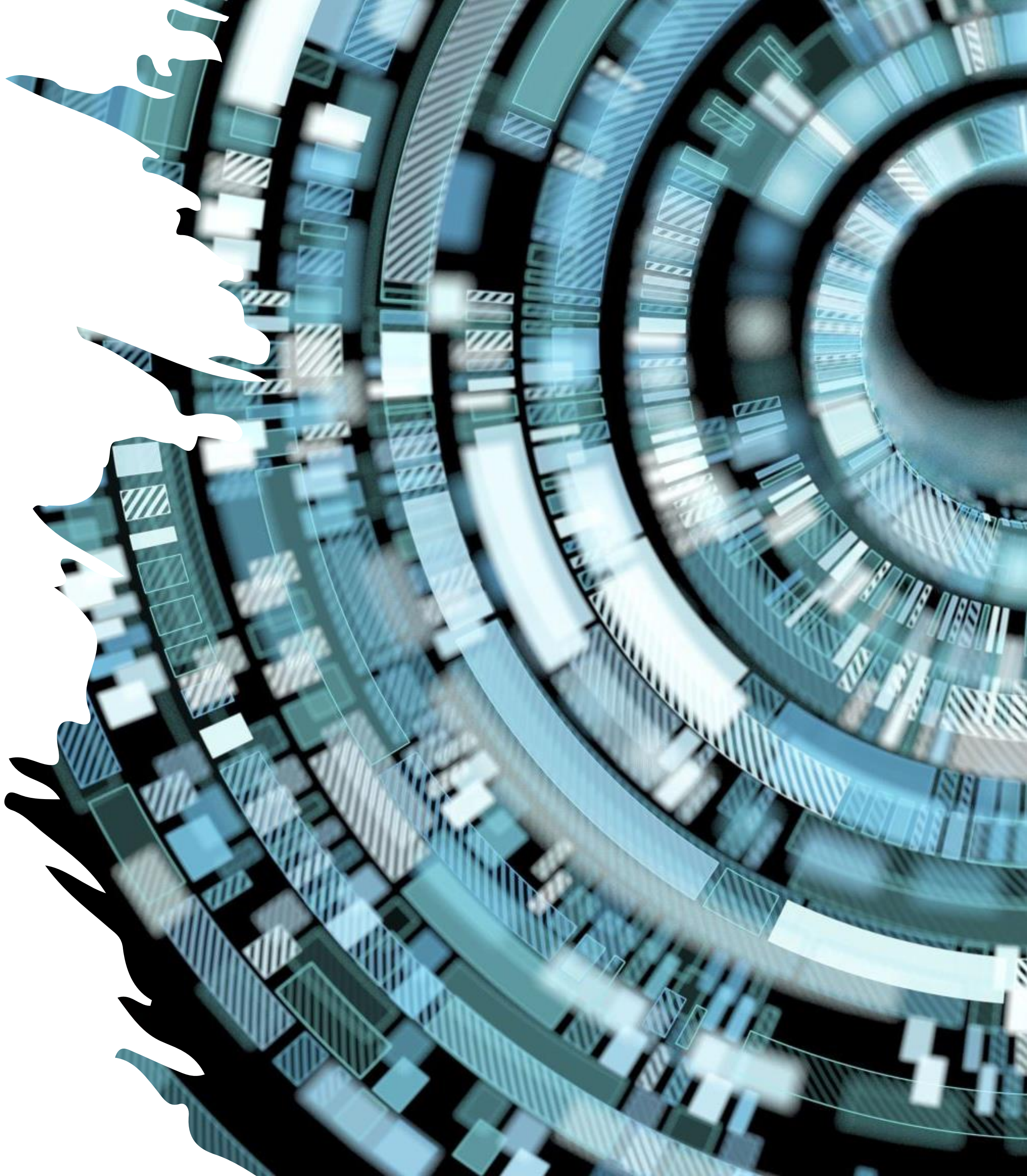
CSET335L (Deep Learning)

Deep learning

Dr. Rohit Kumar Kaliyar

Outline

- Introduction to Deep Learning (DL)
- Why deep learning?
- Machine learning (ML)
    - Features
    - Weights
    - Artificial Neural Networks (ANN)
    - Loss function
    - Cost function
- History of Deep Learning



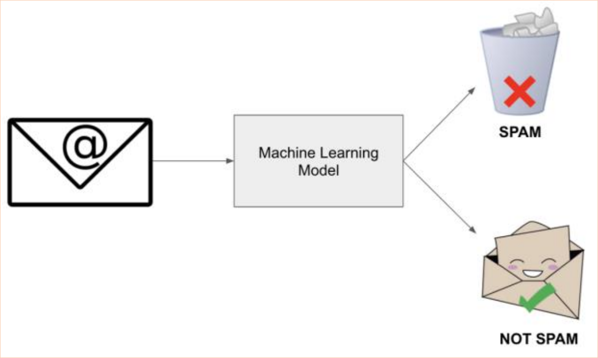
What is Deep learning?

