DEEP LEARNING WITH PYTHON

Meta Brains

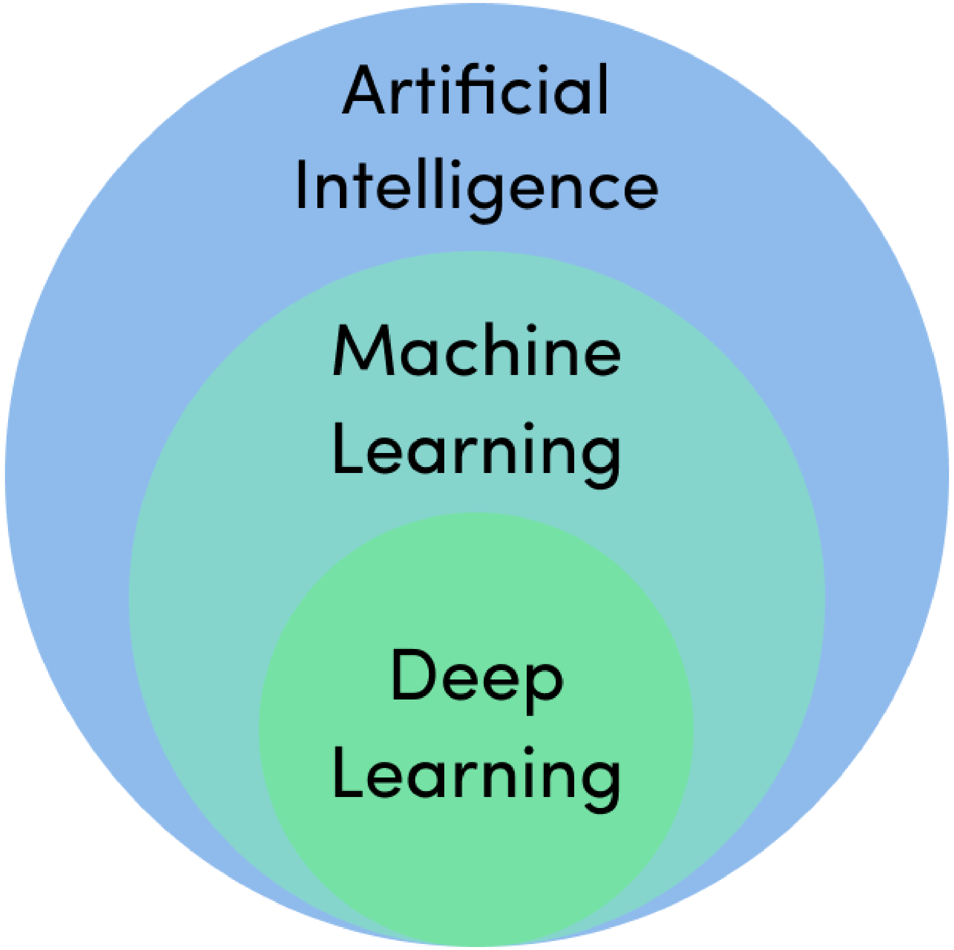
01 What is deep learning?

02 Why is deep learning important?

03 Software and frameworks

Meta Brains

What Is Deep Learning



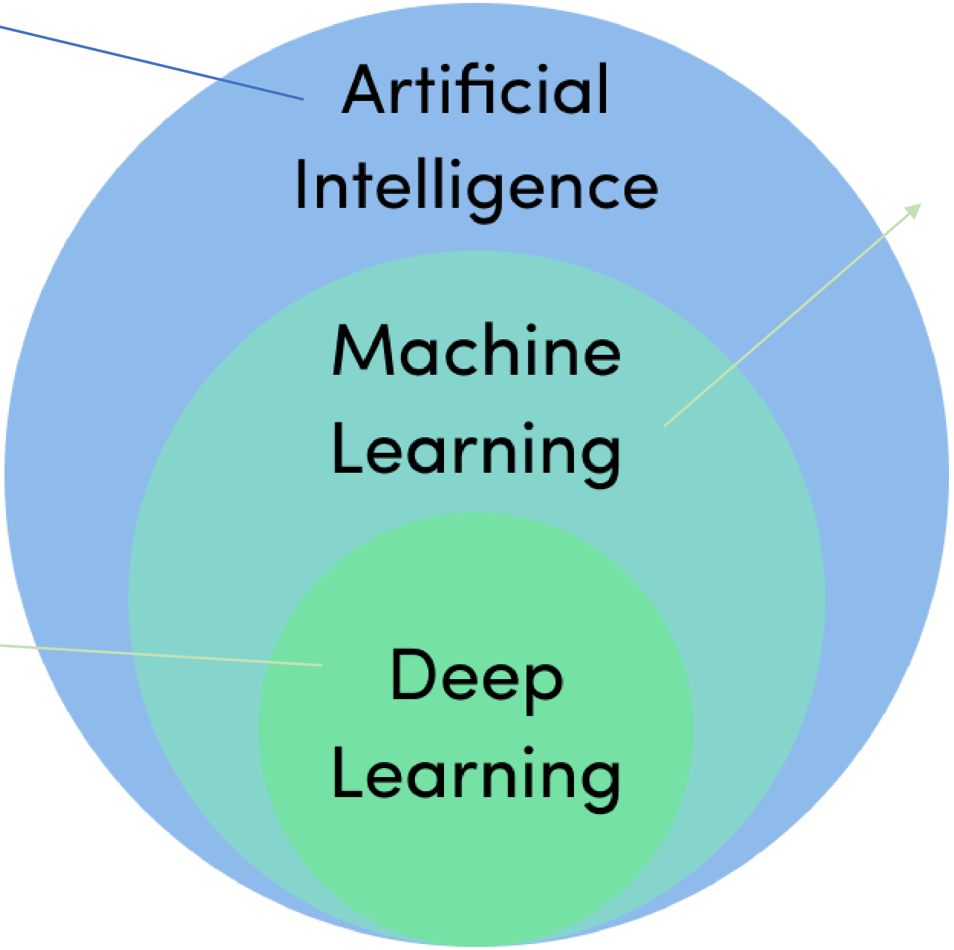
Meta Brains

What is Deep Learning

AI: The computer performs tasks without explicitly programming them .

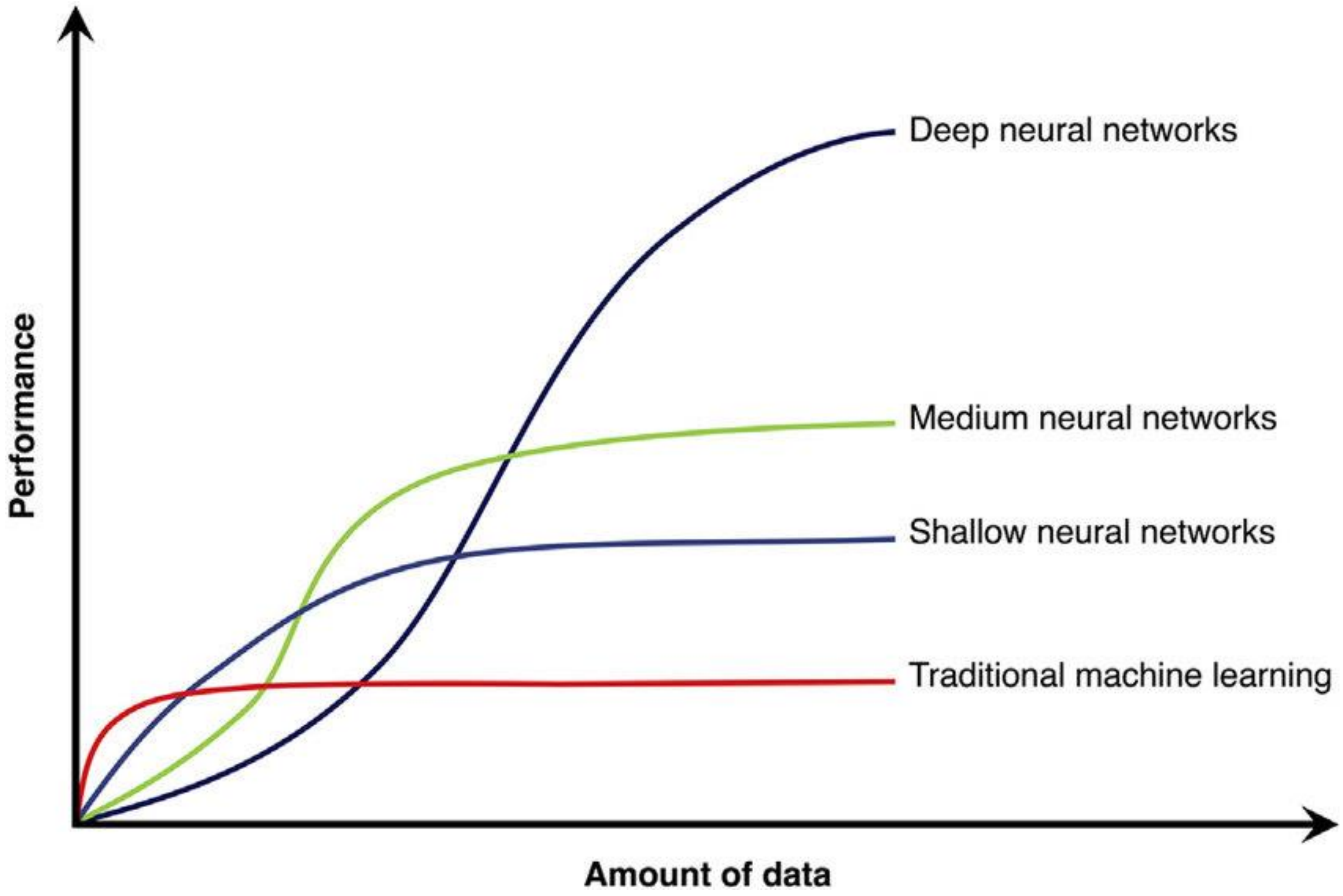
Programming: Data+ Rules = Answers

ML: Data + Answers = Rules



DL: Performing machine learning tasks using a deep neural network (DNN)

Why Is Deep Learning Important?



Meta Brains

01 What is deep learning?

02 Why is deep learning important?

03 Software and frameworks

Software and Frameworks

julia

mxnet

K

DEEP LEARNING

theano

Microsoft CNTK

Caffe2

Chainer

01 Introduction

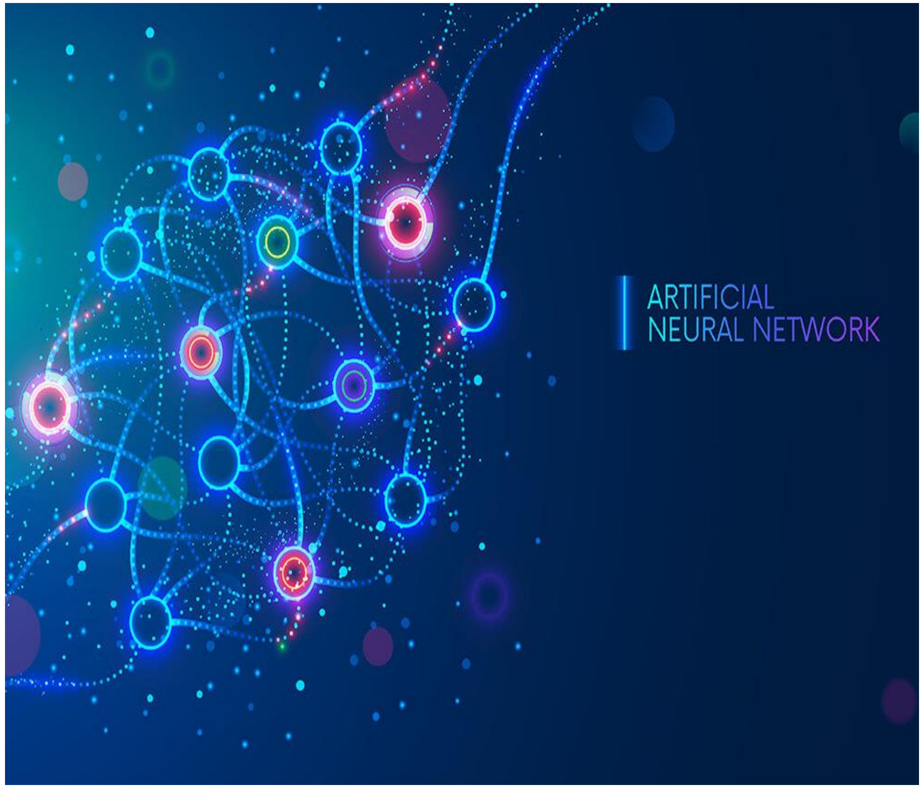
02 Anatomy and function of neurons

03 An introduction to the neural network

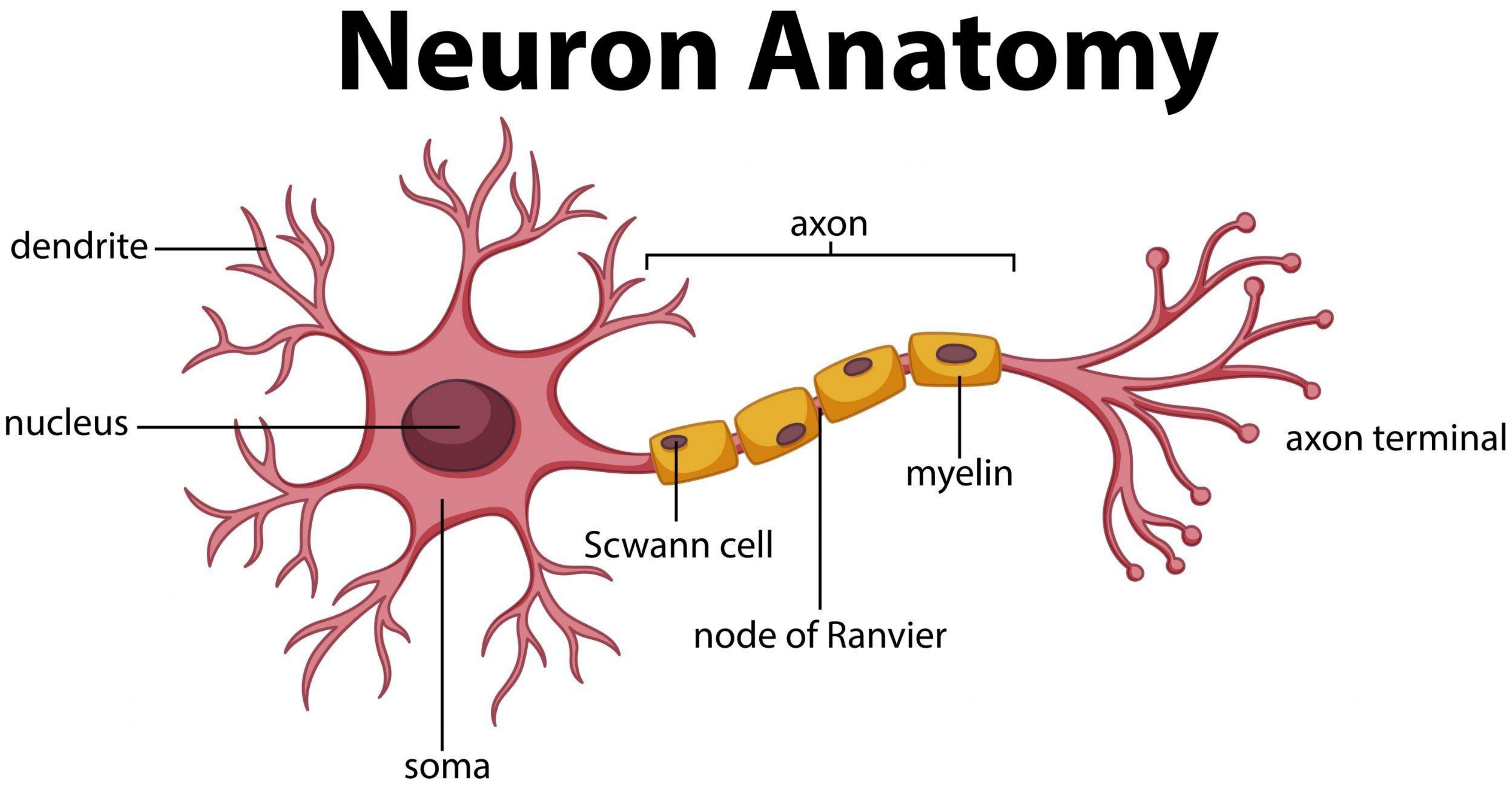
04 Architecture of a neural network

Introduction

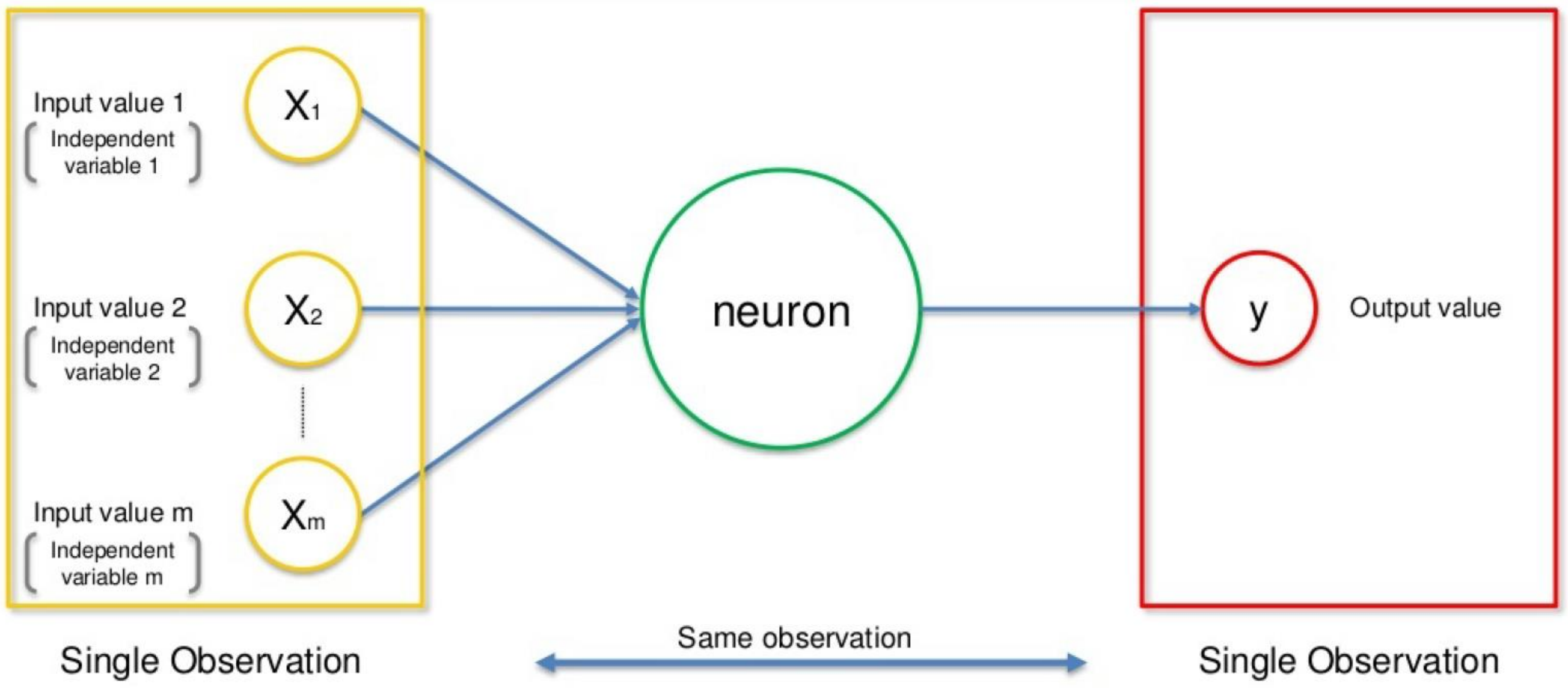
An artificial neuron network is a computational model that mimics the way nerve cells work in the human brain.
There are more complicated and high-end models in the DL approach. However, ANN is a vital element in all the models in DL.



