CNN Digit Classification

Step 1: Importing the libraries

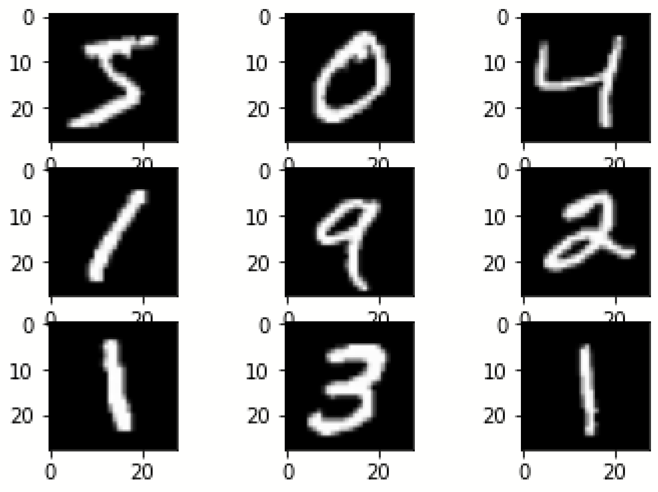
```
import matplotlib.pyplot as plt
import numpy as np
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
```

Step 2: Loading the MNIST Dataset

```
# load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

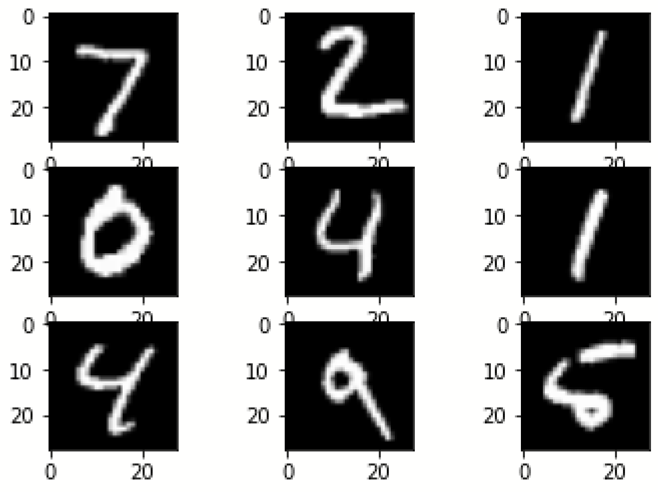
Step 3: Looking into Train Data

```
#Looking into Train Data
for i in range(9):
  # define subplot
  plt.subplot(330 + 1 + i)
  # plot raw pixel data
  plt.imshow(X_train[i], cmap=plt.get_cmap('gray'))
# show the figure
plt.show()
```



Step 4: Looking into Test Data

```
#Looking into test data
for i in range(9):
  # define subplot
  plt.subplot(330 + 1 + i)
  # plot raw pixel data
  plt.imshow(X_test[i], cmap=plt.get_cmap('gray'))
# show the figure
plt.show()
```



Step 5: Reshaping the Images

```
# reshape to be [samples][width][height][channels]
X_train = X_train.reshape((X_train.shape[0], 28, 28, 1)).astype('float32')
X_test = X_test.reshape((X_test.shape[0], 28, 28, 1)).astype('float32')
```

Step 6: Normalizing the Inputs from 0-255 to 0-1

```
# normalize inputs from 0-255 to 0-1
X_train = X_train / 255.0
X_test = X_test / 255.0
```

Step 7: Getting the one-hot encoding outputs

```
# one hot encode outputs
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]
```

Step 8: Defining a CNN Model function

```
# define a simple CNN model
def cnn_model():
  # create model
  model = Sequential()
  model.add(Conv2D(30, (5, 5), input_shape=(28, 28, 1), activation='relu'))
  model.add(MaxPooling2D())
  model.add(Conv2D(15, (3, 3), activation='relu'))
  model.add(MaxPooling2D())
  model.add(Dropout(0.2))
  model.add(Flatten())
  model.add(Dense(128, activation='relu'))
  model.add(Dense(50, activation='relu'))
  model.add(Dense(num_classes, activation='softmax'))
  # Compile model
  model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
  return model
```

Step 8: Training

```
# build the model
model =cnn_model()
# Fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=20,
batch_size=200)
```

Step 7: Accuracy & Error

```
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
print("CNN Error: %.2f%%" % (100-scores[1]*100))
```

```
Accuracy: 99.30%
CNN Error: 0.70%
```

How to verify if our model is working?

Using Gradio interface checking if our model works fine, when the user submits a handwritten digit