assignment06-1_MeyerJake

April 23, 2023

0.1 Assignment 6-1

0.1.1 DSC 650

0.1.2 Jake Meyer

0.1.3 04/22/2023

Using section 5.1 in Deep Learning with Python as a guide (listing 5.3 in particular), create a ConvNet model that classifies images in the MNIST digit dataset. Save the model, predictions, metrics, and validation plots in the dsc650/assignments/assignment06/results directory. If you are using JupyterHub, you can include those plots in your Jupyter notebook.

Using code from deep-learning-with-python-notebooks

[1]:

```
## Import the necessary modules for the assignment above.
import csv
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import keras
import sklearn
from sklearn.model_selection import train_test_split
import itertools
from pathlib import Path
import time
import os

## Import the necessary keras components for the data and CNN
from keras import layers, models
from keras.datasets import mnist
from keras.utils import to_categorical, np_utils
from keras.models import Sequential, load_model
from keras.layers.core import Dense, Dropout, Activation
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
```

```
WARNING:tensorflow:From C:\Users\jkmey\anaconda3\envs\dsc650\lib\site-
packages\tensorflow\python\compat\v2_compat.py:107: disable_resource_variables
(from tensorflow.python.ops.variable_scope) is deprecated and will be removed in
```

```
a future version.
Instructions for updating:
non-resource variables are not supported in the long term
```

[2]:

```
## Print versions of essential packages
print("keras version: {}".format(keras.__version__))
print("tensorflow version: {}".format(tf.__version__))
print("pandas version: {}".format(pd.__version__))
print("numpy version: {}".format(np.__version__))
```

```
keras version: 2.11.0
tensorflow version: 2.11.0
pandas version: 1.5.3
numpy version: 1.24.2
```

[3]:

```
## Try to setup tensorflow to run on GPU using ConfigProto()
## config = tf.compat.v1.ConfigProto
## devices = tf.config.experimental.list_physical_devices("GPU")
## tf.config.experimental.set_memory_growth(devices, True)
```

[4]:

```
## Setup the directories for the assignment
current_dir = Path('C:/Users/jkmey/Documents/Github/DSC650_Course_Assignments/
 ↪dsc650/dsc650/assignments/assignment06')
results_dir = Path('C:/Users/jkmey/Documents/Github/DSC650_Course_Assignments/
 ↪dsc650/dsc650/assignments/assignment06/').joinpath('results')
results_dir.mkdir(parents = True, exist_ok = True)
```

0.1.4 Import the MSNT Dataset

[5]:

```
## Load the data from mnist as specified from Deep Learning with Python␣
 ↪Textbook.
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

[6]:

