In [ ]:
```
import numpy as np
import pandas as pd
from numpy import unique, argmax
from tensorflow.keras.datasets.mnist import load_data
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dropout
from tensorflow.keras.utils import plot_model
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import mnist
```

In [ ]:
```
(train_x, train_y), (test_x, test_y) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 1s 0us/step
```

In [ ]:
```
print(train_x.shape, train_y.shape)
print(test_x.shape , test_y.shape)
```

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

In [ ]:
```
train_x = train_x.astype('float32')/255.0
test_x = test_x.astype('float32')/255.0
```

In [ ]:
```
fig = plt.figure(figsize = (20,5))
for i in range(20):
  ax= fig.add_subplot(2, 10, i+1, xticks=[], yticks=[])
  ax.imshow(np.squeeze(train_x[i]), cmap='gray')
ax.set_title(train_y[i])
```

In [ ]:
```
shape = train_x.shape[1:]
shape
```

Out[ ]:
```
(28, 28)
```

In [ ]:
```
from tensorflow.keras.layers import MaxPooling2D as MaxPool2D
model = Sequential()
#adding convolutional layer
model.add(Conv2D(32, (3,3), activation='relu', input_shape=(28, 28, 1)))
model.add(MaxPool2D((2,2)))
model.add(Conv2D(48, (3,3), activation='relu'))
model.add(MaxPool2D((2,2)))
```

In [ ]:
```
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(500, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

In [ ]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

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

 conv2d_1 (Conv2D)           (None, 11, 11, 48)        13872

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 48)         0
 2D)

 dropout (Dropout)           (None, 5, 5, 48)          0

 flatten (Flatten)           (None, 1200)              0

 dense (Dense)               (None, 500)               600500

 dense_1 (Dense)             (None, 10)                5010

=================================================================
Total params: 619,702
Trainable params: 619,702
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:
```
model.compile(optimizer='adam', loss = 'sparse_categorical_crossentropy',metrics= ['accuracy'] )
x=model.fit(train_x, train_y, epochs=10, batch_size = 128, verbose= 2 , validation_split = 0.1)
```

```
Epoch 1/10
422/422 - 14s - loss: 0.2410 - accuracy: 0.9250 - val_loss: 0.0598 - val_accuracy: 0.9833 - 14s/epoch - 33ms/step
Epoch 2/10
422/422 - 2s - loss: 0.0869 - accuracy: 0.9746 - val_loss: 0.0405 - val_accuracy: 0.9890 - 2s/epoch - 4ms/step
Epoch 3/10
422/422 - 2s - loss: 0.0663 - accuracy: 0.9806 - val_loss: 0.0371 - val_accuracy: 0.9898 - 2s/epoch - 4ms/step
Epoch 4/10
422/422 - 2s - loss: 0.0481 - accuracy: 0.9847 - val_loss: 0.0329 - val_accuracy: 0.9913 - 2s/epoch - 4ms/step
Epoch 5/10
422/422 - 2s - loss: 0.0405 - accuracy: 0.9866 - val_loss: 0.0303 - val_accuracy: 0.9913 - 2s/epoch - 4ms/step
Epoch 6/10
422/422 - 2s - loss: 0.0360 - accuracy: 0.9880 - val_loss: 0.0269 - val_accuracy: 0.9928 - 2s/epoch - 5ms/step
Epoch 7/10
422/422 - 2s - loss: 0.0331 - accuracy: 0.9893 - val_loss: 0.0302 - val_accuracy: 0.9915 - 2s/epoch - 4ms/step
Epoch 8/10
422/422 - 2s - loss: 0.0286 - accuracy: 0.9906 - val_loss: 0.0278 - val_accuracy: 0.9933 - 2s/epoch - 4ms/step
Epoch 9/10
422/422 - 2s - loss: 0.0276 - accuracy: 0.9907 - val_loss: 0.0306 - val_accuracy: 0.9925 - 2s/epoch - 4ms/step
Epoch 10/10
422/422 - 2s - loss: 0.0232 - accuracy: 0.9926 - val_loss: 0.0277 - val_accuracy: 0.9937 - 2s/epoch - 4ms/step
```

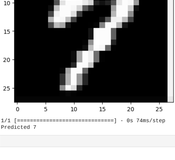
In [ ]:
```
loss, accuracy= model.evaluate(test_x, test_y, verbose = 0)
print(f'Accuracy: {accuracy*100}')
```

```
Accuracy: 99.32000041007996
```

In [14]:
```
model.save(r'C:\Users\82341\Desktop\DipLab\final_model.h5')
```

In [29]:
```
from numpy import argmax
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array
from keras.models import load_model
import matplotlib.pyplot as plt
# load and prepare the image
def load_image(filename, grayscale=True, target_size=(28, 28)):
  plt.figure(figsize=(5,5))
  plt.imshow(img,cmap='gray')
  plt.show()
  img = img_to_array(img)
  img = img.reshape(1, 28, 28, 1)
  img = img.astype('float32')
  img = img / 255.0
  return img
```

In [30]:
```
def run_example():
  img = load_image(r'/content/Screenshot 2023-07-20 055517.png')
  model = load_model(r'C:\Users\82341\Desktop\DipLab\final_model.h5')
  predict_value = model.predict(img)
  digit = argmax(predict_value)
  return 'Predicted',digit
run_example()
```



