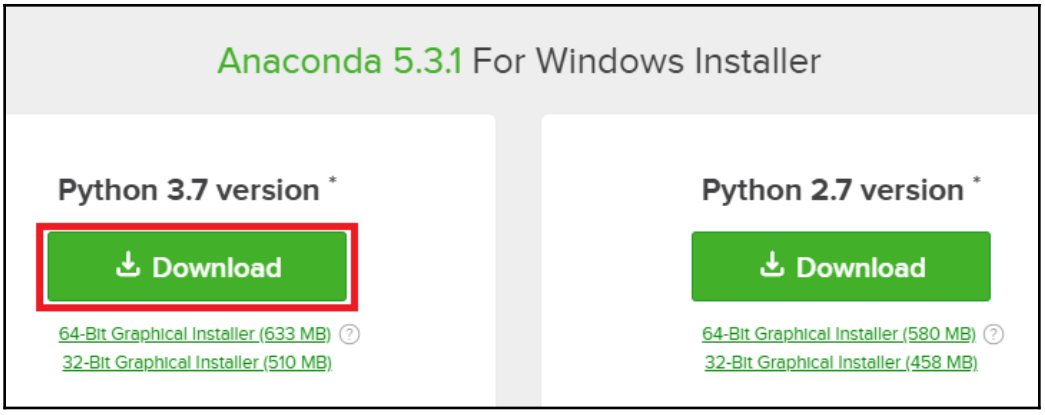
Chapter 01: Setting Up an Anaconda Environment




```
C:\Users\josephs>where python
C:\ProgramData\Anaconda3\python.exe
C:\Users\josephs\AppData\Local\Programs\Python\Python37-32\python.exe
```

```
C:\Users\josephs>where ipython
C:\ProgramData\Anaconda3\Scripts\ipython.exe
C:\Users\josephs\AppData\Local\Programs\Python\Python37-32\Scripts\ipython.exe
```

```
C:\Users\josephs>where pip
C:\ProgramData\Anaconda3\Scripts\pip.exe
C:\Users\josephs\AppData\Local\Programs\Python\Python37-32\Scripts\pip.exe
```

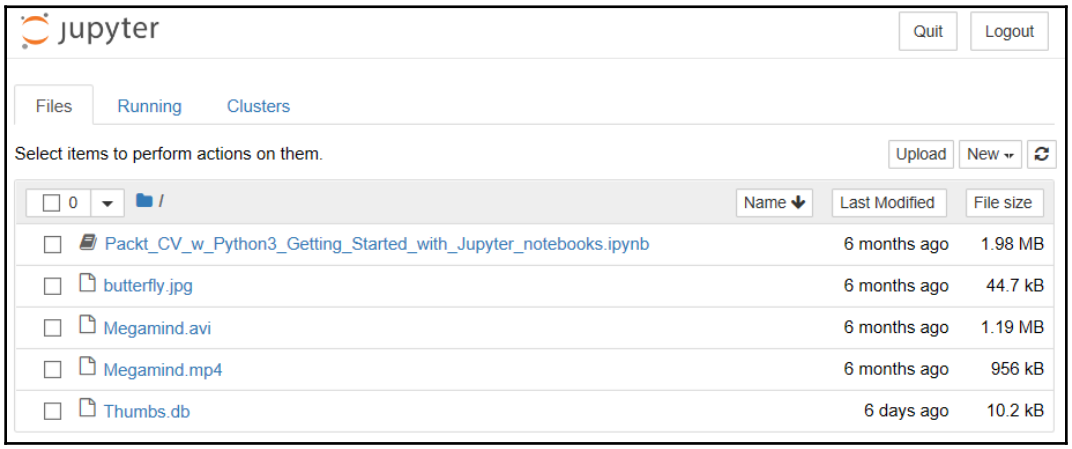
```
C:\Users\josephs>where conda
C:\ProgramData\Anaconda3\Library\bin\conda.bat
C:\ProgramData\Anaconda3\Scripts\conda.exe
```

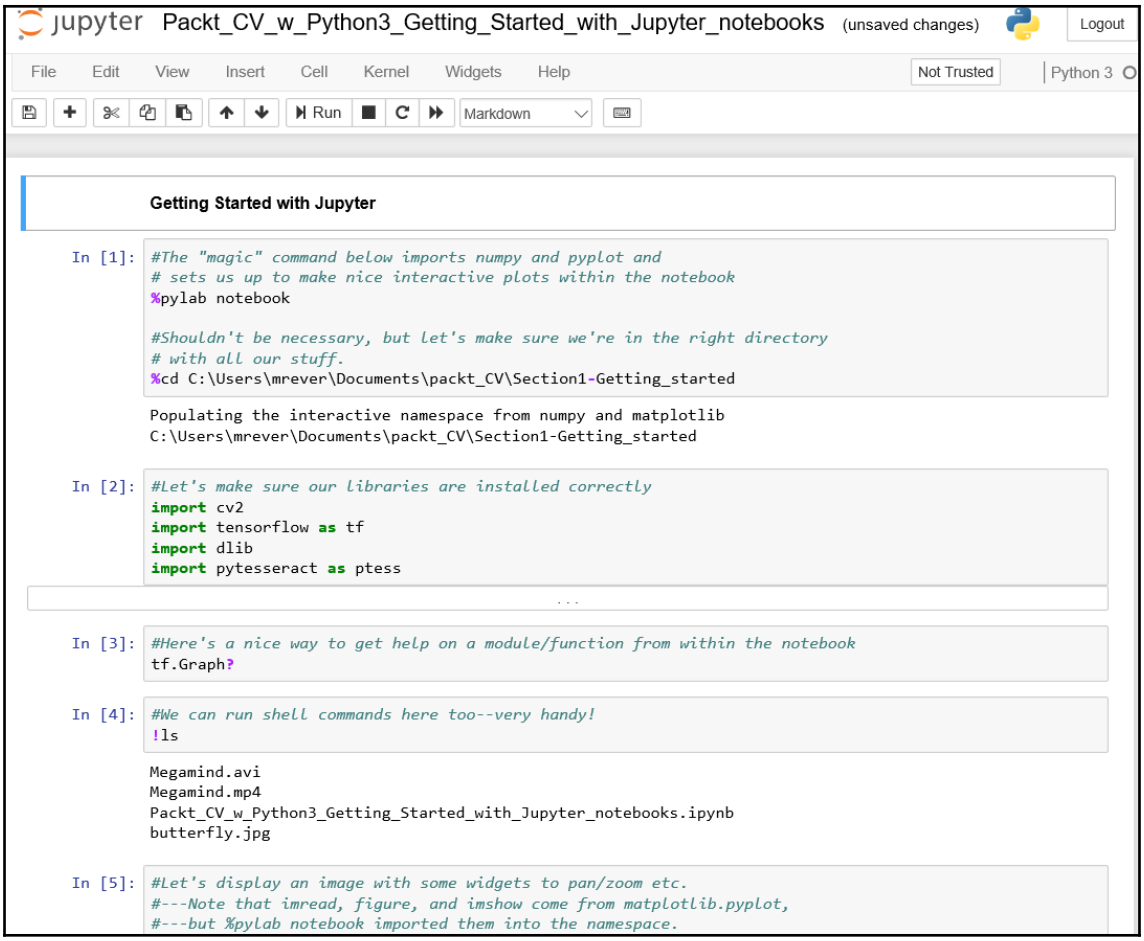
```
C:\Users\josephs>conda install -c conda-forge opencv
Solving environment: done

## Package Plan ##

  environment location: C:\ProgramData\Anaconda3

  added / updated specs:
    - opencv


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    conda-4.5.11               |        py36_1000         655 KB  conda-forge
    libopencv-3.4.1            |       h875b8b8_3        37.0 MB
    py-opencv-3.4.1            |   py36h1b0d24d_3         1.5 MB
    certifi-2018.4.16          |           py36_0         143 KB  conda-forge
    opencv-3.4.1               |   py36h6fd60c2_3           9 KB
    ------------------------------------------------------------
                                           Total:        39.3 MB

The following NEW packages will be INSTALLED:
```

```
C:\Users\josephs>conda install -c menpo dlib
Solving environment: done

## Package Plan ##

  environment location: C:\ProgramData\Anaconda3

  added / updated specs:
    - dlib


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    incremental-17.5.0         |           py35_0          25 KB
    werkzeug-0.14.1            |           py35_0         427 KB
    terminado-0.8.1            |           py35_1          21 KB
    packaging-17.1             |           py35_0          34 KB
    astropy-3.0.4              |   py35hfa6e2cd_0         6.6 MB
```

```
> where tesseract
D:\cygwin64\bin\tesseract.exe
```

```
C:\Users\josephs>pip install tensorflow
Requirement already satisfied: tensorflow in c:\programdata\anaconda3\lib\site-packages (1.12.0)
Requirement already satisfied: protobuf>=3.6.1 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (3.6.1)
Requirement already satisfied: wheel>=0.26 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (0.31.1)
Requirement already satisfied: absl-py>=0.1.6 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (0.4.0)
Requirement already satisfied: grpcio>=1.8.6 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (1.14.1)
Requirement already satisfied: keras-preprocessing>=1.0.5 in c:\programdata\anaconda3\lib\site-packages (from tensorflo
) (1.0.5)
Requirement already satisfied: six>=1.10.0 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (1.11.0)
Requirement already satisfied: keras-applications>=1.0.6 in c:\programdata\anaconda3\lib\site-packages (from tensorflow
 (1.0.6)
Requirement already satisfied: astor>=0.6.0 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (0.7.1)
Requirement already satisfied: gast>=0.2.0 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: numpy>=1.13.3 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (1.14.3)
Requirement already satisfied: tensorboard<1.13.0,>=1.12.0 in c:\programdata\anaconda3\lib\site-packages (from tensorfl
w) (1.12.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\programdata\anaconda3\lib\site-packages (from tensorflow) (1.1.0)
Requirement already satisfied: setuptools in c:\programdata\anaconda3\lib\site-packages (from protobuf>=3.6.1->tensorfl
w) (39.1.0)
Requirement already satisfied: h5py in c:\programdata\anaconda3\lib\site-packages (from keras-applications>=1.0.6->tens
rflow) (2.7.1)
Requirement already satisfied: markdown>=2.6.8 in c:\programdata\anaconda3\lib\site-packages (from tensorboard<1.13.0,>
1.12.0->tensorflow) (2.6.11)
Requirement already satisfied: werkzeug>=0.11.10 in c:\programdata\anaconda3\lib\site-packages (from tensorboard<1.13.0
>=1.12.0->tensorflow) (0.14.1)
```



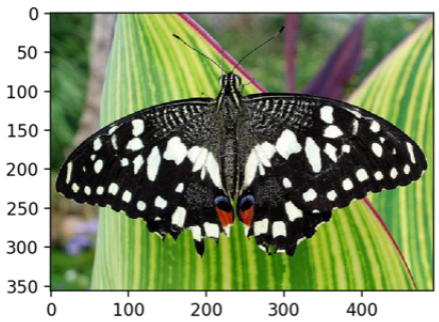



```
%cd C:\Users\mrever\Documents\packt_CV\Section1-Getting_started
Populating the interactive namespace from numpy and matplotlib
C:\Users\mrever\Documents\packt_CV\Section1-Getting_started
```

```
In [4]: #We can run shell commands here too--very handy!