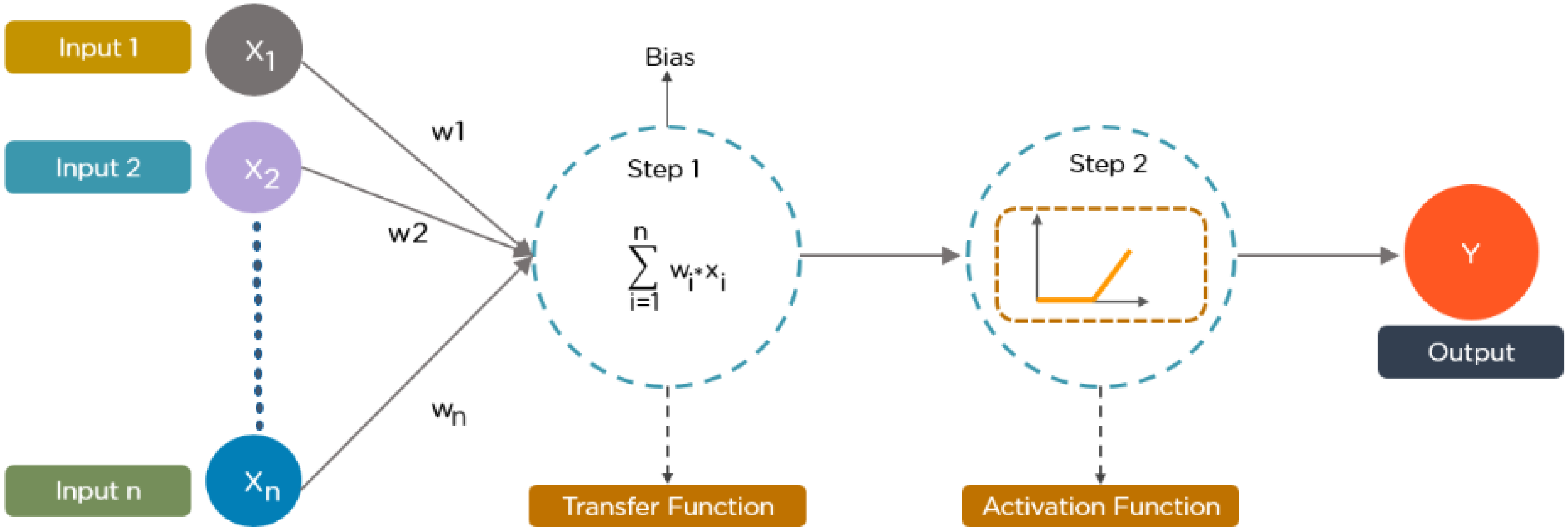
Anatomy and function of neurons



An introduction to the neural network (1/2)

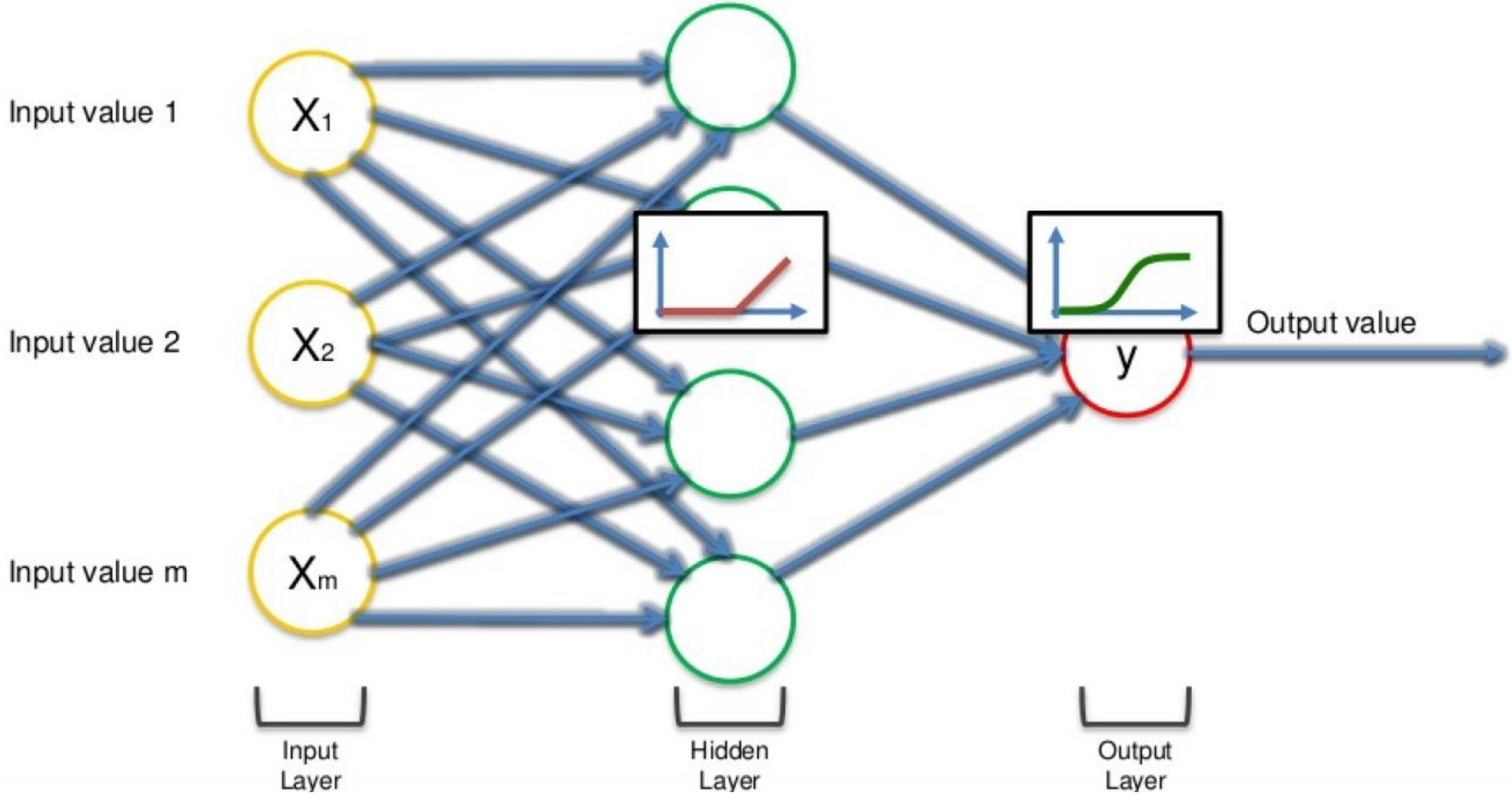


An introduction to the neural network (2/2)



Meta Brains

Architecture of a neural network

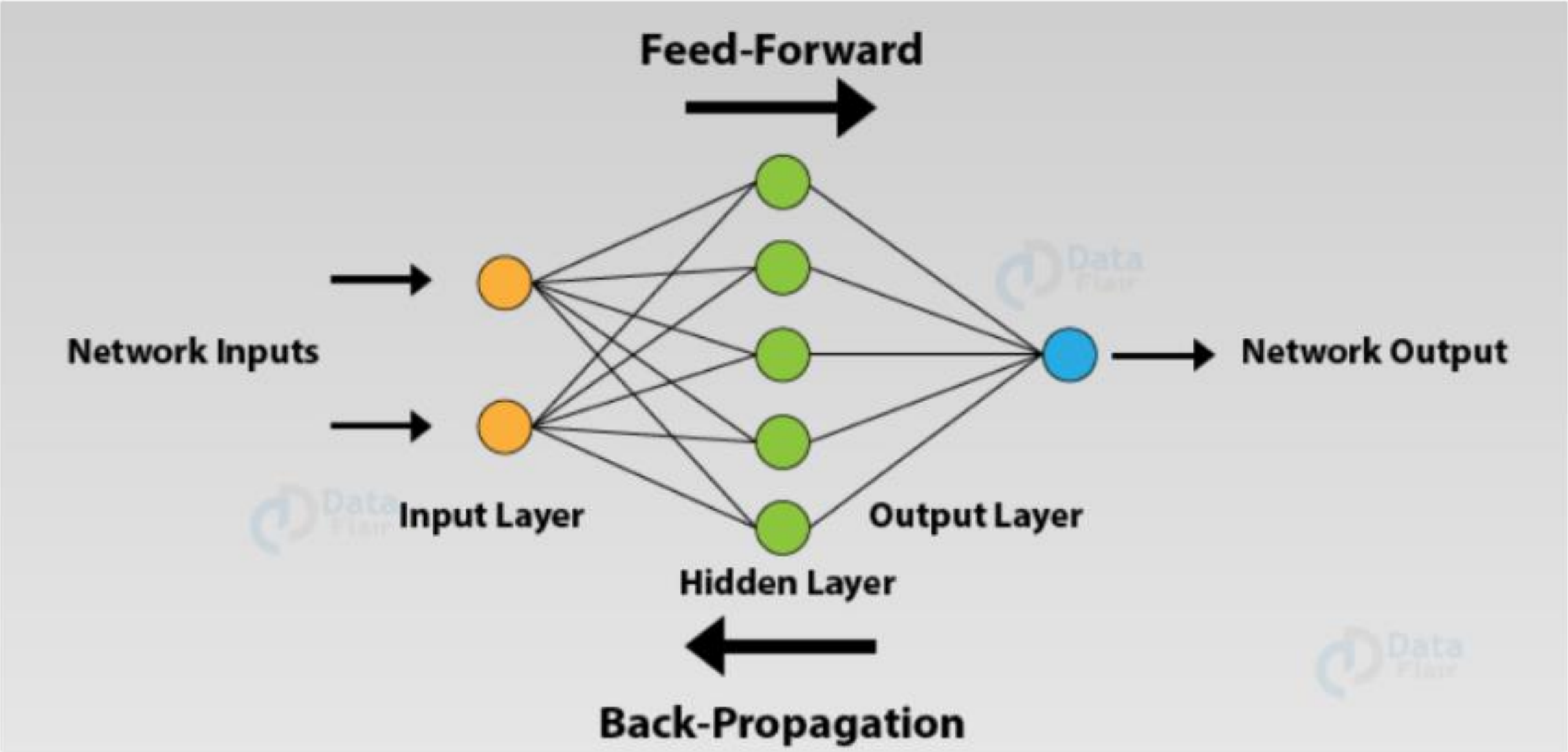


Meta Brains

01 Feed-forward and backpropagation networks

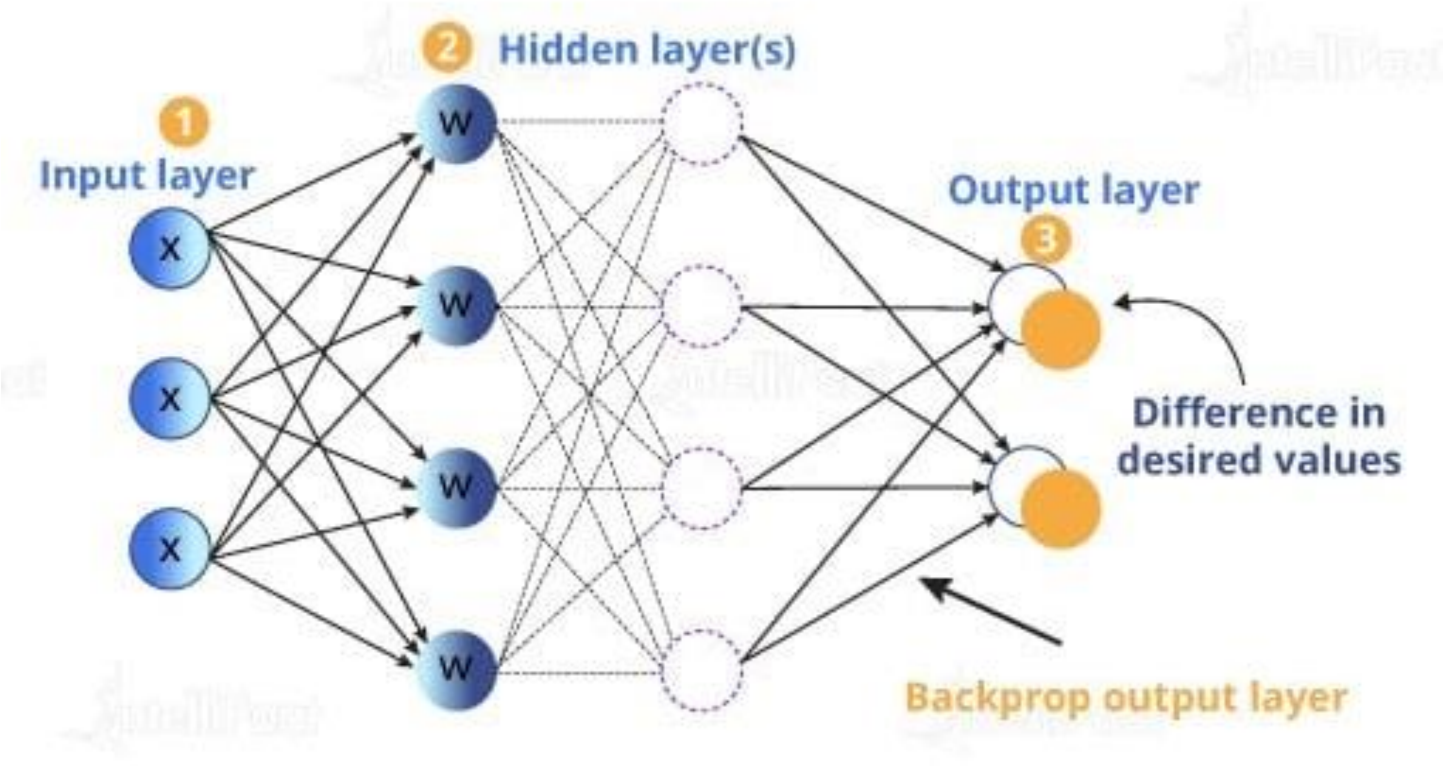
02 Backpropagation in neural networks

03 Minimizing the cost function using backpropagation


Feed-Forward
Network Inputs
Network Output
Input Layer
Output Layer
Hidden Layer
Back-Propagation

Meta Brains

Backpropagation

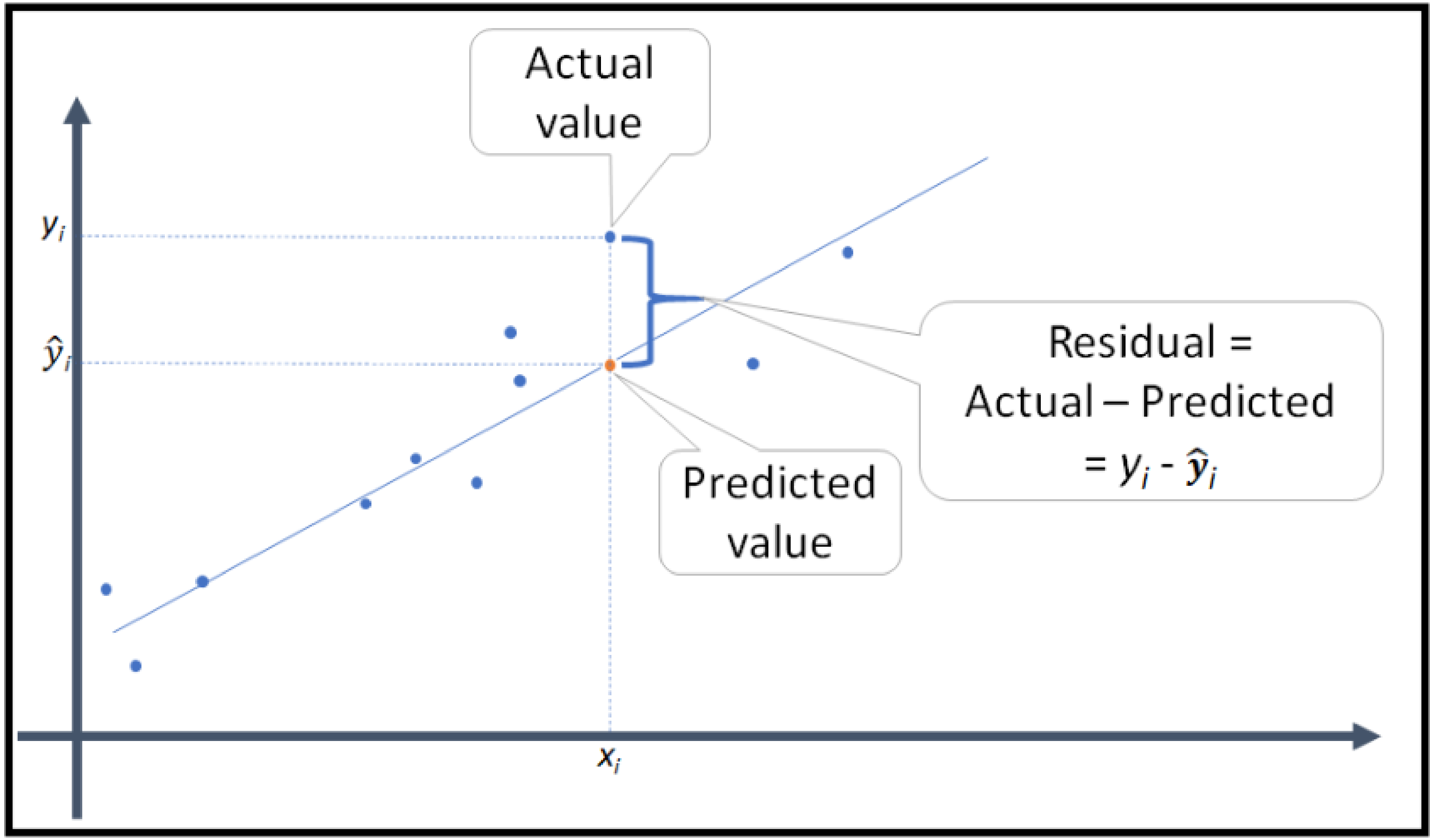


Meta Brains

01 Feed-forward and backpropagation networks

02 Backpropagation in neural networks

03 Minimizing the cost function using backpropagation


Actual value
y_i
$\hat{y}_i$
Residual = Actual – Predicted = y_i - $\hat{y}_i$
Predicted value
x_i

