# assignment06-2b_MeyerJake

April 23, 2023

## 0.1 Assignment 6-2b

### 0.1.1 DSC 650

### 0.1.2 Jake Meyer

### 0.1.3 04/22/2023

Using section 5.2 in Deep Learning with Python as a guide, create a ConvNet model that classifies images CIFAR10 small images classification dataset. This time includes dropout and data-augmentation. 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 Using code from CIFAR-10 Photo Classification Dataset

[1]:
```python
## 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, shutil

## Import the necessary keras components for the data and CNN
from keras import layers, models
from keras.datasets import cifar10
from keras.utils import to_categorical, np_utils
from keras.models import Sequential, load_model
from keras.layers.core import Dense, Dropout, Activation
from keras.layers import Conv2D, MaxPooling2D, Dense, Flatten
from keras.optimizers import SGD
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]:
```
## 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 CIFAR10 Dataset

[4]:
```
## Load the dataset
(trainX, trainy), (testX, testy) = cifar10.load_data()
```

[5]:
```
## Understand the shape of the train and test datasets.
print('trainX: {}'.format(trainX.shape))
print('testX: {}'.format(testX.shape))
print('trainy: {}'.format(trainy.shape))
print('testy: {}'.format(testy.shape))
```

```
trainX: (50000, 32, 32, 3)
testX: (10000, 32, 32, 3)
trainy: (50000, 1)
testy: (10000, 1)
```

### 0.1.5 Show Training Images and Labels

[6]:
```
## Show the first 16 training images and labels for better understanding of the␣
↪data.
fig = plt.figure()
```

```python
for i in range(16):
    plt.subplot(4,4,i+1)
    plt.tight_layout()
    plt.imshow(trainX[i], cmap = 'gray', interpolation='none')
    plt.title("Classify: {}".format(trainy[i]))
    plt.xticks([])
    plt.yticks([])
img_file = results_dir.joinpath('assignment06-2b_Sample_Images_QTY_16.png')
plt.savefig(img_file)
print("First 16 Training Images and Labels")
plt.show()
```

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

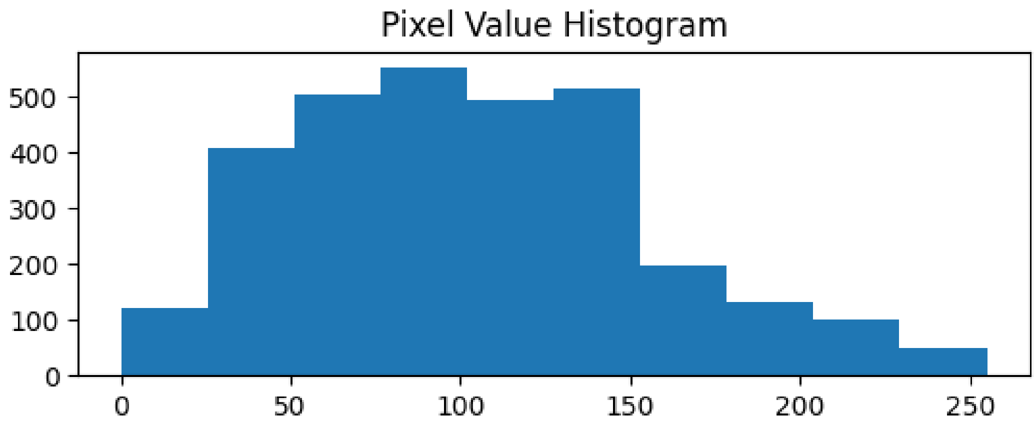

Referenced CIFAR10 for available classes.

[7]:
```python
## Define the classes for images within a list for the image dataset.
image_classes = ['airplane', 'automobile', 'bird', 'cat', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
```

### 0.1.6 Pixel Value Histogram

[12]:
```
## 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(trainX[0], cmap = 'gray', interpolation = 'none')
plt.title('Category: {}'.format(trainy[0]))
plt.xticks([])
plt.yticks([])
img_file = results_dir.joinpath('assignment06-2b_Digit_Overview.png')
plt.savefig(img_file)
plt.show()
```


[13]:
```
## Pixel distribution shown in the plot below for the image chosen in the␣
↪previous cell.
plt.subplot(2,1,2)
plt.hist(trainX[0].reshape(3072)) # Value needs to be 3072 for reshape,␣
↪otherwise error
plt.title("Pixel Value Histogram")
img_file = results_dir.joinpath('assignment06-2b_Pixel_Value_Histogram.png')
plt.savefig(img_file)
plt.show()
```




### 0.1.7 Prepare the Data

[14]:
```
## Normalize the training and test images.
train_images = trainX.astype('float32') / 255
test_images = testX.astype('float32') / 255

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

[15]:
```
## 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 ConvNet Model

[16]:
```
## Use the code from the textbook Github repository for section 5.2. Also,␣
 ↪remember the shape input shape (32,32,3)
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu',␣
 ↪kernel_initializer='he_uniform', padding='same', input_shape=(32, 32, 3)))
model.add(Conv2D(32, (3, 3), activation='relu',␣
 ↪kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu',␣
 ↪kernel_initializer='he_uniform', padding='same'))
model.add(Conv2D(64, (3, 3), activation='relu',␣
 ↪kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2)))
```

```
model.add(Conv2D(128, (3, 3), activation='relu',␣
 ↪kernel_initializer='he_uniform', padding='same'))
model.add(Conv2D(128, (3, 3), activation='relu',␣
 ↪kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
## Per the assignment add model.add(Dropout(0.2, input_shape=(60,)))
model.add(Dropout(0.1))
model.add(Dense(10, activation='softmax'))

## Compile the Model. Choosing categorical crossentropy as loss and accuracy as␣
 ↪metric.
## Also, define an optimizer with a learning rate of 0.001 and momentum of 0.9.
opt = SGD(learning_rate=0.001, momentum=0.9)
model.compile(optimizer=opt, loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
```

[17]:
```
## Show a summary of the model.
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        896

 conv2d_1 (Conv2D)           (None, 32, 32, 32)        9248

 max_pooling2d (MaxPooling2D  (None, 16, 16, 32)       0
 )

 conv2d_2 (Conv2D)           (None, 16, 16, 64)        18496

 conv2d_3 (Conv2D)           (None, 16, 16, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 64)         0
 2D)

 conv2d_4 (Conv2D)           (None, 8, 8, 128)         73856

 conv2d_5 (Conv2D)           (None, 8, 8, 128)         147584

 max_pooling2d_2 (MaxPooling  (None, 4, 4, 128)        0
 2D)

 flatten (Flatten)           (None, 2048)              0
```

```
 dense (Dense)               (None, 128)              262272


 dropout (Dropout)           (None, 128)              0


 dense_1 (Dense)             (None, 10)               1290

=================================================================
Total params: 550,570
Trainable params: 550,570
Non-trainable params: 0
_________________________________________________________________
```

[19]:
```
## Create data generator. Code help from machine learning mastery site listed␣
↪in introduction section.
datagen = keras.preprocessing.image.ImageDataGenerator(width_shift_range=0.1,␣
↪height_shift_range=0.1, horizontal_flip = True)
```

[22]:
```
## Prepare the Iterator. Code help from machine learning mastery site listed in␣
↪introduction section.
it_train = datagen.flow(train_images, train_labels, batch_size = 64)

## Number of epoxh steps to fit the model for training.
steps = int(train_images.shape[0]/64)
```

### 0.1.9 Train the Model

[29]:
```
## Train the model and store the results in the variable history. Code help␣
↪from machine learning mastery site.
history = model.fit(it_train, steps_per_epoch = steps, epochs=75, verbose = 1,
                         validation_data = (train_images_val, train_labels_val))
```

```
625/625 [==============================] - 721s 1s/step - batch: 312.0000 -
size: 64.0000 - loss: 0.5463 - acc: 0.8087 - val_loss: 0.5884 - val_acc: 0.7969
Epoch 36/75
625/625 [==============================] - 64s 102ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.5467 - acc: 0.8069 - val_loss: 0.5994 - val_acc: 0.7997
Epoch 37/75
625/625 [==============================] - 64s 102ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.5311 - acc: 0.8146 - val_loss: 0.5992 - val_acc: 0.7961
Epoch 38/75
625/625 [==============================] - 63s 101ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.5272 - acc: 0.8134 - val_loss: 0.5694 - val_acc: 0.8073
Epoch 39/75
625/625 [==============================] - 63s 101ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.5220 - acc: 0.8157 - val_loss: 0.5693 - val_acc: 0.8050
Epoch 40/75
625/625 [==============================] - 63s 100ms/step - batch: 312.0000 -
```

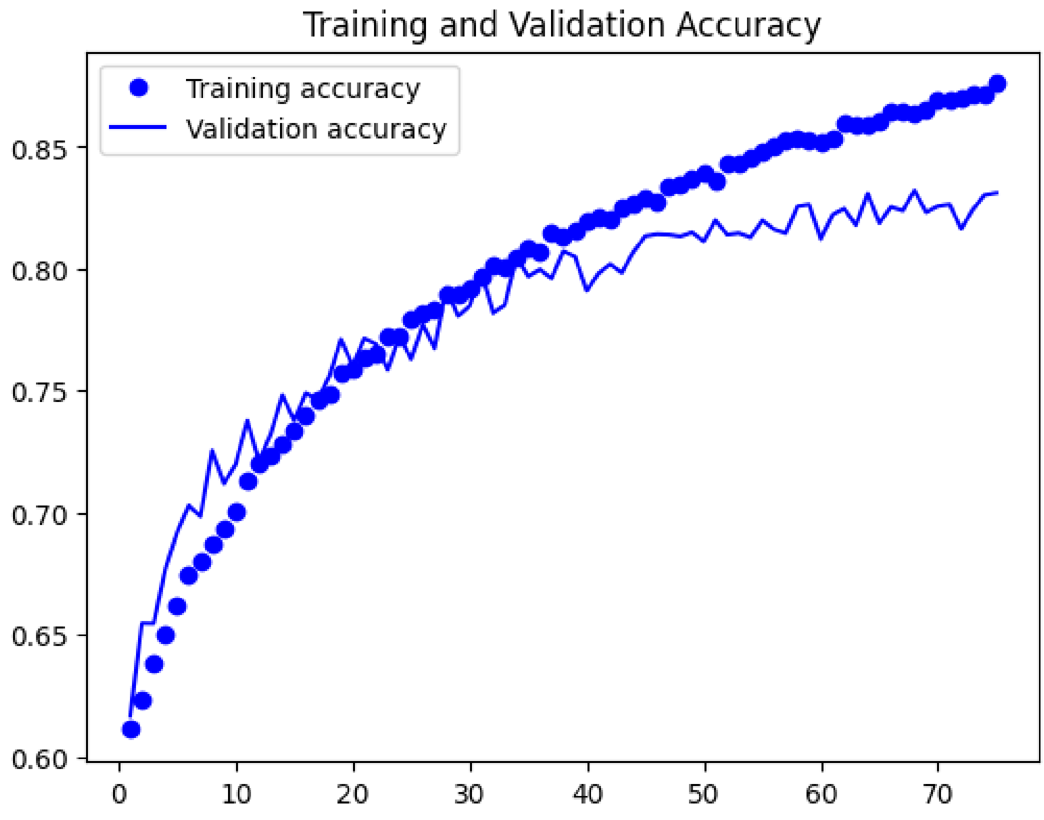
```
size: 64.0000 - loss: 0.5152 - acc: 0.8194 - val_loss: 0.6214 - val_acc: 0.7910
Epoch 41/75
625/625 [==============================] - 63s 101ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.5091 - acc: 0.8213 - val_loss: 0.6015 - val_acc: 0.7981
Epoch 42/75
625/625 [==============================] - 62s 100ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.5077 - acc: 0.8205 - val_loss: 0.5915 - val_acc: 0.8019
Epoch 43/75
625/625 [==============================] - 62s 100ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.5023 - acc: 0.8253 - val_loss: 0.6099 - val_acc: 0.7984
Epoch 44/75
625/625 [==============================] - 62s 100ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.4948 - acc: 0.8263 - val_loss: 0.5717 - val_acc: 0.8071
Epoch 45/75
625/625 [==============================] - 62s 100ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.4832 - acc: 0.8288 - val_loss: 0.5675 - val_acc: 0.8134
Epoch 46/75
625/625 [==============================] - 63s 100ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.4893 - acc: 0.8277 - val_loss: 0.5518 - val_acc: 0.8142
Epoch 47/75
625/625 [==============================] - 62s 100ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.4771 - acc: 0.8337 - val_loss: 0.5690 - val_acc: 0.8140
Epoch 48/75
