Python for Beginners

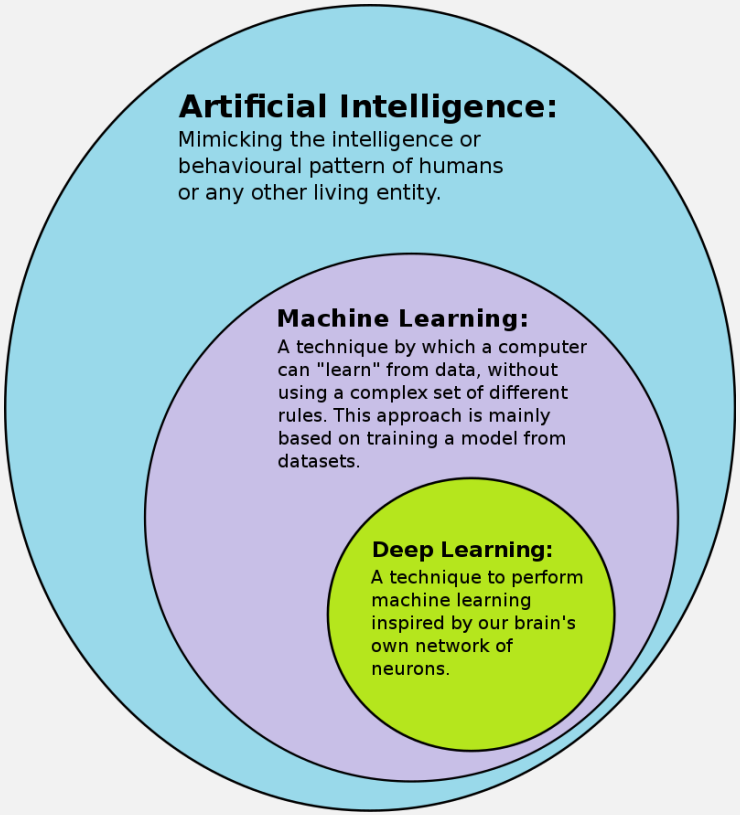
Archer Infotech , PUNE

Python – Deep Learning

What is Deep Learning ?

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers.

These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to “learn” from large amounts of data.



Artificial Neural Network

- **Artificial neural network is usually a computational network based on biological neural networks that construct the structure of the human brain.**

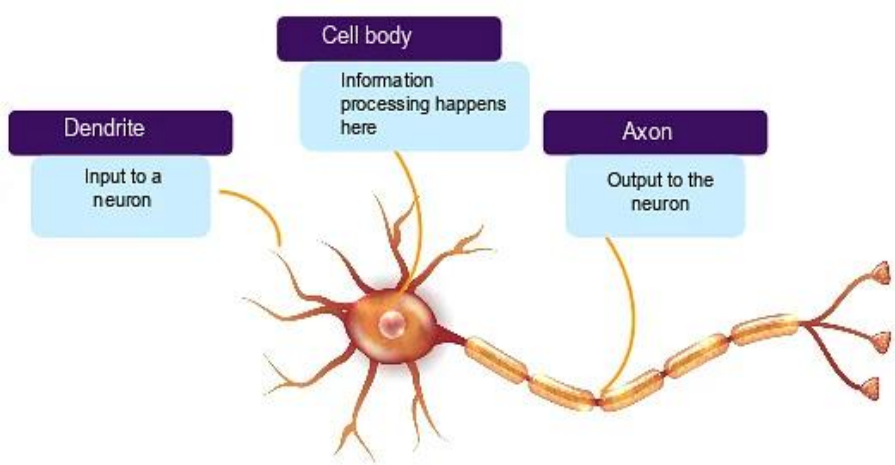
- **Similar to a human brain has neurons interconnected to each other, artificial neural networks also have neurons that are linked to each other in various layers of the networks. These neurons are known as nodes.**

Use cases:

- Pattern Recognition
- Time Series Predictions
- Signal Processing
- Anomaly Detection
- Control

Artificial Neural Network

- **The human brain has interconnected neurons with dendrites that receive inputs, ar based on those inputs, produce an electrical signal output through the axon**

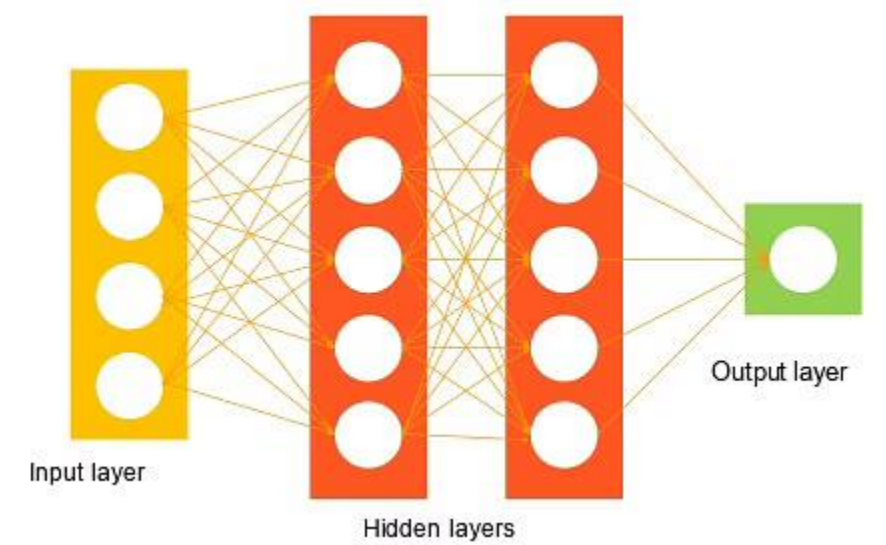


Dendrites
These receive information or signals from other neurons that get connected to it.

Cell Body
Information processing happens in a cell body. These take in all the information coming from the different dendrites and process that information.

Axon
It sends the output signal to another neuron for the flow of information. Here, each of the flanges connects to the dendrite or the hairs on the next one.

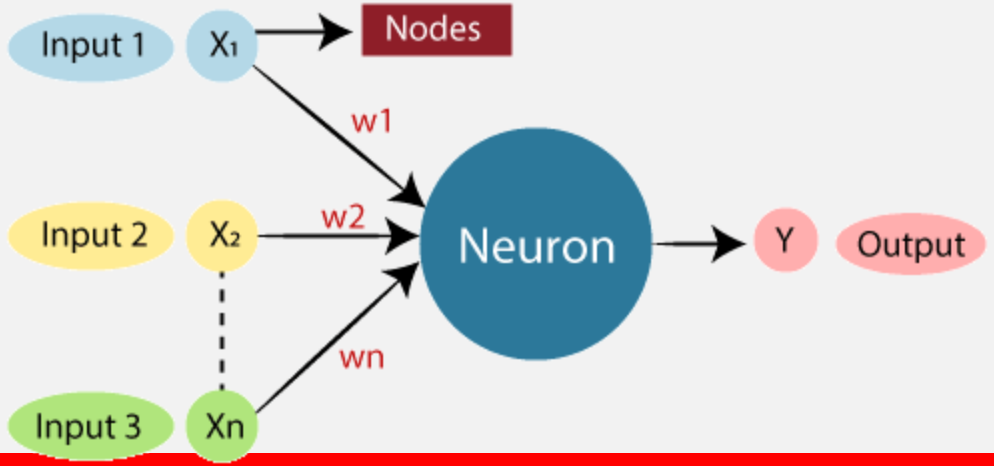
Artificial Neural Network



Input to a neuron - input layer

Neuron - hidden layer

Output to the next neuron - output layer



Artificial Neural Network

- **Artificial Neural Networks(ANN) are part of supervised machine learning where we will be having input as well as corresponding output present in our dataset.**

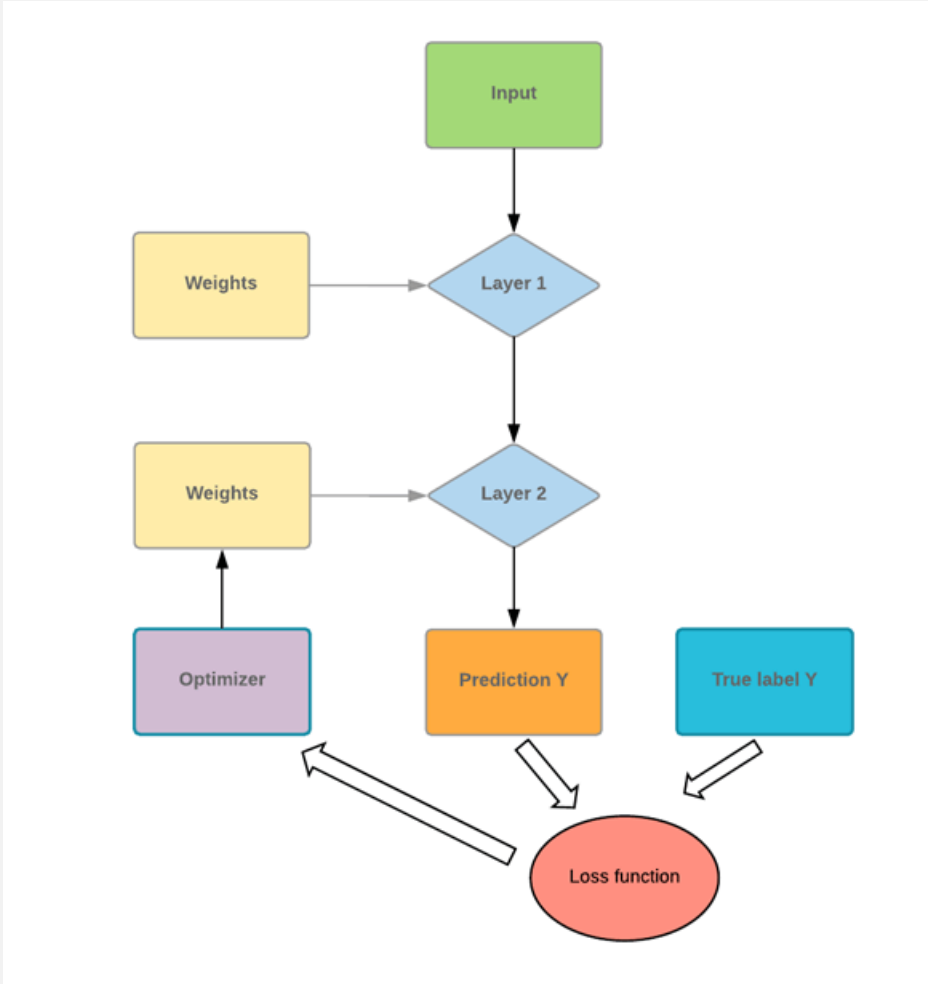
- **Our whole aim is to figure out a way of mapping this input to the respective output.**

- **ANN can be used for solving both regression and classification problems**

Artificial Neural Network

An **Artificial Neural Network** (ANN) is a computer system inspired by biological neural networks for creating artificial brains based on the collection of connected units called artificial neurons.

It is designed to analyze and process information as humans.
Artificial Neural Network has self-learning capabilities to produce better results as more data is available.