```
## Understand the shape of the train and test datasets.
print('train_images: {}'.format(train_images.shape))
print('test_images: {}'.format(test_images.shape))
print('train_labels: {}'.format(train_labels.shape))
print('test_labels: {}'.format(test_labels.shape))
```

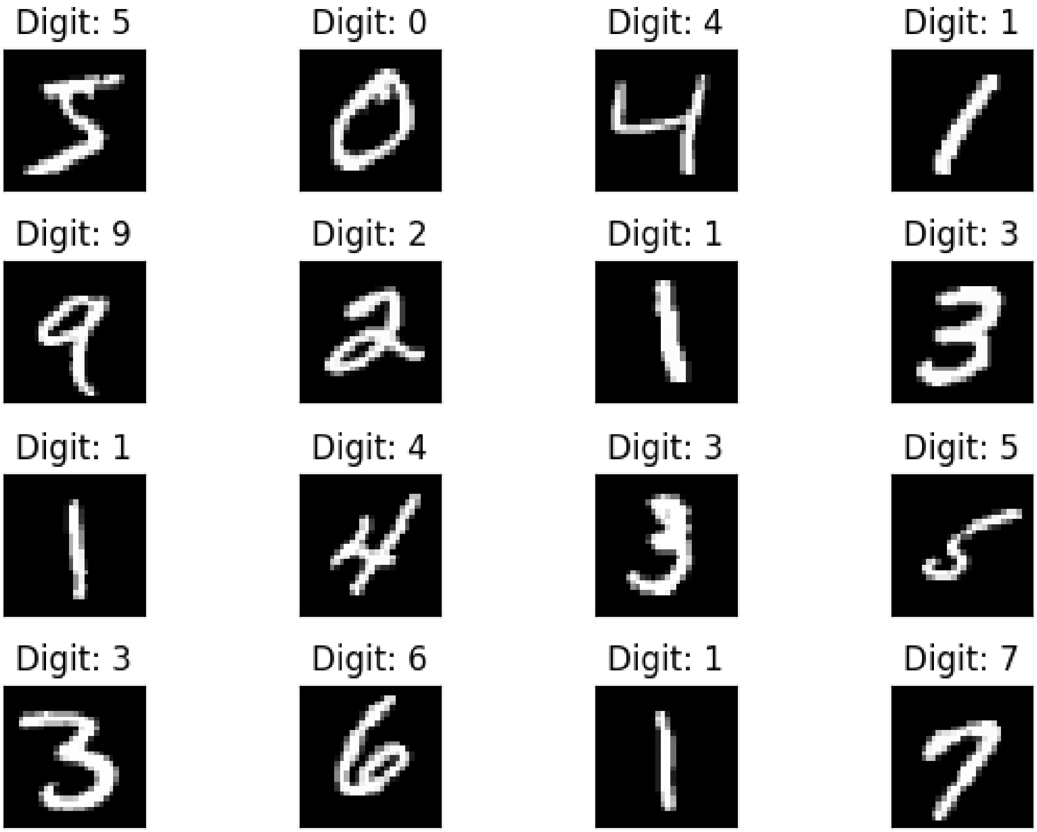
```
train_images: (60000, 28, 28)
test_images: (10000, 28, 28)
train_labels: (60000,)
test_labels: (10000,)
```

0.1.5 Show Training Images and Labels

[7]:

```
## Show the first 16 training images and labels for better understanding of the␣
 ↪data.
fig = plt.figure()
for i in range(16):
    plt.subplot(4,4,i+1)
    plt.tight_layout()
    plt.imshow(train_images[i], cmap = 'gray', interpolation='none')
    plt.title("Digit: {}".format(train_labels[i]))
    plt.xticks([])
    plt.yticks([])
img_file = results_dir.joinpath('assignment06-1_Sample_Digits_QTY_16.png')
plt.savefig(img_file)
print("First 16 Training Images and Labels")
plt.show()
```

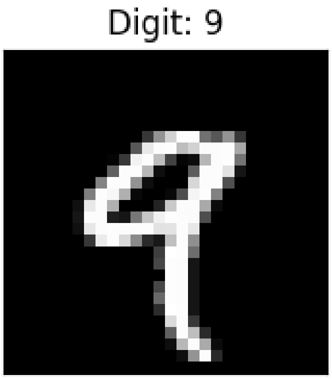
```
First 16 Training Images and Labels
```



0.1.6 Pixel Value Histogram

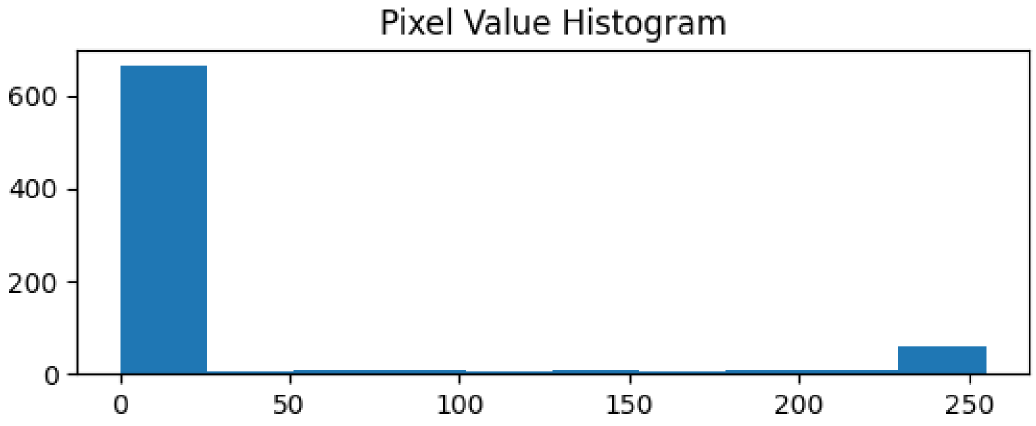
[8]:

```
## Code to check the digit in the train image with the label shown from 0-9.
fig = plt.figure()
plt.subplot(2,1,1)
plt.imshow(train_images[4], cmap = 'gray', interpolation = 'none')
plt.title('Digit: {}'.format(train_labels[4]))
plt.xticks([])
plt.yticks([])
img_file = results_dir.joinpath('assignment06-1_Digit_Overview.png')
plt.savefig(img_file)
plt.show()
```



[9]:

```
## Pixel distribution shown in the plot below for the image chosen in the␣
 ↪previous cell.
plt.subplot(2,1,2)
plt.hist(train_images[4].reshape(784)) # Value needs to be 784 for reshape,␣
 ↪otherwise error
plt.title("Pixel Value Histogram")
img_file = results_dir.joinpath('assignment06-1_Pixel_Value_Histogram.png')
plt.savefig(img_file)
plt.show()
```



0.1.7 Prepare the Data

[10]:

```
## Reshape the training images and normalize.
train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images.astype('float32') / 255

## Reshape the testing images and normalize.
test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images.astype('float32') / 255

## Convert the training and test labels to numbers.
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

[11]:

```
## Split train_images and train_labels into train and validation subsets.
train_images_val = train_images[:10000]
train_images = train_images[10000:]
train_labels_val = train_labels[:10000]
train_labels = train_labels[10000:]
```

0.1.8 Create the CNN Model

[12]:

```
## From the textbook repository, Instantiate the CNN Model
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))

## Add a classifier on top of the CNN (also from the textbook repository)
```

```
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

[13]:
```
## Show a summary of the model that was just created.
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, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 3, 3, 64)          36928

 flatten (Flatten)           (None, 576)               0

 dense (Dense)               (None, 64)                36928

 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
_________________________________________________________________
```

0.1.9 Train the Model

[14]:
```
## Train the model and store the results in the variable history.
history = model.fit(train_images, train_labels, epochs=20, batch_size=128,␣
 ↪verbose = 2,
                          validation_data = (train_images_val, train_labels_val))
```

```
Train on 50000 samples, validate on 10000 samples
WARNING:tensorflow:OMP_NUM_THREADS is no longer used by the default Keras
```

```
config. To configure the number of threads, use tf.config.threading APIs.
Epoch 1/20
```

```
C:\Users\jkmey\anaconda3\envs\dsc650\lib\site-
packages\keras\engine\training_v1.py:2333: UserWarning: `Model.state_updates`
will be removed in a future version. This property should not be used in
TensorFlow 2.0, as `updates` are applied automatically.
  updates = self.state_updates
```

