**UCS2612 Machine Learning Laboratory**

**Assignment 2 : Handwritten Character Recognition using Neural Networks**

Name **: Anandh K**

Roll No **: 3122 21 5001 009**

**---------------------------------------------------------------------------------------------------------**

# Aim

To Develop a python program to recognize handwritten characters using Neural Network (NN) Model.

# Code

**Importing the Necessary Libraries For Single layer perceptron model , Multiple Layer Perceptron Model and CNN Using Keras**

from google.colab import drive

drive.mount('/content/drive')

import numpy as np import pandas as pd

import matplotlib.pyplot as plt

import os

from PIL import Image import cv2

from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import accuracy\_score from sklearn.linear\_model import Perceptron

from sklearn.neural\_network import MLPClassifier

import tensorflow as tf

**Loading the dataset and Pre-Processing the data (Image Enhancement techniques)**

# Image resize

1. **Normalization of image frequencies**

data\_dir = "/content/drive/MyDrive/Image\_dataset/Img"

x=[]

x1=[]

x2=[]

for image in os.listdir(data\_dir):

image\_path = os.path.join(data\_dir, image)

img = Image.open(image\_path).resize((100, 100)) img\_array = np.array(img)

img\_array1 = np.array(img).flatten() / 255.0 img\_array2 = np.array(img).flatten() / 255.0 x.append(img\_array)

x1.append(img\_array1) x2.append(img\_array2)

x = np.array(x) x1 = np.array(x1)

x2 = np.array(x1)

# Printing the Sample Images

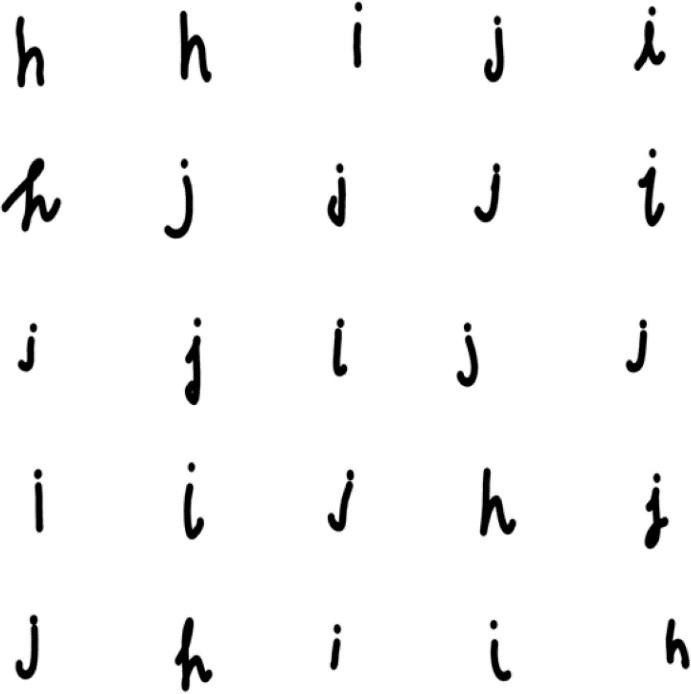
print("the no of Images in dataset : ",len(x1))



