



Do you recognise the music  
playing?

# 6COSC020W: APPLIED AI

## WEEK 8: (ARTIFICIAL) NEURAL NETWORKS

DR. ESTER BONMATI

([e.bonmaticoll@westminster.ac.uk](mailto:e.bonmaticoll@westminster.ac.uk))

# LEARNING OUTCOMES OF THIS SESSION

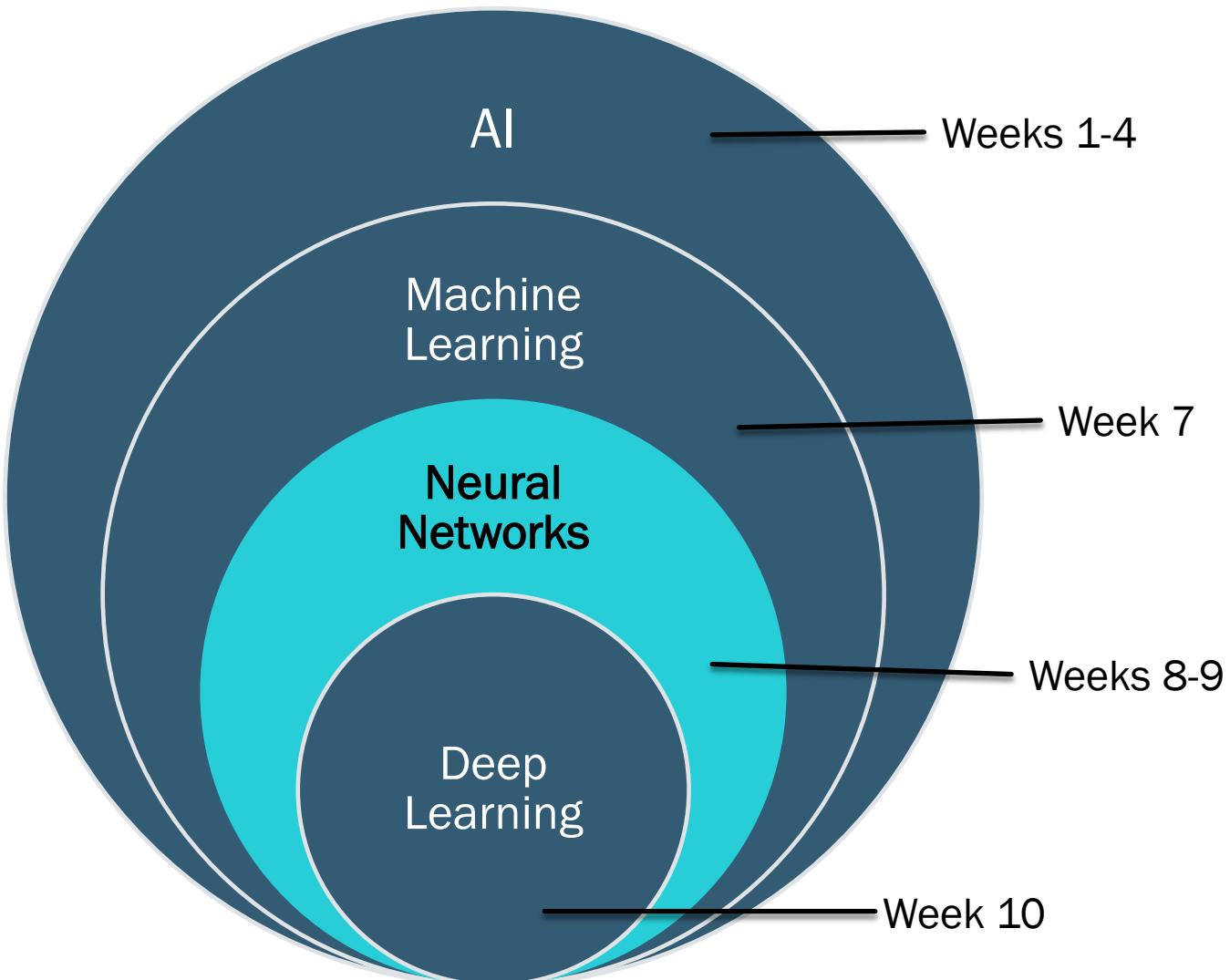
## Perceptron

- Understand the fundamental concept
- Understand inputs and outputs
- Understand weights
- Forward propagation
- Understand activation functions
- Be able to calculate the output of a perceptron

## Neural Networks

- Understand the concept
- Be able to list the name of the different layers
- Be able to describe how neural networks work in your own words
- Understand problems with data
- Understand hyperparameters
- Be able to evaluate performance and calculate accuracy

# NEURAL NETWORKS AND ARTIFICIAL INTELLIGENCE





< Activities

Moderate

Visual settings

Edit



When poll is active, respond at **PollEv.com/esterbonmati**



# What is a neural network?



No responses received yet. They will appear here...

# ARTIFICIAL NEURAL NETWORKS (ANN)

Why are they important?  
ANN can help computers make intelligent decisions with limited human assistance.  
They can learn and model the relationships between input and output data that are nonlinear and complex.



---

# WHAT DO THEY ALL HAVE IN COMMON?

---

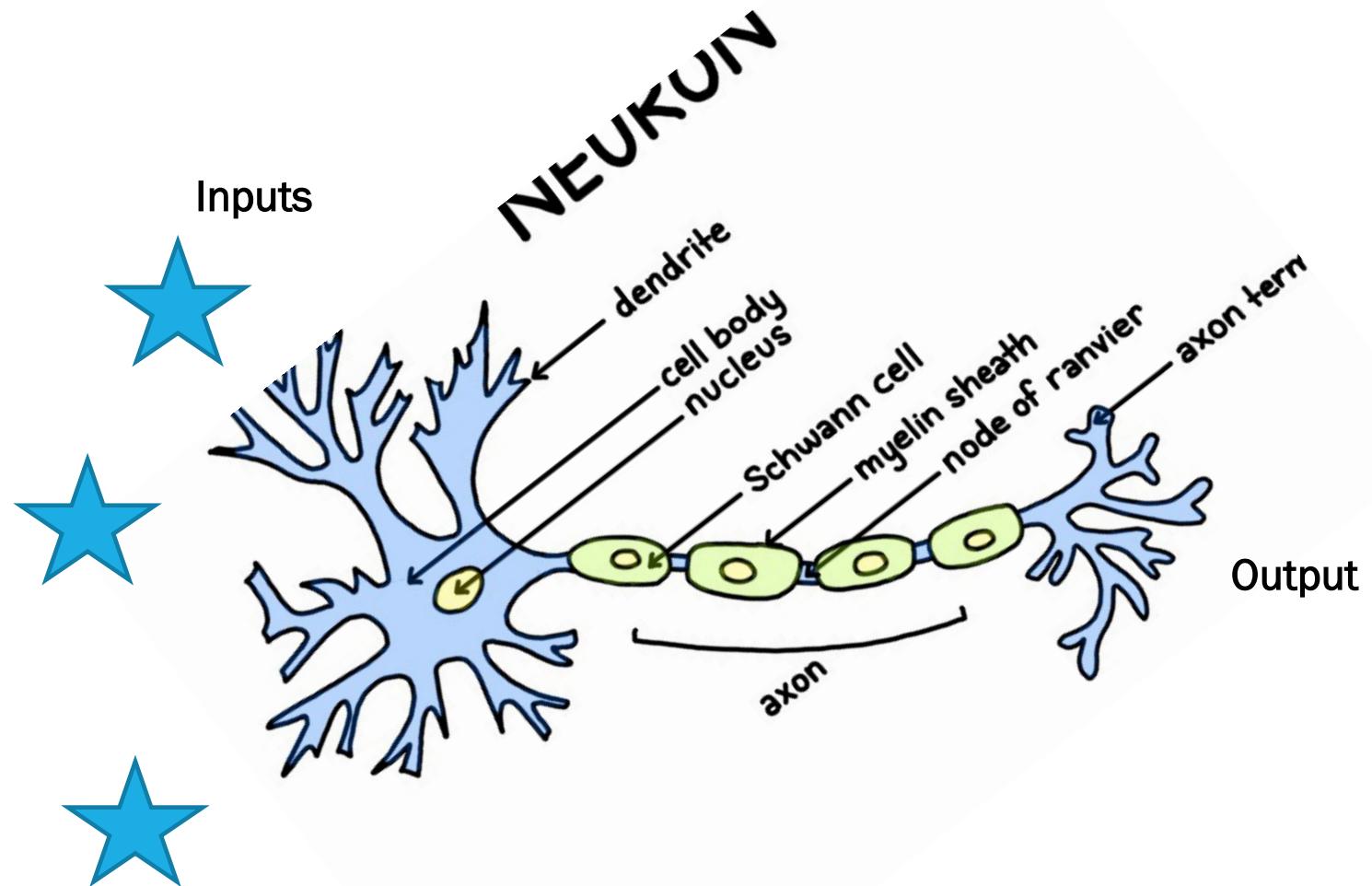


---

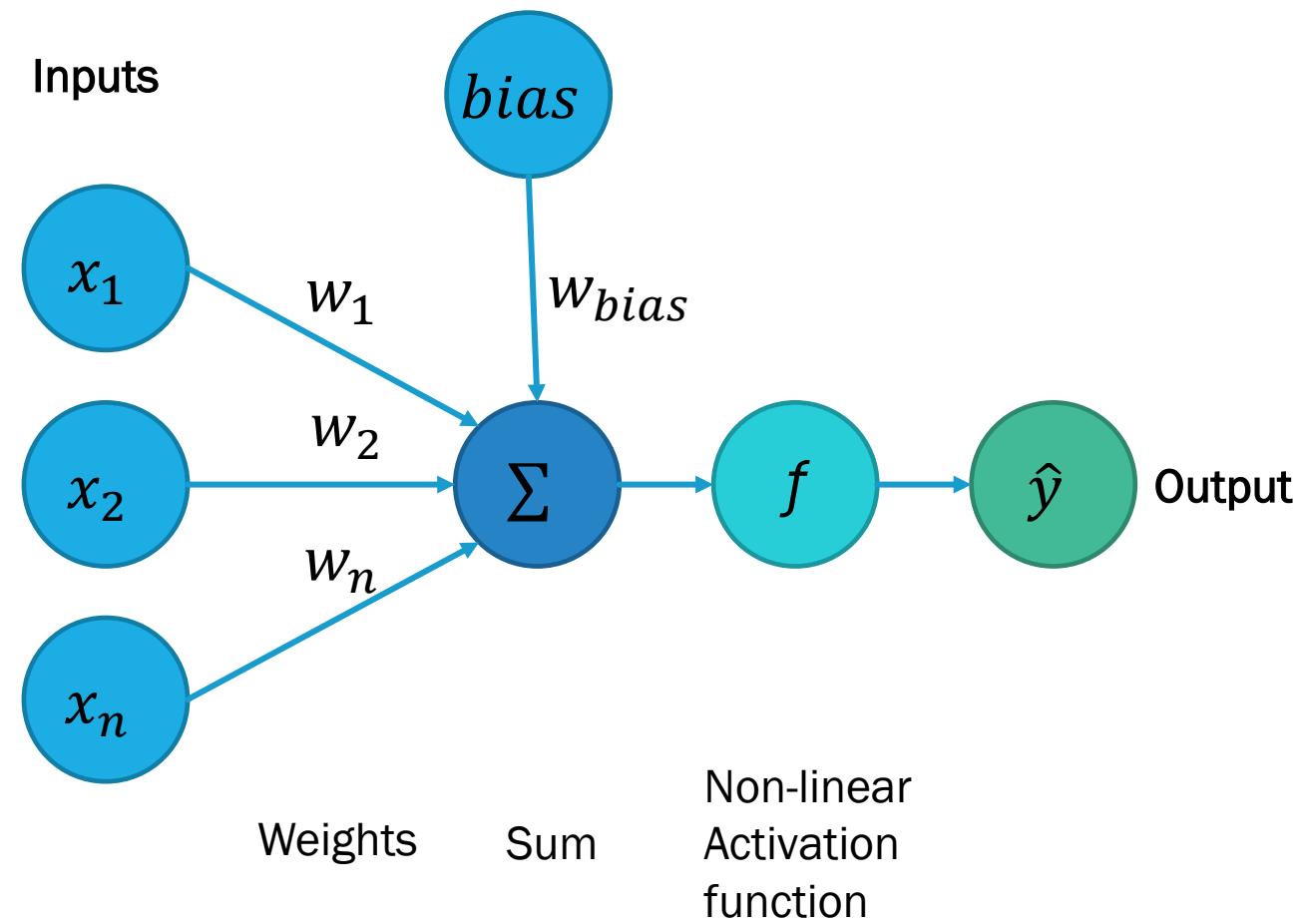
## WHAT DO THEY ALL HAVE IN COMMON?



# NEURON

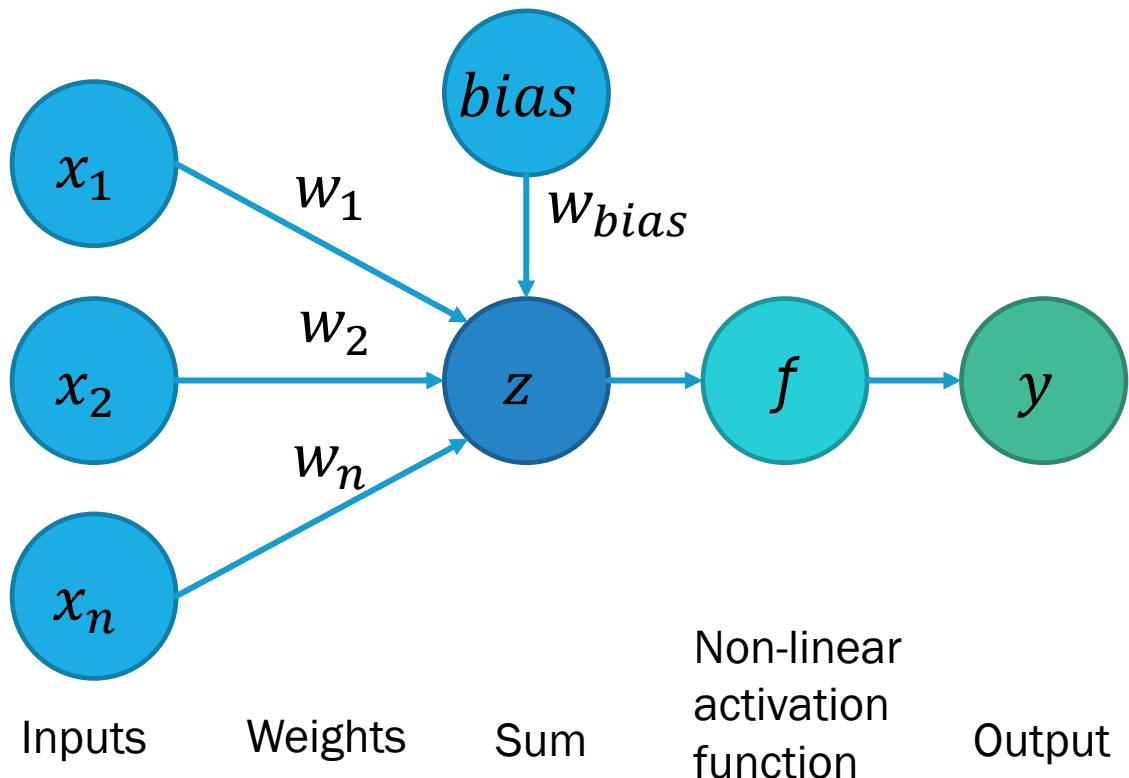


# THE PERCEPTRON



# THE PERCEPTRON

- Building block of neural networks (single neuron)
- Mimics the neuron in the human brain
- Forward propagation (input → neuron → output)



$$y = f \left( \text{bias } w_{bias} + \sum_{i=1}^n x_i w_i \right)$$

Labels for the equation:

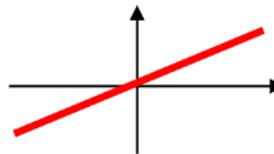
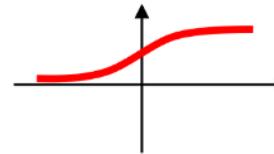
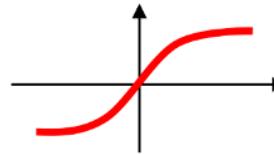
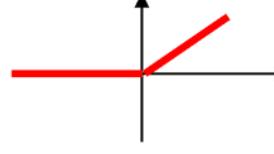
- Output: points to the variable  $y$ .
- Bias: points to the term  $w_{bias}$ .
- Inputs: points to the term  $x_i$ .
- Sum: points to the summation symbol  $\sum$ .
- Weights: points to the term  $w_i$ .
- Non-linear function: points to the function  $f$ .

$$z = x_1 w_1 + x_2 w_2 + \dots + x_n w_n + \text{bias } w_{bias}$$

$$y = f(z)$$

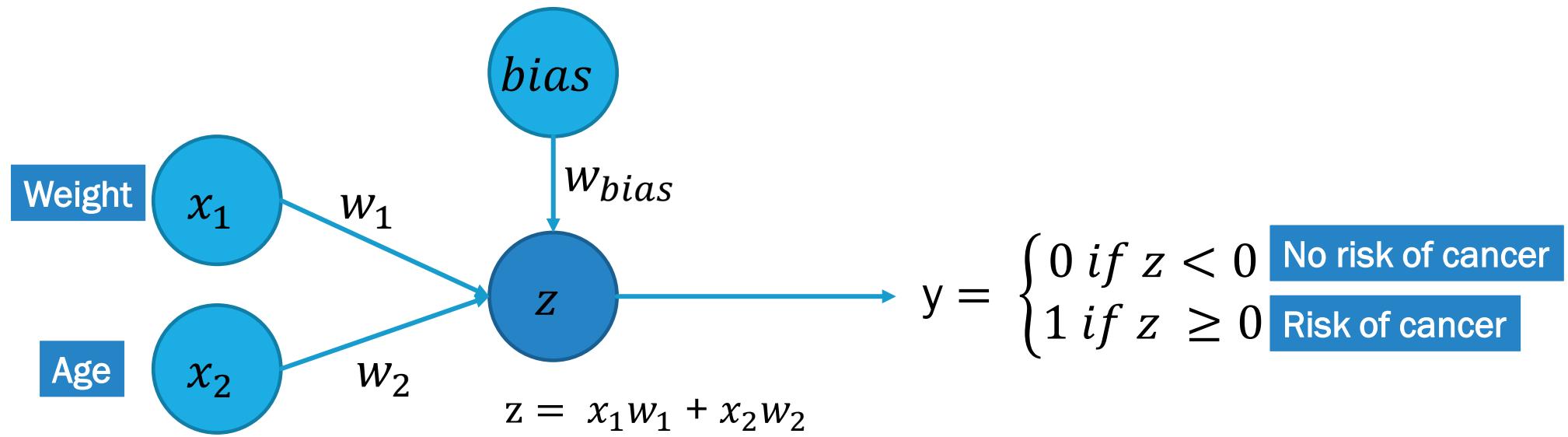
Example:  $y = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$