```
50000/50000 - 11s - loss: 0.2669 - acc: 0.9162 - val_loss: 0.1013 - val_acc:
0.9692 - 11s/epoch - 222us/sample
Epoch 2/20
50000/50000 - 12s - loss: 0.0605 - acc: 0.9811 - val_loss: 0.0536 - val_acc:
0.9853 - 12s/epoch - 234us/sample
Epoch 3/20
50000/50000 - 12s - loss: 0.0402 - acc: 0.9872 - val_loss: 0.0525 - val_acc:
0.9834 - 12s/epoch - 235us/sample
Epoch 4/20
50000/50000 - 11s - loss: 0.0302 - acc: 0.9905 - val_loss: 0.0403 - val_acc:
0.9888 - 11s/epoch - 216us/sample
Epoch 5/20
50000/50000 - 11s - loss: 0.0235 - acc: 0.9923 - val_loss: 0.0360 - val_acc:
0.9900 - 11s/epoch - 227us/sample
Epoch 6/20
50000/50000 - 11s - loss: 0.0182 - acc: 0.9943 - val_loss: 0.0367 - val_acc:
0.9904 - 11s/epoch - 216us/sample
Epoch 7/20
50000/50000 - 11s - loss: 0.0158 - acc: 0.9951 - val_loss: 0.0360 - val_acc:
0.9903 - 11s/epoch - 216us/sample
Epoch 8/20
50000/50000 - 11s - loss: 0.0125 - acc: 0.9964 - val_loss: 0.0430 - val_acc:
0.9889 - 11s/epoch - 216us/sample
Epoch 9/20
50000/50000 - 11s - loss: 0.0099 - acc: 0.9970 - val_loss: 0.0359 - val_acc:
0.9908 - 11s/epoch - 221us/sample
Epoch 10/20
50000/50000 - 11s - loss: 0.0084 - acc: 0.9974 - val_loss: 0.0498 - val_acc:
0.9890 - 11s/epoch - 217us/sample
Epoch 11/20
50000/50000 - 11s - loss: 0.0068 - acc: 0.9977 - val_loss: 0.0442 - val_acc:
0.9906 - 11s/epoch - 215us/sample
Epoch 12/20
50000/50000 - 11s - loss: 0.0060 - acc: 0.9979 - val_loss: 0.0400 - val_acc:
0.9920 - 11s/epoch - 214us/sample
Epoch 13/20
50000/50000 - 11s - loss: 0.0055 - acc: 0.9980 - val_loss: 0.0472 - val_acc:
0.9906 - 11s/epoch - 215us/sample
Epoch 14/20
50000/50000 - 11s - loss: 0.0044 - acc: 0.9987 - val_loss: 0.0524 - val_acc:
```

```
0.9911 - 11s/epoch - 214us/sample
Epoch 15/20
50000/50000 - 11s - loss: 0.0041 - acc: 0.9987 - val_loss: 0.0559 - val_acc:
0.9902 - 11s/epoch - 215us/sample
Epoch 16/20
50000/50000 - 11s - loss: 0.0042 - acc: 0.9987 - val_loss: 0.0518 - val_acc:
0.9916 - 11s/epoch - 215us/sample
Epoch 17/20
50000/50000 - 11s - loss: 0.0033 - acc: 0.9989 - val_loss: 0.0490 - val_acc:
0.9929 - 11s/epoch - 215us/sample
Epoch 18/20
50000/50000 - 11s - loss: 0.0028 - acc: 0.9990 - val_loss: 0.0674 - val_acc:
0.9909 - 11s/epoch - 214us/sample
Epoch 19/20
50000/50000 - 11s - loss: 0.0028 - acc: 0.9992 - val_loss: 0.0488 - val_acc:
0.9922 - 11s/epoch - 216us/sample
Epoch 20/20
50000/50000 - 11s - loss: 0.0030 - acc: 0.9990 - val_loss: 0.0537 - val_acc:
0.9923 - 11s/epoch - 216us/sample
```

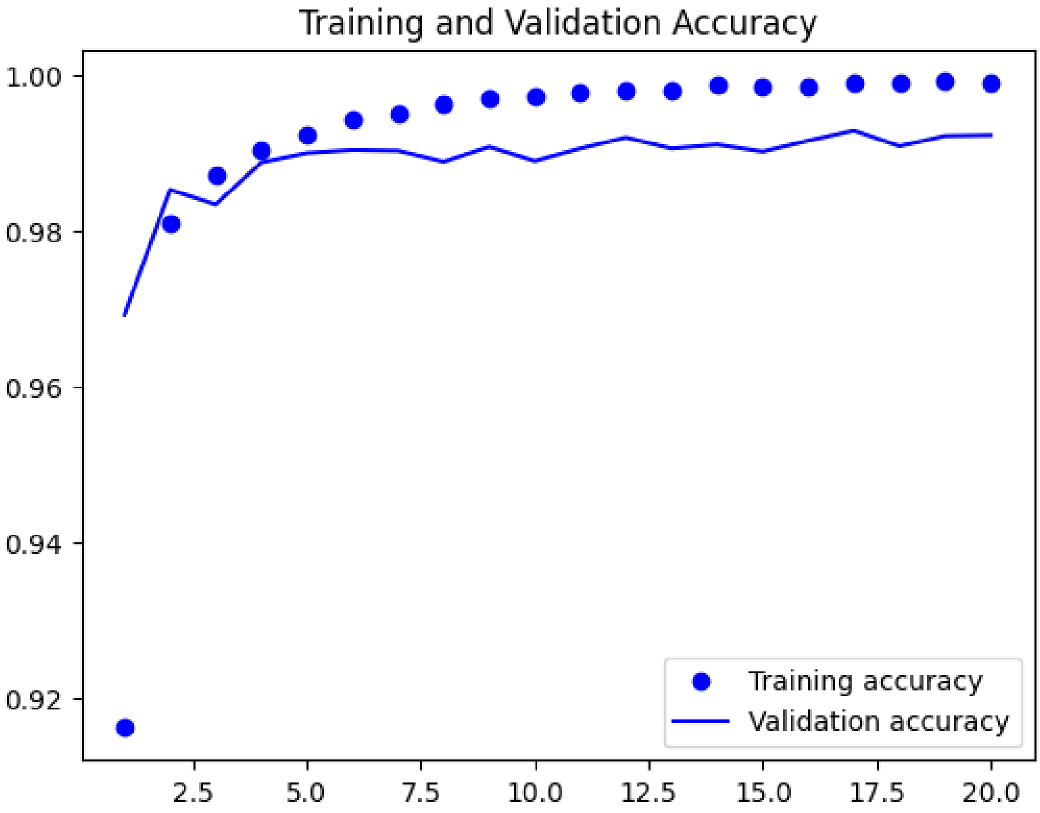
[16]:

```python
## Save the result model file to the results directory.
result_model_file = results_dir.joinpath('assignment06-1_Model.h5')
model.save(result_model_file)
print("Saved the Trained model at %s " % result_model_file)
```

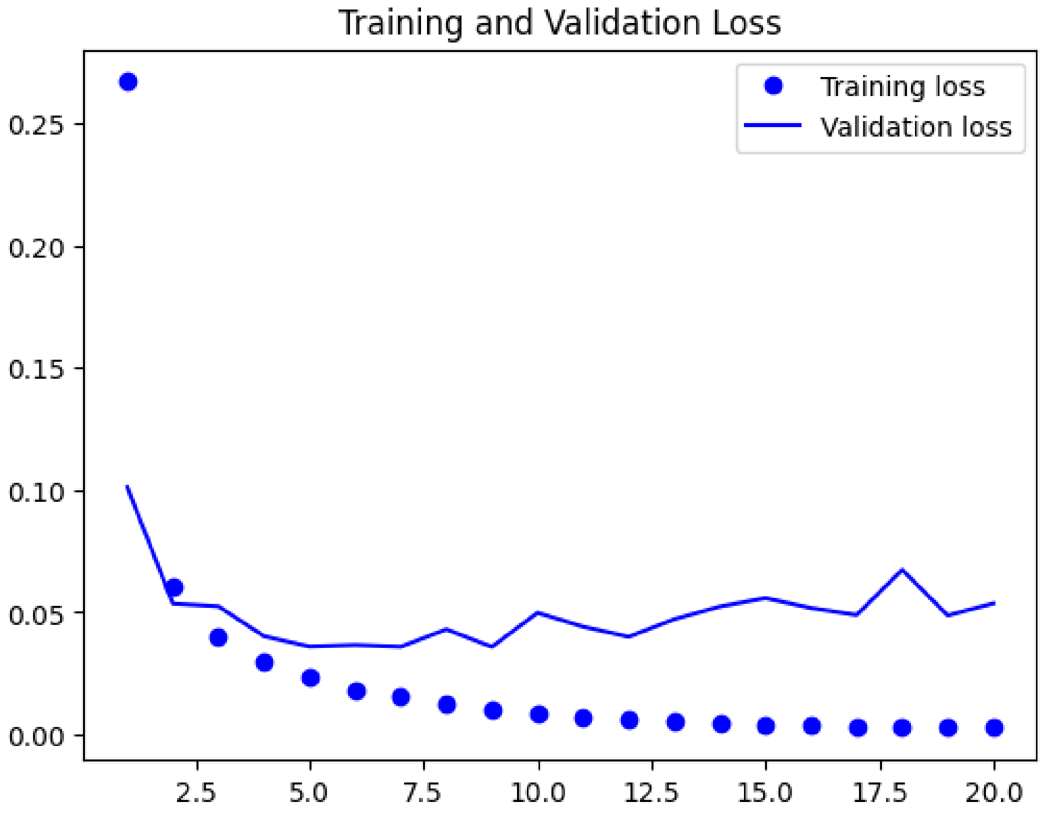
```
Saved the Trained model at C:\Users\jkmey\Documents\Github\DSC650_Course_Assignm
ents\dsc650\dsc650\assignments\assignment06\results\assignment06-1_Model.h5
```

[22]:

```python
## Generate and Save Plot of Training and Validation Accuracy from Model.
accuracy = history.history["acc"]
val_accuracy = history.history["val_acc"]
epochs = range(1, len(accuracy) + 1)
plt.plot(epochs, accuracy, "bo", label="Training accuracy")
plt.plot(epochs, val_accuracy, "b", label="Validation accuracy")
plt.title("Training and Validation Accuracy")
plt.legend()
img_file = results_dir.
 ↪joinpath('assignment06-1_Training_and_Validation_Accuracy_Plot.png')
plt.savefig(img_file)
plt.show()
```



```
[18]: ## Generate and Save Plot of Training and Validation Loss from Model.
loss = history.history["loss"]
val_loss = history.history["val_loss"]
epochs = range(1, len(accuracy) + 1)
plt.plot(epochs, loss, "bo", label="Training loss")
plt.plot(epochs, val_loss, "b", label="Validation loss")
plt.title("Training and Validation Loss")
plt.legend()
img_file = results_dir.
 ↪joinpath('assignment06-1_Training_and_Validation_Loss_Plot.png')
plt.savefig(img_file)
plt.show()
```



0.1.10 CNN Model Results on Test Data

[19]:

```
## Evaluate the model on the test subsets. Code from the textbook repository.
test_loss, test_acc = model.evaluate(test_images, test_labels)
```

[20]:

```
## Show the Test Accuracy and Loss from the cell above.
print("Test Accuracy: {}%".format((test_acc)*100))
print("Test Loss: {}".format(test_loss))
```

```
Test Accuracy: 99.18000102043152%
Test Loss: 0.054932919396300336
```

[24]:

```
## Write the Test Accuracy and Loss to the results folder.
csv_test = results_dir.joinpath('assignment06-1_Test_Accuracy_Loss_Results.csv')