Artificial Neural Network

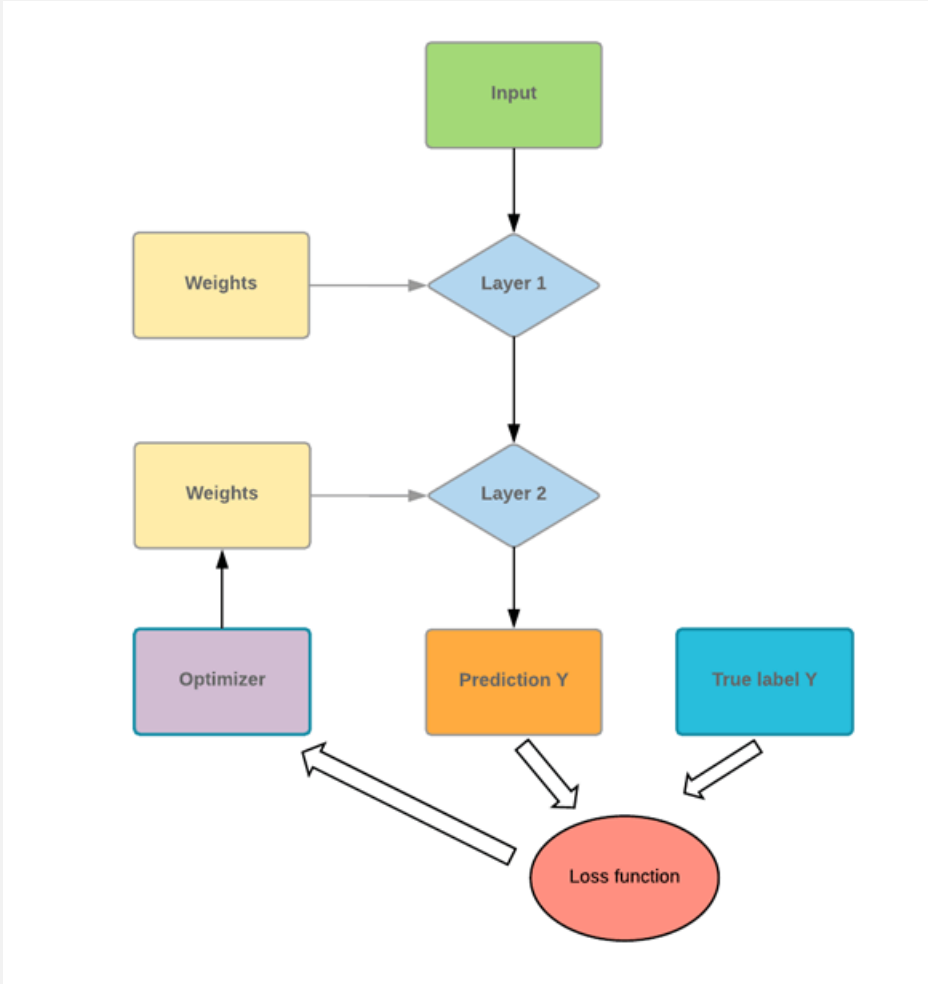
An Artificial Neural Network (ANN) is composed of four principal objects:

Layers:
all the learning occurs in the layers.
There are 3 layers
1) Input 2) Hidden and 3) Output

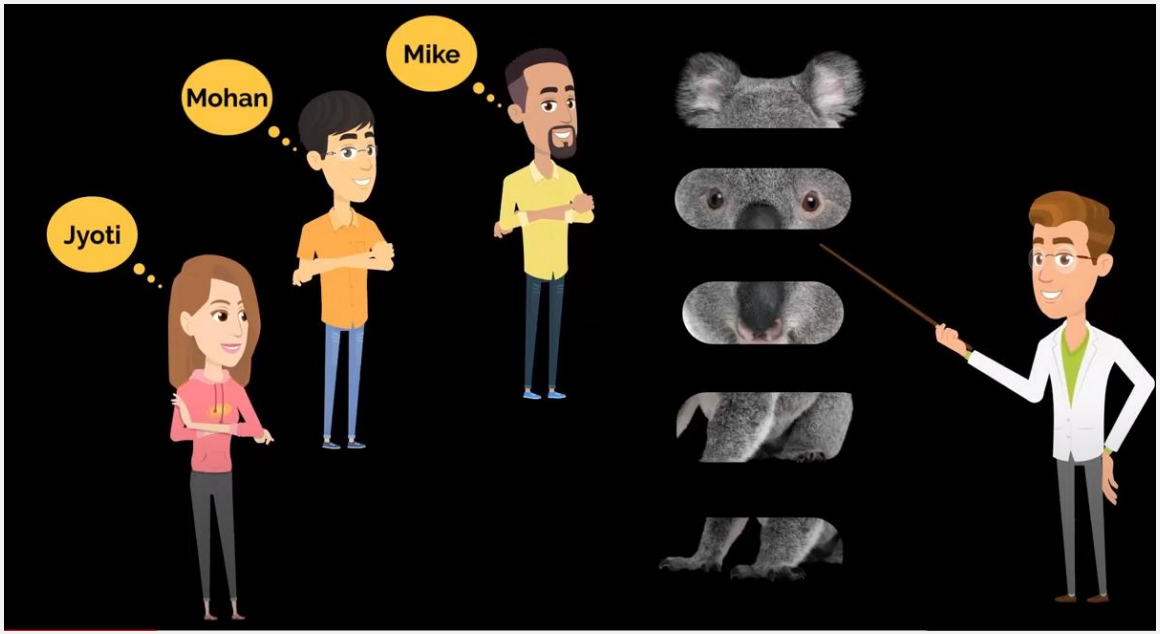
Feature and label:
Input data to the network (features)
and output from the network (labels)

Loss function:
Metric used to estimate the performance of the learning phase

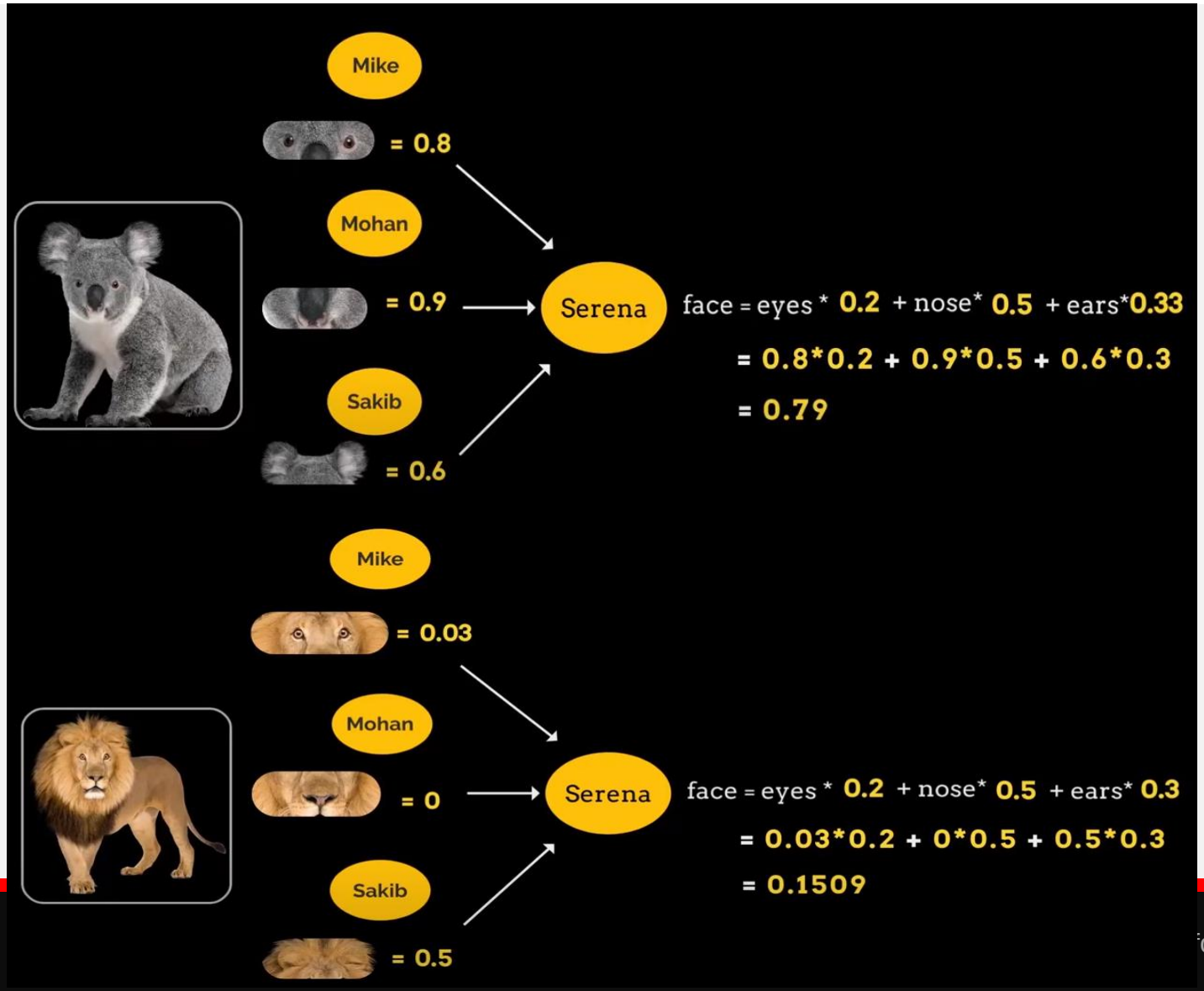
Optimizer: Improve the learning by updating the knowledge in the network