Artificial Intelligence

Any technique that enables computers to mimic human behavior

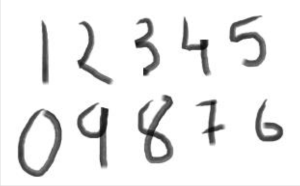
Machine Learning

Ability to program without being explicitly programmed

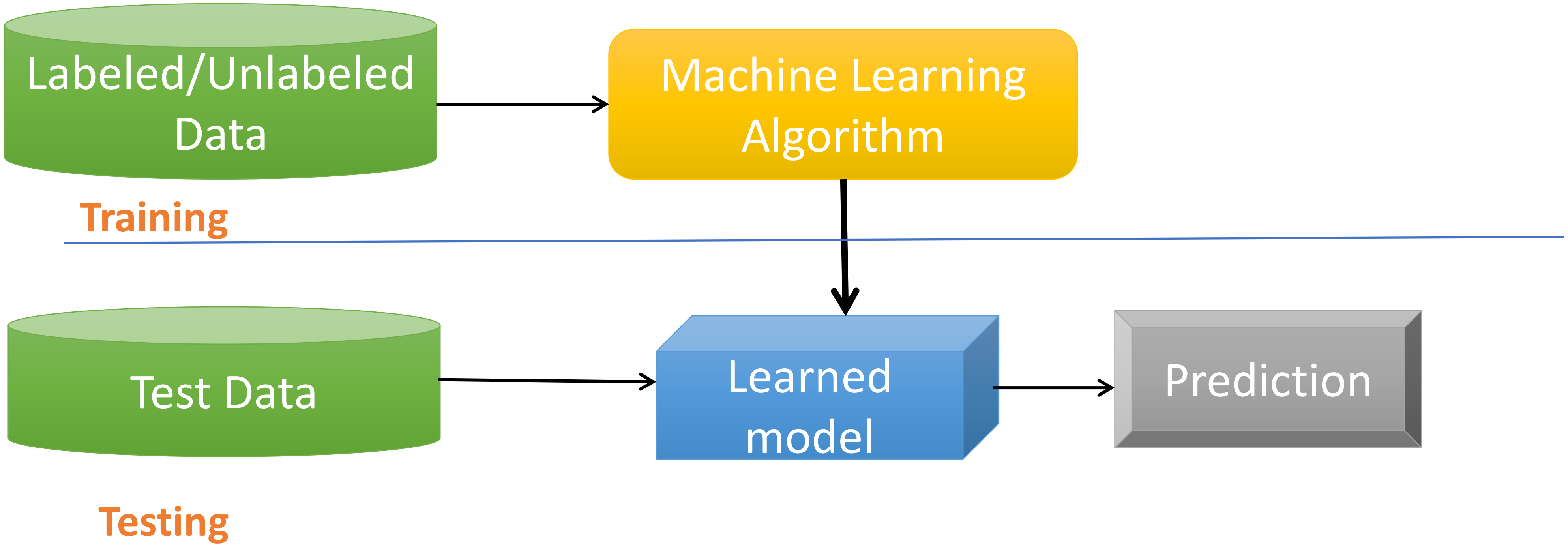


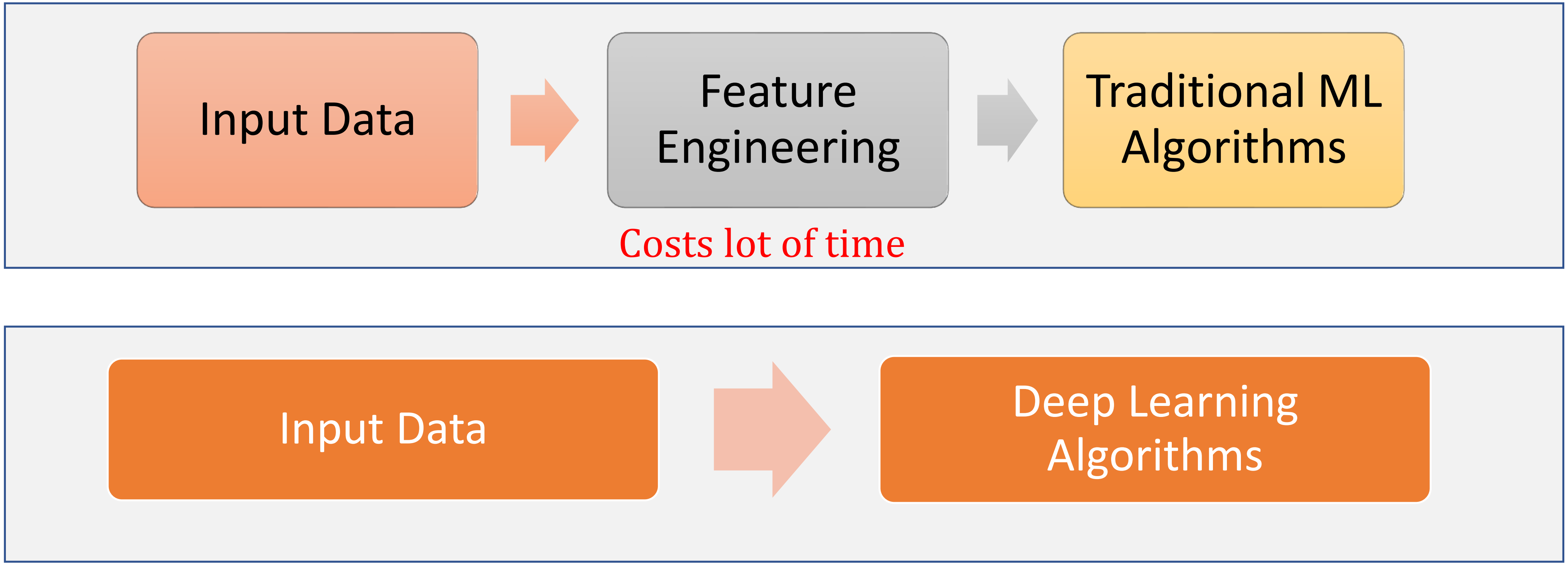
Deep Learning

Extract patterns from data using neural networks



ML gives computers the ability to learn without being explicitly programmed




Input Data
Feature Engineering
Traditional ML Algorithms
Costs lot of time
Input Data
Deep Learning Algorithms


Low Level Features
Mid Level Features
High Level
Lines & Edges
Eyes & Nose & Ears
Facial Structure
Lear
• Deep lear
multiple laye
hierarchical f
can apply ma

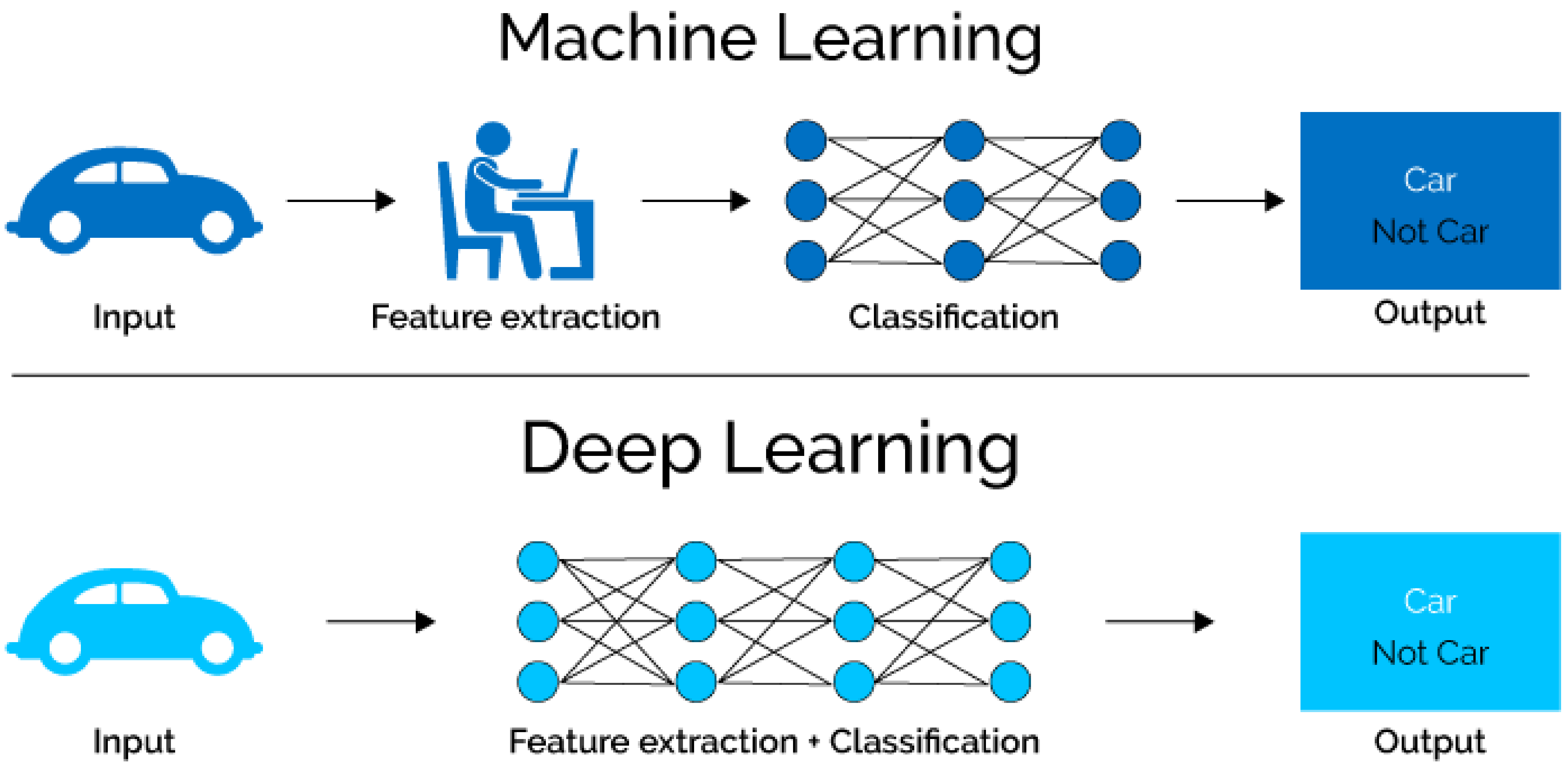
A subfield of machine learning where data representation is learned

Deep learning algorithms attempt to learn (multiple levels of) representation by using a hierarchy of multiple layers

Require tons of data before it begins to understand it and respond in useful ways.

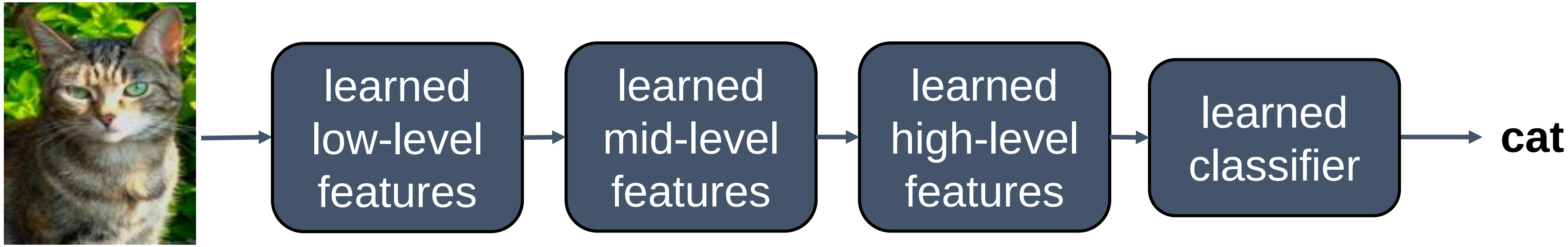
Manually designed features are often over-specified, incomplete, time-consuming, and non-scalable.

