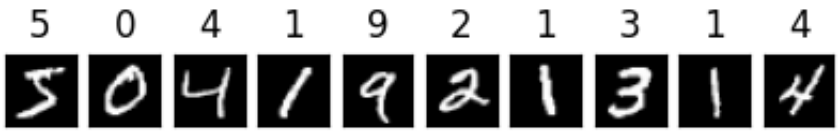
```
from numpy import unique, argmax
from tensorflow.keras.datasets.mnist import load_data
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras. layers import MaxPool2D
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
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[
x_test = x_test.reshape ((x_test.shape[0], x_test.shape[1], x_test.shape[2], 1)


x_train = x_train.astype('float32')/255.0 # 255 =  pixel values are integer val
x_test = x_test.astype('float32')/255.0 # changes in 32 bit memory



fig = plt.figure(figsize= (5,3))
for i in range (15):
    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])
```



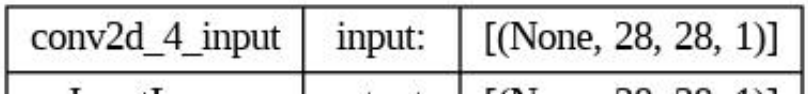
```
# determine the shape of the input images
img_shape = x_train.shape [1:]
print (img_shape)
```

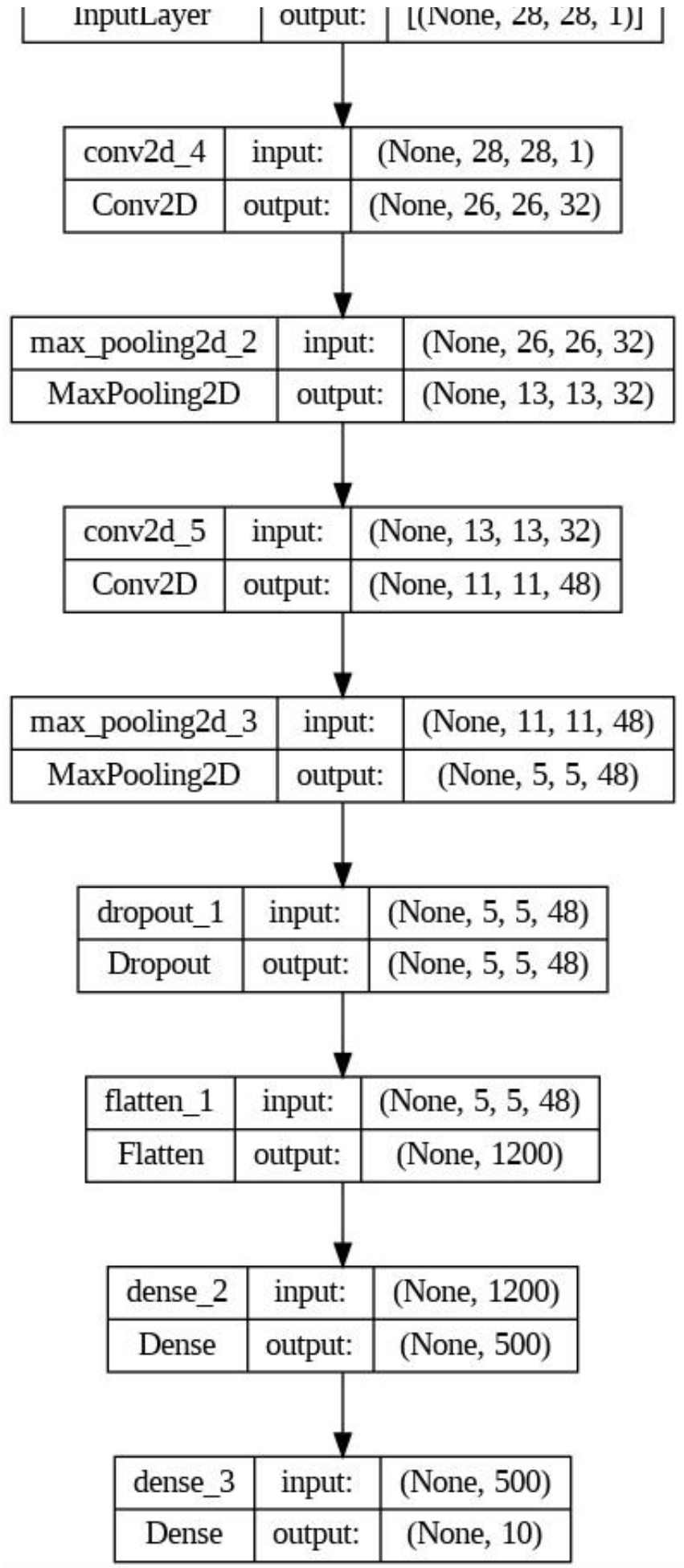
```
(28, 28, 1)
```

```
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'))
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 26, 26, 32)        320

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

 conv2d_5 (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
_________________________________________________________________
```

