Assignment_No.3

Problem Statement:

Build the Image classification model by dividing the model into the following four stages: a. Loading and preprocessing the image data

b. Defining the model's architecture

c. Training the model

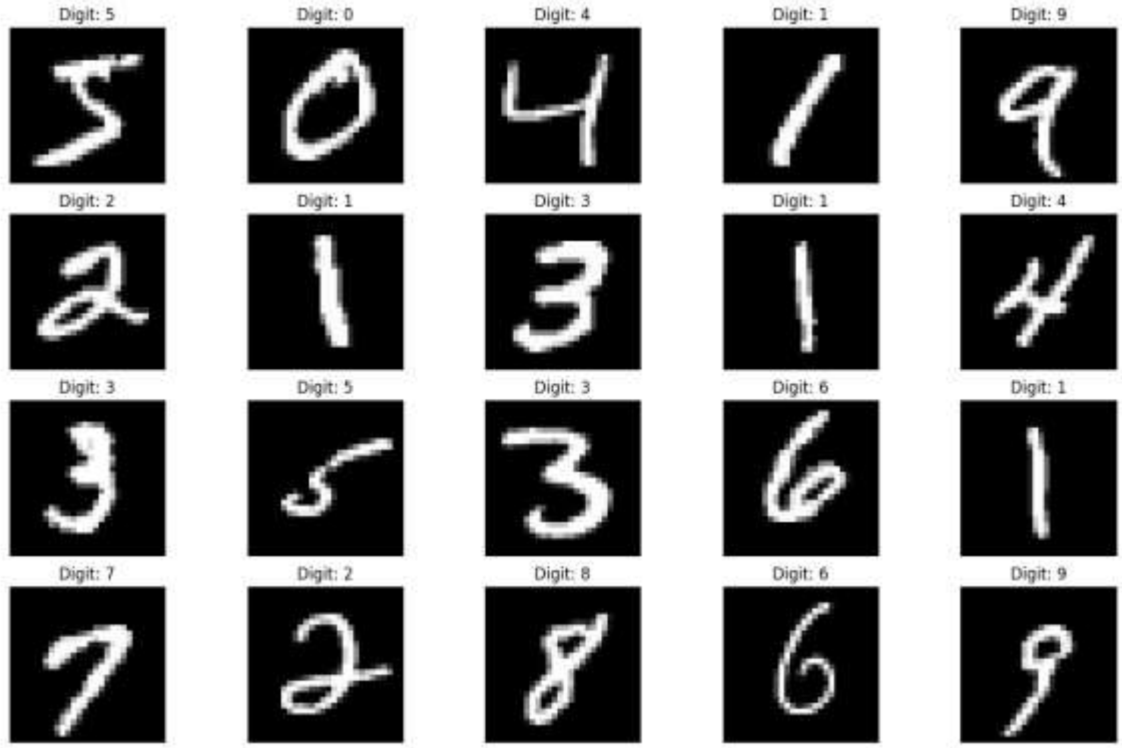
d. Estimating the model's performance

[1]:
```
import numpy as np
import pandas as pd
import random
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Conv2D, Dense, MaxPooling2D
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.datasets import mnist
```

[2]:
```
(X_train, y_train), (X_test, y_test ) = mnist.load_data()
print(X_train.shape)
X_train[0].min(), X_train[0].max()
X_train = (X_train - 0.0) / (255.0 - 0.0)
X_test = (X_test - 0.0) / (255.0 - 0.0)
X_train[0].min(), X_train[0].max()
def plot_digit(image, digit, plt, i ):
    plt.subplot(4, 5,    + 1)
    plt.imshow(image, cmap=plt.get_cmap('gray'))
    plt.title(f"Digit: {digit}")
    plt.xticks([])
    plt.yticks([])
plt.figure(figsize=(16, 10))
for i in range(20):
    plot_digit(X_train[i], y_train[i], plt, i)
plt.show()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-
kerasdatasets/mnist.npz
11490434/11490434  0s
0us/step
(60000, 28, 28)
```



[3]:

```
X_train = X_train.reshape((X_train.shape + (1,)))
X_test = X_test.reshape((X_test.shape + (1,)))
y_train[0:20]
```

[3]:

```
array([5, 0, 4, 1, 9, 2, 1, 3, 1, 4, 3, 5, 3, 6, 1, 7, 2, 8, 6, 9],
      dtype=uint8)
```