Meta Brains

01 Single layer perceptron (SLP) model

02 Radial Basis Network (RBN)

03 Multi-layer perceptron (MLP) Neural Network

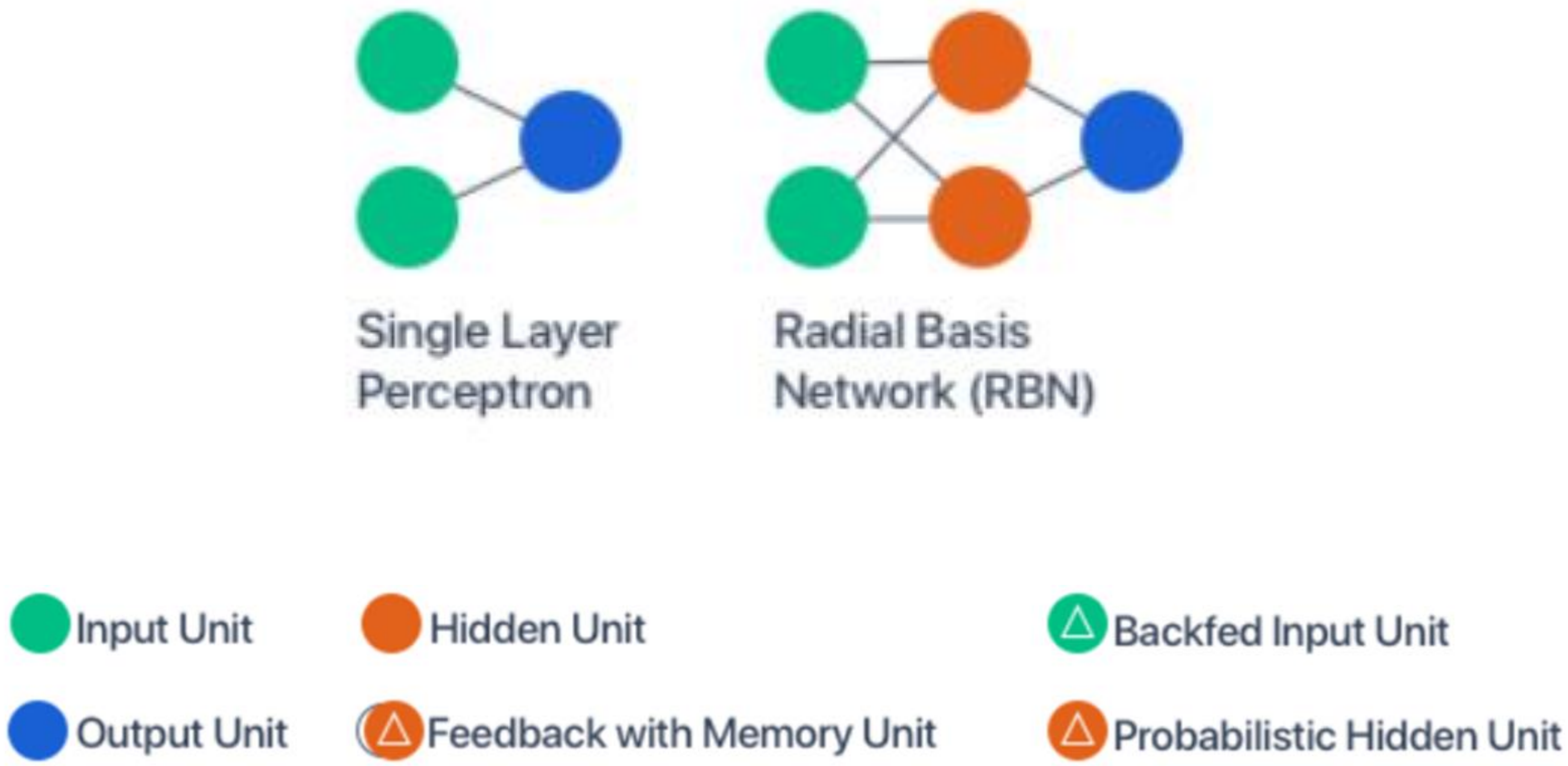
04 Recurrent neural network (RNN)

05 Long Short-Term Memory (LSTM) networks

06 Hopfield neural network

07 Boltzmann Machine Neural Network

NN Architecture Types



Meta Brains

Introduction to Deep Learning

Artificial Neural Networks (ANN)

Propagation of Information in ANNs

Neural Network Architectures

Activation Functions

Gradient Descent Algorithm

Summary-Overview of Neural Networks

Implementation of ANN in Python

Convolutional Neural Networks (CNN)

Implementation of CNN in Python

01 Single layer perceptron (SLP) model

02 Radial Basis Network (RBN)

03 Multi-layer perceptron (MLP) Neural Network

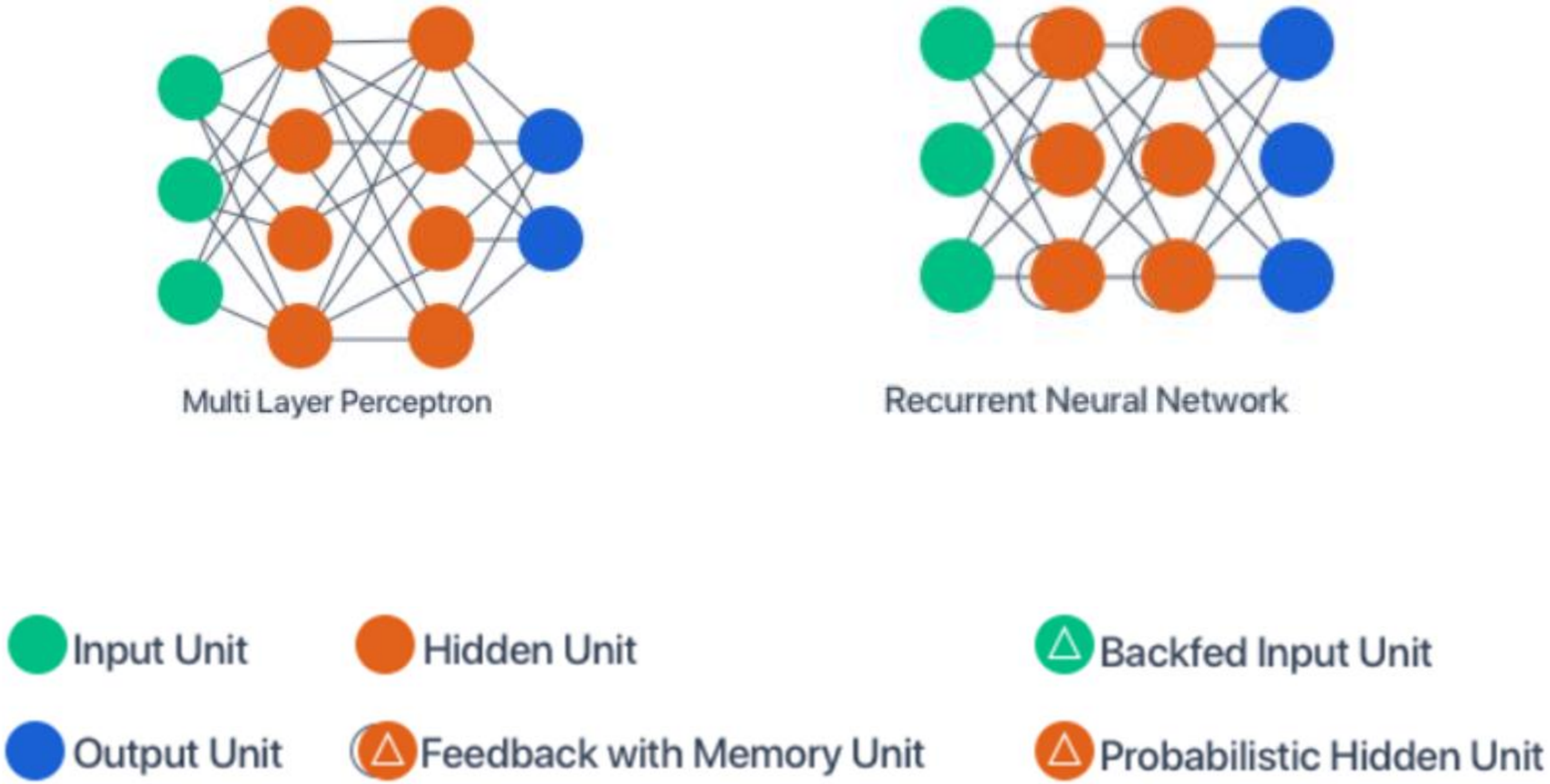
04 Recurrent neural network (RNN)

05 Long Short-Term Memory (LSTM) networks

06 Hopfield neural network

07 Boltzmann Machine Neural Network

NN Architecture Types



Introduction to Deep Learning

Artificial Neural Networks (ANN)

Propagation of Information in ANNs

Neural Network Architectures

Activation Functions

Gradient Descent Algorithm

Summary-Overview of Neural Networks

Implementation of ANN in Python

Convolutional Neural Networks (CNN)

Implementation of CNN in Python

01 Single layer perceptron (SLP) model

02 Radial Basis Network (RBN)

03 Multi-layer perceptron (MLP) Neural Network

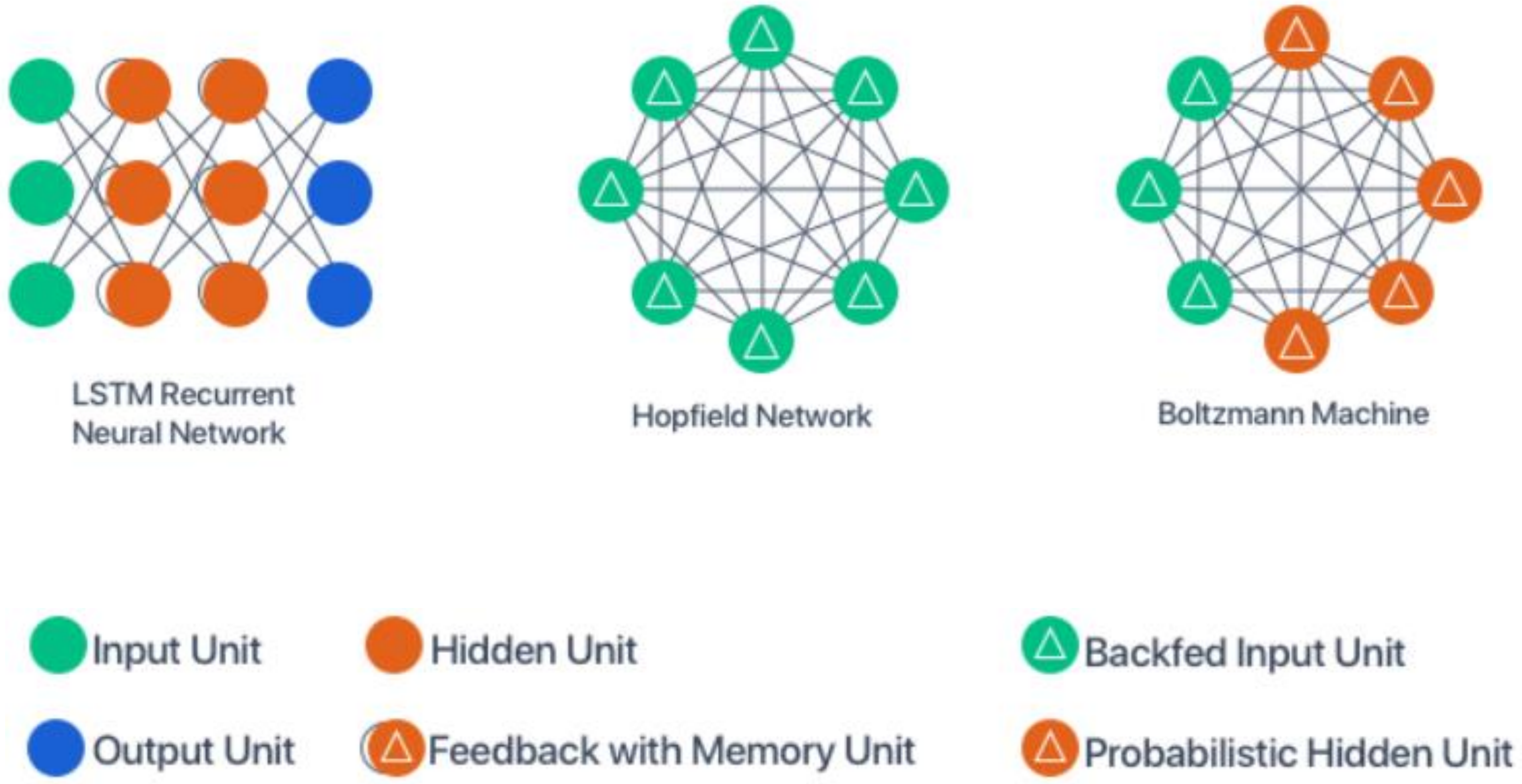
04 Recurrent neural network (RNN)

05 Long Short-Term Memory (LSTM) networks

06 Hopfield neural network

07 Boltzmann Machine Neural Network

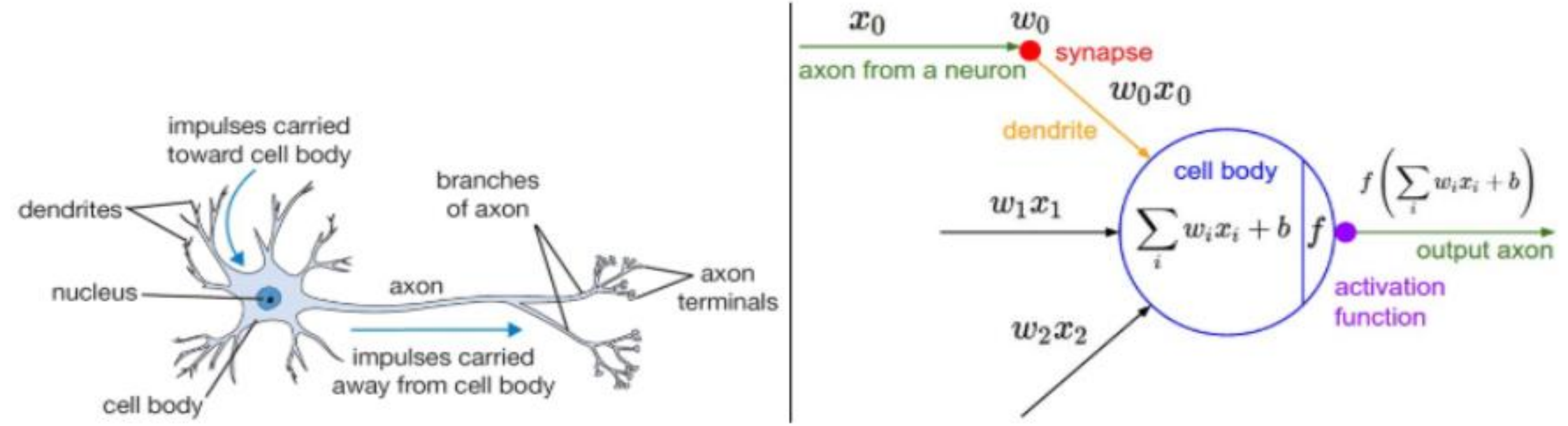
NN Architecture Types



Meta Brains

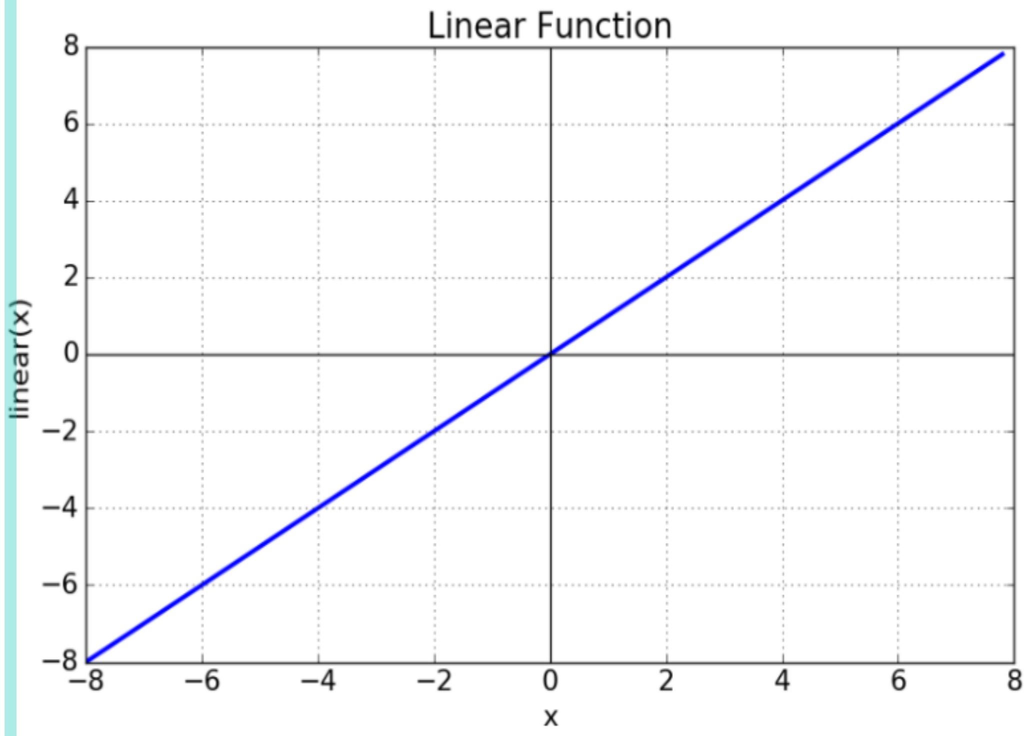
What is the Activation Function? – Superpower for ANN (1/2)

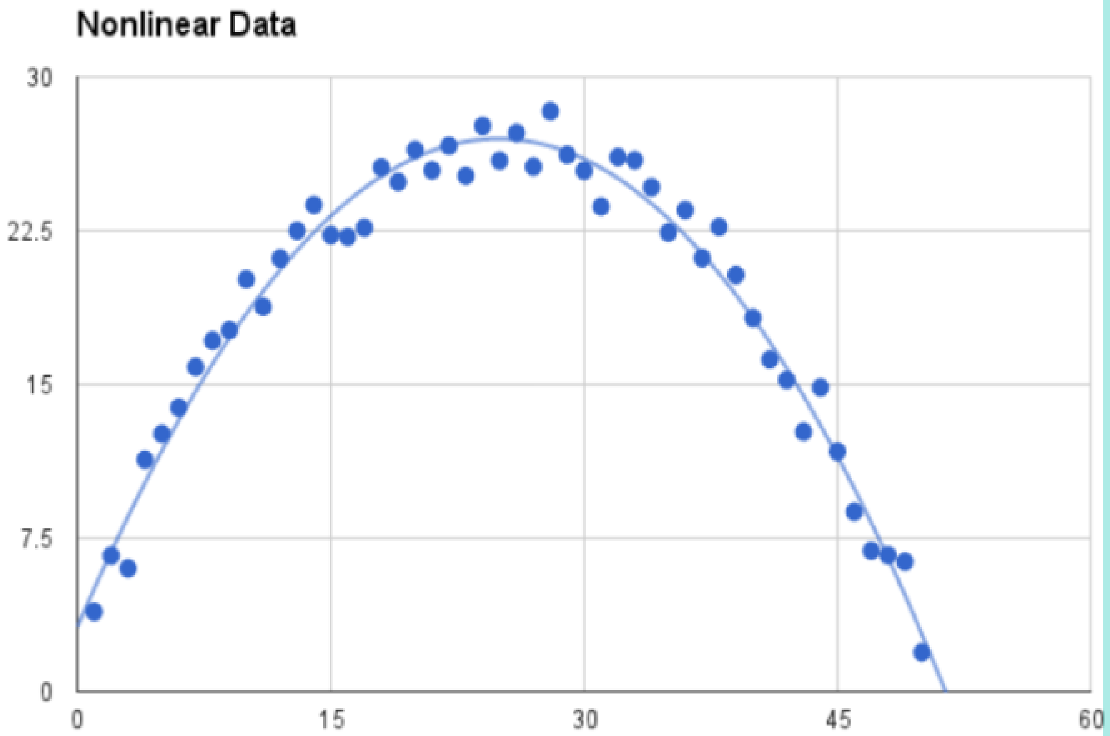
So what does an Activation Function (AF) actually do? AF is a function that is added to an ANN to help the network learn complex patterns in the data.



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

What is the Activation Function? – Superpower for ANN (2/2)





Meta Brains

Important Terminologies

1. **Differential function:** Change in Y-axis w.r.t change in X-axis (slope!).