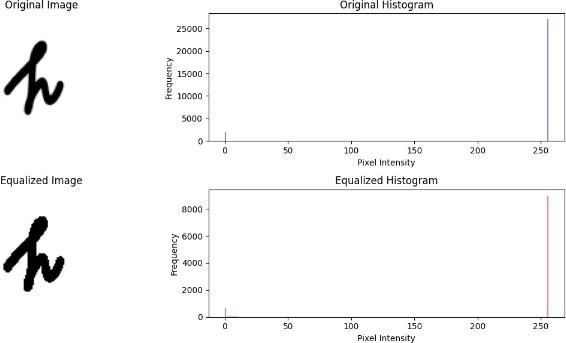
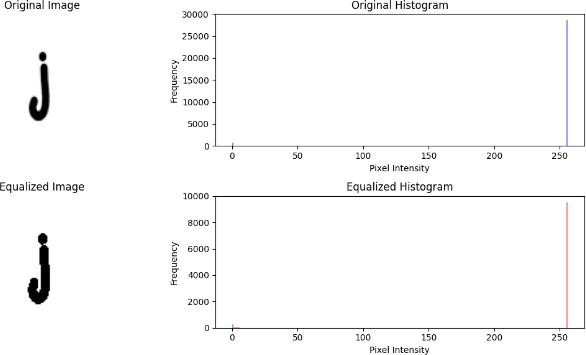
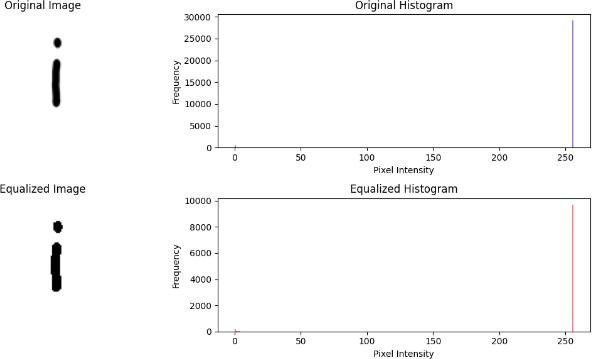
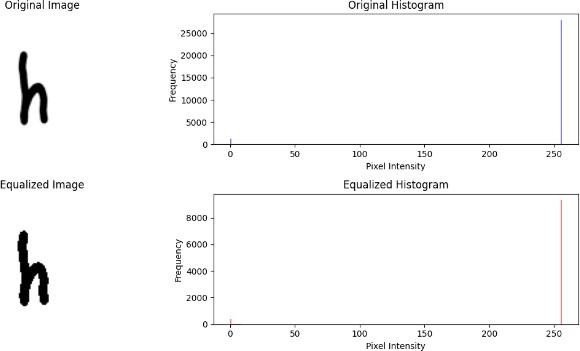
plt.figure(figsize=(10, 10)) for i in range(min(25, len(x))):

plt.subplot(5, 5, i + 1) plt.imshow(x[i], cmap='gray') plt.axis('off')

plt.show()



# Exploratory Data Analysis. Histogram For The Images

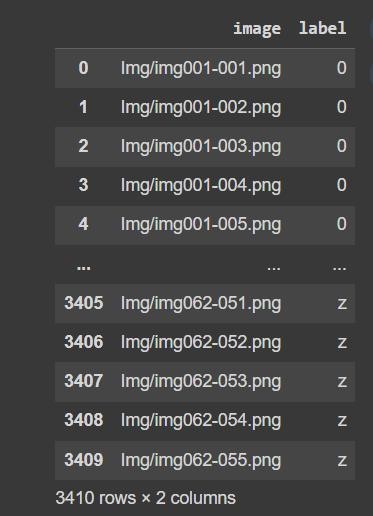


**Split the data into training, testing and validation sets**

y=pd.read\_csv("/content/drive/MyDrive/english.csv")

len(y)

y

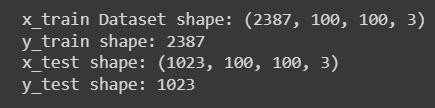


# CNN Model

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.3, random\_state=42)

print("x\_train Dataset shape:", x\_train.shape) print("y\_train shape:", len(y\_train)) print("x\_test shape:", x\_test.shape)

print("y\_test shape:", len(y\_test))



y\_train\_array = y\_train.to\_numpy() y\_test\_array = y\_test.to\_numpy()

y\_train\_indices = np.argmax(y\_train\_array, axis=1) y\_test\_indices = np.argmax(y\_test\_array, axis=1)

label\_encoder = LabelEncoder()

y\_train\_encoded = label\_encoder.fit\_transform(y\_train\_indices) y\_test\_encoded = label\_encoder.transform(y\_test\_indices)