# FOR REFERENCE: ACTIVATION FUNCTIONS

Activation function	Equation	Example	1D Graph
Linear	$\phi(z) = z$	Adaline, linear regression	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent (Tanh)	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = \max(0, z)$	Multi-layer Neural Networks	

Non-linear

# PERCEPTRON: FORWARD PROPAGATION

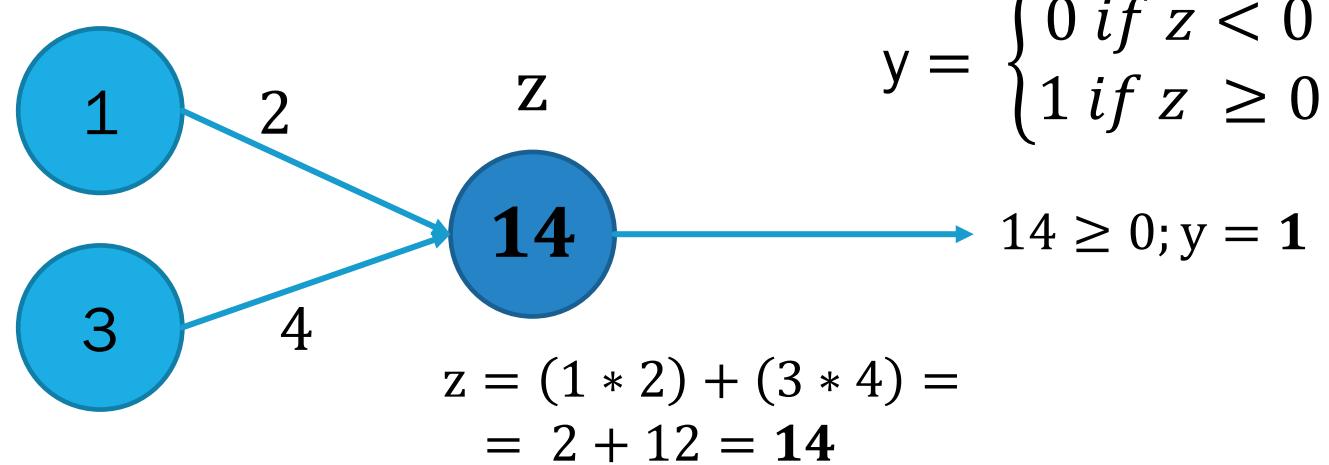


# **Do you know another example of a perceptron?**

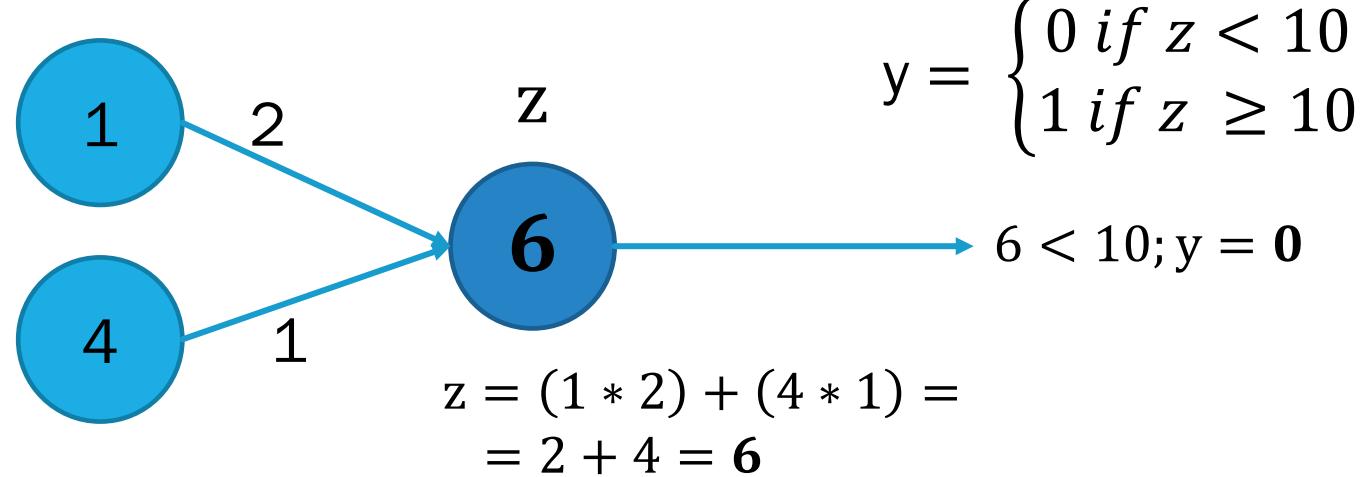
- Logic gates
- Predict who is going to pass this module based on the number of lectures and tutorials attended



## PERCEPTRON: FORWARD PROPAGATION EXAMPLE

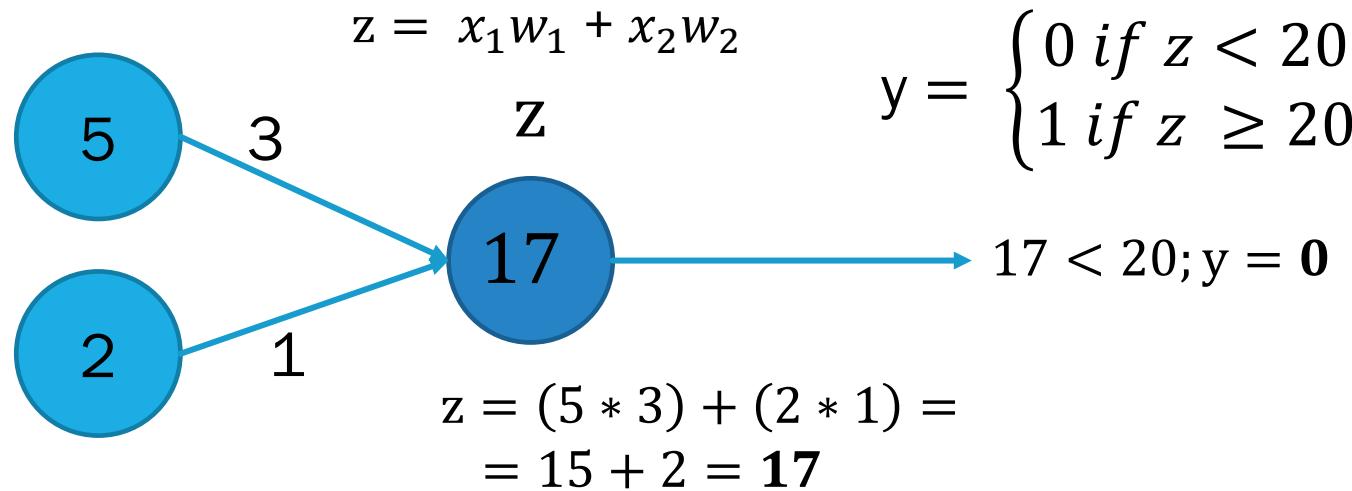


## PERCEPTRON: FORWARD PROPAGATION EXAMPLE



Do you think you can do it?

# PERCEPTRON: FEEDBACK



## Result perceptron 1

[Join by Web](#)



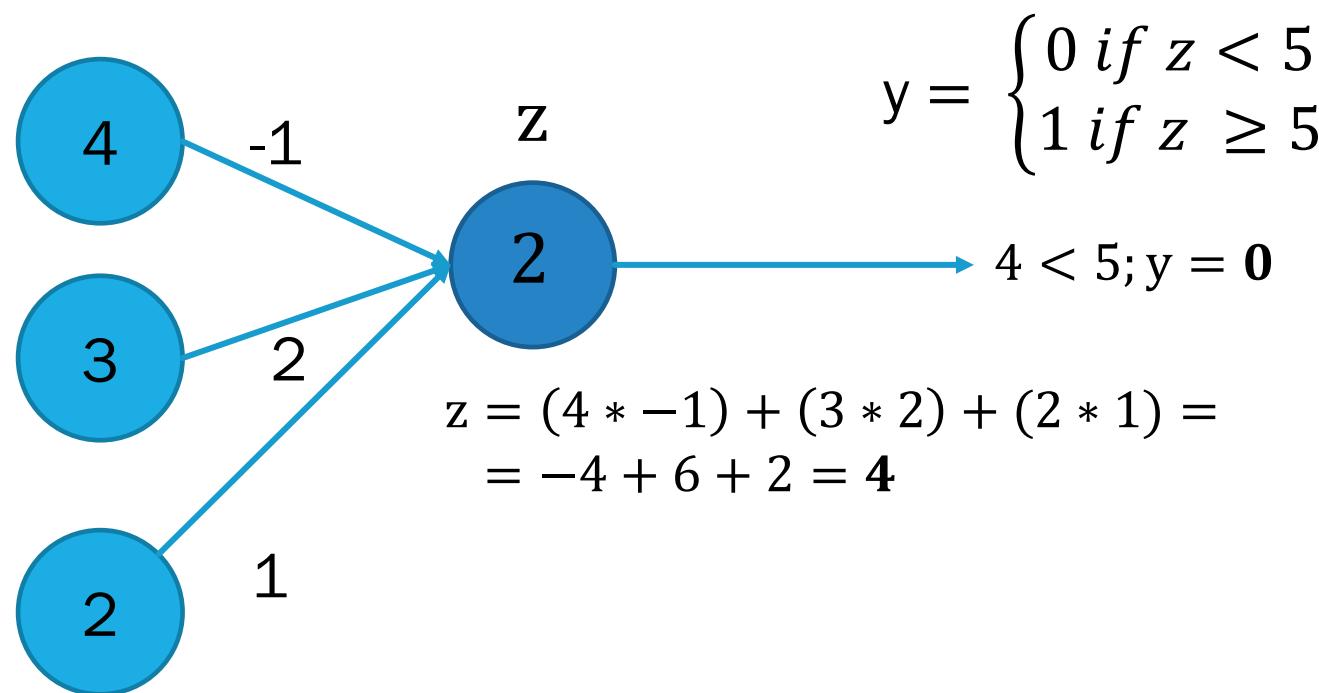
- 1 Go to [PollEv.com](#)
- 2 Enter **ESTERBONMATI**
- 3 Respond to activity

Total Results: 0

Powered by  [Poll Everywhere](#)

Compare answers with the person sitting next to you

# PERCEPTRON: FEEDBACK



Compare answers with the person sitting next to you

## Results perceptron 2

### Join by Web



- 1 Go to [PollEv.com](https://www.PollEv.com)
- 2 Enter **ESTERBONMATI**
- 3 Respond to activity

Total Results: 0

Powered by Poll Everywhere

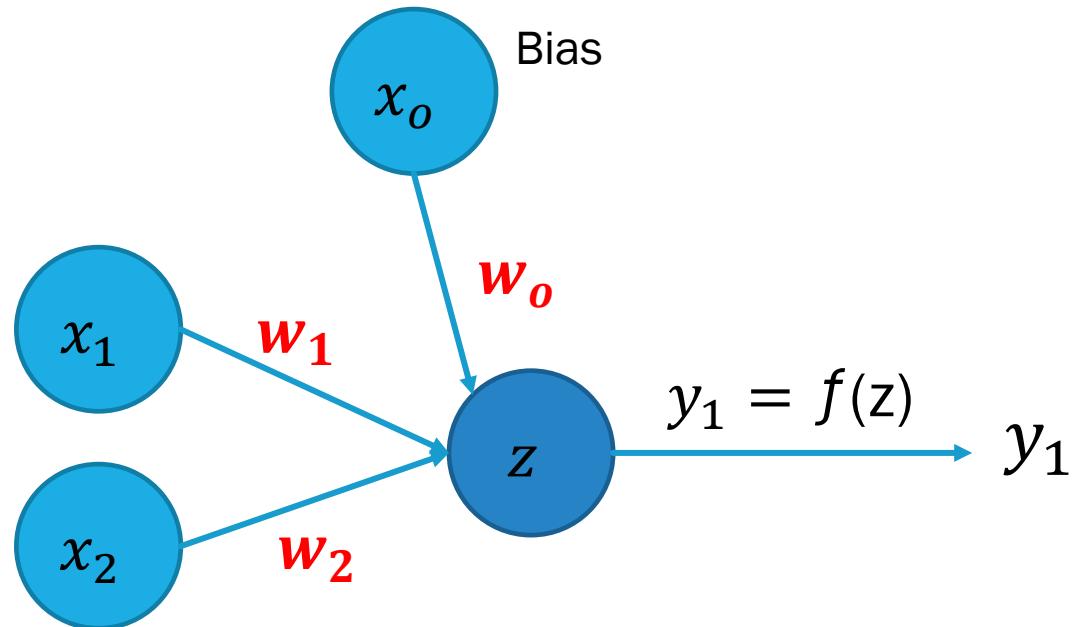
🌐 When poll is active, respond at **PollEv.com/esterbonmati**



# How confident do you feel in calculating the output of a perceptron?



## PERCEPTRON: A PRACTICAL EXAMPLE WITH ONLY 2 INPUTS (PYTHON)



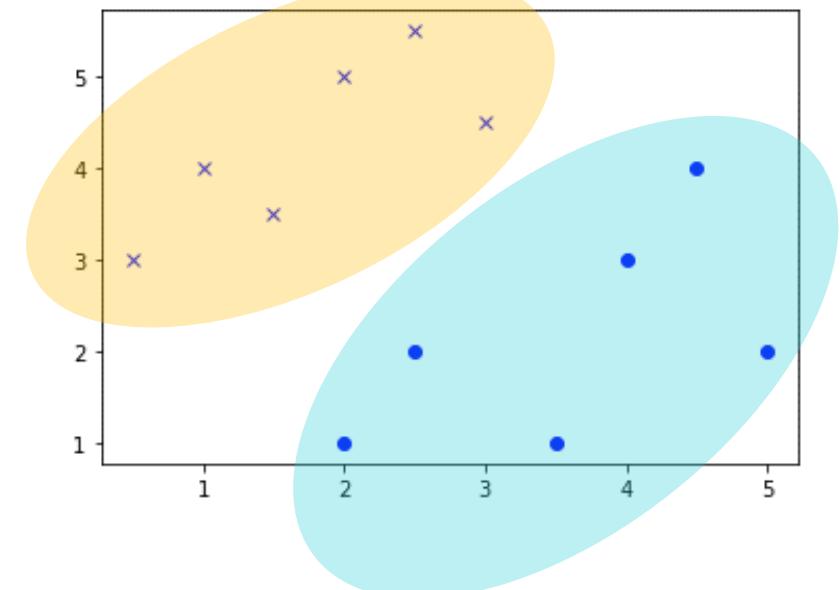
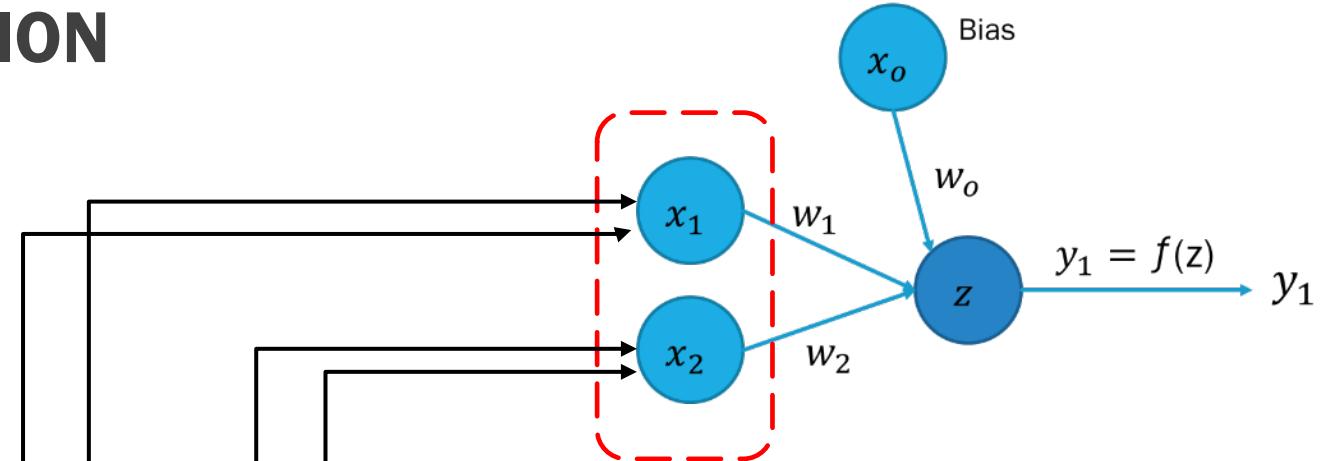
What do the weights mean and how do we find them?

$$z = x_1 w_1 + x_2 w_2 + x_0 w_0$$

# PERCEPTRON: INPUT DEFINITION

Problem: We have 2 groups of samples, each of them defined by 2 values, and we want to use a perceptron to classify them into their group.

```
x1_group1 = [2, 5, 4, 2.5, 3.5, 4.5]  
x2_group1 = [1, 2, 3, 2, 1, 4]  
  
x1_group0 = [0.5, 2.5, 3, 1.5, 1, 2]  
x2_group0 = [3, 5.5, 4.5, 3.5, 4, 5]
```

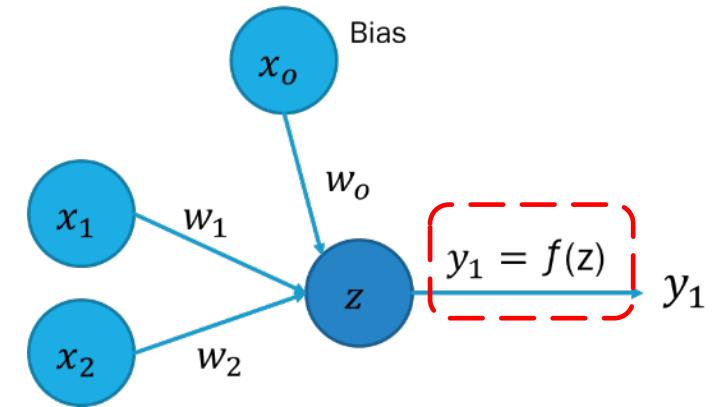


# PERCEPTRON: ACTIVATION FUNCTION

$$y = \begin{cases} -1 & \text{if } z \leq 0 \\ 1 & \text{if } z > 0 \end{cases}$$

If the sum is greater than 0 we return 1, otherwise we return 0

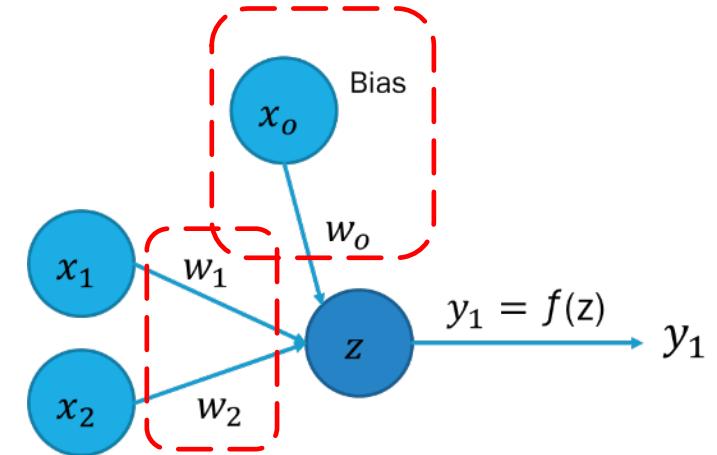
```
In [112]: def activation_function(value):
    if value > 0: return 1
    else: return -1
```



# PERCEPTRON: WEIGHTS AND BIAS DEFINITION

Some random weights...