2. **Monotonic function:** A function that is increasing on its entire domain or decreasing on its entire domain.

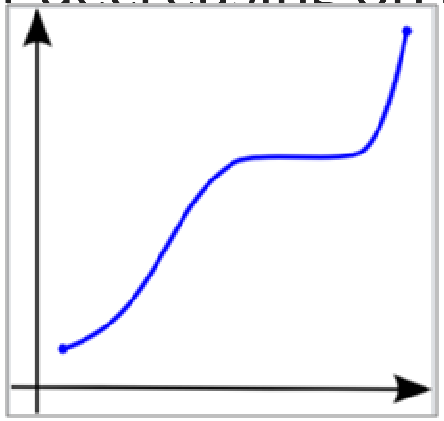


Figure 1 - A monotonically increasing function

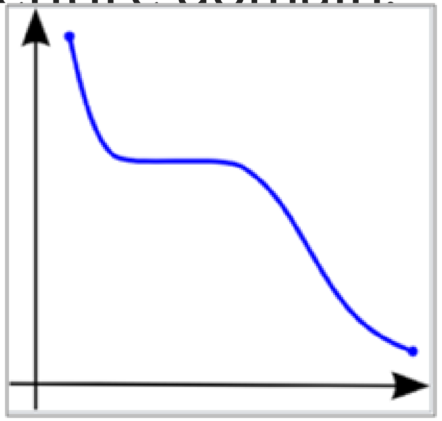


Figure 2 - A monotonically decreasing function

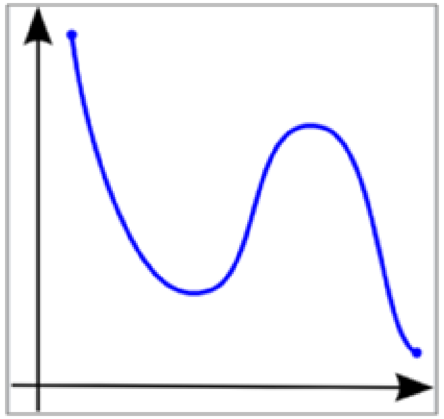
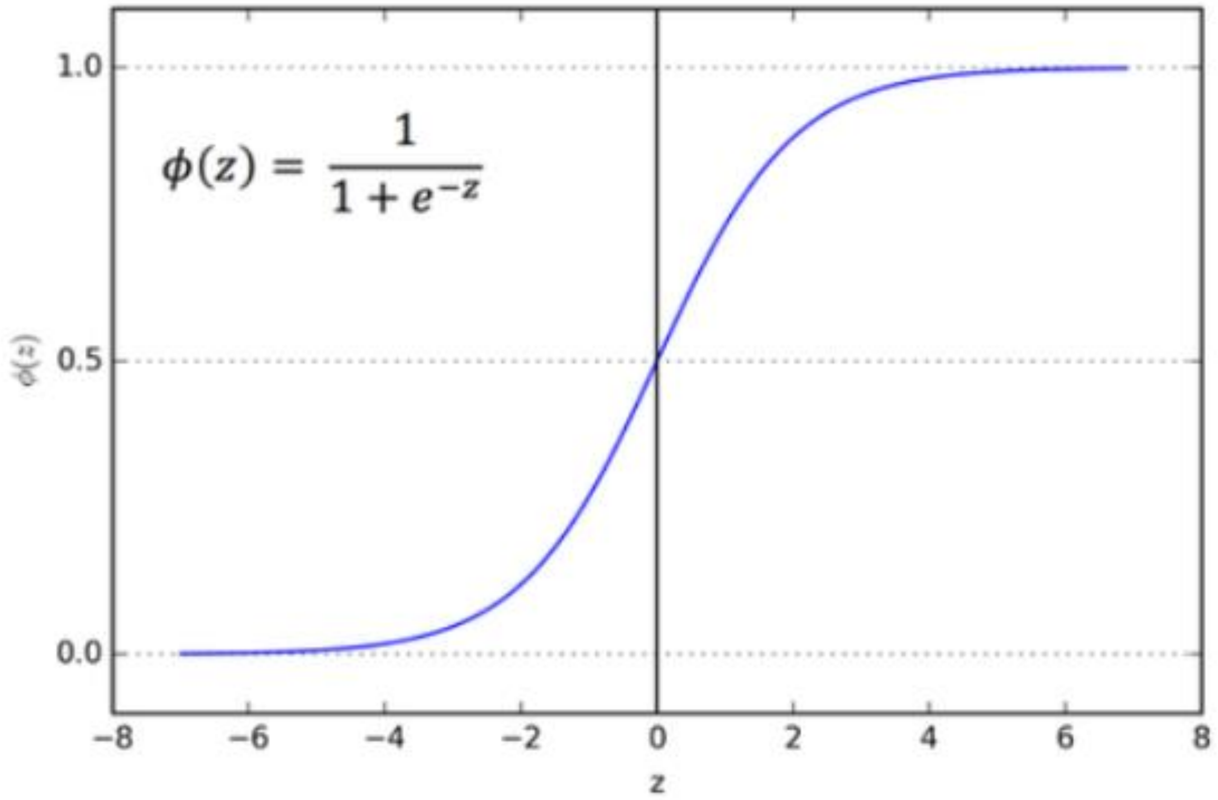


Figure 3 - A function that is not monotonic

Non-Linear Activation Functions (1/5)

1. Sigmoid function

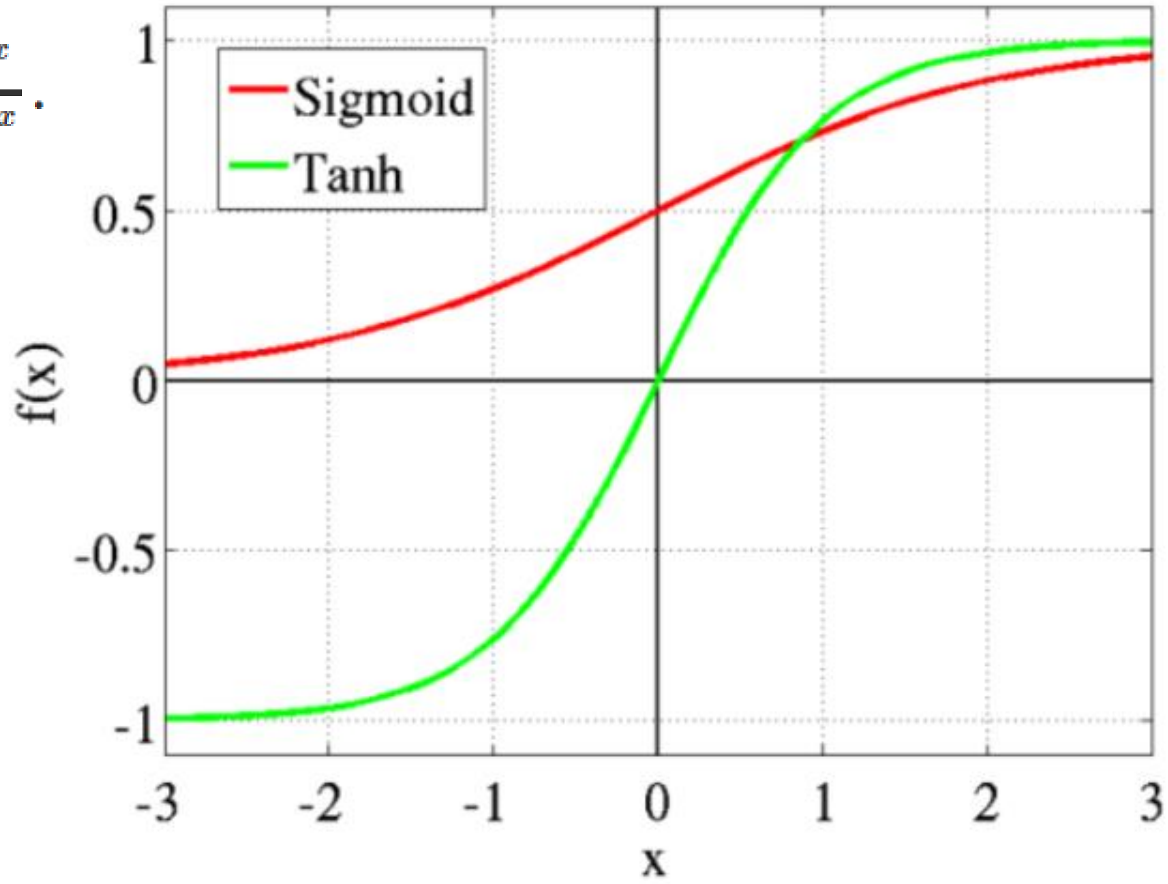


Meta Brains

Non-Linear Activation Functions (2/5)

2. Hyperbolic Tangent function (Tanh)

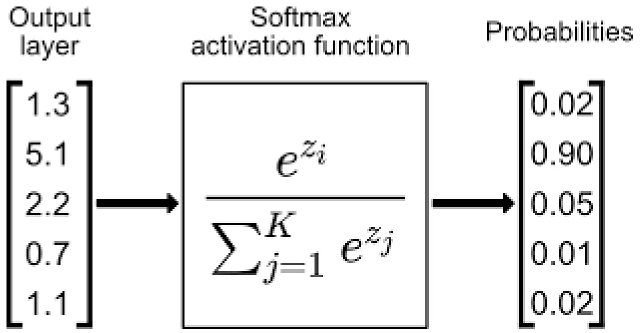
$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

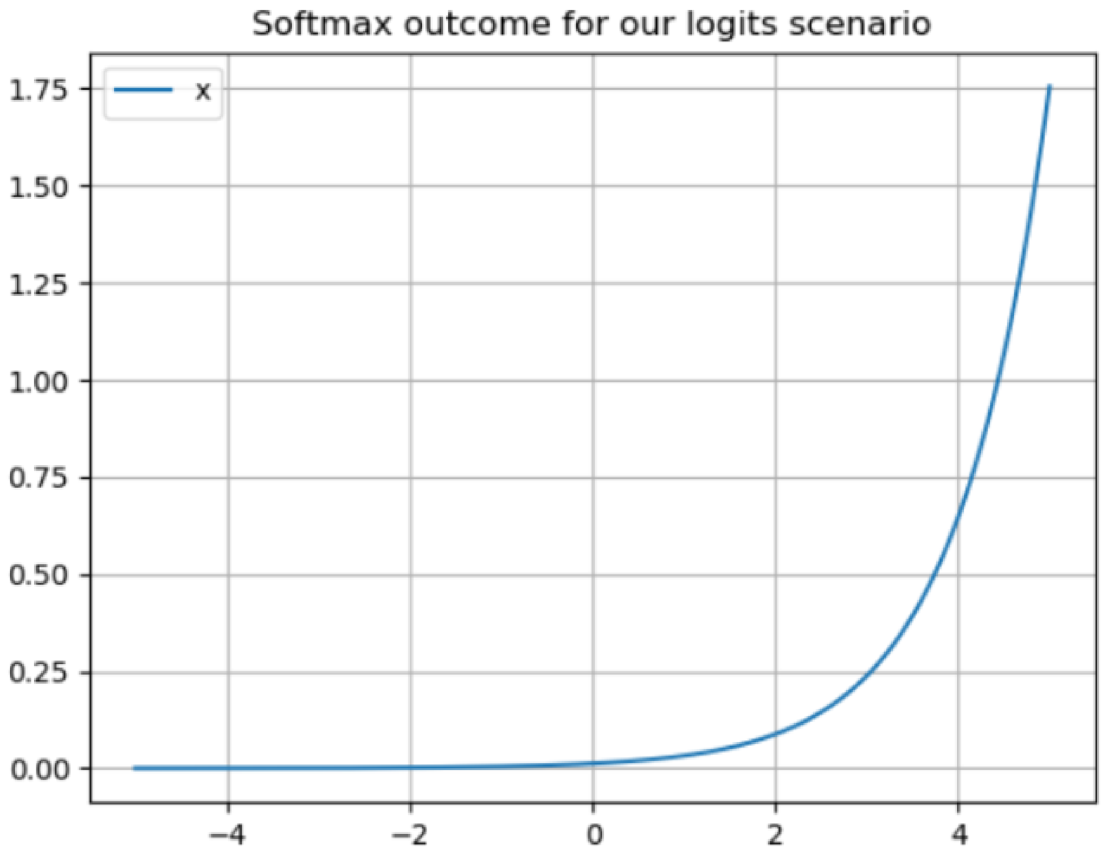


Meta Brains

Non-Linear Activation Functions (3/5)

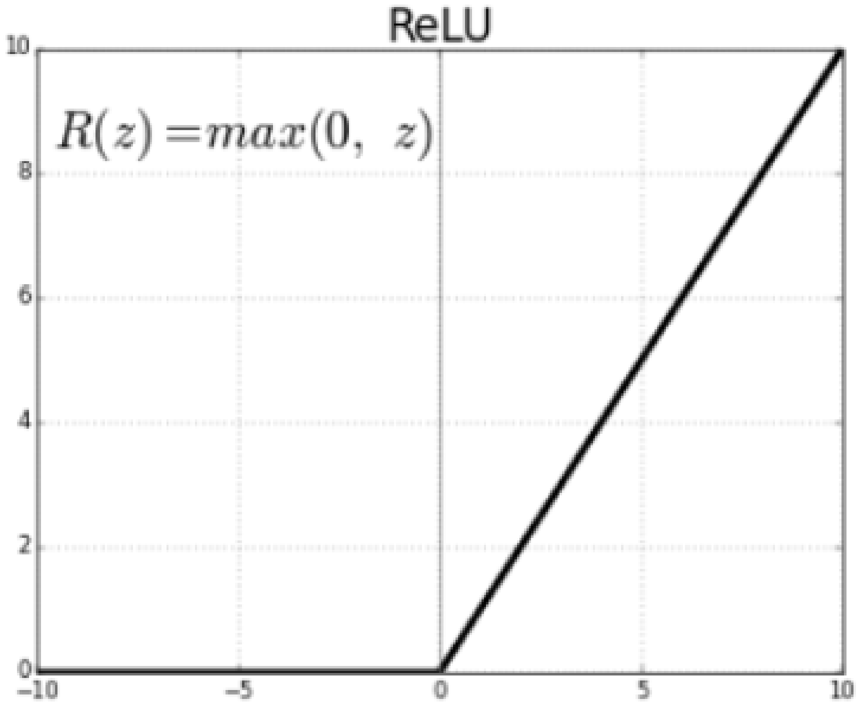
3. Softmax function





Non-Linear Activation Functions (4/5)

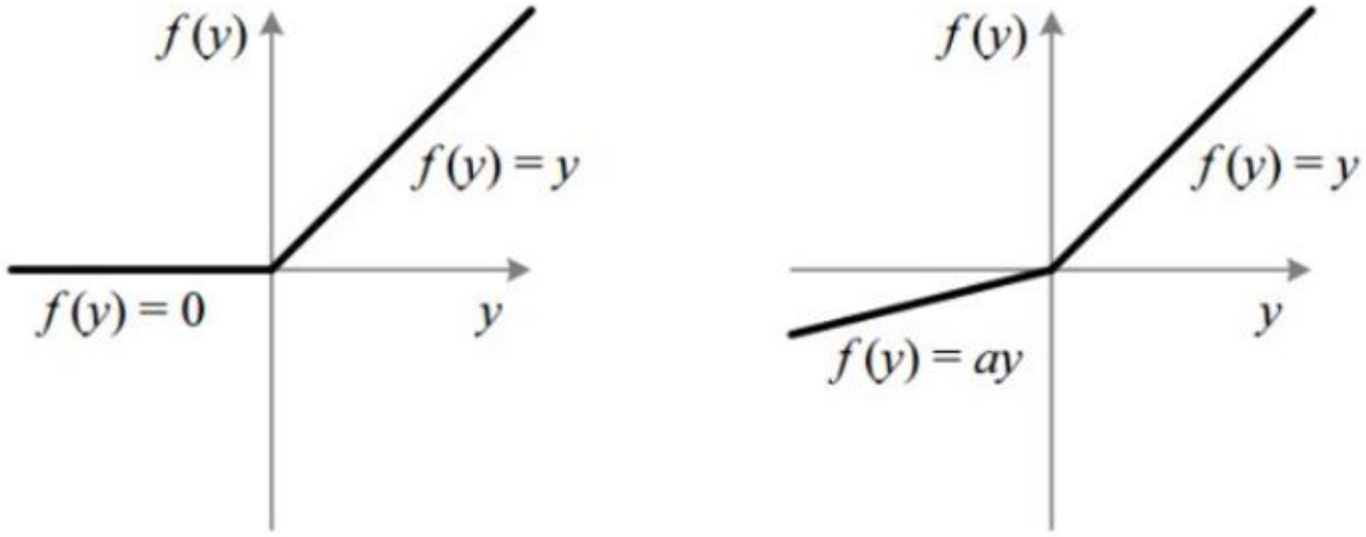
4. Rectified Linear Unit (ReLU) function



Meta Brains

Non-Linear Activation Functions (5/5)

5. Leaky ReLU



Gradient Descent

Gradient: measures how much the output of a function changes if you change the inputs a little bit. In mathematics, it is similar to slope.

