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

$$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation }(x)}$$

Normalization:

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

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

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, as it plays an important role in the analytical process and uncovering reliable answers.

Code:

For MNIST

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras import models,datasets,layers
import matplotlib.pyplot as plt
import matplotlib.image as mp
(train_images,train_labels), (test_images,test_labels)=datasets.mnist.load_dat
a()
train_images=train_images/255
test_images= test_images/255
```

```
model=models.Sequential()
model.add(layers.Flatten(input_shape=(28, 28, 1)))
model.add(layers.Dense(32,activation="relu"))
model.add(layers.Dense(16,activation="relu"))
model.add(layers.Dense(10,activation="softmax"))

model.compile(optimizer='adam',loss="sparse_categorical_crossentropy",metrics=
['accuracy'])

model.fit(train_images, train_labels, epochs=10,validation_data=(test_images,t
est_labels))
```

For CIFAR

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras import models,layers,datasets

(train_images, train_labels),(test_images,test_labels)=datasets.cifar10.load_d
ata()
train_images=train_images/255
test_images=test_images/255

model=models.Sequential()
model.add(layers.Flatten(input_shape=(32, 32, 3)))
model.add(layers.Dense(512,activation="relu"))
#model.add(layers.Dense(256,activation="relu"))
model.add(layers.Dense(128,activation="relu"))
#model.add(layers.Dense(64,activation="relu"))
model.add(layers.Dense(32,activation="relu"))
#model.add(layers.Dense(16,activation="relu"))
model.add(layers.Dense(10,activation="softmax"))

model.compile(optimizer="adam",loss="sparse_categorical_crossentropy",metrics=
["accuracy"])

model.fit(train_images,train_labels,epochs=10,validation_data=(test_images,tes
t_labels))
```

MNIST GUI

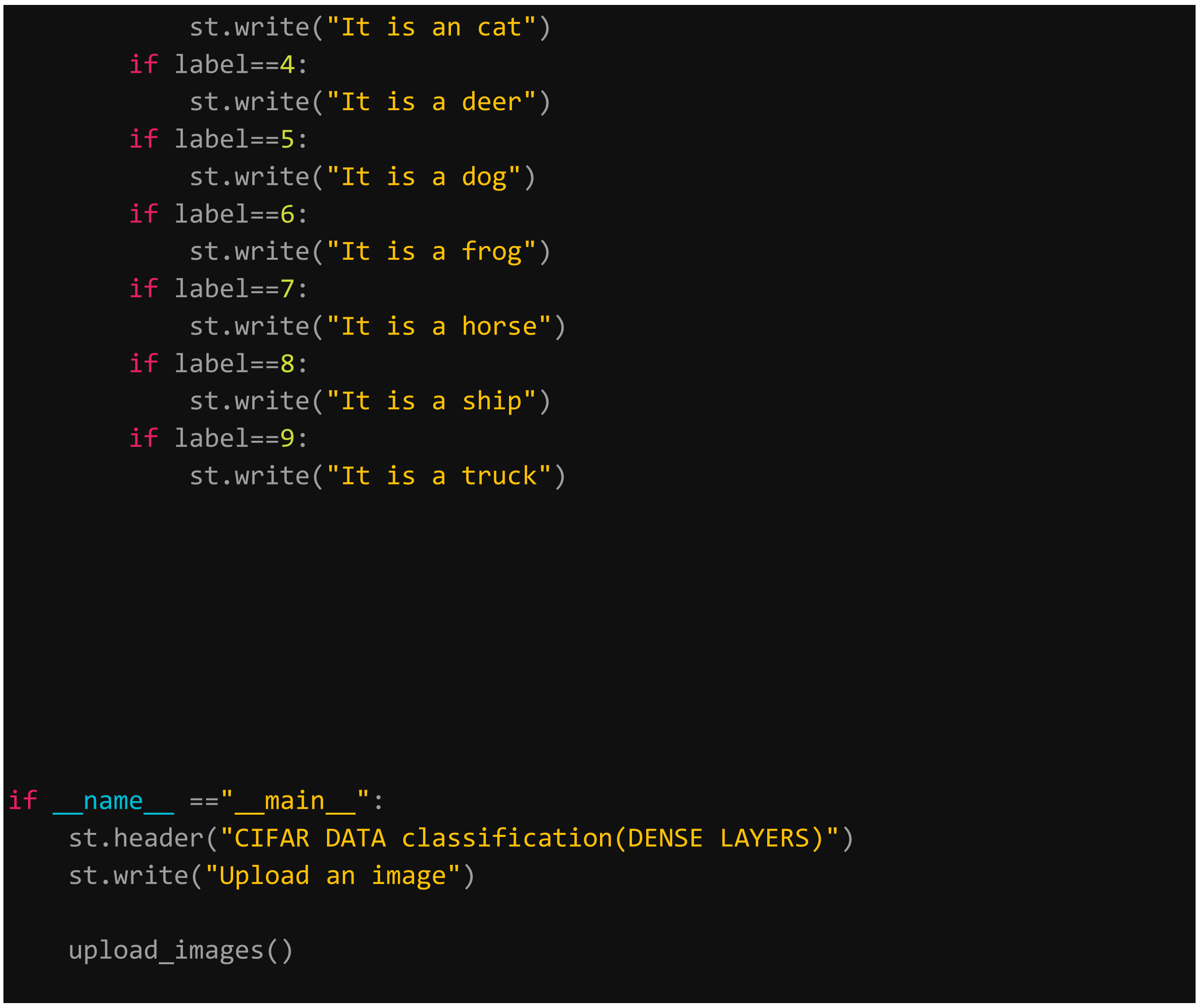
```
from PIL import Image,ImageOps
import os
import streamlit as st
```

```python
import tensorflow as tf
from tensorflow.keras import models
from tensorflow.keras.preprocessing import image
import numpy as np
st.set_option('deprecation.showfileUploaderEncoding', False)
def load_models(img):
    model = models.load_model('/tmp/model')
    image=img.resize((28,28))
    image_array=np.array(image)
    image_array=tf.image.rgb_to_grayscale(image_array)
    image_array=(tf.reshape(image_array,(image_array.shape[0],image_array.shape[0],1)))/255
    image_array=np.array([image_array])

    prediction = model.predict_classes(image_array)
    return prediction



def upload_images():
    uploaded_file = st.file_uploader("Choose an Image ...", type="jpg")
    if uploaded_file is not None:
        image = Image.open(uploaded_file)
        st.image(image, caption='Uploaded The image.', use_column_width=True)
        st.write("")
        st.write("Classifying...")
        label = load_models(image)

        if label==0:
            st.write("0")
        if label==1:
            st.write("1")
        if label==2:
            st.write("2")
        if label==3:
            st.write("3")
        if label==4:
            st.write("4")
        if label==5:
            st.write("5")
        if label==6:
            st.write("6")
        if label==7:
            st.write("7")
        if label==8:
            st.write("8")
        if label==9:
            st.write("9")
```

```python
if __name__ =="__main__":
    st.header("MNIST DATA classification")
    st.write("Upload an image")

    upload_images()
```

CIFAR GUI

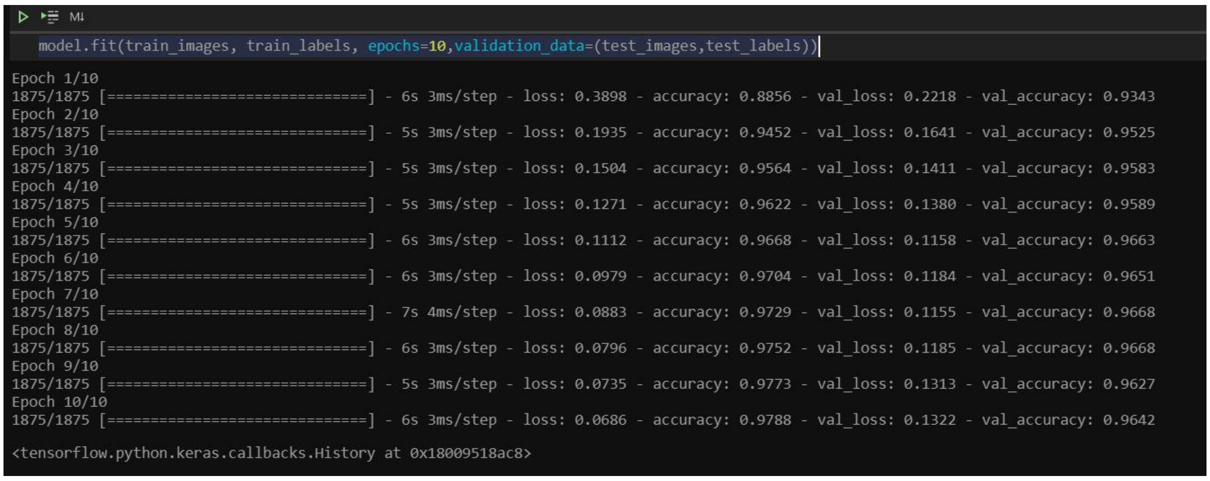
```python
from PIL import Image,ImageOps
import os
import streamlit as st
import tensorflow as tf
from tensorflow.keras import models
from tensorflow.keras.preprocessing import image
import numpy as np
st.set_option('deprecation.showfileUploaderEncoding', False)
def load_models(img):
    model = models.load_model('/tmp/model1')
    image=img.resize((32,32))
    image_array=np.array(image)
    #image_array=tf.image.rgb_to_grayscale(image_array)
    image_array=(tf.reshape(image_array,(image_array.shape[0],image_array.shap
e[0],3)))/255
    image_array=np.array([image_array])

    prediction = model.predict_classes(image_array)
    return prediction



def upload_images():
    uploaded_file = st.file_uploader("Choose an Image ...", type="jpg")
    if uploaded_file is not None:
        image = Image.open(uploaded_file)
        st.image(image, caption='Uploaded The image.', use_column_width=True)
        st.write("")
        st.write("Classifying...")
        label = load_models(image)

        if label==0:
            st.write("It is an aeroplane")
        if label==1:
            st.write("It is an automobile")
        if label==2:
            st.write("It is a bird")
        if label==3:
```



```
            st.write("It is an cat")
        if label==4:
            st.write("It is a deer")
        if label==5:
            st.write("It is a dog")
        if label==6:
            st.write("It is a frog")
        if label==7:
            st.write("It is a horse")
        if label==8:
            st.write("It is a ship")
        if label==9:
            st.write("It is a truck")





if __name__ =="__main__":
    st.header("CIFAR DATA classification(DENSE LAYERS)")
    st.write("Upload an image")

    upload_images()
```

Results:

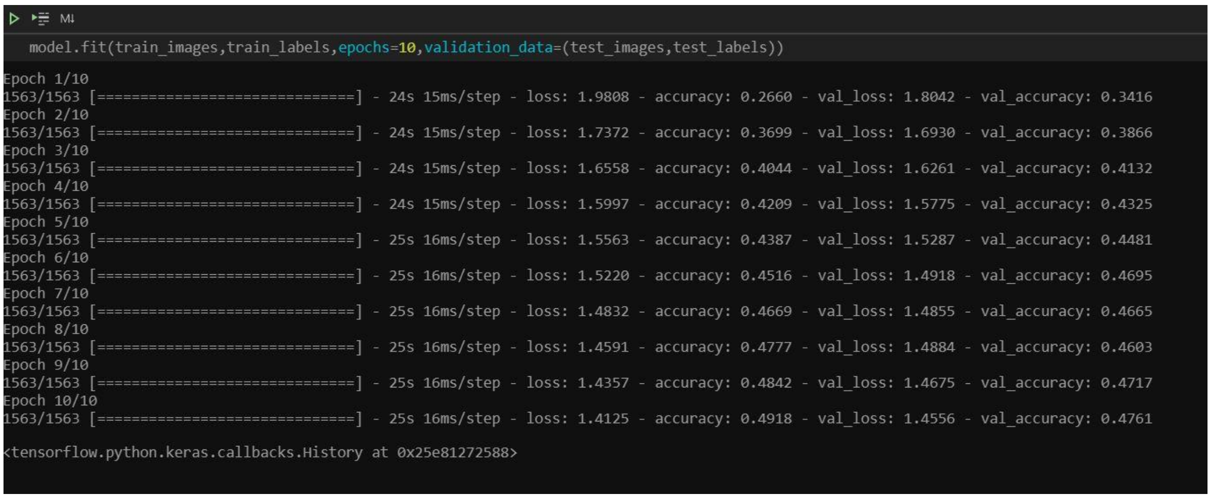
MNIST Training:



```
model.fit(train_images, train_labels, epochs=10,validation_data=(test_images,test_labels))
```

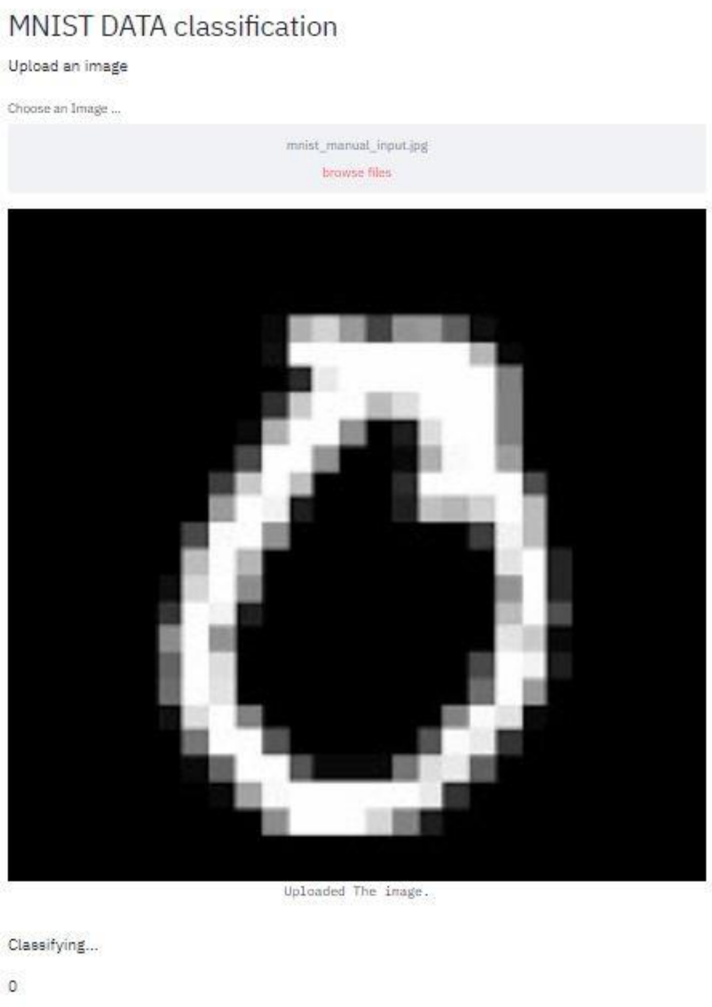
```
Epoch 1/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3898 - accuracy: 0.8856 - val_loss: 0.2218 - val_accuracy: 0.9343
Epoch 2/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1935 - accuracy: 0.9452 - val_loss: 0.1641 - val_accuracy: 0.9525
Epoch 3/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1504 - accuracy: 0.9564 - val_loss: 0.1411 - val_accuracy: 0.9583
Epoch 4/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.1271 - accuracy: 0.9622 - val_loss: 0.1380 - val_accuracy: 0.9589
Epoch 5/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1112 - accuracy: 0.9668 - val_loss: 0.1158 - val_accuracy: 0.9663
Epoch 6/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0979 - accuracy: 0.9704 - val_loss: 0.1184 - val_accuracy: 0.9651
Epoch 7/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0883 - accuracy: 0.9729 - val_loss: 0.1155 - val_accuracy: 0.9668
Epoch 8/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0796 - accuracy: 0.9752 - val_loss: 0.1185 - val_accuracy: 0.9668
Epoch 9/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0735 - accuracy: 0.9773 - val_loss: 0.1313 - val_accuracy: 0.9627
Epoch 10/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0686 - accuracy: 0.9788 - val_loss: 0.1322 - val_accuracy: 0.9642

<tensorflow.python.keras.callbacks.History at 0x18009518ac8>
```

CIFAR10 Training



```
model.fit(train_images,train_labels,epochs=10,validation_data=(test_images,test_labels))

Epoch 1/10
1563/1563 [==============================] - 24s 15ms/step - loss: 1.9808 - accuracy: 0.2660 - val_loss: 1.8042 - val_accuracy: 0.3416
Epoch 2/10
1563/1563 [==============================] - 24s 15ms/step - loss: 1.7372 - accuracy: 0.3699 - val_loss: 1.6930 - val_accuracy: 0.3866
Epoch 3/10
1563/1563 [==============================] - 24s 15ms/step - loss: 1.6558 - accuracy: 0.4044 - val_loss: 1.6261 - val_accuracy: 0.4132
Epoch 4/10
1563/1563 [==============================] - 24s 15ms/step - loss: 1.5997 - accuracy: 0.4209 - val_loss: 1.5775 - val_accuracy: 0.4325
Epoch 5/10
1563/1563 [==============================] - 25s 16ms/step - loss: 1.5563 - accuracy: 0.4387 - val_loss: 1.5287 - val_accuracy: 0.4481
Epoch 6/10
1563/1563 [==============================] - 25s 16ms/step - loss: 1.5220 - accuracy: 0.4516 - val_loss: 1.4918 - val_accuracy: 0.4695
Epoch 7/10
1563/1563 [==============================] - 25s 16ms/step - loss: 1.4832 - accuracy: 0.4669 - val_loss: 1.4855 - val_accuracy: 0.4665
Epoch 8/10
1563/1563 [==============================] - 25s 16ms/step - loss: 1.4591 - accuracy: 0.4777 - val_loss: 1.4884 - val_accuracy: 0.4603
Epoch 9/10
1563/1563 [==============================] - 25s 16ms/step - loss: 1.4357 - accuracy: 0.4842 - val_loss: 1.4675 - val_accuracy: 0.4717
Epoch 10/10
1563/1563 [==============================] - 25s 16ms/step - loss: 1.4125 - accuracy: 0.4918 - val_loss: 1.4556 - val_accuracy: 0.4761

<tensorflow.python.keras.callbacks.History at 0x25e81272588>
```

MNIST GUI



MNIST DATA classification

Upload an image

Choose an Image ...

mnist_manual_input.jpg

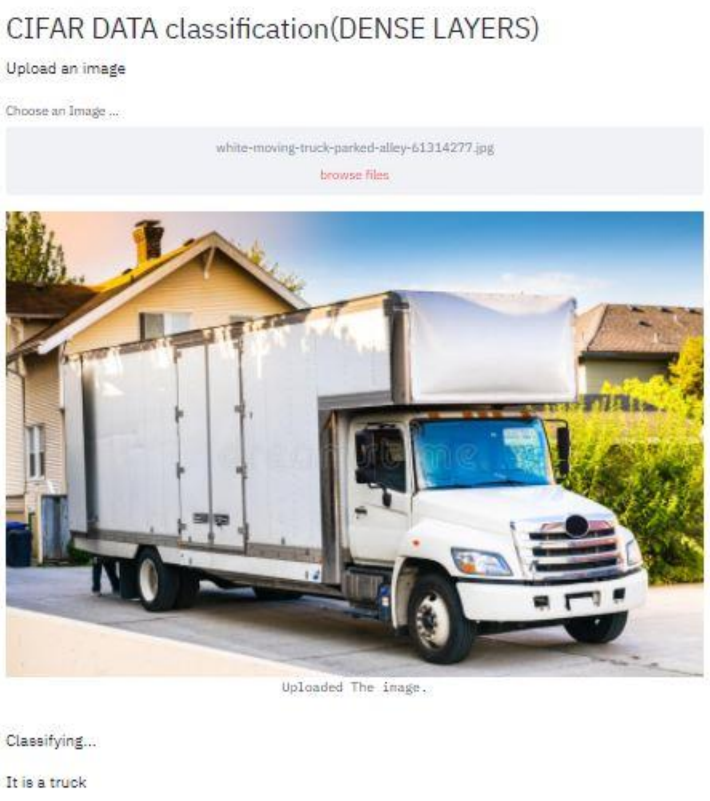
browse files

Uploaded The image.

Classifying...

0

CIFAR10 GUI


CIFAR DATA classification(DENSE LAYERS)

Upload an image

Choose an Image ...

white-moving-truck-parked-alley-61314277.jpg

browse files

Uploaded The image.

Classifying...

It is a truck

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.