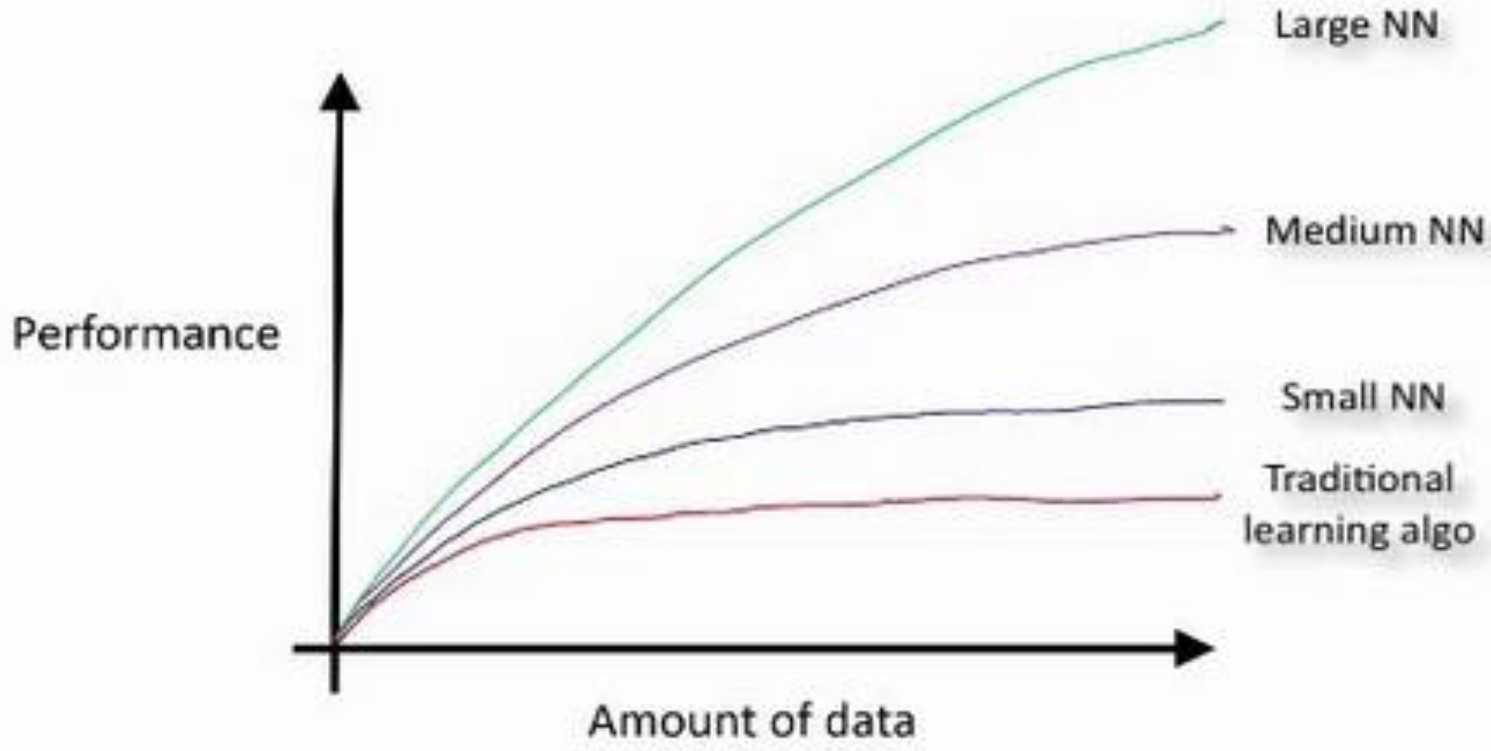
Deep learning learns the features directly from data


Machine Learning
Car
Not Car
Input
Feature extraction
Classification
Output
Deep Learning
Car
Not Car
Input
Feature extraction + Classification
Output

“Traditional” machine learning:

Deep, “end-to-end” learning:




Large NN
Medium NN
Performance
Small NN
Traditional learning algo
Amount of data

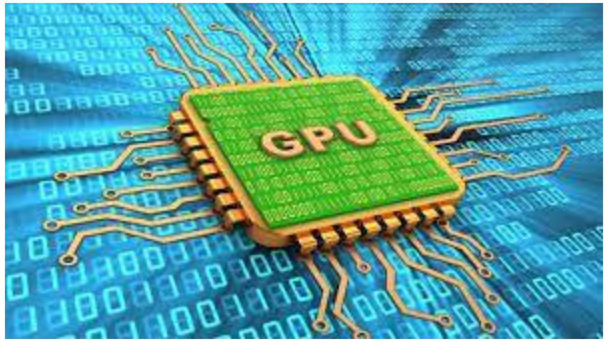
Neural networks (NN) were first proposed in 1940’s and many different architectures of NN have been proposed since then,

So why is the sudden explosion of use of Deep learning now?

