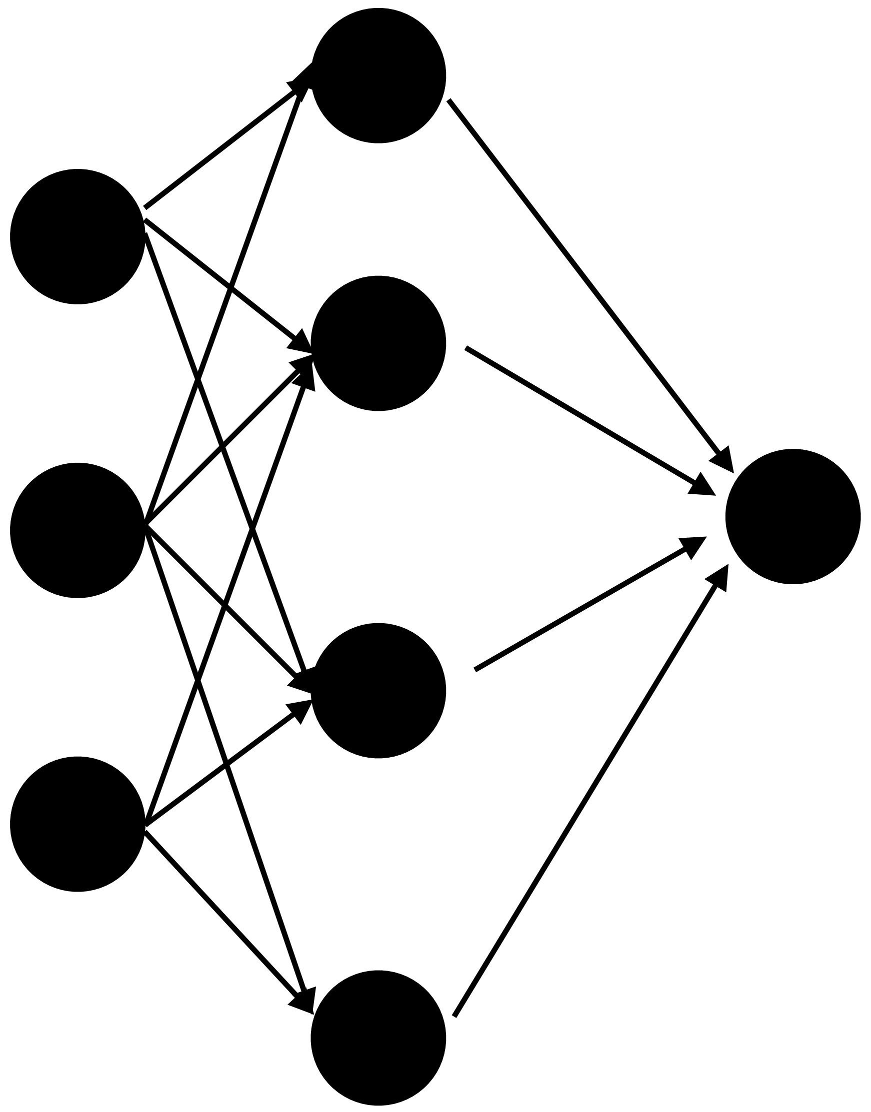
Data

Deep Learning Architecture

Putting Everything Together: Inference (deployment mode)