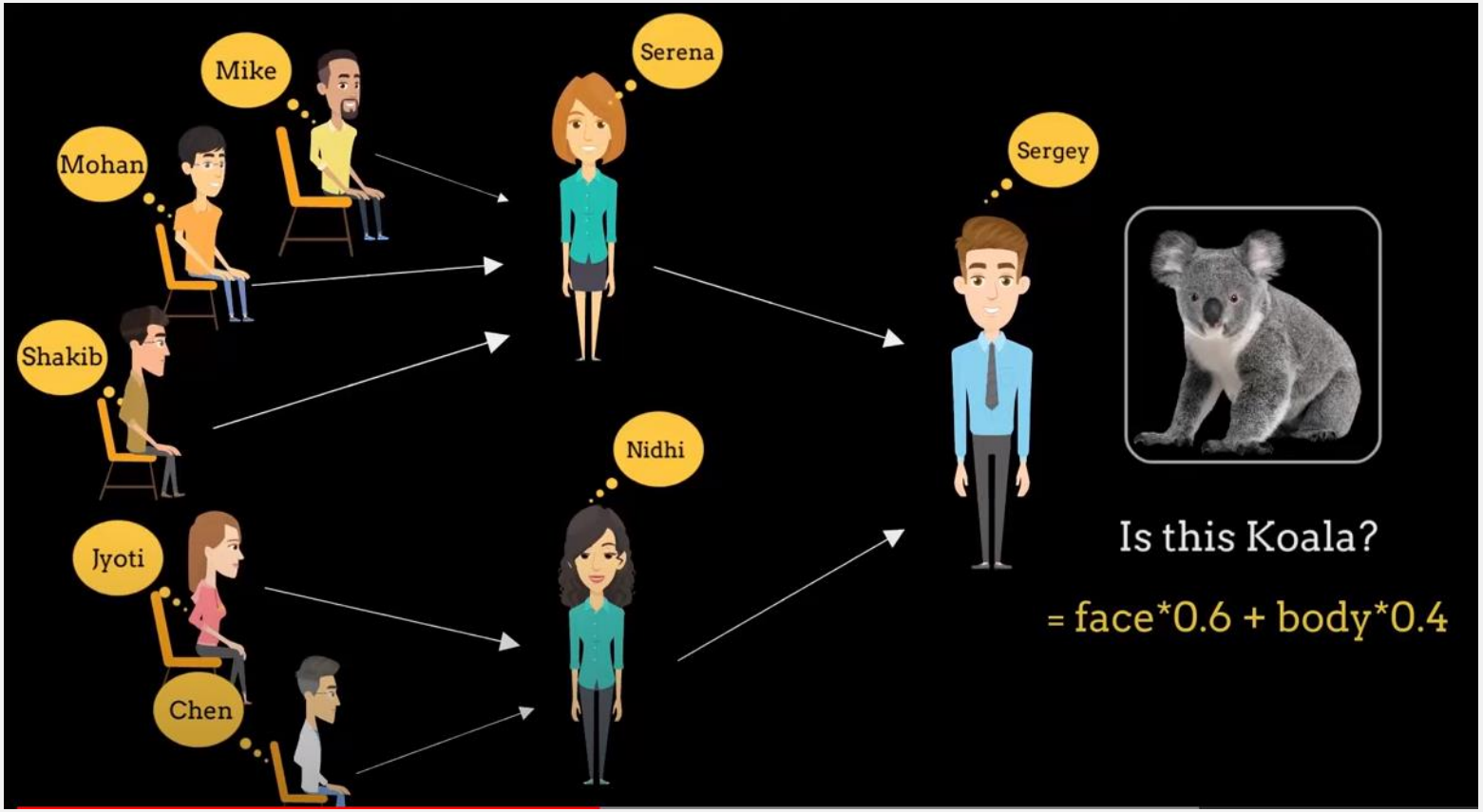
ANN in Simple Way



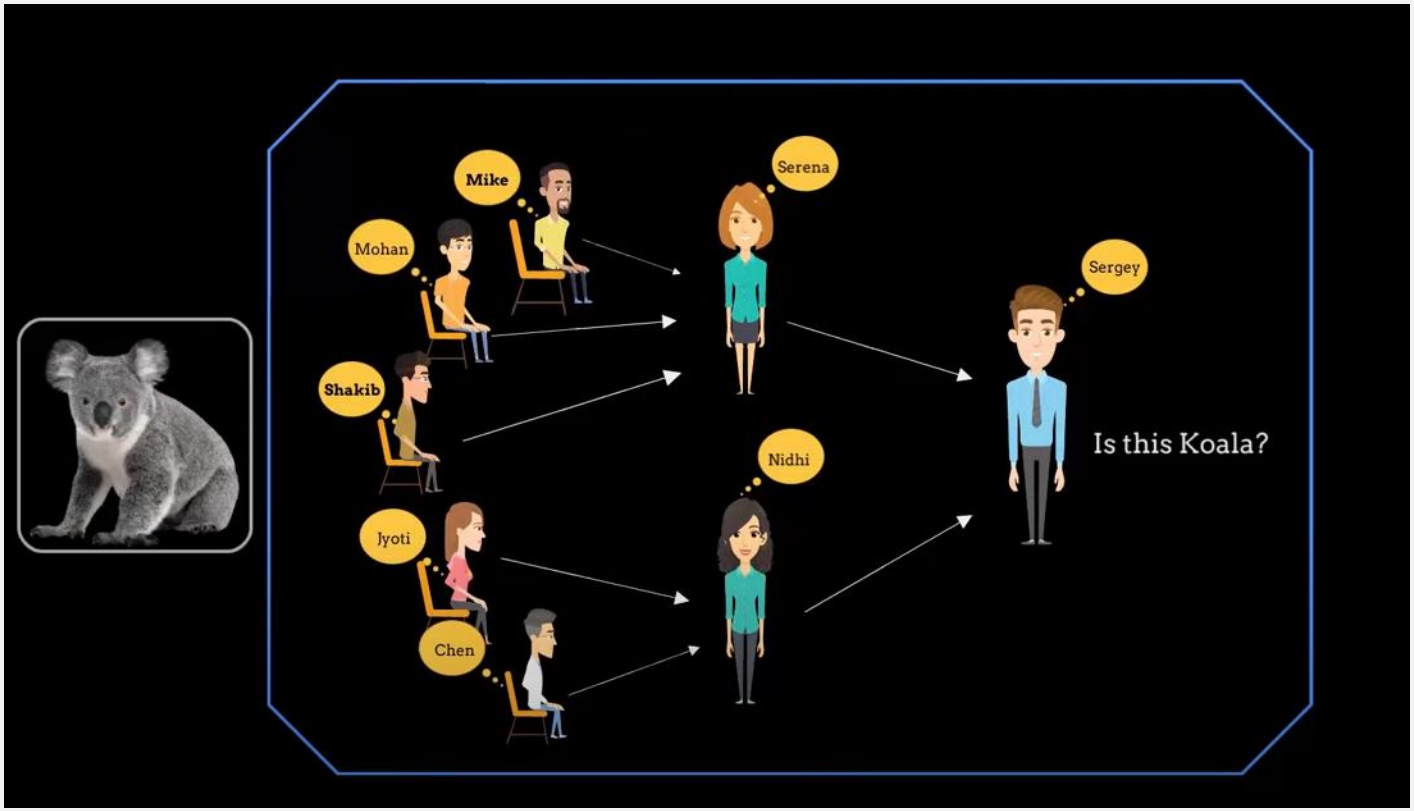
ANN in Simple Way



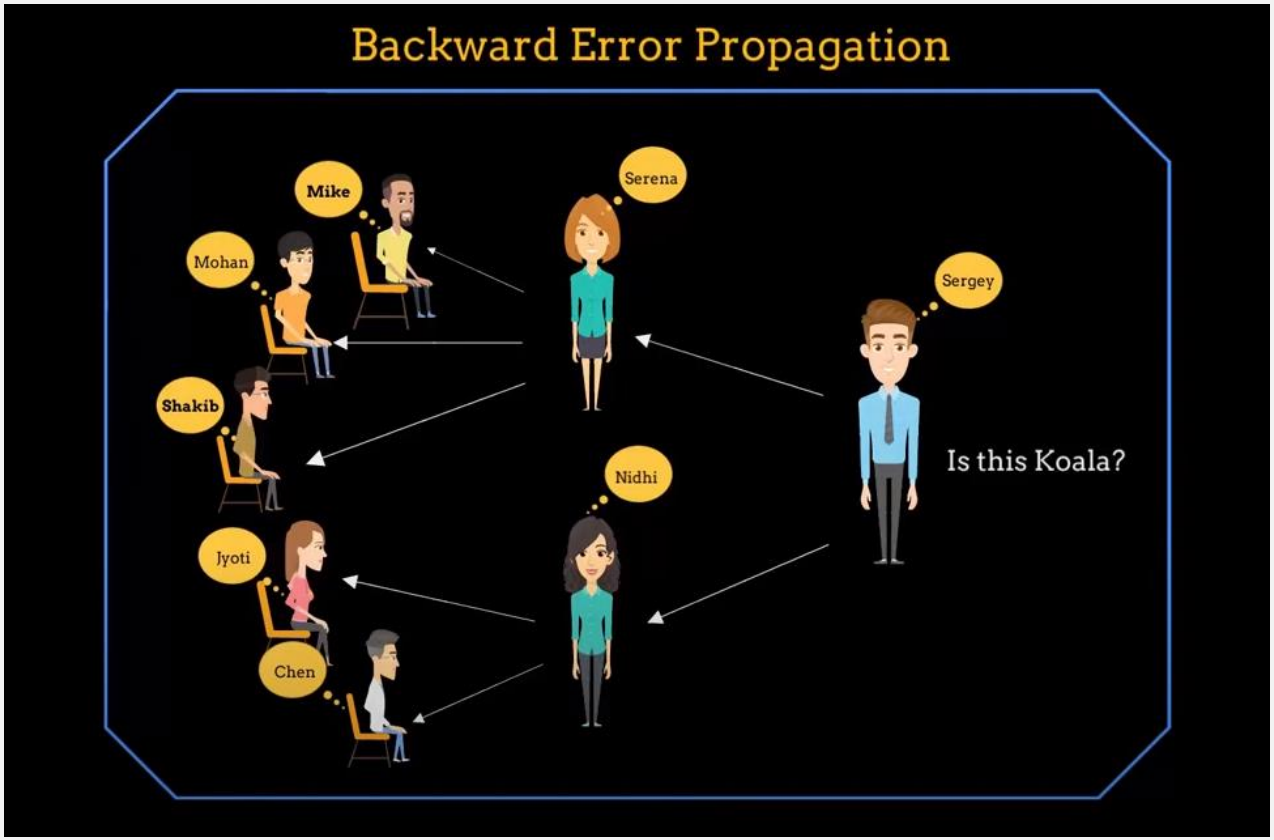
ANN in Simple Way



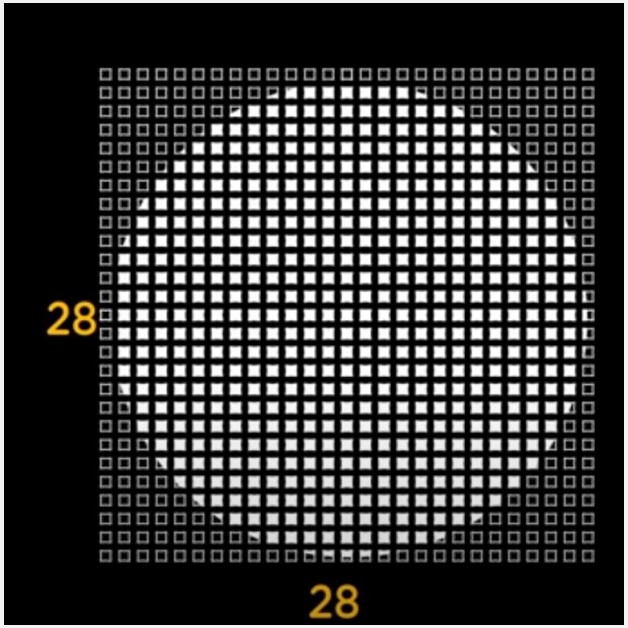
ANN in Simple Way

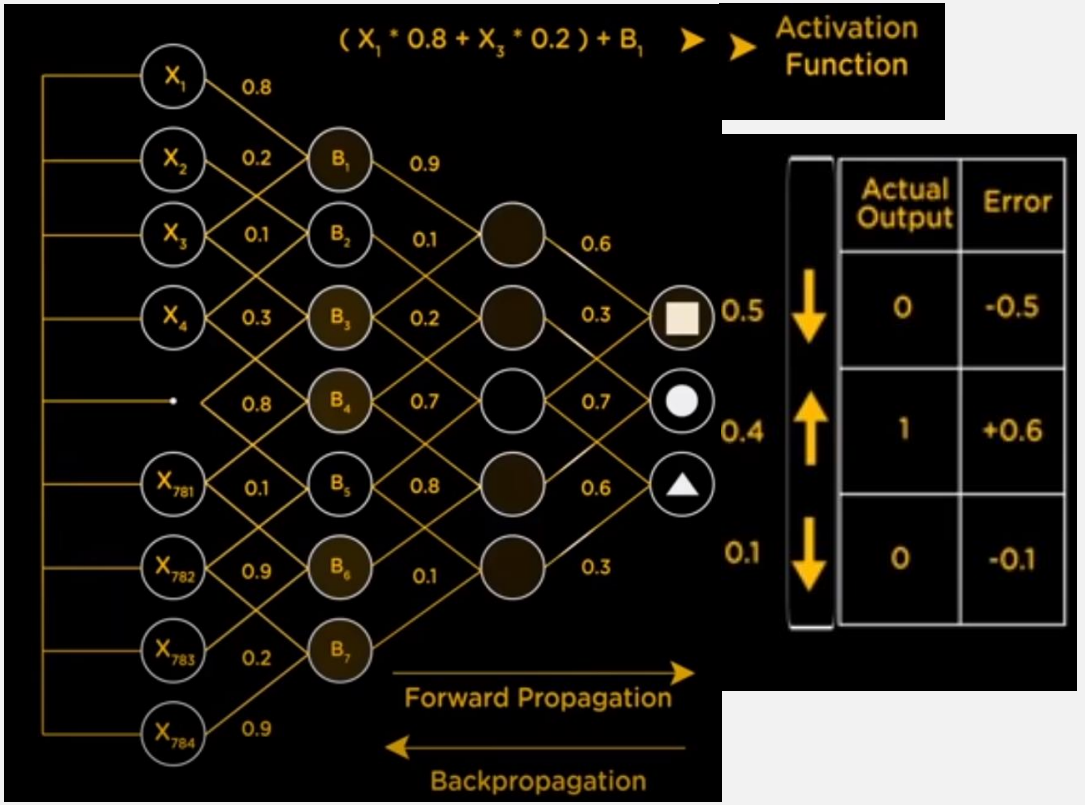


ANN in Simple Way



ANN in Simple Way





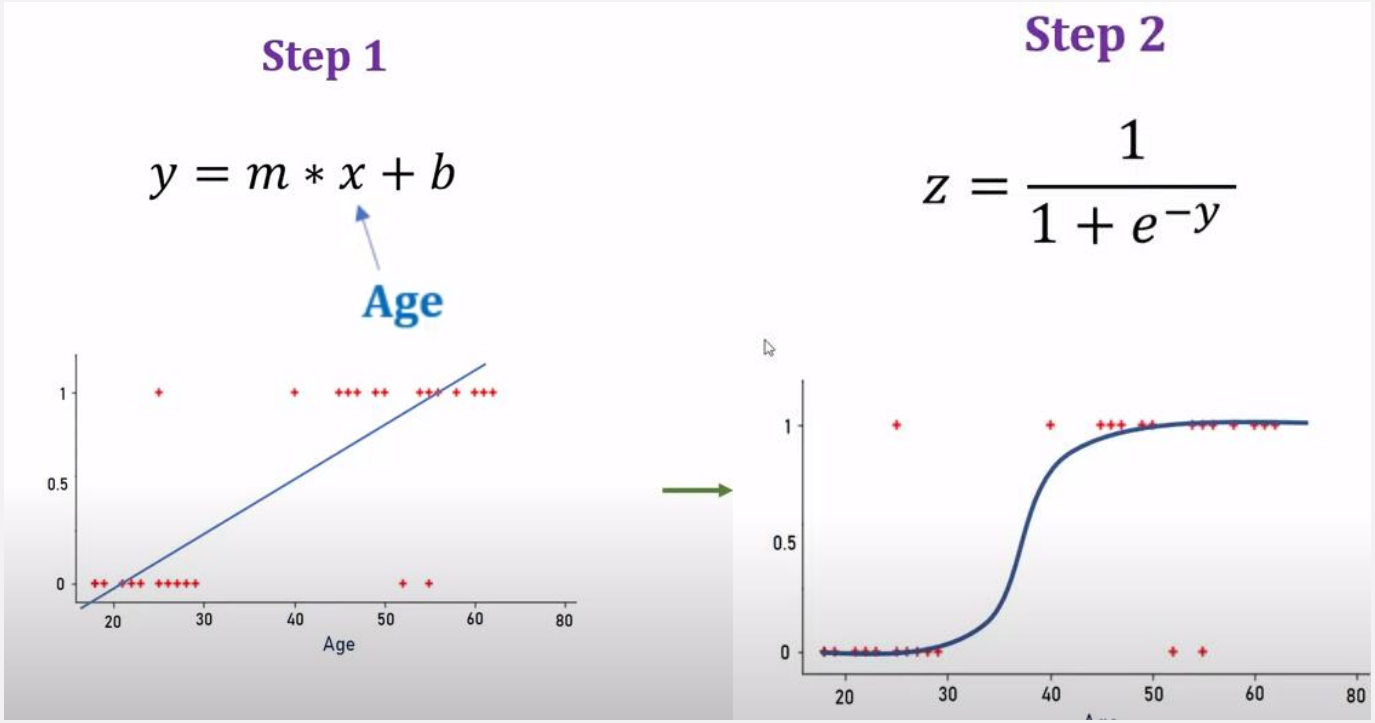
What is Neuron ?

age	have_insurance
22	0
25	0
47	1
52	0
46	1
56	1
55	0
60	1
62	1
61	1
18	0
28	0
27	0
29	0
49	1

Binary Classification

Given an age of a person, come up with a **function** that can predict if person will buy insurance or not

What is Neuron ?