[4]:

```
model = Sequential([
    Conv2D(32,  3, 3), activation="relu", input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(100, activation="relu"),
    Dense(10, activation="softmax")
])

optimizer = SGD(learning_rate=0.01, momentum=0.9)

model.compile(
    optimizer=optimizer,
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)
model.summary()
model.fit(X_train, y_train, epochs=10, batch_size=32)
```

```
/usr/local/lib/python3.10/distpackages/keras/src/layers/convolutio
nal/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

```
Model: "sequential"


 Layer (type)                     Output Shape                     ␣
 ↪Param #

 conv2d (Conv2D)                  (None, 26, 26, 32)                 ␣
 ↪320

 max_pooling2d (MaxPooling2D)     (None, 13, 13, 32)                 ␣
 ↪0

 flatten (Flatten)                (None, 5408)                       ␣
 ↪0

 dense (Dense)                    (None, 100)                      ␣
 ↪540,900

 dense_1 (Dense)                  (None, 10)                        ␣
 ↪1,010


 Total params: 542,230 (2.07 MB)

 Trainable params: 542,230 (2.07 MB)

 Non-trainable params: 0 (0.00 B)

Epoch 1/10
```

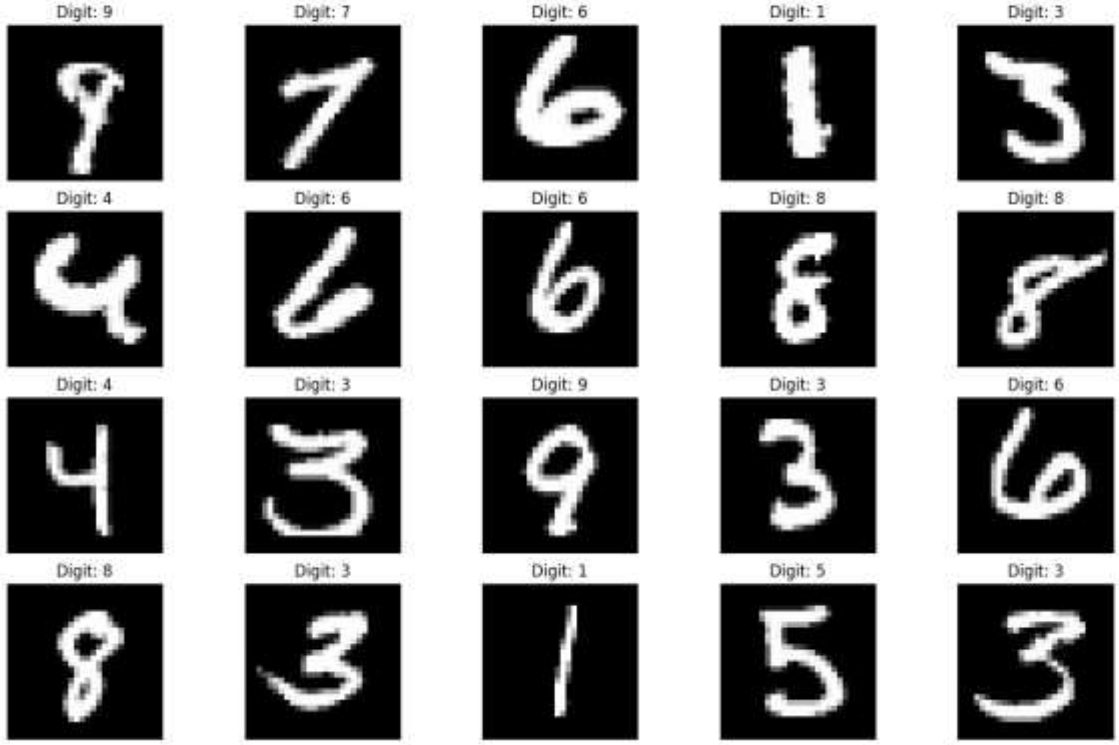
```
1875/1875  36s 19ms/step accuracy:
0.8588 - loss: 0.4448
Epoch 2/10
1875/1875  34s 18ms/step accuracy:
0.9733 - loss: 0.0862
Epoch 3/10
1875/1875  35s 19ms/step accuracy:
0.9843 - loss: 0.0520
Epoch 4/10
1875/1875  33s 18ms/step accuracy:
0.9896 - loss: 0.0346
Epoch 5/10
1875/1875  35s 18ms/step accuracy:
0.9914 - loss: 0.0252
Epoch 6/10
1875/1875  33s 18ms/step accuracy:
0.9946 - loss: 0.0186
Epoch 7/10
1875/1875  33s 17ms/step accuracy:
0.9958 - loss: 0.0153
Epoch 8/10
1875/1875  42s 18ms/step accuracy:
0.9978 - loss: 0.0088
Epoch 9/10
1875/1875  34s 18ms/step accuracy:
0.9983 - loss: 0.0079
Epoch 10/10
1875/1875  33s 18ms/step accuracy:
0.9988 - loss: 0.0056
```