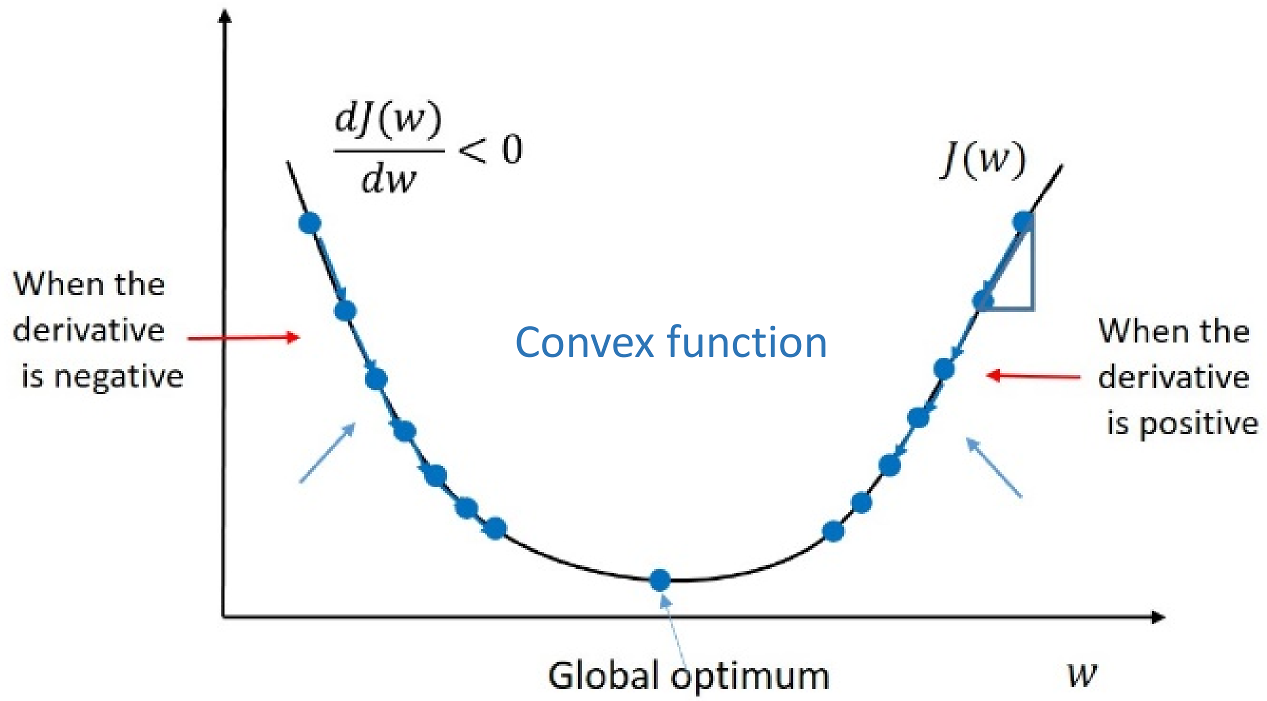
$$\mathbf{b} = \mathbf{a} - \gamma \nabla \mathbf{f}(\mathbf{a})$$

b: next position

a: current position

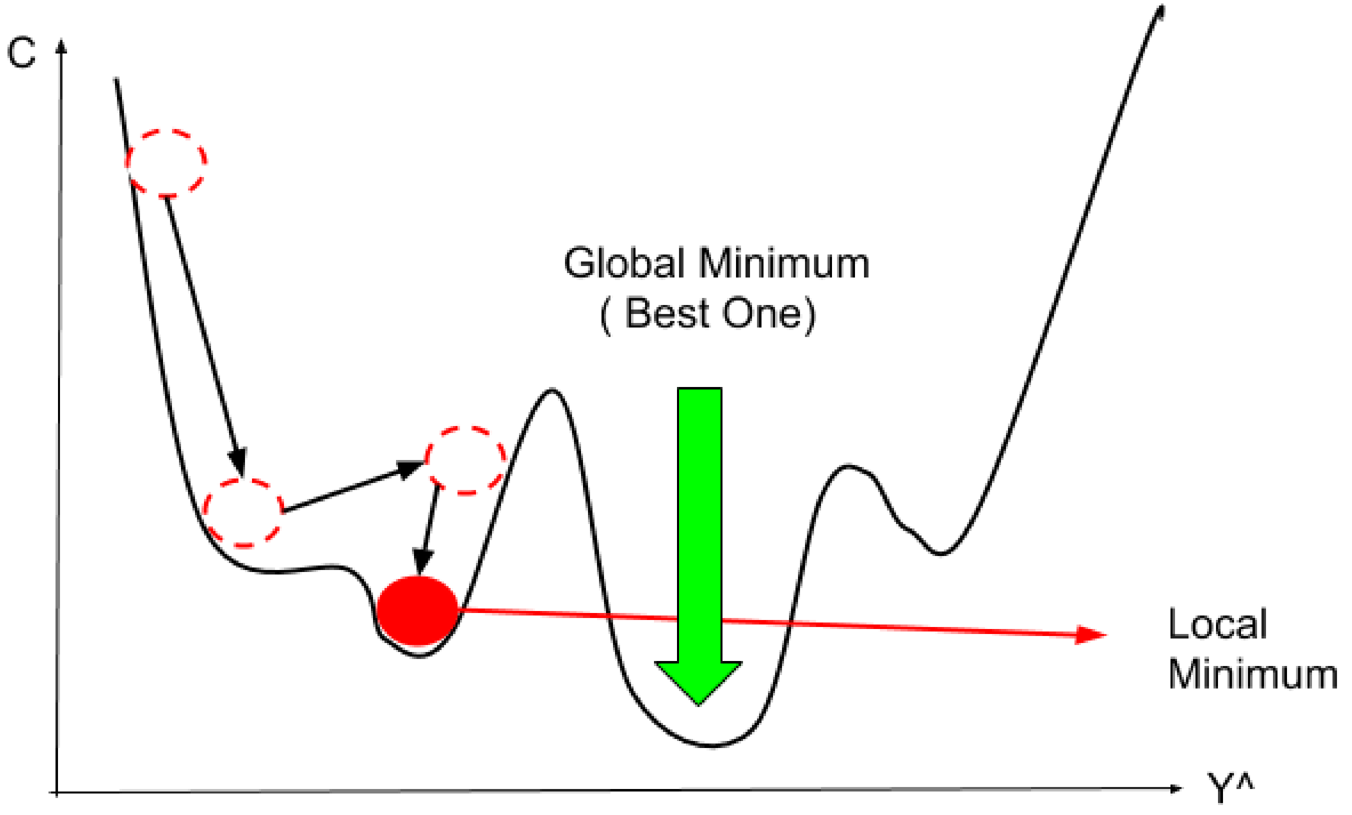
ϒ(gamma) : waiting factor

Δf(a) : direction



Gradient Descent = Descending into Minimum of COST function

Stochastic Descent



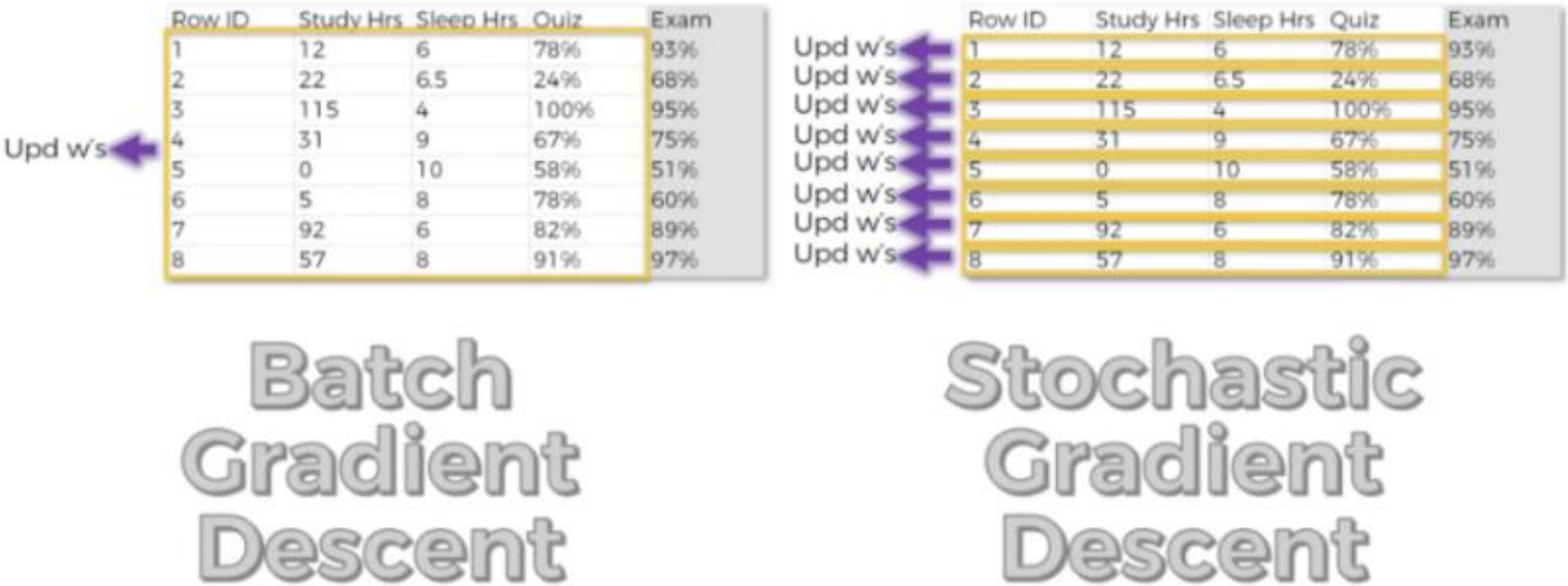
01 What is Gradient Descent?

02 What is Stochastic Gradient Descent?

03 Gradient Descent versus Stochastic Gradient Descent

Stochastic Descent (2/2)

For more clarification between GD and SGD, see the following picture:



Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

* Mini Batch Gradient Descent = Gradient Descent + Stochastic Gradient Descent

Introduction to Deep Learning

Artificial Neural Networks (ANN)

Propagation of Information in ANNs

Neural Network Architectures

Activation Functions

Gradient Descent Algorithm

Summary-Overview of Neural Networks

Implementation of ANN in Python

Convolutional Neural Networks (CNN)

Implementation of CNN in Python

01 **How artificial neural networks work**

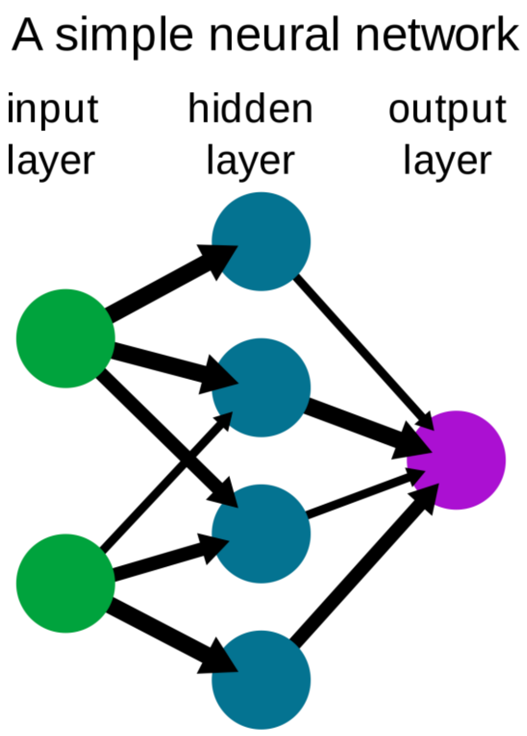
02 Advantages of neural networks

03 Disadvantages of neural networks

04 Applications of neural networks

How a network works – Overall Summary (1/4)

- It is consisted of a number of simple but highly interconnected elements or nodes, called 'neurons'. Those neurons are organized in layers that process information using dynamic state responses to external inputs.
- The first layer is the input layer. The last layer is the output layer by default.



Meta Brains

How a network works – Overall Summary (2/4)

- Let's assume that we need to calculate the price of the house and we don’t have a hidden layer. So, the price will be simply the summation of the product of inputs and their weight respectively. This is an approach of a machine learning model.

- So, what makes DL approach something better than the machine learning model is the addition of a hidden layer, which gives some flexibility and power, to increase accuracy.

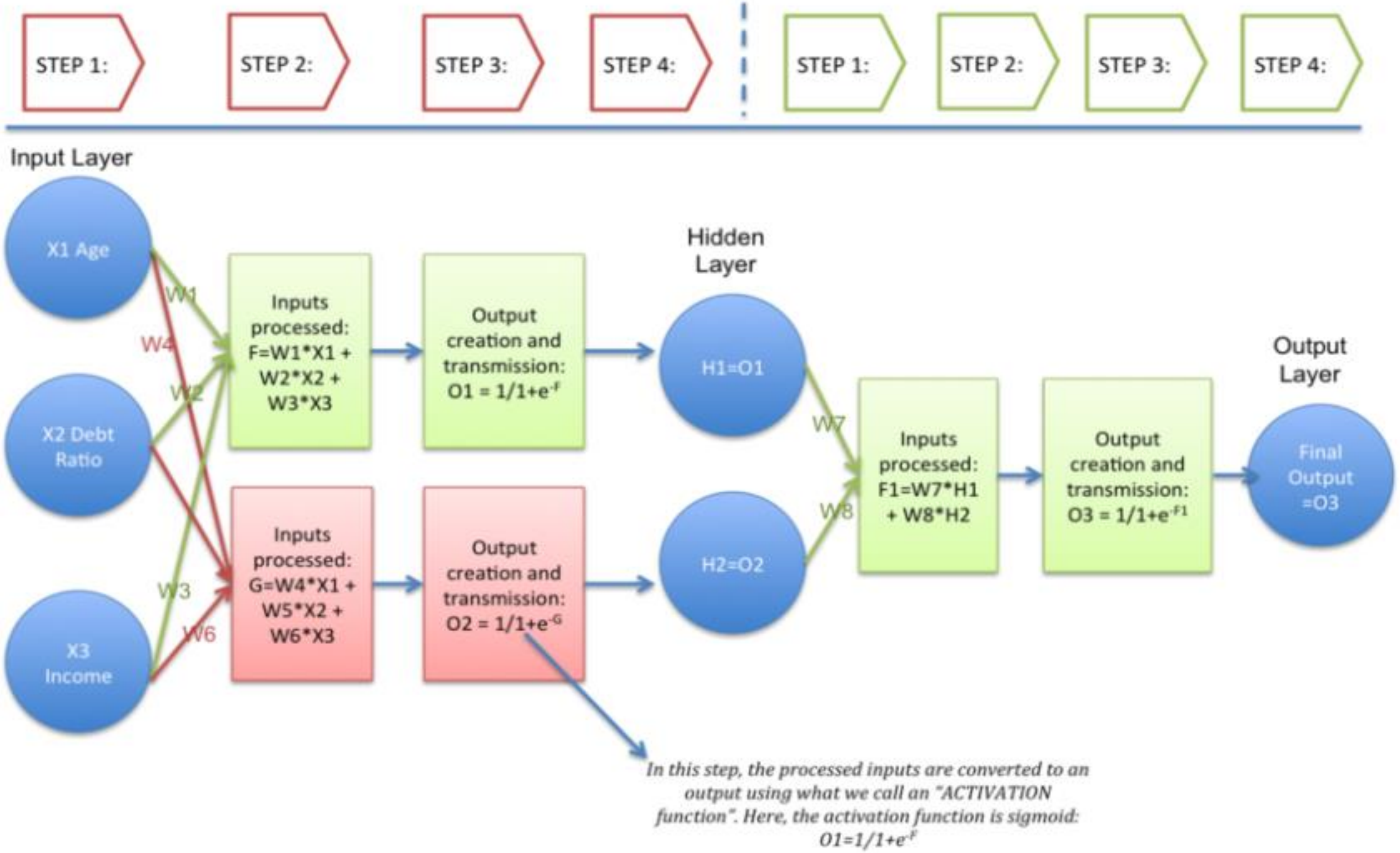
Meta Brains

How a network works – Overall Summary (3/4)

- Let us consider estimating the cost of the house for the same example with ANN. All input neurons are connected to all the neurons in the hidden layer, which means it won't have the same weightage from the input layer, which means depending on the priorities, the weightage value changes. Each neuron in the hidden layer has some priorities and weightage for application-related like how our brain works.

- If we keep on increasing the hidden layers for some importance or weightage or for solving complex calculations, the accuracy gets increased; it's like fine-tuning the network to get the optimal output.

How a network works – Overall Summary (4/4)



Introduction to Deep Learning

Artificial Neural Networks (ANN)

Propagation of Information in ANNs

Neural Network Architectures

Activation Functions

Gradient Descent Algorithm

Summary-Overview of Neural Networks

Implementation of ANN in Python

Convolutional Neural Networks (CNN)

Implementation of CNN in Python

01 How artificial neural networks work

02 Advantages of neural networks

03 Disadvantages of neural networks

04 Applications of neural networks

Advantages of ANNs

1. Ability to work with incomplete knowledge
2. Fault tolerance
3. Parallel processing capability

Introduction to Deep Learning

Artificial Neural Networks (ANN)

Propagation of Information in ANNs

Neural Network Architectures

Activation Functions

Gradient Descent Algorithm

Summary-Overview of Neural Networks

Implementation of ANN in Python

Convolutional Neural Networks (CNN)

Implementation of CNN in Python

01 How artificial neural networks work

02 Advantages of neural networks

03 Disadvantages of neural networks

04 Applications of neural networks

Disadvantages of ANNs

1. Hardware dependence

2. Determination of proper network structure

01 How artificial neural networks work

02 Advantages of neural networks

03 Disadvantages of neural networks

04 Applications of neural networks

Applications

1. Handwritten/handwriting recognition
2. Image compression
3. Stock exchange prediction

Quiz Time

1.What is backpropagation?

a) It is another name given to the curvy function in perceptron
b) It is the transmission of error back through the network to adjust the input
c) It is the transmission of error back through the network to allow weights to be adjusted so that the network can learn
d) None of the above

Quiz Time

1.What is backpropagation?

a) It is another name given to the curvy function in perceptron
b) It is the transmission of error back through the network to adjust the input
c) It is the transmission of error back through the network to allow weights to be adjusted so that the network can learn
d) None of the above

Introduction to Deep Learning

Artificial Neural Networks (ANN)

Propagation of information in ANNs

Neural Network Architectures

Activation Functions

Gradient Descent Algorithm

Summary-Overview of Neural Networks

Implementation of ANN in Python

Convolutional Neural Networks (CNN)

Implementation of CNN in Python

11. Building the Artificial Neural Network
12. Adding the input layer and the first hidden layer
13. Adding the next hidden layer
14. Adding the output layer
15. Compiling the artificial neural network
16. Fitting the ANN model to the training set
17. Predicting the test set results

ANN Implementation in Python

For implementation, I am going to use the **Churn Modeling Dataset**. You can download the dataset from **Kaggle**.

Artificial Neural Network can be used for **both classification and regression**. Here, we are going to use **ANN for classification**.