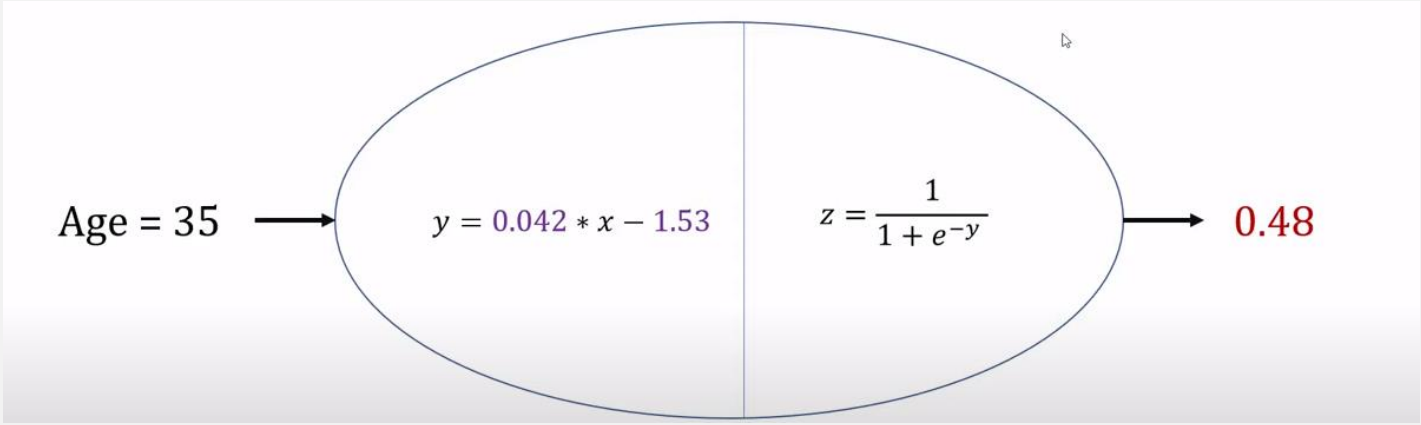


$y = 0.042$

$$sigmoid(z) = \frac{1}{1 + e^{-z}}$$

e = Euler's number ~ 2.71828

What is Neuron ?



value < 0.5 = person will not buy insurance

value >= 0.5 = person **will** buy insurance

What is Neuron ?

$$y = 0.042 * x - 1.53$$

Age

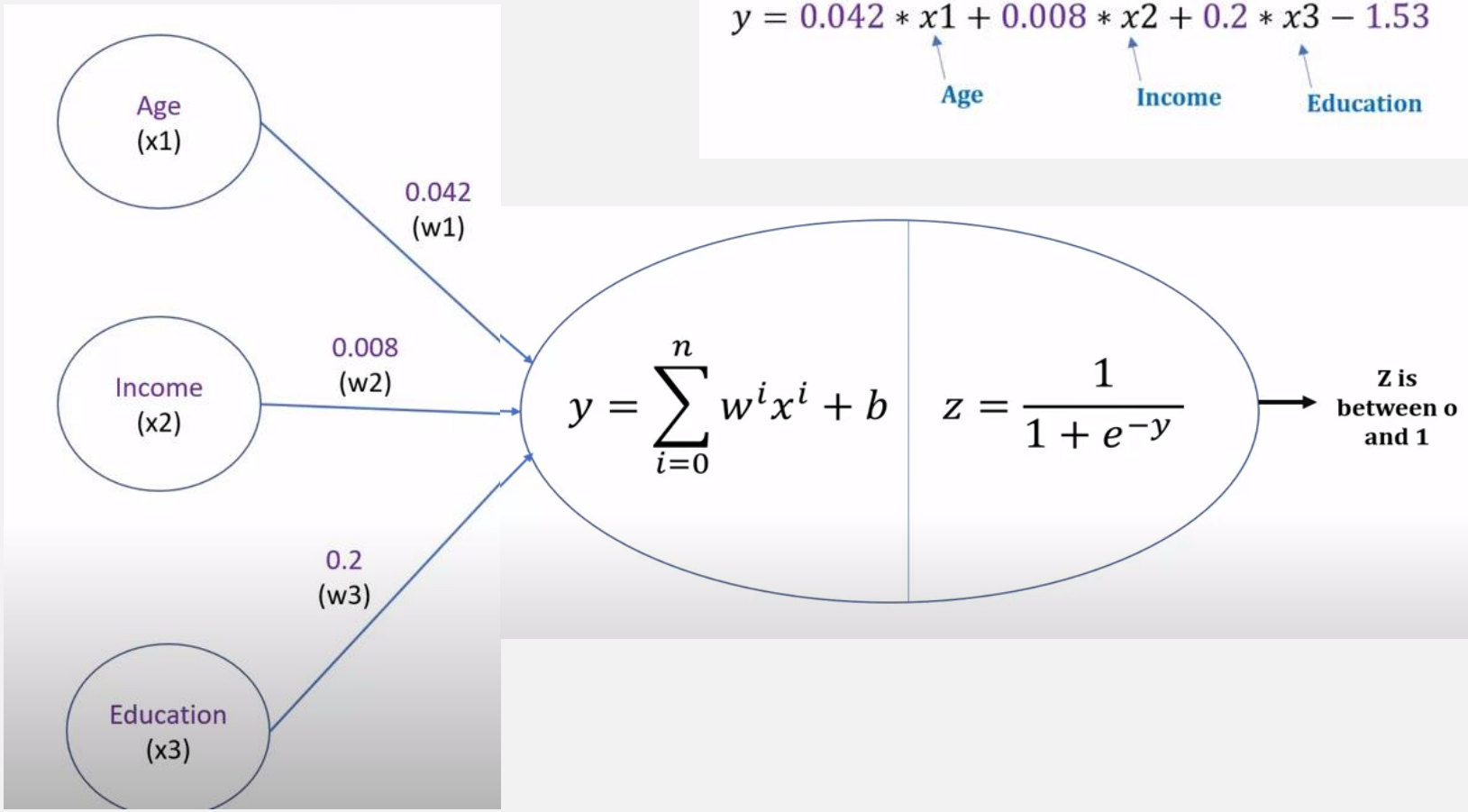
$$y = 0.042 * x1 + 0.008 * x2 + 0.2 * x3 - 1.53$$

Age Income Education

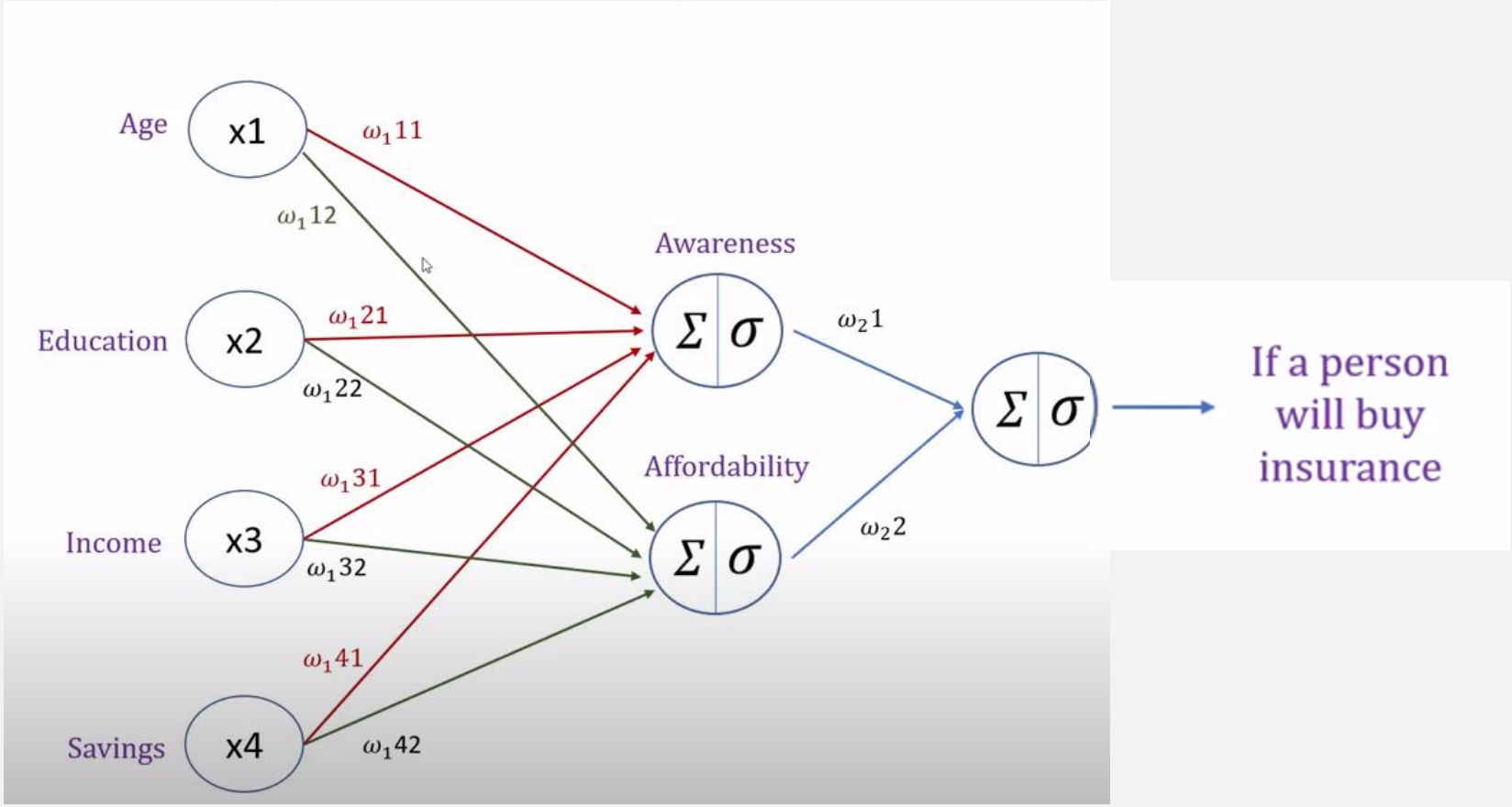
$$y = w1 * x1 + w2 * x2 + w3 * x3 + b$$

$$y = \sum_{i=0}^{n} w^i x^i + b$$

What is Neuron ?



