Introduction to Neural Networks

Week 7 – Session 1

BATH SPA UNIVERSITY

Data Analytics and Machine Learning

Overview

10:00 – 12:30
LECTURE

1:00 – 1:30
LAB SESSION

1:30 – 2:00
SESSION WRAP UP

Lesson Objectives

- To be able to discuss the concept of neural networks
- Able to understand the structure of a neural network.
- Bonus topic – using APIs in Python and Power BI

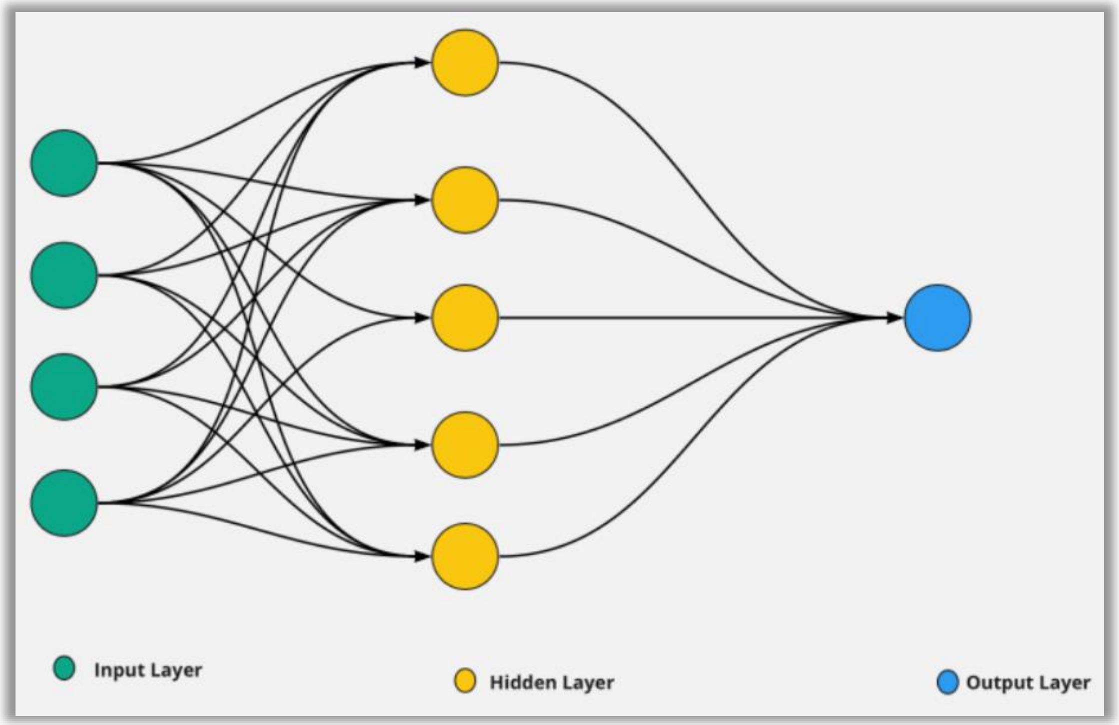
Keyword	Description
Data Science	An interdisciplinary field that uses scientific methods, processes, algorithms, and systems to extract knowledge and insights from structured and unstructured data.
Machine Learning	A type of artificial intelligence that enables computers to learn from data and make decisions or predictions without being explicitly programmed.
Supervised Learning	A type of machine learning where the model learns from labelled data, i.e., data where the target variable is known.
Regression	A type of supervised learning task where the goal is to predict a continuous target variable.
Classification	This is a type of supervised learning task where the goal is to predict a categorical target variable.
Unsupervised Learning	Unsupervised learning, also known as unsupervised machine learning, uses machine learning algorithms to analyse and cluster unlabelled datasets.
Clustering	Clustering or cluster analysis is a machine learning technique which groups the unlabelled dataset. It can be defined as "A way of grouping the data points into different clusters, consisting of similar data points. The objects with the possible similarities remain in a group that has less or no similarities with another group."
Dimensionality reduction	Dimensionality reduction is a machine learning (ML) or statistical technique of reducing the number of random variables in a problem by obtaining a set of principal variables.
Hyperparameter	These are parameters of the learning algorithm itself, not derived from the data, that need to be set before training the model.
Neural Network	A set of algorithms modelled after the human brain, designed to recognize patterns. They interpret sensory data through a kind of machine perception, labelling or clustering raw input.

Introduction to Neural Networks

Data Analytics and Machine Learning

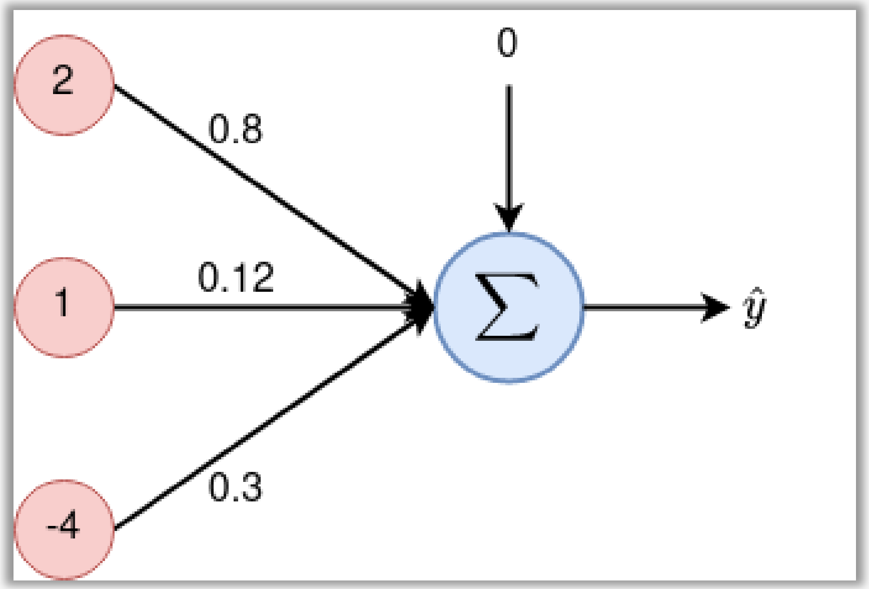
What is a Neural Network(NN)?

- A group of interconnected neurons that can influence each other’s behaviour
- Neural networks try to emulate the human brain
- Subset of machine learning at the heart of deep learning algorithms
- Structure is inspired by the human brain
- Imitates biological neurons (how they signal to one another)
- Perform mathematical operations to detect patterns in data



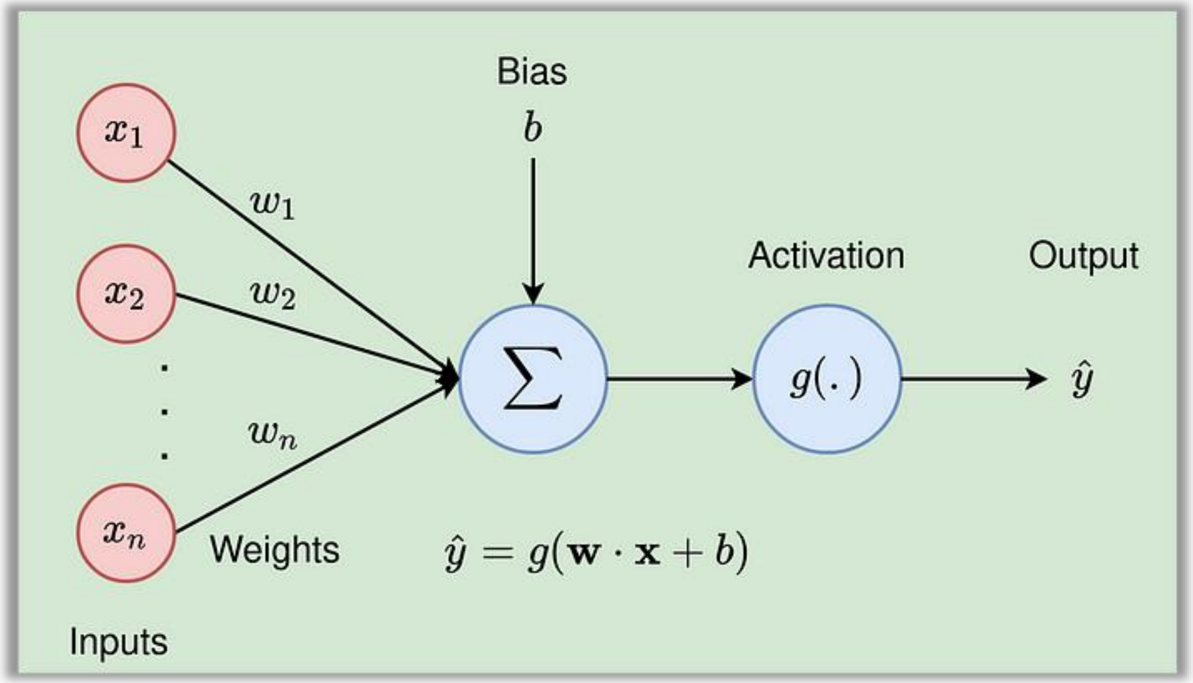
Neural Network Definitions

- **Neuron**
  - This is a basic building block of a NN
  - Takes weighted values
  - Performs mathematical calculation
  - Produces an output
  - Also called unit, node or perceptron
- **Inputs**
  - The data or values passed to the neurons
- **Deep Neural Network (DNN)**
  - Neural Network with many hidden layers between the input (first layer) and the output (last layer)