Big Data

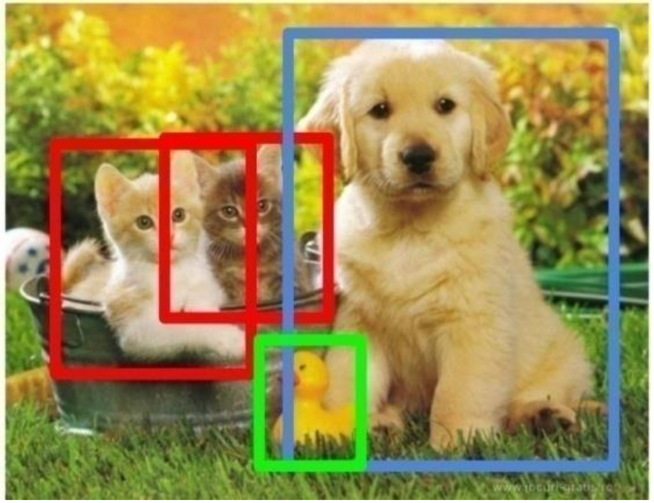
Hardware
GPUs

Software
New models
and toolboxes



Classification

Classification + Localization

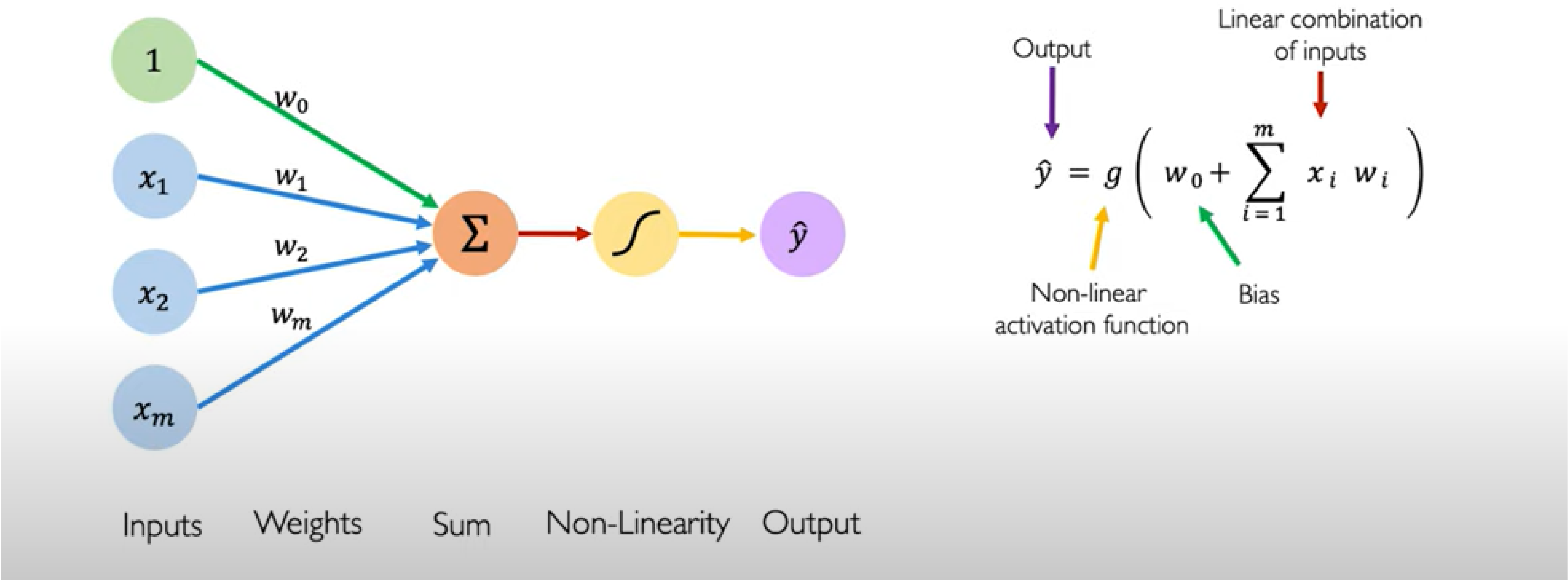


Object Detection

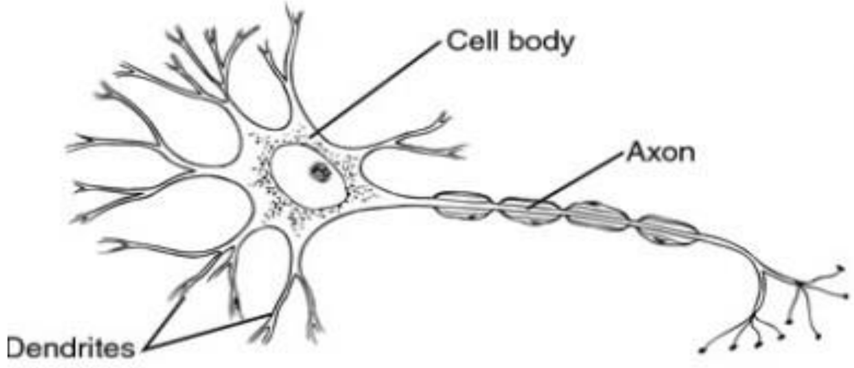
Speech Recognition
Voice assistant

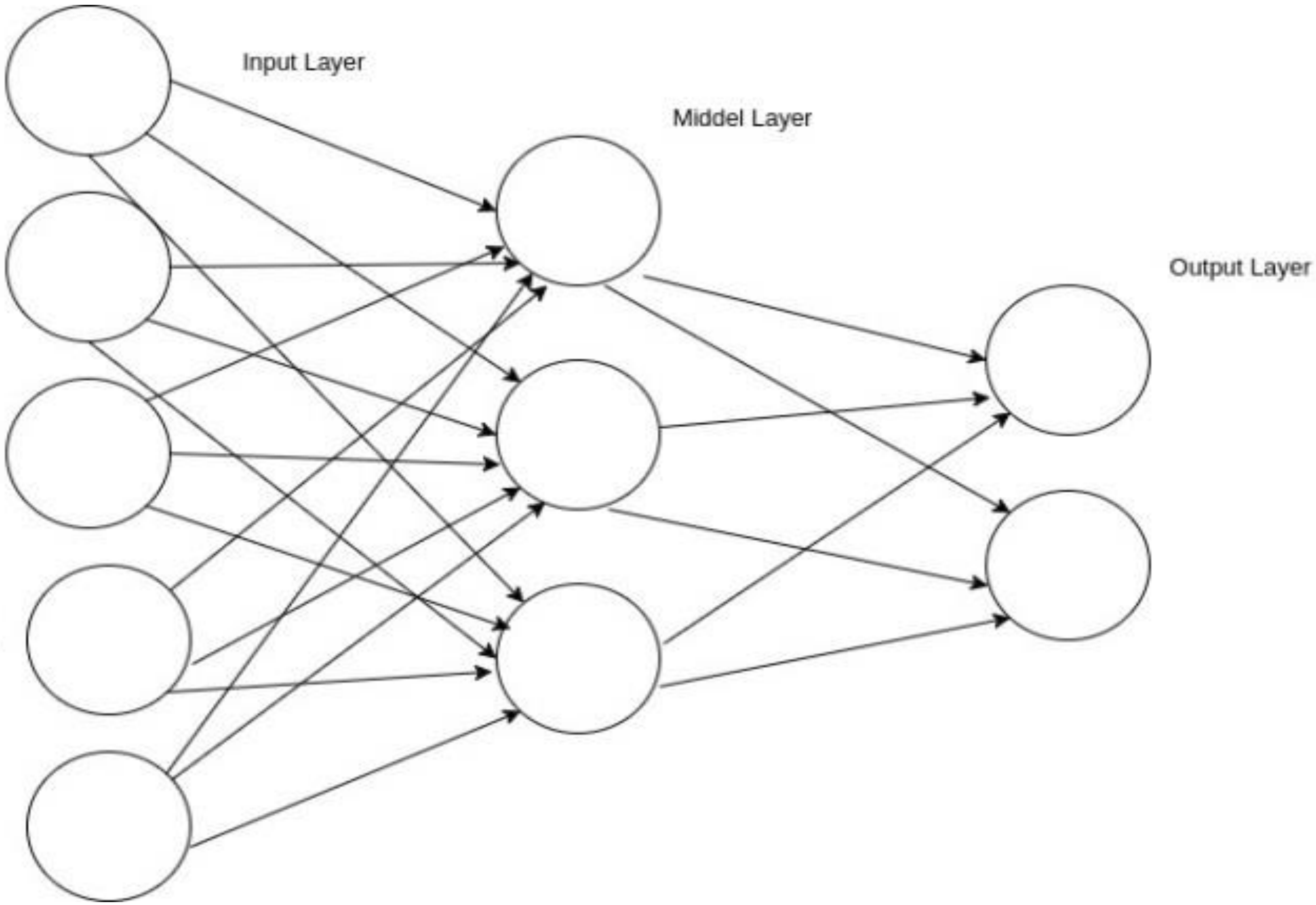
Self-driving cars

Perceptron (Single layer neural network) is the structural building block of deep learning models