ANN


Age
x1
$\omega_1 11$
$\omega_1 12$
Education
x2
$\omega_1 21$
$\omega_1 22$
Income
x3
$\omega_1 31$
$\omega_1 32$
Savings
x4
$\omega_1 41$
$\omega_1 42$
Awareness
Σ σ
Affordability
Σ σ
$\omega_2 1$
$\omega_2 2$
Σ σ
If a person will buy insurance

Deep Learning Platform

Keras Sequential Model

A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.

```
model = keras.Sequential(
    [
        layers.Dense(2, activation="relu"),
        layers.Dense(3, activation="relu"),
        layers.Dense(4),
    ]
)
```

```
model = keras.Sequential()
model.add(layers.Dense(2, activation="relu"))
model.add(layers.Dense(3, activation="relu"))
model.add(layers.Dense(4))
```

Keras Dense Layer

```
tf.keras.layers.Dense(
    units,
    activation=None,
    use_bias=True,
    kernel_initializer="glorot_uniform",
    bias_initializer="zeros",
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs
)
```

Keras Compiling Model

```
Model.compile(
    optimizer="rmsprop",
    loss=None,
    metrics=None,
    loss_weights=None,
    weighted_metrics=None,
    run_eagerly=None,
    steps_per_execution=None,
    **kwargs
)
```

Available optimizers

- SGD
- RMSprop
- Adam
- Adadelta
- Adagrad
- Adamax
- Nadam
- Ftrl

Keras fit method

```
Model.fit(
    x=None,
    y=None,
    batch_size=None,
    epochs=1,
    verbose="auto",
    callbacks=None,
    validation_split=0.0,
    validation_data=None,
    shuffle=True,
    class_weight=None,
    sample_weight=None,
    initial_epoch=0,
    steps_per_epoch=None,
    validation_steps=None,
    validation_batch_size=None,
    validation_freq=1,
    max_queue_size=10,
    workers=1,
    use_multiprocessing=False,
)
```

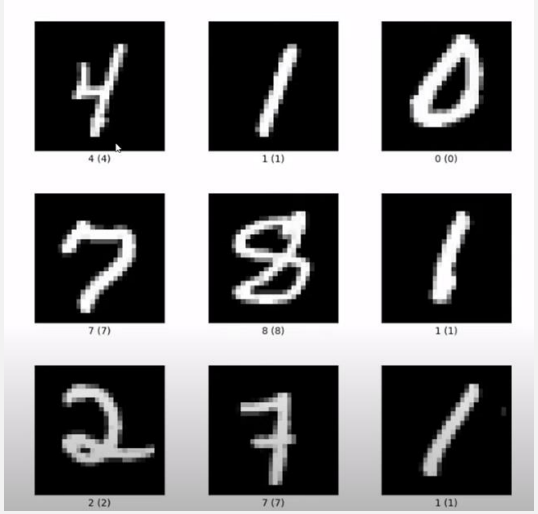
Trains the model for a fixed number of epochs (iterations on a dataset).

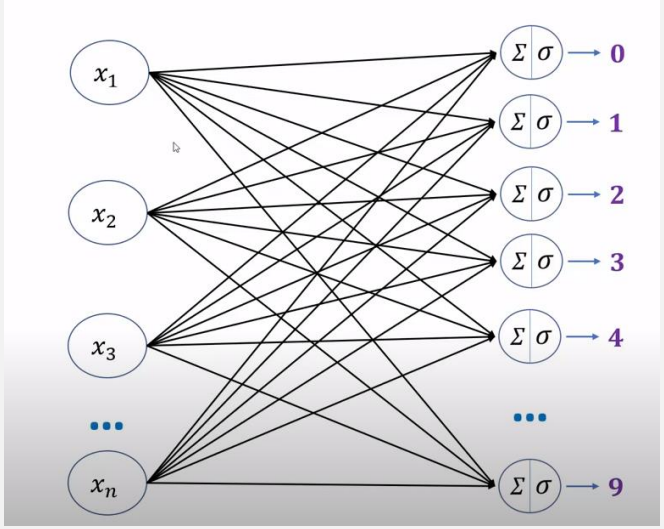
Keras Load & Save Model

```
model = ...  # Get model (Sequential, Functional Model, or Model subclass)
model.save('path/to/location')
```

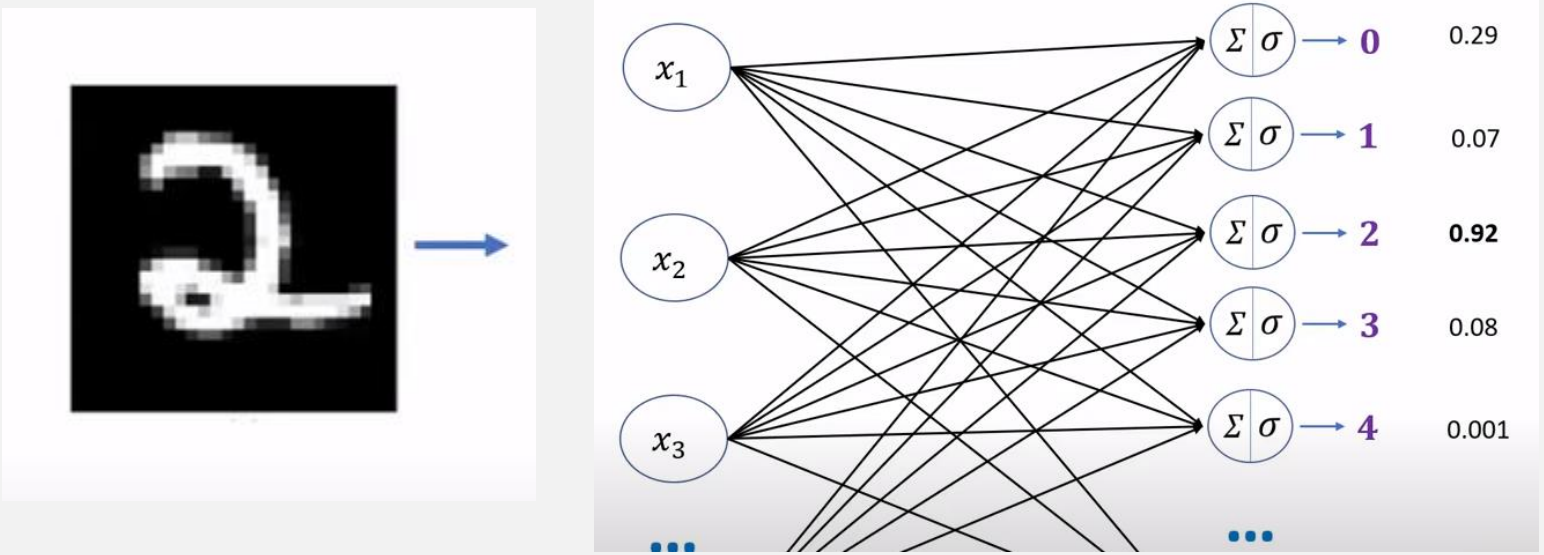
```
from tensorflow import keras
model = keras.models.load_model('path/to/location')
```

Handwritten Digit Recognition

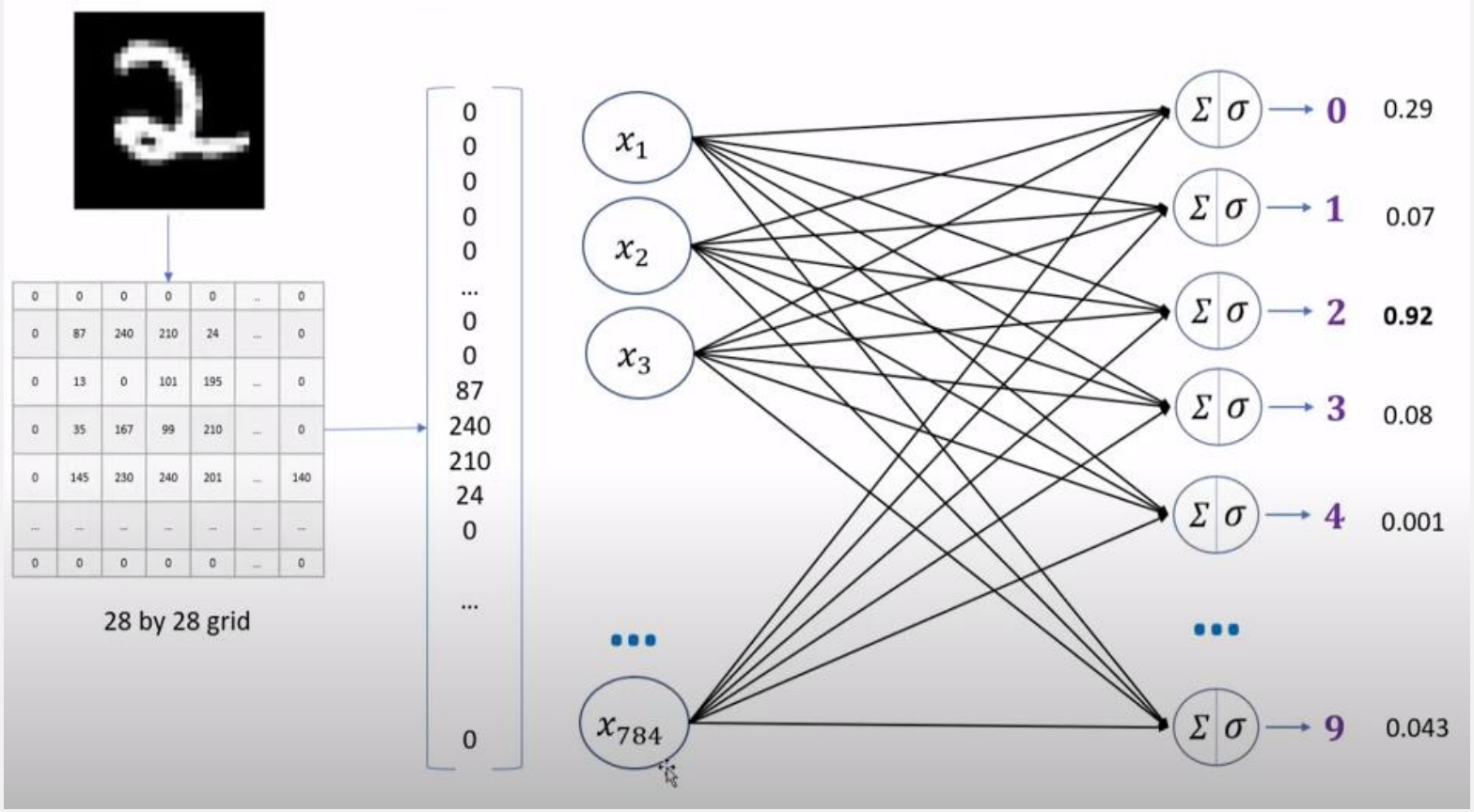




Handwritten Digit Recognization



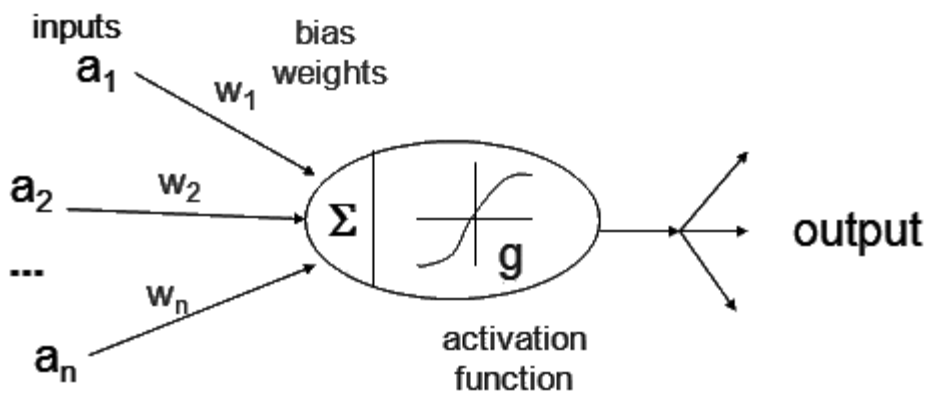
Handwritten Digit Recognition



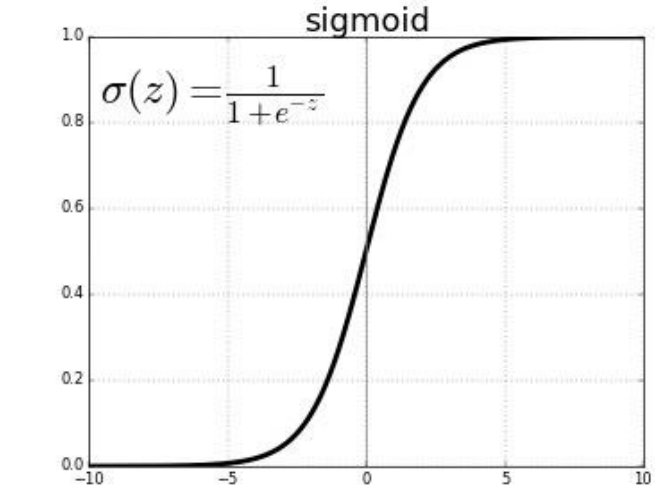
Activation Functions

Activation function decides, whether a neuron should be activated or not by calculating weighted sum and further adding bias with it.