625/625 [==============================] - 64s 103ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.4742 - acc: 0.8345 - val_loss: 0.5654 - val_acc: 0.8133
Epoch 49/75
625/625 [==============================] - 64s 102ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.4657 - acc: 0.8367 - val_loss: 0.5506 - val_acc: 0.8149
Epoch 50/75
625/625 [==============================] - 62s 99ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.4605 - acc: 0.8392 - val_loss: 0.5661 - val_acc: 0.8113
Epoch 51/75
625/625 [==============================] - 62s 99ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.4640 - acc: 0.8358 - val_loss: 0.5540 - val_acc: 0.8199
Epoch 52/75
625/625 [==============================] - 63s 102ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.4502 - acc: 0.8436 - val_loss: 0.5723 - val_acc: 0.8140
Epoch 53/75
625/625 [==============================] - 62s 99ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.4460 - acc: 0.8433 - val_loss: 0.5547 - val_acc: 0.8146
Epoch 54/75
625/625 [==============================] - 62s 99ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.4390 - acc: 0.8456 - val_loss: 0.5592 - val_acc: 0.8129
Epoch 55/75
625/625 [==============================] - 62s 99ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.4306 - acc: 0.8476 - val_loss: 0.5541 - val_acc: 0.8199
Epoch 56/75
625/625 [==============================] - 62s 99ms/step - batch: 312.0000 -
```

```
size: 64.0000 - loss: 0.4303 - acc: 0.8501 - val_loss: 0.5674 - val_acc: 0.8162
Epoch 57/75
625/625 [==============================] - 62s 99ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.4262 - acc: 0.8526 - val_loss: 0.5604 - val_acc: 0.8146
Epoch 58/75
625/625 [==============================] - 62s 100ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.4212 - acc: 0.8534 - val_loss: 0.5343 - val_acc: 0.8256
Epoch 59/75
625/625 [==============================] - 61s 98ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.4173 - acc: 0.8524 - val_loss: 0.5392 - val_acc: 0.8264
Epoch 60/75
625/625 [==============================] - 61s 98ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.4155 - acc: 0.8517 - val_loss: 0.5835 - val_acc: 0.8122
Epoch 61/75
625/625 [==============================] - 64s 102ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.4118 - acc: 0.8536 - val_loss: 0.5482 - val_acc: 0.8222
Epoch 62/75
625/625 [==============================] - 61s 98ms/step - batch: 312.0000 -
size: 64.0000 - loss: 0.3990 - acc: 0.8600 - val_loss: 0.5332 - val_acc: 0.8247
Epoch 63/75
131/625 [=====>…] - ETA: 45s - batch: 65.0000 - size:
64.0000 - loss: 0.3913 - acc: 0.8646
```

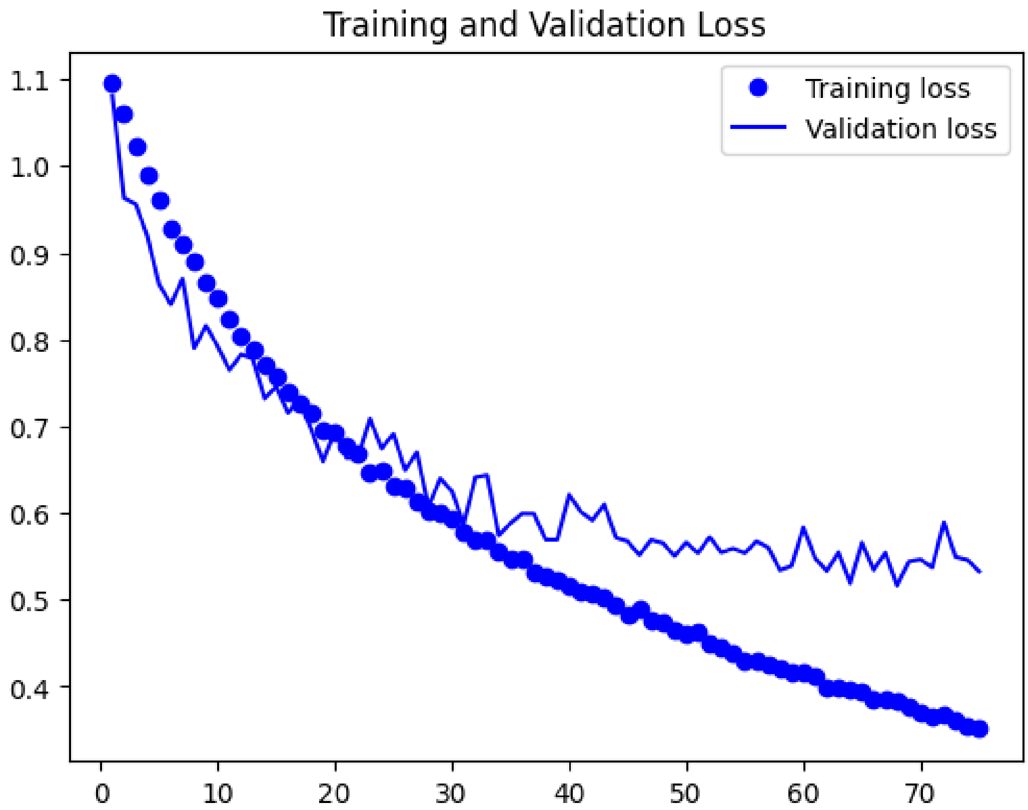
[33]:
```python
## Save the result model file to the results directory.
result_model_file = results_dir.joinpath('assignment06-2b_Model.h5')
model.save(result_model_file)
print("Saved the Trained model at %s " % result_model_file)
```

```
Saved the Trained model at C:\Users\jkmey\Documents\Github\DSC650_Course_Assignm
ents\dsc650\dsc650\assignments\assignment06\results\assignment06-2b_Model.h5
```

[34]:
```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('assignment06-2b_Training_and_Validation_Accuracy_Plot.png')
plt.savefig(img_file)
plt.show()
```




[35]:
```
## 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('assignment06-2b_Training_and_Validation_Loss_Plot.png')
plt.savefig(img_file)
plt.show()
```




#### 0.1.10 CNN Results on Test Data

[36]:
```
## Evaluate the model on the test subsets. Code from the textbook repository.
test_loss, test_acc = model.evaluate(test_images, test_labels)
```

[37]:
```
## 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: 82.05000162124634%
Test Loss: 0.5744333950042725
```

[41]:
```
## Write the Test Accuracy and Loss to the results folder.
csv_test = results_dir.joinpath('assignment06-2b_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)
```

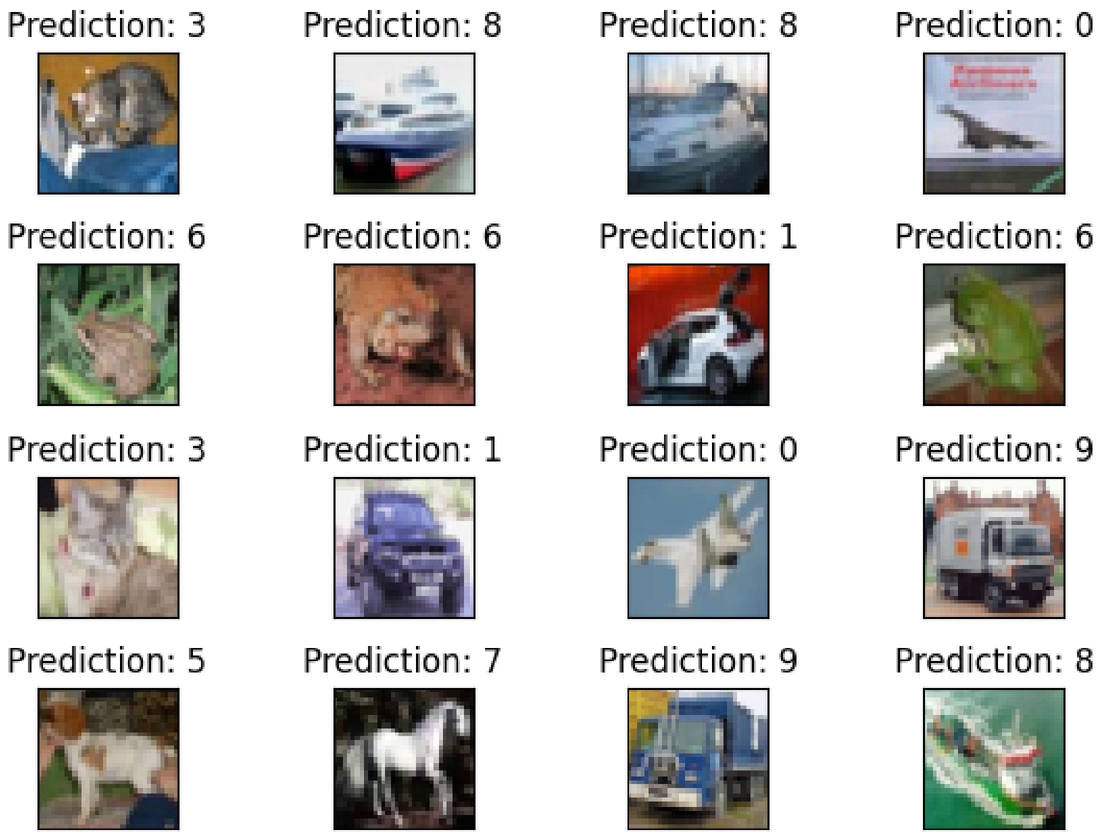
```
    for key, value in test_dict.items():
        writer.writerow([key,value])
```

### 0.1.11 Model Predictions

[42]:
```
## 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)
```

[43]:
```
## 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('assignment06-2b_Prediction_Images_QTY_16.png')
plt.savefig(img_file)
print("16 Prediction Images and Labels")
plt.show()
```

```
16 Prediction Images and Labels
```


Prediction: 3
Prediction: 8
Prediction: 8
Prediction: 0
Prediction: 6
Prediction: 6
Prediction: 1
Prediction: 6
Prediction: 3
Prediction: 1
Prediction: 0
Prediction: 9
Prediction: 5
Prediction: 7
Prediction: 9
Prediction: 8