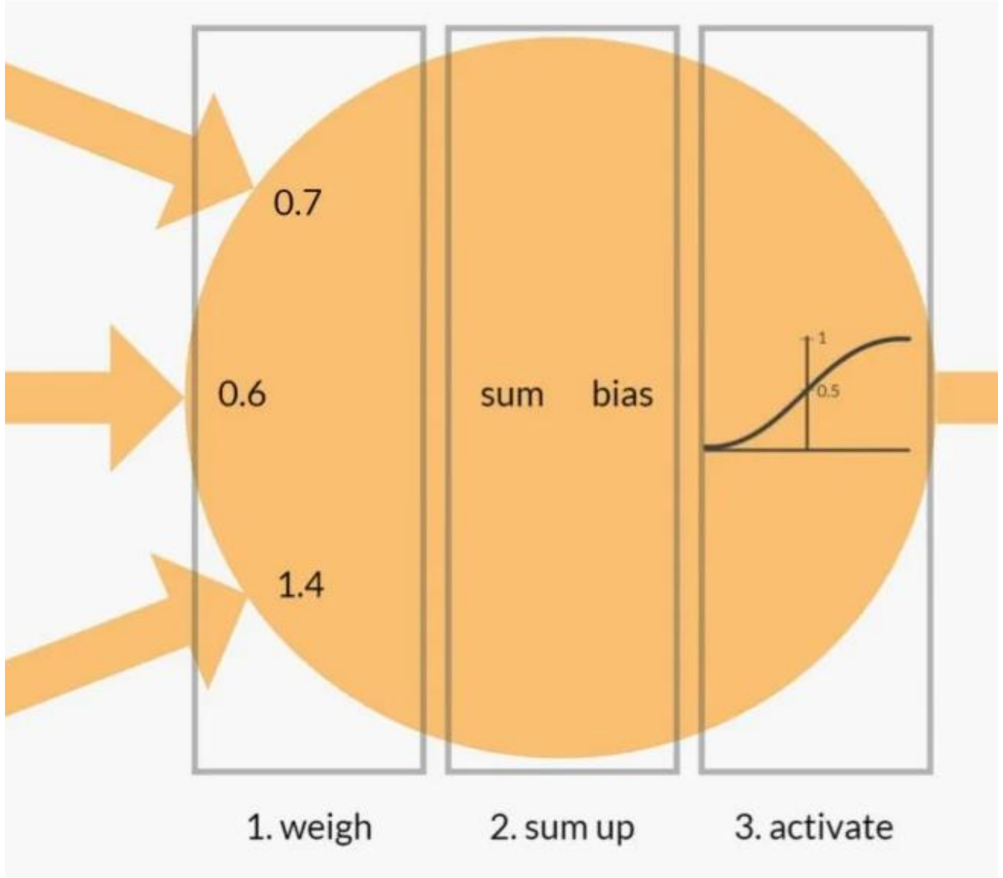


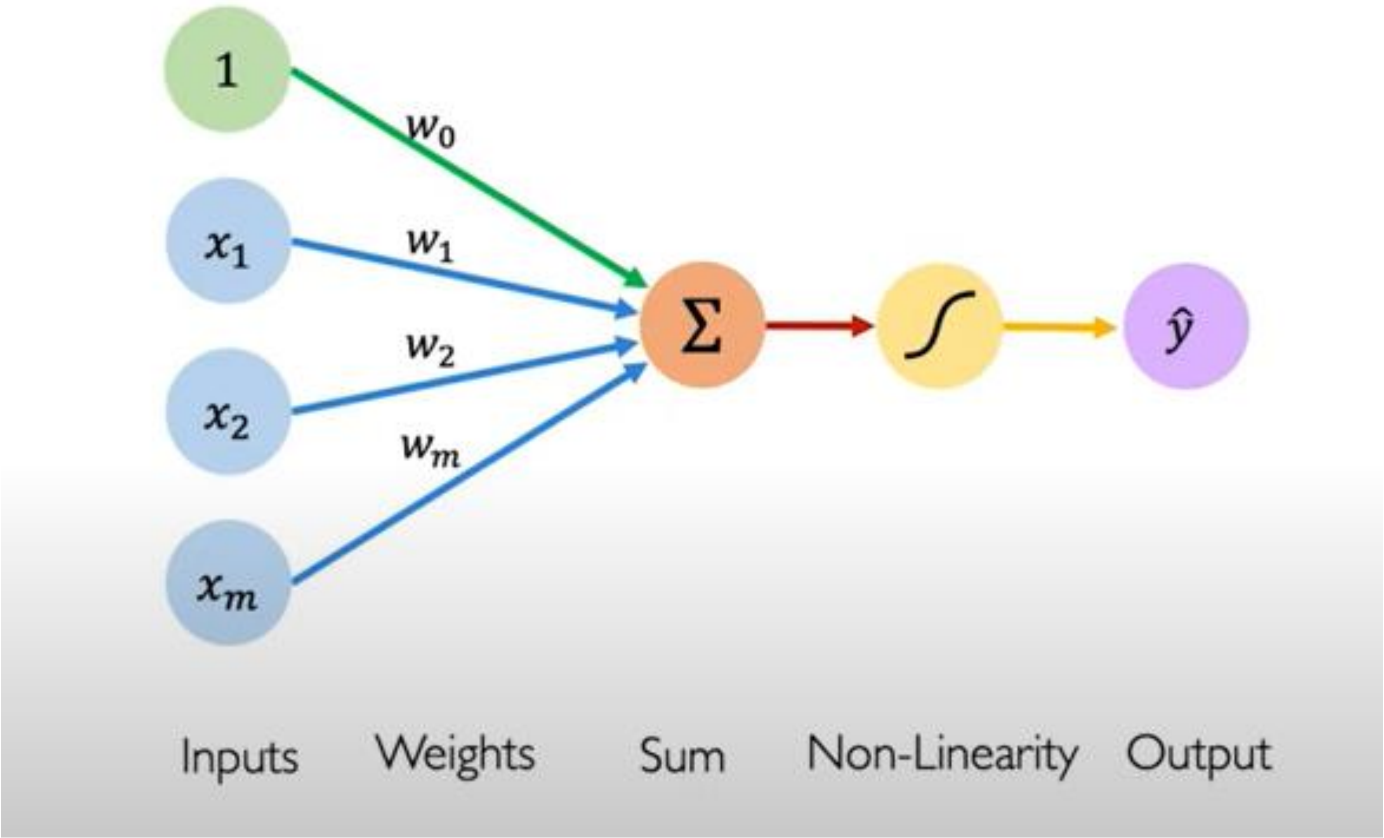
NN


Cell body
Axon
Dendrites


Input Layer
Middel Layer
Output Layer

Neuron forward pass





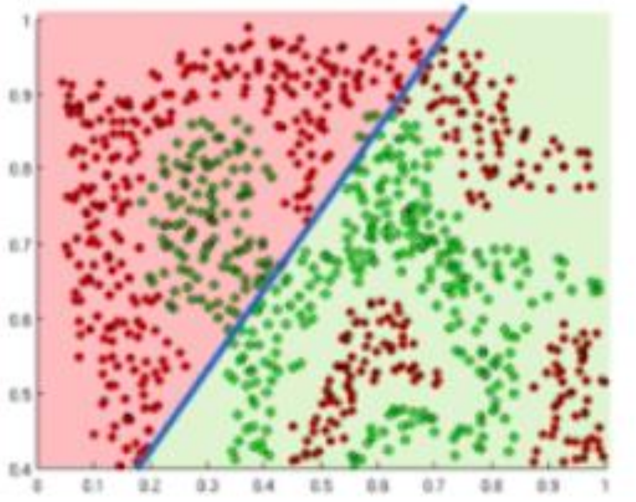
$$\hat{y} = g(w_0 + \boldsymbol{X}^T \boldsymbol{W})$$

where: $\boldsymbol{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$ and $\boldsymbol{W} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$

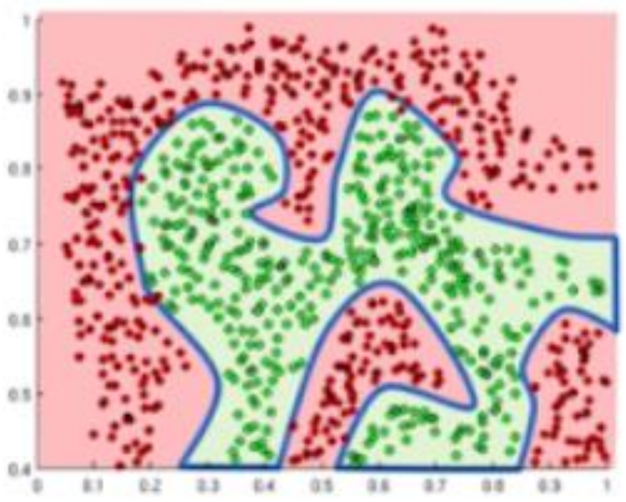
- A neural network is comprised of layers of nodes and learns to map examples of inputs to outputs.
- For a given node, the inputs are multiplied by the weights in a node and summed together. This value is then transformed via an activation function and defines the specific output or “activation” of the node.

- The simplest activation function is referred to as **linear activation**, where no transform is applied at all.
- A network comprised of only linear activation functions is very easy to train but cannot learn complex mapping functions. Linear activation functions are still used in the output layer for networks that predict a quantity (e.g., regression problems).
- **Nonlinear activation functions** are preferred as they allow the nodes to learn more complex structures in the data.

- Activation function introduce non-linearity in the network
- Non-linear functions approximate complex functions that make neural networks extremely powerful

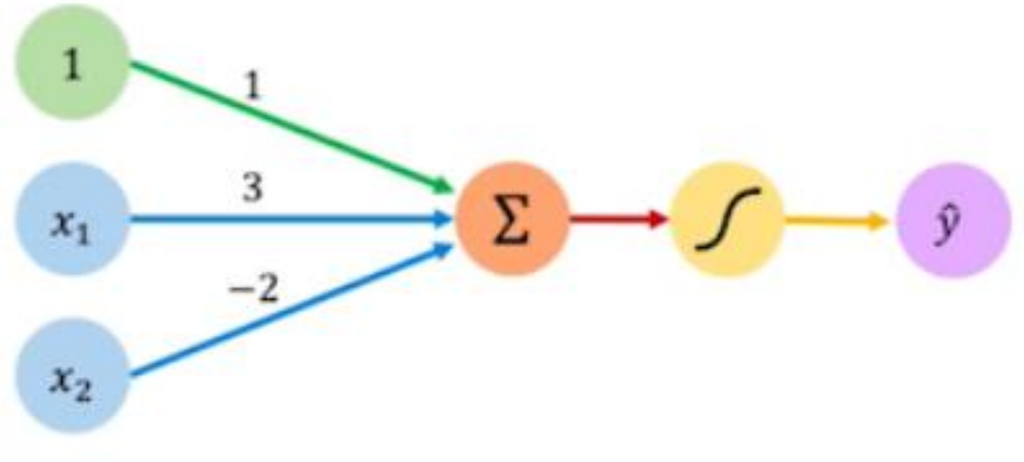


Linear activation functions produce linear decisions no matter the network size



Non-linearities allow us to approximate arbitrarily complex functions

Example: Suppose we have a trained network with weights W and two inputs x1 and x2.

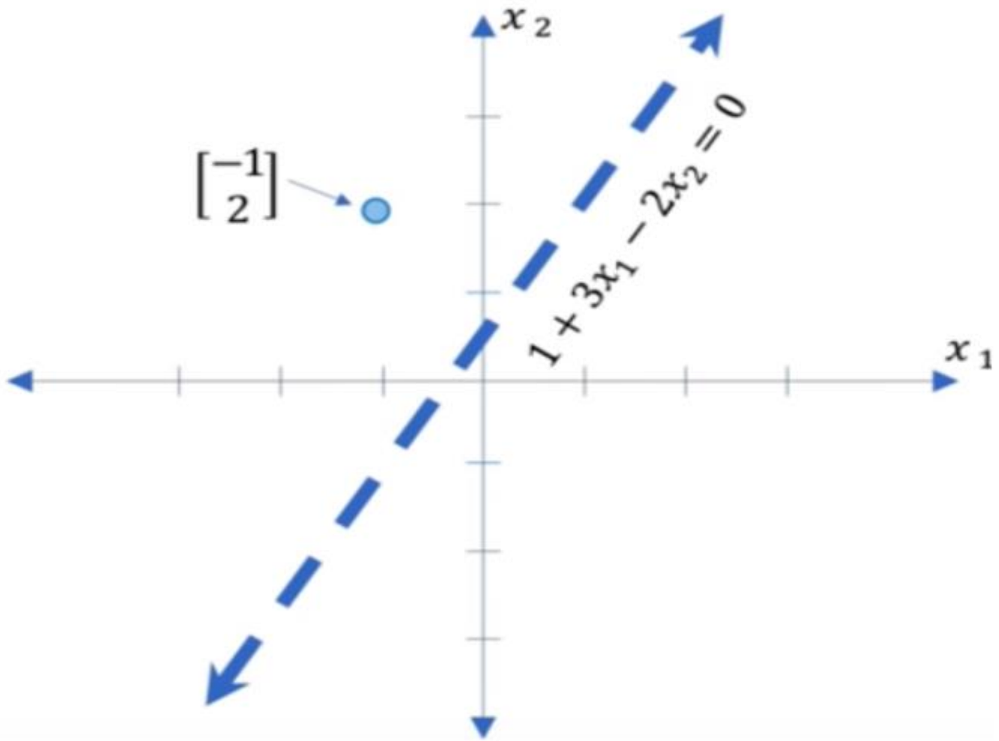


We have: $w_0 = 1$ and $\boldsymbol{W} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$

$$\hat{y} = g(w_0 + \boldsymbol{X}^T \boldsymbol{W})$$
$$= g\left(1 + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 3 \\ -2 \end{bmatrix}\right)$$
$$\hat{y} = g(\underbrace{1 + 3x_1 - 2x_2}_{\text{This is just a line in 2D!}})$$

Till the summation step, if we feed x1=-1 and x2 = 2, we get -6.