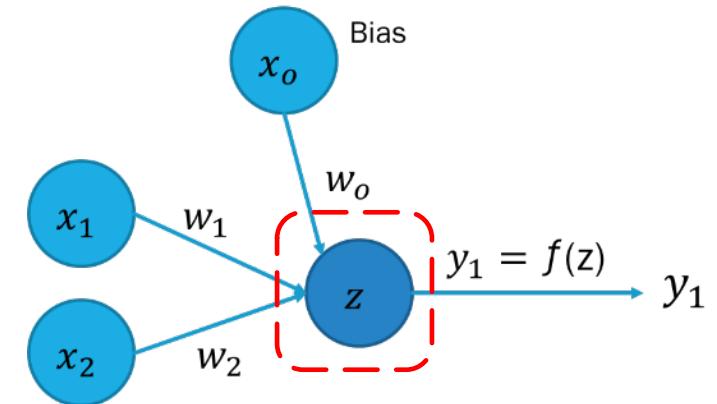
```
w1 = 0.75  
w2 = -0.5  
bias = 1  
wbias = 1
```



How good are these weights in grouping the input data?

# PERCEPTRON: FORWARD PROPAGATION

```
def perceptron(x1, x2, bias):  
    z = x1*w1 + x2*w2 + bias*wbias  
    return z
```

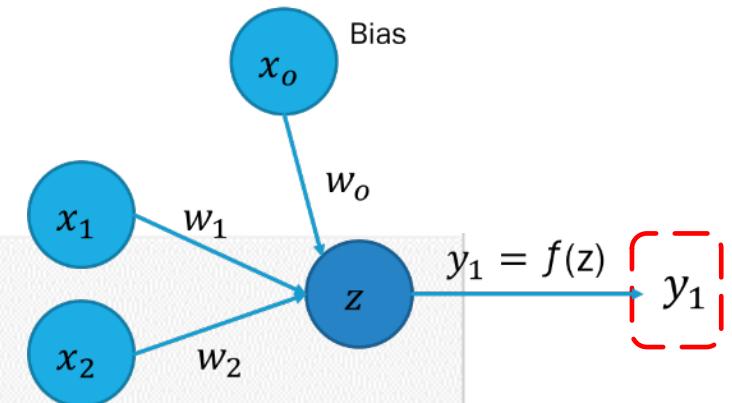


# PERCEPTRON: PREDICTION AND RESULTS

```
group = activation_function(z)
print('Point ' + str(i) + ' in group 1 returned ' + str(group))

for i in range(len(x1_group0)):
    z = perceptron(x1_group0[i], x2_group0[i], bias)
    group = activation_function(z)
    print('Point ' + str(i) + ' in group 0 returned ' + str(group))
```

Point 0 in group 1 returned 1  
Point 1 in group 1 returned 1  
Point 2 in group 1 returned 1  
Point 3 in group 1 returned 1  
Point 4 in group 1 returned 1  
Point 5 in group 1 returned 1  
Point 0 in group 0 returned -1  
Point 1 in group 0 returned 1  
Point 2 in group 0 returned 1  
Point 3 in group 0 returned 1  
Point 4 in group 0 returned -1  
Point 5 in group 0 returned -1



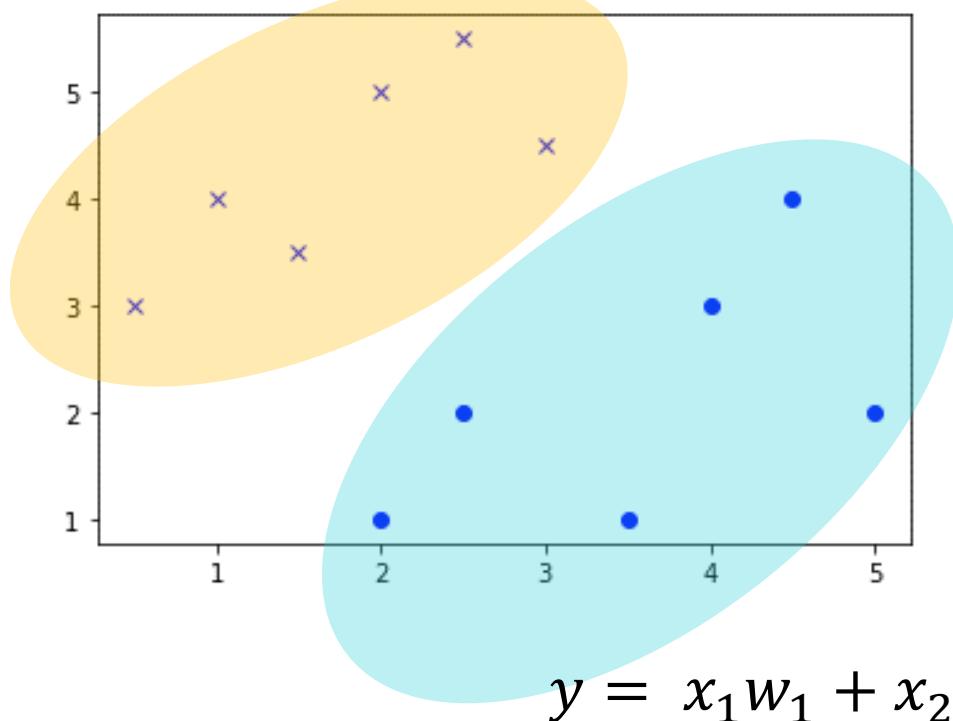
$$\text{accuracy} = \frac{\text{number correct samples}}{\text{total number of samples}} = \frac{9}{12} = 0.75 \rightarrow 75\%$$

Can we improve the accuracy if we change the weights?

# PERCEPTRON: VISUALISATION

## Plot the points

```
In [117]: fig = plt.figure()
ax = plt.axes()
plt.plot(x1_group1, x2_group1, 'ob')
plt.plot(x1_group0, x2_group0, 'xb')
plt.show()
```

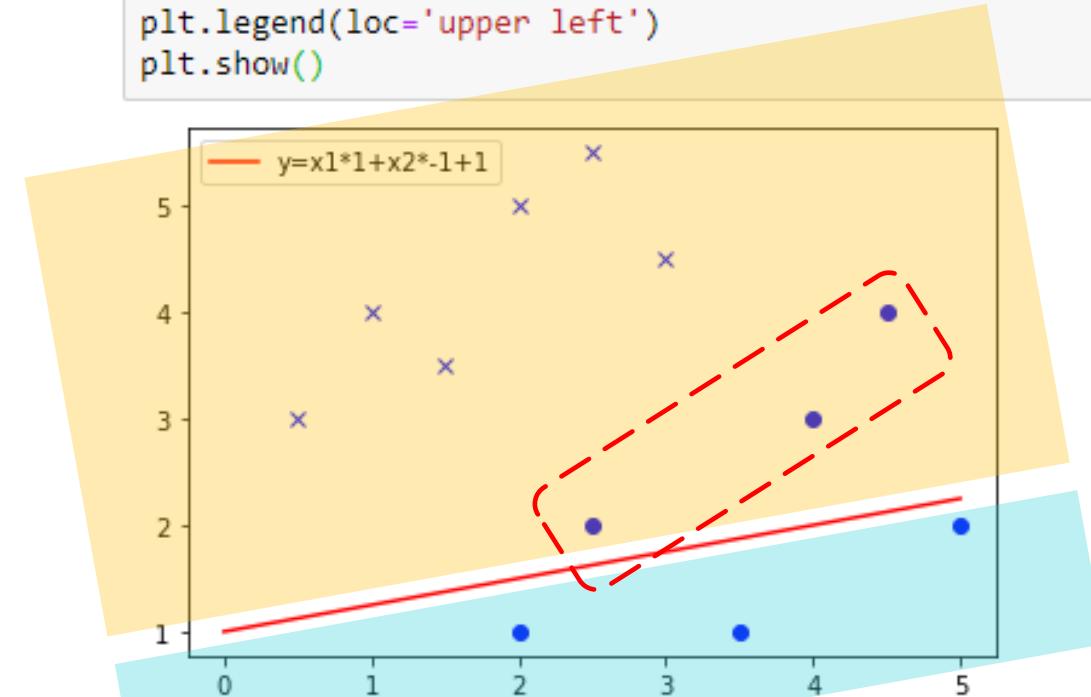


```
In [101]:
```

```
x1 = np.linspace(0,5,100)
x2 = np.linspace(0,5,100)
y_1 = w1*x1 + w2*x2 + 1
```

```
plt.plot(x1_group1, x2_group1, 'ob')
plt.plot(x1_group0, x2_group0, 'xb')
plt.plot(x1, y_1, '-r', label='y=x1*0.75+x2*-0.5+1*1')
plt.legend(loc='upper left')
plt.show()
```

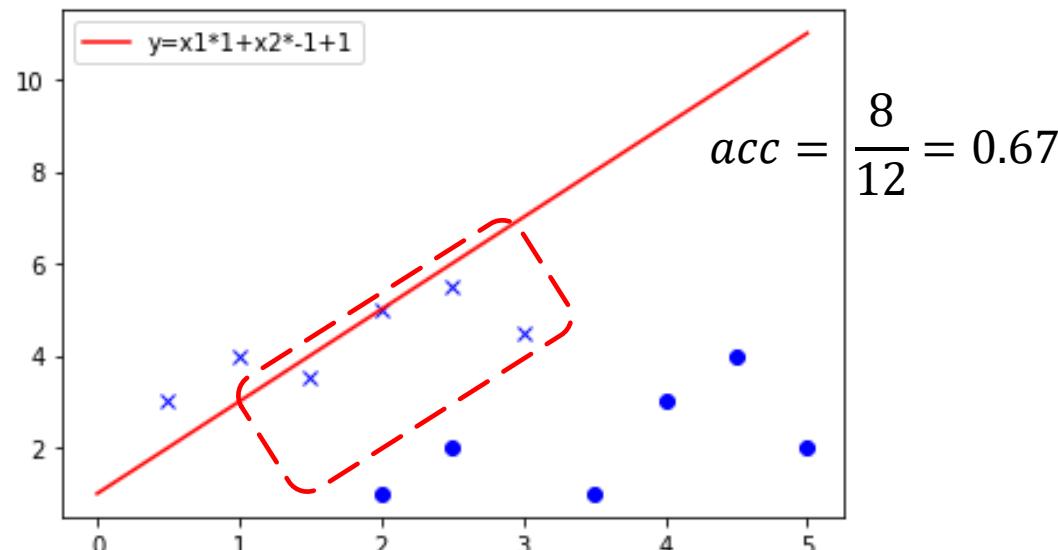
## Plot linear activation



$$accuracy = \frac{9}{12} = 0.75$$

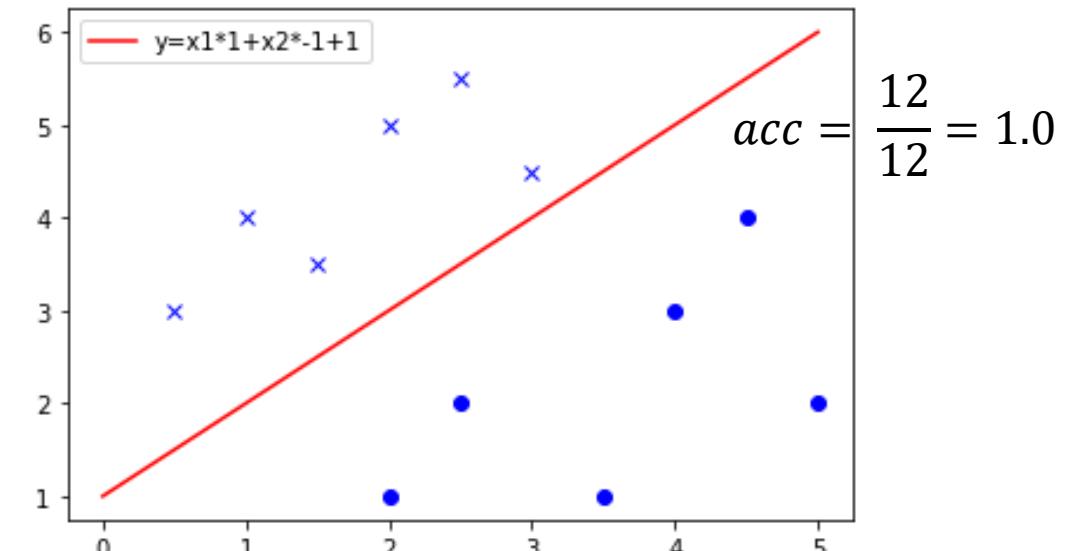
# PERCEPTRON: UPDATE WEIGHTS

```
In [102]: w1 = 1  
w2 = 1  
bias = 1  
wbias = 1  
  
y_1 = w1*x1 + w2*x2 + bias*wbias  
  
plt.plot(x1_group1, x2_group1, 'ob')  
plt.plot(x1_group0, x2_group0, 'xb')  
plt.plot(x1, y_1, '-r', label='y=x1*1+x2*1+1*1')  
plt.legend(loc='upper left')  
plt.show()
```



To change the prediction output, we need to change the weights

```
In [103]: w1 = 3  
w2 = -2  
bias = 1  
wbias = -1  
  
y_1 = x1*w1 + x2*w2 + bias*bias  
  
plt.plot(x1_group1, x2_group1, 'ob')  
plt.plot(x1_group0, x2_group0, 'xb')  
plt.plot(x1, y_1, '-r', label='y=x1*3+x2*-2+1*-1')  
plt.legend(loc='upper left')  
plt.show()
```



# PERCEPTRON: NEW PREDICTION

```
In [104]: for i in range(len(x1_group1)):
    sum = perceptron(x1_group1[i], x2_group1[i], bias)
    group = activation_function(sum)
    print('Point ' + str(i) + ' in group 1 returned ' + str(group))

for i in range(len(x1_group0)):
    sum = perceptron(x1_group0[i], x2_group0[i], bias)
    group = activation_function(sum)
    print('Point ' + str(i) + ' in group 0 returned ' + str(group))
```