Meta Brains

Meta Brains

ANN Implementation in Python

This dataset has the following features:

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
1	15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101348.88	1
2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
3	15619304	Onio	502	France	Female	42	8	159660.8	3	1	0	113931.57	1
4	15701354	Boni	699	France	Female	39	1	0	2	0	0	93826.63	0
5	15737888	Mitchell	850	Spain	Female	43	2	125510.8	1	1	1	79084.1	0
6	15574012	Chu	645	Spain	Male	44	8	113755.8	2	1	0	149756.71	1
7	15592531	Bartlett	822	France	Male	50	7	0	2	1	1	10062.8	0
8	15656148	Obinna	376	Germany	Female	29	4	115046.7	4	1	0	119346.88	1
9	15792365	He	501	France	Male	44	4	142051.1	2	0	1	74940.5	0
10	15592389	H?	684	France	Male	27	2	134603.9	1	1	1	71725.73	0
11	15767821	Bearce	528	France	Male	31	6	102016.7	2	0	0	80181.12	0
12	15737173	Andrews	497	Spain	Male	24	3	0	2	1	0	76390.01	0
13	15632264	Kay	476	France	Female	34	10	0	2	1	0	26260.98	0
14	15691483	Chin	549	France	Female	25	5	0	2	0	0	190857.79	0
15	15600882	Scott	635	Spain	Female	35	7	0	2	1	1	65951.65	0
16	15643966	Goforth	616	Germany	Male	45	3	143129.4	2	0	1	64327.26	0
17	15737452	Romeo	653	Germany	Male	58	1	132602.9	1	1	0	5097.67	1
18	15788218	Henderso	549	Spain	Female	24	9	0	2	1	1	14406.41	0
19	15661507	Muldrow	587	Spain	Male	45	6	0	1	0	0	158684.81	0
20	15568982	Hao	726	France	Female	24	6	0	2	1	1	54724.03	0
21	15577657	McDonald	732	France	Male	41	8	0	2	1	1	170886.17	0
22	15597945	Dellucci	636	Spain	Female	32	8	0	2	1	0	138555.46	0
23	15699309	Gerasimov	510	Spain	Female	38	4	0	1	1	0	118913.53	1
24	15725737	Mosman	669	France	Male	46	3	0	2	0	1	8487.75	0
25	15625047	Yen	846	France	Female	38	5	0	1	1	1	187616.16	0
26	15738191	Maclean	577	France	Male	25	3	0	2	0	1	124508.29	0

Churn_Modelling

- Introduction to Deep Learning
- Artificial Neural Networks (ANN)
- Propagation of Information in ANNs
- Neural Network Architectures
- Activation Functions
- Gradient Descent Algorithm
- Summary-Overview of Neural Networks
- **Implementation of ANN in Python**
- Convolutional Neural Networks (CNN)
- Implementation of CNN in Python

01 Introduction

02 Exploring the dataset

03 Problem statement

04 Data pre-processing

05 Loading the dataset

06 Splitting the dataset into independent and dependent variables

07 Label encoding using scikit-learn

08 One-hot encoding using scikit-learn

09 Training and Test Sets: Splitting Data

10 Feature scaling

ANN Implementation in Python

PROBLEM STATEMENT

Meta Brains

Data Preprocessing

In data preprocessing, the first step is
Import the Libraries:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Introduction to Deep Learning

Artificial Neural Networks (ANN)

Propagation of Information in ANNs

Neural Network Architectures

Activation Functions

Gradient Descent Algorithm

Summary-Overview of Neural Networks

Implementation of ANN in Python

Convolutional Neural Networks (CNN)

Implementation of CNN in Python

01 Introduction

02 Exploring the dataset

03 Problem statement

04 Data pre-processing

05 Loading the dataset

06 Splitting the dataset into independent and dependent variables

07 Label encoding using scikit-learn

08 One-hot encoding using scikit-learn

09 Training and Test Sets: Splitting Data

10 Feature scaling

Data Preprocessing

Load the dataset:

```
dataset = pd.read_csv('Churn_Modelling.csv')
```

So, when you load the dataset after running this line of code, you will get your data as follows:

dataset - DataFrame

Index	towNumbe	Customerlc	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	mOfProdu	HasCrCard	ctiveMeml	imatedSal	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101349	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.9	1	0	1	112543	0
2	3	15619304	Onio	502	France	Female	42	8	159661	3	1	0	113932	1
3	4	15701354	Boni	699	France	Female	39	1	0	2	0	0	93826.6	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125511	1	1	1	79084.1	0
5	6	15574012	Chu	645	Spain	Male	44	8	113756	2	1	0	149757	1
6	7	15592531	Bartlett	822	France	Male	50	7	0	2	1	1	10062.8	0
7	8	15656148	Obinna	376	Germany	Female	29	4	115047	4	1	0	119347	1
8	9	15792365	He	501	France	Male	44	4	142051	2	0	1	74940.5	0
9	10	15592389	H?	684	France	Male	27	2	134604	1	1	1	71725.7	0
10	11	15767821	Bearce	528	France	Male	31	6	102017	2	0	0	80181.1	0
11	12	15737173	Andrews	497	Spain	Male	24	3	0	2	1	0	76390	0
12	13	15632264	Kay	476	France	Female	34	10	0	2	1	0	26261	0
13	14	15691483	Chin	549	France	Female	25	5	0	2	0	0	190858	0
14	15	15600882	Scott	635	Spain	Female	35	7	0	2	1	1	65951.6	0
15	16	15643966	Goforth	616	Germany	Male	45	3	143129	2	0	1	64327.3	0
16	17	15737452	Romeo	653	Germany	Male	58	1	132603	1	1	0	5097.67	1
17	18	15788218	Henderson	549	Spain	Female	24	9	0	2	1	1	14406.4	0
18	19	15661507	Muldrow	587	Spain	Male	45	6	0	1	0	0	158685	0
19	20	15568982	Hao	726	France	Female	24	6	0	2	1	1	54724	0
20	21	15577657	McDonald	732	France	Male	41	8	0	2	1	1	170886	0

Meta Brains

- Introduction to Deep Learning
- Artificial Neural Networks (ANN)
- Propagation of Information in ANNs
- Neural Network Architectures
- Activation Functions
- Gradient Descent Algorithm
- Summary-Overview of Neural Networks
- **Implementation of ANN in Python**
- Convolutional Neural Networks (CNN)
- Implementation of CNN in Python

01 Introduction

02 Exploring the dataset

03 Problem statement

04 Data pre-processing

05 Loading the dataset

06 Splitting the dataset into independent and dependent variables

07 Label encoding using scikit-learn

08 One-hot encoding using scikit-learn

09 Training and Test Sets: Splitting Data

10 Feature scaling

Data Preprocessing

Split the dataset into X and Y:

```
X = dataset.iloc[:, 3:13].values
y = dataset.iloc[:, 13].values
```

Meta Brains

Data Preprocessing

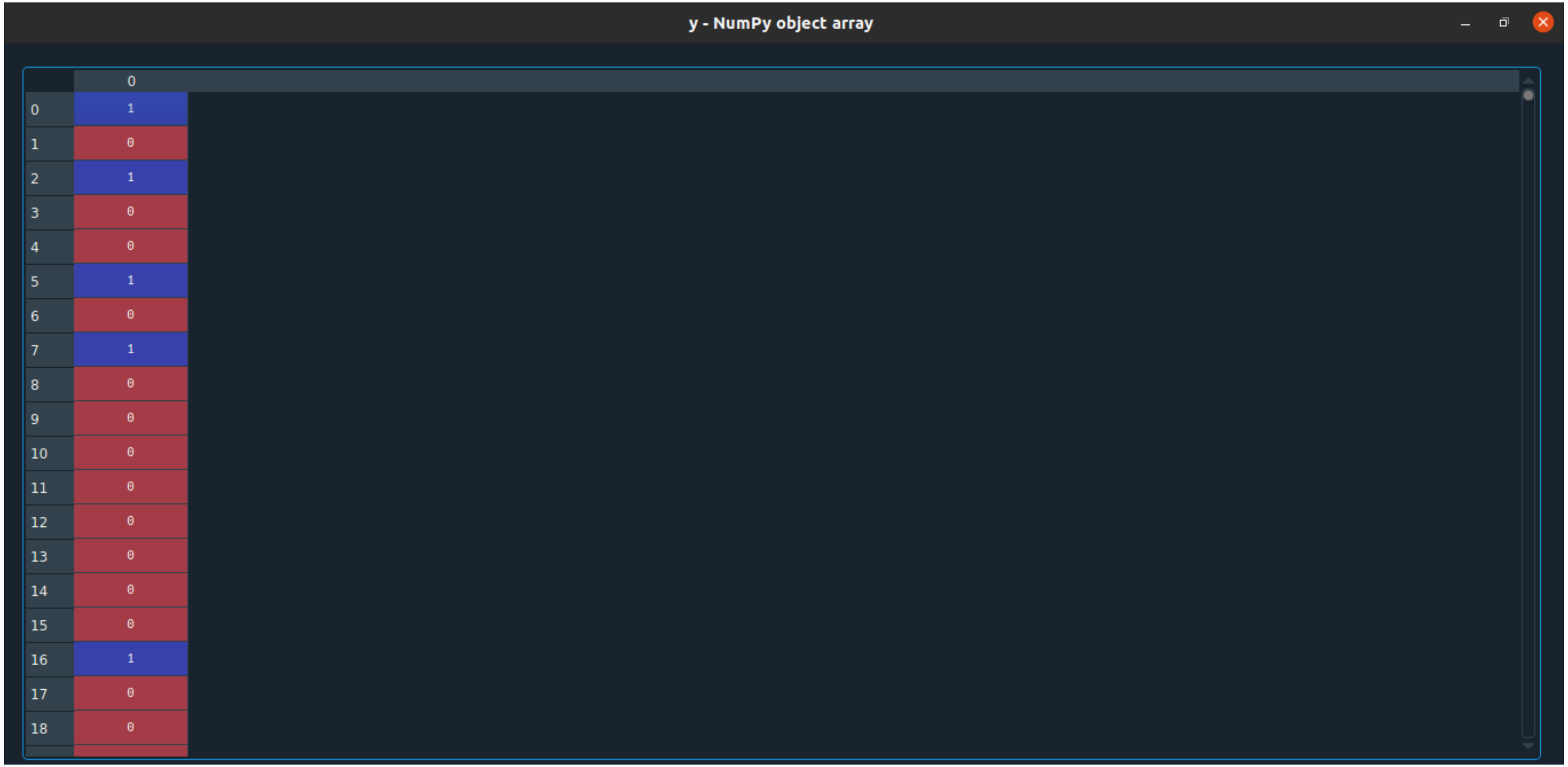
Independent Variable (X):

X - NumPy object array (read only)

	0	1	2	3	4	5	6	7	8	9
0	619	France	Female	42	2	0.0	1	1	1	101348.88
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58
2	502	France	Female	42	8	159660.8	3	1	0	113931.57
3	699	France	Female	39	1	0.0	2	0	0	93826.63
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.1
5	645	Spain	Male	44	8	113755.78	2	1	0	149756.71
6	822	France	Male	50	7	0.0	2	1	1	10062.8
7	376	Germany	Female	29	4	115046.74	4	1	0	119346.88
8	501	France	Male	44	4	142051.07	2	0	1	74940.5
9	684	France	Male	27	2	134603.88	1	1	1	71725.73
10	528	France	Male	31	6	102016.72	2	0	0	80181.12
11	497	Spain	Male	24	3	0.0	2	1	0	76390.01
12	476	France	Female	34	10	0.0	2	1	0	26260.98
13	549	France	Female	25	5	0.0	2	0	0	190857.79
14	635	Spain	Female	35	7	0.0	2	1	1	65951.65
15	616	Germany	Male	45	3	143129.41	2	0	1	64327.26
16	653	Germany	Male	58	1	132602.88	1	1	0	5097.67
17	549	Spain	Female	24	9	0.0	2	1	1	14406.41
18	587	Spain	Male	45	6	0.0	1	0	0	158684.81

Data Preprocessing

Dependent Variable (Y):



Data Preprocessing

Encode the categorical data:

First, let's perform label encoding for the gender

variable:

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X_2 = LabelEncoder()
X[:, 2] = labelencoder_X_2.fit_transform(X[:, 2])
```

Data Preprocessing

X - NumPy object array (read only)

	0	1	2	3	4	5	6	7	8	9
0	619	France	0	42	2	0.0	1	1	1	101348.88
1	608	Spain	0	41	1	83807.86	1	0	1	112542.58
2	502	France	0	42	8	159660.8	3	1	0	113931.57
3	699	France	0	39	1	0.0	2	0	0	93826.63
4	850	Spain	0	43	2	125510.82	1	1	1	79084.1
5	645	Spain	1	44	8	113755.78	2	1	0	149756.71
6	822	France	1	50	7	0.0	2	1	1	10062.8
7	376	Germany	0	29	4	115046.74	4	1	0	119346.88
8	501	France	1	44	4	142051.07	2	0	1	74940.5
9	684	France	1	27	2	134603.88	1	1	1	71725.73
10	528	France	1	31	6	102016.72	2	0	0	80181.12
11	497	Spain	1	24	3	0.0	2	1	0	76390.01
12	476	France	0	34	10	0.0	2	1	0	26260.98
13	549	France	0	25	5	0.0	2	0	0	190857.79
14	635	Spain	0	35	7	0.0	2	1	1	65951.65
15	616	Germany	1	45	3	143129.41	2	0	1	64327.26
16	653	Germany	1	58	1	132602.88	1	1	0	5097.67
17	549	Spain	0	24	9	0.0	2	1	1	14406.41
18	587	Spain	1	45	6	0.0	1	0	0	158684.81

Data Preprocessing

The next step is to perform one-hot encoding for the geography variable:

```
from sklearn.compose import ColumnTransformer
ct = ColumnTransformer([('ohe', OneHotEncoder(), [1])], remainder='passthrough')
X = np.array(ct.fit_transform(X), dtype = np.str) #.toarray()
X = X[:, 1:]
```

Data Preprocessing

X - NumPy object array

	0	1	2	3	4	5	6	7	8	9	10
0	0.0	0.0	619	0	42	2	0.0	1	1	1	101348.88
1	0.0	1.0	608	0	41	1	83807.86	1	0	1	112542.58
2	0.0	0.0	502	0	42	8	159660.8	3	1	0	113931.57
3	0.0	0.0	699	0	39	1	0.0	2	0	0	93826.63
4	0.0	1.0	850	0	43	2	125510.82	1	1	1	79084.1
5	0.0	1.0	645	1	44	8	113755.78	2	1	0	149756.71
6	0.0	0.0	822	1	50	7	0.0	2	1	1	10062.8
7	1.0	0.0	376	0	29	4	115046.74	4	1	0	119346.88
8	0.0	0.0	501	1	44	4	142051.07	2	0	1	74940.5
9	0.0	0.0	684	1	27	2	134603.88	1	1	1	71725.73
10	0.0	0.0	528	1	31	6	102016.72	2	0	0	80181.12
11	0.0	1.0	497	1	24	3	0.0	2	1	0	76390.01
12	0.0	0.0	476	0	34	10	0.0	2	1	0	26260.98
13	0.0	0.0	549	0	25	5	0.0	2	0	0	190857.79
14	0.0	1.0	635	0	35	7	0.0	2	1	1	65951.65
15	1.0	0.0	616	1	45	3	143129.41	2	0	1	64327.26
16	1.0	0.0	653	1	58	1	132602.88	1	1	0	5097.67
17	0.0	1.0	549	0	24	9	0.0	2	1	1	14406.41
18	0.0	1.0	587	1	45	6	0.0	1	0	0	158684.81

Data Preprocessing

Split the X and Y dataset into training set and test set:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = 0.2, random_state = 0)
```

Introduction to Deep Learning

Artificial Neural Networks (ANN)

Propagation of Information in ANNs

Neural Network Architectures

Activation Functions

Gradient Descent Algorithm

Summary-Overview of Neural Networks

Implementation of ANN in Python

Convolutional Neural Networks (CNN)

Implementation of CNN in Python

01 Introduction

02 Exploring the dataset

03 Problem statement

04 Data pre-processing

05 Loading the dataset

06 Splitting the dataset into independent and dependent variables

07 Label encoding using scikit-learn

08 One-hot encoding using scikit-learn

09 Training and Test Sets: Splitting Data

10 Feature scaling

Data Preprocessing

Perform feature scaling:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Data Preprocessing

	0	1	2	3	4	5	6	7	8	9	10
0	-0.569844	1.74309	0.169582	-1.09169	-0.464608	0.00666099	-1.21572	0.809503	0.642595	-1.03227	1.10643
1	1.75487	-0.573694	-2.30456	0.916013	0.301026	-1.37744	-0.00631193	-0.921591	0.642595	0.968738	-0.748664
2	-0.569844	-0.573694	-1.1912	-1.09169	-0.943129	-1.03142	0.579935	-0.921591	0.642595	-1.03227	1.48533
3	-0.569844	1.74309	0.0355658	0.916013	0.109617	0.00666099	0.473128	-0.921591	0.642595	-1.03227	1.27653
4	-0.569844	1.74309	2.05611	-1.09169	1.73659	1.04474	0.810193	0.809503	0.642595	0.968738	0.558378
5	1.75487	-0.573694	1.29325	-1.09169	-0.177495	-1.03142	0.442535	0.809503	0.642595	-1.03227	1.63252
6	-0.569844	-0.573694	1.61283	0.916013	0.779547	-1.37744	0.304328	-0.921591	-1.55619	-1.03227	0.481496
7	-0.569844	1.74309	-0.541734	0.916013	0.205321	1.04474	-1.21572	0.809503	0.642595	0.968738	1.07382
8	-0.569844	1.74309	-0.149995	0.916013	3.55497	1.39076	0.80633	-0.921591	0.642595	0.968738	-1.0495
9	-0.569844	-0.573694	-0.29432	-1.09169	-0.656016	0.352686	1.48636	0.809503	0.642595	-1.03227	0.0153936
10	-0.569844	-0.573694	0.324216	-1.09169	-0.560312	1.04474	-0.017786	-0.921591	0.642595	0.968738	-1.17149
11	-0.569844	-0.573694	0.612865	0.916013	1.44948	0.352686	1.5191	-0.921591	0.642595	0.968738	1.16193
12	1.75487	-0.573694	-0.58297	-1.09169	-0.943129	-0.68539	0.874975	-0.921591	0.642595	-1.03227	-0.680001
13	-0.569844	1.74309	1.49943	0.916013	0.205321	1.04474	0.502554	-0.921591	0.642595	-1.03227	-1.41935
14	1.75487	-0.573694	-0.459262	0.916013	-0.0817912	-0.68539	0.380665	-0.921591	-1.55619	-1.03227	-1.09703
15	-0.569844	-0.573694	-0.222157	0.916013	0.492434	0.00666099	-1.21572	4.27169	-1.55619	-1.03227	0.302431
16	-0.569844	-0.573694	-1.25305	0.916013	0.301026	-1.37744	1.30115	-0.921591	0.642595	0.968738	-0.311041
17	-0.569844	-0.573694	1.68499	0.916013	-0.751721	-1.37744	0.683863	-0.921591	0.642595	-1.03227	0.0266257
18	-0.569844	-0.573694	-0.108759	-1.09169	-0.751721	1.39076	1.00325	-0.921591	0.642595	-1.03227	-0.895658

Building the Artificial Neural Network

The first step is **Import the Keras libraries and packages:**

```
from tensorflow.keras.models import Sequential
#from keras.models import Sequential
from keras.layers import Dense
```

Initialize the Artificial Neural Network:

```
classifier = Sequential()
```

Building the Artificial Neural Network

Add the input layer and the first hidden layer:

```
classifier.add(Dense(units = 6, kernel_initializer = 'uniform', activation = 'relu', input_dim = 11))
```

Meta Brains

Introduction to Deep Learning

Artificial Neural Networks (ANN)

Propagation of Information in ANNs

Neural Network Architectures

Activation Functions

Gradient Descent Algorithm

Summary-Overview of Neural Networks

Implementation of ANN in Python

Convolutional Neural Networks (CNN)

Implementation of CNN in Python

11 Building the Artificial Neural Network

12 Adding the input layer and the first hidden layer

13 Adding the next hidden layer

14 Adding the output layer

15 Compiling the artificial neural network

16 Fitting the ANN model to the training set

17 Predicting the test set results

Building the Artificial Neural Network

Add the second hidden layer:

```
classifier.add(Dense(units = 6, kernel_initializer = 'uniform', activation = 'relu'))
```

Meta Brains

Building the Artificial Neural Network

Add the output layer:

```
classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))
```

Introduction to Deep Learning

Artificial Neural Networks (ANN)

Propagation of Information in ANNs

Neural Network Architectures

Activation Functions

Gradient Descent Algorithm

Summary-Overview of Neural Networks

Implementation of ANN in Python

Convolutional Neural Networks (CNN)

Implementation of CNN in Python

11 Building the Artificial Neural Network

12 Adding the input layer and the first hidden layer

13 Adding the next hidden layer

14 Adding the output layer

15 Compiling the artificial neural network

16 Fitting the ANN model to the training set

17 Predicting the test set results

Train the ANN

The training part requires two steps: Compiling the ANN and fitting the ANN to the training set. So, let's start with the first step.