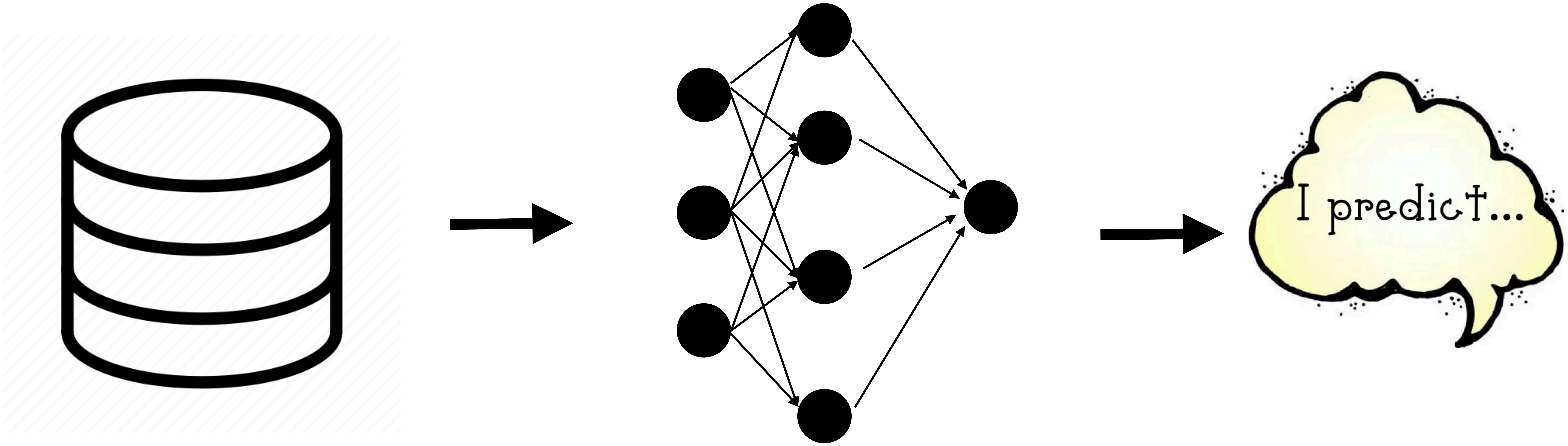
Data

Deep Learning Architecture



Putting Everything Together: Inference (deployment mode)