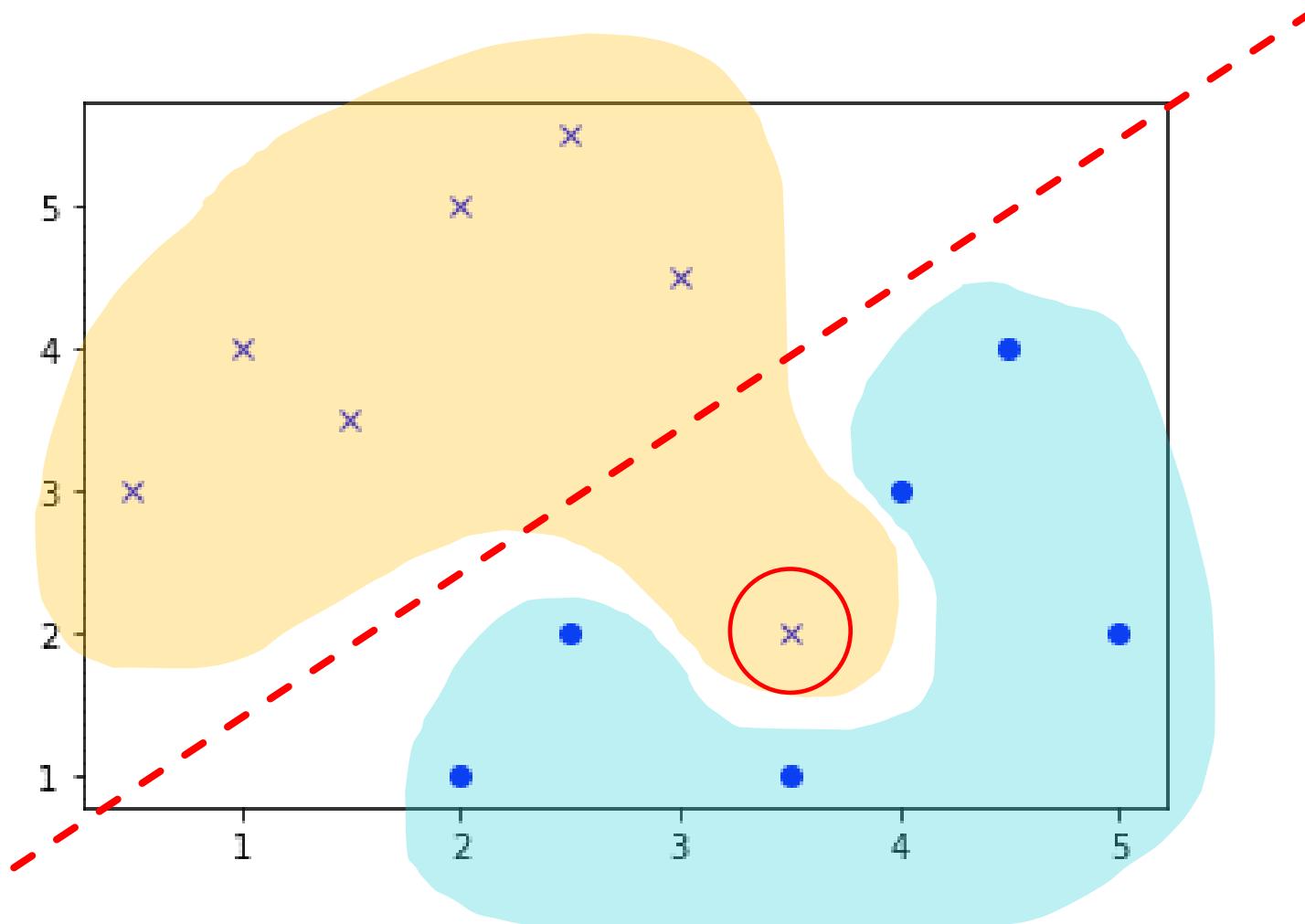
```
Point 0 in group 1 returned 1
Point 1 in group 1 returned 1
Point 2 in group 1 returned 1
Point 3 in group 1 returned 1
Point 4 in group 1 returned 1
Point 5 in group 1 returned 1
Point 0 in group 0 returned -1
Point 1 in group 0 returned -1
Point 2 in group 0 returned -1
Point 3 in group 0 returned -1
Point 4 in group 0 returned -1
Point 5 in group 0 returned -1
```



$$acc = \frac{12}{12} = 1.0 \rightarrow 100\%$$



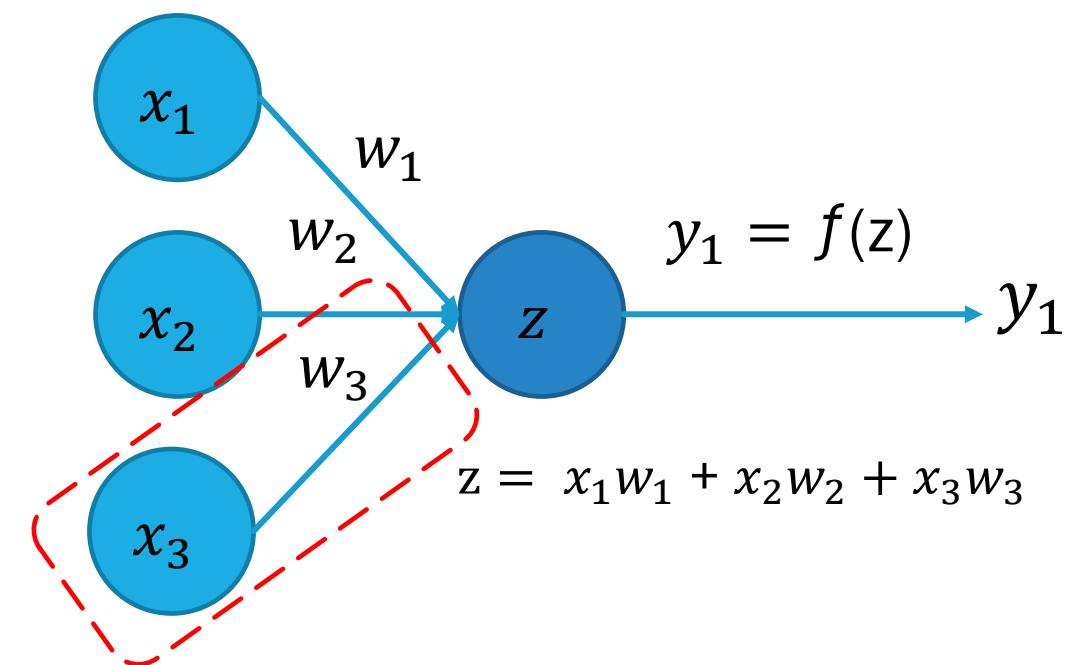
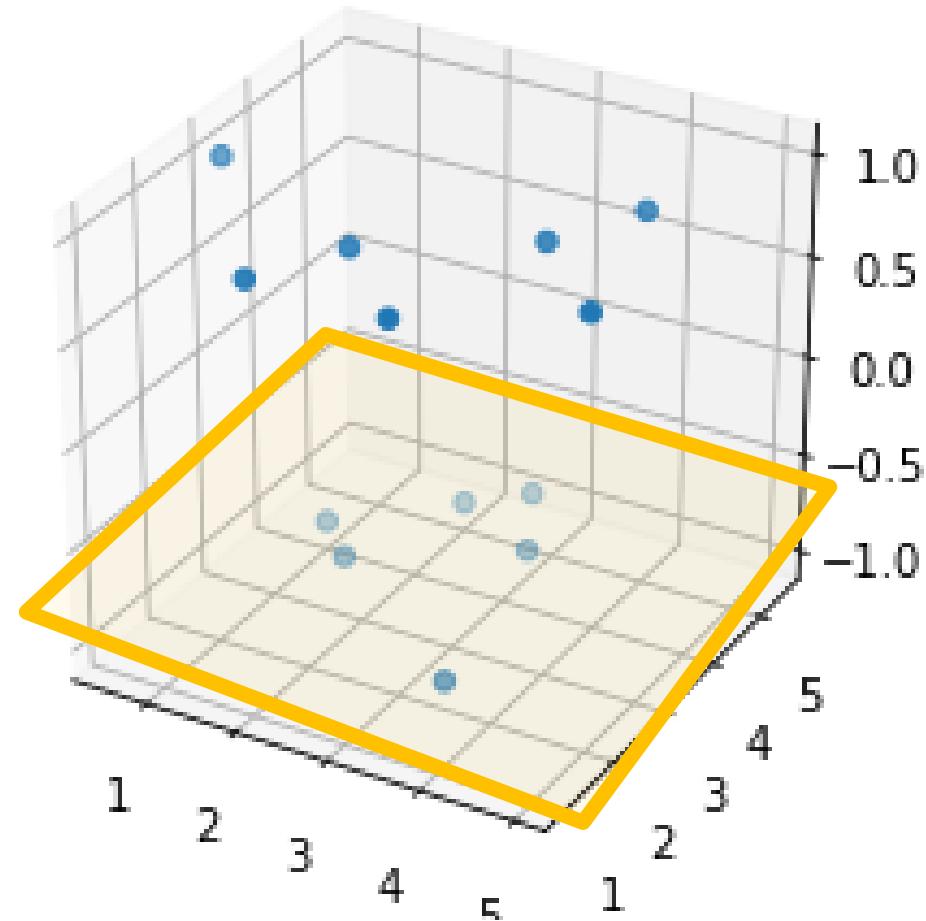
## NEURAL NETWORKS: ANOTHER EXAMPLE



# NEURAL NETWORKS: ANOTHER EXAMPLE

Bias

$$y = x_1 w_1 + x_2 w_2 + \boxed{x_3 w_3} + b$$







< Activities



Visual settings



Edit



When poll is active, respond at [PollEv.com/esterbonmati](https://PollEv.com/esterbonmati)



## How many inputs can have a perceptron?

2  
3  
n



< Activities



Visual settings



Edit



When poll is active, respond at [PollEv.com/esterbonmati](https://PollEv.com/esterbonmati)



## How many weights has each input?

0  
1  
2  
n



When poll is active, respond at [PollEv.com/esterbonmati](https://PollEv.com/esterbonmati)



## We have a perceptron with 2 inputs. How do you calculate Z?

$z = x_1 * w_1 + x_2 * w_2 +$   
bias \* wbias

$z = x_1 * w_1 + x_2 * w_2$

$z = x_1 * w_1 + x_2 * w_2 +$   
 $x_3 * w_3 + \text{bias} * \text{wbias}$



< Activities



When poll is active, respond at **PollEv.com/esterbonmati**

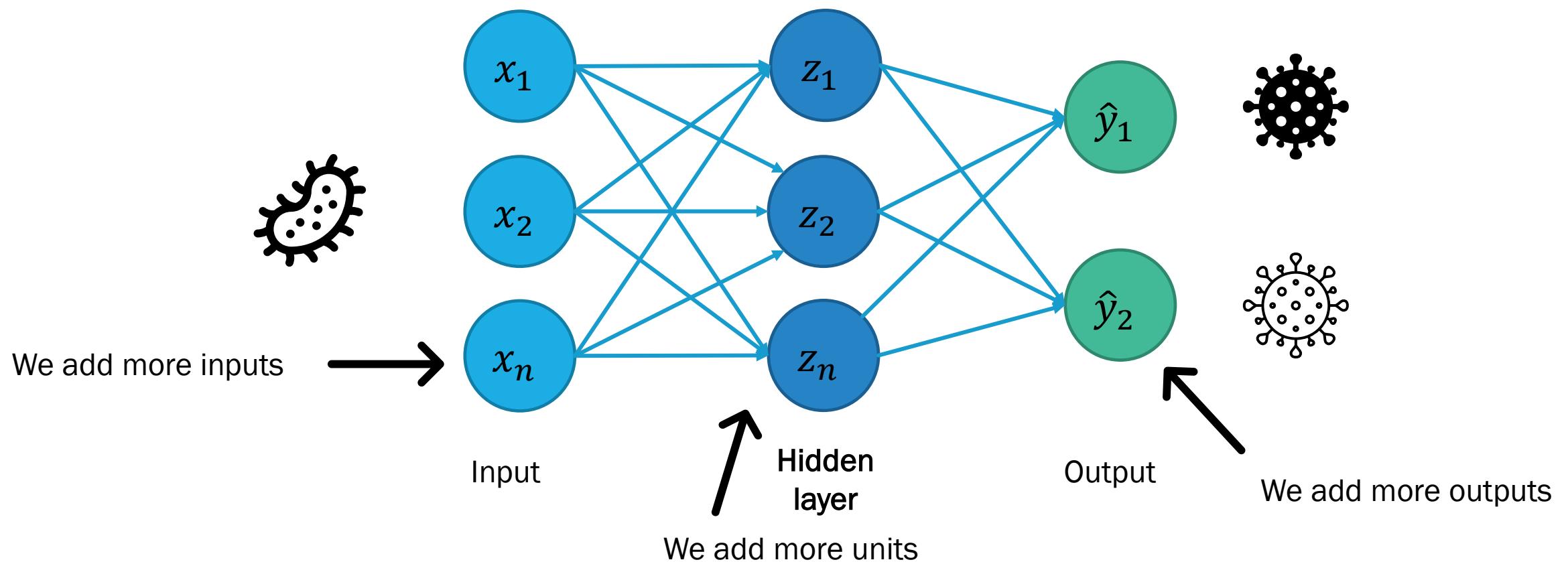


## Q&A

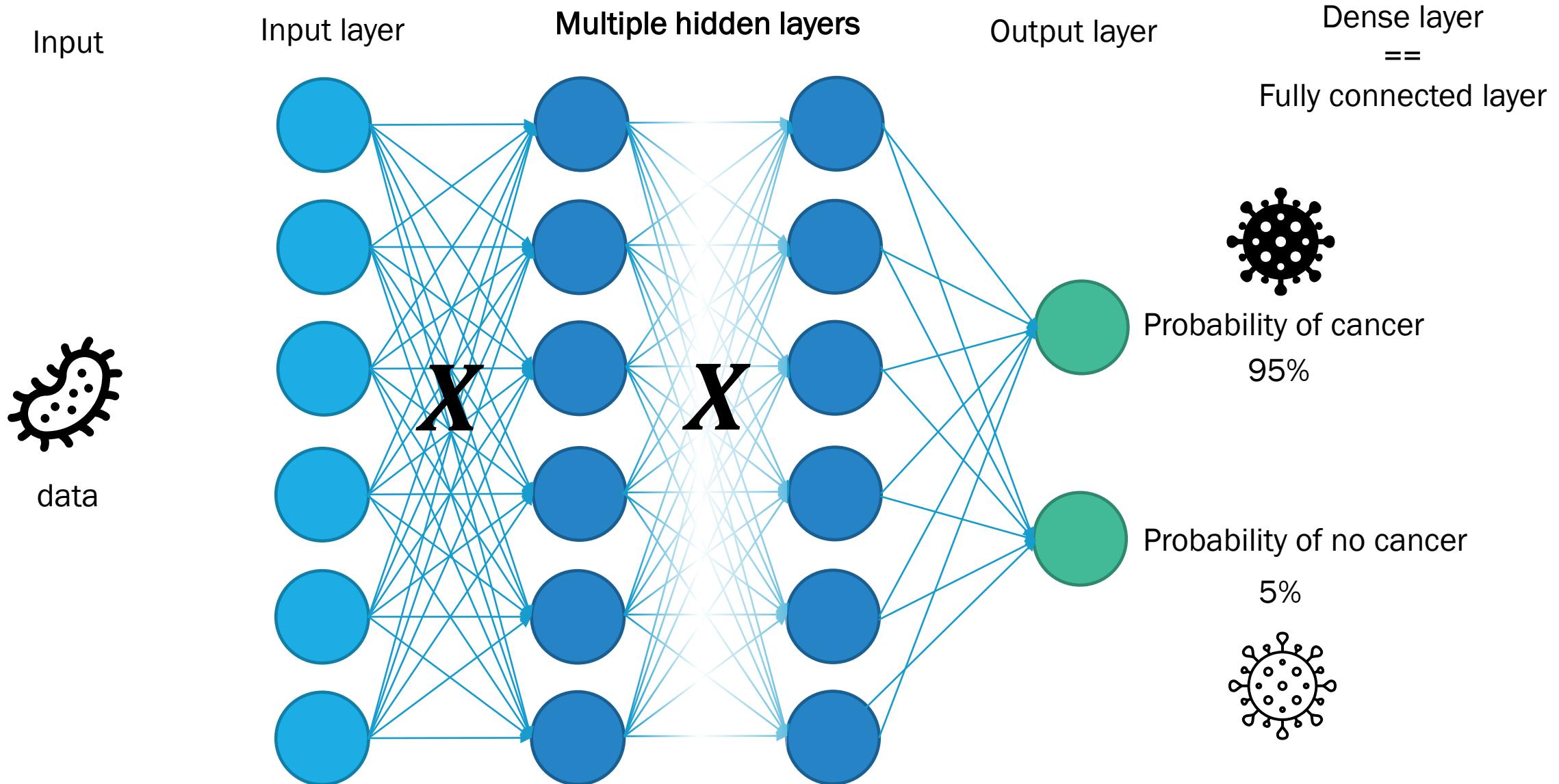
Top

No responses received yet. They will appear here...