model = tf.keras.Sequential([

tf.keras.layers.Conv2D(32, (3, 3), activation='relu',

input\_shape=(100, 100, 3)),

tf.keras.layers.MaxPooling2D((2, 2)),

tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),

tf.keras.layers.MaxPooling2D((2, 2)),

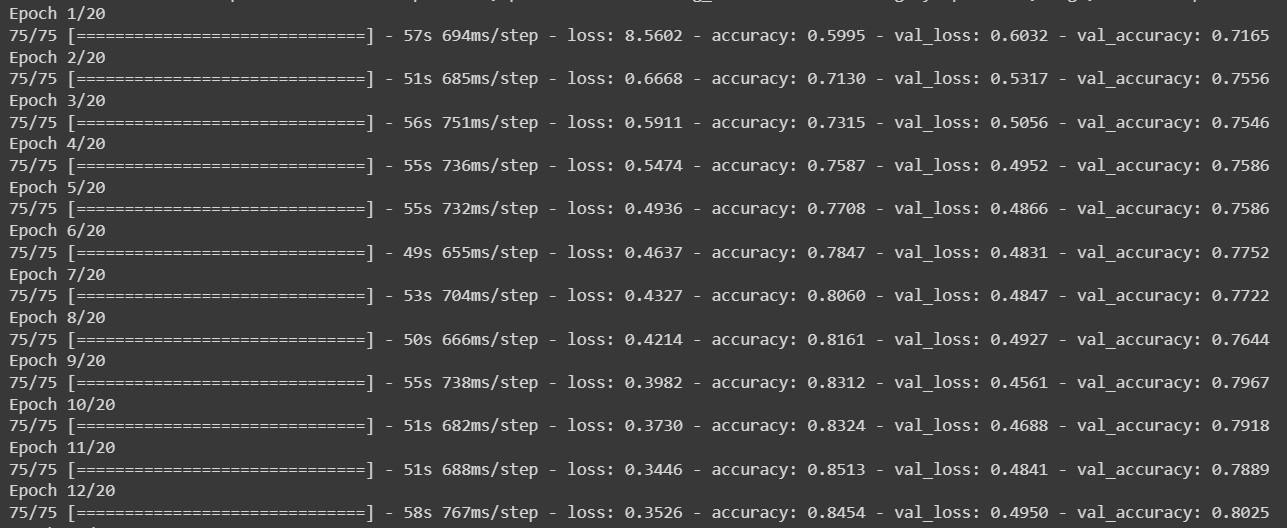
tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),

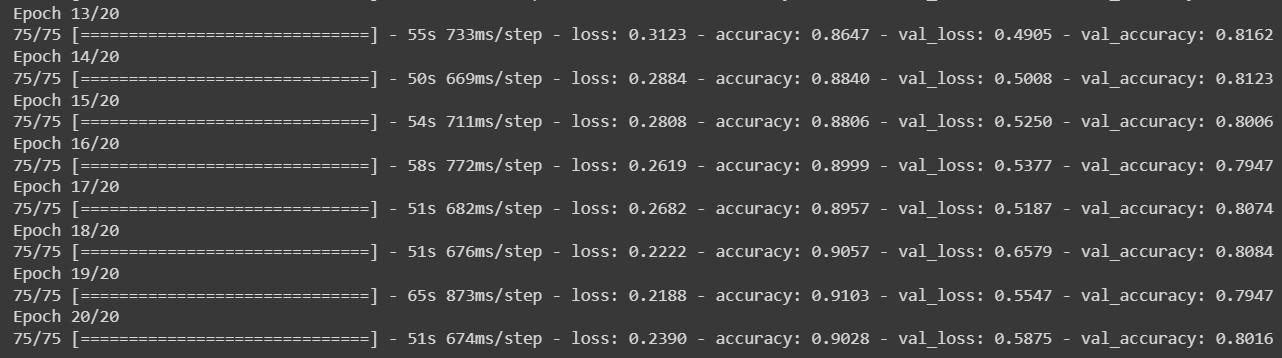
tf.keras.layers.MaxPooling2D((2, 2)), tf.keras.layers.Flatten(), tf.keras.layers.Dense(256, activation='relu'), tf.keras.layers.Dropout(0.5), tf.keras.layers.Dense(128, activation='relu'), tf.keras.layers.Dropout(0.5), tf.keras.layers.Dense(62, activation='softmax')

])

model.compile(optimizer=tf.keras.optimizers.Adam(lr=.01), loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

history = model.fit(x\_train, y\_train\_encoded, epochs=20, validation\_data=(x\_test, y\_test\_encoded))





