**Arya Teli**

**2213138**

**AIA-1-A**

**Assignment No. 2**

**Aim:** Develop multi class classifier using deep multilayer perceptron (Keras/tensorflow/pytorch) for MNIST hand recognition dataset and CIFAR10. Fine the parameters for better accuracy.

Develop application with GUI to upload input to the system.

Test the model

**Objectives:**

1. Learn Deep Neural Network modeling
2. Learn to develop and deploy models

**Theory:**

**Standardisation**

This is one of the most use type of scalar in data preprocessing . This is known as z-score . This re distribute the data in such a way that mean (μ) = 0 and standard deviation (σ) =1 .

Here is the below formula for calculation



**Normalization:**

Normalization scales the feature between 0.0 & 1.0, retaining their proportional range to each other.



The range of normal distribution is [-1,1] with mean =0.

**Data Splitting**

Train Test Split is one of the important steps in Machine Learning. It is very important because your model needs to be evaluated before it has been deployed. And that evaluation needs to be done on unseen data because when it is deployed, all incoming data is unseen.

The main idea behind the train test split is to convert original data set into 2 parts

* train
* test where train consists of training data and training labels and test consists of testing data and testing labels.

The easiest way to do it is by using *scikit-learn*, which has a built-in function *train\_test\_split*

**Data Cleaning**

Data cleaning is the process of preparing data for analysis by removing or modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted.

This data is usually not necessary or helpful when it comes to analyzing data because it may hinder the process or provide inaccurate results. There are several methods for cleaning data depending on how it is stored along with the answers being sought.

Data cleaning is not simply about erasing information to make space for new data, but rather finding a way to maximize a data set’s accuracy without necessarily deleting information.

For one, data cleaning includes more actions than removing data, such as fixing spelling and syntax errors, standardizing data sets, and correcting mistakes such as empty fields, missing codes, and identifying duplicate data points. Data cleaning is considered a foundational element of the [data science basics,](https://www.sisense.com/glossary/data-science-basics/) as it plays an important role in the analytical process and uncovering reliable answers.

MNIST CODE:

import keras

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense, Dropout, Flatten

from keras.layers import Conv2D, MaxPooling2D

from keras import backend as K

# the data, split between train and test sets

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

print(x\_train.shape, y\_train.shape)

import matplotlib.pyplot as plt

plt.imshow(x\_train[20])

x\_train = x\_train.reshape(x\_train.shape[0], 28, 28, 1)

x\_test = x\_test.reshape(x\_test.shape[0], 28, 28, 1)

input\_shape = (28, 28, 1)

# convert class vectors to binary class matrices

num\_classes = 10 # There are 10 classes (digits 0-9)

y\_train = keras.utils.to\_categorical(y\_train, num\_classes)

y\_test = keras.utils.to\_categorical(y\_test, num\_classes)

x\_train = x\_train.astype('float32')

x\_test = x\_test.astype('float32')

x\_train /= 255

x\_test /= 255

print('x\_train shape:', x\_train.shape)

print(x\_train.shape[0], 'train samples')

print(x\_test.shape[0], 'test samples')

batch\_size = 128

num\_classes = 10

epochs = 25

model = Sequential()

model.add(Conv2D(32, kernel\_size=(3, 3),activation='relu',input\_shape=input\_shape))

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(256, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(num\_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical\_crossentropy,optimizer=keras.optimizers.Adadelta(),metrics=['accuracy'])

hist = model.fit(x\_train, y\_train,batch\_size=batch\_size,epochs=epochs,verbose=1,validation\_data=(x\_test, y\_test))

print("The model has successfully trained")

model.save('mnist.h5')

print("Saving the model as mnist.h5")

score = model.evaluate(x\_test, y\_test, verbose=0)

print('Test loss:', score[0])

print('Test accuracy:', score[1])

from keras.models import load\_model

from tkinter import \*

import tkinter as tk

import win32gui

from PIL import ImageGrab, Image

import numpy as np

model = load\_model('mnist.h5')

def predict\_digit(img):