Neural Network Definitions

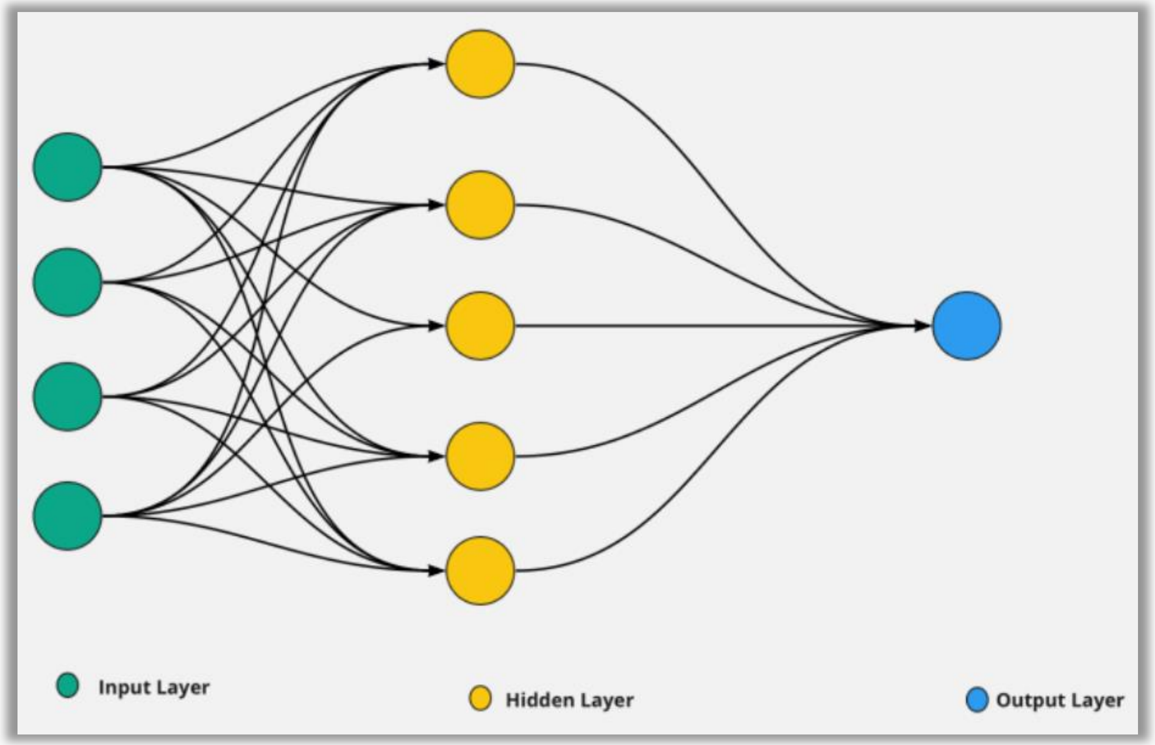
- **Weights**
  - Degree of importance of connection between any two neurons
  - Strength of connection
- **Bias**
  - Constant value added to the sum of the product between input values and respective weights
  - Used to accelerate or delay the activation of a given node
- **Activation function**
  - Introduce the non-linearity phenomenon into the NN system
  - Allows the network to learn more complex patterns



Note: Network learns patterns by tuning weights and biases to get the best predictions

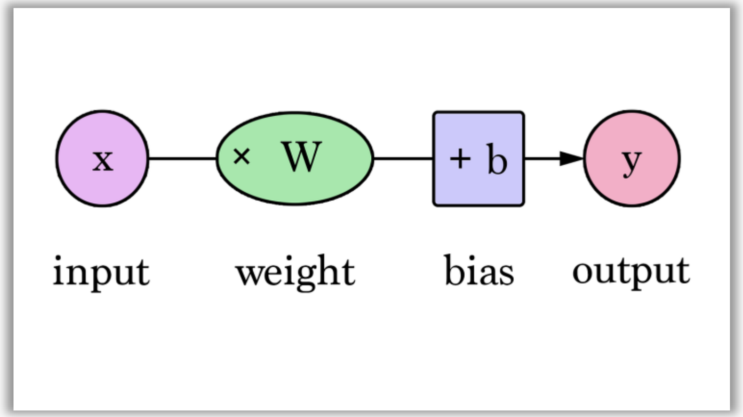
Structure of Neural Networks

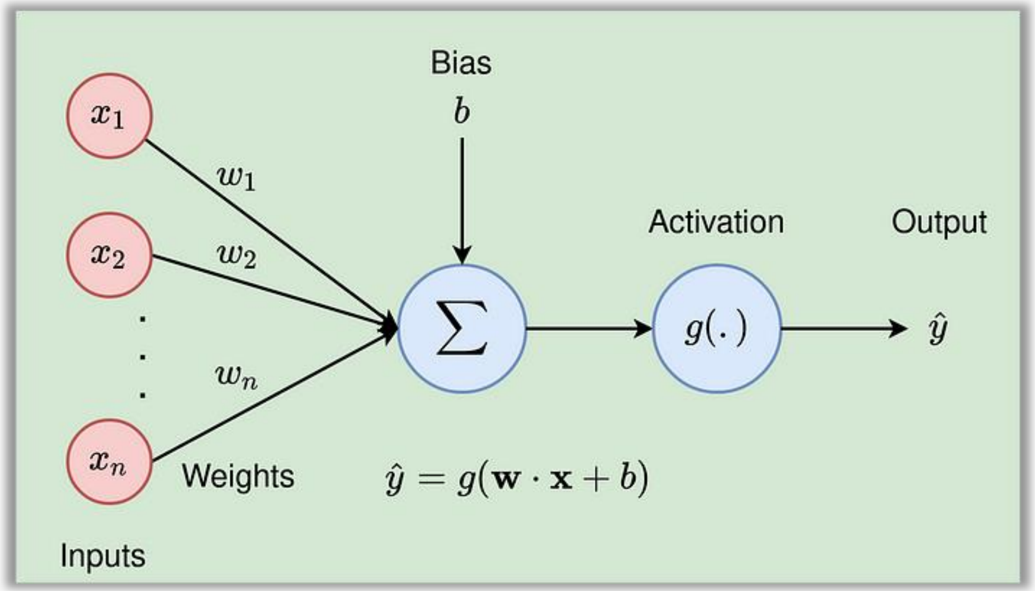
- Use case: Deep learning algorithm for assessing images
  - Identifies objects in images
- **Layers**
  - Composed of input, output, and hidden layers
  - Layers alter data to learn specific features
- **Neurons**
  - Core of NNs with adjustable weights and biases
  - Organised into layers performing specific tasks
  - Input layer receives information from the outside world
- **Convolution**
  - CNNs use convolutions for efficient data analysis
  - Extracts meaningful features from images or sound waves
- **Learning Process**
  - Successive layers have fewer parameters
  - Algorithm learns more by analysing smaller data chunks
  - Identifies patterns, allowing it to learn from images or video



How Neural Networks Work

- Think of each individual **node** as its own linear regression model
  - Input data
  - Weights
  - A bias or threshold
  - An output
- **Output** is passed through an activation function
  - Determines the output
- If output exceeds a given threshold
  - '**Activates**' the node, passing data to the next layer in the network



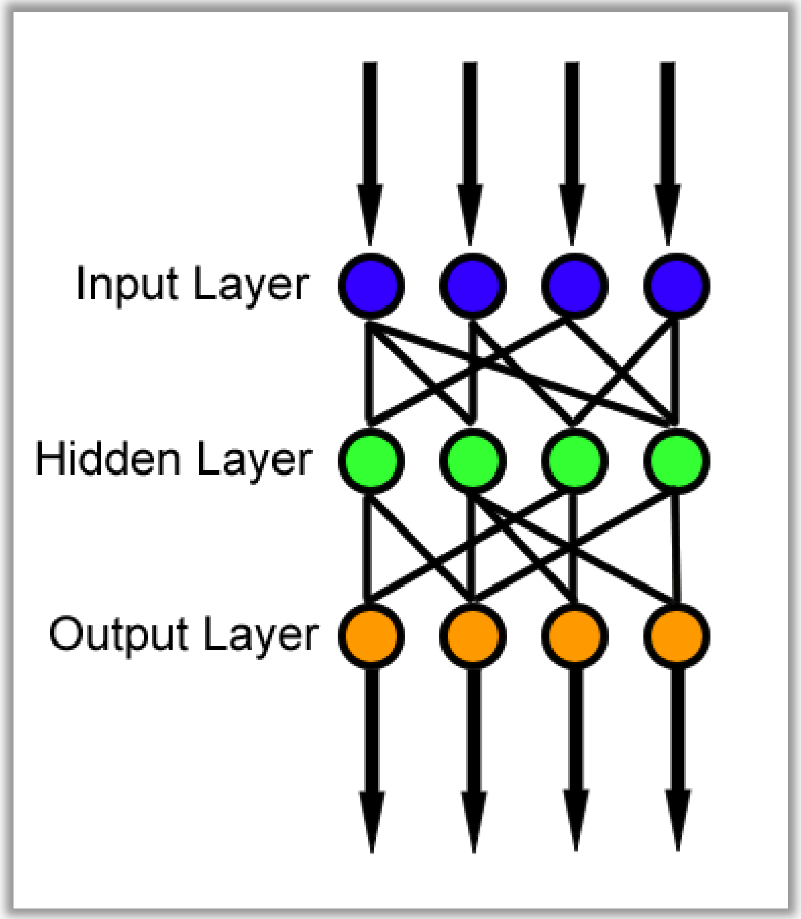


Types of Neural Networks - MLPs

- Neural networks are classified for different purposes

- **Perceptron**
  - Oldest neural network
  - Created by Frank Rosenblatt in 1958