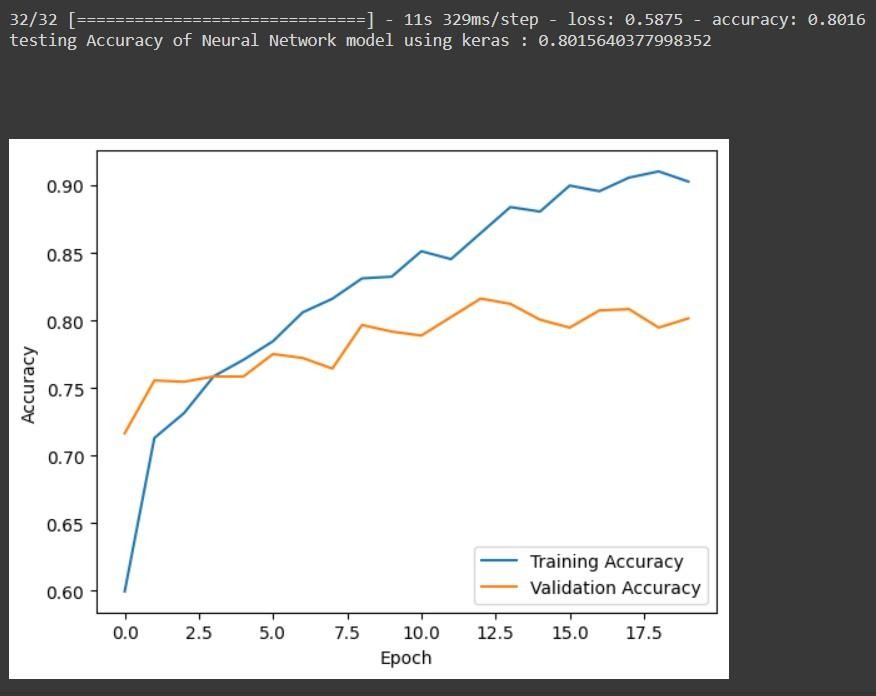
test\_loss, test\_accuracy = model.evaluate(x\_test, y\_test\_encoded) print(f"testing Accuracy of Neural Network model using keras :

{test\_accuracy}\n\n\n\n")

plt.plot(history.history['accuracy'], label='Training Accuracy') plt.plot(history.history['val\_accuracy'], label='Validation Accuracy') plt.xlabel('Epoch')

plt.ylabel('Accuracy') plt.legend(loc='lower right')

plt.show()



**Neural network model Using Single Layer perceptron model**

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x1, y, test\_size=0.3, random\_state=42)

y\_train\_array = y\_train.to\_numpy() y\_test\_array = y\_test.to\_numpy()

y\_train\_indices = np.argmax(y\_train\_array, axis=1) y\_test\_indices = np.argmax(y\_test\_array, axis=1)

label\_encoder = LabelEncoder()

y\_train\_encoded = label\_encoder.fit\_transform(y\_train\_indices) y\_test\_encoded = label\_encoder.transform(y\_test\_indices)

perceptron\_model = Perceptron(max\_iter=20, eta0=0.01)

train\_accuracies = [] val\_accuracies = []

for epoch in range(20):

perceptron\_model.partial\_fit(x\_train, y\_train\_encoded, classes=np.unique(y\_train\_encoded))

y\_train\_pred = perceptron\_model.predict(x\_train)

train\_accuracy = accuracy\_score(y\_train\_encoded, y\_train\_pred) train\_accuracies.append(train\_accuracy)

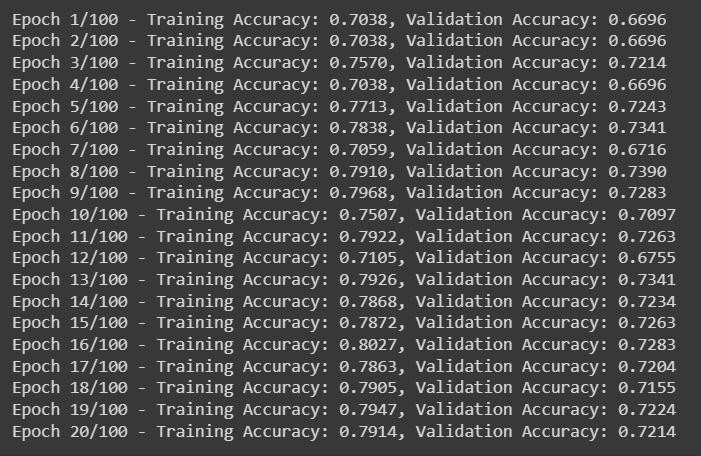
y\_val\_pred = perceptron\_model.predict(x\_test)

val\_accuracy = accuracy\_score(y\_test\_encoded, y\_val\_pred) val\_accuracies.append(val\_accuracy)

print(f"Epoch {epoch+1}/100 - Training Accuracy:

{train\_accuracy:.4f}, Validation Accuracy: {val\_accuracy:.4f}")

y\_test\_pred = perceptron\_model.predict(x\_test)



test\_accuracy = accuracy\_score(y\_test\_encoded, y\_test\_pred)

print(f"\n\n\nTesting Accuracy fo Single Layer perceptron model :

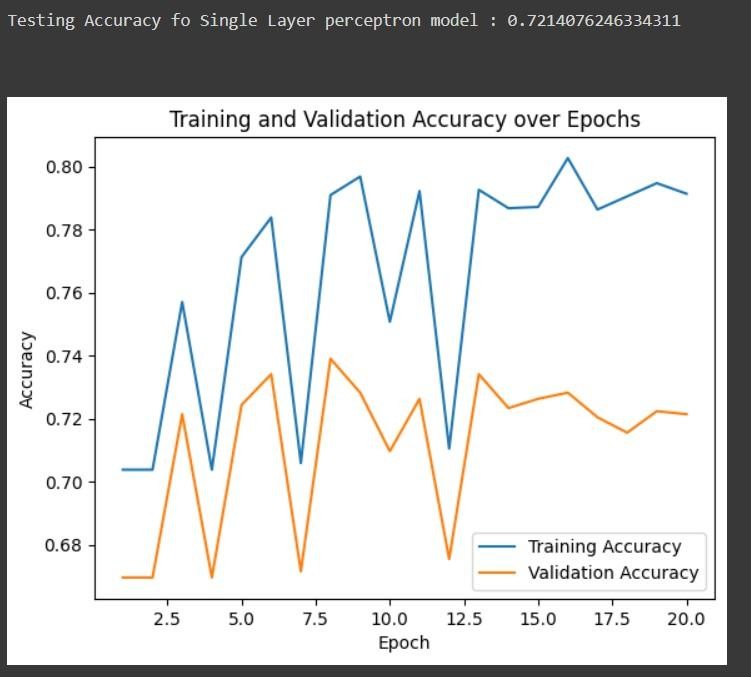
{test\_accuracy}\n\n\n")

plt.plot(range(1, 21), train\_accuracies, label='Training Accuracy') plt.plot(range(1, 21), val\_accuracies, label='Validation Accuracy') plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend(loc='lower right')

plt.title('Training and Validation Accuracy over Epochs') plt.show()



# Neural network model Using multiple Layer perceptron model

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x2, y, test\_size=0.3, random\_state=42)

y\_train\_array = y\_train.to\_numpy() y\_test\_array = y\_test.to\_numpy()

y\_train\_indices = np.argmax(y\_train\_array, axis=1) y\_test\_indices = np.argmax(y\_test\_array, axis=1)

label\_encoder = LabelEncoder()

y\_train\_encoded = label\_encoder.fit\_transform(y\_train\_indices) y\_test\_encoded = label\_encoder.transform(y\_test\_indices)

mlp\_model = MLPClassifier(hidden\_layer\_sizes=(100,), max\_iter=20, learning\_rate\_init=0.01)

train\_accuracies = [] val\_accuracies = []

for epoch in range(20): mlp\_model.partial\_fit(x\_train, y\_train\_encoded,

classes=np.unique(y\_train\_encoded)) y\_train\_pred = mlp\_model.predict(x\_train)

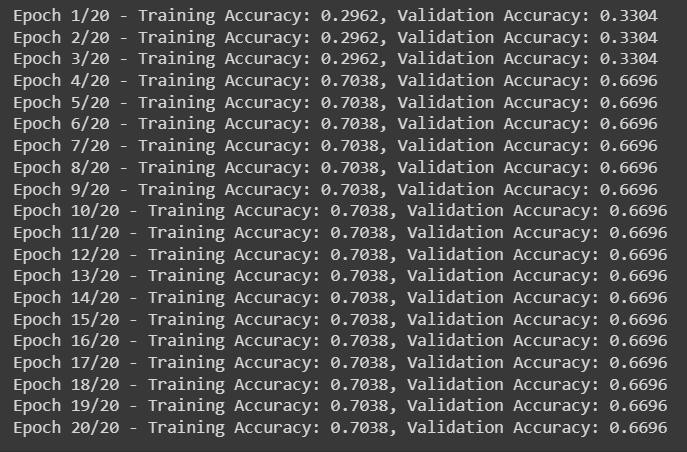
train\_accuracy = accuracy\_score(y\_train\_encoded, y\_train\_pred) train\_accuracies.append(train\_accuracy)

y\_val\_pred = mlp\_model.predict(x\_test)

val\_accuracy = accuracy\_score(y\_test\_encoded, y\_val\_pred) val\_accuracies.append(val\_accuracy)

print(f"Epoch {epoch+1}/20 - Training Accuracy:

{train\_accuracy:.4f}, Validation Accuracy: {val\_accuracy:.4f}") y\_test\_pred = mlp\_model.predict(x\_test)



test\_accuracy = accuracy\_score(y\_test\_encoded, y\_test\_pred)

print(f"Testing Accuracy fo Multiple Layer perceptron model :

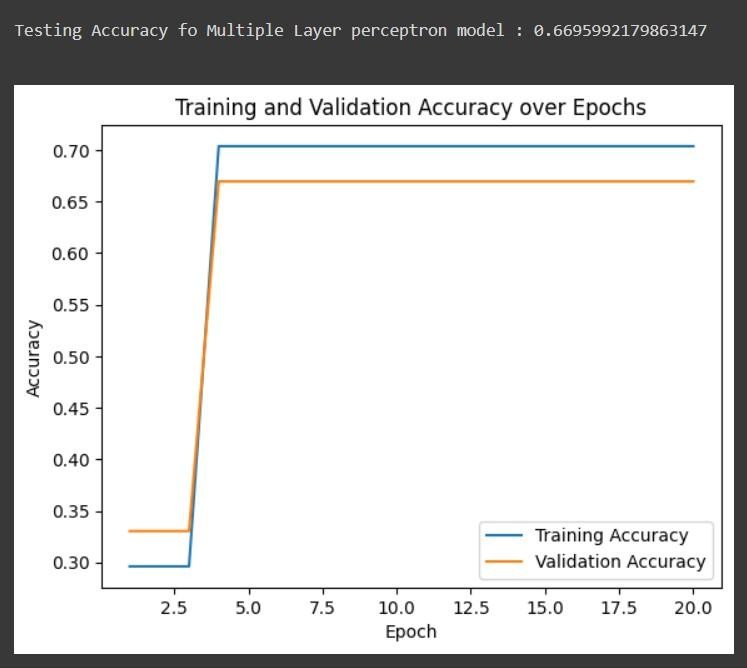
{test\_accuracy}")

plt.plot(range(1, 21), train\_accuracies, label='Training Accuracy') plt.plot(range(1, 21), val\_accuracies, label='Validation Accuracy') plt.xlabel('Epoch')

plt.ylabel('Accuracy') plt.legend(loc='lower right')

plt.title('Training and Validation Accuracy over Epochs')

plt.show()



# Inferences

* All the Images in the dataset are **normalized**
* All the Images in the dataset are **resized** for th**e image Enhancement**
* The same Input Images are divided for **training and the testing** with the **ratio** of

**70 : 30** for the Three Models

* The Accuracy For the Three model are shown below

|  |  |
| --- | --- |
| 1) **Convolutional neural network** | **80.15640377998352** |
| **2) Single Layer perceptron model** | **72.14076246334311** |
| **3) Multiple Layer perceptron model** | **66.95992179863147** |

**Github Link:** https://github.com/Anandh-007/handwritten-charachter-recognition

# Learning Outcome

* The Size of the Image dataset will affect the ML model
* While reading the Images we need to consider the image size because it will affect the quality of the Images. It will affect the Accuracy of the Model also
* The Maximum number of Epochs for the Neural network Model plays the major role in the accuracy
* We need to consider the Learning rate for the models.