#resize image to 28x28 pixels

img = img.resize((28,28))

#convert rgb to grayscale

img = img.convert('L')

img = np.array(img)

#reshaping to support our model input and normalizing

img = img.reshape(1,28,28,1)

img = img/255.0

#predicting the class

res = model.predict([img])[0]

return np.argmax(res), max(res)

class App(tk.Tk):

def \_\_init\_\_(self):

tk.Tk.\_\_init\_\_(self)

self.x = self.y = 0

# Creating elements

self.canvas = tk.Canvas(self, width=300, height=300, bg = "white", cursor="cross")

self.label = tk.Label(self, text="Thinking..", font=("Helvetica", 48))

self.classify\_btn = tk.Button(self, text = "Recognise", command =self.classify\_handwriting)

self.button\_clear = tk.Button(self, text = "Clear", command = self.clear\_all)

# Grid structure

self.canvas.grid(row=0, column=0, pady=2, sticky=W, )

self.label.grid(row=0, column=1,pady=2, padx=2)

self.classify\_btn.grid(row=1, column=1, pady=2, padx=2)

self.button\_clear.grid(row=1, column=0, pady=2)

#self.canvas.bind("<Motion>", self.start\_pos)

self.canvas.bind("<B1-Motion>", self.draw\_lines)

def clear\_all(self):

self.canvas.delete("all")

def classify\_handwriting(self):

HWND = self.canvas.winfo\_id() # get the handle of the canvas

rect = win32gui.GetWindowRect(HWND) # get the coordinate of the canvas

im = ImageGrab.grab(rect)

digit, acc = predict\_digit(im)

self.label.configure(text= str(digit)+', '+ str(int(acc\*100))+'%')

def draw\_lines(self, event):

self.x = event.x

self.y = event.y

r=8

self.canvas.create\_oval(self.x-r, self.y-r, self.x + r, self.y + r, fill='black')

app = App()

mainloop()

OUTPUT :

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

CIFAR-10 CODE:

import tensorflow as tf

from tensorflow.keras import datasets, layers, models

import matplotlib.pyplot as plt

import numpy as np

(Xtrain, ytrain), (Xtest, ytest) = datasets.cifar10.load\_data()

Xtrain.shape

Xtest.shape

#ytrain is a 2d array, but we need 1D array only for classification

ytrain.shape

ytrain[:5]

ytrain = ytrain.reshape(-1,)

ytrain[:5]

classes = ["airplane","automobile","bird","cat","deer","dog","frog","horse","ship","truck"]

classes[2]

plt.imshow(Xtrain[2])

plt.xlabel(classes[ytrain[2]])

#create a function to plot the image

def plot\_sample(X, y, index):

plt.imshow(X[index])

plt.xlabel(classes[y[index]])

plot\_sample(Xtrain, ytrain, 45)

<h4>normalize the data</h4>

Xtrain = Xtrain/255

Xtest = Xtest/255

cnn = models.Sequential([

#cnn

layers.Conv2D(filters = 32, kernel\_size = (3,3), activation="relu", input\_shape=(32,32,3)),

layers.MaxPooling2D((2,2)),

layers.Conv2D(filters = 64, kernel\_size = (3,3), activation="relu"),

layers.MaxPooling2D((2,2)),

#dense NN

layers.Flatten(),

layers.Dense(64, activation='relu'),

layers.Dense(10, activation='softmax')

])

cnn.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

cnn.fit(Xtrain, ytrain, epochs=15)

cnn.evaluate(Xtest, ytest)

ytest = ytest.reshape(-1,)

plot\_sample(Xtest, ytest, 2)

ytest[:10]

ypred = cnn.predict(Xtest)

y\_pred\_classes = [np.argmax(element) for element in ypred]

y\_pred\_classes[:10]

plot\_sample(Xtest, ytest, 3)

classes[y\_pred\_classes[3]]

cnn.save('cnn\_cifar10.h5')

GUI CIFAR –

import tkinter as tk

from tkinter import filedialog, Label, Button

from PIL import Image, ImageTk

import numpy as np

import tensorflow as tf

# Load your model

model = tf.keras.models.load\_model('cnn\_cifar10.h5')

classes = ["airplane","automobile","bird","cat","deer","dog","frog","horse","ship","truck"]

def preprocess\_image(image\_path):

    img = Image.open(image\_path)

    img = img.resize((32, 32))

    img = np.array(img)

    img = img / 255.0

    img = np.expand\_dims(img, axis=0)

    return img

def predict\_image(image\_path):

    img = preprocess\_image(image\_path)

    prediction = model.predict(img)

    class\_index = np.argmax(prediction)

    return class\_index

def load\_image():

    file\_path = filedialog.askopenfilename()

    if file\_path:

        img = Image.open(file\_path)

        img = img.resize((128, 128))

        img = ImageTk.PhotoImage(img)

        image\_label.config(image=img)

        image\_label.image = img

        class\_index = predict\_image(file\_path)

        class\_name = classes[class\_index]

        result\_label.config(text=f"Predicted Class: {class\_name}")

# Create the main application window

root = tk.Tk()

root.title("CIFAR-10 Image Classifier")

# Create widgets

load\_button = Button(root, text="Load Image", command=load\_image)

load\_button.pack(pady=10)

image\_label = Label(root)

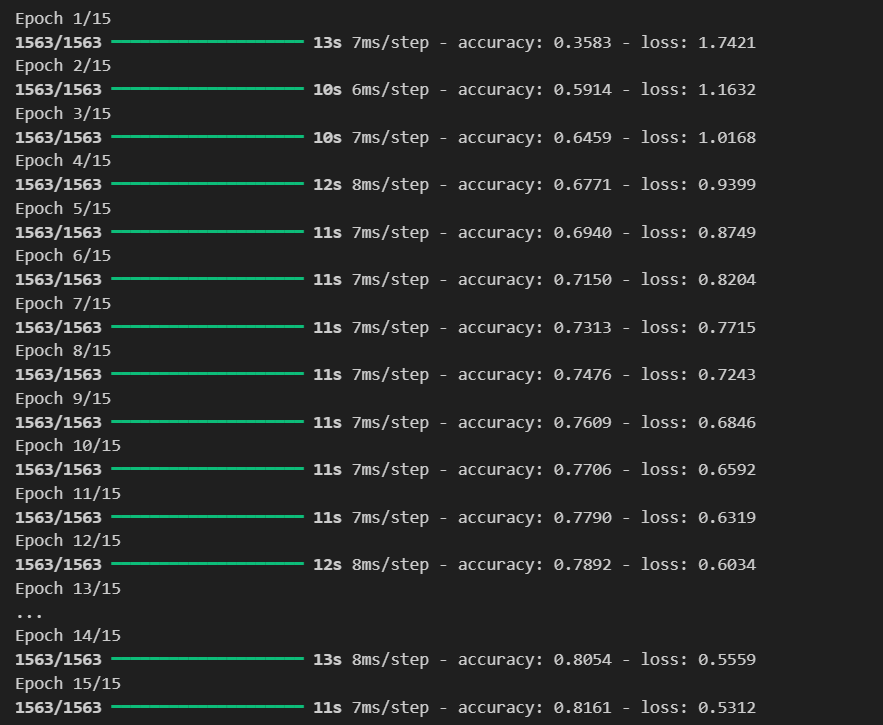
image\_label.pack()

result\_label = Label(root, text="")

result\_label.pack(pady=10)

# Start the main event loop

root.mainloop()



A screenshot of a computer

Description automatically generated

**Conclusion:**

Thus, we have understood the syntax and basic model creation in TensorFlow for 2 different task.

We have also learned how to create a GUI using services to do so.