```
1/1 [==============================] - 0s 160ms/step
```

Out[30]:
```
('Predicted', 7)
```

In [37]:
```
import numpy as np
import pandas as pd
from numpy import unique, argmax
from tensorflow.keras.datasets.mnist import load_data
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dropout
from tensorflow.keras.utils import plot_model
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import mnist
```

In [38]:
```
(train_x, train_y), (test_x, test_y) = mnist.load_data()
```

In [39]:
```
print(train_x.shape, train_y.shape)
print(test_x.shape , test_y.shape)
```

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

In [40]:
```
train_x = train_x.astype('float32')/255.0
test_x = test_x.astype('float32')/255.0
```

In [41]:
```
fig = plt.figure(figsize = (20,5))
for i in range(20):
  ax= fig.add_subplot(2, 10, i+1, xticks=[], yticks=[])
  ax.imshow(np.squeeze(train_x[i]), cmap='gray')
ax.set_title(train_y[i])
```


In [42]:
```
shape = train_x.shape[1:]
shape
```

Out[42]:
```
(28, 28)
```

In [43]:
```
from tensorflow.keras.layers import MaxPooling2D as MaxPool2D
model = Sequential()
#adding convolutional layer
model.add(Conv2D(32, (3,3), activation='relu', input_shape=(28, 28, 1)))
model.add(MaxPool2D((2,2)))
model.add(Conv2D(48, (3,3), activation='relu'))
model.add(MaxPool2D((2,2)))
```

In [44]:
```
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(500, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

In [45]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 26, 26, 32)        320

 max_pooling2d_2 (MaxPooling  (None, 13, 13, 32)       0
 2D)

 conv2d_3 (Conv2D)           (None, 11, 11, 48)        13872

 max_pooling2d_3 (MaxPooling  (None, 5, 5, 48)         0
 2D)

 dropout_1 (Dropout)         (None, 5, 5, 48)          0

 flatten_1 (Flatten)         (None, 1200)              0

 dense_2 (Dense)             (None, 500)               600500

 dense_3 (Dense)             (None, 10)                5010

=================================================================
Total params: 619,702
Trainable params: 619,702
Non-trainable params: 0
_________________________________________________________________
```

In [46]:
```
model.compile(optimizer='adam', loss = 'sparse_categorical_crossentropy',metrics= ['accuracy'] )
x=model.fit(train_x, train_y, epochs=10, batch_size = 128, verbose= 2 , validation_split = 0.1)
```

```
Epoch 1/10
422/422 - 6s - loss: 0.2389 - accuracy: 0.9275 - val_loss: 0.0619 - val_accuracy: 0.9825 - 6s/epoch - 14ms/step
Epoch 2/10
422/422 - 2s - loss: 0.0783 - accuracy: 0.9749 - val_loss: 0.0456 - val_accuracy: 0.9868 - 2s/epoch - 4ms/step
Epoch 3/10
422/422 - 2s - loss: 0.0590 - accuracy: 0.9822 - val_loss: 0.0339 - val_accuracy: 0.9910 - 2s/epoch - 4ms/step
Epoch 4/10
422/422 - 2s - loss: 0.0489 - accuracy: 0.9841 - val_loss: 0.0377 - val_accuracy: 0.9898 - 2s/epoch - 4ms/step
Epoch 5/10
422/422 - 2s - loss: 0.0427 - accuracy: 0.9861 - val_loss: 0.0302 - val_accuracy: 0.9915 - 2s/epoch - 4ms/step
Epoch 6/10
422/422 - 2s - loss: 0.0343 - accuracy: 0.9891 - val_loss: 0.0260 - val_accuracy: 0.9932 - 2s/epoch - 5ms/step
Epoch 7/10
422/422 - 2s - loss: 0.0304 - accuracy: 0.9903 - val_loss: 0.0275 - val_accuracy: 0.9923 - 2s/epoch - 5ms/step
Epoch 8/10
422/422 - 2s - loss: 0.0280 - accuracy: 0.9910 - val_loss: 0.0267 - val_accuracy: 0.9930 - 2s/epoch - 4ms/step
Epoch 9/10
422/422 - 2s - loss: 0.0250 - accuracy: 0.9921 - val_loss: 0.0254 - val_accuracy: 0.9935 - 2s/epoch - 4ms/step
Epoch 10/10
422/422 - 2s - loss: 0.0244 - accuracy: 0.9916 - val_loss: 0.0245 - val_accuracy: 0.9938 - 2s/epoch - 4ms/step
```

In [47]:
```
loss, accuracy= model.evaluate(test_x, test_y, verbose = 0)
print(f'Accuracy: {accuracy*100}')
```

```
Accuracy: 99.27999973297119
```

In [48]:
```
model.save(r'C:\Users\82341\Desktop\DipLab\final_model.h5')
```

In [57]:
```
from numpy import argmax
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array
from keras.models import load_model
import matplotlib.pyplot as plt
# load and prepare the image
def load_image(filename, grayscale=True, target_size=(28, 28)):
  img = load_img(filename, grayscale=True, target_size=(28, 28))
  plt.figure(figsize=(5,5))
  plt.imshow(img,cmap='gray')
  plt.show()
  # convert to array
  img = img_to_array(img)
  # reshape into a single sample with 1 channel
  img = img.reshape(1, 28, 28, 1)
  # prepare pixel data
  img = img.astype('float32')
  img = img / 255.0
  return img
# load an image and predict the class
def run_example():
  # load the image
  img = load_image(r'/content/Screenshot 2023-07-20 055517.png')
  # load model
  model = load_model(r'C:\Users\82341\Desktop\DipLab\final_model.h5')
  # predict the class
  predict_value = model.predict(img)
  digit = argmax(predict_value)
  print('Predicted',digit)
# entry point, run the example
run_example()
```

```
/usr/local/lib/python3.10/dist-packages/keras/utils/image_utils.py:409: UserWarning: grayscale is deprecated. Please use color_mode = "grayscale"
  warnings.warn(
```



```
1/1 [==============================] - 0s 74ms/step
Predicted 7
```

In [ ]:
```
```