- **Feedforward Neural Networks (MLPs)**
  - Multilayer perceptron
    - An input layer
    - Hidden layers
    - An output layer
  - Uses backpropagation for training
  - Fully connected
    - Each node in one layer connects a certain weight to every node in the following layer
  - **Use cases:**
    - Computer vision, natural language processing, etc



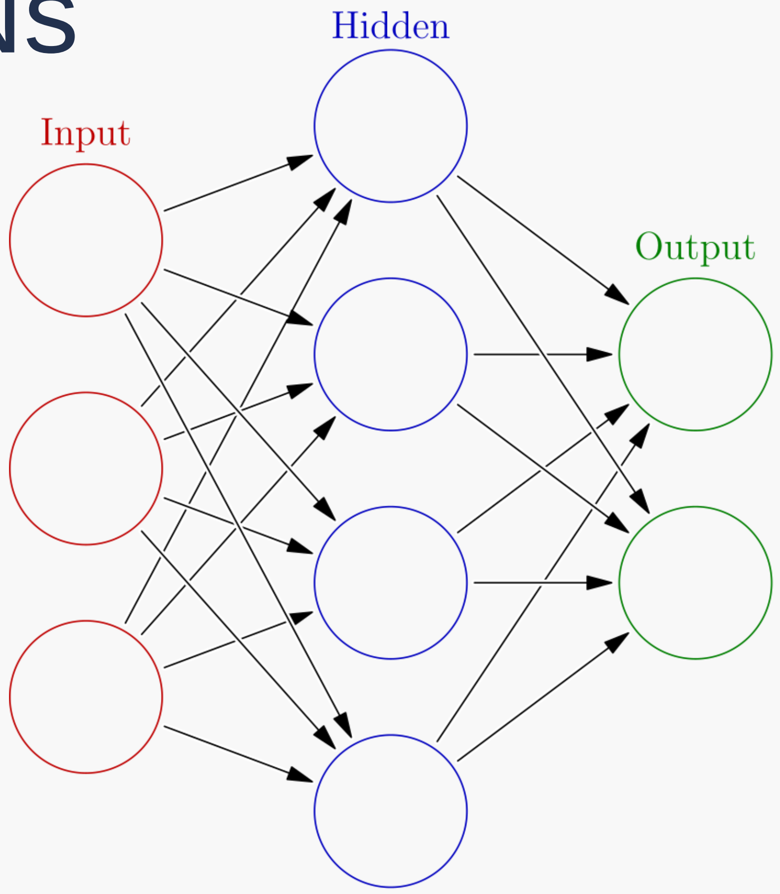
BATH SPA UNIVERSITY

Types of Neural Networks - ANNs

Artificial Neural Networks (ANN)

- Built using principles of neuronal organisation – imitates the biological neural network connection of a brain
- Comprised of connected units or nodes (artificial neurons)

- **Input Layer**
  - Buffers the input signal
- **Hidden Layer**
  - Internal to the network
  - No direct interaction with the system
- **Output Layer**
  - Generates the output of the network
- **Fully Connected Layers**
  - Each neuron in a layer is connected to all neurons in the next layer
- **Use cases:**
  - Data analysis, speech recognition, facial recognition, weather prediction



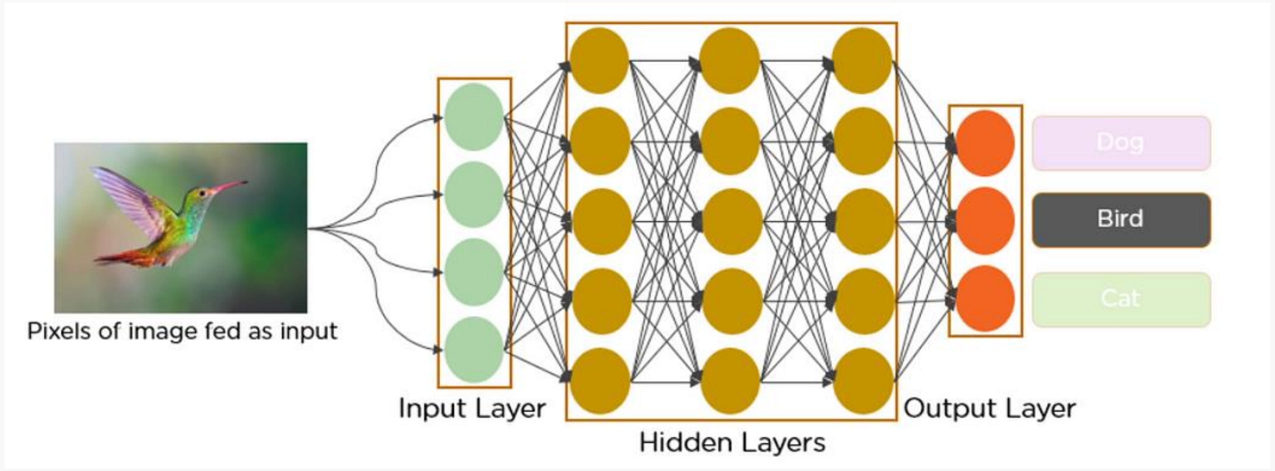
Artificial Neural Network (ANN)

Structure of Neural Networks - ANNs

- Artificial neural networks (ANNs)
  - Comprised of node layers
    - An input layer
    - One or more hidden layers
    - An output layer

- Each node, or artificial neuron:
  - Connects to another
  - Has an associated weight and threshold

- If the output of any individual node is above the specified threshold value
  - The node is activated
  - Sends data to the next layer of the network



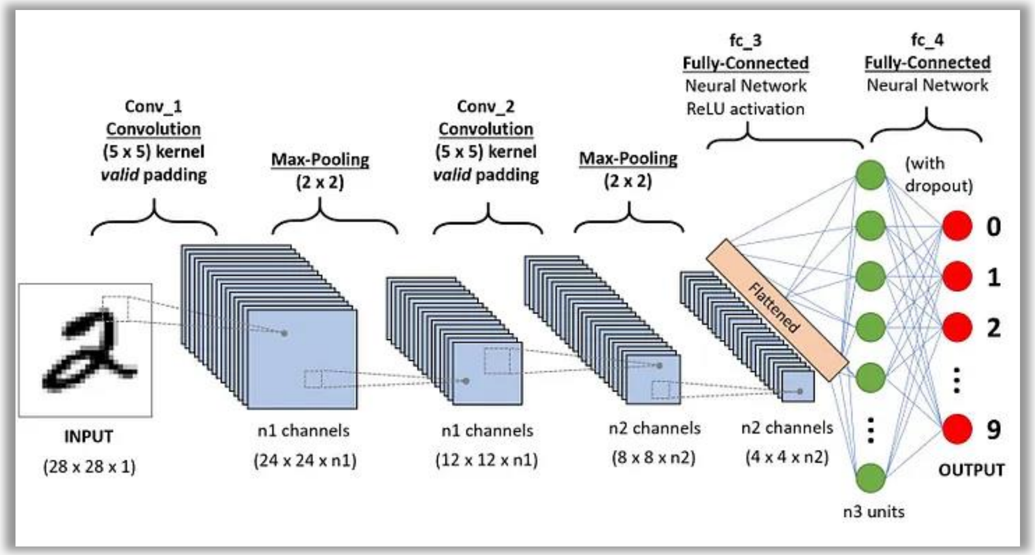
Types of Neural Networks - CNNs

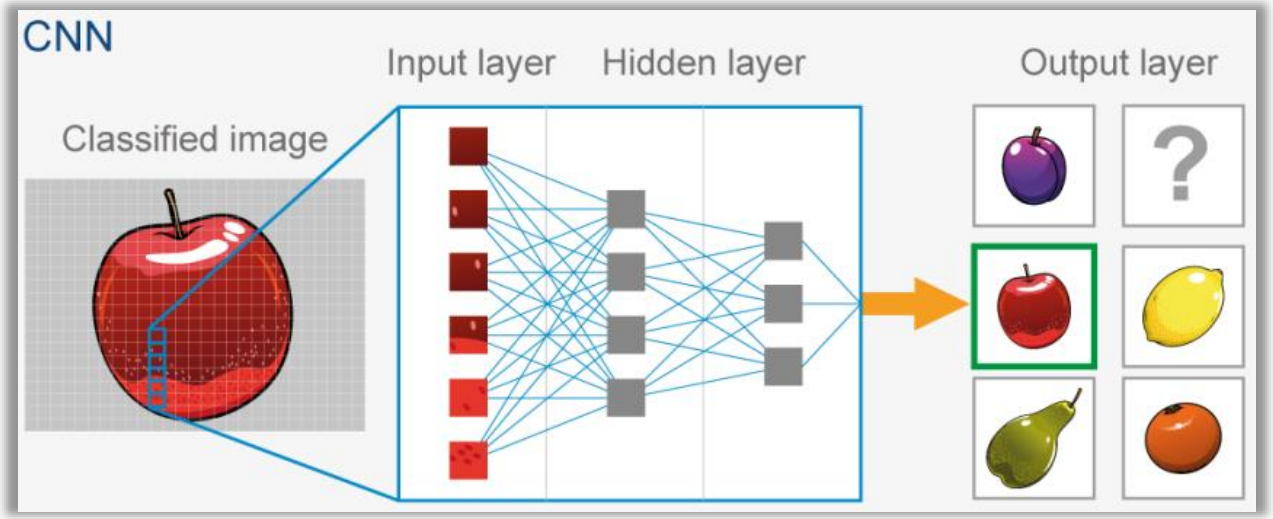
Convolutional Neural Networks (CNNs)

- Similar to feedforward networks

- **Convolutional Layer**
  - Extracts various features from the input images
- **Pooling Layer**
  - Reduces the spatial dimensions of the data
- **Fully Connected Layer**
  - Utilises the output from the convolution process
  - Predicts the class of the image
- **Dropout**
  - Regularisation technique to prevent overfitting
- **Activation Layer**
  - Applies an activation function to the output of the previous layer

- **Use Cases:**
  - Image recognition, pattern recognition, and computer vision

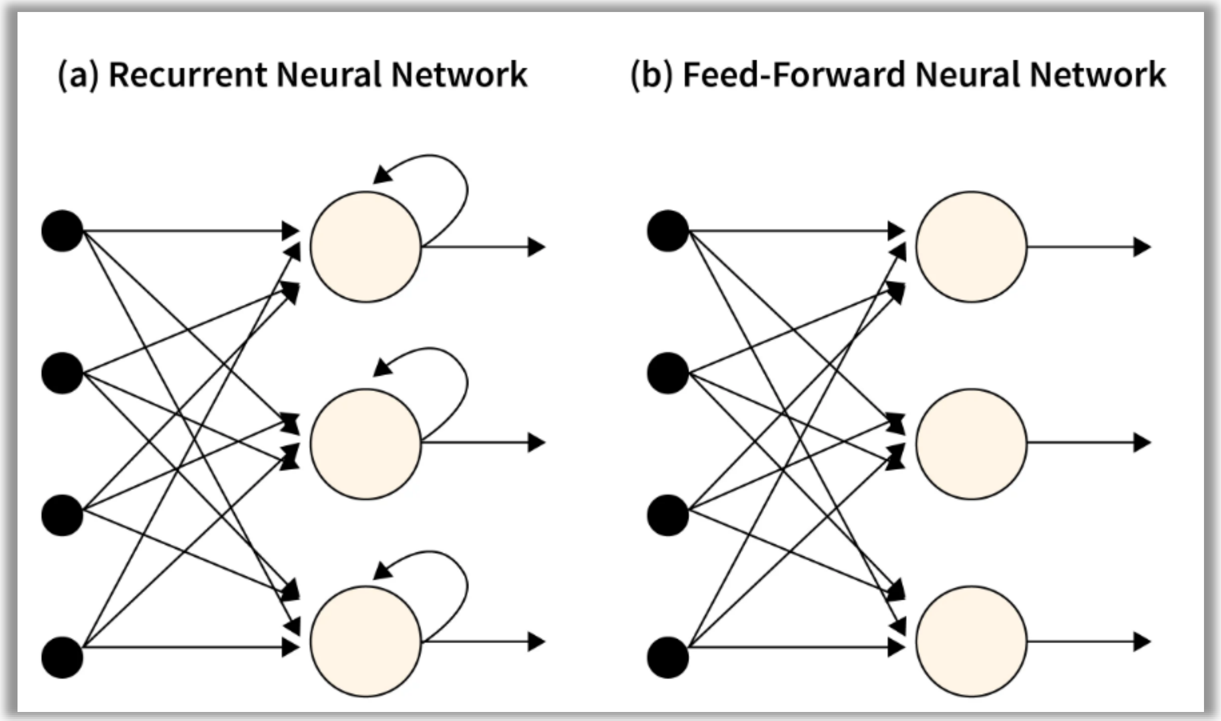




More information: Cnn - https://www.datacamp.com/tutorial/cnn-tensorflow-python

Types of Neural Networks – RNNs/GNNs

- **Recurrent Neural Networks (RNNs)**
  - Identified by their feedback loops
  - Time-series data predictions e.g. stock market predictions or sales forecasting
- **Graph Neural Networks (GNN)**
  - Class of deep learning methods designed for graph data
  - Can be directly applied to graphs
  - Utilised in prediction tasks
    - node-level, edge-level, and graph-level
- **Use Cases:**
  - Natural language processing(NLP), citation networks, chemistry, physics, chemistry



More information: GNN - https://www.datacamp.com/tutorial/comprehensive-introduction-graph-neural-networks-gnns-tutorial

Activation Functions

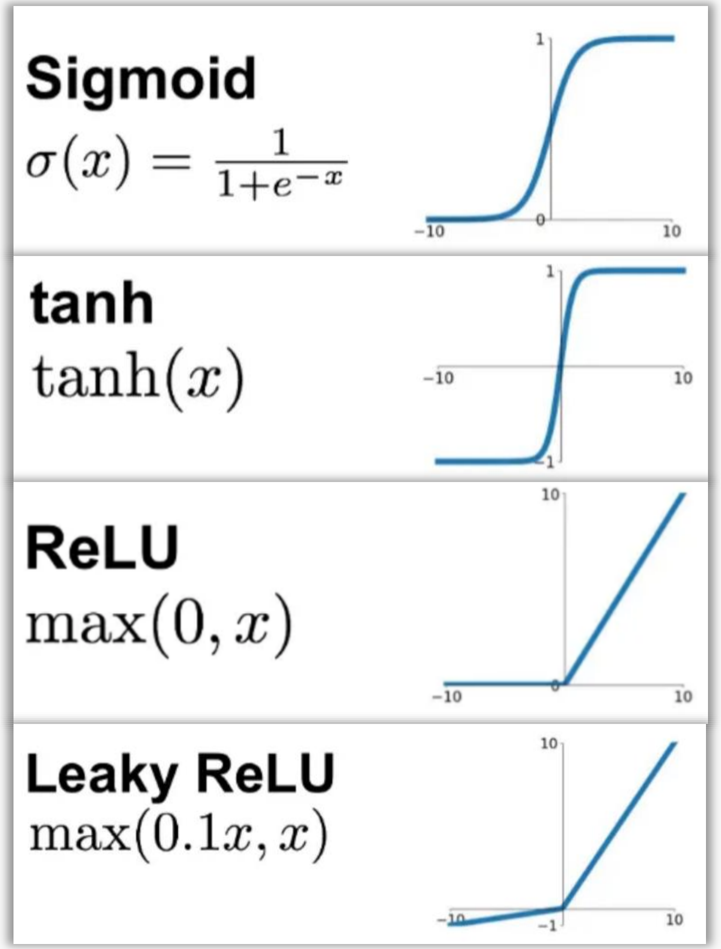
Activation Functions

- Mathematical functions attached to each neuron
- Determine whether a neuron should be activated

- **Neuron Operation**
  - Neurons perform a linear transformation on the input using weight and bias
  - Activation function is applied to the result
  - Output of current layer becomes input for next layer

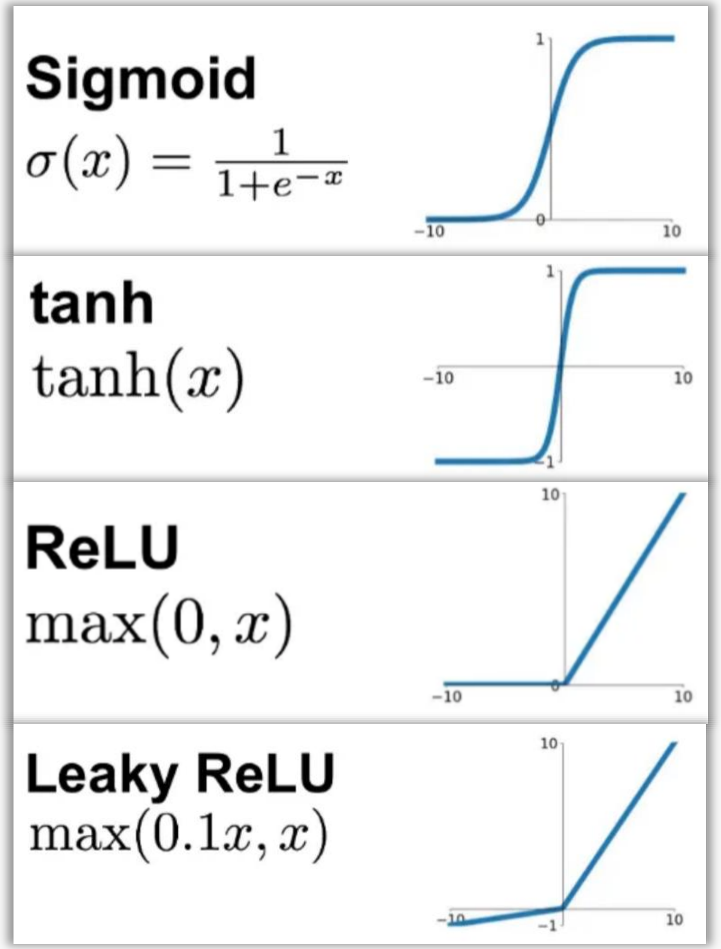
- **Forward Propagation**
  - Process of information movement forward in the network

- **Significance of Activation Functions**
  - Introduce non-linearity transformation in the neural network
  - Network can learn and perform complex tasks
  - Some activation functions can normalise the output of each neuron



Activation Functions

- Sigmoid function
  - Logistic activation function
- Tanh Function
- Rectified Linear Unit (ReLU) Function
- Leaky ReLU Function
- Softmax Function

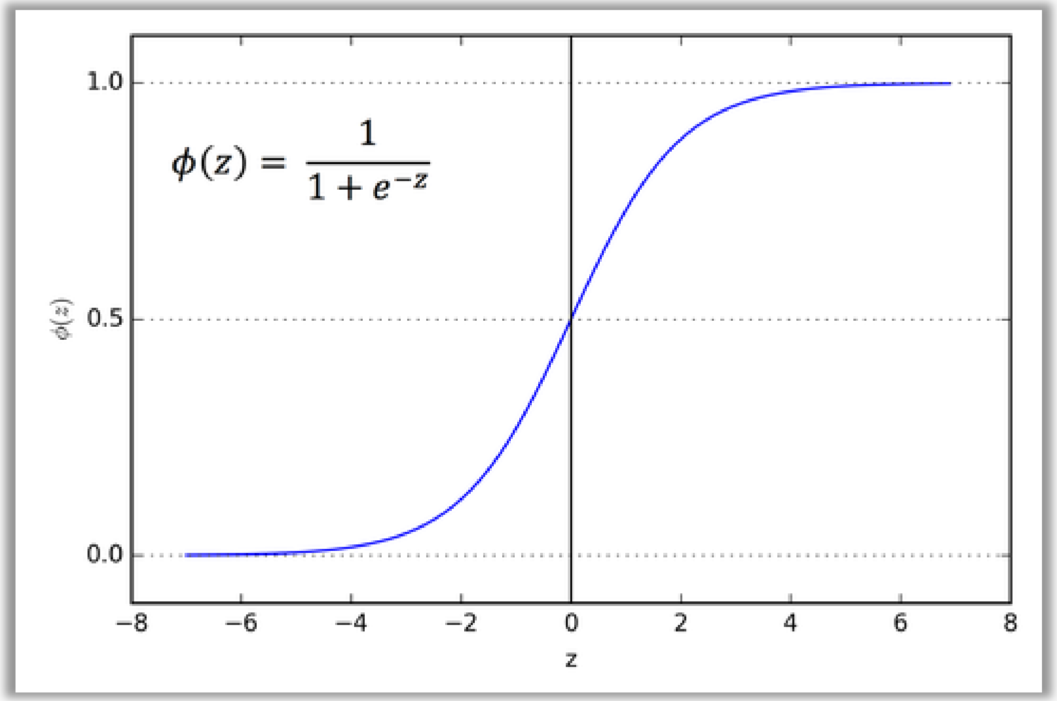


Sigmoid Function

- Curve has an S-shape
- Used because it exists between 0 and 1
- Especially used for models predicting probability

- **Properties of Sigmoid Function**
  - The function is differentiable
    - Take its derivative (or slope) at any point
    - Allows for gradient descent optimisations in NNs
  - Can cause a neural network to get stuck at training time

- **Softmax Function**
  - A more generalised logistic activation function
  - Used for multiclass classification

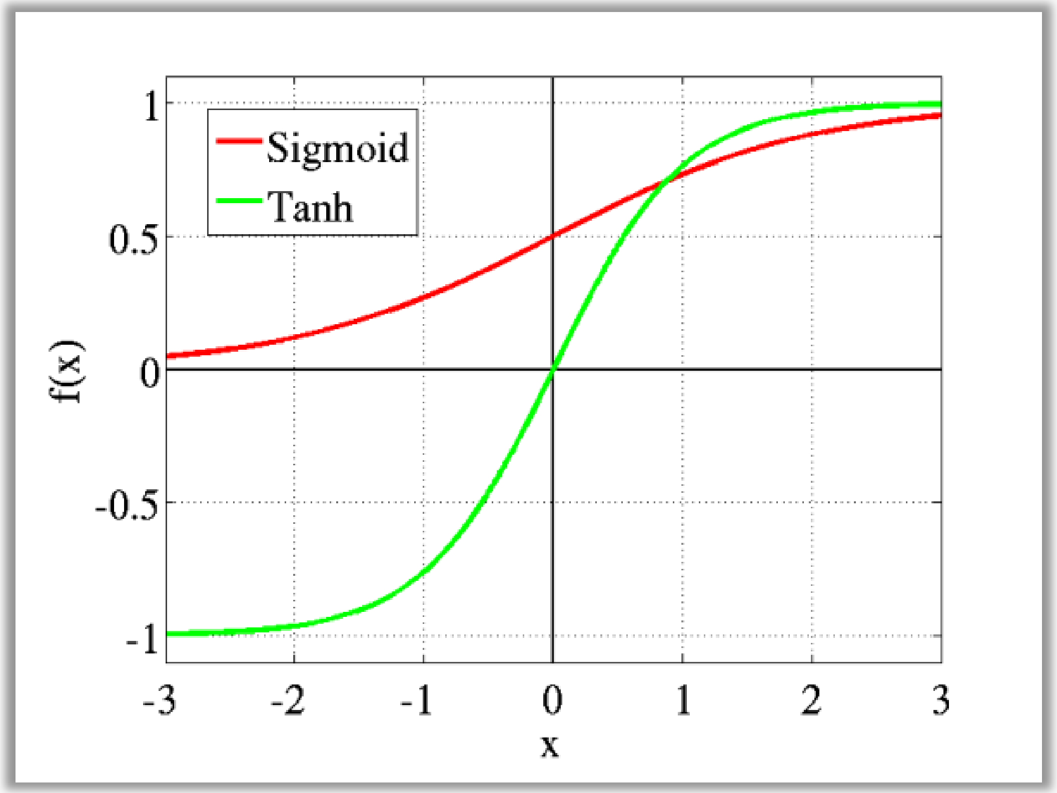


Tanh Function

- Like sigmoid function
  - Curve has an S-shape
  - But with range from -1 to 1
- Negative inputs map strongly negative
- Zero inputs map near zero

- **Properties of Tanh Function**
  - Differentiable
    - Allowing for gradient-based optimisation

- **Usage**
  - Used for classification between two classes
  - Used in feed-forward nets, along with sigmoid activation functions



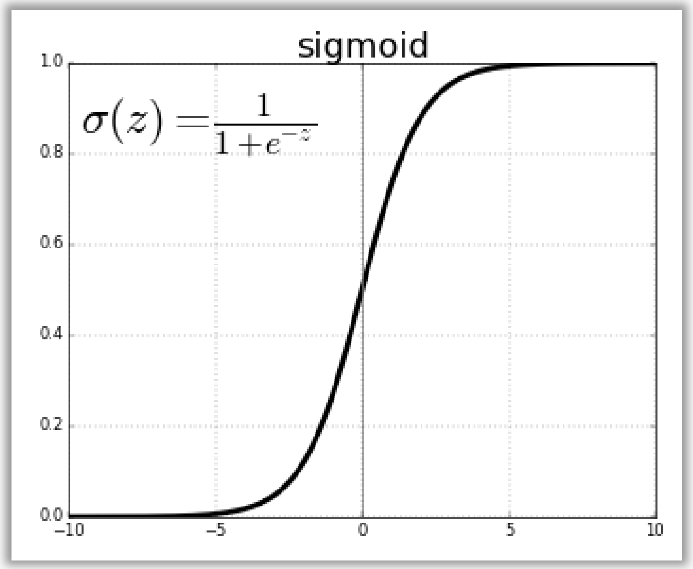
ReLU (Rectified Linear Unit) Activation Function

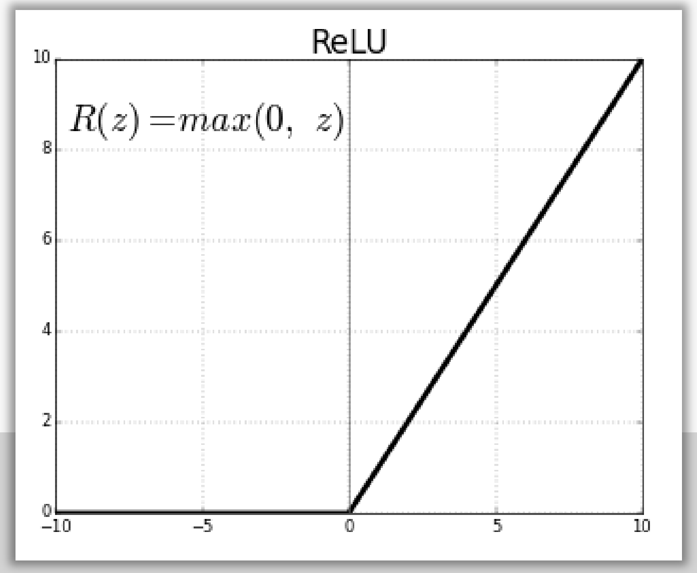
- Most used activation function in deep learning
- Used in almost all convolutional neural networks
- Range can be from 0 to infinity

- **ReLU Properties**
  - Half rectified from bottom
  - **z**
    - Zero
      - When z is less than zero
    - Is equal to z
      - When z is above or equal to zero

- **Characteristics of ReLU**
  - Relu dying problem
    - Negative values become zero decreasing the model's ability to fit or train data properly
    - Neuron fails to learn

- **Usage**
  - Image classification, object recognition, and segmentation

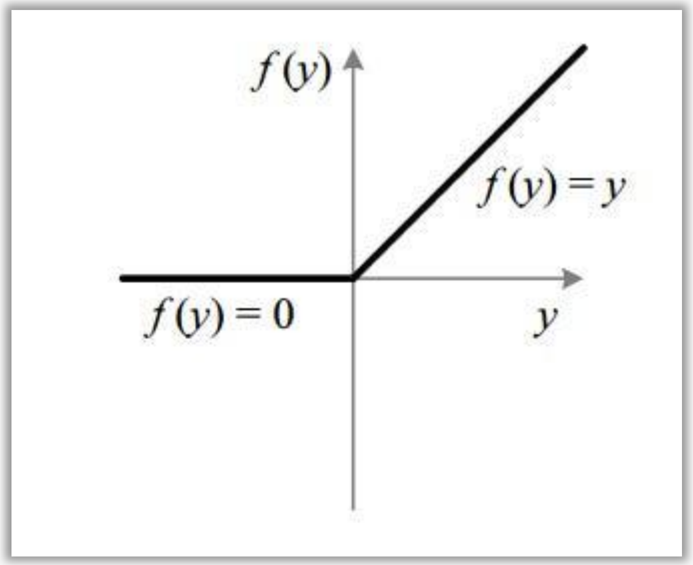


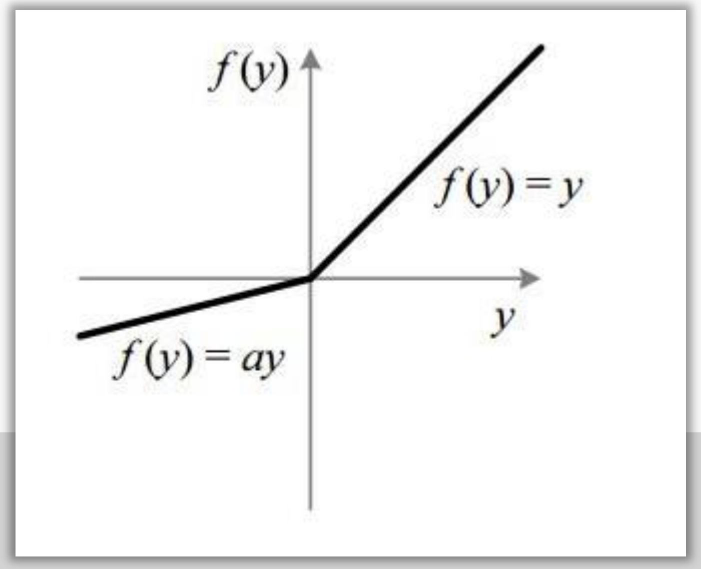


Leaky ReLU

- Attempt to solve the dying ReLU problem
- Leak helps to increase the range of the ReLU function
  - Value of **a** is usually 0.01
- Range can be from negative infinity to infinity

- **Randomised ReLU**
  - When **a** is not 0.01





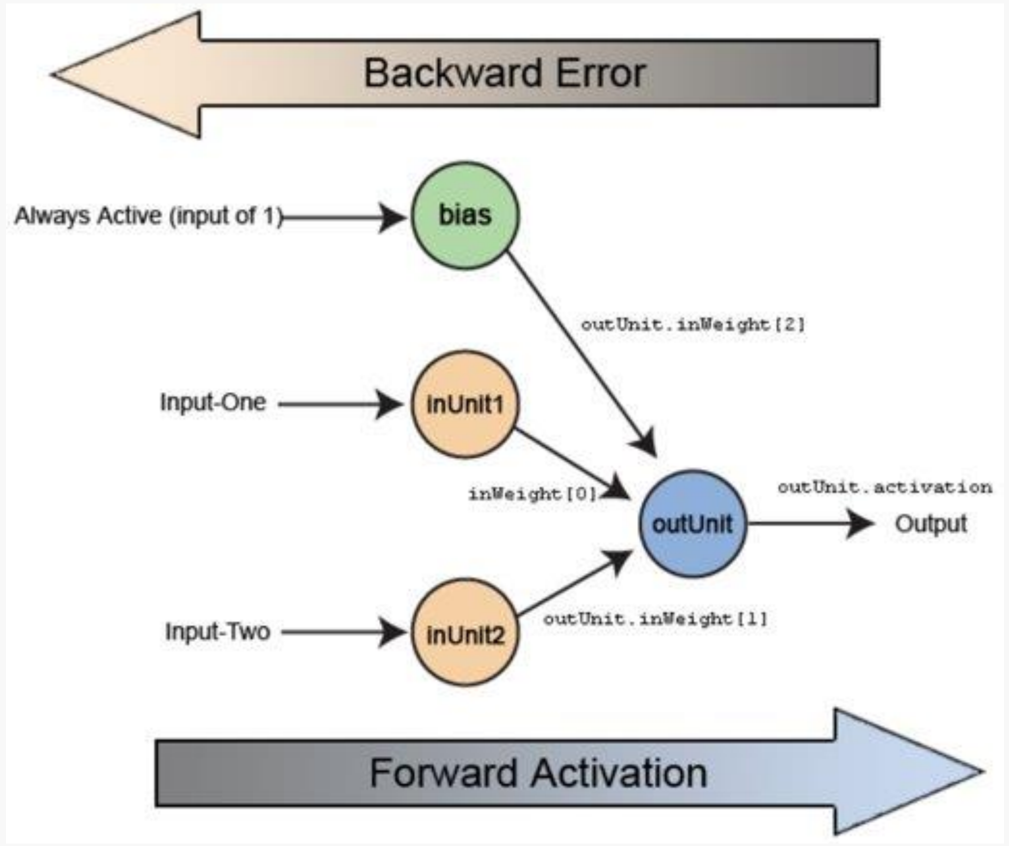
Loss Functions

- Measure neural network model performance
- Calculate difference between predicted and actual output
- Crucial for optimizing model performance
- Choice of loss function matches predictive modelling problem
- Different types used for specific tasks

- **Forward Propagation**
  - Predictions based on input data

- **Backpropagation**
  - Adjusts weights and biases to minimise loss function value
  - Uses optimisation algorithms e.g., gradient descent

- **Minimisation**
  - Goal is to minimize loss function value
  - Lower loss value indicates better model performance



Loss Functions

- Mean Square Error (MSE)
- Mean Absolute Error (MAE)
- Mean Absolute Percentage Error (MAPE)
- Binary Cross-Entropy Loss
- Hinge Loss
  - Multiclass SVM loss
- Categorical Cross entropy
- Sparse Categorical Cross entropy
- Kullback Leibler Divergence
- Poisson
- Cosine Proximity

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$$

$$MAE = \frac{\sum_{i=1}^{n} | y_i - \hat{y}_i |}{n}$$

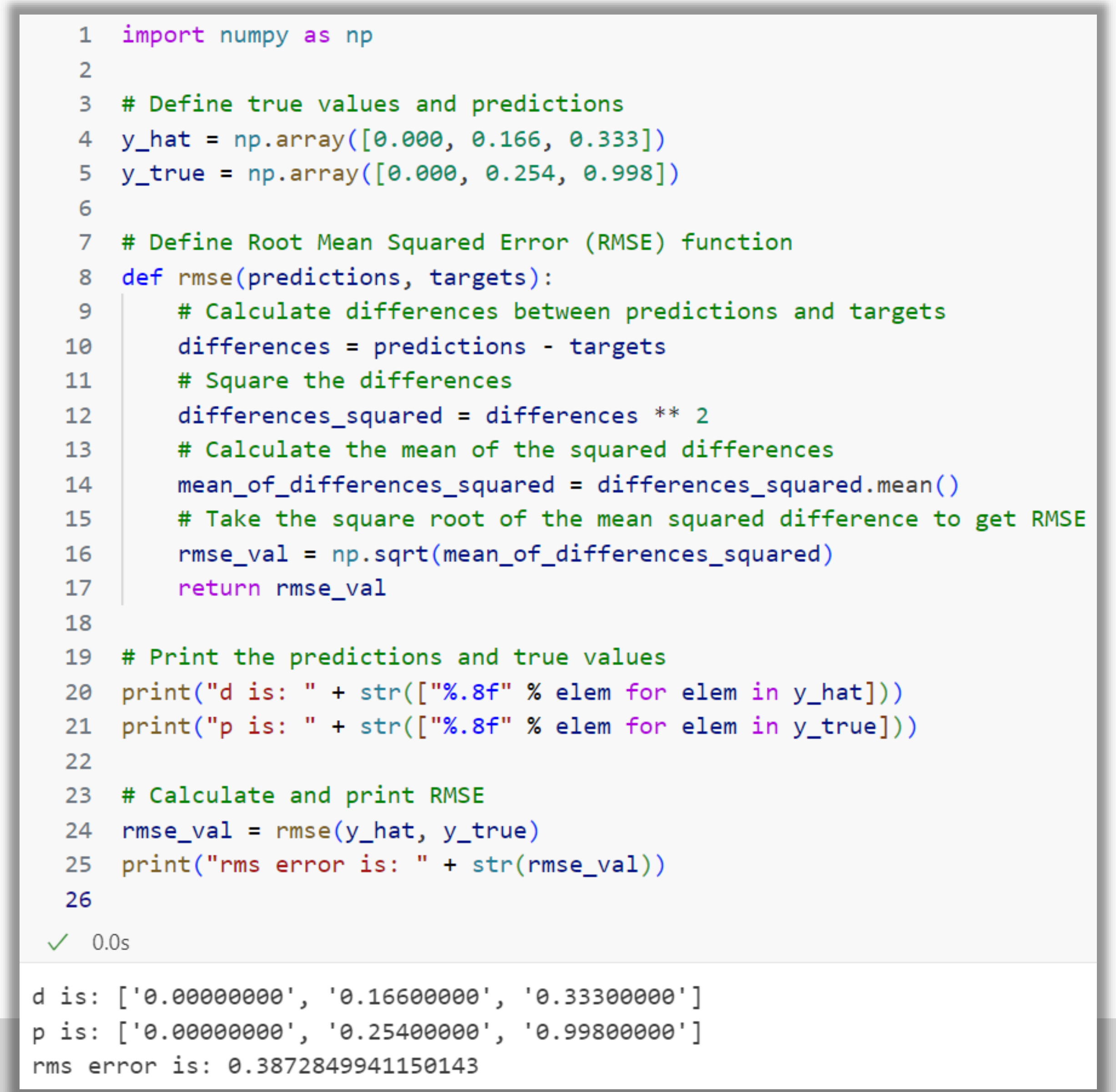
$$SVMLoss = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1)$$

$$CrossEntropyLoss = -(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i))$$