The purpose of the activation function is to introduce non-linearity into the output of a neuron.



Activation Functions

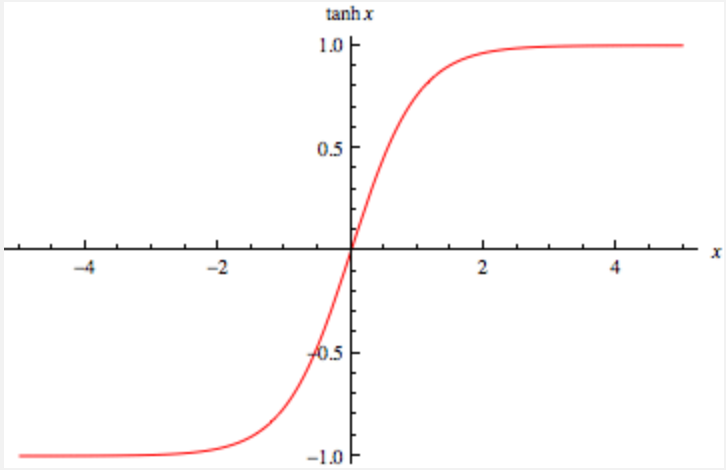


$$sigmoid(z) = \frac{1}{1 + e^{-z}}$$

Value Range : 0 to 1

Uses : Usually used in output layer of a binary classification, where result is either 0 or 1, as value for sigmoid function lies between 0 and 1 only so, result can be predicted easily to be ***1*** if value is greater than **0.5** and ***0*** otherwise.

Activation Functions

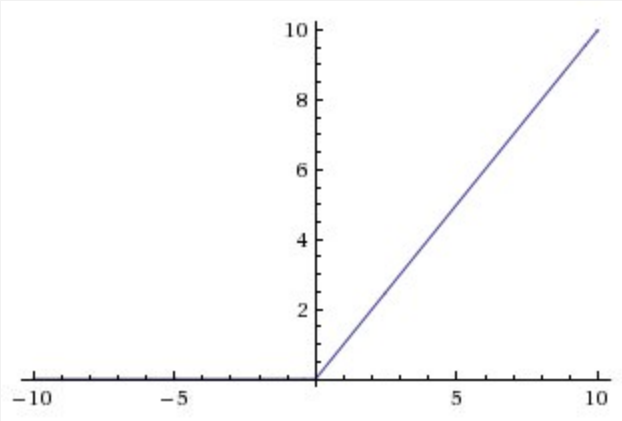


$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

Value Range :- -1 to +1

Uses :- Usually used in hidden layers of a neural network as it's values lies between **-1 to 1** hence the mean for the hidden layer comes out be 0 or very close to it, hence helps in *centering the data* by bringing mean close to 0. This makes learning for the next layer much easier.

Activation Functions



$$ReLU(z) = \max(0, x)$$

For hidden layers if you are not sure which activation function to use, just use ReLU as your default choice

Rectified Linear Unit(ReLU)

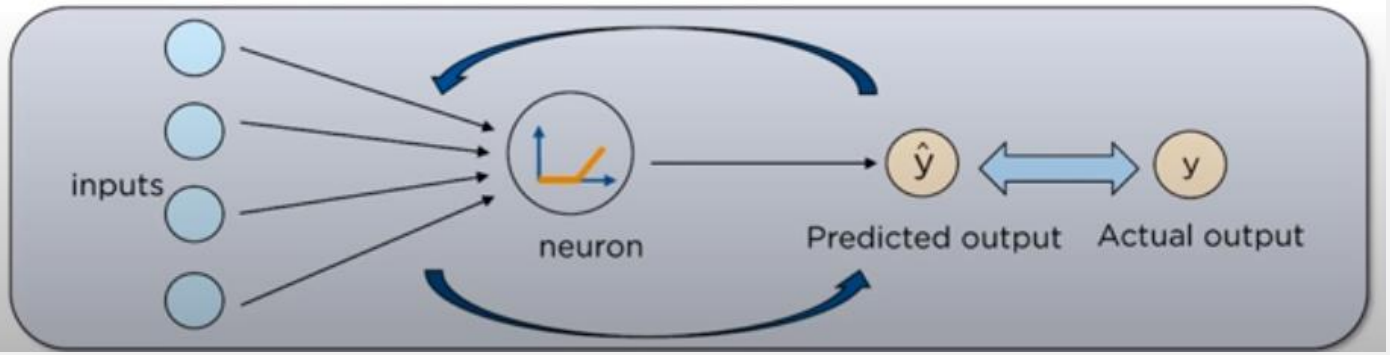
Rectified linear unit or ReLU is most widely used activation function right now which ranges from **0 to infinity**, All the negative values are converted into zero,

Loss Function

The purpose of loss functions is to compute the quantity that a model should seek to minimize during training.

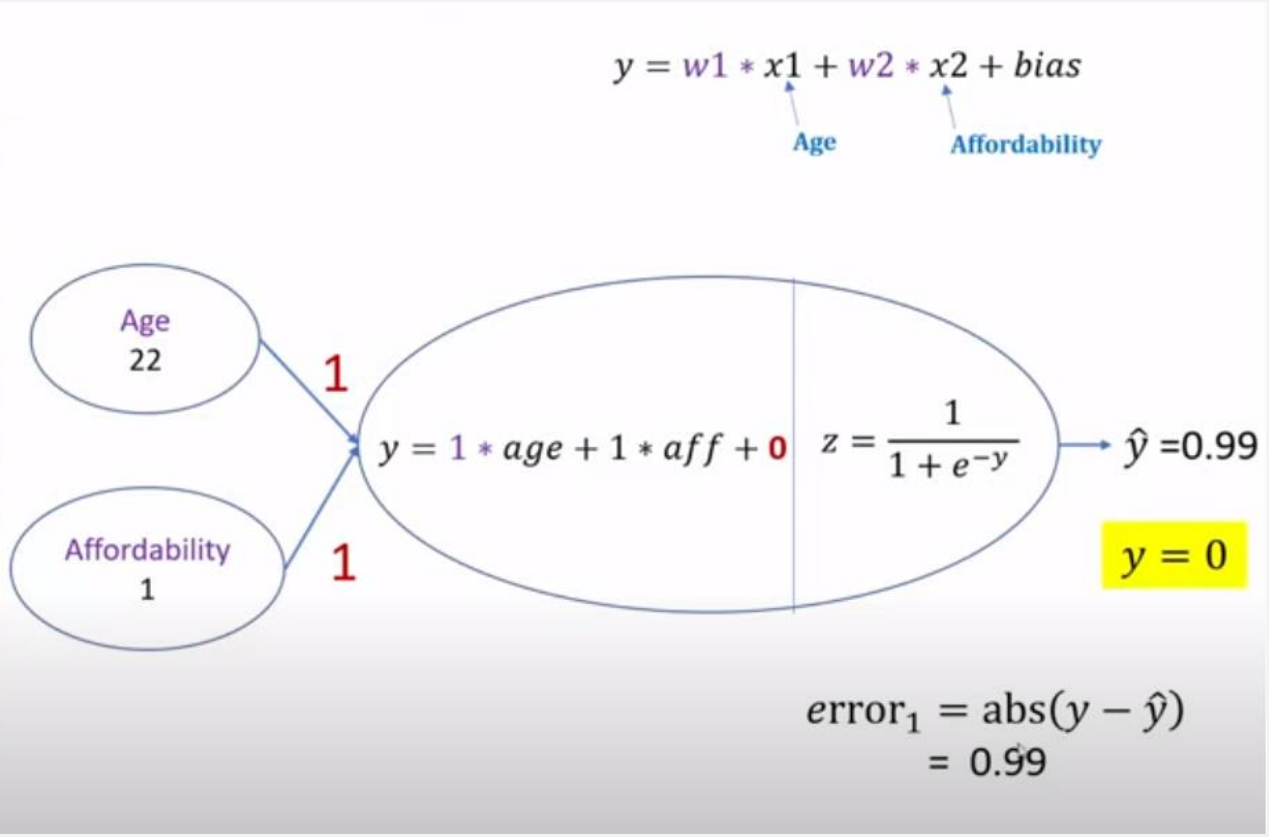
The *Cost* value is the difference between the neural nets predicted output and the actual output from a set of labeled training data

The least cost value is obtained by making adjustments to the weights and biases iteratively throughout the training process



Loss/Cost Function

age	affordability	have_insurance
22	1	0
25	0	0
47	1	1
52	0	0
46	1	1
56	1	1
55	0	0
60	0	1
62	1	1
61	1	1
18	1	0
28	1	0
27	0	1



Loss/Cost Function

$$\text{Total error} = error_1 + error_2 + \ldots + error_{13}$$

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

Loss

$$Mean\ Absolute\ Error\ (MAE) = \frac{1}{n}\sum_{i=1}^{n} abs(y_i - \hat{y}_i)$$

Cost Function

Loss/Cost Function

$$Mean\ Absolute\ Error\ (MAE) = \frac{1}{n}\sum_{i=1}^{n} abs(y_i - \hat{y}_i)$$

```
model.compile(optimizer='adam',
              loss='mean_absolute_error',
              metrics=['accuracy'])
```

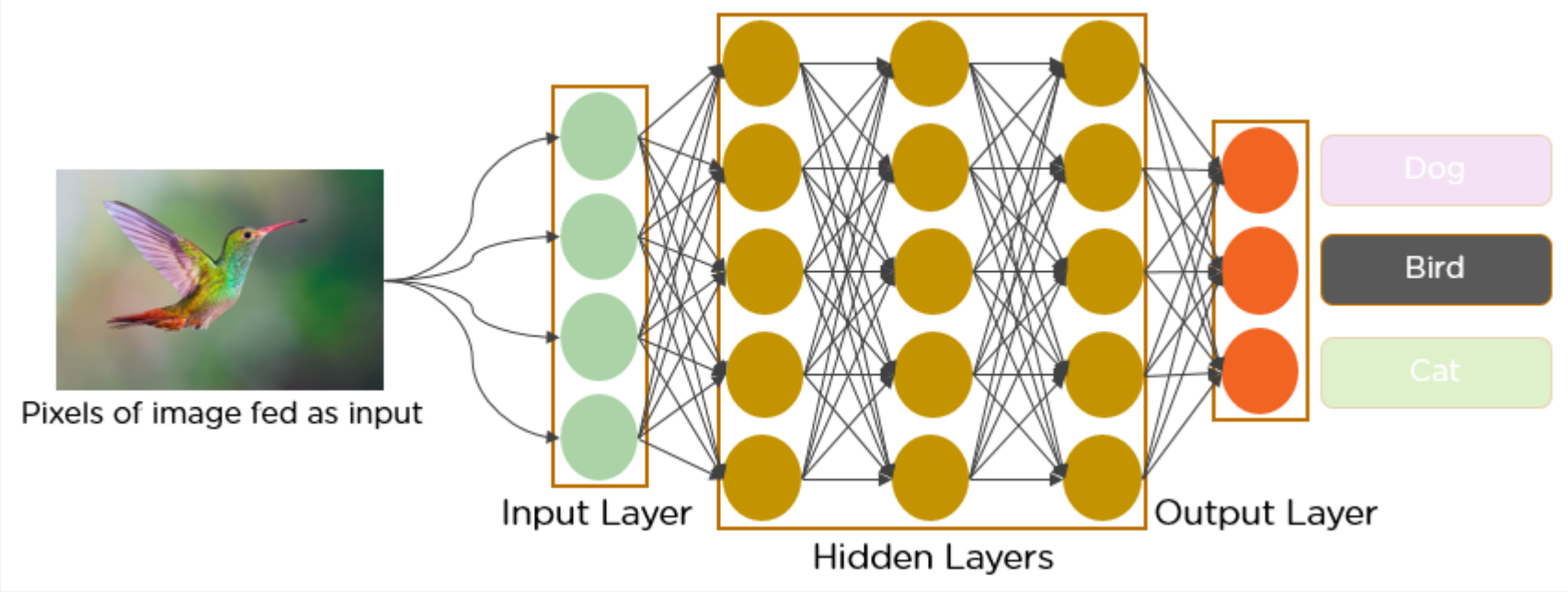
$$Mean\ Squared\ Error\ (MSE) = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

```
model.compile(optimizer='adam',
              loss='mean_squared_error',
              metrics=['accuracy'])
```