test_dict = {'Test Accuracy': test_acc,
             'Test Loss': test_loss}

with open(csv_test, 'w') as csv_file:
    writer = csv.writer(csv_file)
    for key, value in test_dict.items():
        writer.writerow([key,value])
```

0.1.11 Model Predictions

[25]:

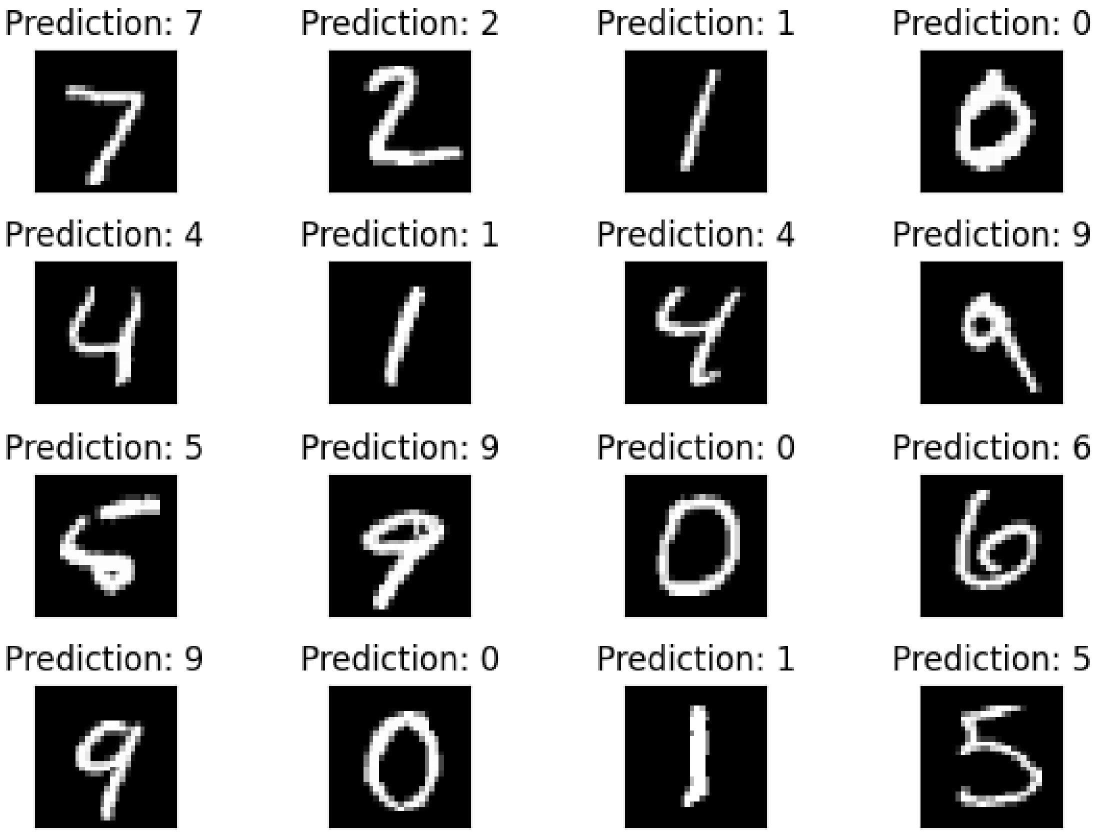
```
## Setup predictions from the model.
predict_test_labels = model.predict(test_images)
predict_classes = np.argmax(predict_test_labels, axis = 1)
predict_prob = np.max(predict_test_labels, axis = 1)
```

```
C:\Users\jkmey\anaconda3\envs\dsc650\lib\site-
packages\keras\engine\training_v1.py:2357: UserWarning: `Model.state_updates`
will be removed in a future version. This property should not be used in
TensorFlow 2.0, as `updates` are applied automatically.
  updates=self.state_updates,
```

[26]:

```
## Show an example predictions for the model.
fig = plt.figure()
for i in range(16):
    plt.subplot(4,4,i+1)
    plt.tight_layout()
    plt.imshow(test_images[i], cmap = 'gray', interpolation='none')
    plt.title("Prediction: {}".format(predict_classes[i]))
    plt.xticks([])
    plt.yticks([])
img_file = results_dir.joinpath('assignment06-1_Prediction_Images_QTY_16.png')
plt.savefig(img_file)
print("16 Prediction Images and Labels")
plt.show()
```

```
16 Prediction Images and Labels
```


Prediction: 7
Prediction: 2
Prediction: 1
Prediction: 0
Prediction: 4
Prediction: 1
Prediction: 4
Prediction: 9
Prediction: 5
Prediction: 9
Prediction: 0
Prediction: 6
Prediction: 9
Prediction: 0
Prediction: 1
Prediction: 5