• n	- Number of training examples
• i	- ith training example
• y(i)	- Ground truth label for ith training example
• y_hat(i)	- Prediction for ith training example

MSE

- Average of squared difference
  - Between predictions
    - And actual observations
- Heavily penalises predictions far from actual values
  - Due to squaring
- Simplify gradient calculation
  - Using mathematical properties



```python
import numpy as np

# Define true values and predictions
y_hat = np.array([0.000, 0.166, 0.333])
y_true = np.array([0.000, 0.254, 0.998])

# Define Root Mean Squared Error (RMSE) function
def rmse(predictions, targets):
    # Calculate differences between predictions and targets
    differences = predictions - targets
    # Square the differences
    differences_squared = differences ** 2
    # Calculate the mean of the squared differences
    mean_of_differences_squared = differences_squared.mean()
    # Take the square root of the mean squared difference to get RMSE
    rmse_val = np.sqrt(mean_of_differences_squared)
    return rmse_val

# Print the predictions and true values
print("d is: " + str(["%.8f" % elem for elem in y_hat]))
print("p is: " + str(["%.8f" % elem for elem in y_true]))

# Calculate and print RMSE
rmse_val = rmse(y_hat, y_true)
print("rms error is: " + str(rmse_val))

```

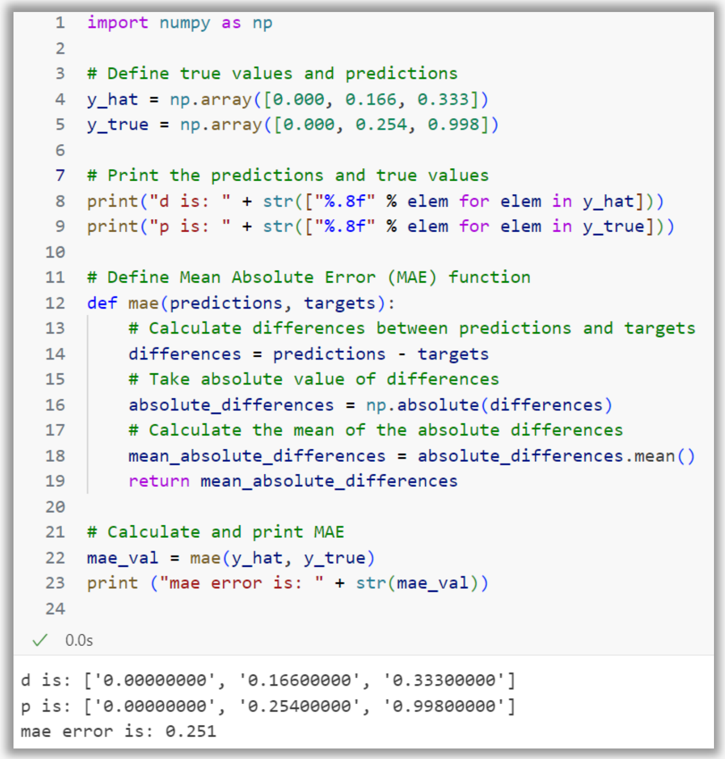
✓ 0.0s

```
d is: ['0.00000000', '0.16600000', '0.33300000']
p is: ['0.00000000', '0.25400000', '0.99800000']
rms error is: 0.3872849941150143
```

BATH SPA UNIVERSITY

MAE

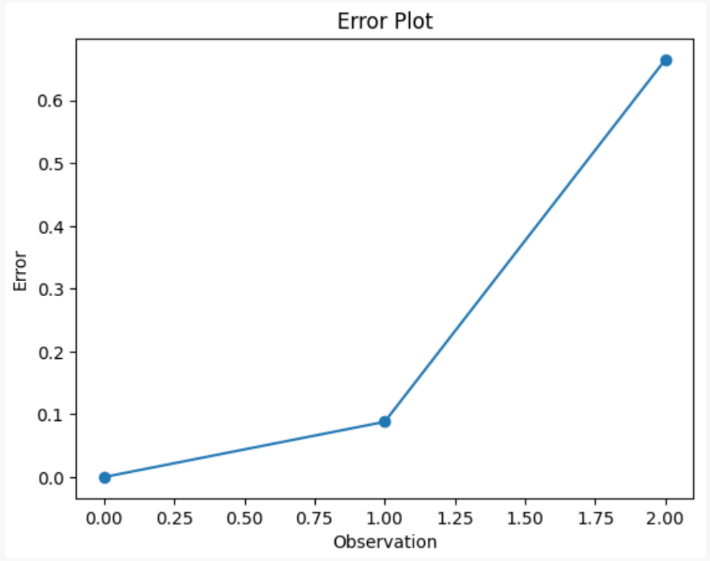
- Average of sum of absolute differences between predictions and actual observations
- Measures the magnitude of error without considering their direction
- More complicated to compute gradients compared with MSE
- More robust to outliers
  - Does not use squaring



```python
import numpy as np

# Define true values and predictions
y_hat = np.array([0.000, 0.166, 0.333])
y_true = np.array([0.000, 0.254, 0.998])

# Print the predictions and true values
print("d is: " + str(["%.8f" % elem for elem in y_hat]))
print("p is: " + str(["%.8f" % elem for elem in y_true]))

# Define Mean Absolute Error (MAE) function
def mae(predictions, targets):
    # Calculate differences between predictions and targets
    differences = predictions - targets
    # Take absolute value of differences
    absolute_differences = np.absolute(differences)
    # Calculate the mean of the absolute differences
    mean_absolute_differences = absolute_differences.mean()
    return mean_absolute_differences

# Calculate and print MAE
mae_val = mae(y_hat, y_true)
print ("mae error is: " + str(mae_val))

```

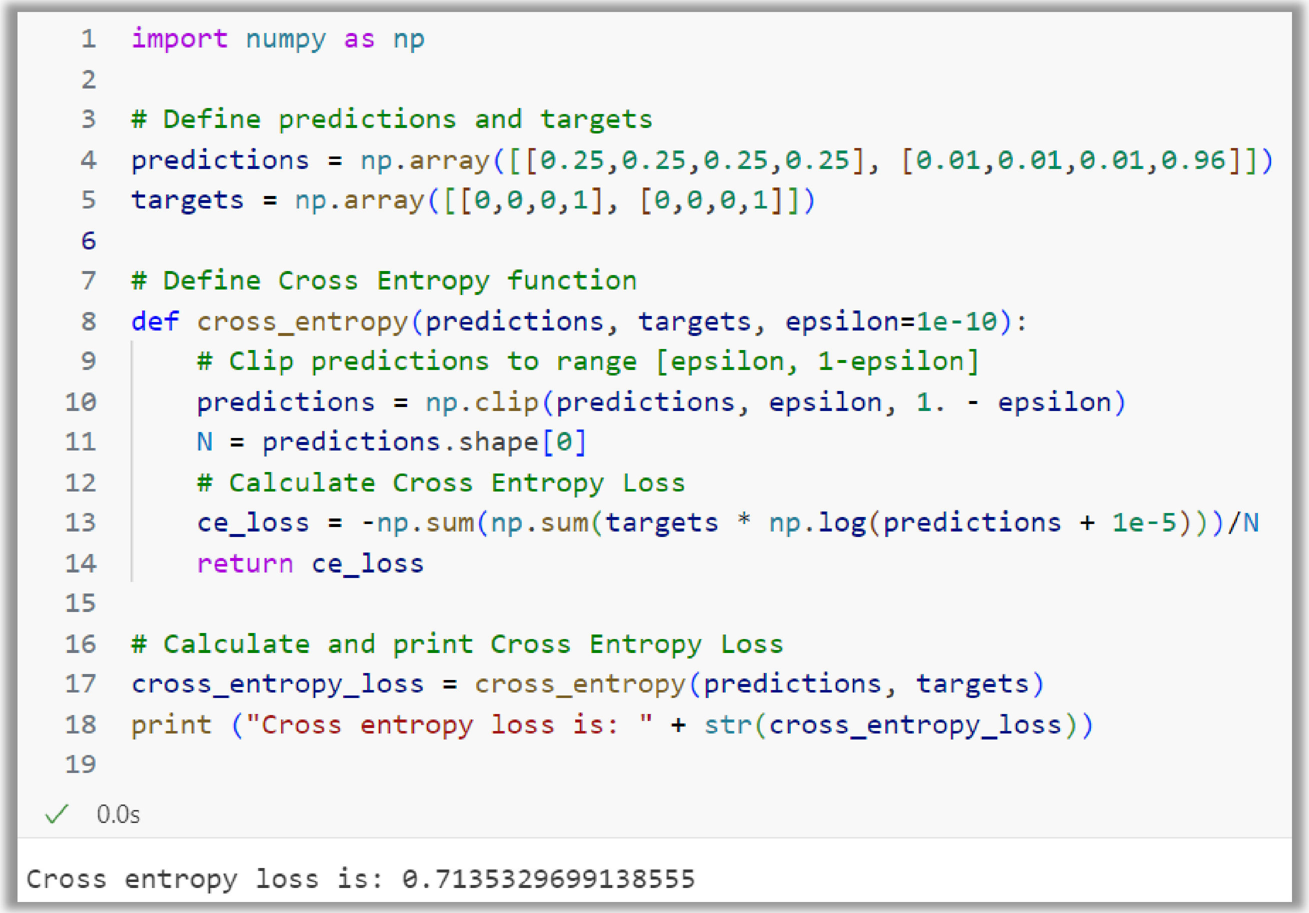
✓ 0.0s

```
d is: ['0.00000000', '0.16600000', '0.33300000']
p is: ['0.00000000', '0.25400000', '0.99800000']
mae error is: 0.251
```