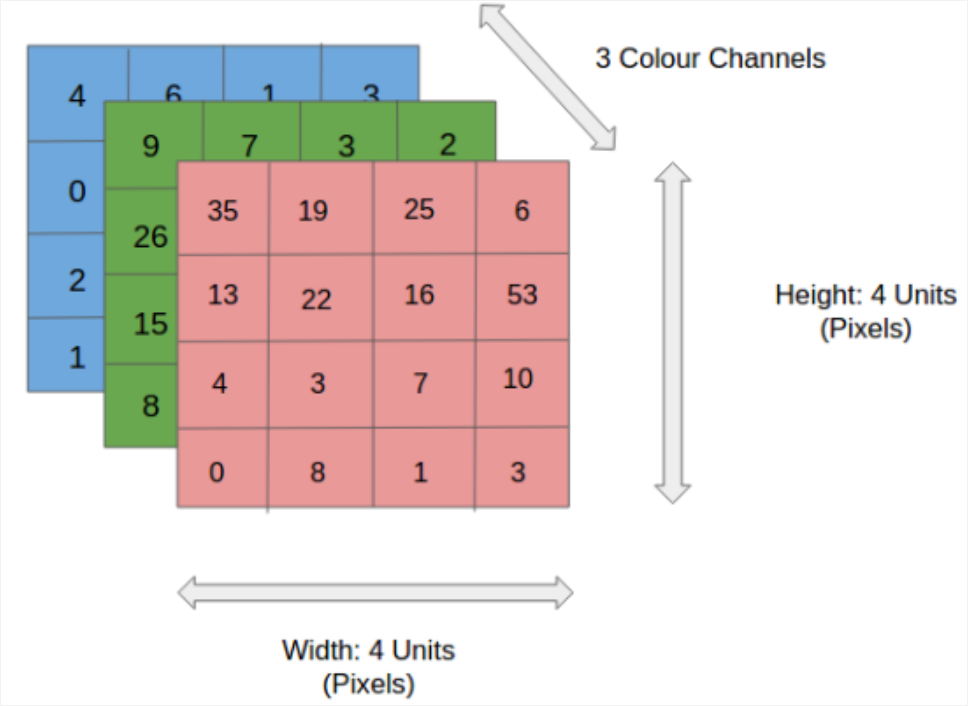
- sparse_categorical_crossentropy
- binary_crossentropy
- categorical_crossentropy
- mean_absolute_error
- mean_squared_error

Convolutional Neural Network

A **Convolutional Neural Network (ConvNet/CNN)** is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.



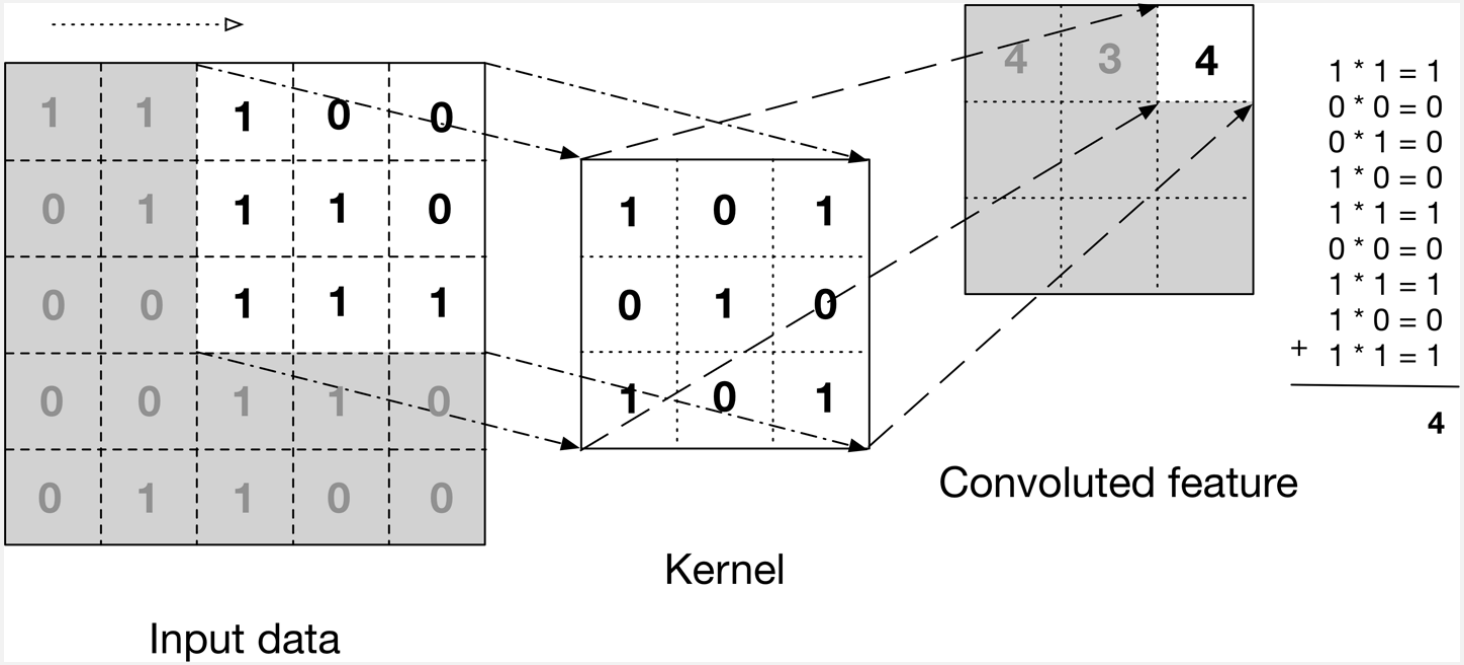
RGB IMAGE



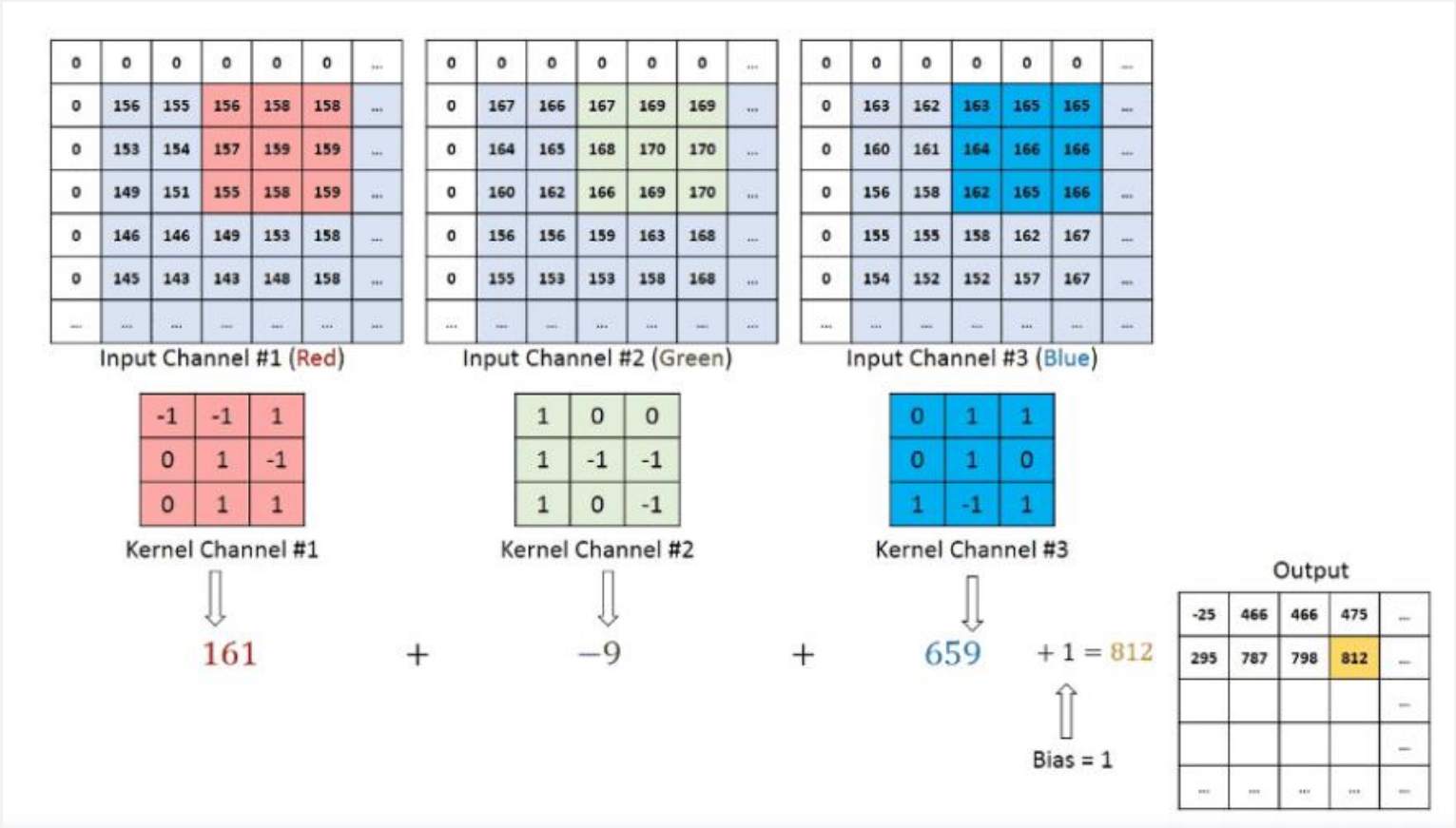
An RGB image is nothing but a matrix of pixel values having three planes whereas a grayscale image is the same but it has a single plane

Convolution Layer

We take a filter/kernel(3×3 matrix) and apply it to the input image to get the convolved feature. This convolved feature is passed on to the next layer.



Convolution - RGB



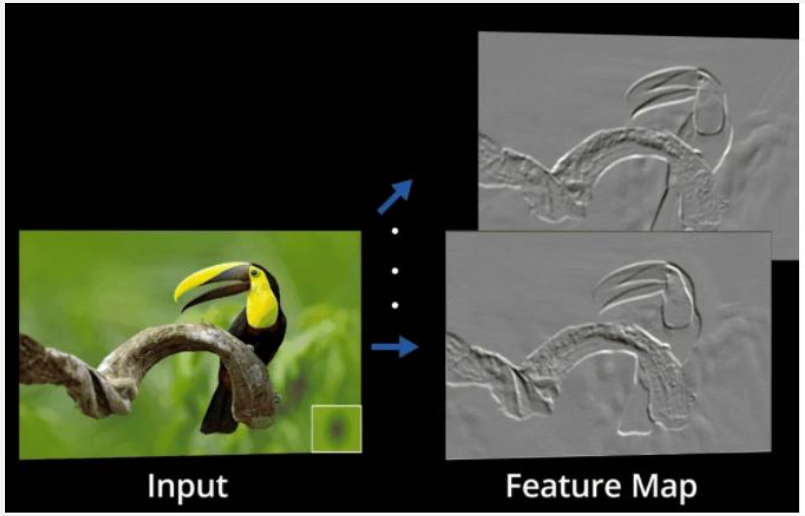
RELU Layer

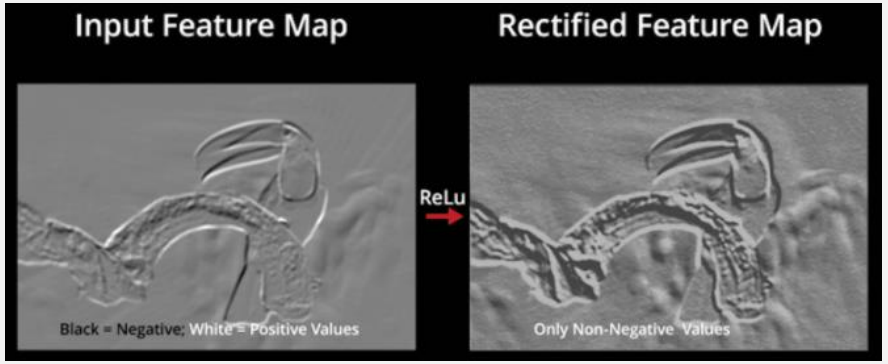
ReLU stands for the rectified linear unit. Once the feature maps are extracted, the next step is to move them to a ReLU layer.