Compile the ANN:

```
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

Train the ANN

The next step is **Fit the ANN to the training set:**

```
classifier.fit(X_train, y_train, batch_size = 10, epochs = 100)
```

Train the ANN

In the first few epochs, the accuracy scores are as follows:

```
Epoch 1/100
800/800 [==============================] - 2s 2ms/step - loss: 0.5559 - accuracy: 0.7959
Epoch 2/100
800/800 [==============================] - 1s 2ms/step - loss: 0.4309 - accuracy: 0.8191
Epoch 3/100
800/800 [==============================] - 1s 2ms/step - loss: 0.4158 - accuracy: 0.8263
```

Which is around 80%, but after running all 100 epoch, the accuracy increase, and we get the final accuracy-

```
Epoch 98/100
800/800 [==============================] - 1s 2ms/step - loss: 0.3323 - accuracy: 0.8609
Epoch 99/100
800/800 [==============================] - 2s 2ms/step - loss: 0.3381 - accuracy: 0.8611
Epoch 100/100
800/800 [==============================] - 1s 2ms/step - loss: 0.3531 - accuracy: 0.8544: 0s - loss: 0.3555 - accura
```

That is 85.44%. Quite good. Now, we are done with the training part. The last but not least part is predicting the test set results.

Train the ANN

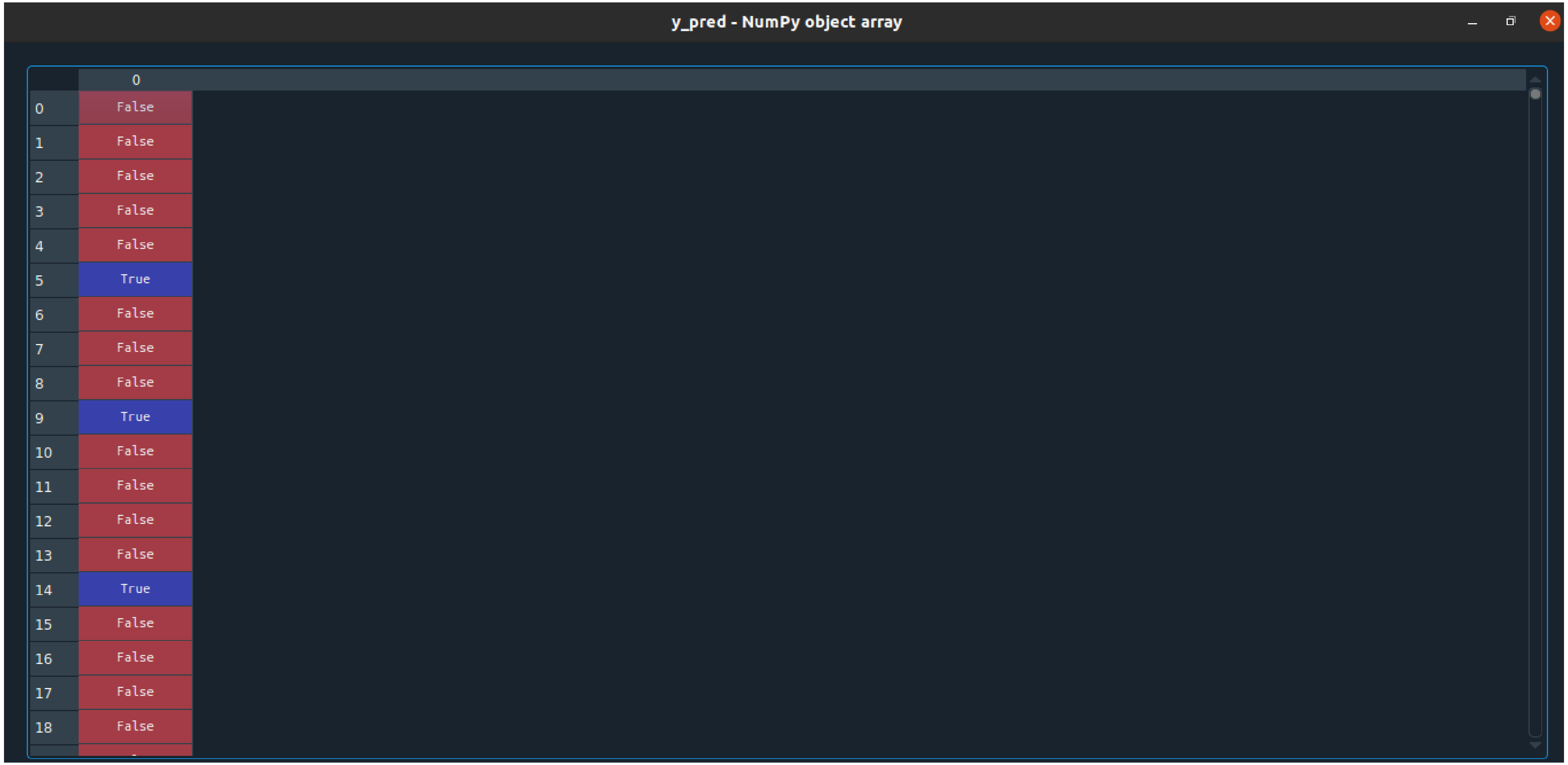
The next step is

Prediction:

```
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)
```

Predict the Test Set Results

So, after running this code, you will get y_pred, as follows:



Make the Confusion Matrix

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test,y_pred)
```

```
[[1499   96]
 [ 189  216]]

0.8575
```

And we get 85.75% accuracy.

Quiz Time

The number of nodes in the input layer is 10 and in the hidden layer is 5. The maximum number of connections from the input layer to the hidden layer are?

a) 50

b) Less than 50

c) More than 50

d) It is an arbitrary value

Quiz Time

The number of nodes in the input layer is 10 and in the hidden layer is 5. The maximum number of connections from the input layer to the hidden layer are?

a) 50

b) Less than 50

c) More than 50

d) It is an arbitrary value

Introduction to Deep Learning

Artificial Neural Networks (ANN)

Propagation of Information in ANNs

Neural Network Architectures

Activation Functions

Gradient Descent Algorithm

Summary-Overview of Neural Networks

Implementation of ANN in Python

Convolutional Neural Networks (CNN)

Implementation of CNN in Python

Introduction **01**

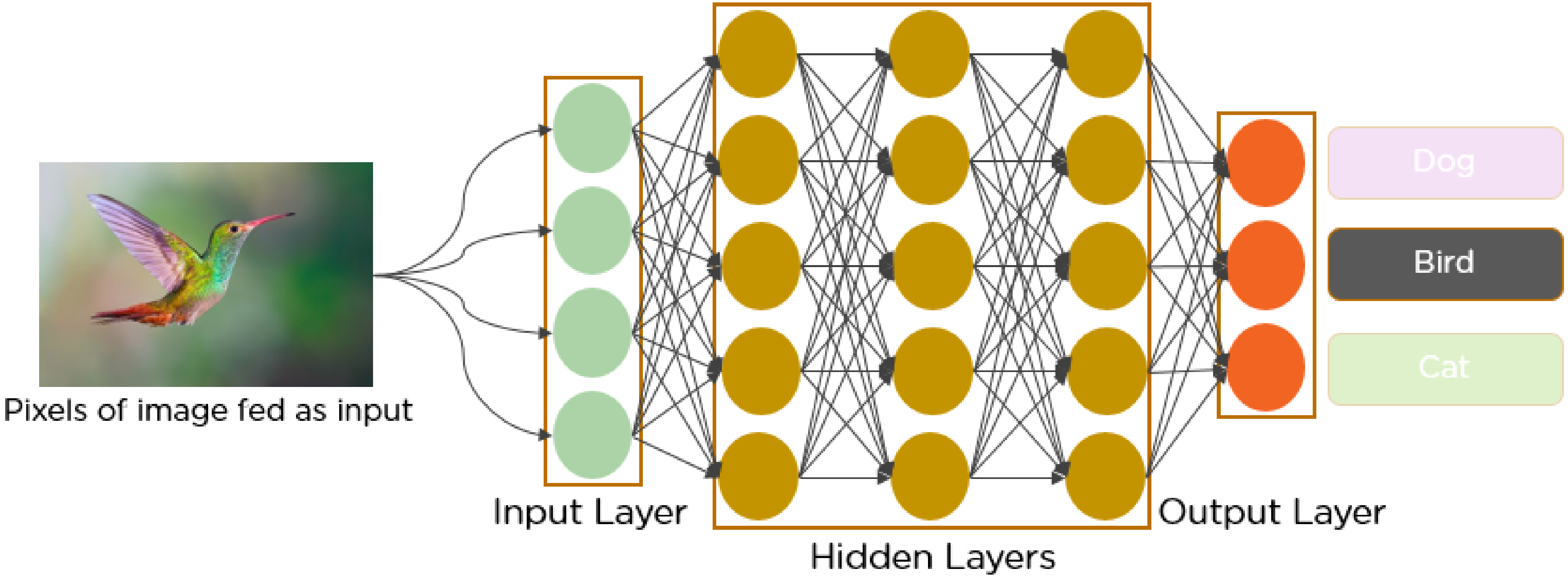
02 Components of convolutional neural networks

Convolution layer 03

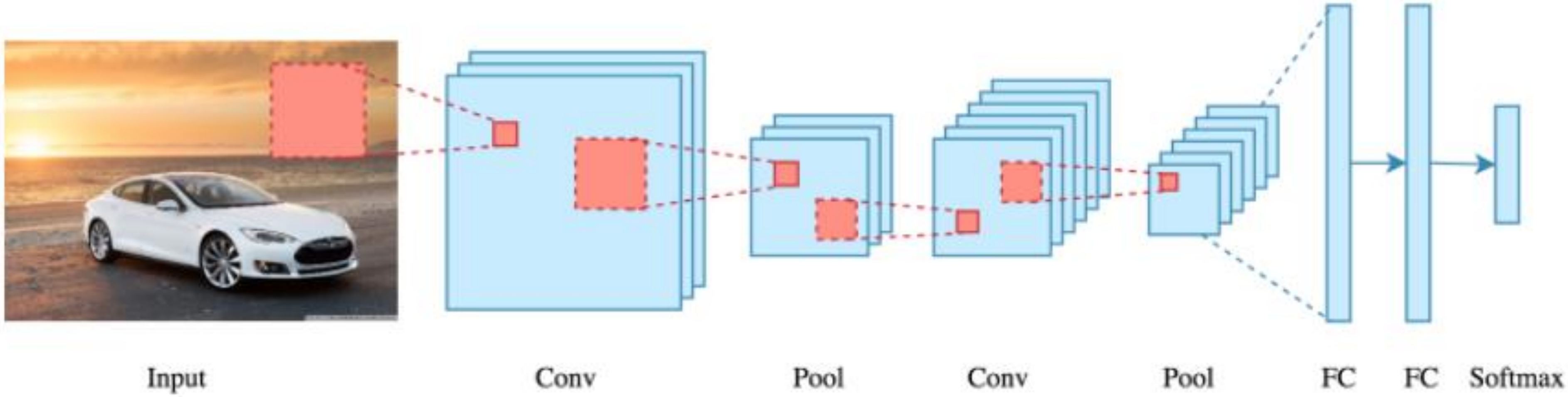
04 Pooling layer

Fully connected layer 05

CNN-Introduction



What exactly is a CNN



Bottom line is that the role of the ConvNet is to reduce the images into a form that is easier to process, without losing features that are critical for getting a good prediction.

Components of CNN

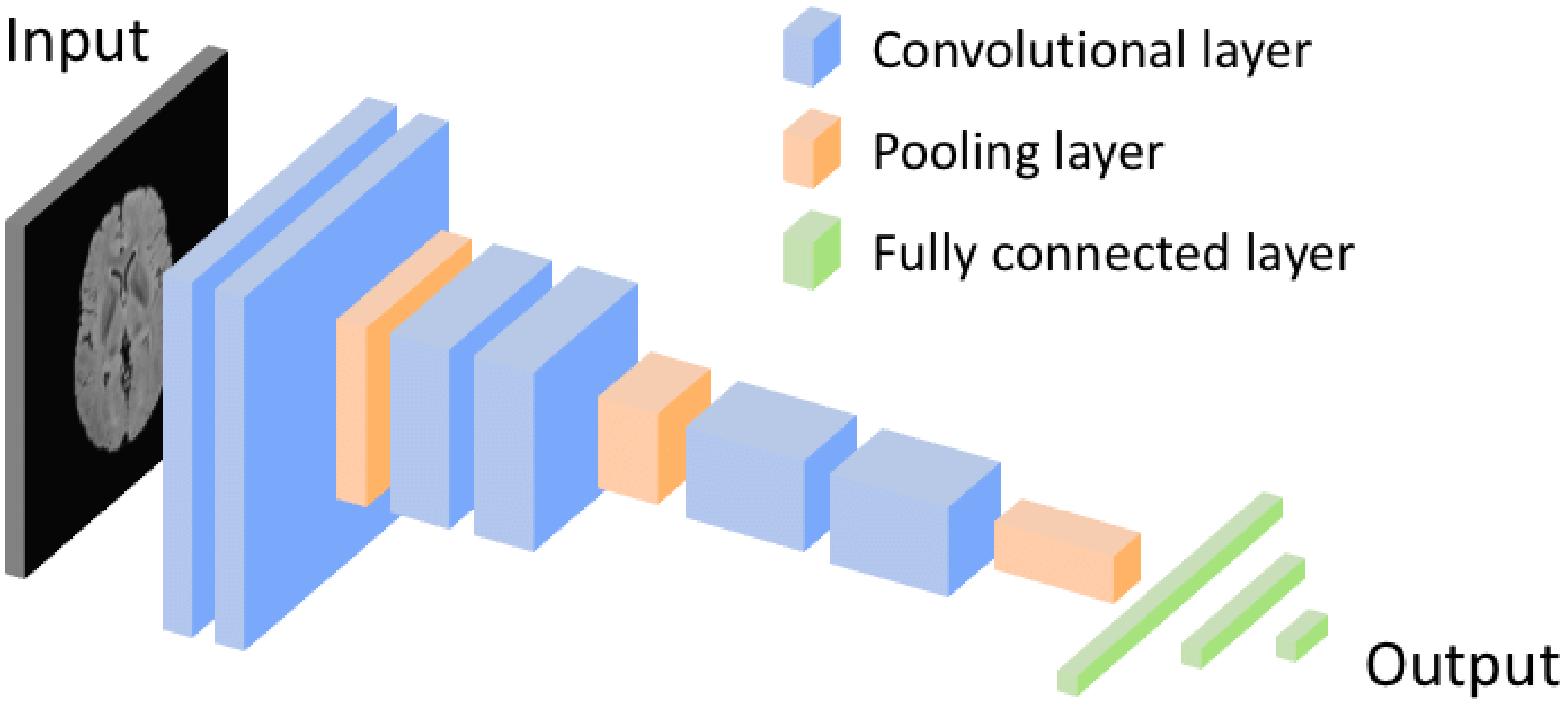
The CNN model works in two steps: **feature extraction and classification.**

Feature extraction is a phase where various filters and layers are applied to the images to extract the information and features out of it; once it's done, it is passed on to the next phase, i.e., **classification,** where they are classified based on the target variable of the problem.

A typical CNN model looks as follows:

- Input layer
- Convolution layer + activation function
- Pooling layer
- Fully connected layer

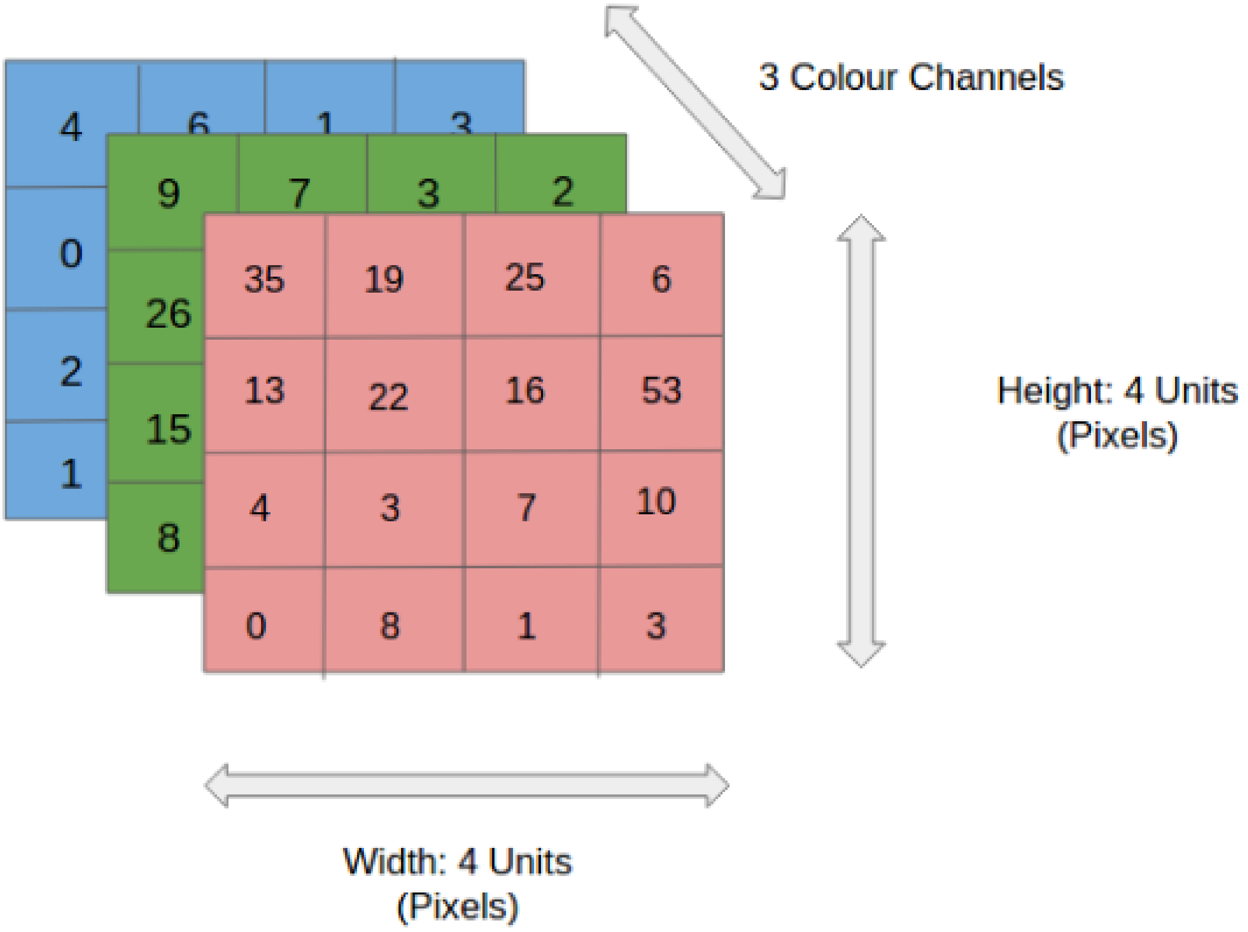
Components of CNN



Source: researchgate.net

Meta Brains

Input Layer

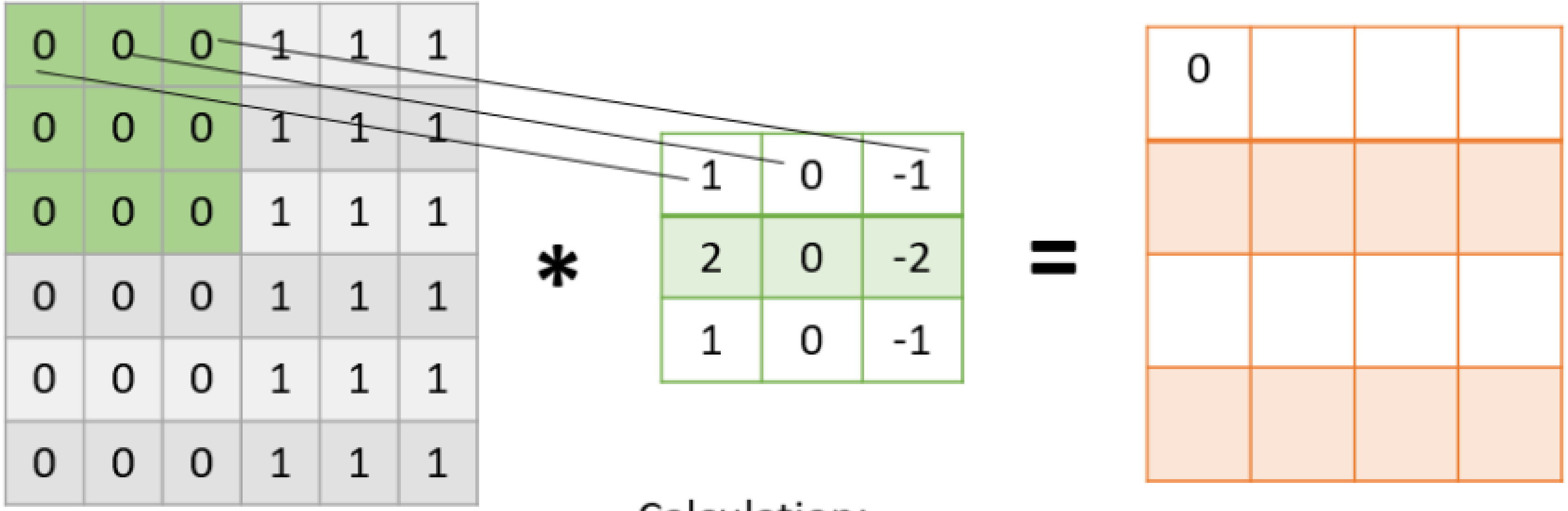

3 Colour Channels
Height: 4 Units
(Pixels)
Width: 4 Units
(Pixels)
35 19 25 6
13 22 16 53
4 3 7 10
0 8 1 3

Convolution Layer (1/4)