```
[4]: <keras.src.callbacks.history.History at 0x7db4d9820100>
```

```
[5]: plt.figure(figsize=(16, 10))
     for i in range(20):
         image = random.choice(X_test).squeeze()
         digit = np.argmax(model.predict(image.reshape((1, 28, 28, 1)))[0], axis=-1)
         plot_digit(image, digit, plt, i)
     plt.show()
```

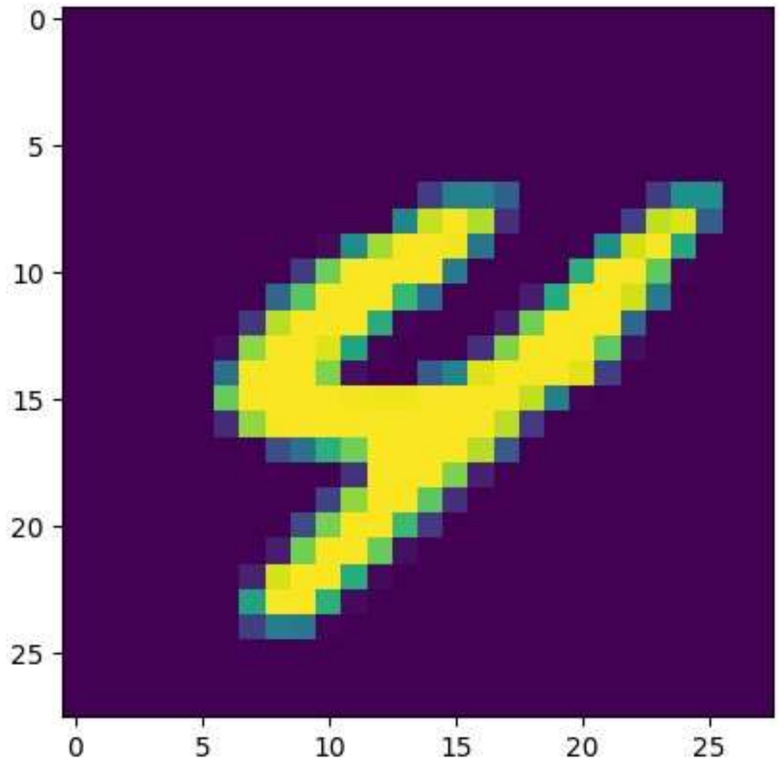
```
1/1           0s 166ms/step
1/1  0s 40ms/step
1/1  0s 63ms/step
1/1  0s 103ms/step
1/1  0s 24ms/step
1/1  0s 23ms/step
1/1  0s 35ms/step
1/1  0s 24ms/step
```

```
1/1  0s 24ms/step
1/1  0s 24ms/step
1/1  0s 26ms/step
1/1  0s 25ms/step
1/1  0s 24ms/step
1/1  0s 28ms/step
1/1  0s 27ms/step
1/1  0s 34ms/step
1/1  0s 24ms/step
1/1  0s 22ms/step
1/1  0s 36ms/step
1/1  0s 35ms/step
```



```
[6]: predictions = np.argmax(model.predict(X_test),
axis=-1) accuracy_score(y_test, predictions)
n=random.randint(0,9999) plt.imshow(X_test[n])
plt.show()
predicted_value=model.predict(X_test) print("Handwritten number in
the image is= %d" %np.argmax(predicted_value[n])) score =
model.evaluate(X_test, y_test, verbose=0) print('Test loss:',
score[0]) #Test loss: 0.0296396646054 print('Test accuracy:',
score[1])
```

```
313/313  4s 11ms/step
```



```
313/313  4s 14ms/step
Handwritten number in the image is=
                                 4
Test loss:
0.043948426842689514 Test
accuracy: 0.9866999983787537
```

[ ]: