DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

Assignment 06 Code and Outputs

For Assignment 6, I executed the code locally and had the following outputs with the example code. The chapter code for the CNN assignment was referenced through the Deep Learning with Python Github Link: deep-learning-with-python-notebooks/first_edition at master · fchollet/deep-learning-with-python-notebooks (github.com).

Assignment 6.1 Code and Outputs:

Assignment 6-1

DSC 650

Jake Meyer

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

In [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 depr
ecated and will be removed in a future version.
Instructions for updating:
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

non-resource variables are not supported in the long term

In [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

In [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)
```

In [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)
```

Import the MSNT Dataset

In [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()
```

In [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,)

Show Training Images and Labels

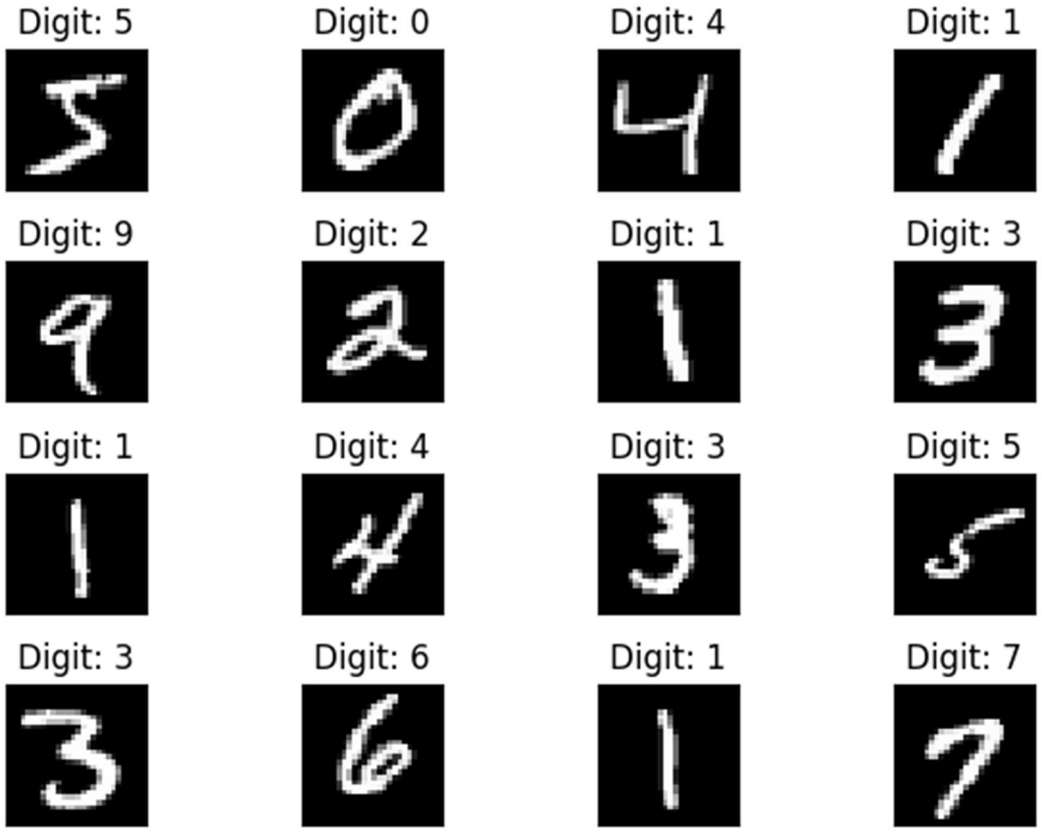
In [7]:

```
## Show the first 16 training images and labels for better understanding of the data.
fig = plt.figure()
for i in range(16):
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

```
    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



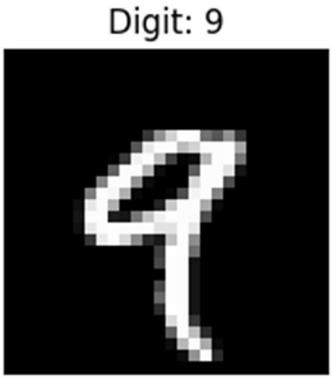
Pixel Value Histogram

In [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)
```

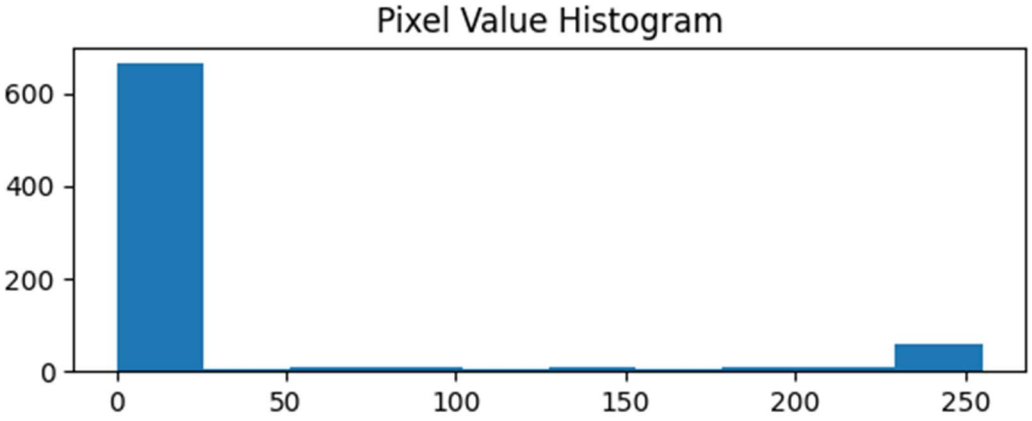
DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

```
plt.show()
```



In [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()
```



Prepare the Data

In [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.
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

In [11]:

```
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

In [12]:

```
## 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:]
```

Create the CNN Model

In [13]:

```
## 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'])
```

In [14]:

```
## 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
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

dense_1 (Dense) (None, 10) 650

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

Train the Model

In [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 nu
mber 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: UserWarni
ng: `Model.state_updates` will be removed in a future version. This property should not be used in TensorFlo
w 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/sampl
e
Epoch 2/20
50000/50000 - 12s - loss: 0.0605 - acc: 0.9811 - val_loss: 0.0536 - val_acc: 0.9853 - 12s/epoch - 234us/sampl
e
Epoch 3/20
50000/50000 - 12s - loss: 0.0402 - acc: 0.9872 - val_loss: 0.0525 - val_acc: 0.9834 - 12s/epoch - 235us/sampl
e
Epoch 4/20
50000/50000 - 11s - loss: 0.0302 - acc: 0.9905 - val_loss: 0.0403 - val_acc: 0.9888 - 11s/epoch - 216us/sampl
e
Epoch 5/20
50000/50000 - 11s - loss: 0.0235 - acc: 0.9923 - val_loss: 0.0360 - val_acc: 0.9900 - 11s/epoch - 227us/sampl
e
Epoch 6/20
50000/50000 - 11s - loss: 0.0182 - acc: 0.9943 - val_loss: 0.0367 - val_acc: 0.9904 - 11s/epoch - 216us/sampl
e
Epoch 7/20
50000/50000 - 11s - loss: 0.0158 - acc: 0.9951 - val_loss: 0.0360 - val_acc: 0.9903 - 11s/epoch - 216us/sampl
e
Epoch 8/20
50000/50000 - 11s - loss: 0.0125 - acc: 0.9964 - val_loss: 0.0430 - val_acc: 0.9889 - 11s/epoch - 216us/sampl
e
Epoch 9/20
50000/50000 - 11s - loss: 0.0099 - acc: 0.9970 - val_loss: 0.0359 - val_acc: 0.9908 - 11s/epoch - 221us/sampl
e
Epoch 10/20
50000/50000 - 11s - loss: 0.0084 - acc: 0.9974 - val_loss: 0.0498 - val_acc: 0.9890 - 11s/epoch - 217us/sampl
e
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

```
Epoch 11/20
50000/50000 - 11s - loss: 0.0068 - acc: 0.9977 - val_loss: 0.0442 - val_acc: 0.9906 - 11s/epoch - 215us/sampl
e
Epoch 12/20
50000/50000 - 11s - loss: 0.0060 - acc: 0.9979 - val_loss: 0.0400 - val_acc: 0.9920 - 11s/epoch - 214us/sampl
e
Epoch 13/20
50000/50000 - 11s - loss: 0.0055 - acc: 0.9980 - val_loss: 0.0472 - val_acc: 0.9906 - 11s/epoch - 215us/sampl
e
Epoch 14/20
50000/50000 - 11s - loss: 0.0044 - acc: 0.9987 - val_loss: 0.0524 - val_acc: 0.9911 - 11s/epoch - 214us/sampl
e
Epoch 15/20
50000/50000 - 11s - loss: 0.0041 - acc: 0.9987 - val_loss: 0.0559 - val_acc: 0.9902 - 11s/epoch - 215us/sampl
e
Epoch 16/20
50000/50000 - 11s - loss: 0.0042 - acc: 0.9987 - val_loss: 0.0518 - val_acc: 0.9916 - 11s/epoch - 215us/sampl
e
Epoch 17/20
50000/50000 - 11s - loss: 0.0033 - acc: 0.9989 - val_loss: 0.0490 - val_acc: 0.9929 - 11s/epoch - 215us/sampl
e
Epoch 18/20
50000/50000 - 11s - loss: 0.0028 - acc: 0.9990 - val_loss: 0.0674 - val_acc: 0.9909 - 11s/epoch - 214us/sampl
e
Epoch 19/20
50000/50000 - 11s - loss: 0.0028 - acc: 0.9992 - val_loss: 0.0488 - val_acc: 0.9922 - 11s/epoch - 216us/sampl
e
Epoch 20/20
50000/50000 - 11s - loss: 0.0030 - acc: 0.9990 - val_loss: 0.0537 - val_acc: 0.9923 - 11s/epoch - 216us/sampl
e
```

In [16]:

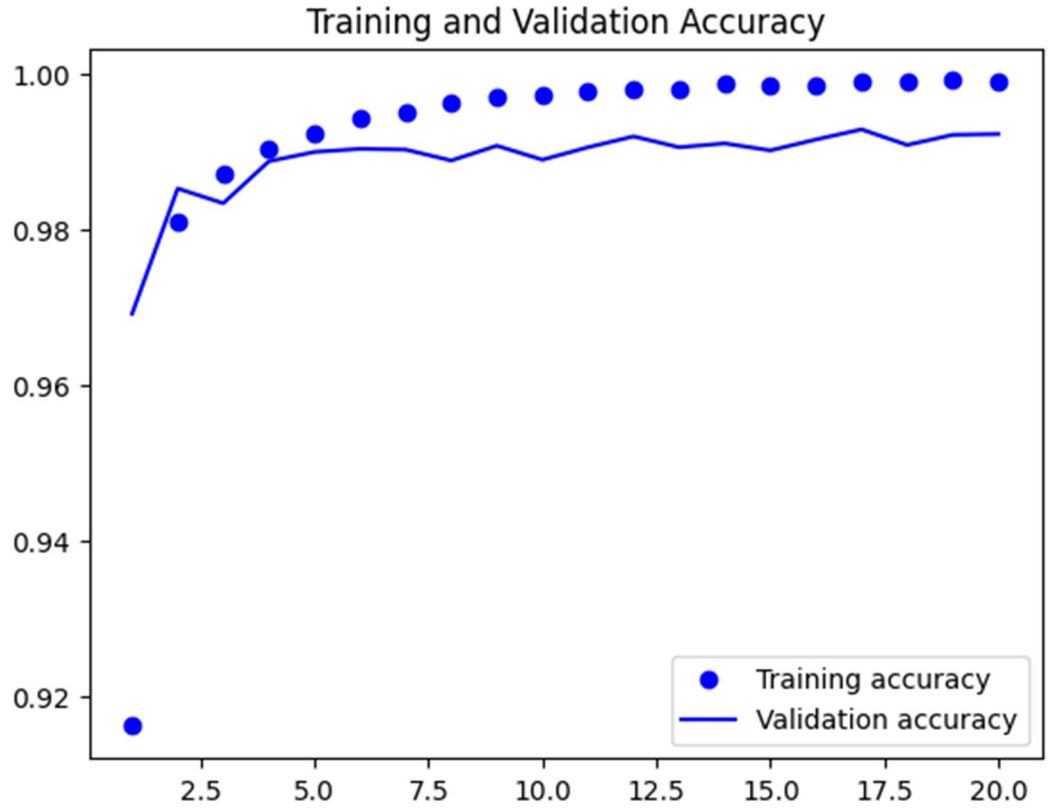
```
## Save the result model file to the results directory.
result_model_file = results_dir.joinpath('assignment06-1_Model.h5')
model.save(result_model_file)
print("Saved the Trained model at %s " % result_model_file)
```

```
Saved the Trained model at C:\Users\jkmey\Documents\Github\DSC650_Course_Assignments\dsc650\dsc6
50\assignments\assignment06\results\assignment06-1_Model.h5
```

In [22]:

```
## 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()
```

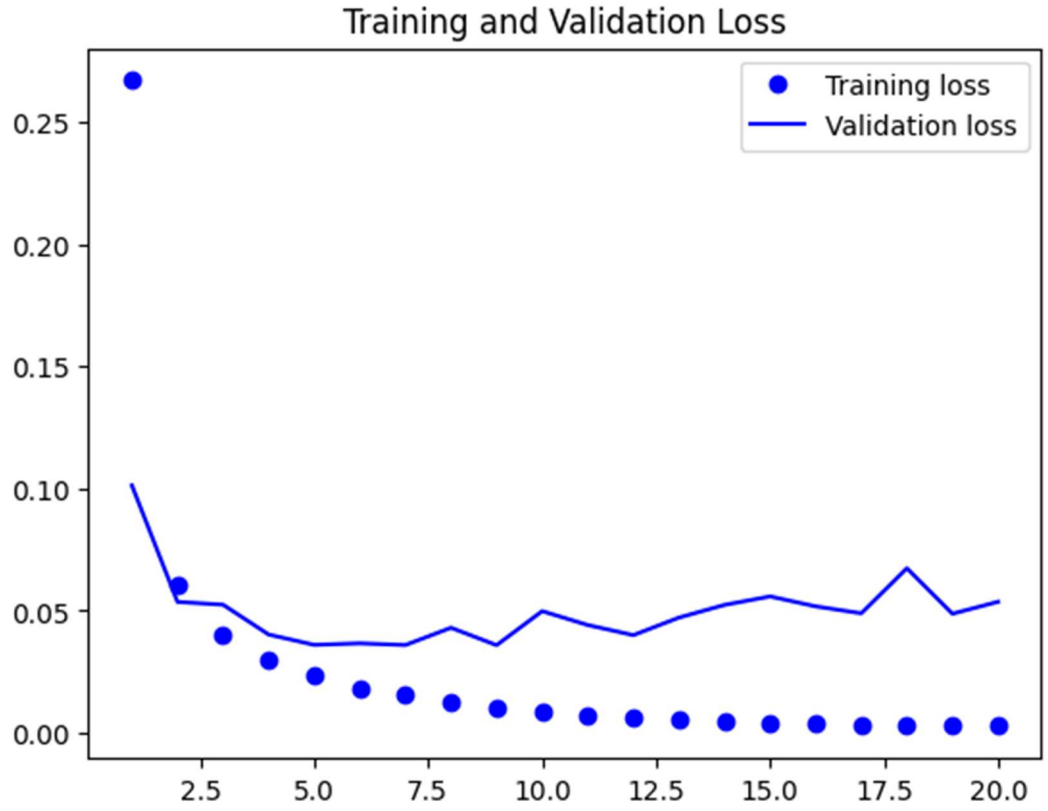
DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023



In [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()
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023



CNN Model Results on Test Data

In [19]:

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

In [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

In [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():
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

```
    writer.writerow([key,value])
```

Model Predictions

In [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)
```

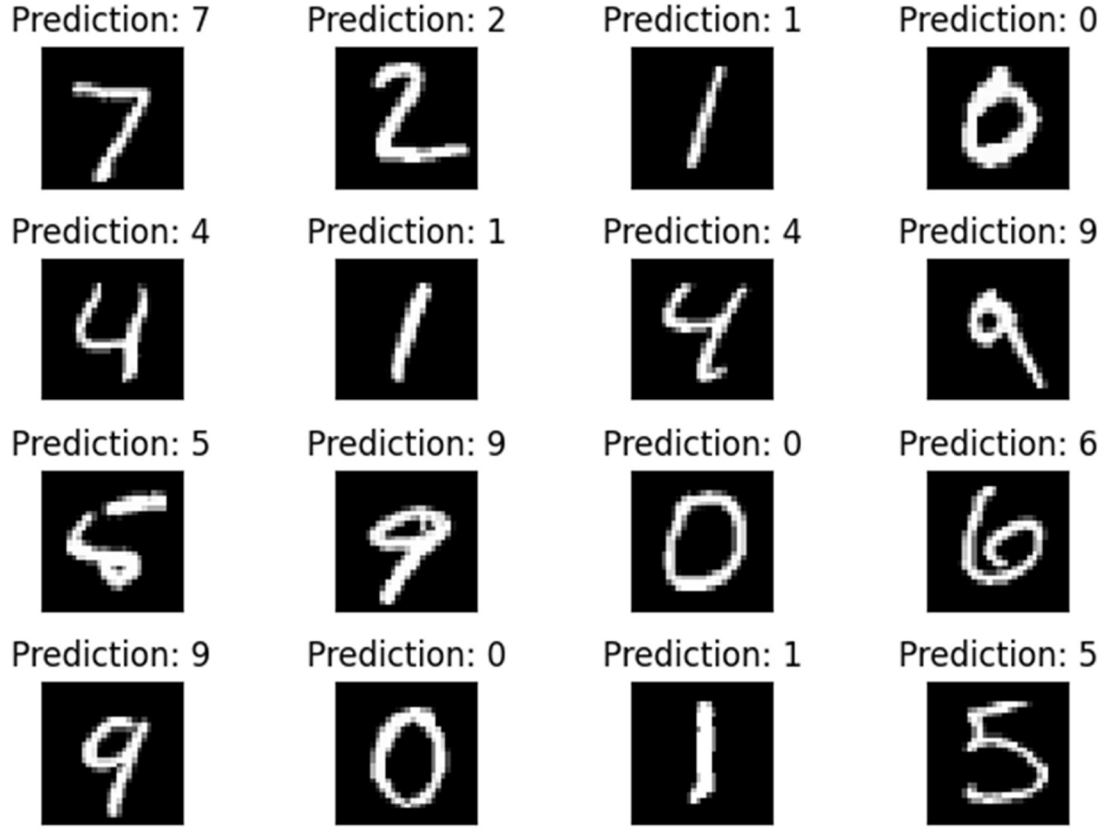
```
C:\Users\jkmey\anaconda3\envs\dsc650\lib\site-packages\keras\engine\training_v1.py:2357: UserWarni
ng: `Model.state_updates` will be removed in a future version. This property should not be used in TensorFlo
w 2.0, as `updates` are applied automatically.
  updates=self.state_updates,
```

In [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
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023



DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

Assignment 6.2a Code and Outputs:

Assignment 6-2a

DSC 650

Jake Meyer

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. Do not use dropout or data-augmentation in this part. 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

In [3]:

```
## 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 depr
ecated and will be removed in a future version.
Instructions for updating:
non-resource variables are not supported in the long term
```

In [4]:

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

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

```
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
```

In [5]:

```
## 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)
```

Import the CIFAR10 Dataset

In [6]:

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

In [7]:

```
## 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)
```

Show Training Images and Labels

In [8]:

```
## 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(trainX[i], cmap = 'gray', interpolation='none')
   plt.title("Classify: {}".format(trainy[i]))
   plt.xticks([])
   plt.yticks([])
img_file = results_dir.joinpath('assignment06-2a_Sample_Images_QTY_16.png')
plt.savefig(img_file)
print("First 16 Training Images and Labels")
plt.show()
```

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

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

Referenced CIFAR10 for available classes.

In [9]:

```
## 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']
```

Pixel Value Histogram

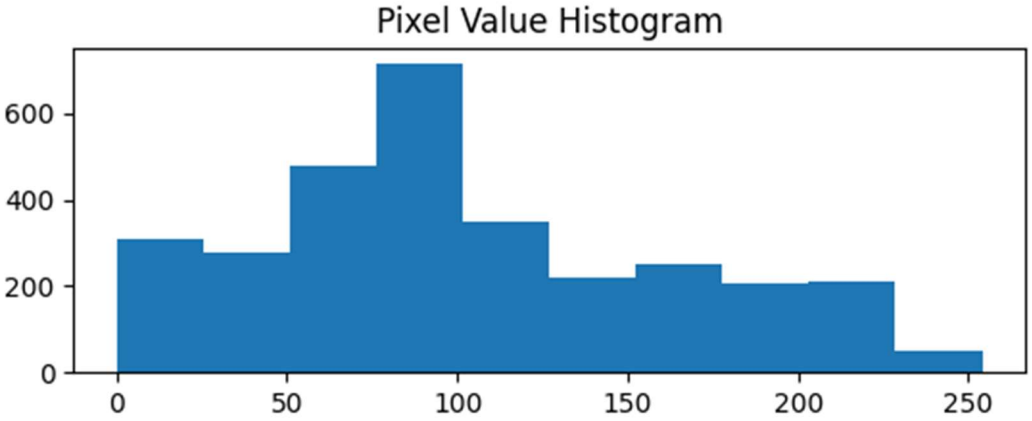
In [10]:

```
## 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[4], cmap = 'gray', interpolation = 'none')
plt.title('Category: {}'.format(trainy[4]))
plt.xticks([])
plt.yticks([])
img_file = results_dir.joinpath('assignment06-2a_Digit_Overview.png')
plt.savefig(img_file)
plt.show()
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

In [11]:

```
## Pixel distribution shown in the plot below for the image chosen in the previous cell.
plt.subplot(2,1,2)
plt.hist(trainX[4].reshape(3072)) # Value needs to be 3072 for reshape, otherwise error
plt.title("Pixel Value Histogram")
img_file = results_dir.joinpath('assignment06-2a_Pixel_Value_Histogram.png')
plt.savefig(img_file)
plt.show()
```



Prepare the Data

In [12]:

```
## 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)
```

In [13]:

```
## Split train_images and train_labels into train and validation subsets.
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

```
train_images_val = train_images[:10000]
train_images = train_images[10000:]
train_labels_val = train_labels[:10000]
train_labels = train_labels[10000:]
```

Create the ConvNet Model

In [14]:

```
## 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'))
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'])
```

In [15]:

```
## 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
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

conv2d_5 (Conv2D)	(None, 8, 8, 128)	147584
max_pooling2d_2 (MaxPooling 2D)	(None, 4, 4, 128)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 128)	262272
dense_1 (Dense)	(None, 10)	1290

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

Train the Model

In [16]:

```
## Train the model and store the results in the variable history.
history = model.fit(train_images, train_labels, epochs=20, batch_size=32, verbose = 1,
                    validation_data = (train_images_val, train_labels_val))
Train on 40000 samples, validate on 10000 samples
WARNING:tensorflow:OMP_NUM_THREADS is no longer used by the default Keras config. To configure the nu
mber of threads, use tf.config.threading APIs.
Epoch 1/20
40000/40000 [==============================] - ETA: 0s - loss: 1.6764 - acc: 0.3981
C:\Users\jkmey\anaconda3\envs\dsc650\lib\site-packages\keras\engine\training_v1.py:2333: UserWarni
ng: `Model.state_updates` will be removed in a future version. This property should not be used in TensorFlo
w 2.0, as `updates` are applied automatically.
  updates = self.state_updates
40000/40000 [==============================] - 75s 2ms/sample - loss: 1.6764 - acc: 0.3981 - val_los
s: 1.4222 - val_acc: 0.4860
Epoch 2/20
40000/40000 [==============================] - 74s 2ms/sample - loss: 1.2945 - acc: 0.5376 - val_los
s: 1.1595 - val_acc: 0.5926
Epoch 3/20
40000/40000 [==============================] - 75s 2ms/sample - loss: 1.1051 - acc: 0.6096 - val_los
s: 1.0206 - val_acc: 0.6448
Epoch 4/20
40000/40000 [==============================] - 75s 2ms/sample - loss: 0.9723 - acc: 0.6598 - val_los
s: 0.9429 - val_acc: 0.6678
Epoch 5/20
40000/40000 [==============================] - 73s 2ms/sample - loss: 0.8608 - acc: 0.6981 - val_los
s: 0.9123 - val_acc: 0.6813
Epoch 6/20
40000/40000 [==============================] - 69s 2ms/sample - loss: 0.7757 - acc: 0.7319 - val_los
s: 0.8571 - val_acc: 0.6960
Epoch 7/20
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

```
40000/40000 [==============================] - 70s 2ms/sample - loss: 0.6954 - acc: 0.7597 - val_los
s: 0.8162 - val_acc: 0.7134
Epoch 8/20
40000/40000 [==============================] - 76s 2ms/sample - loss: 0.6233 - acc: 0.7838 - val_los
s: 0.8270 - val_acc: 0.7147
Epoch 9/20
40000/40000 [==============================] - 75s 2ms/sample - loss: 0.5516 - acc: 0.8061 - val_los
s: 0.8151 - val_acc: 0.7261
Epoch 10/20
40000/40000 [==============================] - 70s 2ms/sample - loss: 0.4909 - acc: 0.8275 - val_los
s: 0.8760 - val_acc: 0.7085
Epoch 11/20
40000/40000 [==============================] - 72s 2ms/sample - loss: 0.4224 - acc: 0.8512 - val_los
s: 0.8633 - val_acc: 0.7323
Epoch 12/20
40000/40000 [==============================] - 70s 2ms/sample - loss: 0.3670 - acc: 0.8707 - val_los
s: 0.8991 - val_acc: 0.7184
Epoch 13/20
40000/40000 [==============================] - 72s 2ms/sample - loss: 0.3093 - acc: 0.8911 - val_los
s: 1.0054 - val_acc: 0.7067
Epoch 14/20
40000/40000 [==============================] - 73s 2ms/sample - loss: 0.2593 - acc: 0.9099 - val_los
s: 0.9878 - val_acc: 0.7169
Epoch 15/20
40000/40000 [==============================] - 78s 2ms/sample - loss: 0.2159 - acc: 0.9231 - val_los
s: 1.1270 - val_acc: 0.7107
Epoch 16/20
40000/40000 [==============================] - 79s 2ms/sample - loss: 0.1790 - acc: 0.9366 - val_los
s: 1.1179 - val_acc: 0.7231
Epoch 17/20
40000/40000 [==============================] - 74s 2ms/sample - loss: 0.1497 - acc: 0.9484 - val_los
s: 1.2099 - val_acc: 0.7205
Epoch 18/20
40000/40000 [==============================] - 78s 2ms/sample - loss: 0.1279 - acc: 0.9551 - val_los
s: 1.3091 - val_acc: 0.7018
Epoch 19/20
40000/40000 [==============================] - 73s 2ms/sample - loss: 0.1105 - acc: 0.9610 - val_los
s: 1.2970 - val_acc: 0.7260
Epoch 20/20
40000/40000 [==============================] - 79s 2ms/sample - loss: 0.1050 - acc: 0.9634 - val_los
s: 1.4012 - val_acc: 0.7208
```

In [17]:

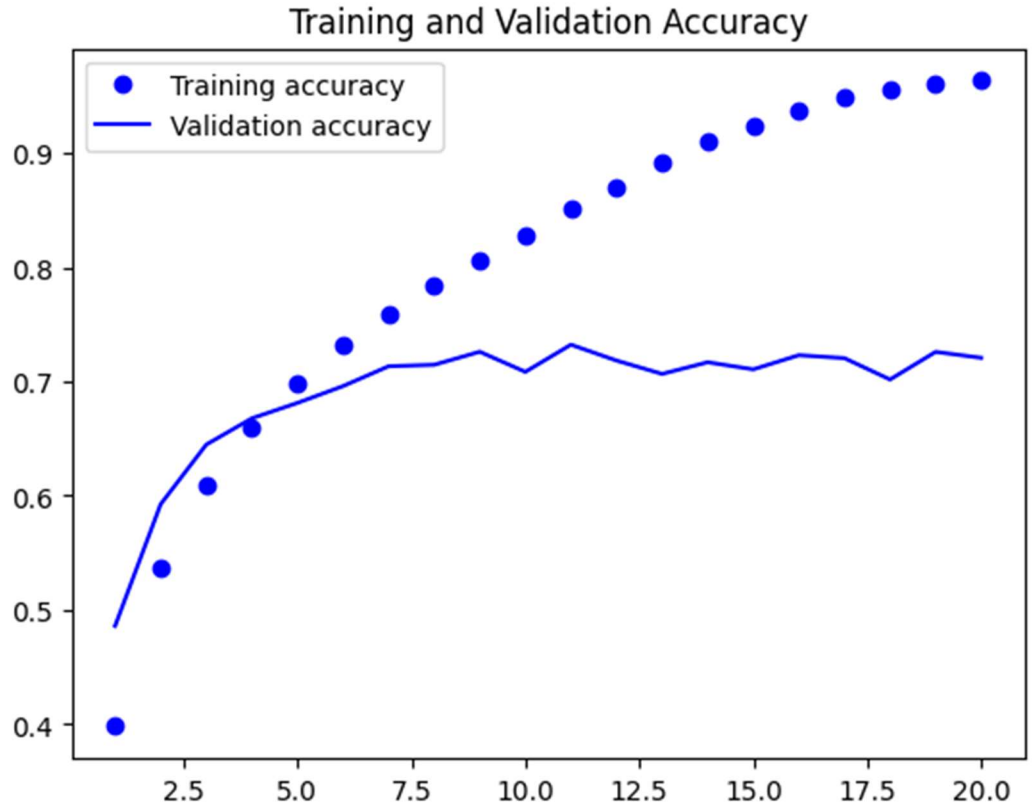
```
## Save the result model file to the results directory.
result_model_file = results_dir.joinpath('assignment06-2a_Model.h5')
model.save(result_model_file)
print("Saved the Trained model at %s " % result_model_file)
```

```
Saved the Trained model at C:\Users\jkmey\Documents\Github\DSC650_Course_Assignments\dsc650\dsc6
50\assignments\assignment06\results\assignment06-2a_Model.h5
```

In [18]:

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

```
## 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('assignment06-2a_Training_and_Validation_Accuracy_Plot.png')
plt.savefig(img_file)
plt.show()
```

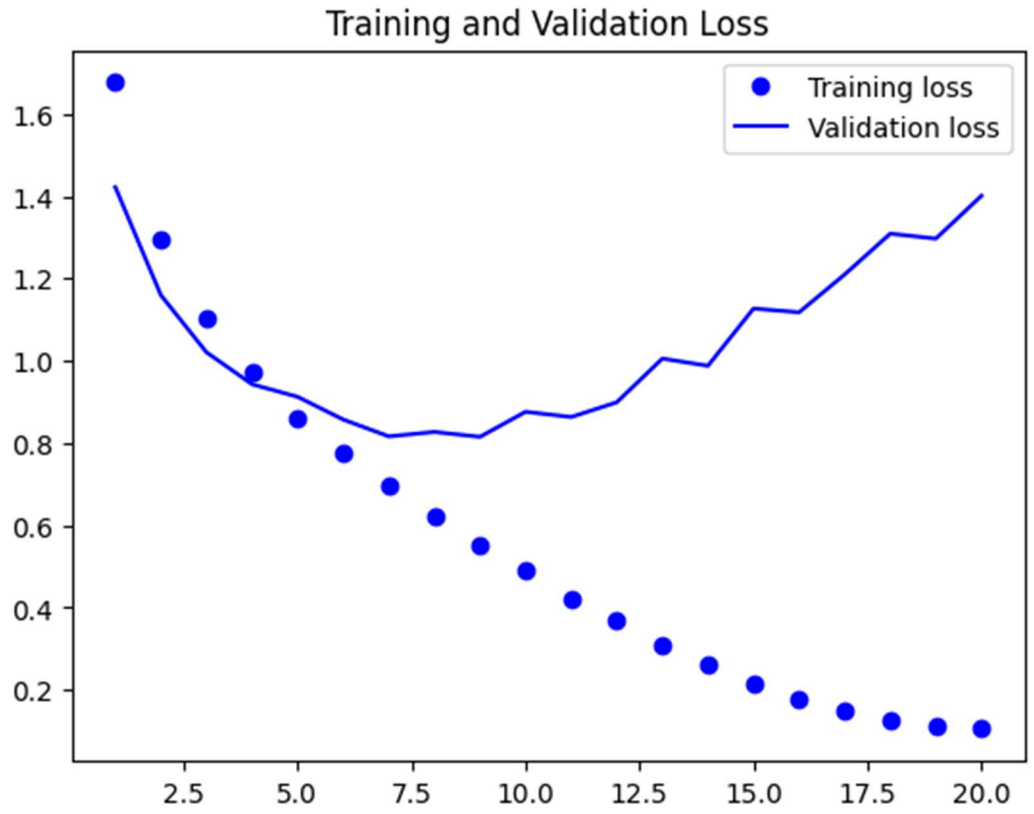


In [19]:

```
## 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('assignment06-2a_Training_and_Validation_Loss_Plot.png')
plt.savefig(img_file)
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

plt.show()



CNN Results on Test Data

In [20]:

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

In [21]:

```
## 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: 71.10999822616577%
Test Loss: 1.493364258670807

In [22]:

```
## Write the Test Accuracy and Loss to the results folder.
csv_test = results_dir.joinpath('assignment06-2a_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)
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

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

Model Predictions

In [23]:

```
## 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: UserWarni
ng: `Model.state_updates` will be removed in a future version. This property should not be used in TensorFlo
w 2.0, as `updates` are applied automatically.
  updates=self.state_updates,
```

In [24]:

```
## 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('assignment06-2a_Prediction_Images_QTY_16.png')
plt.savefig(img_file)
print("16 Prediction Images and Labels")
plt.show()
```

```
16 Prediction Images and Labels
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

Prediction: 5

Prediction: 8

Prediction: 8

Prediction: 8

Prediction: 6

Prediction: 6

Prediction: 3

Prediction: 2

Prediction: 4

Prediction: 1

Prediction: 4

Prediction: 9

Prediction: 5

Prediction: 7

Prediction: 9

Prediction: 8

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

Assignment 6.2b Code and Output:

Assignment 6-2b

DSC 650

Jake Meyer

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

In [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, 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 depr
ecated and will be removed in a future version.
Instructions for updating:
non-resource variables are not supported in the long term
```

In [2]:

```
## Print versions of essential packages
print("keras version: {}".format(keras.__version__))
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

```
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
```

In [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)
```

Import the CIFAR10 Dataset

In [4]:

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

In [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)
```

Show Training Images and Labels

In [6]:

```
## 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(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
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

Referenced CIFAR10 for available classes.

In [7]:

```
## 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']
```

Pixel Value Histogram

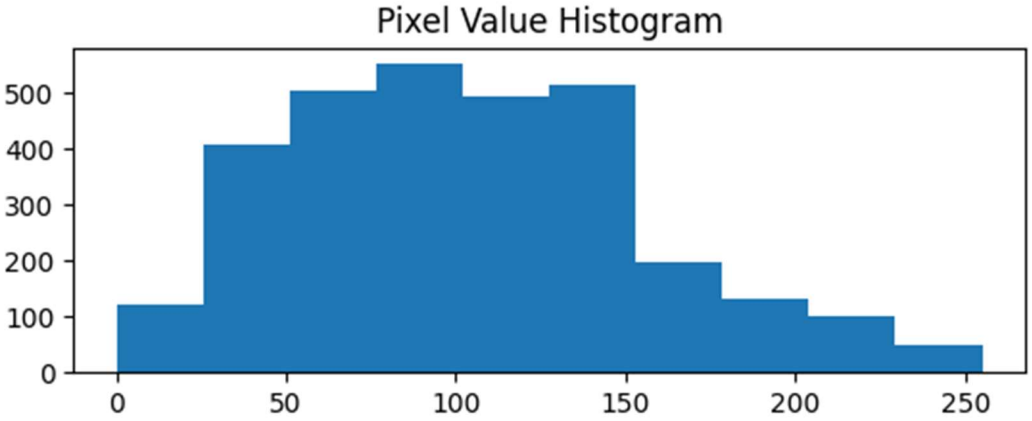
In [12]:

```
## 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()
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

In [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()
```



Prepare the Data

In [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)
```

In [15]:

```
## Split train_images and train_labels into train and validation subsets.
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

```
train_images_val = train_images[:10000]
train_images = train_images[10000:]
train_labels_val = train_labels[:10000]
train_labels = train_labels[10000:]
```

Create the ConvNet Model

In [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'])
```

In [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)
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

```
 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
_________________________________________________________________
```

In [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)
```

In [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)
```

Train the Model

In [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: 0.54
63 - 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
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

```
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
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

```
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
```

In [33]:

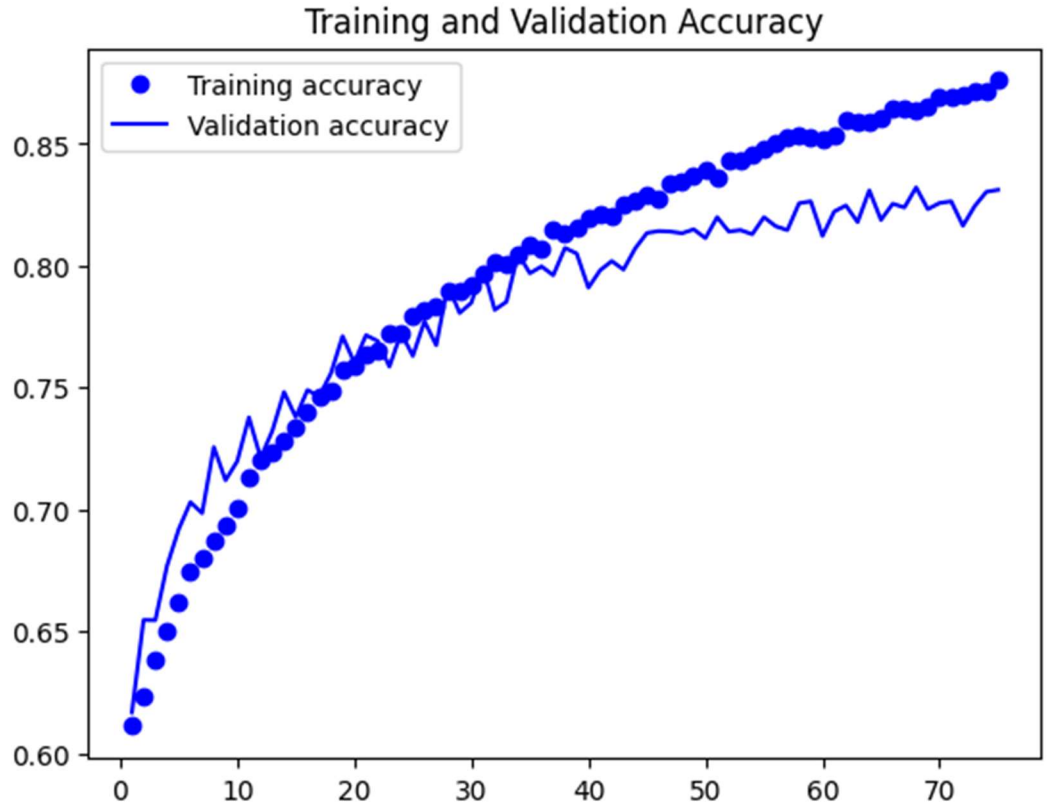
```
## Save the result model file to the results directory.
result_model_file = results_dir.joinpath('assignment06-2b_Model.h5')
model.save(result_model_file)
print("Saved the Trained model at %s " % result_model_file)
```

```
Saved the Trained model at C:\Users\jkmey\Documents\Github\DSC650_Course_Assignments\dsc650\dsc6
50\assignments\assignment06\results\assignment06-2b_Model.h5
```

In [34]:

```
## 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()
```

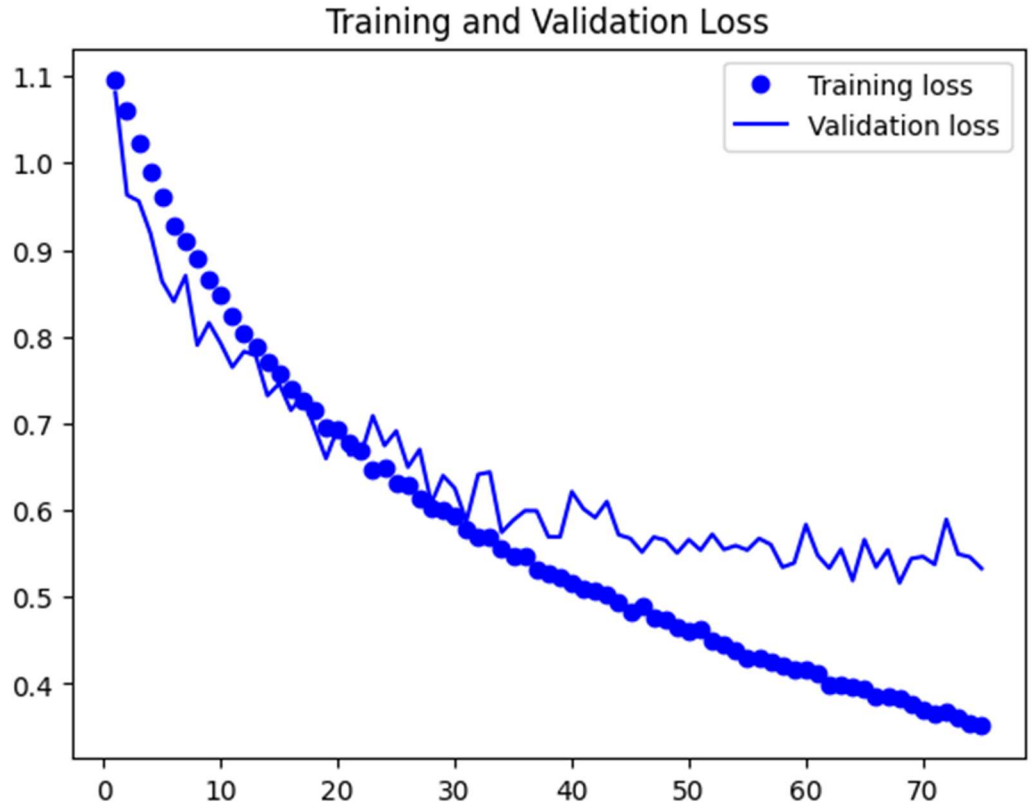
DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023



In [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()
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023



CNN Results on Test Data

In [36]:

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

In [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

In [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():
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

```
    writer.writerow([key,value])
```

Model Predictions

In [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)
```

In [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

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

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


DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

Assignment 6.3 Code and Output:

Assignment 6-3

DSC 650

Jake Meyer

04/23/2023

Load the ResNet50 model. Perform image classification on five to ten images of your choice. They can be personal images or publically available images. Include the images in dsc650/assignments/assignment06/images/. Save the predictions dsc650/assignments/assignment06/results/predictions/resnet50 directory. If you are using JupyterHub, you can include those plots in your Jupyter notebook.

Using code from deep-learning-with-python-notebooks
Using ResNet50 function api site

In [1]:

```
## Import the necessary modules for the assignment above.
import csv
import cv2
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import keras
import sklearn
from pathlib import Path
import time
import os

## Import the necessary keras components for the data and CNN
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.imagenet_utils import decode_predictions
from tensorflow.keras.applications.imagenet_utils import preprocess_input
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications import resnet50
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 depr
ecated and will be removed in a future version.
Instructions for updating:
non-resource variables are not supported in the long term
```

In [2]:

```
## Print versions of essential packages
print("keras version: {}".format(keras.__version__))
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

```
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
```

In [3]:

```
## Setup the directories for the assignment
current_dir =
Path('C:/Users/jkmey/Documents/Github/DSC650_Course_Assignments/dsc650/dsc650/assignments/assig
nment06')
image_dir =
Path('C:/Users/jkmey/Documents/Github/DSC650_Course_Assignments/dsc650/dsc650/assignments/assig
nment06/images')
results_dir =
Path('C:/Users/jkmey/Documents/Github/DSC650_Course_Assignments/dsc650/dsc650/assignments/assig
nment06/').joinpath('results')
predictions_dir =
Path('C:/Users/jkmey/Documents/Github/DSC650_Course_Assignments/dsc650/dsc650/assignments/assig
nment06/results').joinpath('predictions')
resnet_dir =
Path('C:/Users/jkmey/Documents/Github/DSC650_Course_Assignments/dsc650/dsc650/assignments/assig
nment06/results/predictions').joinpath('resnet50')
resnet_dir.mkdir(parents = True, exist_ok = True)
```

Load the ResNet50 Model

In [4]:

```
## Load the ResNet50 model as shown in the ResNet50 site listed above.
model = ResNet50(weights = 'imagenet')
```

```
WARNING:tensorflow:From C:\Users\jkmey\anaconda3\envs\dsc650\lib\site-packages\keras\layers\norm
alization\batch_normalization.py:561: _colocate_with (from tensorflow.python.framework.ops) is deprecated
 and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:OMP_NUM_THREADS is no longer used by the default Keras config. To configure the nu
mber of threads, use tf.config.threading APIs.
```

Pulled 10 random images from the internet of various animals. Per the assignment, will write code to classify these images with ResNet50.

Using ResNet50 Model for Predictions on 10 Images

In [5]:

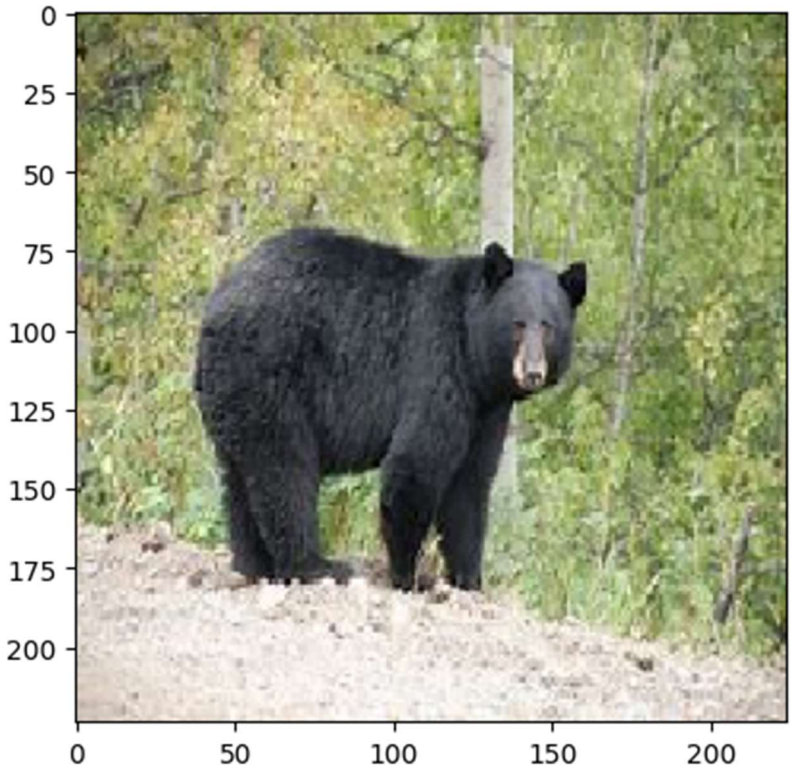
```
## Load and resize  (224x224 per the ResNet50 site) for the 10 images.
img1 = image.load_img('images/bear.png', target_size = (224, 224))
img2 = image.load_img('images/cat.png', target_size = (224, 224))
img3 = image.load_img('images/dog.png', target_size = (224, 224))
img4 = image.load_img('images/elephant.png', target_size = (224, 224))
img5 = image.load_img('images/giraffe.png', target_size = (224, 224))
img6 = image.load_img('images/gorilla.png', target_size = (224, 224))
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

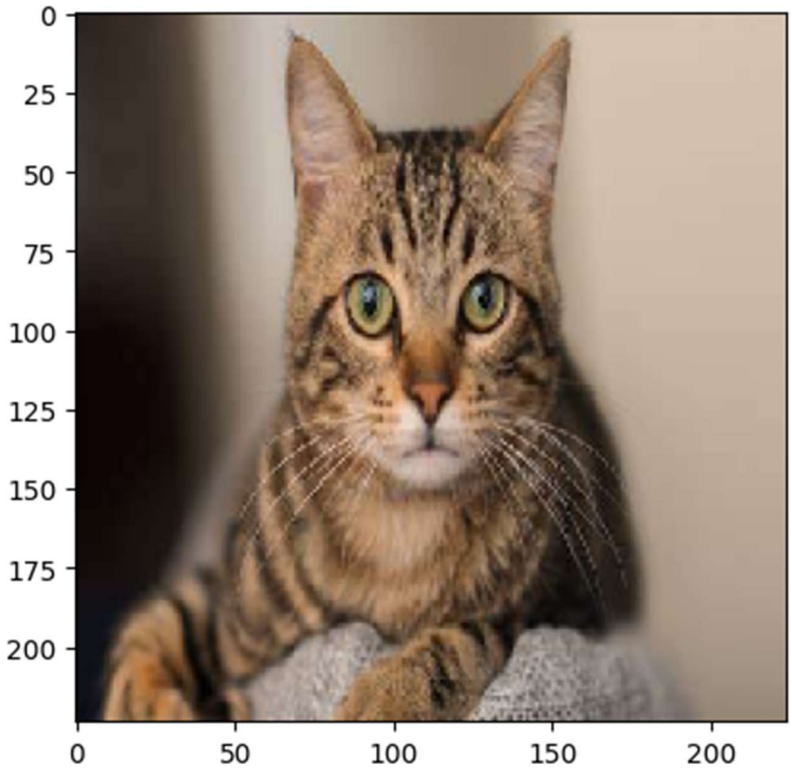
```
img7 = image.load_img('images/hippo.png', target_size = (224, 224))
img8 = image.load_img('images/leapord.png', target_size = (224, 224))
img9 = image.load_img('images/penguin.png', target_size = (224, 224))
img10 = image.load_img('images/tiger.png', target_size = (224, 224))
```

In [6]:

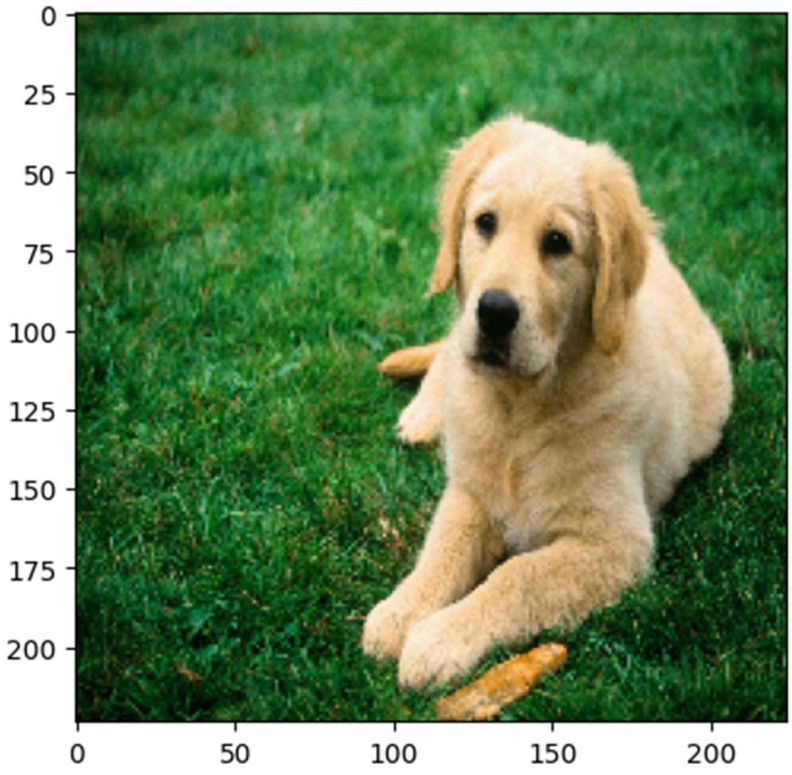
```
## Show the 10 images (after resizing) for reference prior to classifying.
image_list = [img1, img2, img3, img4, img5, img6, img7, img8, img9, img10]

for animal in image_list:
    plt.imshow(animal)
    plt.show()
```



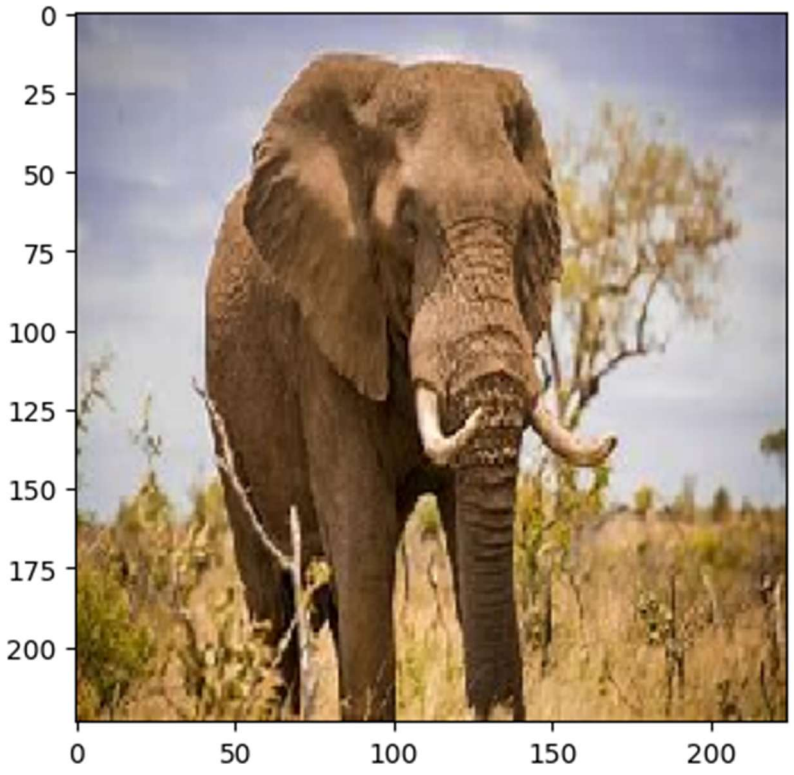
DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023



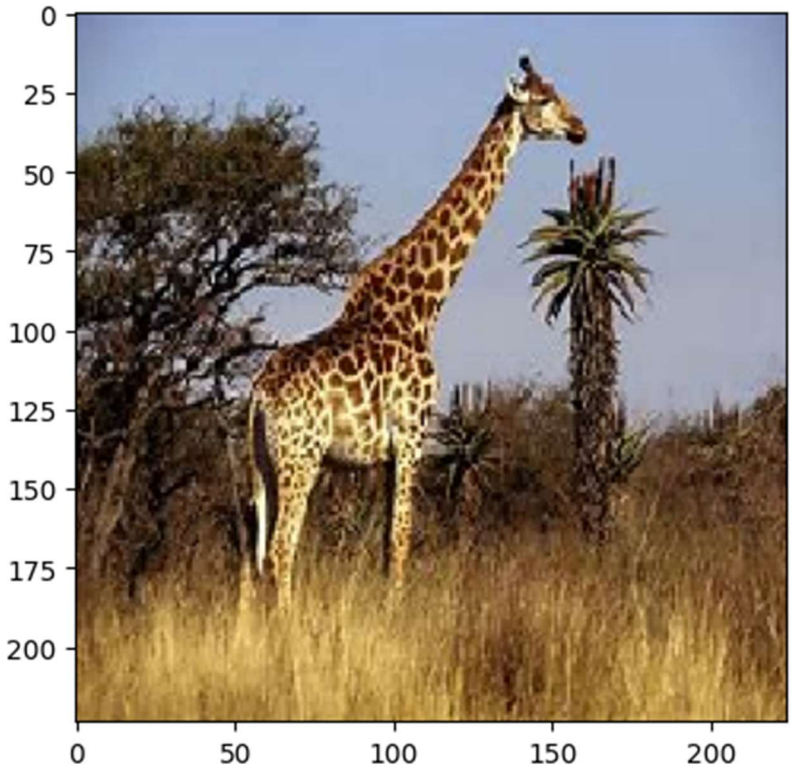
DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023



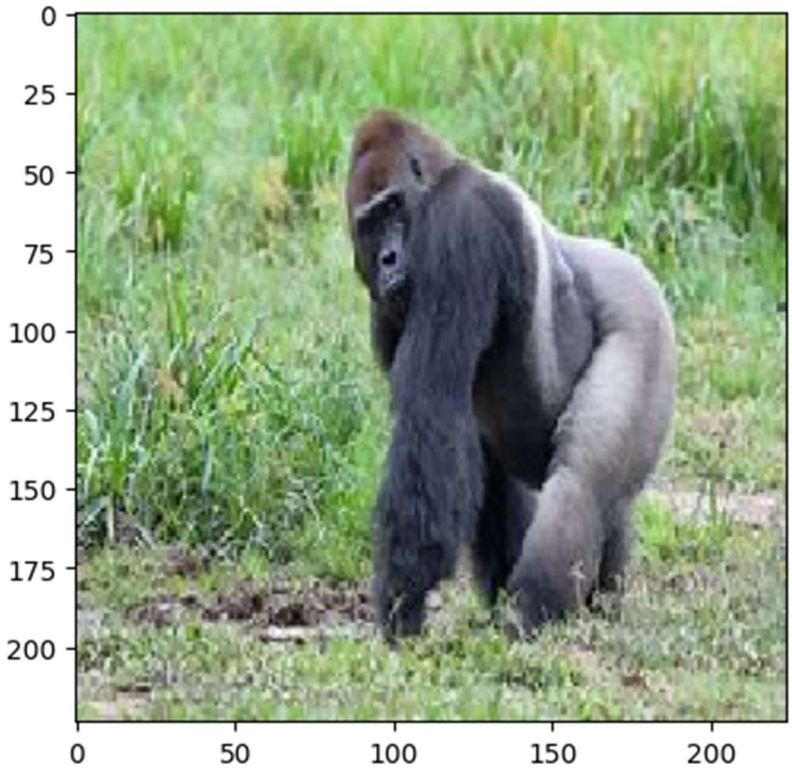
DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023



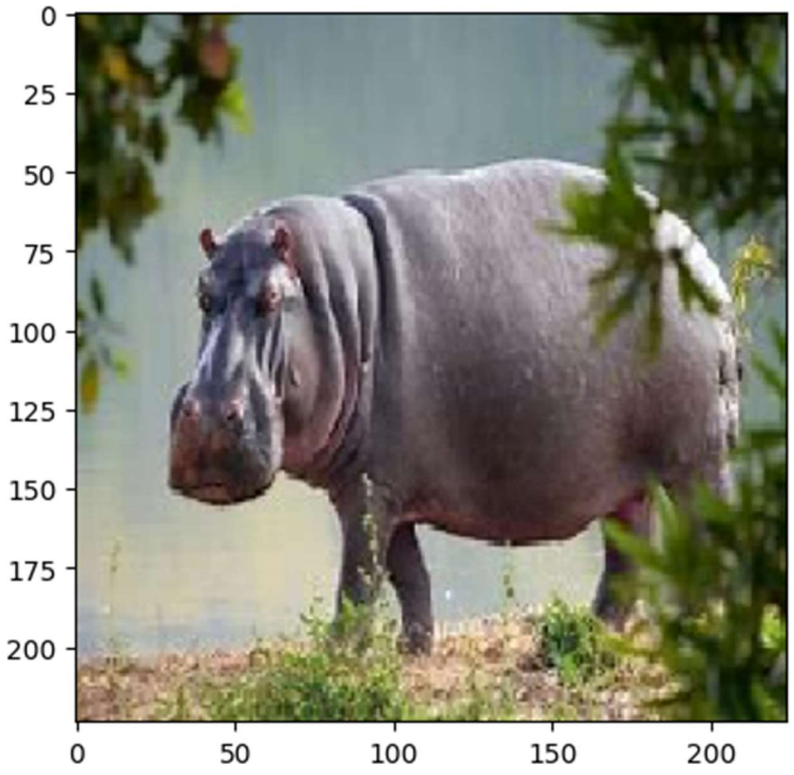
DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023



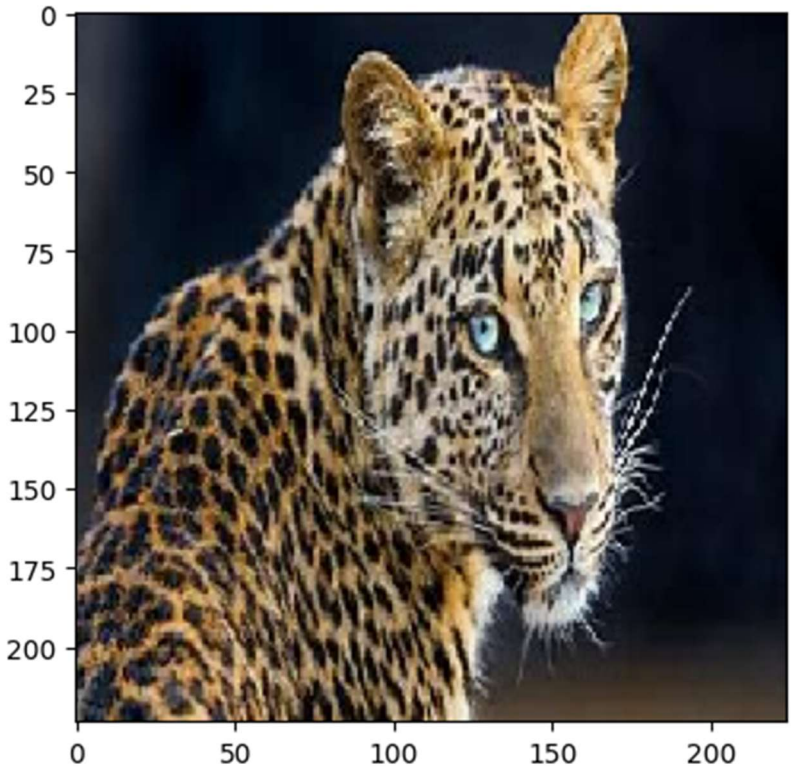
DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023



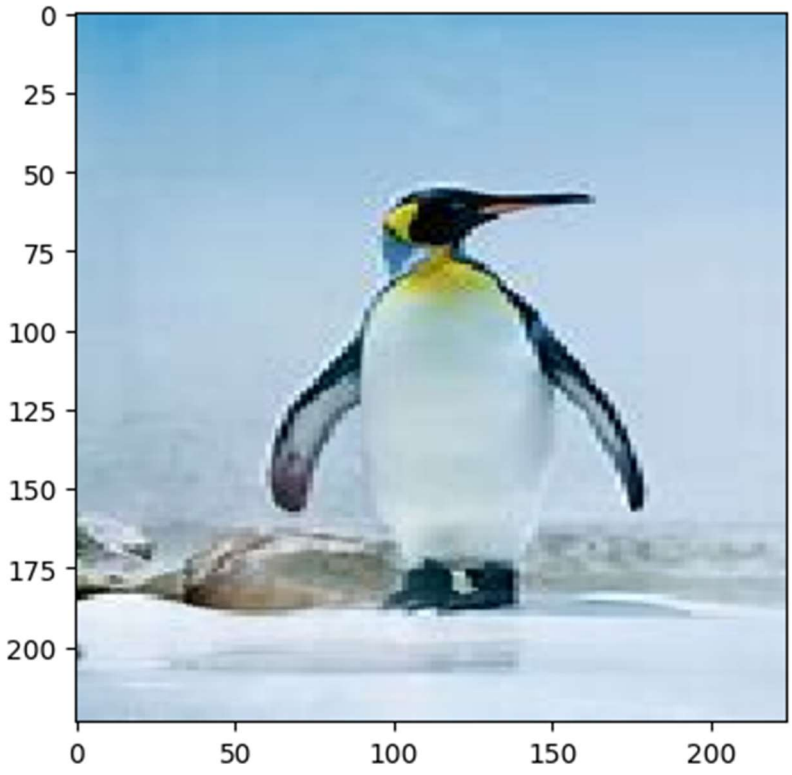
DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023



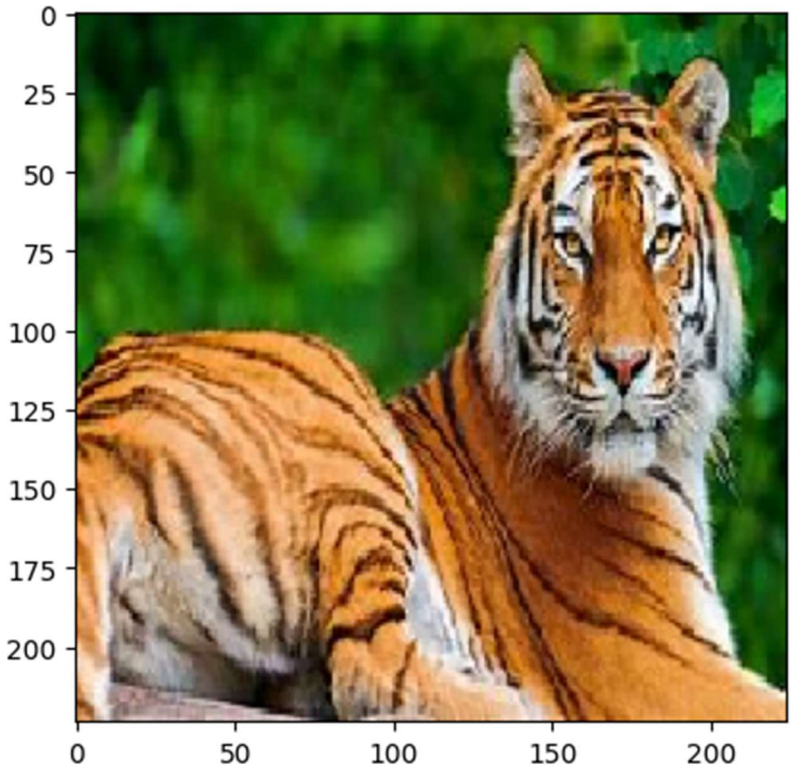
DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023



DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023



DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023



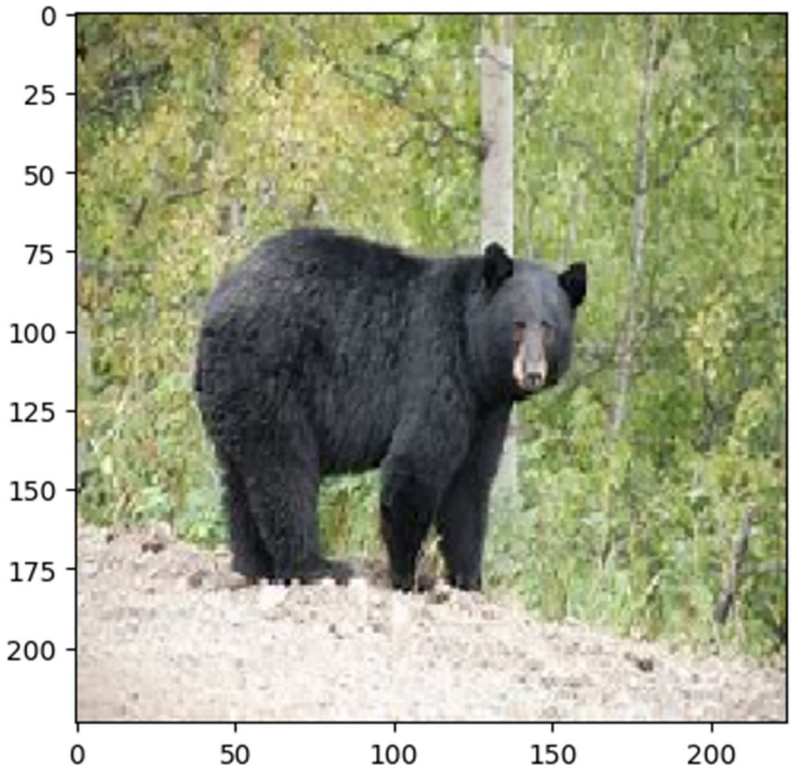
Create the predictions for each image using the ResNet50 model.

In [16]:

```
## Adjust the image. Convert to numpy array, add batch dimension, and preprocess.
image_array1 = image.img_to_array(img1)
image_array1 = np.expand_dims(image_array1, axis =0)
image_array1 = preprocess_input(image_array1)
prediction1 = model.predict(image_array1)

## Print the image and prediction here in the Jupyter Notebook.
print("Prediction: {}".format(decode_predictions(prediction1, top = 1)[0]))
plt.imshow(img1)
plt.show()

## Export the prediction to a prediction file as specified in the document.
resnet50_predictions = resnet_dir.joinpath('assignment06-3_resnet50_predictions.csv')

test_dict1 = {'Image 1': str(decode_predictions(prediction1, top = 1)[0])}

with open(resnet50_predictions, 'w') as csv_file:
    writer = csv.writer(csv_file)
    for key, value in test_dict1.items():
        writer.writerow([key,value])
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

Prediction: [('n02133161', 'American_black_bear', 0.99549896)]

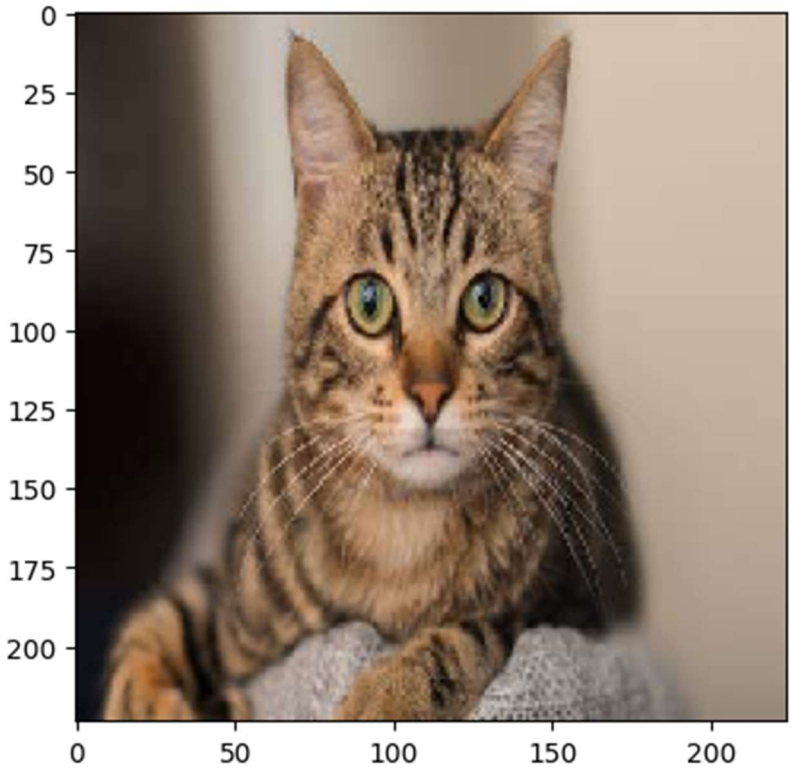


In [17]:

```
## Adjust the image. Convert to numpy array, add batch dimension, and preprocess.
image_array2 = image.img_to_array(img2)
image_array2 = np.expand_dims(image_array2, axis =0)
image_array2 = preprocess_input(image_array2)
prediction2 = model.predict(image_array2)

## Print the image and prediction here in the Jupyter Notebook.
print("Prediction: {}".format(decode_predictions(prediction2, top = 1)[0]))
plt.imshow(img2)
plt.show()

## Export the prediction to a prediction file as specified in the document.
resnet50_predictions = resnet_dir.joinpath('assignment06-3_resnet50_predictions.csv')

test_dict2 = {'Image 2': str(decode_predictions(prediction2, top = 1)[0])}

with open(resnet50_predictions, 'a') as csv_file:
    writer = csv.writer(csv_file)
    for key, value in test_dict2.items():
        writer.writerow([key,value])
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

Prediction: [('n02123045', 'tabby', 0.64965636)]

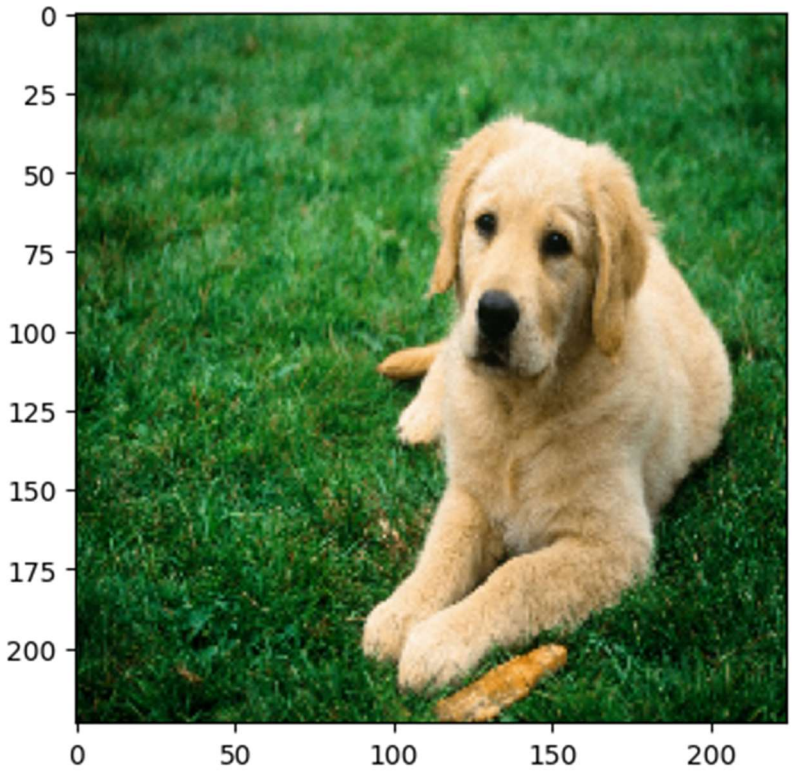


In [18]:

```
## Adjust the image. Convert to numpy array, add batch dimension, and preprocess.
image_array3 = image.img_to_array(img3)
image_array3 = np.expand_dims(image_array3, axis =0)
image_array3 = preprocess_input(image_array3)
prediction3 = model.predict(image_array3)

## Print the image and prediction here in the Jupyter Notebook.
print("Prediction: {}".format(decode_predictions(prediction3, top = 1)[0]))
plt.imshow(img3)
plt.show()

## Export the prediction to a prediction file as specified in the document.
resnet50_predictions = resnet_dir.joinpath('assignment06-3_resnet50_predictions.csv')

test_dict3 = {'Image 3': str(decode_predictions(prediction3, top = 1)[0])}

with open(resnet50_predictions, 'a') as csv_file:
    writer = csv.writer(csv_file)
    for key, value in test_dict3.items():
        writer.writerow([key,value])
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

Prediction: [('n02099601', 'golden_retriever', 0.9001567)]

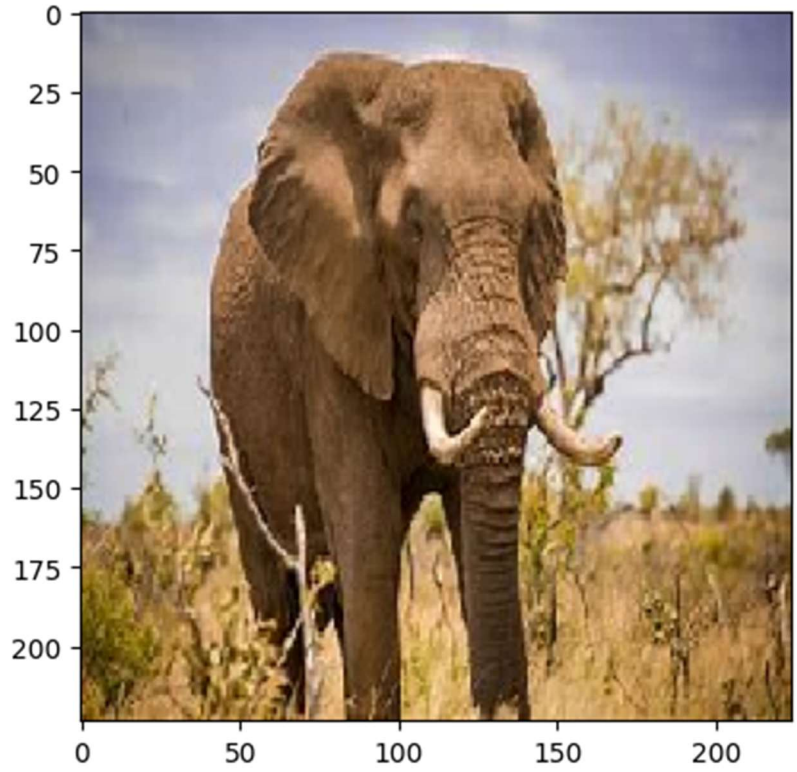


In [19]:

```
## Adjust the image. Convert to numpy array, add batch dimension, and preprocess.
image_array4 = image.img_to_array(img4)
image_array4 = np.expand_dims(image_array4, axis =0)
image_array4 = preprocess_input(image_array4)
prediction4 = model.predict(image_array4)

## Print the image and prediction here in the Jupyter Notebook.
print("Prediction: {}".format(decode_predictions(prediction4, top = 1)[0]))
plt.imshow(img4)
plt.show()

## Export the prediction to a prediction file as specified in the document.
resnet50_predictions = resnet_dir.joinpath('assignment06-3_resnet50_predictions.csv')

test_dict4 = {'Image 4': str(decode_predictions(prediction4, top = 1)[0])}

with open(resnet50_predictions, 'a') as csv_file:
    writer = csv.writer(csv_file)
    for key, value in test_dict4.items():
        writer.writerow([key,value])
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

Prediction: [('n01871265', 'tusker', 0.6794936)]



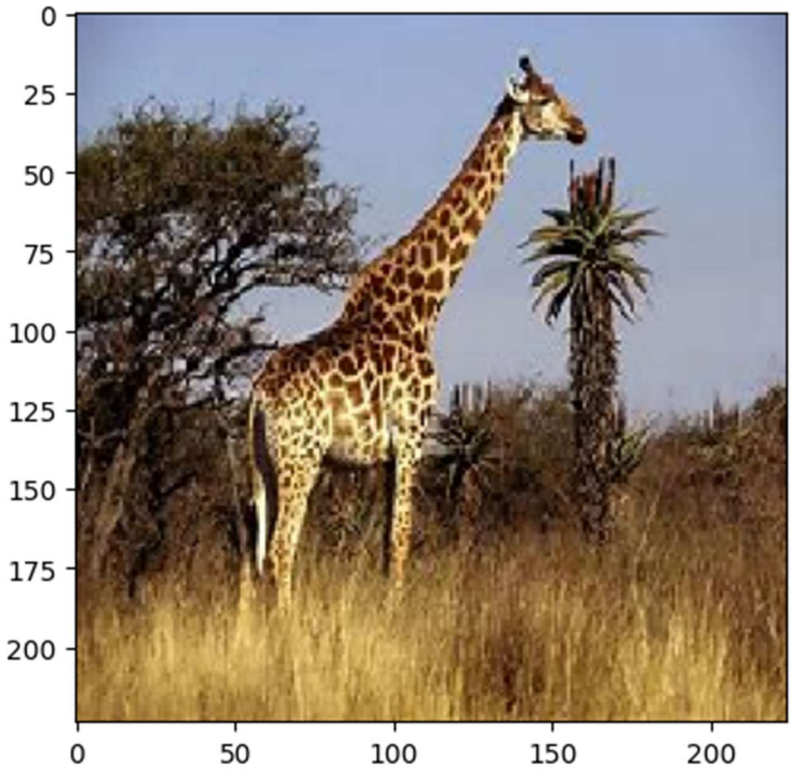
In [20]:

```
## Adjust the image. Convert to numpy array, add batch dimension, and preprocess.
image_array5 = image.img_to_array(img5)
image_array5 = np.expand_dims(image_array5, axis =0)
image_array5 = preprocess_input(image_array5)
prediction5 = model.predict(image_array5)

## Print the image and prediction here in the Jupyter Notebook.
print("Prediction: {}".format(decode_predictions(prediction5, top = 1)[0]))
plt.imshow(img5)
plt.show()

## Export the prediction to a prediction file as specified in the document.
resnet50_predictions = resnet_dir.joinpath('assignment06-3_resnet50_predictions.csv')

test_dict5 = {'Image 5': str(decode_predictions(prediction5, top = 1)[0])}

with open(resnet50_predictions, 'a') as csv_file:
    writer = csv.writer(csv_file)
    for key, value in test_dict5.items():
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

```
    writer.writerow([key,value])
```

Prediction: [('n02130308', 'cheetah', 0.98081505)]



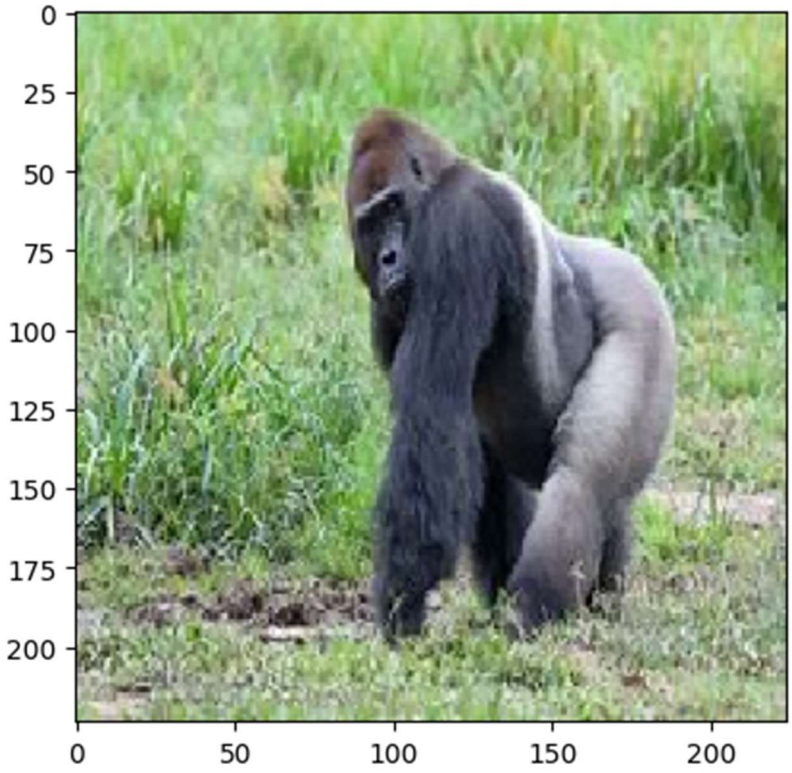
In [21]:

```
## Adjust the image. Convert to numpy array, add batch dimension, and preprocess.
image_array6 = image.img_to_array(img6)
image_array6 = np.expand_dims(image_array6, axis =0)
image_array6 = preprocess_input(image_array6)
prediction6 = model.predict(image_array6)

## Print the image and prediction here in the Jupyter Notebook.
print("Prediction: {}".format(decode_predictions(prediction6, top = 1)[0]))
plt.imshow(img6)
plt.show()

## Export the prediction to a prediction file as specified in the document.
resnet50_predictions = resnet_dir.joinpath('assignment06-3_resnet50_predictions.csv')

test_dict6 = {'Image 6': str(decode_predictions(prediction6, top = 1)[0])}

with open(resnet50_predictions, 'a') as csv_file:
  writer = csv.writer(csv_file)
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

```
    for key, value in test_dict6.items():
        writer.writerow([key,value])
```

Prediction: [('n02480855', 'gorilla', 0.9995832)]



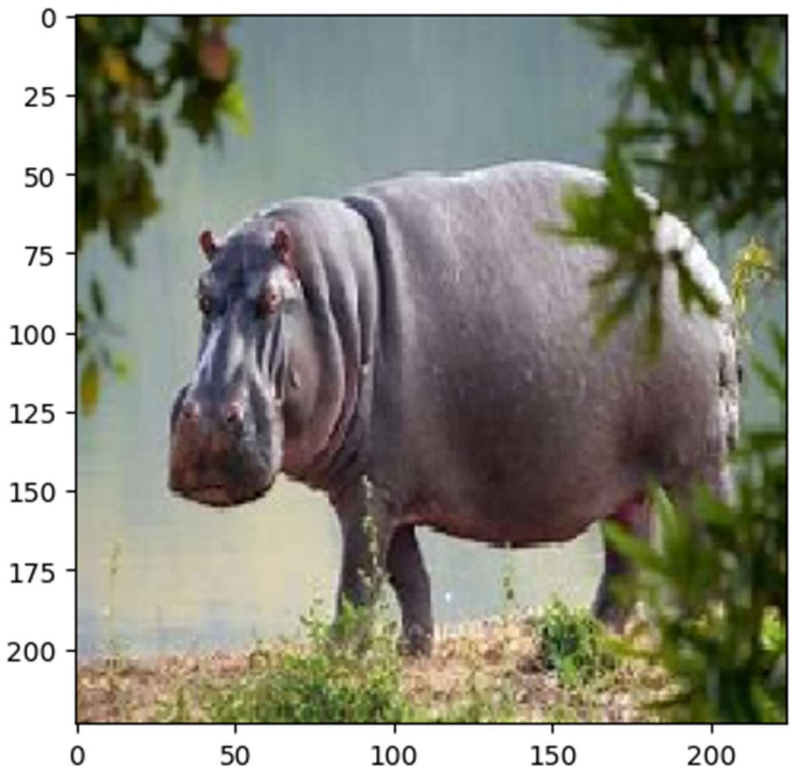
In [22]:

```
## Adjust the image. Convert to numpy array, add batch dimension, and preprocess.
image_array7 = image.img_to_array(img7)
image_array7 = np.expand_dims(image_array7, axis =0)
image_array7 = preprocess_input(image_array7)
prediction7 = model.predict(image_array7)

## Print the image and prediction here in the Jupyter Notebook.
print("Prediction: {}".format(decode_predictions(prediction7, top = 1)[0]))
plt.imshow(img7)
plt.show()

## Export the prediction to a prediction file as specified in the document.
resnet50_predictions = resnet_dir.joinpath('assignment06-3_resnet50_predictions.csv')

test_dict7 = {'Image 7': str(decode_predictions(prediction7, top = 1)[0])}

with open(resnet50_predictions, 'a') as csv_file:
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

```
    writer = csv.writer(csv_file)
    for key, value in test_dict7.items():
        writer.writerow([key,value])
```

Prediction: [('n02398521', 'hippopotamus', 0.9997882)]



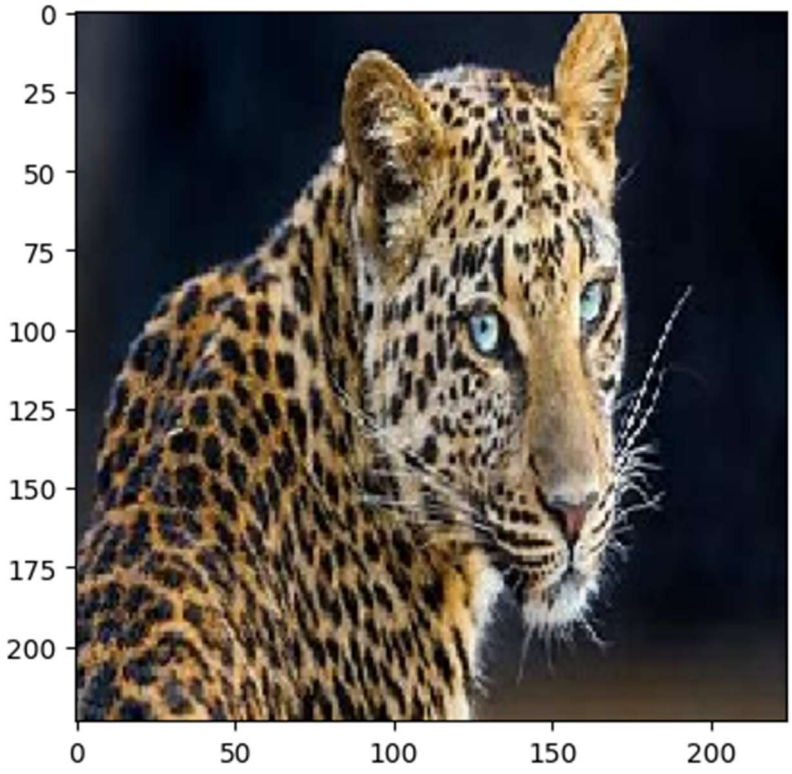
In [23]:

```
## Adjust the image. Convert to numpy array, add batch dimension, and preprocess.
image_array8 = image.img_to_array(img8)
image_array8 = np.expand_dims(image_array8, axis =0)
image_array8 = preprocess_input(image_array8)
prediction8 = model.predict(image_array8)

## Print the image and prediction here in the Jupyter Notebook.
print("Prediction: {}".format(decode_predictions(prediction8, top = 1)[0]))
plt.imshow(img8)
plt.show()

## Export the prediction to a prediction file as specified in the document.
resnet50_predictions = resnet_dir.joinpath('assignment06-3_resnet50_predictions.csv')

test_dict8 = {'Image 8': str(decode_predictions(prediction8, top = 1)[0])}
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

```
with open(resnet50_predictions, 'a') as csv_file:
   writer = csv.writer(csv_file)
   for key, value in test_dict8.items():
      writer.writerow([key,value])
```

Prediction: [('n02128385', 'leopard', 0.91118056)]



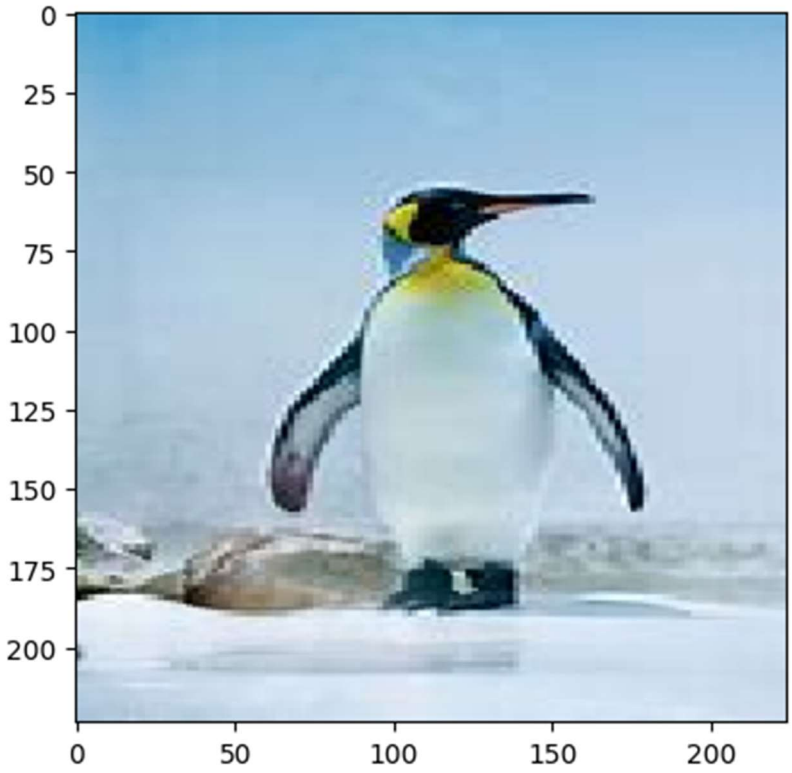
In [24]:

```
## Adjust the image. Convert to numpy array, add batch dimension, and preprocess.
image_array9 = image.img_to_array(img9)
image_array9 = np.expand_dims(image_array9, axis =0)
image_array9 = preprocess_input(image_array9)
prediction9 = model.predict(image_array9)

## Print the image and prediction here in the Jupyter Notebook.
print("Prediction: {}".format(decode_predictions(prediction9, top = 1)[0]))
plt.imshow(img9)
plt.show()

## Export the prediction to a prediction file as specified in the document.
resnet50_predictions = resnet_dir.joinpath('assignment06-3_resnet50_predictions.csv')

test_dict9 = {'Image 9': str(decode_predictions(prediction9, top = 1)[0])}
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

```
with open(resnet50_predictions, 'a') as csv_file:
   writer = csv.writer(csv_file)
   for key, value in test_dict9.items():
      writer.writerow([key,value])
```

Prediction: [('n02056570', 'king_penguin', 0.99983263)]



In [25]:

```
## Adjust the image. Convert to numpy array, add batch dimension, and preprocess.
image_array10 = image.img_to_array(img10)
image_array10 = np.expand_dims(image_array10, axis =0)
image_array10 = preprocess_input(image_array10)
prediction10 = model.predict(image_array10)

## Print the image and prediction here in the Jupyter Notebook.
print("Prediction: {}".format(decode_predictions(prediction10, top = 1)[0]))
plt.imshow(img10)
plt.show()

## Export the prediction to a prediction file as specified in the document.
resnet50_predictions = resnet_dir.joinpath('assignment06-3_resnet50_predictions.csv')
```

DSC650-T302 Big Data (2235-1)
Professor Iranitalab
Assignment 06 Code and Outputs
Jake Meyer
04/23/2023

```
test_dict10 = {'Image 10': str(decode_predictions(prediction10, top = 1)[0])}

with open(resnet50_predictions, 'a') as csv_file:
   writer = csv.writer(csv_file)
   for key, value in test_dict10.items():
      writer.writerow([key,value])
```

Prediction: [('n02129604', 'tiger', 0.85221666)]