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}}_{6*6} * \underbrace{\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}}_{3*3} = \underbrace{\begin{bmatrix} 0 & -4 & -4 & 0 \\ 0 & -4 & -4 & 0 \\ 0 & -4 & -4 & 0 \\ 0 & -4 & -4 & 0 \end{bmatrix}}_{4*4}$$

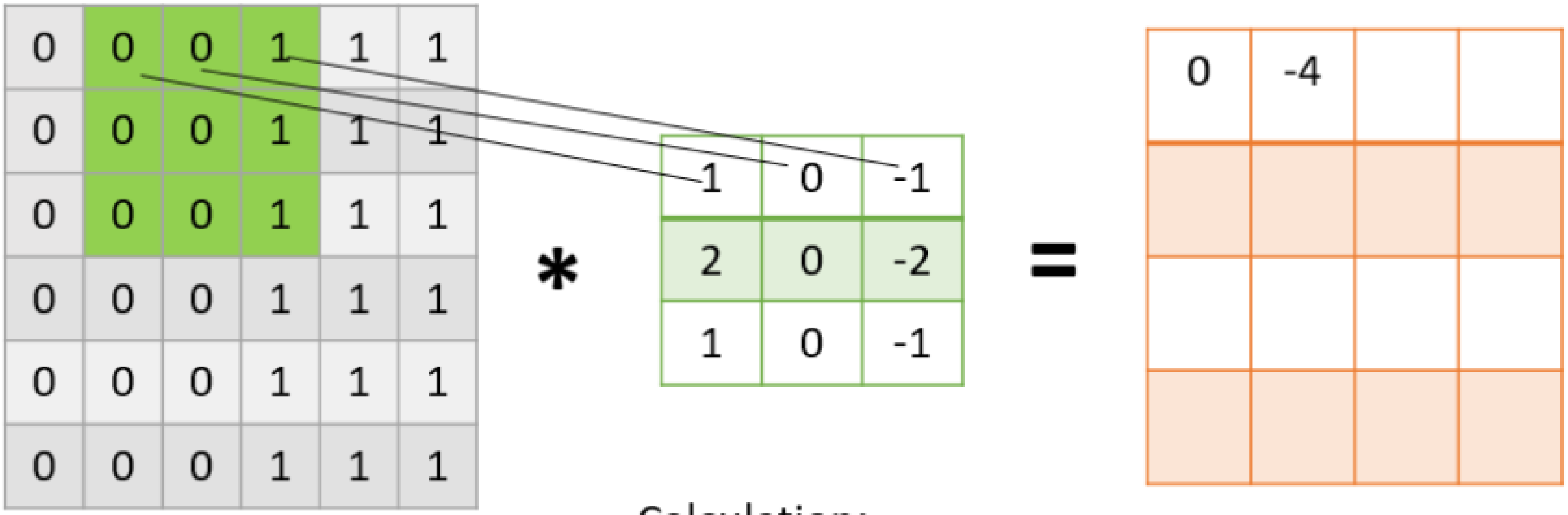
Convolution Layer (2/4)

The result of applying the filter to the image is that we get a feature map of 4*4.



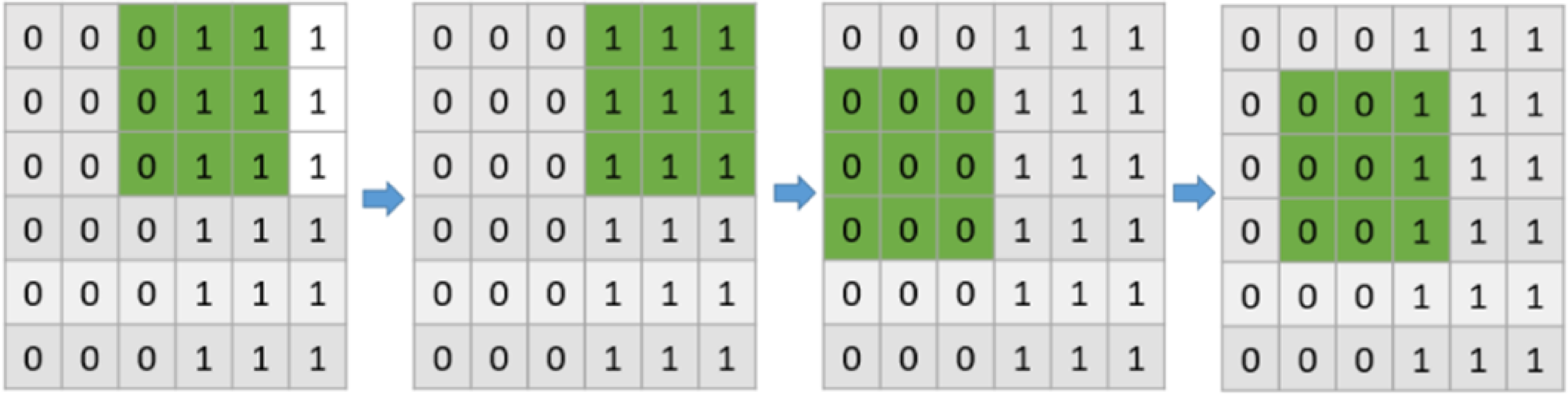
Calculation:
0*1 + 0*0 + 0*-1 +
0*2 + 0*0 + 0*-2 +
0*1 + 0*0 + 0*-1

Convolution Layer (3/4)



Calculation:
0*1 + 0*0 + 1*-1 +
0*2 + 0*0 + 1*-2 +
0*1 + 0*0 + 1*-1

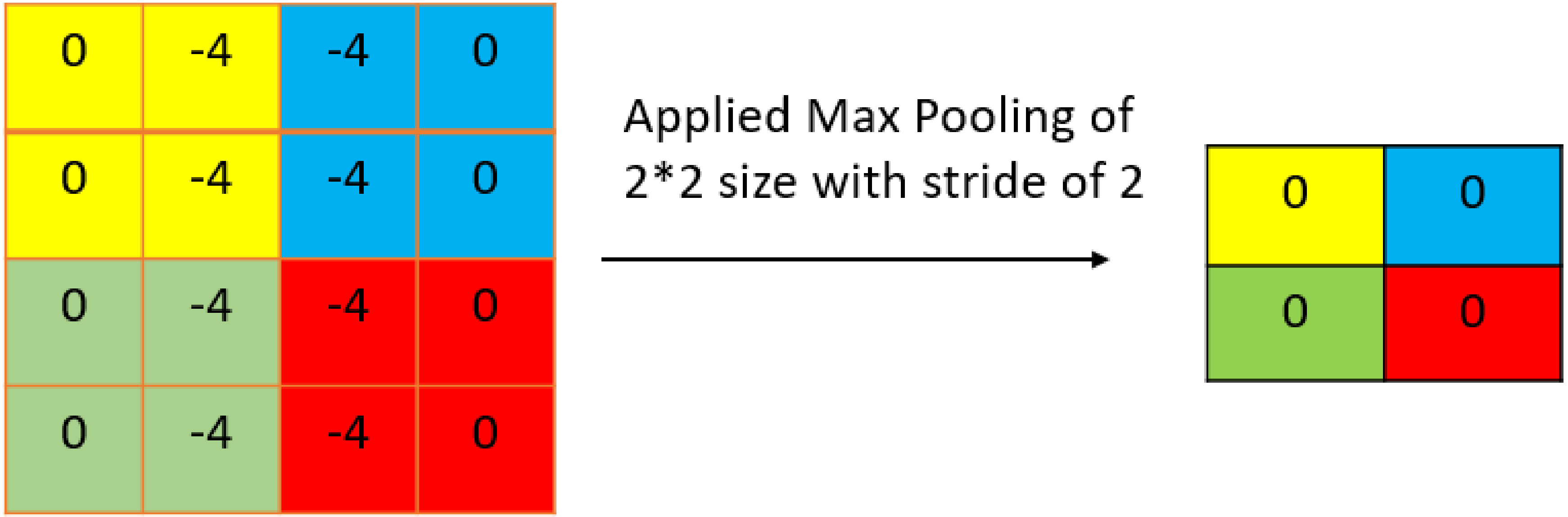
Convolution Layer (4/4)



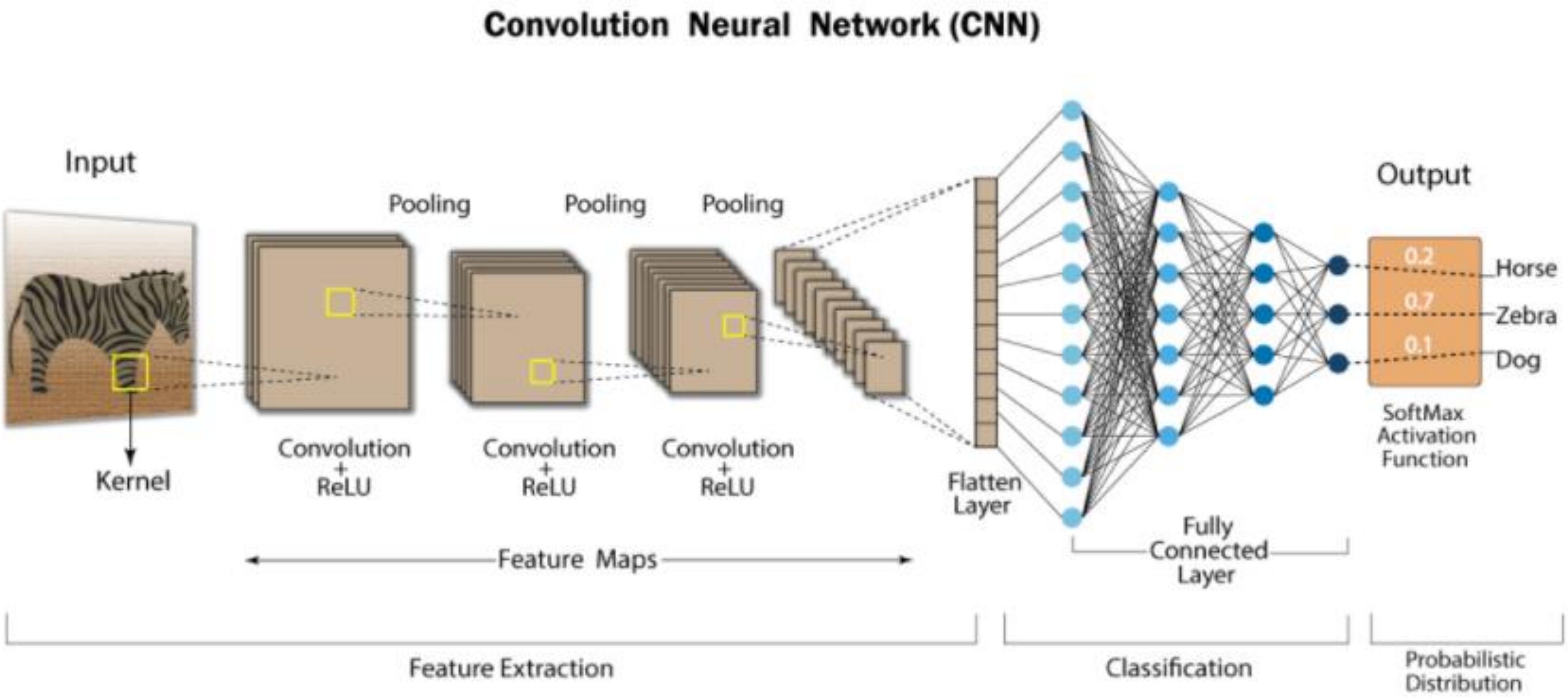
This is how a filter passes through the entire image with the stride of 1

Pooling Layer

The pooling layer is applied after the convolutional layer.
It is used to reduce the dimensions of feature maps. Thus, it reduces the number of parameters to learn. It preserves the important information or features of the input image.



Fully Connected Layer



Meta Brains

01 Dataset

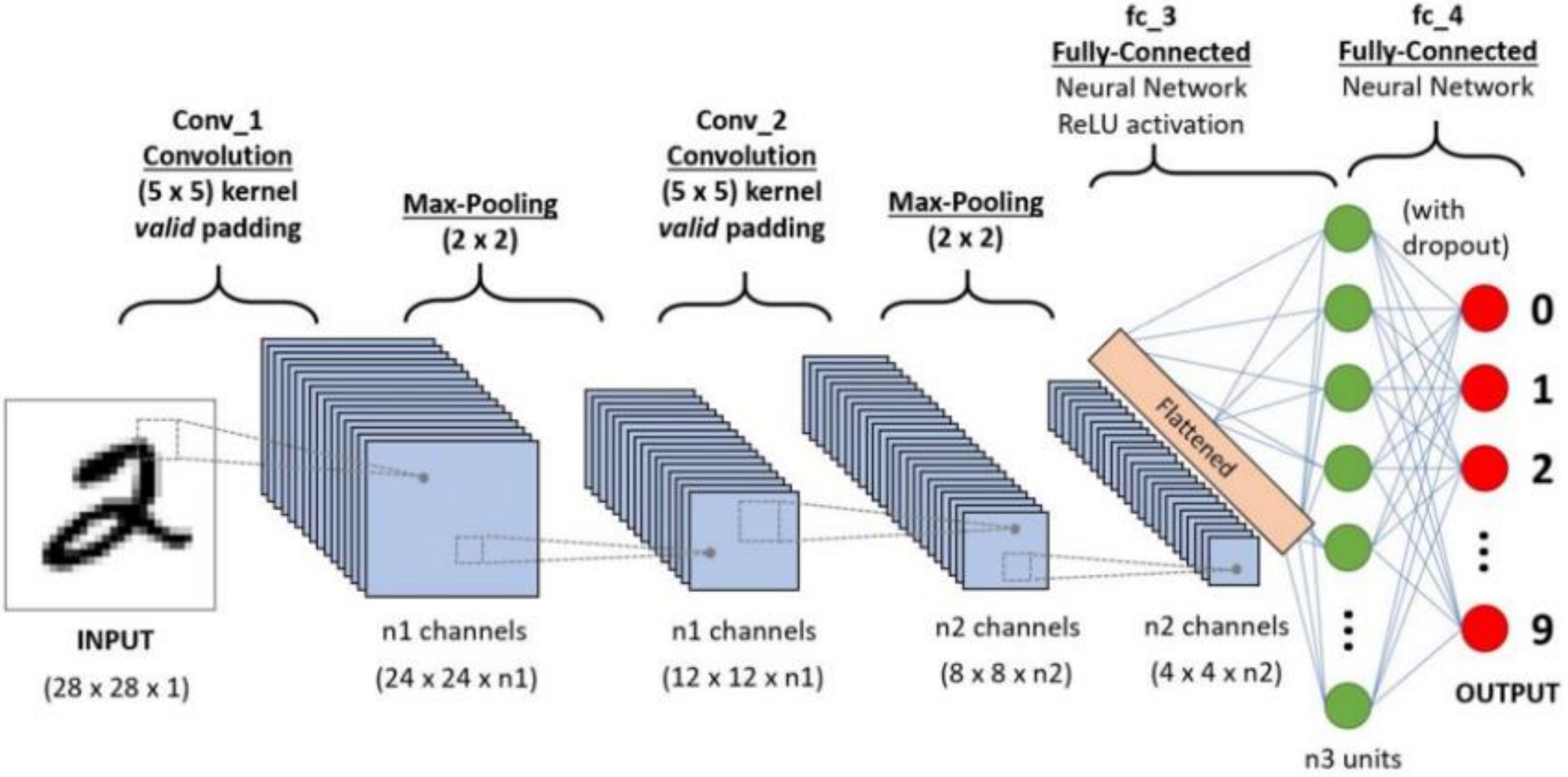
02 Importing libraries

03 Building the CNN model

04 Accuracy of the model

We are going to use MNIST Handwritten Digit data, which contains the images of handwritten digits.

Backaround of CNNs



Meta Brains

Implementation (1/5)

```python
#importing the required libraries
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPool2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Dense
```

Implementation (2/5)

```
#loading data
(X_train,y_train) , (X_test,y_test)=mnist.load_data()
#reshaping data
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], X_train.shape[2], 1))
X_test = X_test.reshape((X_test.shape[0],X_test.shape[1],X_test.shape[2],1))
#checking the shape after reshaping
print(X_train.shape)
print(X_test.shape)
#normalizing the pixel values
X_train=X_train/255
X_test=X_test/255
```

```
(60000, 28, 28, 1)
(10000, 28, 28, 1)
```

Implementation (3/5)

```python
#defining model
model=Sequential()
#adding convolution layer
model.add(Conv2D(32,(3,3),activation='relu',input_shape=(28,28,1)))
#adding pooling layer
model.add(MaxPool2D(2,2))
#adding fully connected layer
model.add(Flatten())
model.add(Dense(100,activation='relu'))
#adding output layer
model.add(Dense(10,activation='softmax'))
#compiling the model
model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
#fitting the model
model.fit(X_train,y_train,epochs=10)
```

Meta Brains

Implementation (4/5)

Output:

```
Epoch 1/10
1875/1875 [==============================] - 33s 16ms/step - loss: 0.3192 - accuracy: 0.9057
Epoch 2/10
1875/1875 [==============================] - 30s 16ms/step - loss: 0.0523 - accuracy: 0.9854
Epoch 3/10
1875/1875 [==============================] - 30s 16ms/step - loss: 0.0343 - accuracy: 0.9899
Epoch 4/10
1875/1875 [==============================] - 31s 16ms/step - loss: 0.0229 - accuracy: 0.9929
Epoch 5/10
1875/1875 [==============================] - 31s 16ms/step - loss: 0.0141 - accuracy: 0.9955
Epoch 6/10
1875/1875 [==============================] - 31s 16ms/step - loss: 0.0098 - accuracy: 0.9965
Epoch 7/10
1875/1875 [==============================] - 32s 17ms/step - loss: 0.0070 - accuracy: 0.9982
Epoch 8/10
1875/1875 [==============================] - 31s 17ms/step - loss: 0.0047 - accuracy: 0.9986
Epoch 9/10
1875/1875 [==============================] - 31s 17ms/step - loss: 0.0059 - accuracy: 0.9978
Epoch 10/10
1875/1875 [==============================] - 31s 17ms/step - loss: 0.0030 - accuracy: 0.9991
```

Implementation (5/5)

```
#evaluting the model
model.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 2s 6ms/step - loss: 0.0610 - accuracy: 0.9864

[0.060965631157159805, 0.9864000082015991]
```

Quiz Time

When a pooling layer is added to a convolutional neural network, translation in-variance is preserved. True or False?

a) True

b) False

Quiz Time

When a pooling layer is added to a convolutional neural network, translation in-variance is preserved. True or False?

a) True

b) False

THANK YOU

Meta Brains