Cross Entropy

- Measures the performance of a classification model whose output is a probability value between 0 and 1
- Loss increases as the predicted probability diverges from the actual label
- Heavily penalises confident but wrong predictions
- Commonly used in classification problems



```python
import numpy as np

# Define predictions and targets
predictions = np.array([[0.25,0.25,0.25,0.25], [0.01,0.01,0.01,0.96]])
targets = np.array([[0,0,0,1], [0,0,0,1]])

# Define Cross Entropy function
def cross_entropy(predictions, targets, epsilon=1e-10):
    # Clip predictions to range [epsilon, 1-epsilon]
    predictions = np.clip(predictions, epsilon, 1. - epsilon)
    N = predictions.shape[0]
    # Calculate Cross Entropy Loss
    ce_loss = -np.sum(np.sum(targets * np.log(predictions + 1e-5)))/N
    return ce_loss

# Calculate and print Cross Entropy Loss
cross_entropy_loss = cross_entropy(predictions, targets)
print ("Cross entropy loss is: " + str(cross_entropy_loss))

```

✓ 0.0s

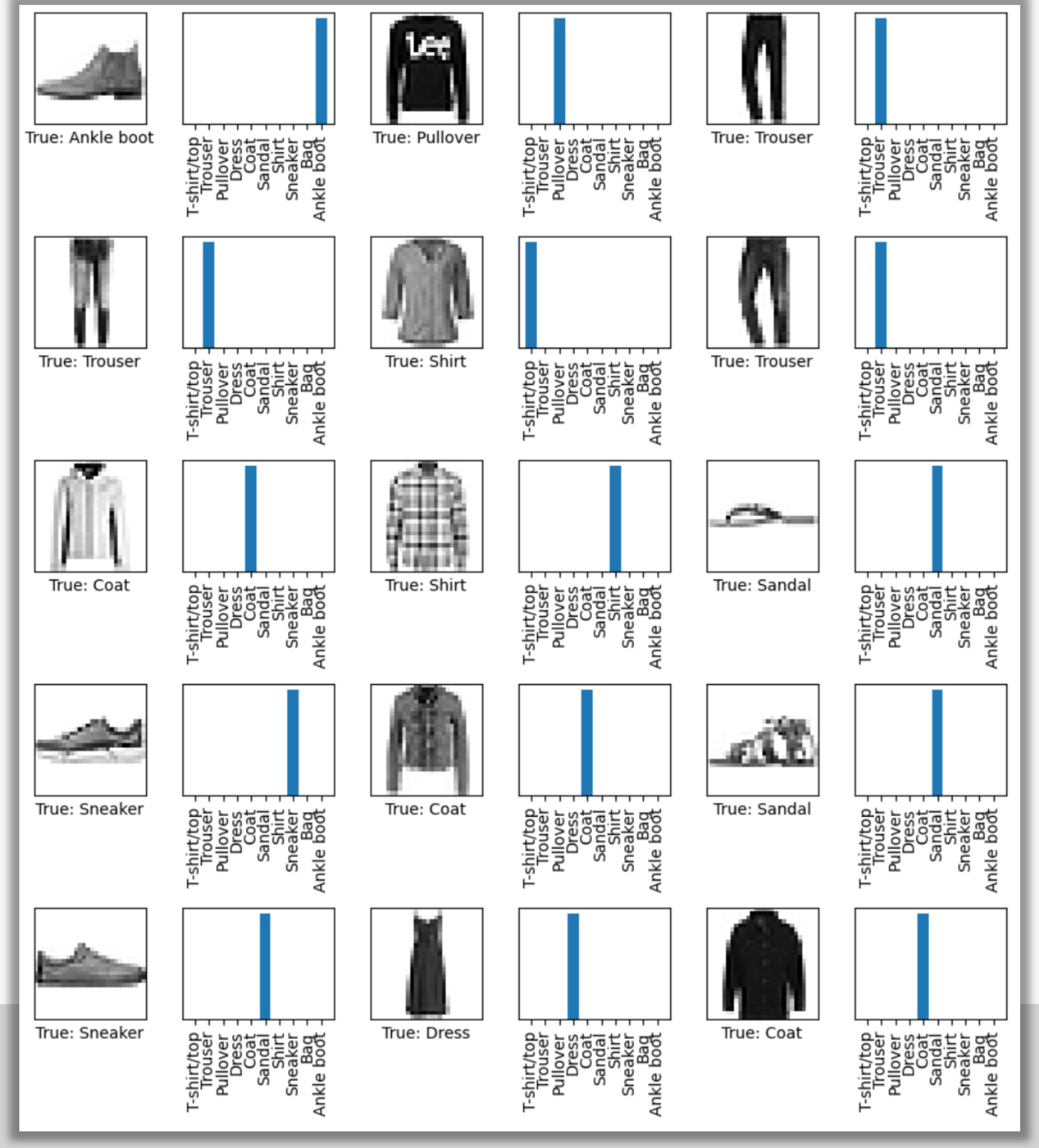
```
Cross entropy loss is: 0.7135329699138555
```

Training a Neural Network

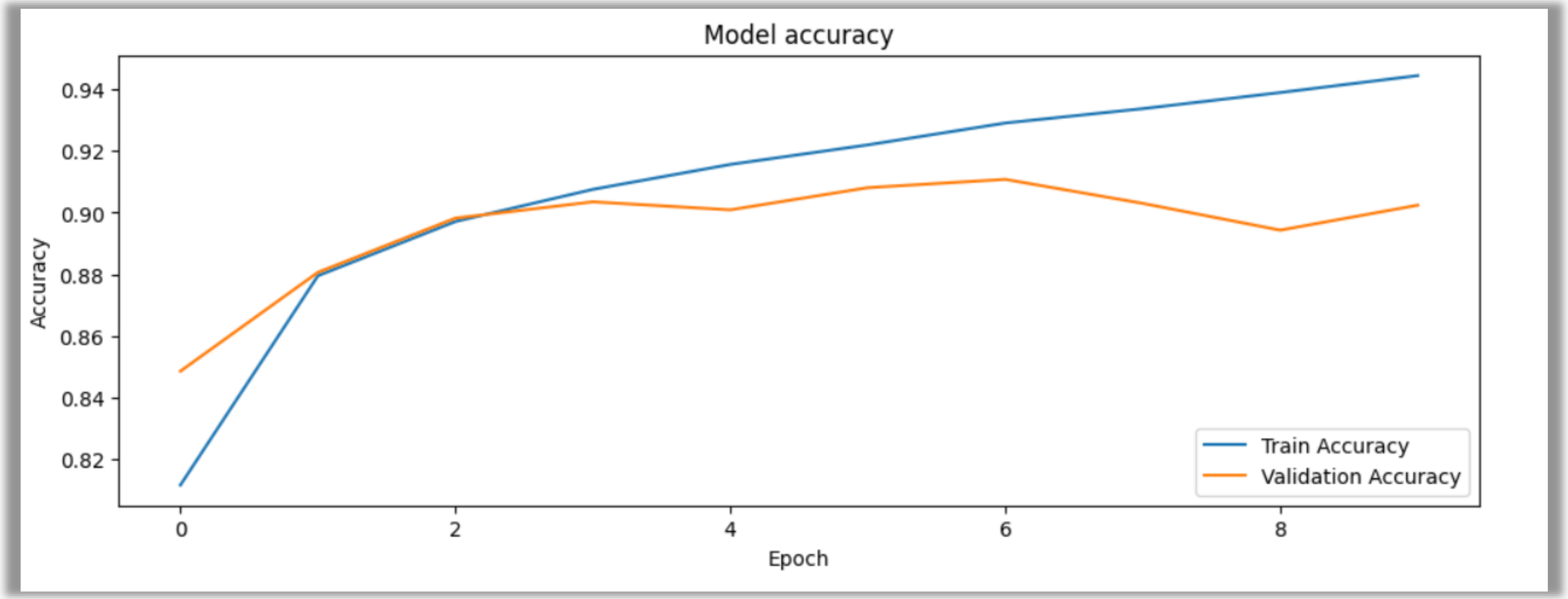
Tensorflow

Training a Neural Network

1. **Install TensorFlow**
2. **Look at fashion dataset**
3. **Data Preprocessing**
4. **Create first Neural Network**
   - TensorFlow
5. **Predict**
   - Type of clothing is shown in images
   - New images will be used
   - Testing set
   - Neural Network has never seen



Training a Neural Network – CNN Model Accuracy



Session Review

- Able to discuss the concept of neural networks
- Able to understand the structure of a neural network.
- Bonus topic – using APIs in Python and Power BI