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



Assume we have input: $\boldsymbol{X} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$

$$\begin{aligned} \hat{y} &= g\big(1 + (3 * -1) - (2 * 2)\big) \\ &= g(-6) \approx 0.002 \end{aligned}$$

- Unit step function (Output a 0 or 1)
- Sigmoid / Logistic function (S-shaped | Output value between 0 and 1)
- Hyperbolic tangent (tanh) function (Output value between -1 to 1)
- Rectified linear (ReLU) function (Output value between 0 to infinity)

Sigmoid function: Non-zero centered
Vanishing gradient problem

Tanh function: Zero-centered
Vanishing gradient problem

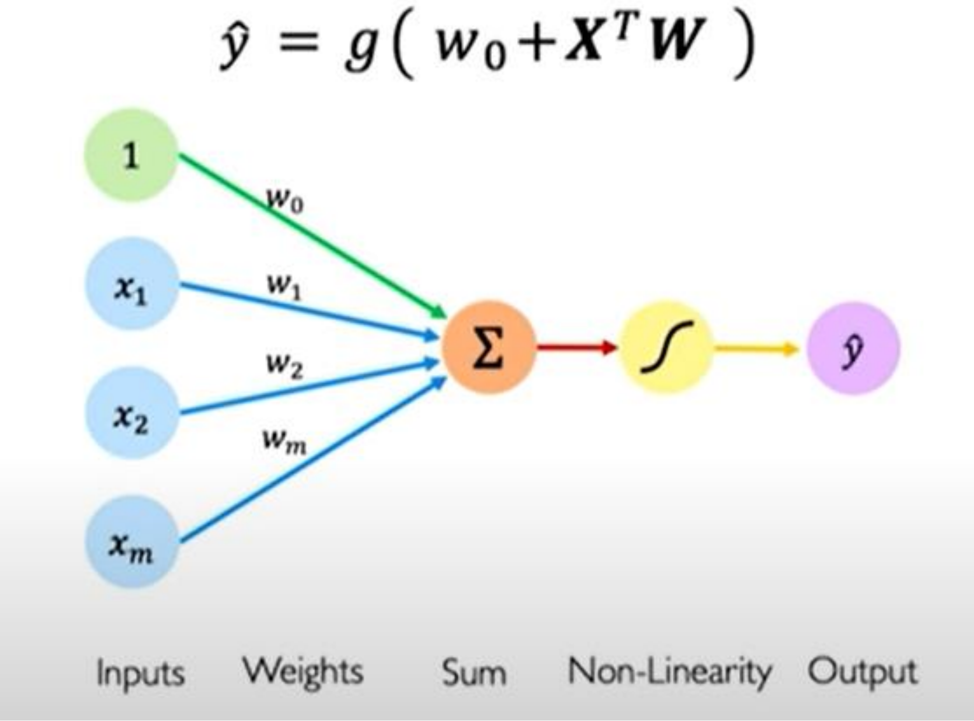
ReLu function: Non-zero centered
Vanishing gradient problem does not exist

More reading on activation functions:
https://www.analyticsvidhya.com/blog/2021/04/activation-functions-and-their-derivatives-a-quick-complete-guide/

For a long time till early 1990s, **sigmoid function** was the **default activation** used on neural networks.

In the later 1990s and through the 2000s, the **tanh was preferred over the sigmoid** function as it makes training easier and had better predictive performance.

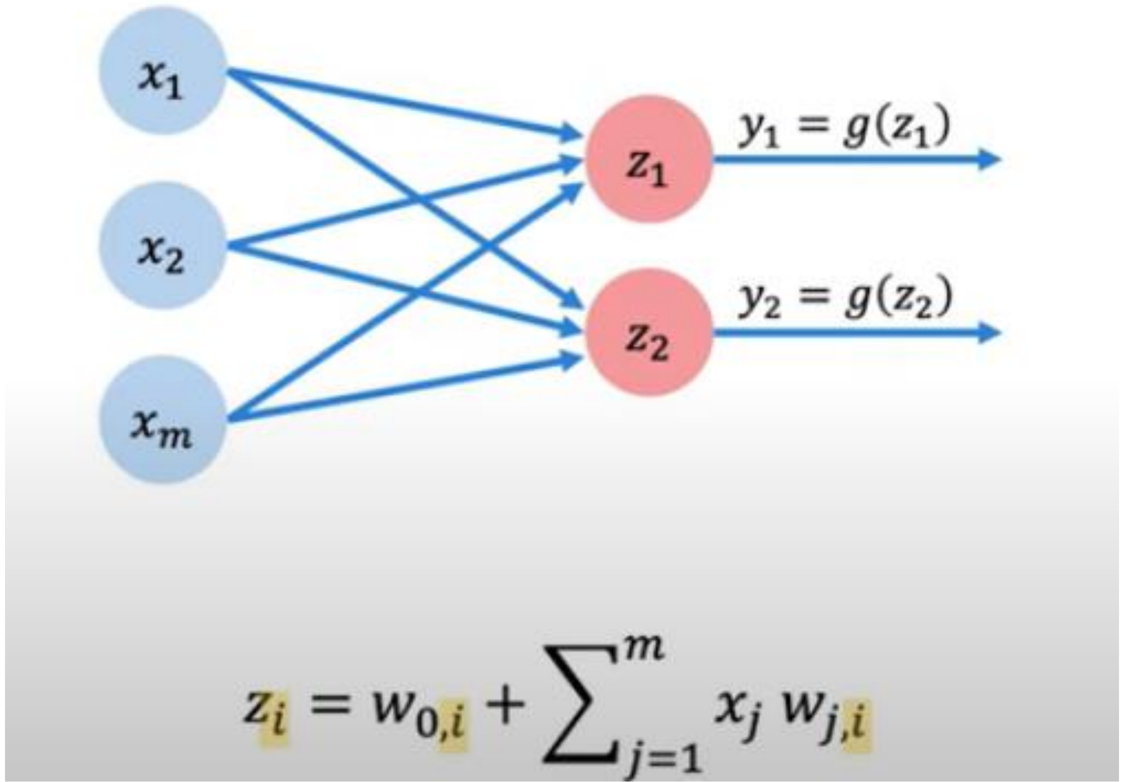
Later, **Relu is mostly used for deep learning models** because of its ability to map complex relationships



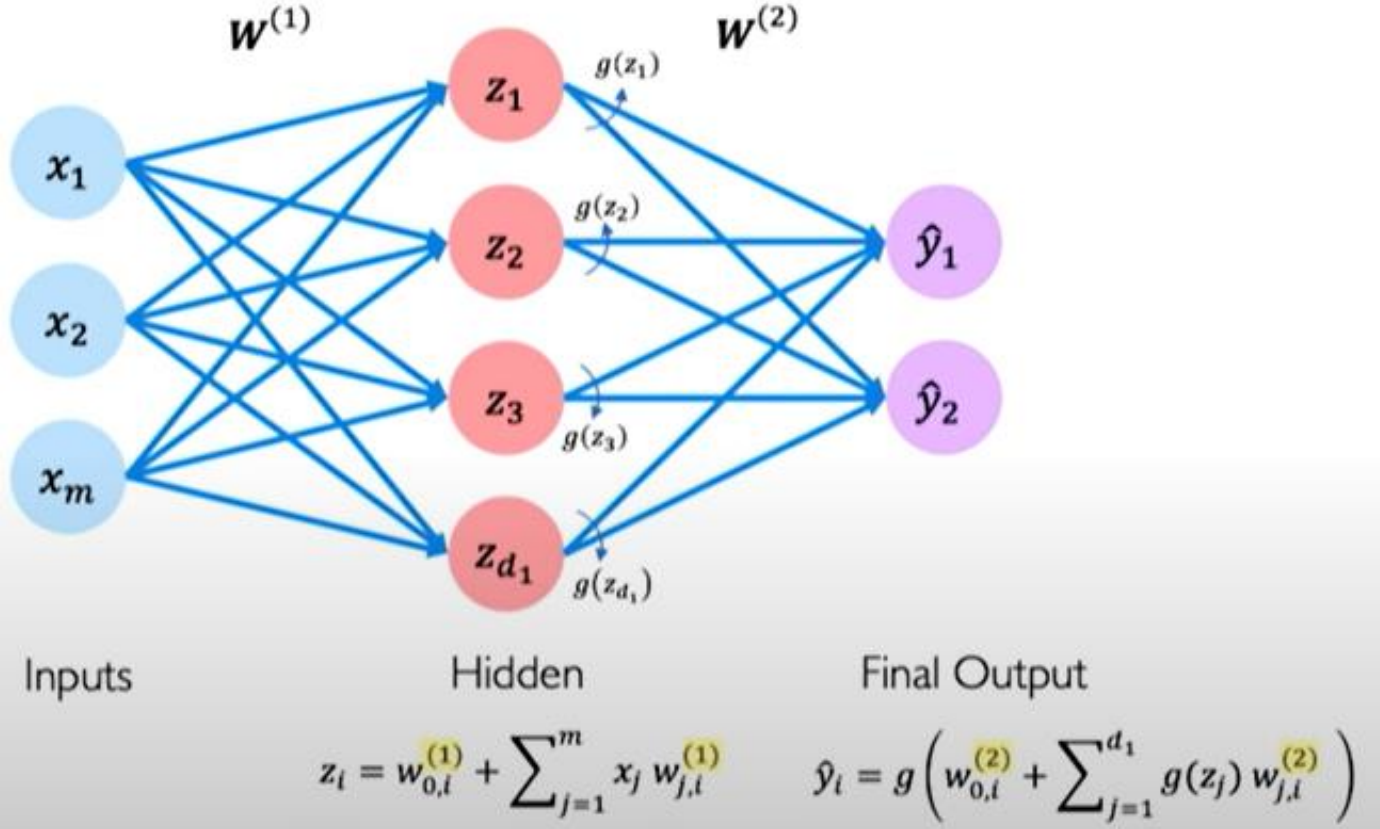
3 steps of Perceptron:

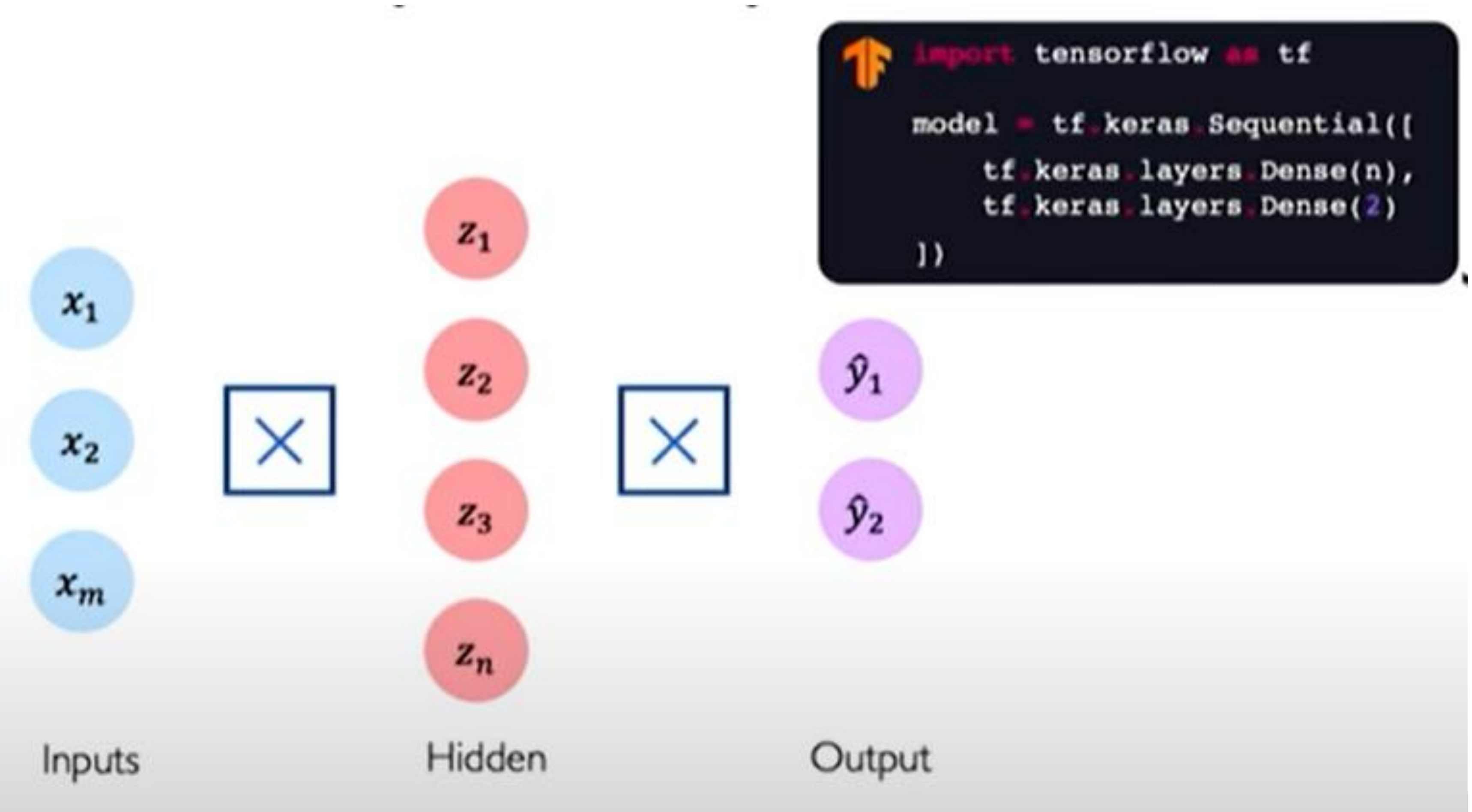
1. Dot product (Inputs multiplied with weights)
2. Add bias
3. Apply non-linearity

Here all inputs are connected to all outputs, these layers are called dense layers

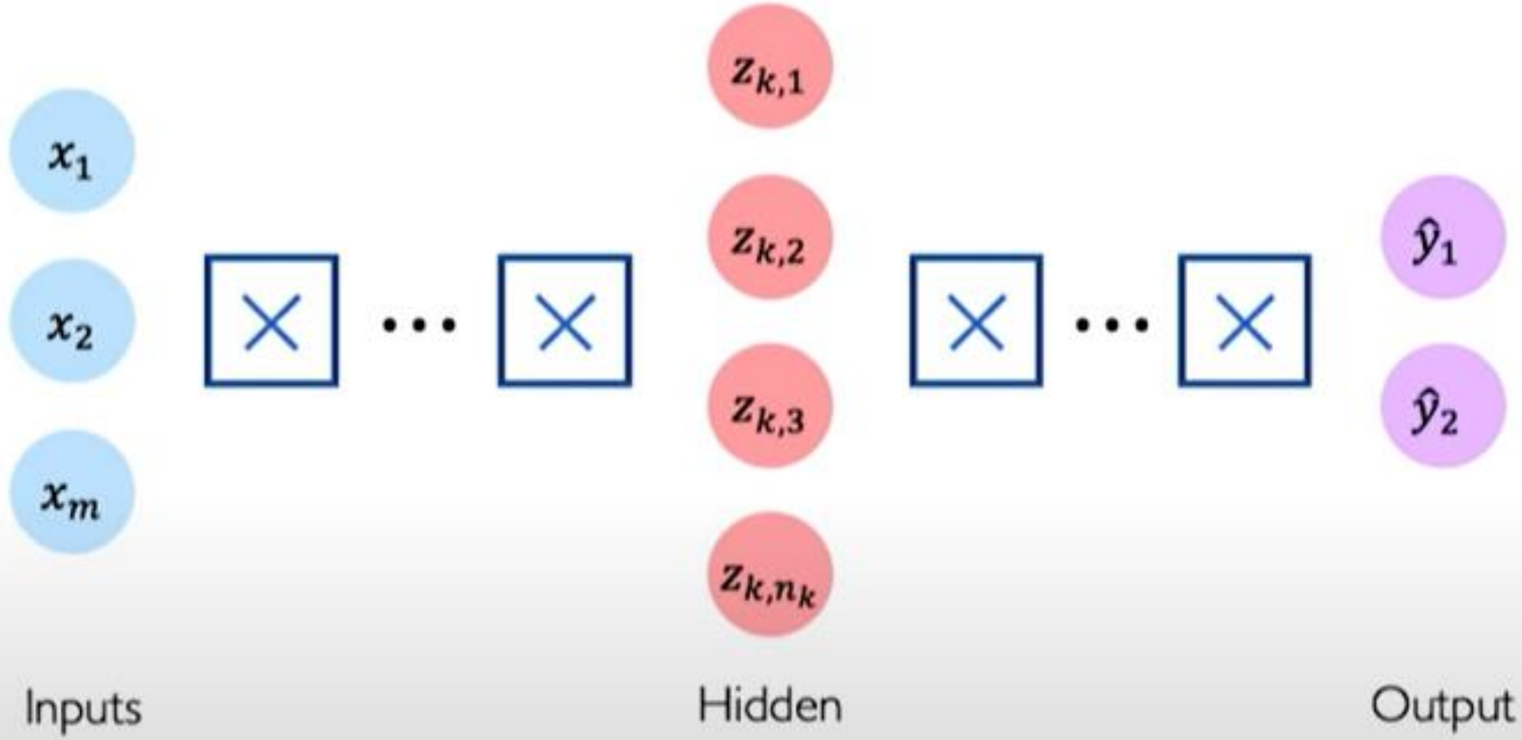


Here all inputs are connected to all outputs, these layers are called dense layers





x_1
x_2
x_m
z_1
z_2
z_3
z_n
$\hat{y}_1$
$\hat{y}_2$
Inputs
Hidden
Output
import tensorflow as tf
model = tf.keras.Sequential([
tf.keras.layers.Dense(n),
tf.keras.layers.Dense(2)
])