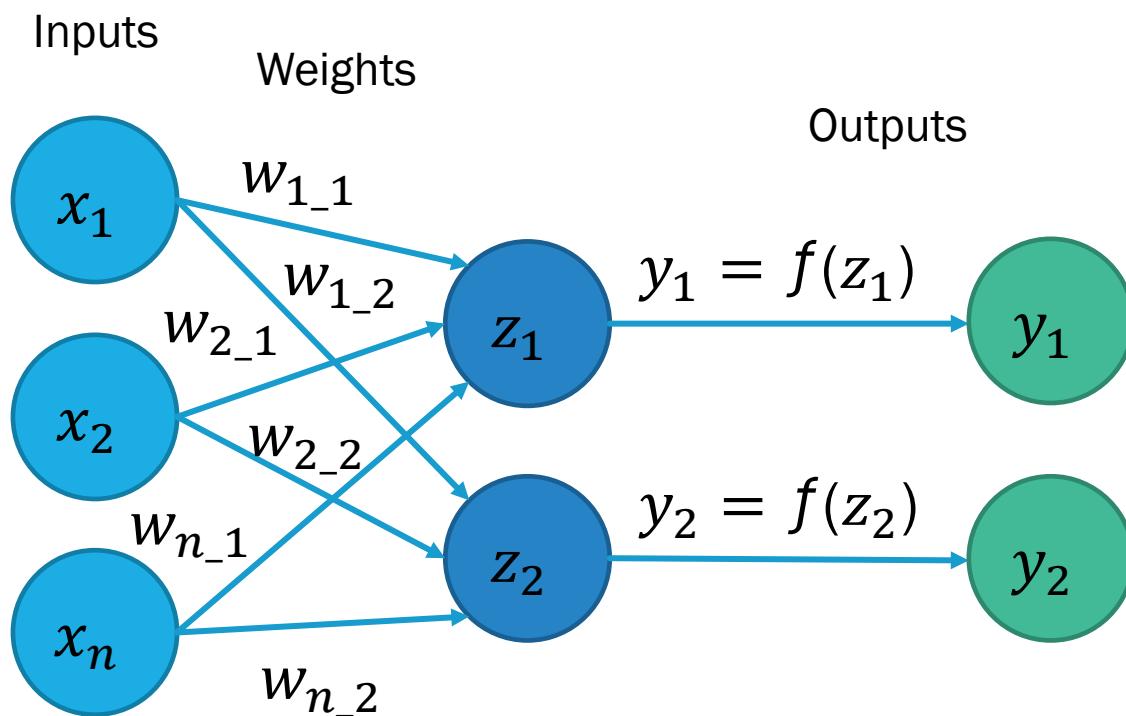
# FROM PERCEPTRON TO A NEURAL NETWORK



# NEURAL NETWORK EXAMPLE



# ARTIFICIAL NEURAL NETWORKS (ANN)



Each output has  
its own weight  
for each input

$$z = w_0 + \sum_{i=1}^n x_i w_i$$

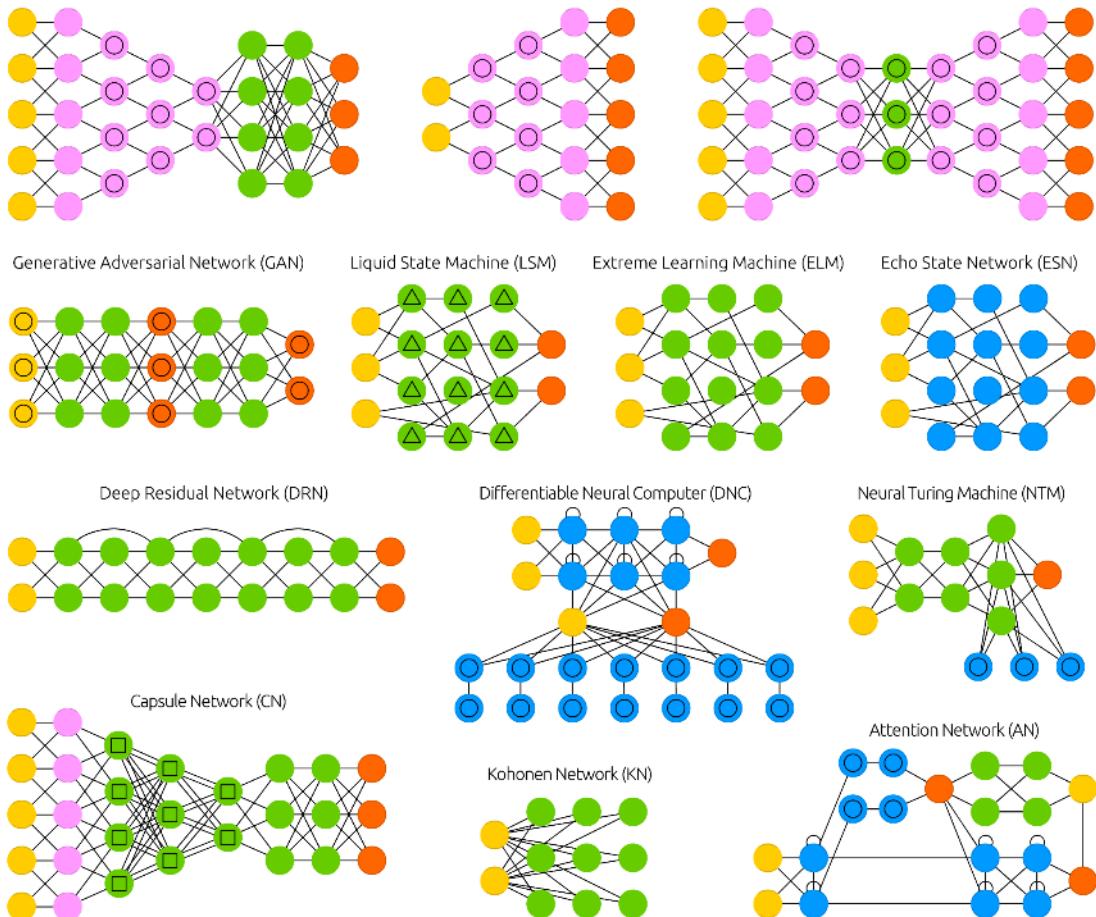
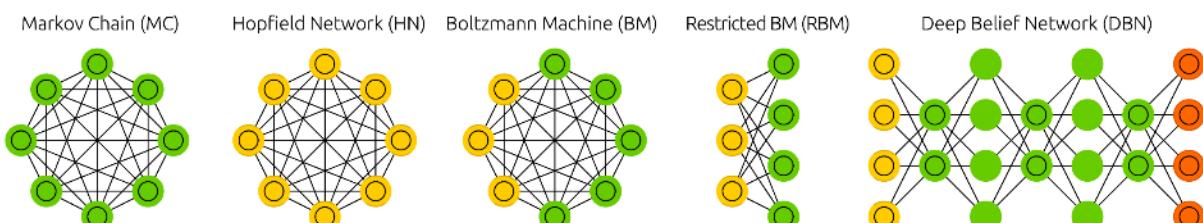
$$\hat{y} = f \left( w_0 + \sum_{i=1}^n x_i w_i \right)$$

# EXAMPLES OF NEURAL NETWORKS

- Input Cell
- Backfed Input Cell
- △ Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- △ Spiking Hidden Cell
- Capsule Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- △ Gated Memory Cell
- Kernel
- Convolution or Pool

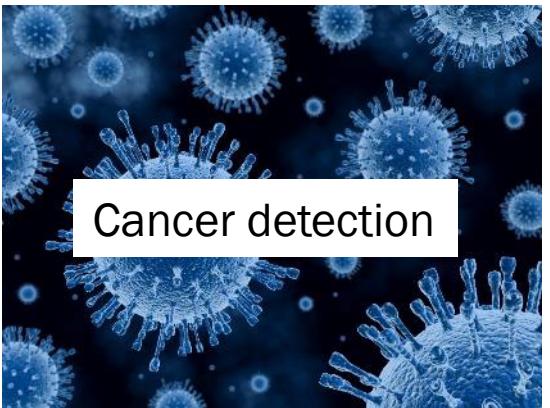
## A mostly complete chart of Neural Networks

©2019 Fjodor van Veen & Stefan Leijnen asimovinstitute.org



Pictures from:  
<https://www.asimovinstitute.org/neural-network-zoo/>

# WHAT CAN WE DO WITH ANN?



Cancer detection

Shopping prediction



Recognition/Classification



Interests prediction



Weather prediction



Stock market prediction



Chatbots



Drug discovery

# RESEARCH

## Forecasting Climatic Trends Using Neural Networks: An Experimental Study Using Global Historical Data

Takeshi Ise<sup>1,2\*</sup> and Yurika Oba<sup>1</sup>

<sup>1</sup> Field Science Education and Research Center (FSERC), Kyoto University, Kyoto, Japan, <sup>2</sup> Japan Science and Technology Agency (JST), Kawaguchi, Japan

IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, VOL. 32, NO. 10, OCTOBER 2021

Review Article

## Artificial neural networks in the cancer genomics frontier

Andrew Oustimov<sup>1</sup>, Vincent Vu<sup>2</sup>

<sup>1</sup>Department of Epidemiology & Biostatistics, College of Public Health, University of South Florida, Tampa, FL 33620, USA; <sup>2</sup>Department of Mathematics and Statistics, University of California, Los Angeles, CA, USA

Correspondence to: Andrew Oustimov, MPH. Department of Epidemiology & Biostatistics, College of Public Health, University of South Florida, 13201 Bruce B Downs Blvd, Tampa, FL 33620, USA. Email: aoustimo@mail.usf.edu.

## Attention in Natural Language Processing

Andrea Galassi<sup>1</sup>, Marco Lippi<sup>1</sup>, and Paolo Torroni<sup>1</sup>

4291

## AUTOMATIC LANGUAGE IDENTIFICATION USING DEEP NEURAL NETWORKS

Ignacio Lopez-Moreno<sup>1</sup>, Javier Gonzalez-Dominguez<sup>1,2</sup>, Oldrich Plchot<sup>3</sup>, David Martinez<sup>4</sup>, Joaquin Gonzalez-Rodriguez<sup>2</sup>, Pedro Moreno<sup>1</sup>

<sup>1</sup>Google Inc., New York, USA

<sup>2</sup>ATVS-Biometric Recognition Group, Universidad Autonoma de Madrid, Spain

<sup>3</sup>Brno University of Technology, Czech Republic

<sup>4</sup>Aragon Institute for Engineering Research (I3A), University of Zaragoza, Spain

{elnota, jgd}@google.com

Tran et al. *Genome Medicine* (2021) 13:152  
<https://doi.org/10.1186/s13073-021-00968-x>

Genome Medicine

REVIEW

Open Access

## Deep learning in cancer diagnosis, prognosis and treatment selection

Khoa A. Tran<sup>1,2</sup>, Olga Kondrashova<sup>1</sup>, Andrew Bradley<sup>4</sup>, Elizabeth D. Williams<sup>2,3</sup>, John V. Pearson<sup>1</sup> and Nicola Waddell<sup>1\*</sup>



# DEEP FAKE

- Synthetic image or video generated using someone else's face or a new generated face.
- Brings concerns about fraud, credibility and authenticity, etc.

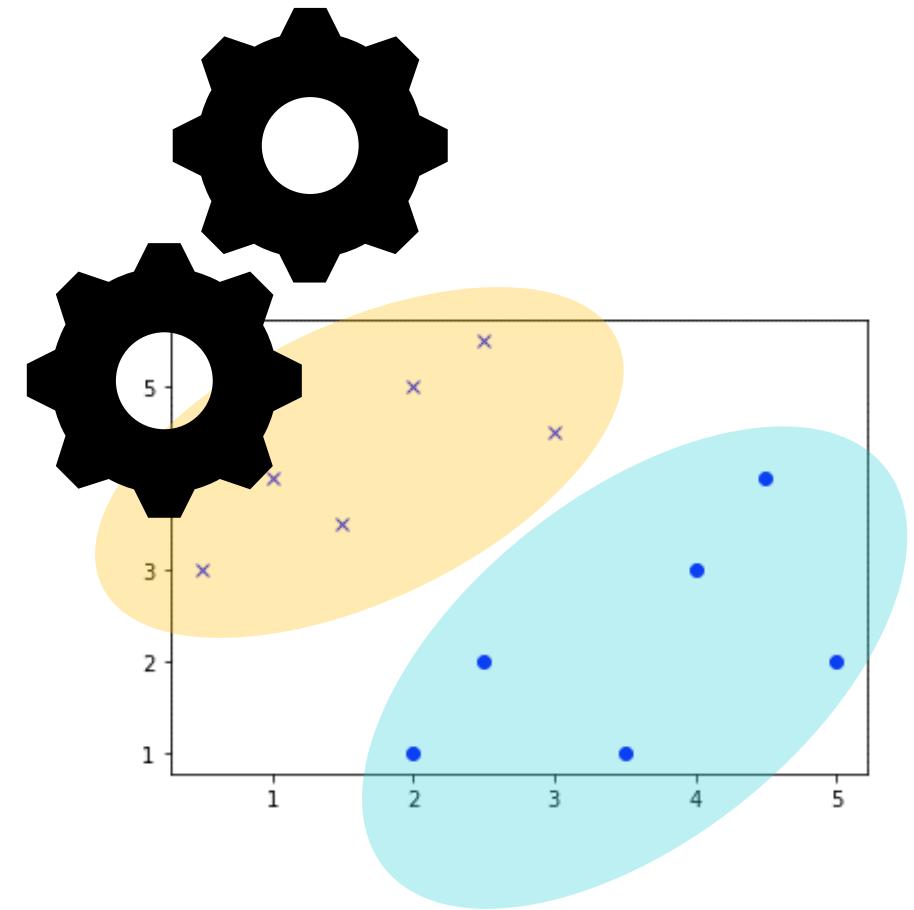


Are you sure?



Video generated  
using FaceMagic  
APP

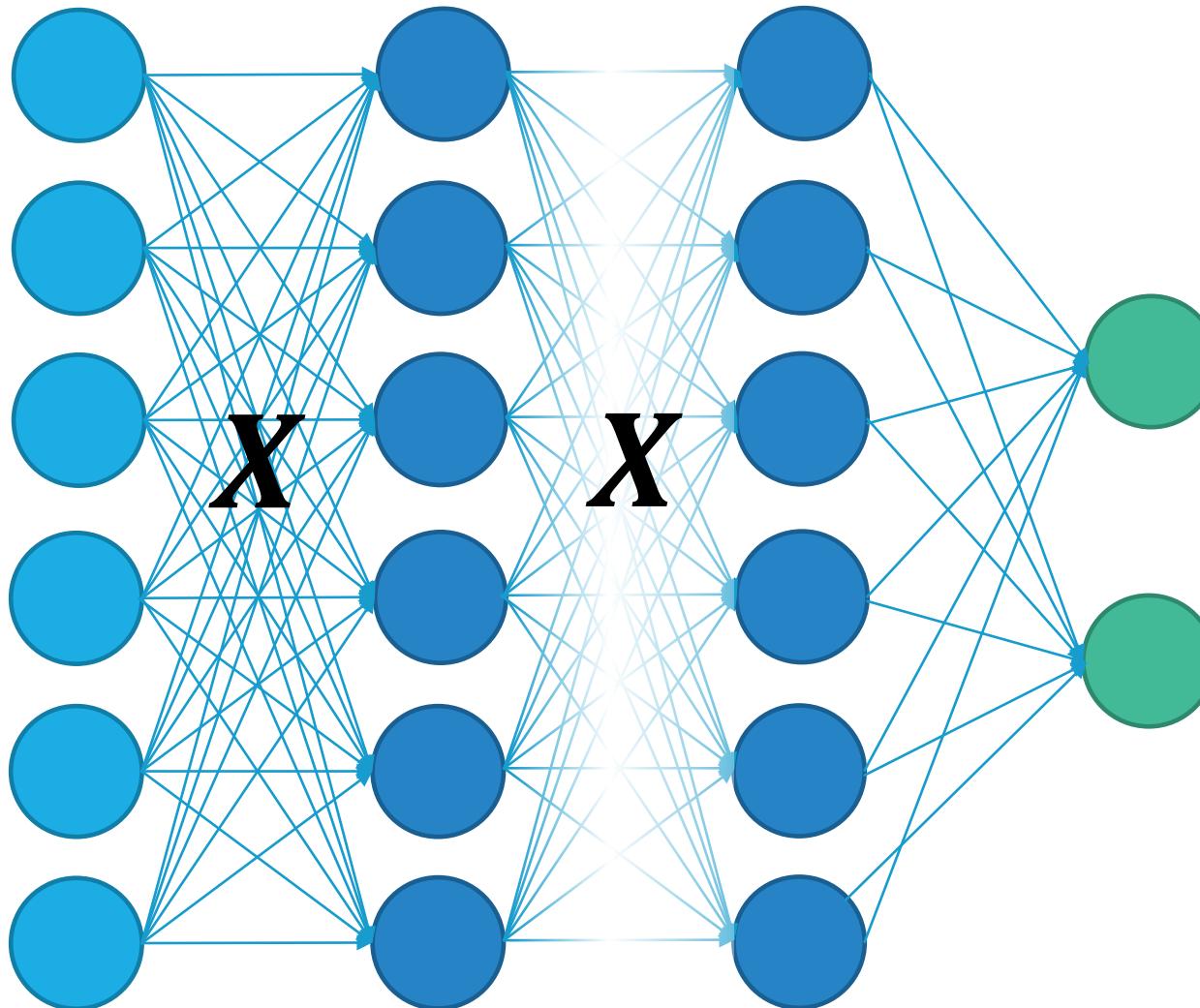
# SUPERVISED VS UNSUPERVISED



We have both **supervised** and **unsupervised** neural networks. We will focus the rest of the lecture on supervised learning.

# NEURAL NETWORK EXAMPLE

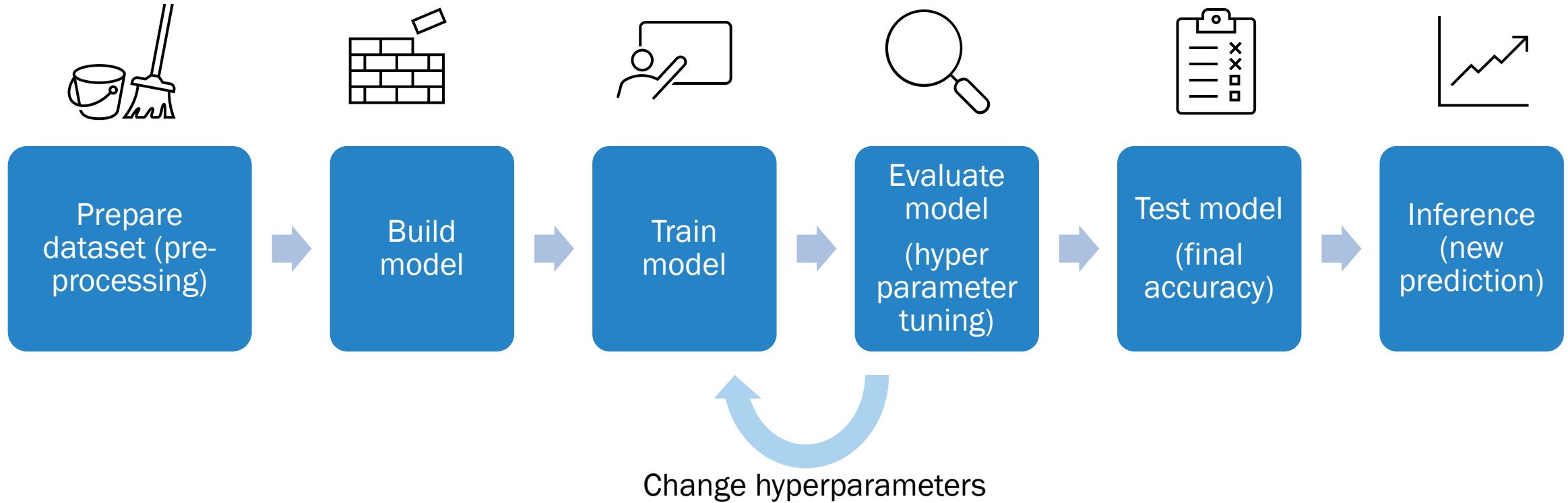
Input              Input layer              Multiple hidden layers              Output layer



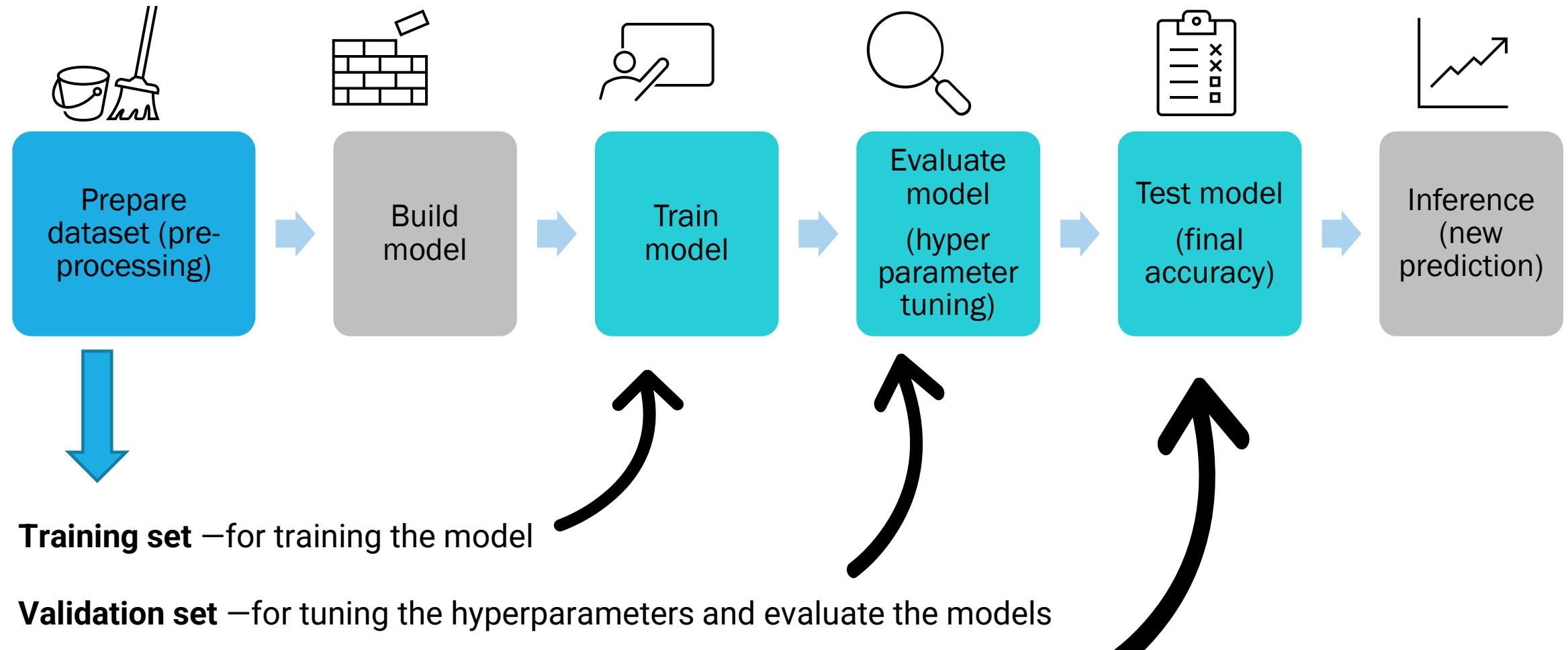
How do we initialise weights?

