Name - Arkadipta Mojumder

Registration Number - 22MCA0201

Lab Assessment-5

Q. Develop Convolution Neural Network model for Handwritten digits dataset and compare accuracy with ANN Model.

Importing Necessary Libraries

In [4]:
```
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, MaxPooling2D
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
```

Loading Data

In [5]:
```
(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 [==============================] - 0s 0us/step
```

In [6]:
```
print(train_x.shape, train_y.shape)
print(test_x.shape , test_y.shape)
```

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

Visulaization and Preprocessing

In [9]:
```
train_x = train_x.reshape((train_x.shape[0], train_x.shape[1], train_x.shape[2], 1))
 test_x = test_x .reshape((test_x.shape[0], test_x.shape[1], test_x.shape[2],1))
```

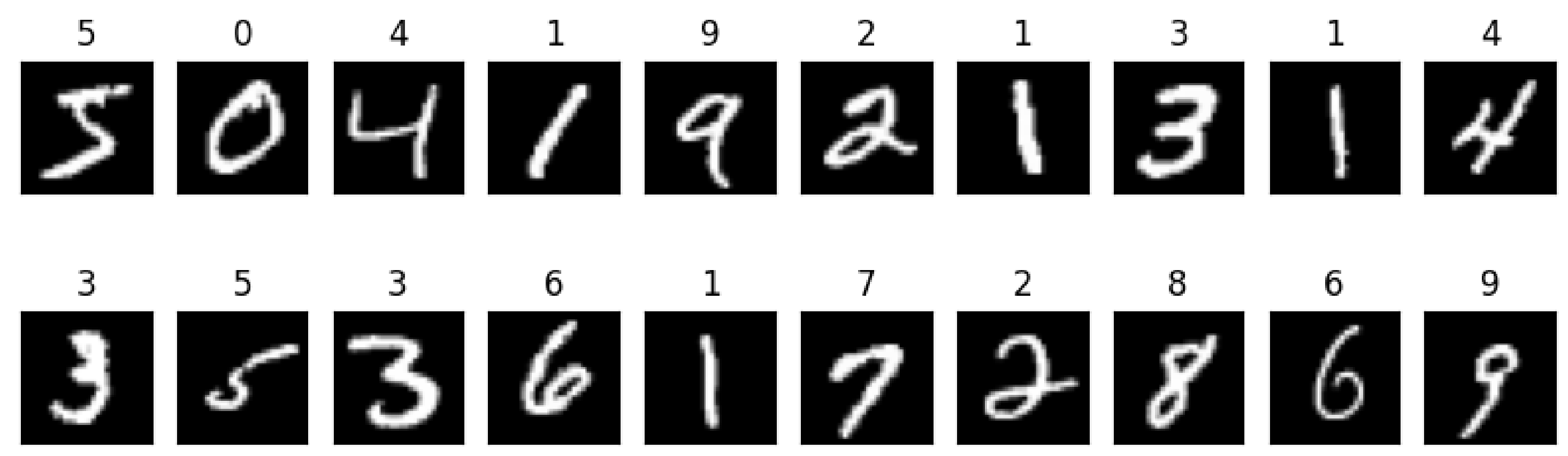
In [10]:
```
print(train_x.shape, train_y.shape)
print(test_x.shape, test_y.shape)
```

```
(60000, 28, 28, 1) (60000,)
(10000, 28, 28, 1) (10000,)
```

In [11]:
```
train_x = train_x.astype('float32')/255.0
test_x = test_x.astype('float32')/255.0
```

In [14]:
```
#plotting images of dataset

fig = plt.figure(figsize = (10, 3))
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 [15]:
```
shape = train_x.shape[1 : ]
shape
```

Out[15]:
```
(28, 28, 1)
```

Training

In [16]:
```
#CNN Model

model = Sequential()

#adding convolutional layer
model.add(Conv2D(32, (3,3), activation='relu', input_shape= shape))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(48, (3,3), activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(500, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

In [17]:
```
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
_________________________________________________________________
```

Compiling and Training the model

In [18]:
```
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 - 48s - loss: 0.2492 - accuracy: 0.9257 - val_loss: 0.0544 - val_accuracy: 0.9855 - 48s/epoch - 114ms/step
Epoch 2/10
422/422 - 46s - loss: 0.0798 - accuracy: 0.9745 - val_loss: 0.0392 - val_accuracy: 0.9885 - 46s/epoch - 110ms/step
Epoch 3/10
422/422 - 46s - loss: 0.0582 - accuracy: 0.9814 - val_loss: 0.0338 - val_accuracy: 0.9905 - 46s/epoch - 108ms/step
Epoch 4/10
422/422 - 46s - loss: 0.0469 - accuracy: 0.9854 - val_loss: 0.0336 - val_accuracy: 0.9910 - 46s/epoch - 110ms/step
Epoch 5/10
422/422 - 46s - loss: 0.0408 - accuracy: 0.9867 - val_loss: 0.0290 - val_accuracy: 0.9923 - 46s/epoch - 110ms/step
Epoch 6/10
422/422 - 45s - loss: 0.0358 - accuracy: 0.9885 - val_loss: 0.0272 - val_accuracy: 0.9923 - 45s/epoch - 107ms/step
Epoch 7/10
422/422 - 46s - loss: 0.0310 - accuracy: 0.9901 - val_loss: 0.0295 - val_accuracy: 0.9910 - 46s/epoch - 110ms/step
Epoch 8/10
422/422 - 45s - loss: 0.0274 - accuracy: 0.9909 - val_loss: 0.0317 - val_accuracy: 0.9915 - 45s/epoch - 107ms/step
Epoch 9/10
422/422 - 47s - loss: 0.0260 - accuracy: 0.9915 - val_loss: 0.0240 - val_accuracy: 0.9925 - 47s/epoch - 111ms/step
Epoch 10/10
422/422 - 47s - loss: 0.0233 - accuracy: 0.9923 - val_loss: 0.0253 - val_accuracy: 0.9940 - 47s/epoch - 111ms/step
```

In [19]:
```
loss, accuracy= model.evaluate(test_x, test_y, verbose =0)
print(f'Accuracy: {accuracy * 100}')
```

```
Accuracy: 99.36000108718872
```