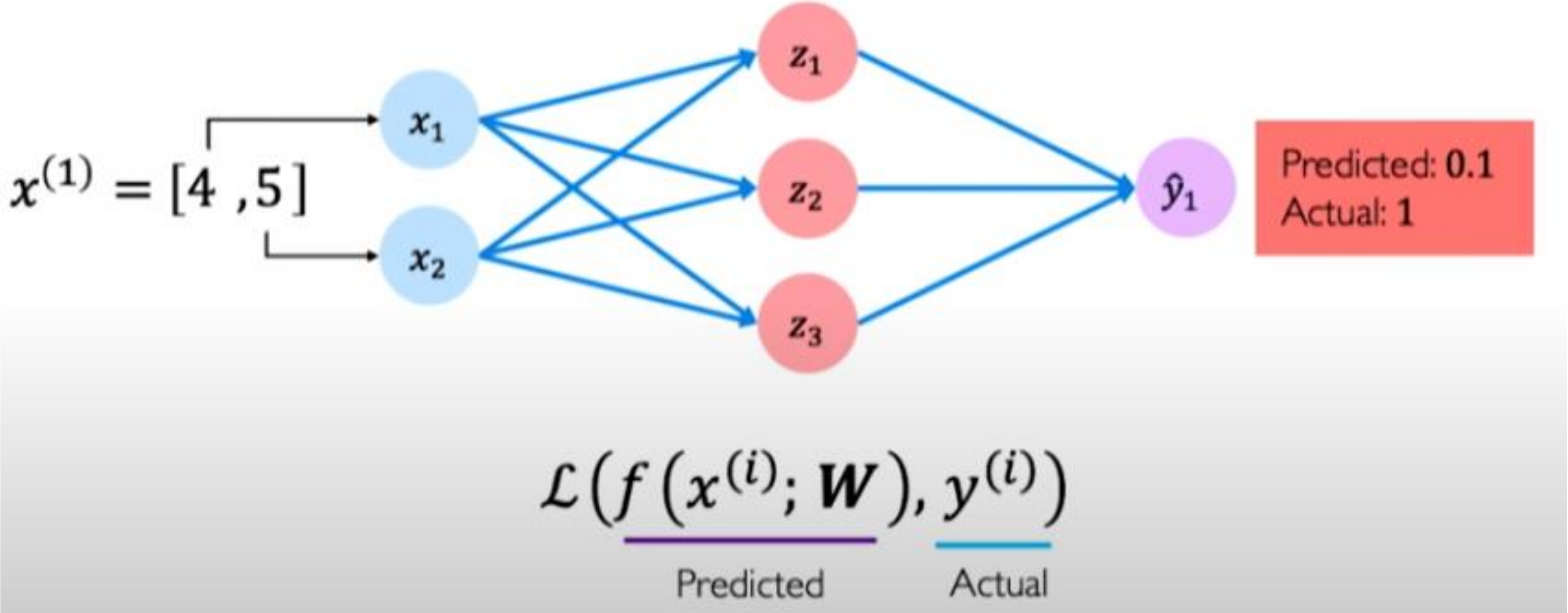


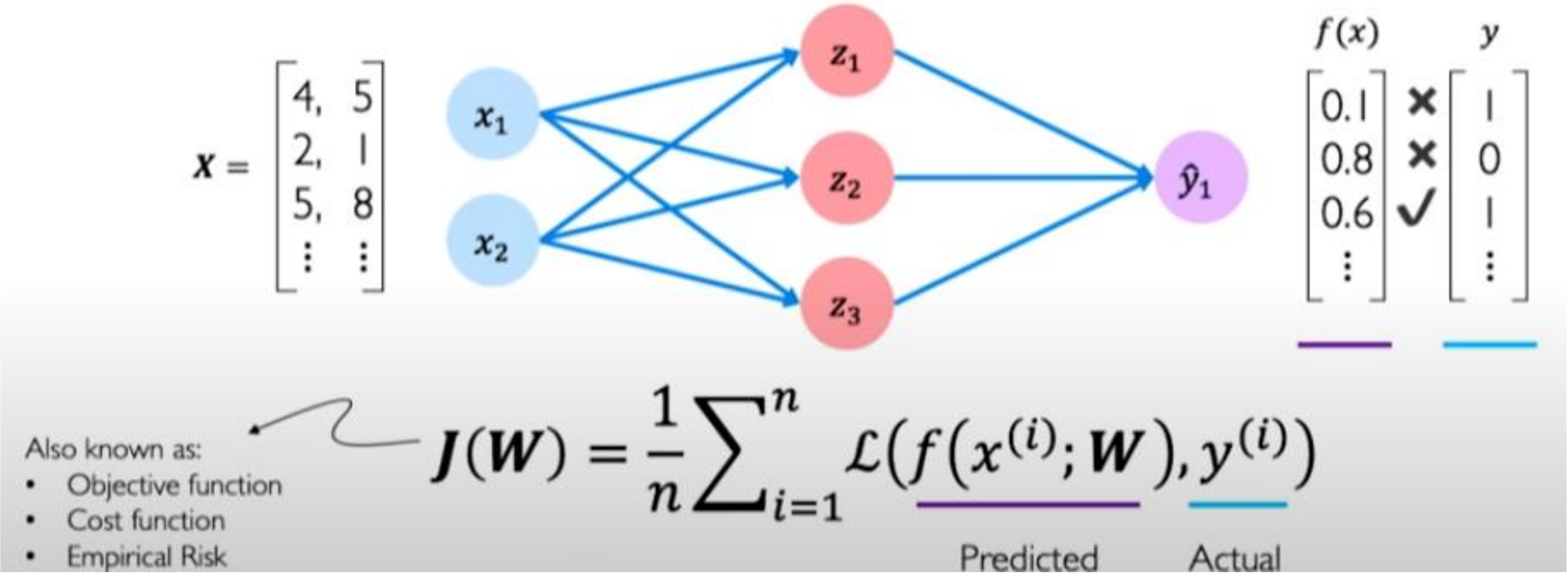

```python
import tensorflow as tf

model = tf.keras.Sequential([
  tf.keras.layers.Dense(n₁),
  tf.keras.layers.Dense(n₂),
  ⋮
  tf.keras.layers.Dense(2)
])
```

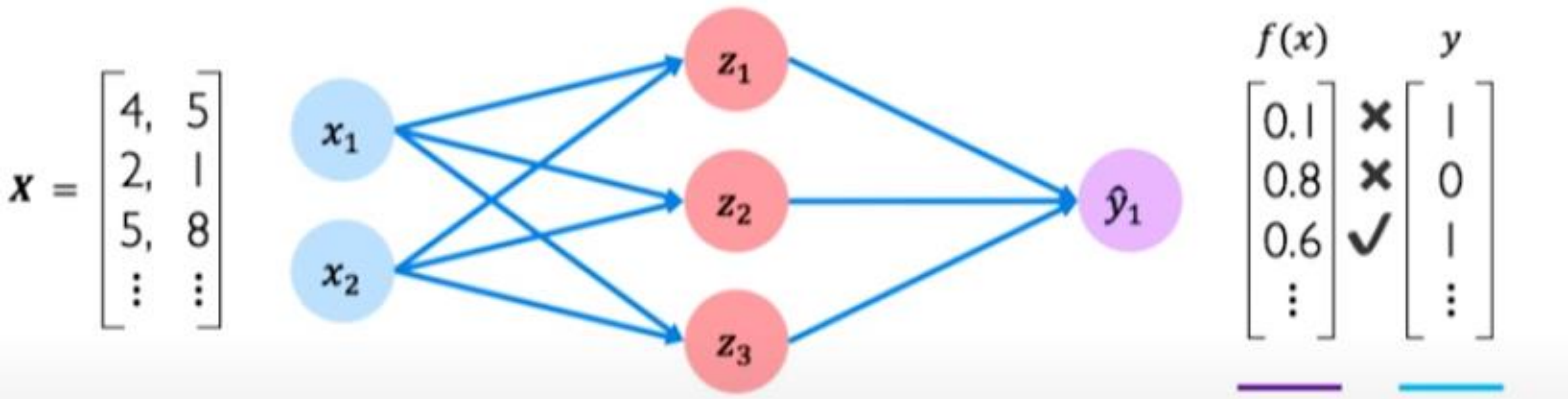
$$z_{k,i} = w_{0,i}^{(k)} + \sum_{j=1}^{n_{k-1}} g(z_{k-1,j})\, w_{j,i}^{(k)}$$

The loss function measures the cost incurred by incorrect predictions.




$X = \begin{bmatrix} 4, & 5 \\ 2, & 1 \\ 5, & 8 \\ \vdots & \vdots \end{bmatrix}$
x_1
x_2
z_1
z_2
z_3
$\hat{y}_1$
$f(x)$
y
$\begin{bmatrix} 0.1 \\ 0.8 \\ 0.6 \\ \vdots \end{bmatrix}$ ✗ ✗ ✓ $\begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$
Also known as:
• Objective function
• Cost function
• Empirical Risk
$J(W) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\big(f(x^{(i)}; W), y^{(i)}\big)$
Predicted
Actual

Output a probability between 0 and 1



$$J(\boldsymbol{W}) = -\frac{1}{n}\sum_{i=1}^{n} \underbrace{y^{(i)}}_{\text{Actual}} \log\left(\underbrace{f\left(x^{(i)}; \boldsymbol{W}\right)}_{\text{Predicted}}\right) + \left(1 - \underbrace{y^{(i)}}_{\text{Actual}}\right) \log\left(1 - \underbrace{f\left(x^{(i)}; \boldsymbol{W}\right)}_{\text{Predicted}}\right)$$

Used for regression models; Output continuous numbers