How can we find the optimum weights so that the error is minimal?

# WORKFLOW



# WORKFLOW



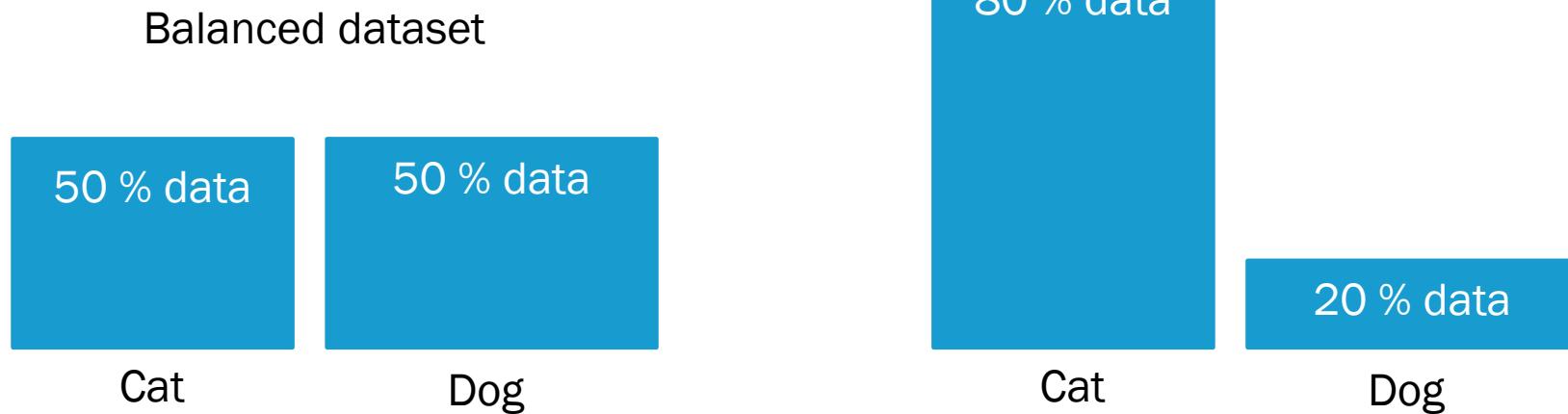
Prepare  
data (pre-  
processing)

# DATA

- Data split

Training set	Validation set	Test set	
80%	10%	10%	(small datasets)
60%	20%	20%	(larger datasets)

- Know your data:



# TRAINING A NEURAL NETWORK

Train model

- Objective: find the optimum weights that will give the lowest **error** [loss]
- Optimisation (Gradient Descent)
  1. Randomly initialise weights
  2. Find how much the weights need to change (by computing the derivates)
  3. Update weights by taking a small step [learning rate]
  4. Repeat until convergence (until there is no improvement) [number of epochs]

Train model

# REFERENCE: LOSS OPTIMISATION

- Identify (train) a set of weights that will give us the minimum error
- Loss functions: Cross entropy, Mean Squared Error, Mean Absolute Error.
- Needs to be differentiable (the derivative of the function exists for all points).

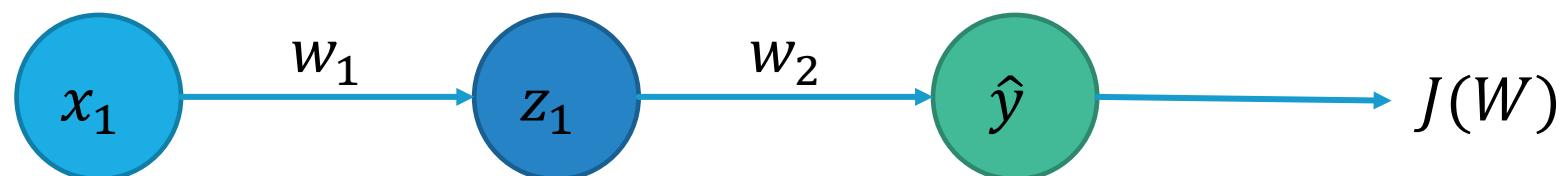
Binary classification	Multiclass classification	Regression
Loss: Cross Entropy	Loss: Cross Entropy	Loss: Mean Squared Error
Activation: Sigmoid	Activation: Softmax	

# BACKPROPAGATION: BACKWARD PROPAGATION OF ERRORS

Train model

- Compute the gradients (computationally expensive!)

How does a small change on the weights affect the final loss  $J(W)$ ?



Watch:  
<https://www.youtube.com/watch?v=IN2XmBhILt4&list=PLblh5JKOoLUIxGDQs4LFFD--41Vzf-ME1&index=4>

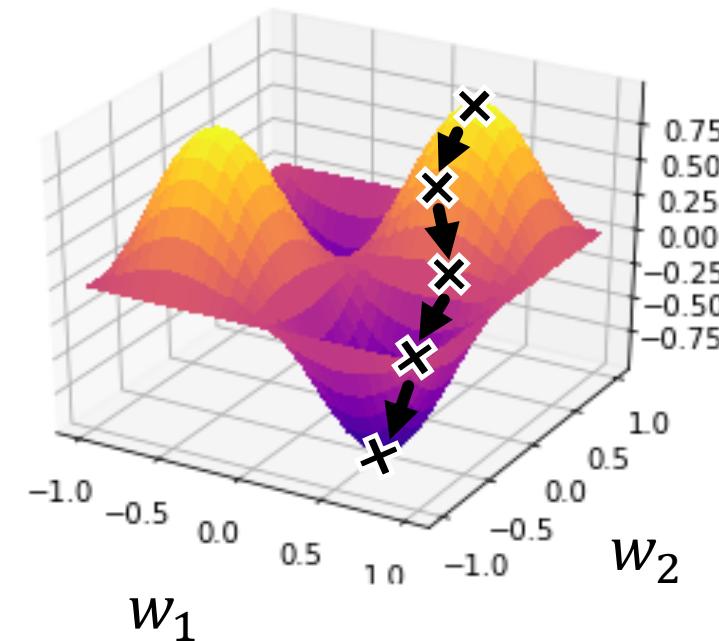
How much a change in  $w_2$  will affect the total error (loss)  $J(W)$ ?

The partial derivative of the loss with respect to  $w_2$

$$\frac{\partial J(W)}{\partial w_2} = \frac{\partial J(W)}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial w_2}$$

How much a change in  $w_1$  will affect the total error (loss)  $J(W)$ ?

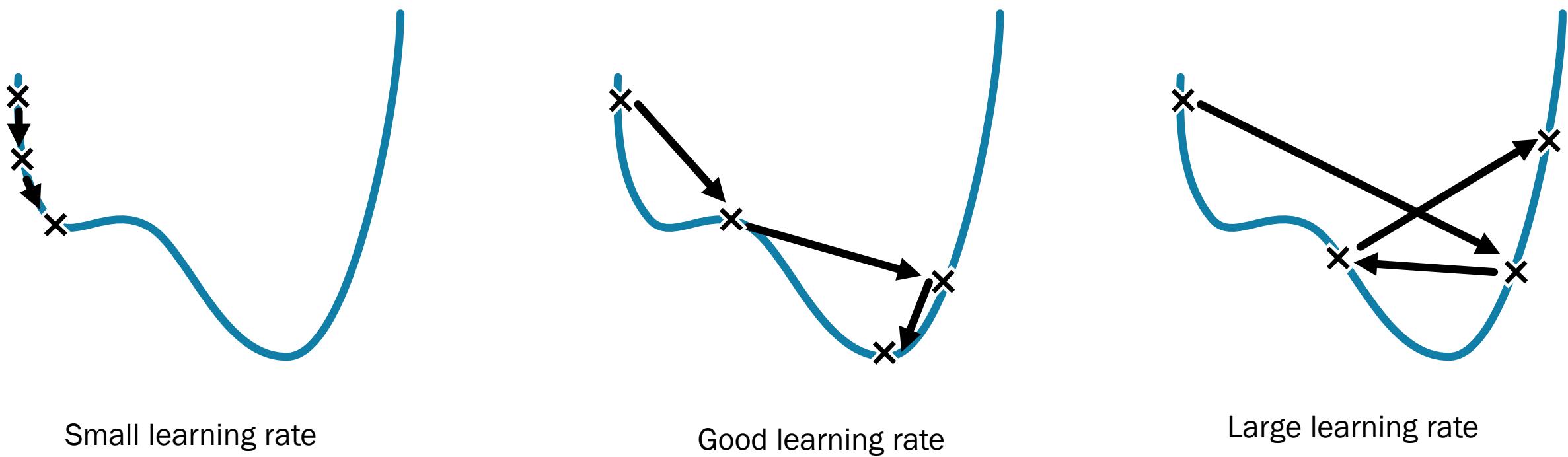
$$\frac{\partial J(W)}{\partial w_1} = \frac{\partial J(W)}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z_1} * \frac{\partial z_1}{\partial w_1}$$



Train model

# LEARNING RATE

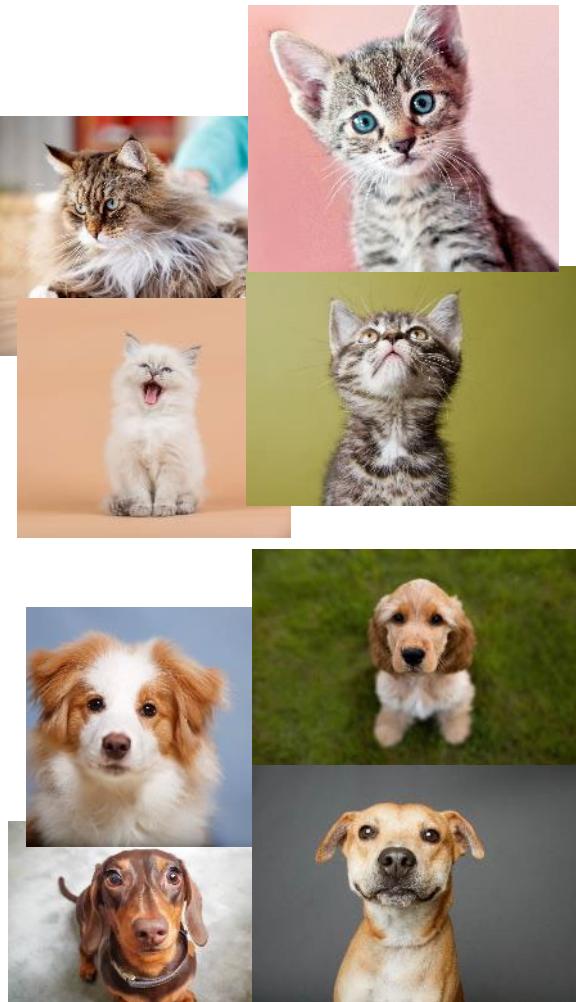
- How much step we take in the gradient?



# FORWARD PROPAGATION AND BACKPROPAGATION

Train model

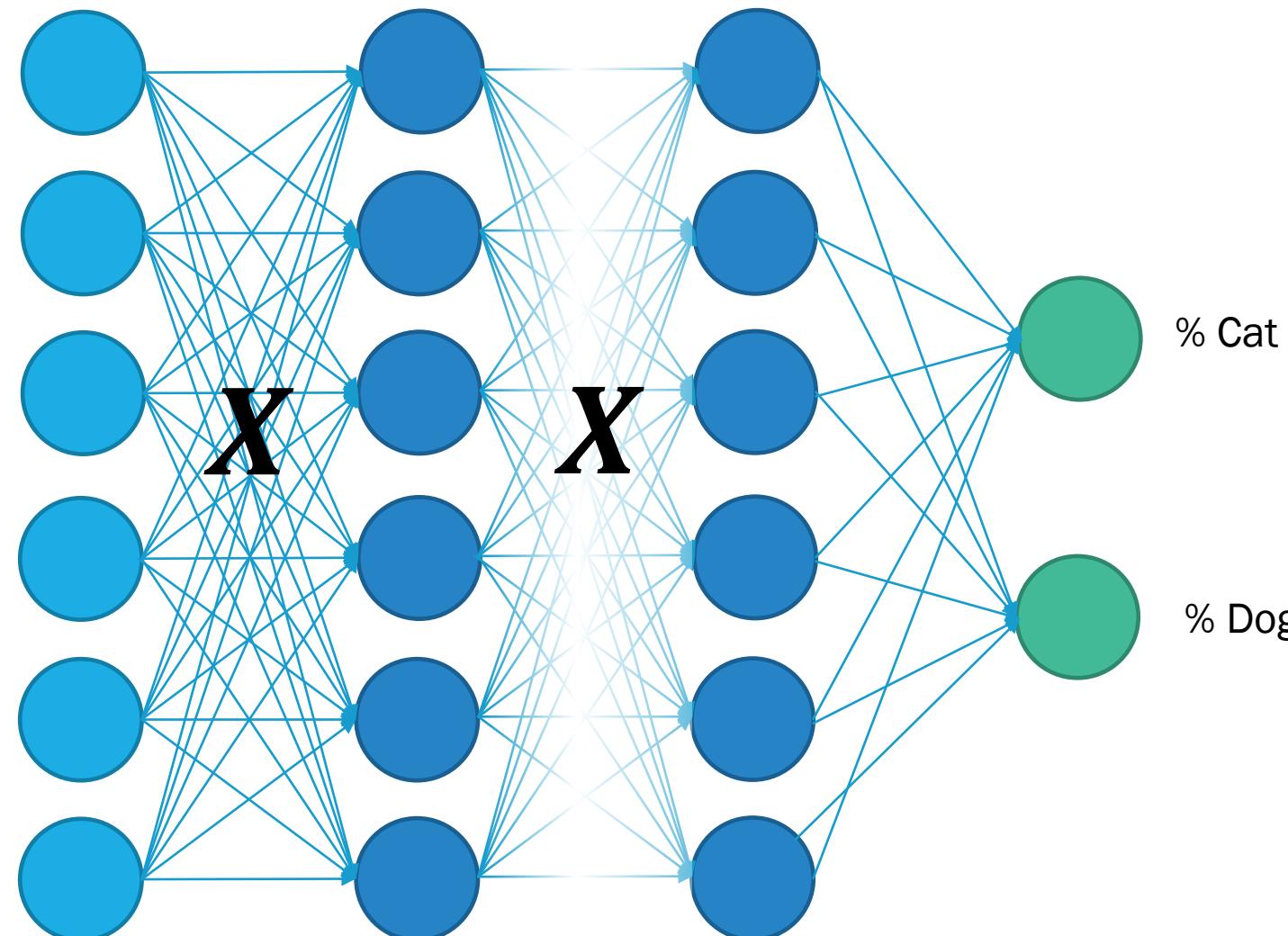
Input  
(flatten)



