SPRINT 4

Date	16 November 2022
Team ID	PNT2022TMID03743

In [19]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from keras.utils import np_utils
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Dense, Flatten
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import load_model
from PIL import Image, ImageOps
import numpy
```

In [20]:
```python
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

In [21]:
```python
print(X_train.shape)
print(X_test.shape)
```

```
(60000, 28, 28)
(10000, 28, 28)
```

In [22]:
```python
X_train[0]
```

Out[22]:
```
array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,
         18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170,
        253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253, 253,
        253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  18, 219, 253, 253, 253, 253,
        253, 198, 182, 247, 241,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253,
        205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,
         90,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 139, 253,
        190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11, 190,
        253,  70,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  35,
        241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         81, 240, 253, 253, 119,  25,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,  45, 186, 253, 253, 150,  27,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,  16,  93, 252, 253, 187,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0, 249, 253, 249,  64,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,  46, 130, 183, 253, 253, 207,   2,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  39,
        148, 229, 253, 253, 253, 250, 182,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  24, 114, 221,
        253, 253, 253, 253, 201,  78,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
```

```
 [  0,   0,   0,   0,   0,   0,   0,   0,  23,  66, 213, 253, 253,
  253, 253, 198,  81,   2,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0],
 [  0,   0,   0,   0,   0,   0,  18, 171, 219, 253, 253, 253, 253,
  195,  80,   9,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0],
 [  0,   0,   0,   0,  55, 172, 226, 253, 253, 253, 253, 244, 133,
   11,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0],
 [  0,   0,   0,   0, 136, 253, 253, 253, 212, 135, 132,  16,   0,
    0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0]], dtype=uint8)
```

In [23]:

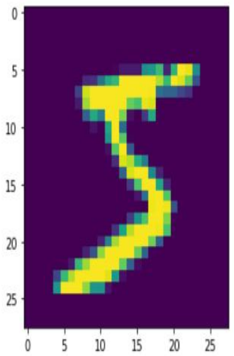
```
y_train[0]
```

Out[23]: 5

In [24]:

```
plt.imshow(X_train[0])
```

Out[24]:



In [25]:

```
X_train = X_train.reshape(60000, 28, 28, 1).astype('float32')
X_test = X_test.reshape(10000, 28, 28, 1).astype('float32')
```

In [26]:

```
number_of_classes = 10
Y_train = np_utils.to_categorical(y_train, number_of_classes)
Y_test = np_utils.to_categorical(y_test, number_of_classes)
```

In [27]:

```
Y_train[0]
```

Out[27]: array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.], dtype=float32)

In [28]:

```
model = Sequential()
model.add(Conv2D(64, (3, 3), input_shape=(28, 28, 1), activation="relu"))
model.add(Conv2D(32, (3, 3), activation="relu"))
model.add(Flatten())
model.add(Dense(number_of_classes, activation="softmax"))
```

In [29]:

```
model.compile(loss='categorical_crossentropy', optimizer="Adam", metrics=["accuracy"])
```

In [30]:
```
model.fit(X_train, Y_train, batch_size=32, epochs=5, validation_data=(X_test,Y_test))
```

```
Epoch 1/5
1875/1875 [==============================] - 218s 116ms/step - loss: 0.2507 - accuracy: 0.9514 - val_loss: 0.0951 - val_accuracy: 0.9688
Epoch 2/5
1875/1875 [==============================] - 204s 109ms/step - loss: 0.0722 - accuracy: 0.9777 - val_loss: 0.1012 - val_accuracy: 0.9712
Epoch 3/5
1875/1875 [==============================] - 205s 109ms/step - loss: 0.0484 - accuracy: 0.9848 - val_loss: 0.0888 - val_accuracy: 0.9781
Epoch 4/5
1875/1875 [==============================] - 203s 108ms/step - loss: 0.0364 - accuracy: 0.9887 - val_loss: 0.1191 - val_accuracy: 0.9700
Epoch 5/5
1875/1875 [==============================] - 207s 111ms/step - loss: 0.0321 - accuracy: 0.9903 - val_loss: 0.0968 - val_accuracy: 0.9778
```

Out[30]:

```
Epoch 1/5
1875/1875 [==============================] - 205s 109ms/step - loss: 0.0252 - accuracy: 0.9925 - val_loss: 0.1100 - val_accuracy: 0.9768
Epoch 2/5
 488/1875 [======>.....................] - ETA: 2:24 - loss: 0.0157 - accuracy: 0.9953
```

In [31]:
```
metrics = model.evaluate(X_test, Y_test, verbose=0)
print("Metrics (Test Loss & Test Accuracy): ")
print(metrics)
```

```
Metrics (Test Loss & Test Accuracy):
[0.09684503823518753, 0.9778000116348267]
```

In [32]:
```
prediction = model.predict(X_test[:4])
print(prediction)
```

```
1/1 [==============================] - 0s 77ms/step
[[4.8965359e-14 2.0322942e-19 1.8146091e-09 1.6368229e-07 9.2744400e-19
  2.7513203e-15 7.2106393e-18 9.9999988e-01 5.0361281e-16 5.0518121e-15]
 [9.3867920e-15 3.4357353e-13 1.0000000e+00 5.5536516e-13 8.2924691e-17
  6.1402964e-22 2.4520308e-09 3.0643714e-15 1.0961375e-12 1.8752804e-21]
 [6.2456024e-08 9.9998009e-01 4.0463109e-07 5.5024592e-12 9.9512099e-06
  2.2487791e-08 5.2307603e-09 7.0259448e-06 2.5401673e-06 3.7284091e-12]
 [1.0000000e+00 2.3800321e-17 1.7932804e-09 6.9831990e-17 1.4945141e-13
  1.1589770e-14 4.0941242e-09 1.5503991e-12 2.5338605e-09 1.8694037e-12]]
```

In [33]:
```
print(numpy.argmax(prediction, axis=1))
print(Y_test[:4])
```

```
[7 2 1 0]
[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

In [34]:
```
model.save("model.h5")
```

In [35]:
```
model=load_model("model.h5")
```

In [36]:
```
from keras.datasets import mnist
from matplotlib import pyplot
(X_train,y_train),(X_test,y_test)=mnist.load_data()
print('X_train:' +str(X_train.shape))
print('y_train:' +str(y_train.shape))
print('X_test:' +str(X_test.shape))
print('y_test:' +str(y_test.shape))
from matplotlib import pyplot
for i in range(9):
  pyplot.subplot(330+1+i)
  pyplot.imshow(X_train[i],cmap=pyplot.get_cmap('gray'))
  pyplot.show()
```

```
X_train:(60000, 28, 28)
y_train:(60000,)
X_test:(10000, 28, 28)
y_test:(10000,)
```