CODEXCUE

DEEP LEARNING

Handwritten Digit Classification:

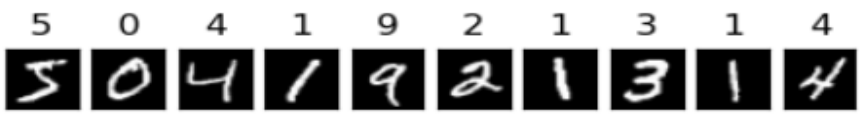
```
 #pip install tensorflow
# Importing necessary libraries
from numpy import unique, argmax
from tensorflow.keras.datasets.mnist import load_data
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPool2D
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dropout
from tensorflow.keras.utils import plot_model
from matplotlib import pyplot
import matplotlib.pyplot as plt
import numpy as np
```

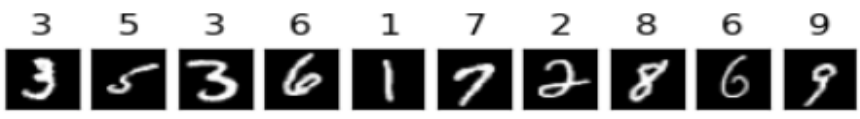
```
# Loading the MNIST dataset
(x_train, y_train), (x_test, y_test) = load_data()

# Reshaping the training and testing data
x_train = x_train.reshape((x_train.shape[0], x_train.shape[1], x_train.shape[2], 1))
x_test = x_test.reshape((x_test.shape[0], x_test.shape[1], x_test.shape[2], 1))
```

```
# Normalizing the values of pixels of images
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0
```

```
# Displaying some of the training images
fig = plt.figure(figsize=(5, 3))
for i in range(20):
    ax = fig.add_subplot(2, 10, i + 1, xticks=[], yticks=[])
    ax.imshow(np.squeeze(x_train[i]), cmap='gray')
    ax.set_title(y_train[i])
```





```
# Determine the shape of the input images
img_shape = x_train.shape[1:]
print(img_shape)
```

```
(28, 28, 1)
```

```
# Defining the model
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=img_shape))
model.add(MaxPool2D((2, 2)))
model.add(Conv2D(48, (3, 3), activation='relu'))
model.add(MaxPool2D((2, 2)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(500, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

```
# Displaying the model summary
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_2 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_3 (Conv2D)	(None, 11, 11, 48)	13,872
max_pooling2d_3 (MaxPooling2D)	(None, 5, 5, 48)	0
dropout_1 (Dropout)	(None, 5, 5, 48)	0
flatten_1 (Flatten)	(None, 1200)	0
dense_2 (Dense)	(None, 500)	600,500
dense_3 (Dense)	(None, 10)	5,010

Total params: 619,702 (2.36 MB)

Trainable params: 619,702 (2.36 MB)

Non-trainable params: 0 (0.00 B)

```
plot_model(model, 'model.jpg' , show_shapes=True)
```

```
You must install graphviz (see instructions at https://graphviz.gitlab.io/download/) for `plot_model` to work.
```

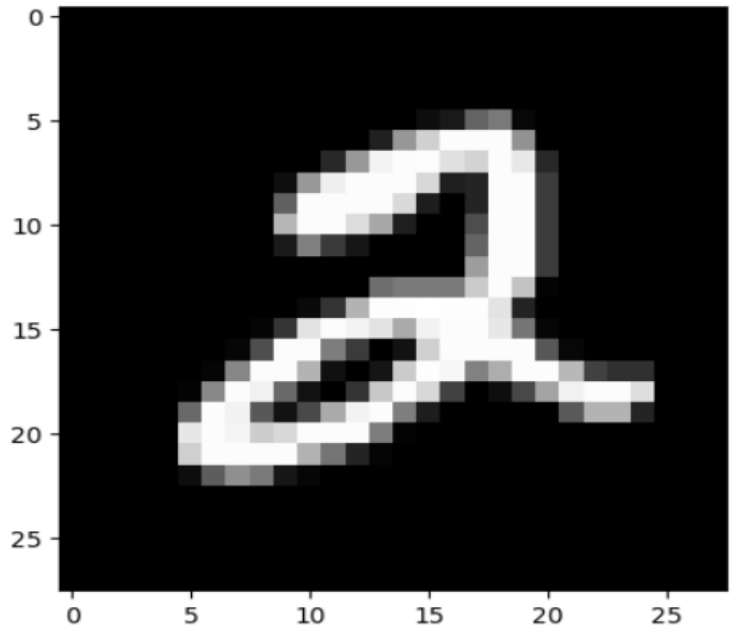
```
# Compiling the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
# Training the model
history = model.fit(x_train, y_train, epochs=10, batch_size=128, verbose=2, validation_split=0.1)
```

```
Epoch 1/10
422/422 - 23s - 54ms/step - accuracy: 0.9258 - loss: 0.2444 - val_accuracy: 0.9855 - val_loss: 0.0514
Epoch 2/10
422/422 - 16s - 38ms/step - accuracy: 0.9750 - loss: 0.0801 - val_accuracy: 0.9875 - val_loss: 0.0438
Epoch 3/10
422/422 - 15s - 37ms/step - accuracy: 0.9819 - loss: 0.0597 - val_accuracy: 0.9905 - val_loss: 0.0327
Epoch 4/10
422/422 - 16s - 37ms/step - accuracy: 0.9851 - loss: 0.0483 - val_accuracy: 0.9903 - val_loss: 0.0323
Epoch 5/10
422/422 - 16s - 38ms/step - accuracy: 0.9865 - loss: 0.0412 - val_accuracy: 0.9933 - val_loss: 0.0287
Epoch 6/10
422/422 - 16s - 38ms/step - accuracy: 0.9887 - loss: 0.0369 - val_accuracy: 0.9925 - val_loss: 0.0288
Epoch 7/10
422/422 - 15s - 37ms/step - accuracy: 0.9897 - loss: 0.0309 - val_accuracy: 0.9918 - val_loss: 0.0246
Epoch 8/10
422/422 - 15s - 37ms/step - accuracy: 0.9903 - loss: 0.0292 - val_accuracy: 0.9913 - val_loss: 0.0277
Epoch 9/10
422/422 - 16s - 37ms/step - accuracy: 0.9915 - loss: 0.0263 - val_accuracy: 0.9938 - val_loss: 0.0241
Epoch 10/10
422/422 - 16s - 37ms/step - accuracy: 0.9921 - loss: 0.0234 - val_accuracy: 0.9922 - val_loss: 0.0288
```

```
# Evaluating the model
loss, accuracy = model.evaluate(x_test, y_test, verbose=0)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

```
Accuracy: 98.99%
```

```
# Displaying the image we want to predict
image = x_train[5]
plt.imshow(np.squeeze(image), cmap='gray')
plt.show()
```



```
image=image.reshape(1,image.shape[0],image.shape[1],image.shape[2])
p=model.predict([image])
print('predicted:{}'.format(argmax(p)))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 238ms/step
predicted:2
```