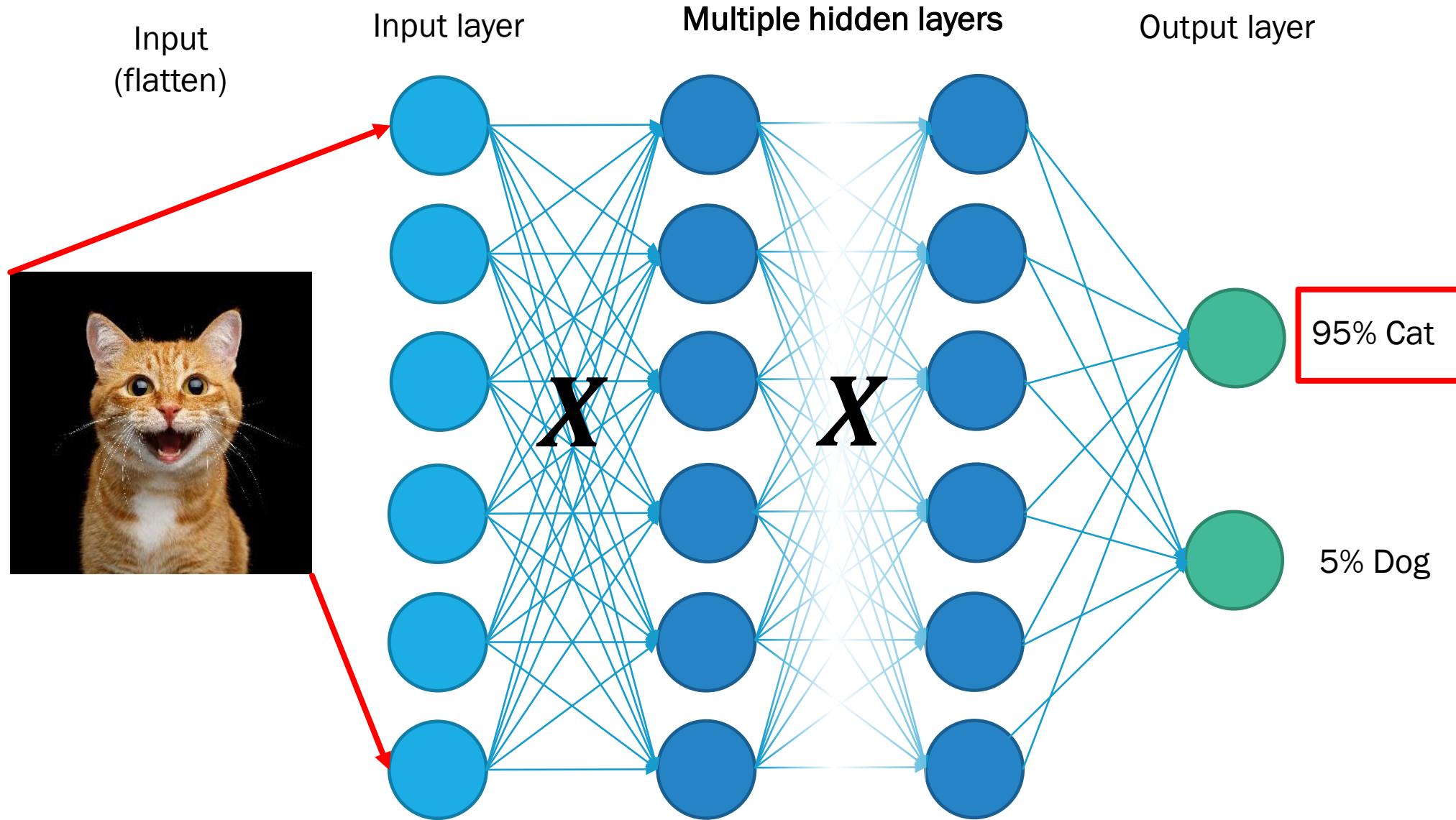
Input layer

Multiple hidden layers

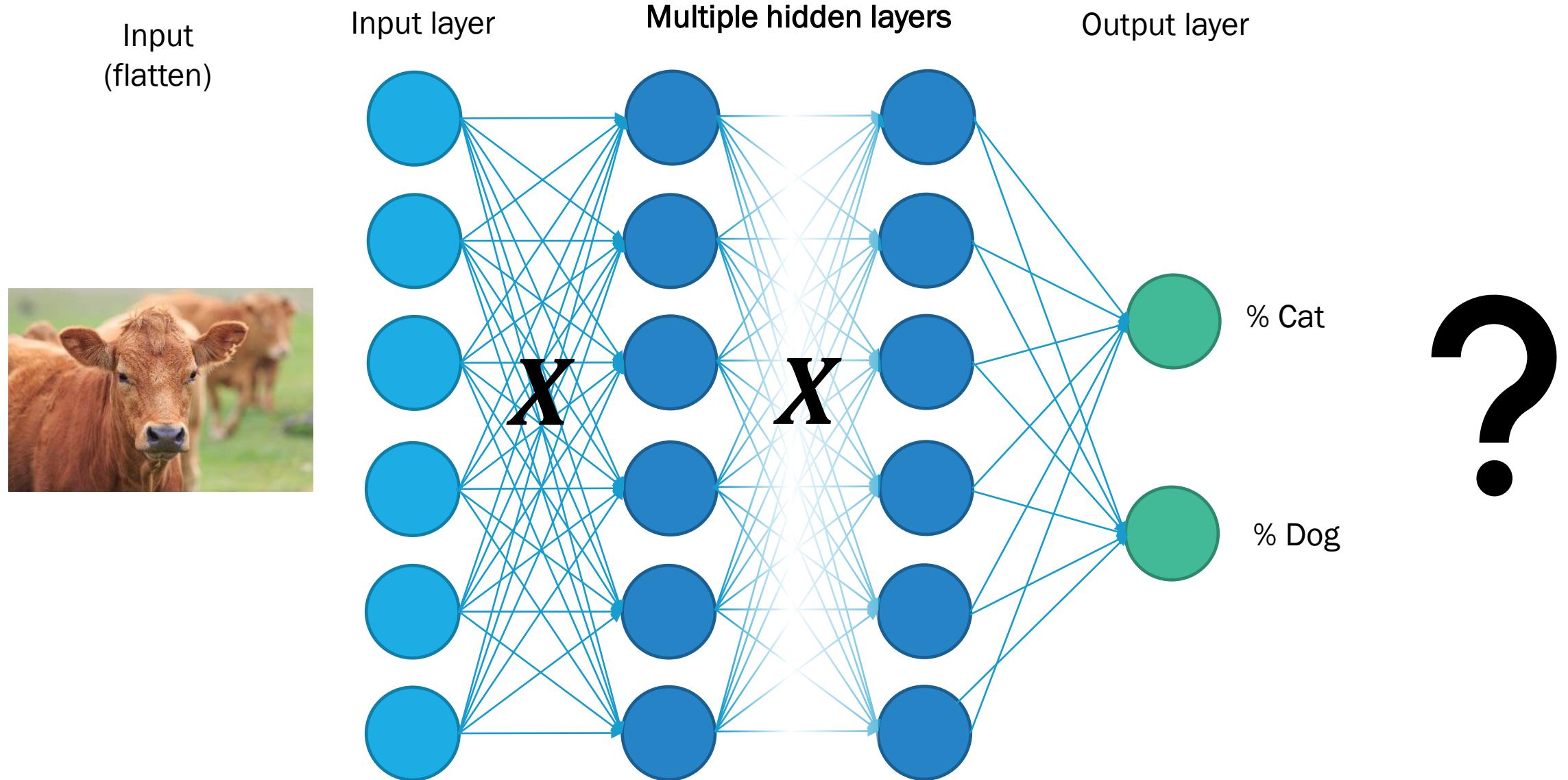
Output layer



# FORWARD PROPAGATION AND BACKPROPAGATION



# FORWARD PROPAGATION AND BACKPROPAGATION

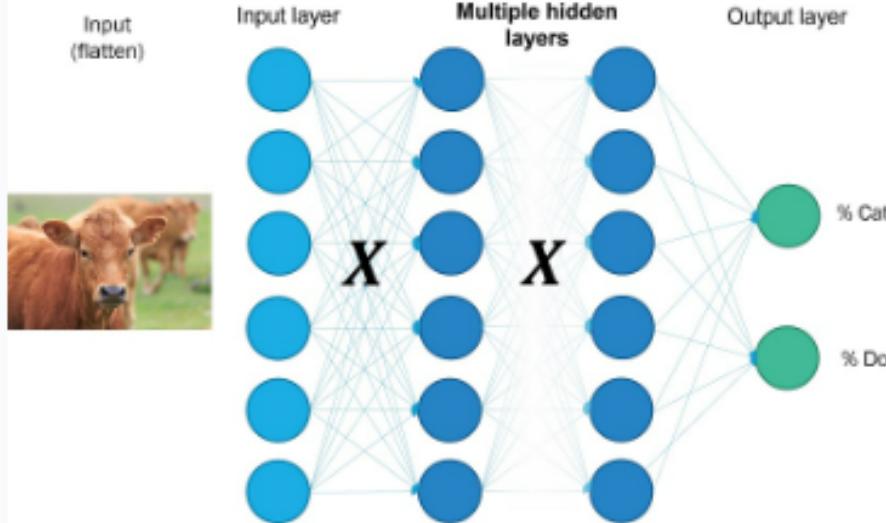


When poll is active, respond at [PollEv.com/esterbonmati](https://PollEv.com/esterbonmati)



## What will the NN predict?

### FORWARD PROPAGATION AND BACKPROPAGATION



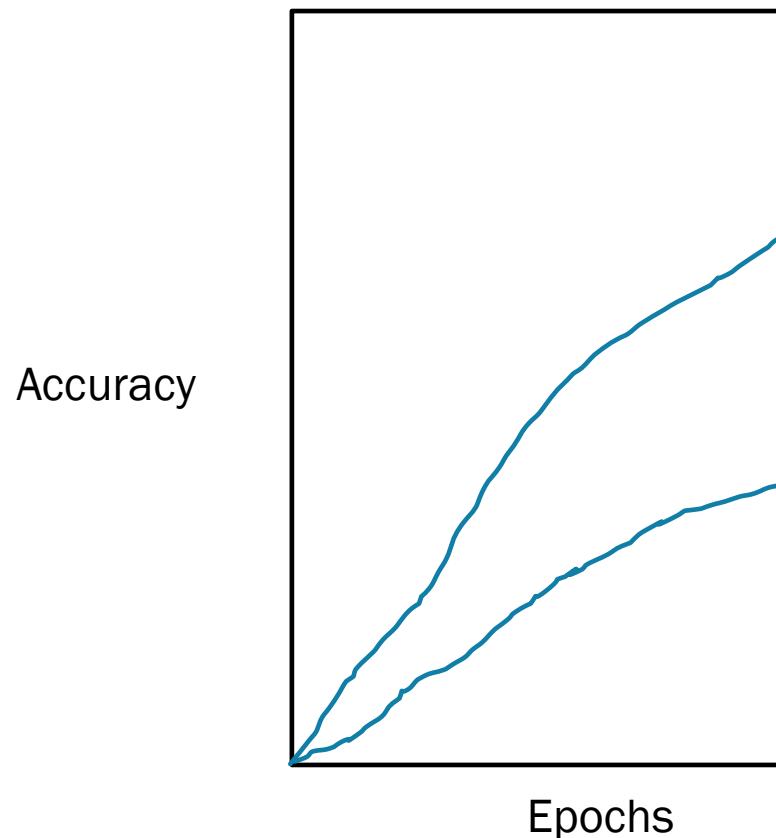
Is a cow

Is a cat

Is a dog

Will not make  
a prediction

# TRAINING



When poll is active, respond at [PollEv.com/esterbonmati](https://PollEv.com/esterbonmati)

**What would you do next?**

Decrease the learning rate

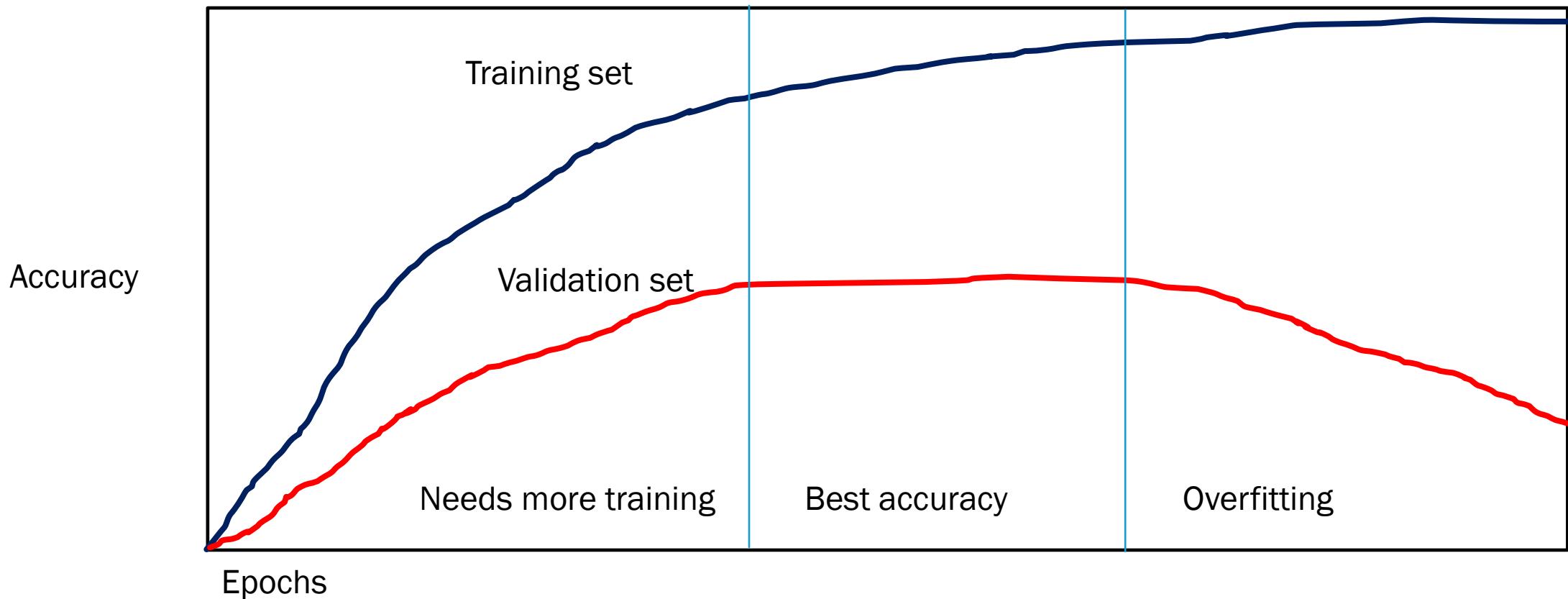
Increase the number of epochs

Powered by  **Poll Everywhere**



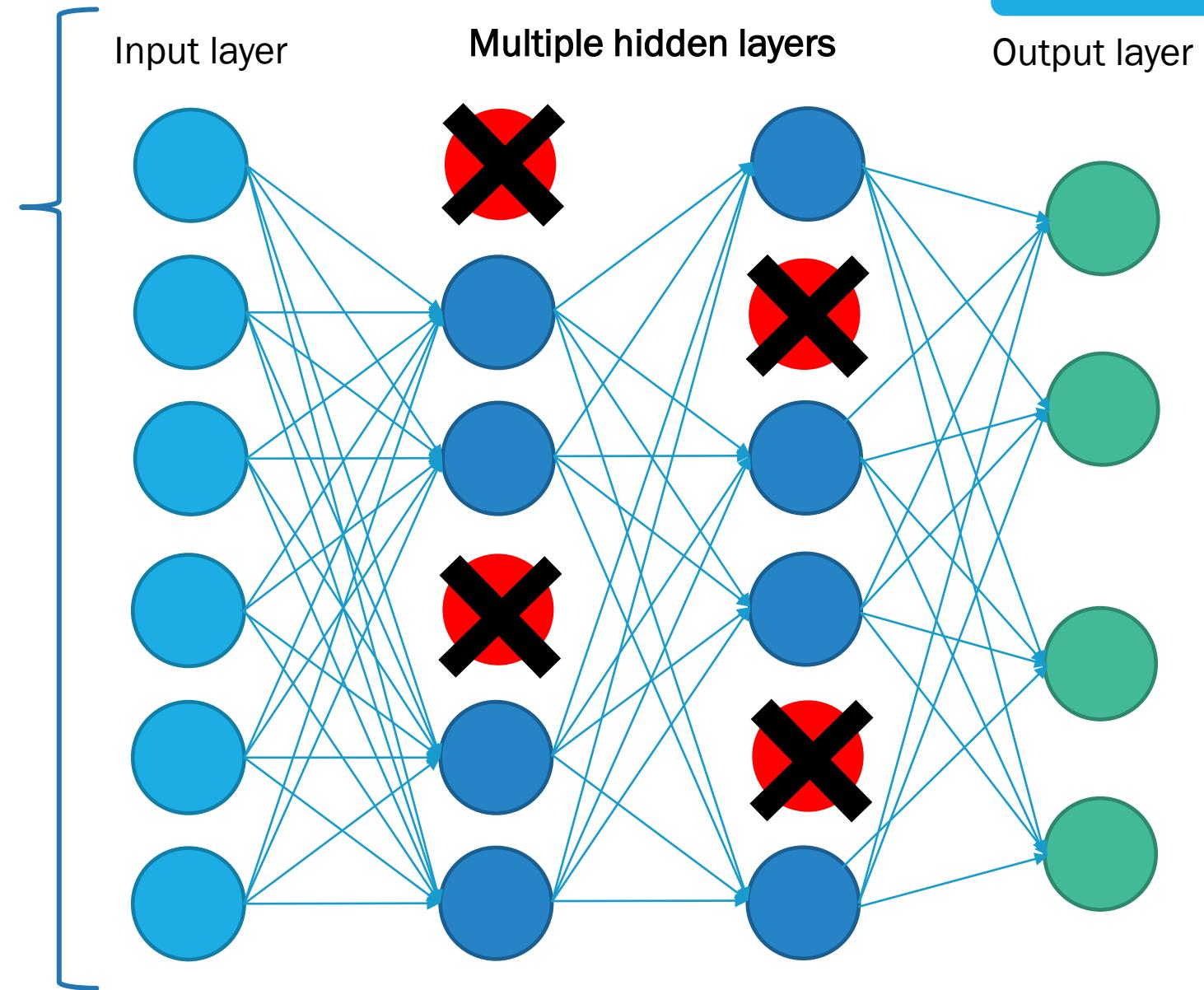
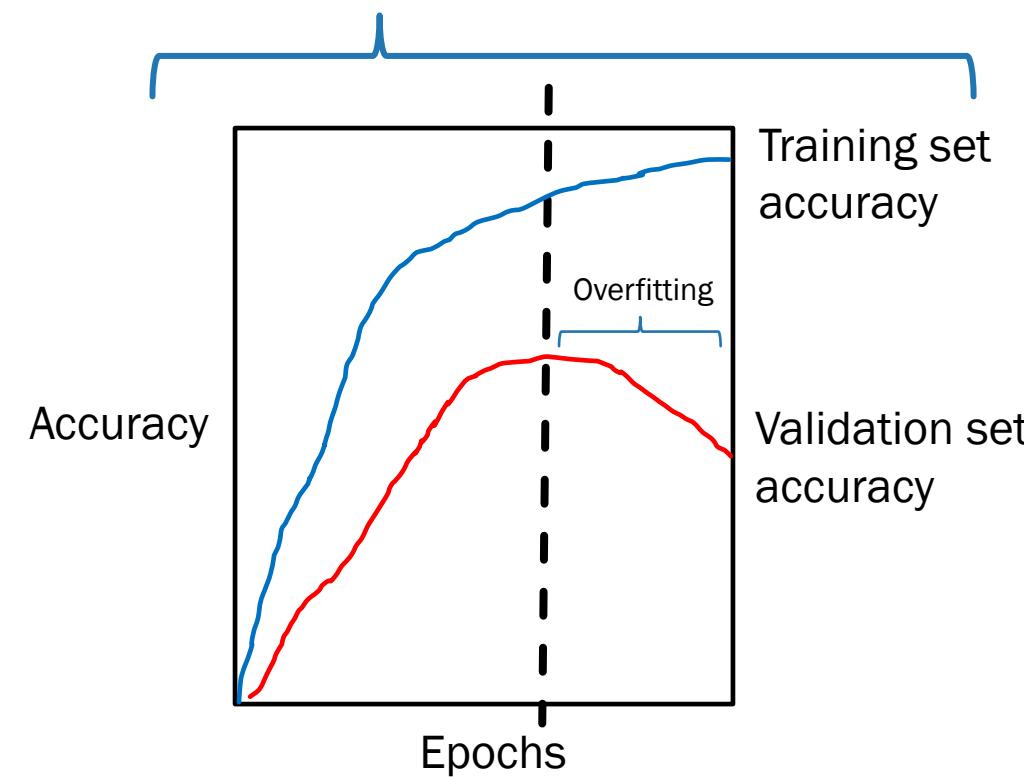
# TRAINING