Data **Deep Learning Architecture** **Predictions**

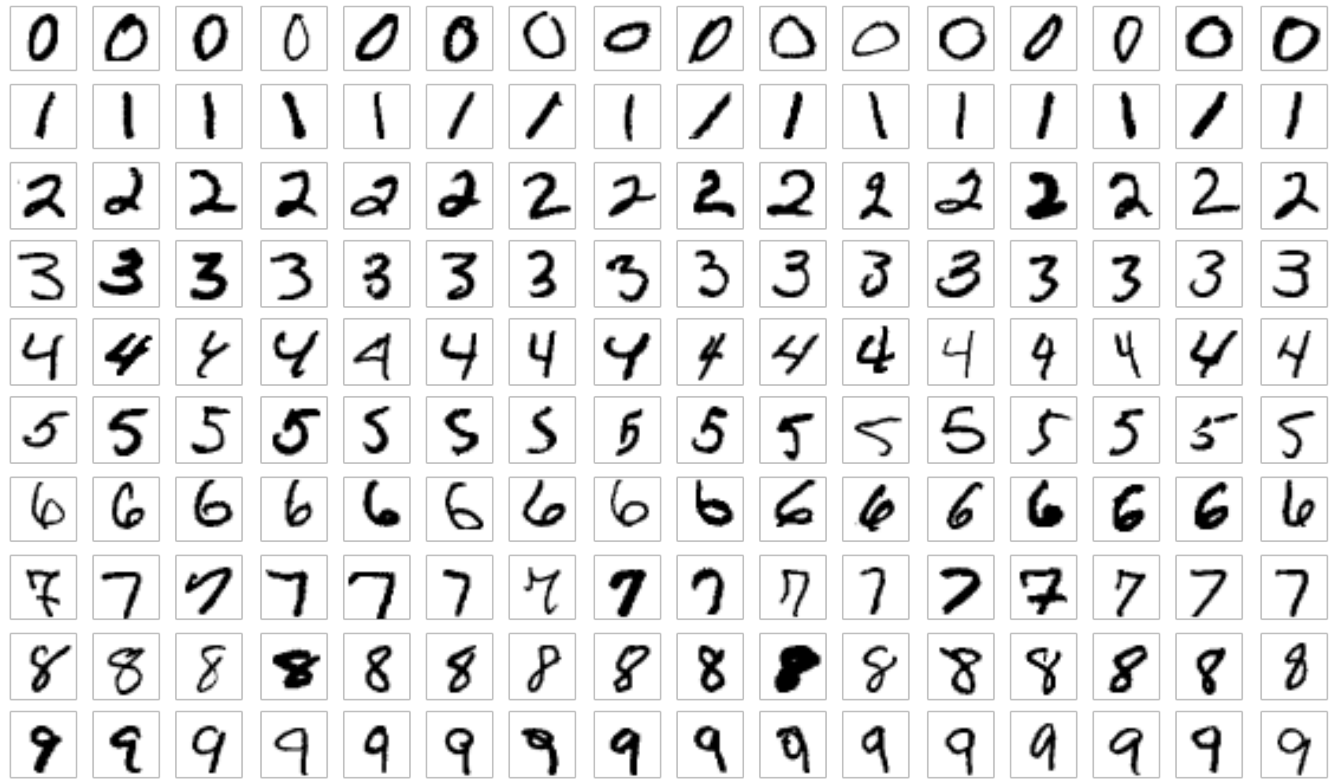


Let’s code now

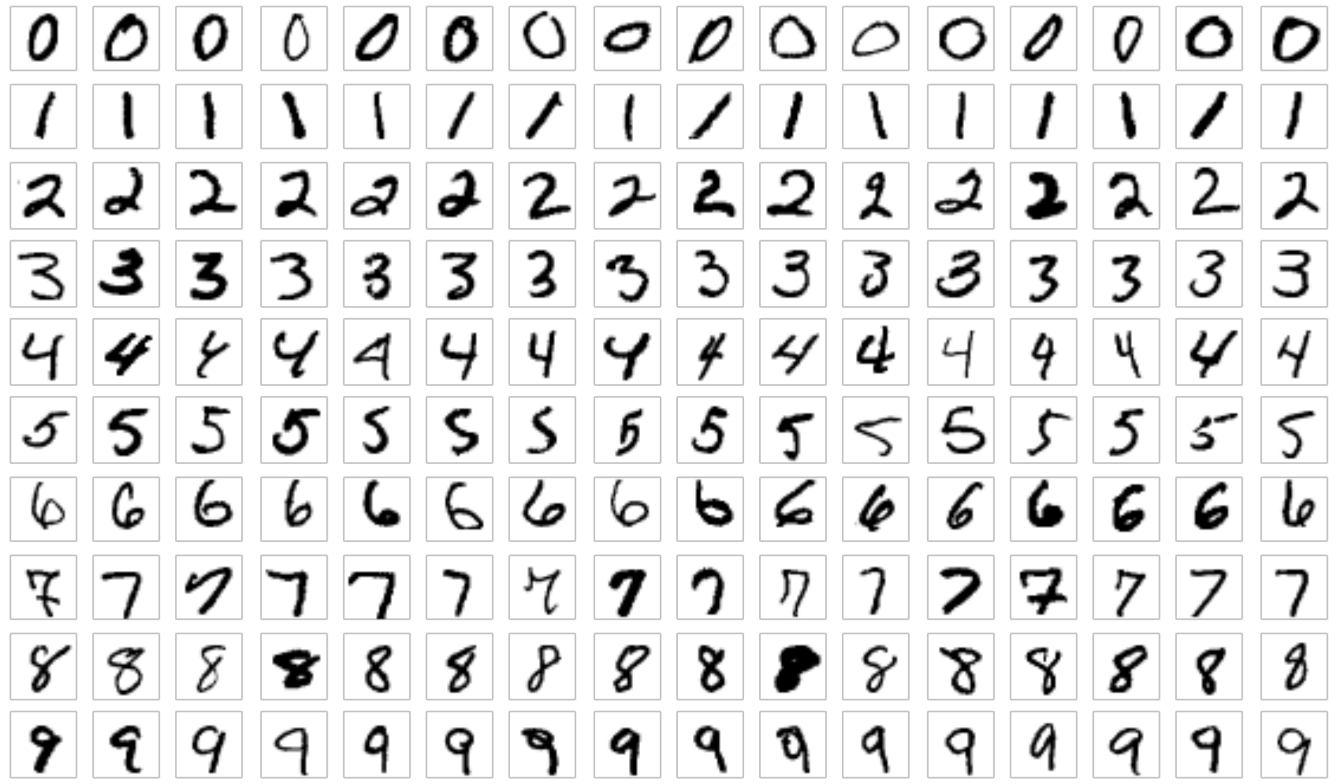
What will we code?

What will we code?

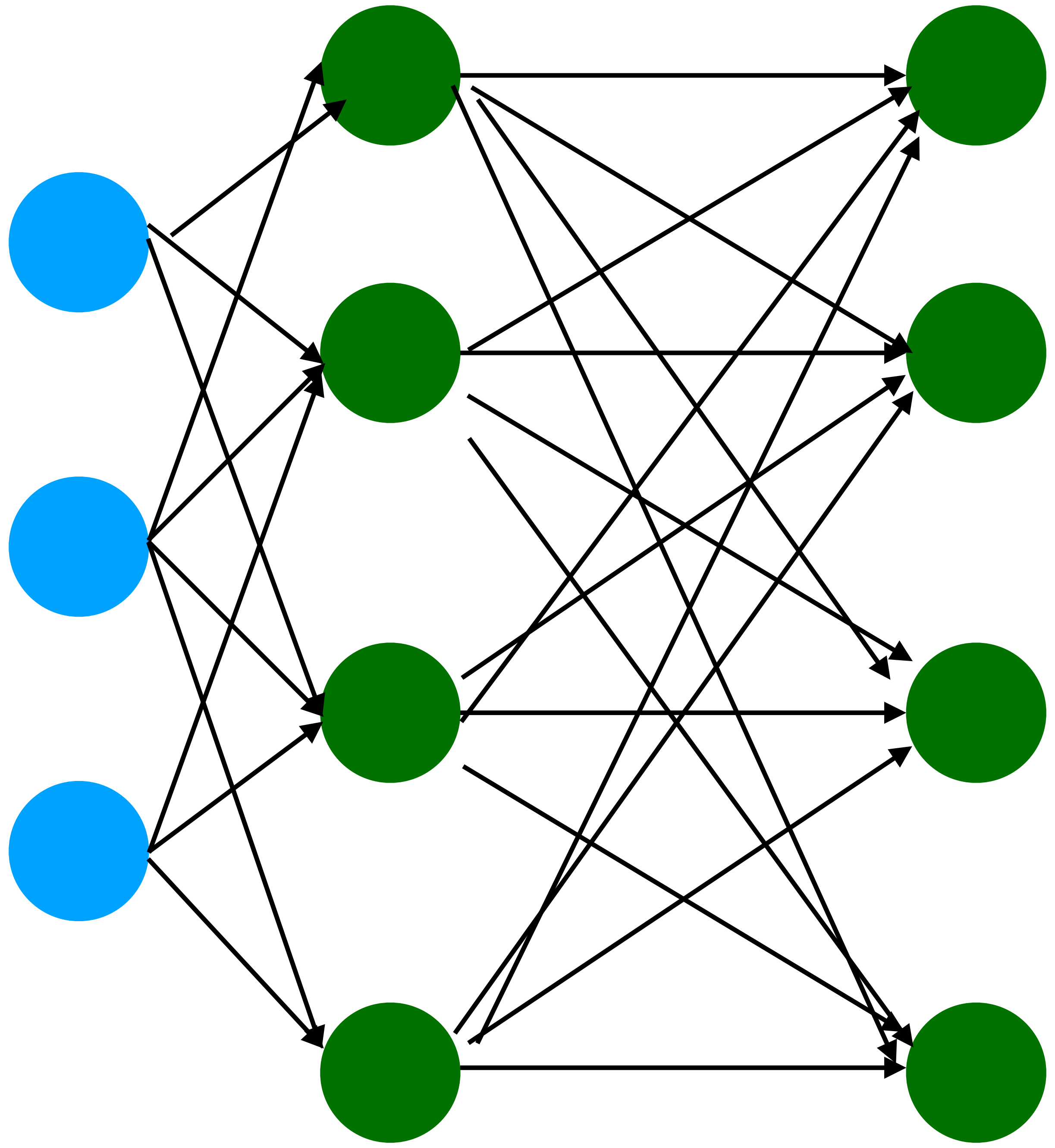
MNIST



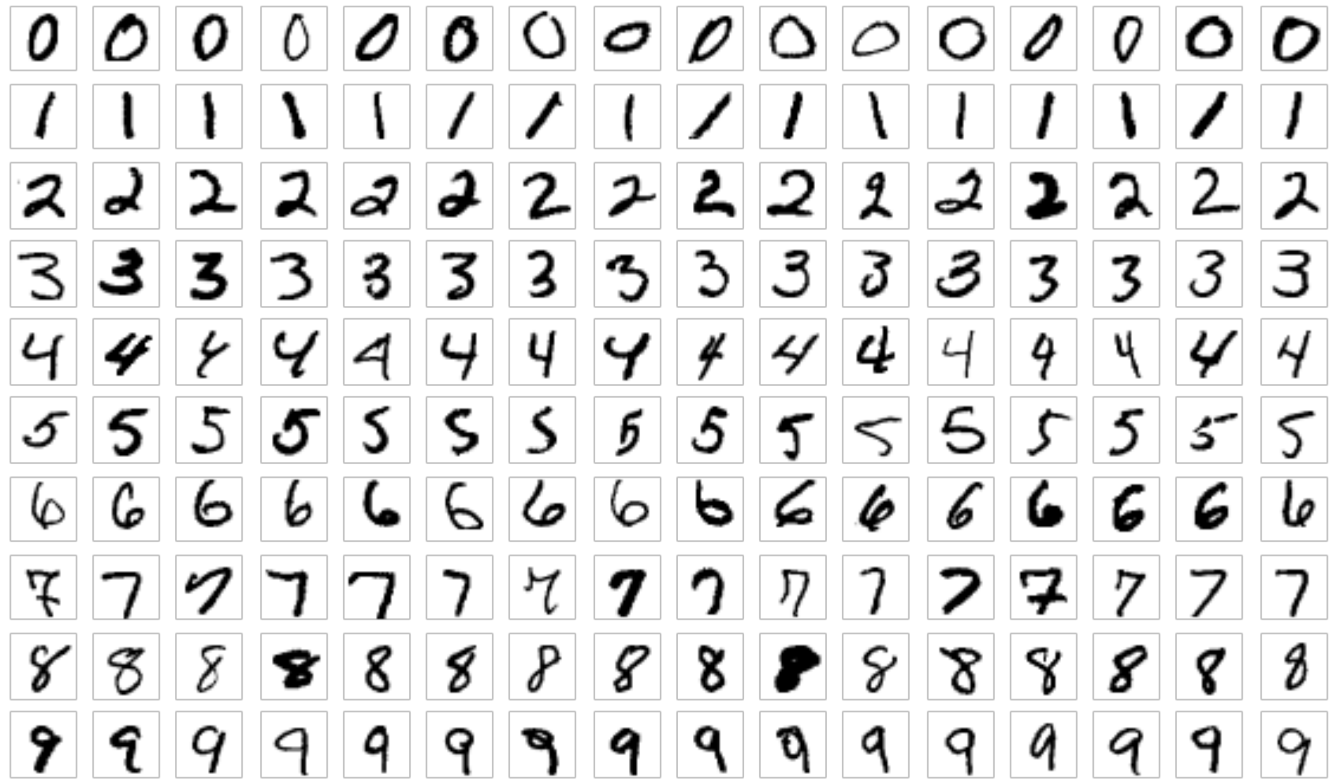
MNIST



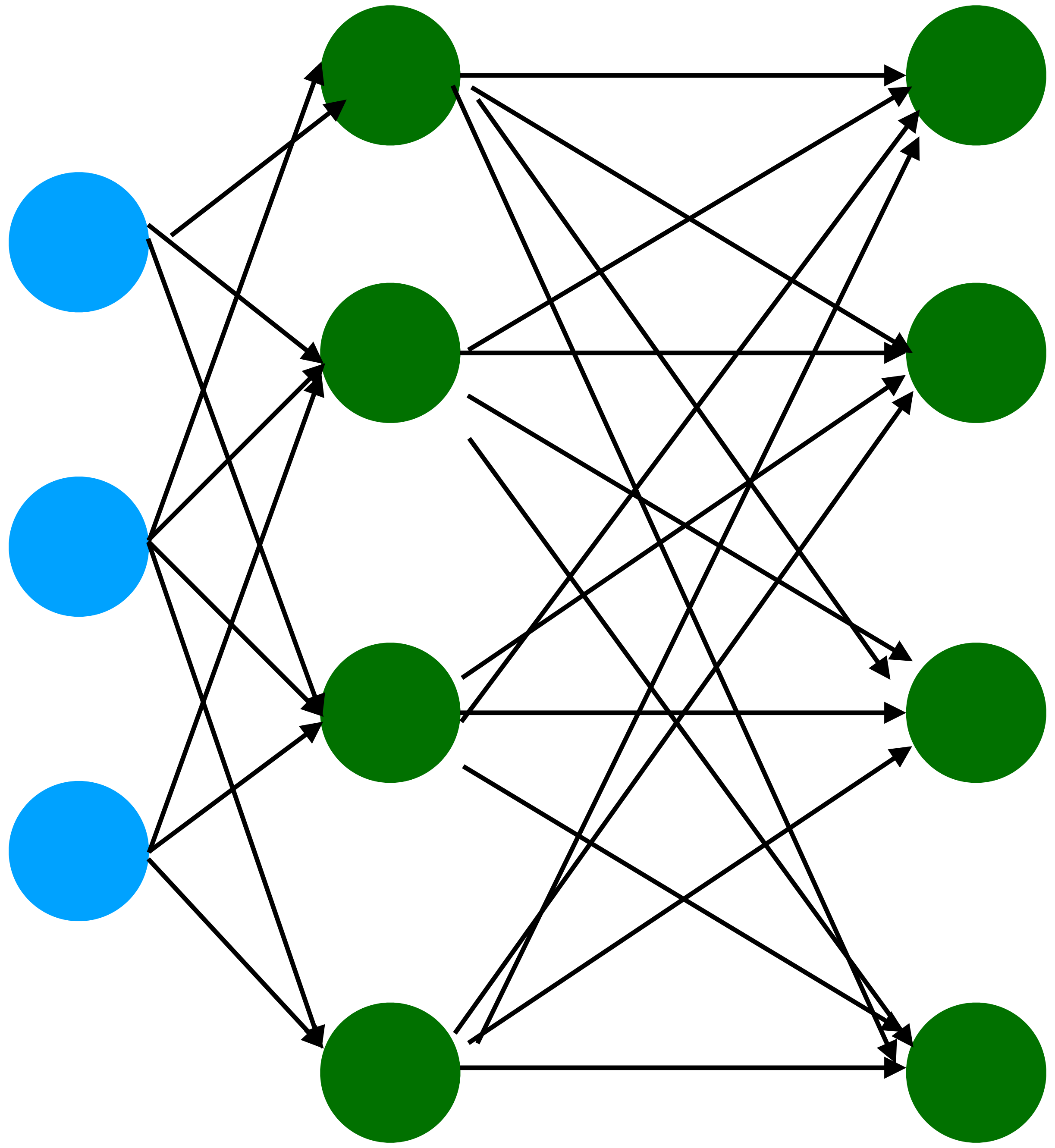
Neural Network 1 - hidden layer



MNIST

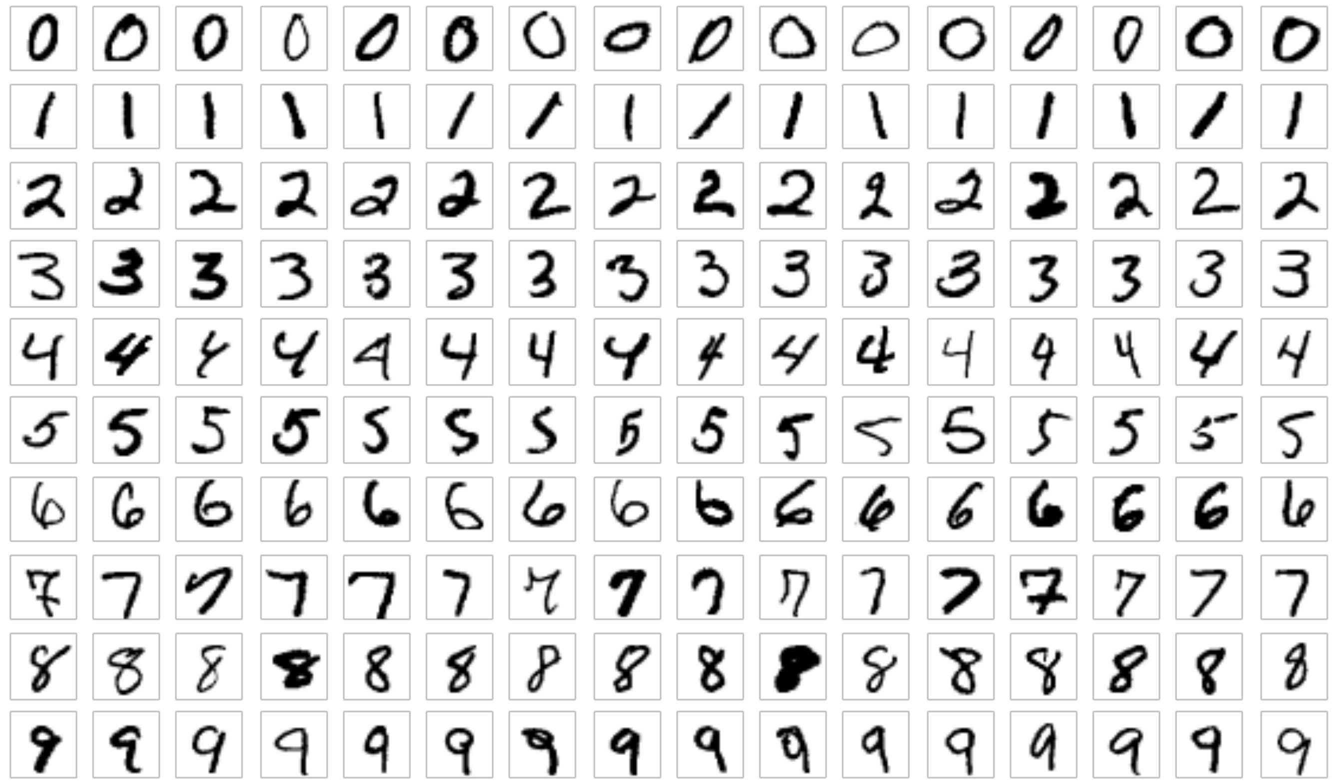


Neural Network 1 - hidden layer

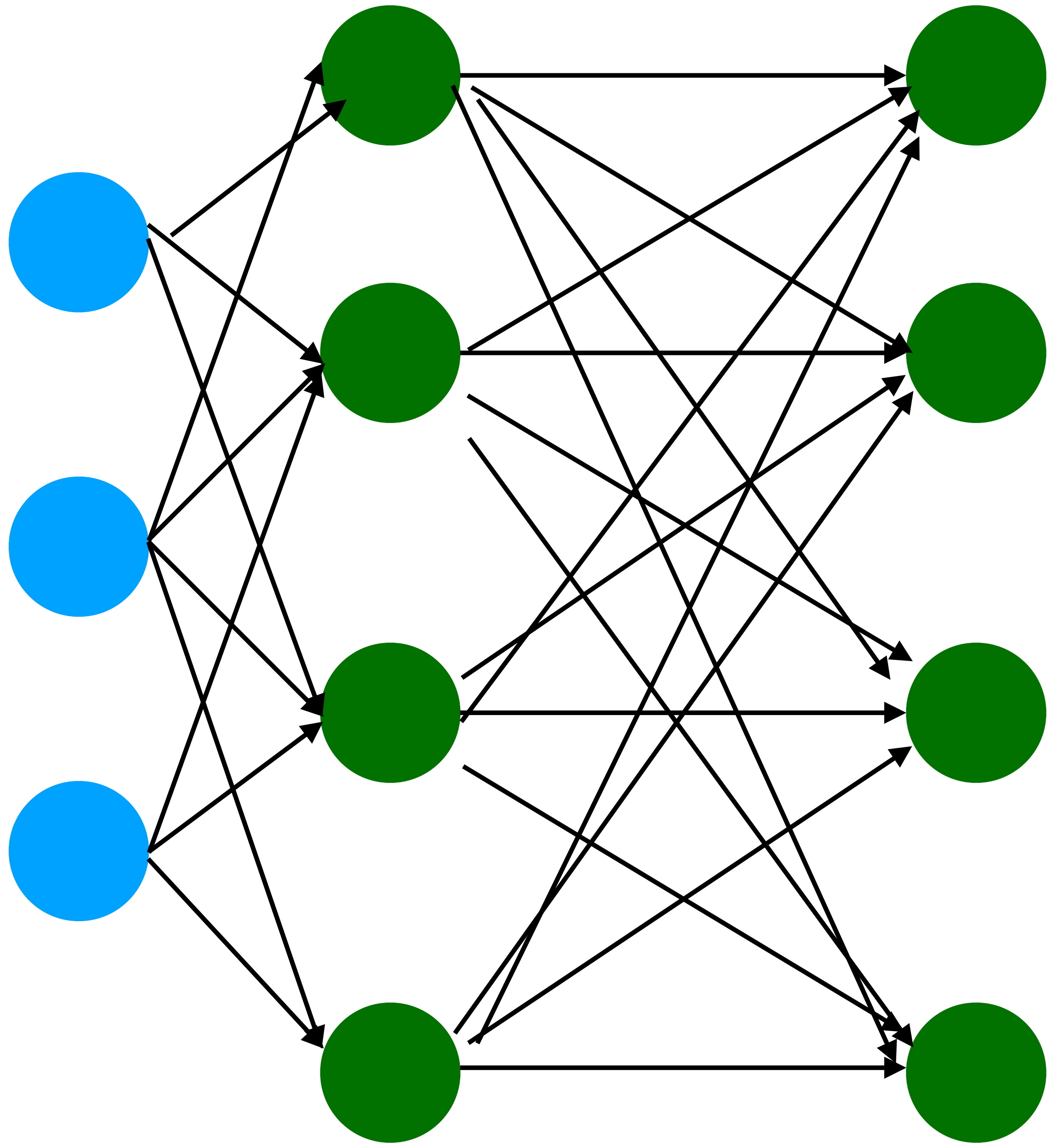


Automatically classify handwritten digits

MNIST



Neural Network 1 - hidden layer



Automatically classify handwritten digits

I want that you code, train and evaluate your first neural network !