ReLU performs an element-wise operation and sets all the negative pixels to 0. It introduces non-linearity to the network, and the generated output is a rectified feature map.

Feature Map

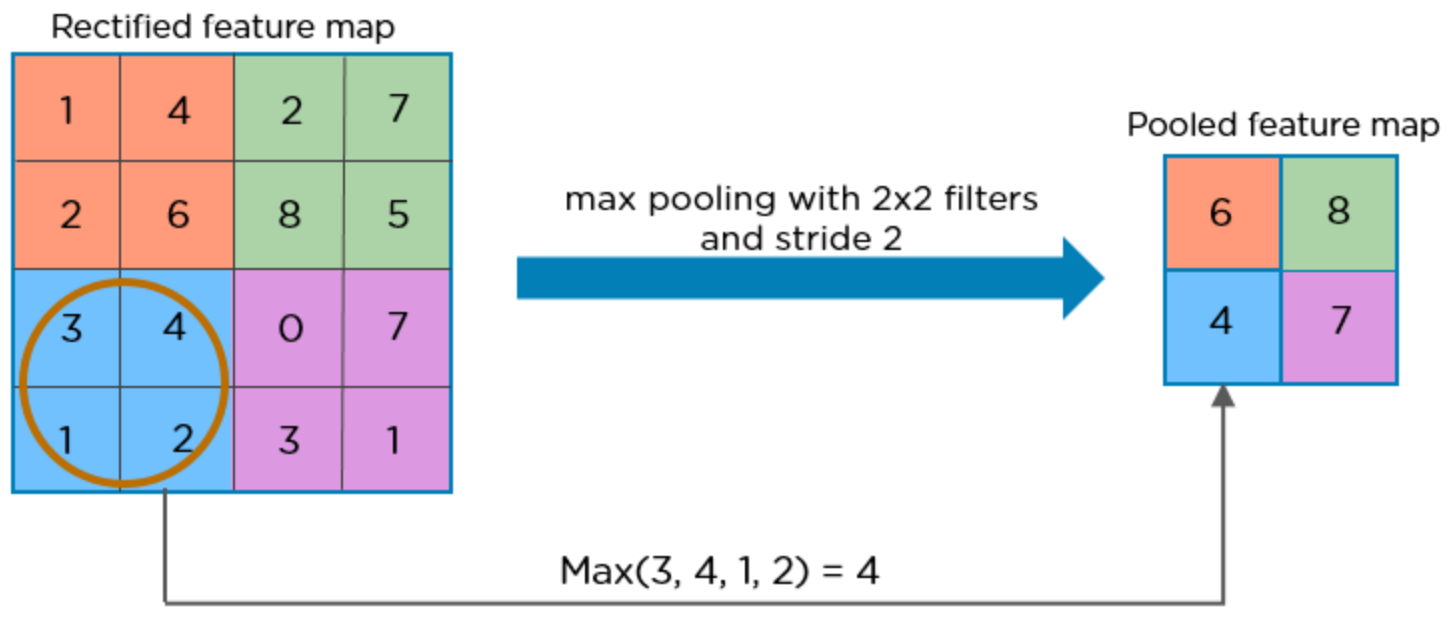
The original image is scanned with multiple convolutions and ReLU layers for locating the features





Pooling Layer

Pooling is a down-sampling operation that reduces the dimensionality of the feature map. The rectified feature map now goes through a pooling layer to generate a pooled feature map

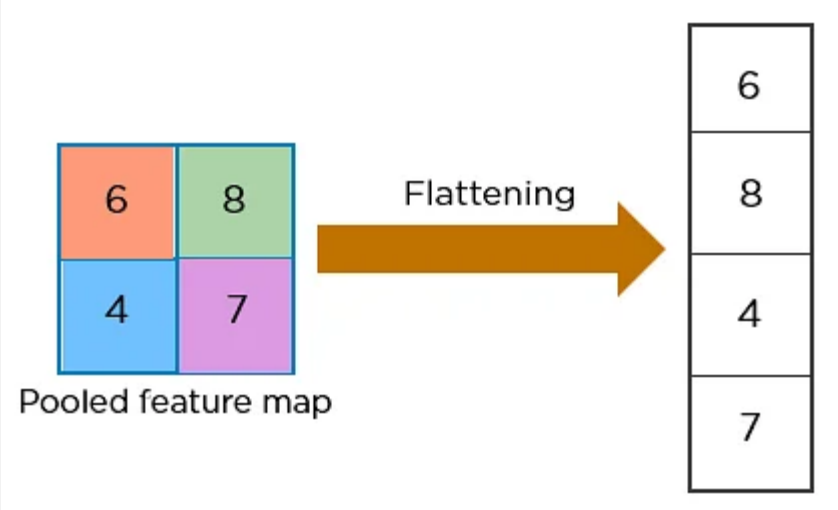


Pooling Layer

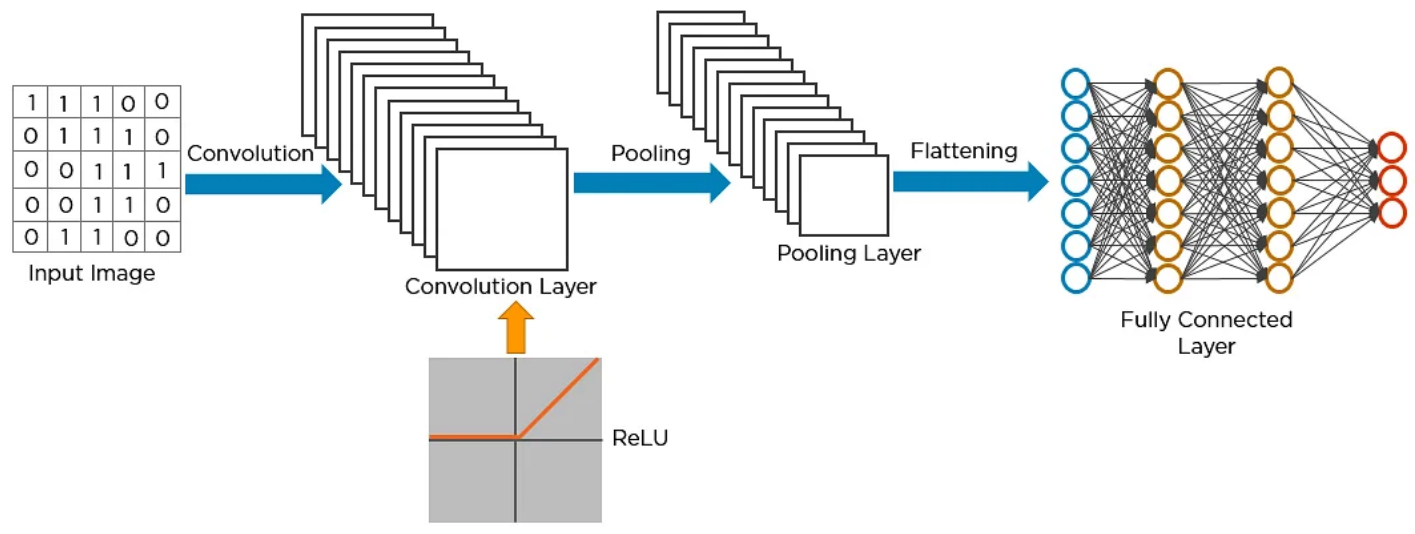
The pooling layer uses various filters to identify different parts of the image like edges, corners, body, feathers, eyes, and beak.

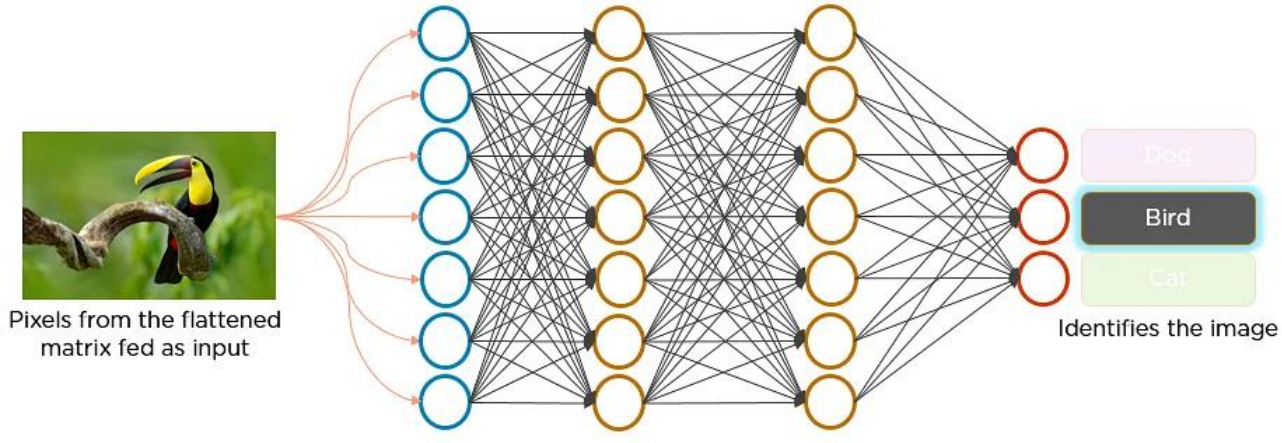
Flattening Layer

The next step in the process is called flattening. Flattening is used to convert all the resultant 2-Dimensional arrays from pooled feature maps into a single long continuous linear vector.



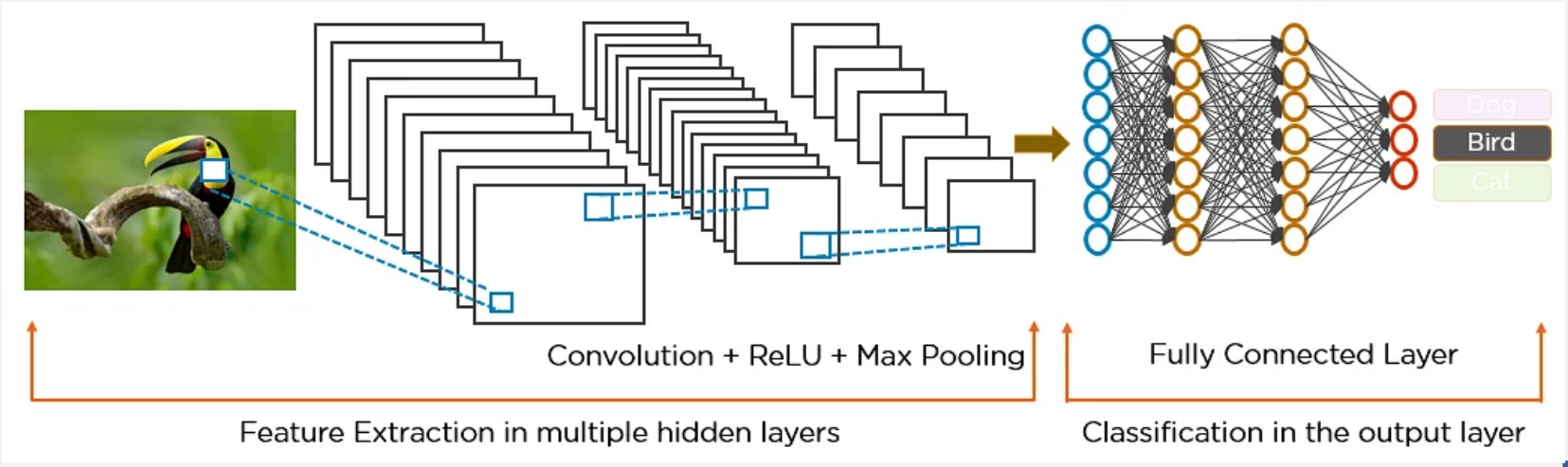
Final CNN


1 1 1 0 0
0 1 1 1 0
0 0 1 1 1
0 0 1 1 0
0 1 1 0 0
Input Image
Convolution
Convolution Layer
ReLU
Pooling
Pooling Layer
Flattening
Fully Connected Layer


Pixels from the flattened matrix fed as input
Dog
Bird
Cat
Identifies the image

CNN Steps

- The pixels from the image are fed to the convolutional layer that performs the convolution operation
- It results in a convolved map
- The convolved map is applied to a ReLU function to generate a rectified feature map
- The image is processed with multiple convolutions and ReLU layers for locating the features
- Different pooling layers with various filters are used to identify specific parts of the image
- The pooled feature map is flattened and fed to a fully connected layer to get the final output



THANK YOU !!!

Amol Patil - 9822291613