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




InputLayer | output: | [(None, 28, 28, 1)]
conv2d_4 | input: | (None, 28, 28, 1)
Conv2D | output: | (None, 26, 26, 32)
max_pooling2d_2 | input: | (None, 26, 26, 32)
MaxPooling2D | output: | (None, 13, 13, 32)
conv2d_5 | input: | (None, 13, 13, 32)
Conv2D | output: | (None, 11, 11, 48)
max_pooling2d_3 | input: | (None, 11, 11, 48)
MaxPooling2D | output: | (None, 5, 5, 48)
dropout_1 | input: | (None, 5, 5, 48)
Dropout | output: | (None, 5, 5, 48)
flatten_1 | input: | (None, 5, 5, 48)
Flatten | output: | (None, 1200)
dense_2 | input: | (None, 1200)
Dense | output: | (None, 500)
dense_3 | input: | (None, 500)
Dense | output: | (None, 10)

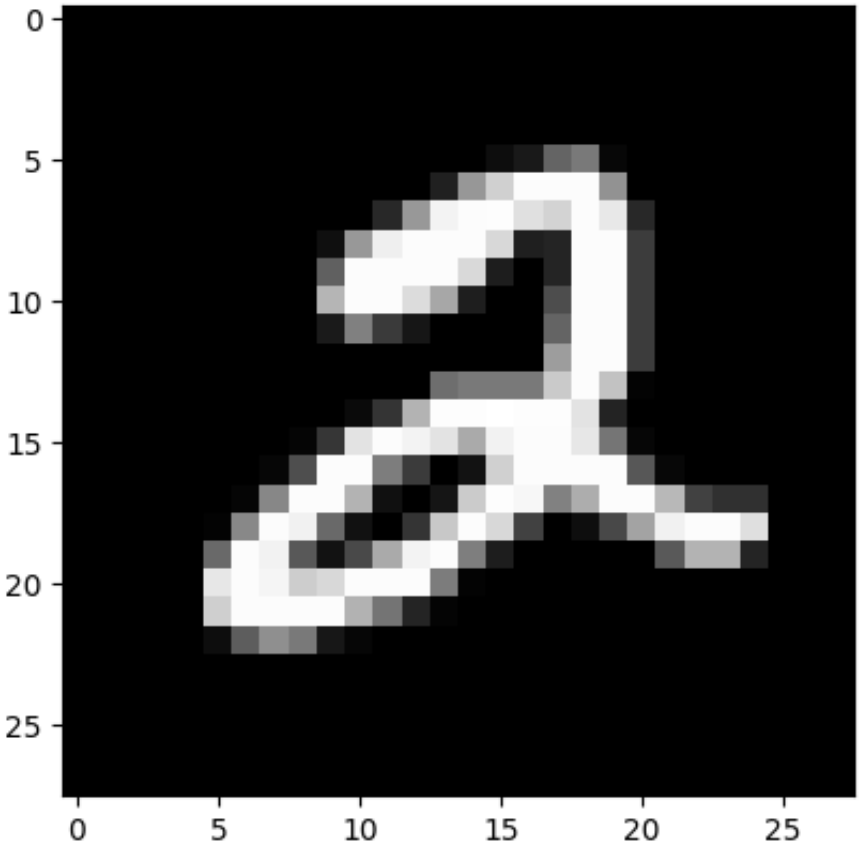
```
model.compile (optimizer='adam', loss='sparse_categorical_crossentropy', metrics
x = model.fit(x_train, y_train, epochs=10, batch_size=128, verbose=2, validation
```

```
Epoch 1/10
422/422 - 50s - loss: 0.2365 - accuracy: 0.9278 - val_loss: 0.0561 - val_ac
Epoch 2/10
422/422 - 43s - loss: 0.0813 - accuracy: 0.9744 - val_loss: 0.0398 - val_ac
Epoch 3/10
422/422 - 43s - loss: 0.0599 - accuracy: 0.9809 - val_loss: 0.0359 - val_ac
Epoch 4/10
422/422 - 44s - loss: 0.0472 - accuracy: 0.9844 - val_loss: 0.0305 - val_ac
Epoch 5/10
422/422 - 43s - loss: 0.0409 - accuracy: 0.9872 - val_loss: 0.0310 - val_ac
Epoch 6/10
422/422 - 44s - loss: 0.0356 - accuracy: 0.9881 - val_loss: 0.0309 - val_ac
Epoch 7/10
422/422 - 43s - loss: 0.0324 - accuracy: 0.9895 - val_loss: 0.0308 - val_ac
Epoch 8/10
422/422 - 44s - loss: 0.0278 - accuracy: 0.9909 - val_loss: 0.0261 - val_ac
Epoch 9/10
422/422 - 44s - loss: 0.0261 - accuracy: 0.9914 - val_loss: 0.0256 - val_ac
Epoch 10/10
422/422 - 43s - loss: 0.0239 - accuracy: 0.9920 - val_loss: 0.0211 - val_ac
```

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

```
Accuracy: 99.27999973297119
```

```
image = x_train[5]
# lets display the image which we want to predict
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 109ms/step
Predicted: 2
```

Colab paid products - Cancel contracts here

✓ 0s completed at 6:06 PM