Train model

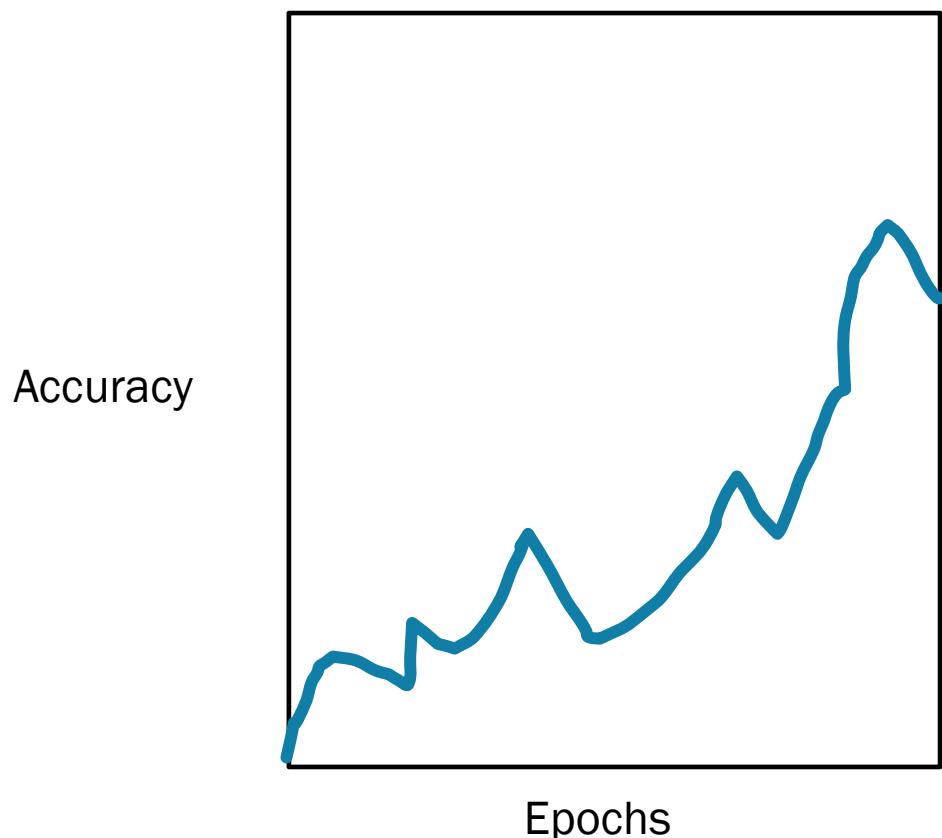


# PROBLEMS: OVERFITTING

- Regularisation
  - Dropout: During training, we set some activation to 0 (usually 50%)
  - Early stopping: stop training before it overfits the data



## WHAT TO TRY NEXT?



When poll is active, respond at [PollEv.com/esterbonmati](https://PollEv.com/esterbonmati)

### What would you do next?

Decrease  
the learning  
rate

Increase the  
learning rate

None of the  
above

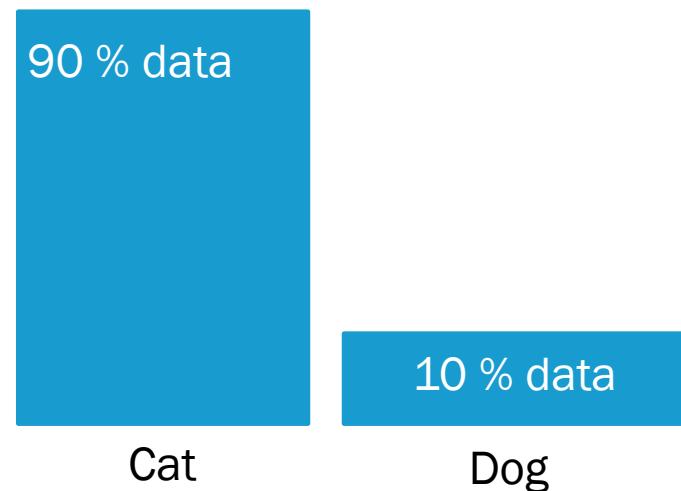
Powered by  Poll Everywhere



# PROBLEMS WITH IMBALANCED DATASETS: ACCURACY

Prepare  
data (pre-  
processing)

Imbalanced dataset (100 samples)



If we predict everything as a cat....

What is for you a  
good accuracy?

$$\text{accuracy} = \frac{\text{number correct samples}}{\text{total number of samples}}$$

$$\text{accuracy} = \frac{90}{100} = \mathbf{0.9 (90\%!!)}$$

This model would have a 90%  
accuracy even if predicts everything  
as a cat.



Data split

Data

Inputs and outputs

Hyper parameters

Hidden layer

Overfitting

Activation function

Weights

Gradient

Forward propagation

Back propagation

Learning rate

Epochs



Concepts to remember

# TENSORFLOW & KERAS

- <https://www.tensorflow.org/tutorials/keras/classification>

The screenshot shows the TensorFlow Core Tutorials page. At the top, there are navigation links: Install, Learn (with a dropdown), More (with a dropdown), a search bar, a language selector (English), and GitHub/Sign in buttons.

The main content area is titled "TensorFlow Core" and "Tutorials". Below the title, there are tabs: Overview, Tutorials (which is selected), Guide, Migrate to TF2, and TF 1 ↗.

The "Tutorials" section contains a sidebar with categories: TensorFlow tutorials, Quickstart for beginners, Quickstart for experts, BEGINNER, ADVANCED, Load and preprocess data, Customization, Distributed training, and Images. The "BEGINNER" category is highlighted with a red dashed box around its content.

The content area for the "Basic image classification" tutorial includes:

- Breadcrumbs: TensorFlow > Learn > TensorFlow Core > Tutorials
- A "Was this helpful?" button with thumbs up and down icons.
- The title: "Basic classification: Classify images of clothing" with a "Edit" icon.
- A "On this page" sidebar with links: Import the Fashion MNIST dataset, Explore the data, Preprocess the data, Build the model, Set up the layers, Compile the model, Train the model, Feed the model, and an ellipsis (...).
- Buttons for "Run in Google Colab", "View source on GitHub", and "Download notebook".
- A descriptive paragraph: "This guide trains a neural network model to classify images of clothing, like sneakers and shirts. It's okay if you don't understand all the details; this is a fast-paced overview of a complete TensorFlow program with the details explained as you go."
- A note at the bottom: "This guide uses `tf.keras`, a high-level API to build and train models in TensorFlow."

# PYTORCH

- [https://pytorch.org/tutorials/beginner/basics/buildmodel\\_tutorial.html](https://pytorch.org/tutorials/beginner/basics/buildmodel_tutorial.html)

PyTorch

Table of Contents

[Run in Microsoft Learn](#) [Run in Google Colab](#) [Download Notebook](#) [View on GitHub](#)

[Learn the Basics](#) || [Quickstart](#) || [Tensors](#) || [Datasets & DataLoaders](#) || [Transforms](#) || **Build Model** || [Autograd](#) || [Optimization](#) || [Save & Load](#)

## BUILD THE NEURAL NETWORK

Neural networks comprise of layers/modules that perform operations on data. The `torch.nn` namespace provides all the building blocks you need to build your own neural network. Every module in PyTorch subclasses the `nn.Module`. A neural network is a module itself that consists of other modules (layers). This nested structure allows for building and managing complex architectures easily.

In the following sections, we'll build a neural network to classify images in the FashionMNIST dataset.

```
import os
import torch
from torch import nn
from torch.utils.data import DataLoader
from torchvision import datasets, transforms
```

# ALSO IN MATLAB

- <https://uk.mathworks.com/help/deeplearning/ug/classification-with-a-2-input-perceptron.html>

MathWorks® Products Solutions Academia Support Community Events Get MATLAB  

## Help Center

Search Help Center Help Center ▾ 

CONTENTS

- « Documentation Home
- « AI, Data Science, and Statistics
- « Deep Learning Toolbox
- « Function Approximation, Clustering, and Control
- « Function Approximation and Clustering
- « Define Shallow Neural Network Architectures

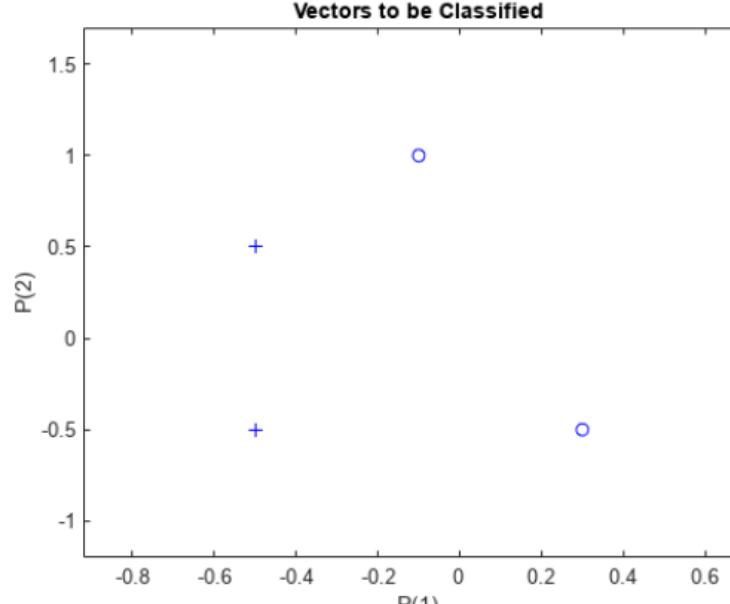
**Classification with a Two-Input Perceptron**

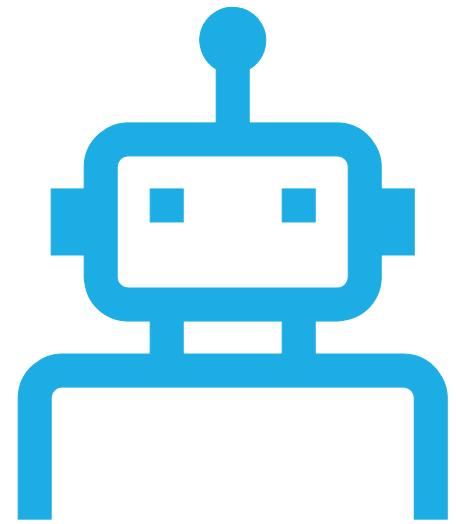
A two-input hard limit neuron is trained to classify four input vectors into two categories.

Each of the four column vectors in X defines a two-element input vectors and a row vector T defines the vector's target categories. We can plot these vectors with PLOTPV.

```
X = [ -0.5 -0.5 +0.3 -0.1; ...  
      -0.5 +0.5 -0.5 +1.0];  
T = [1 1 0 0];  
plotpv(X,T);
```

**Vectors to be Classified**





---

# TUTORIAL: SUPPORTING MATERIAL

# NEURAL NETWORKS: A PRACTICAL EXAMPLE WITH TENSORFLOW

- Example with TensorFlow
- We will work with this example in the Tutorial

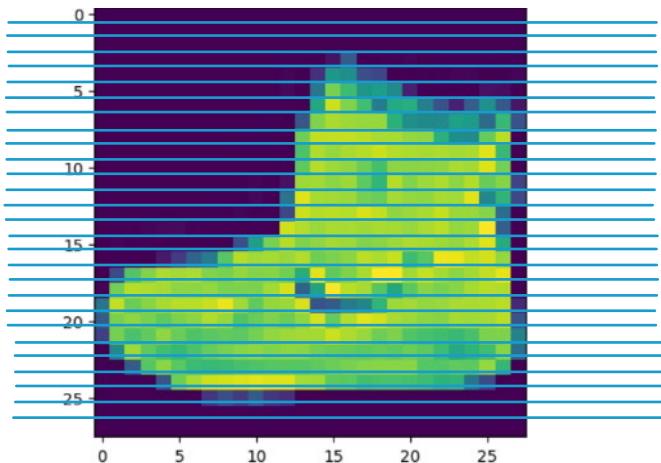
```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np

fashion_mnist = tf.keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

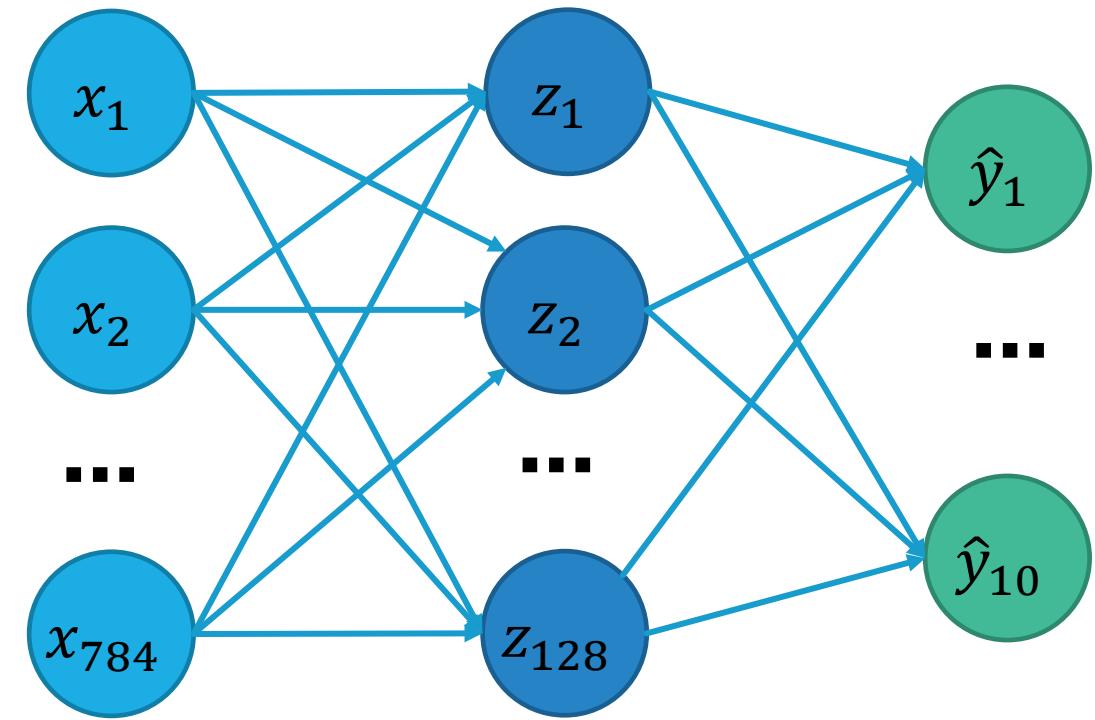


# MODEL CREATION



```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10) ])
```

Output



Input =  $28 \times 28 = 784$

Hidden layer with  
128 units and a ReLu  
activation function

Total trainable  
parameters:  
 $(784 \times 128) + 128 +$   
 $(128 \times 10) + 10 =$   
**101,770**

# SET OPTIMISER AND TRAINING

```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.00001),  
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),  
              metrics=['accuracy'])  
  
epochs = 10  
history = model.fit(train_images, train_labels, validation_data=[test_images, test_labels],  
                     epochs=epochs)
```

Number of epochs (iterations)

Optimiser

Learning rate

Training set

Loss

Validation set

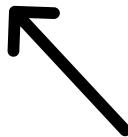


# TEST & INFERENCE (PREDICTION)

```
probability_model = tf.keras.Sequential([model, tf.keras.layers.Softmax() ])  
predictions = probability_model.predict(test_images)
```



Array of predicted values for each test image



Make  
prediction



Images to predict



Softmax to normalise  
the logits (output) into  
probabilities



< Untitled folder

Moderate

Visual settings

Edit



When poll is active, respond at [PollEv.com/esterbonmati](https://PollEv.com/esterbonmati)



## Q&A

Top

No responses received yet. They will appear here...

# FEEDBACK ON THIS LECTURE

Activities < Edit < >

Respond at [pollev.com/esterbonmati](http://pollev.com/esterbonmati)

### Feedback on this lecture

0 done

0 underway

Powered by  Poll Everywhere

# LEARNING OUTCOMES OF THIS SESSION

## Perceptron

- Understand the fundamental concept
- Understand inputs and outputs
- Understand weights
- Forward propagation
- Understand activation functions
- Be able to calculate the output of a perceptron

## Neural Networks

- Understand the concept
- Be able to list the name of the different layers
- Be able to describe how neural networks work in your own words
- Understand problems with data
- Understand hyperparameters
- Be able to evaluate performance and calculate accuracy

---

## ACKNOWLEDGEMENTS & NOTES

Acknowledgements: This lecture was inspired by the following material:

- <http://introtodeeplearning.com/>: **MIT Introduction to Deep Learning | 6.S191**
- Neural Networks (Applied AI 21/22 – Dr. Artie Basukoski)

Notes:

- We have some ‘guest’ students attending this lecture.
- Anne-Gaelle Colom (Assistant Head of School) will observe the second hour of the tutorial.