        !ls
Megamind.avi
Megamind.mp4
Packt_CV_w_Python3_Getting_Started_with_Jupyter_notebooks.ipynb
butterfly.jpg
```

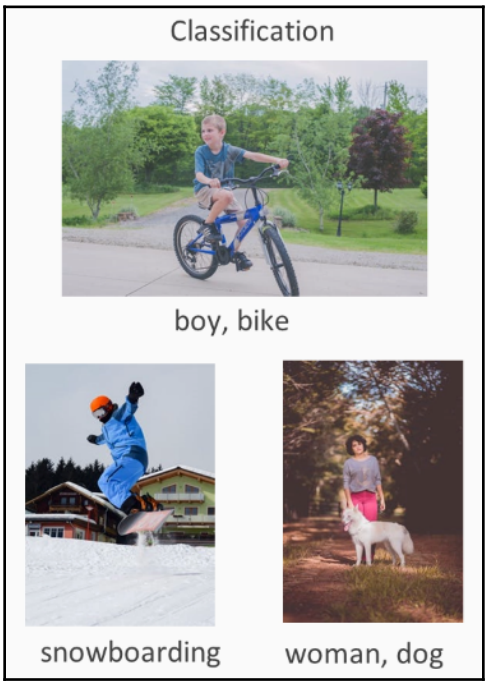
In [5]:
```
#Let's display an image with some widgets to pan/zoom etc.
#---Note that imread, figure, and imshow come from matplotlib.pyplot,
#---but %pylab notebook imported them into the namespace.
#---This is good for notebooks, but not necessarily good for general practice.

testim = imread('butterfly.jpg')
figure()
imshow(testim)
```

```
<IPython.core.display.Javascript object>
```

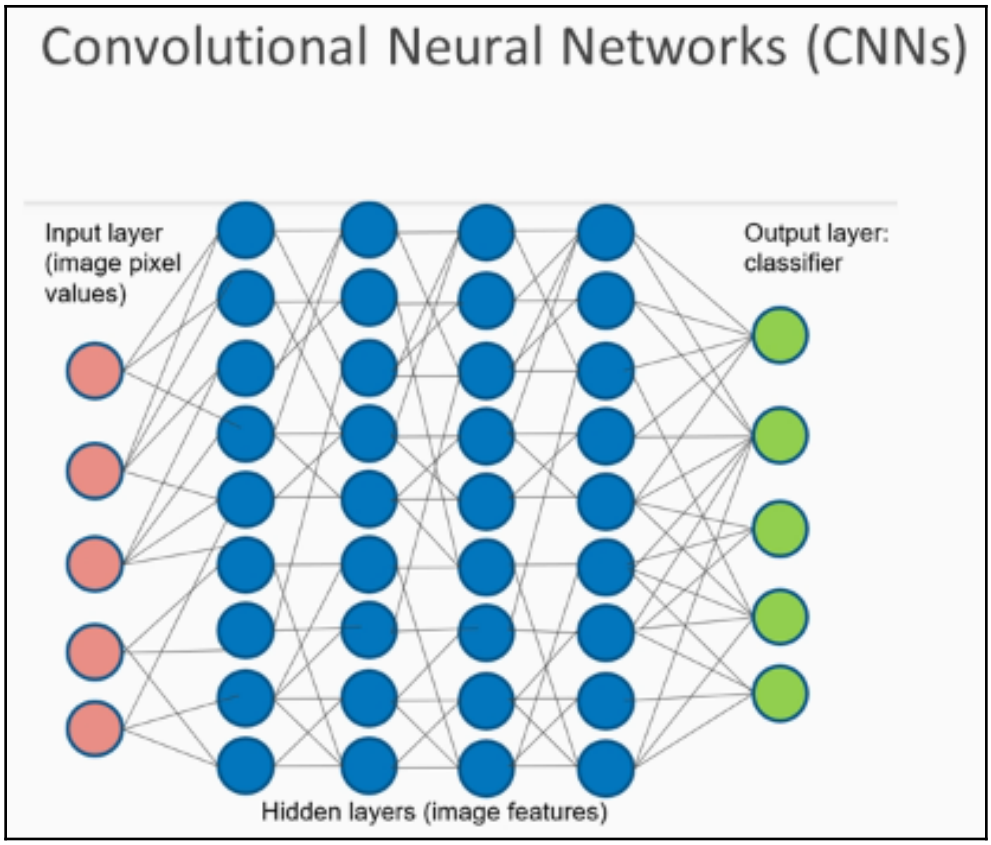


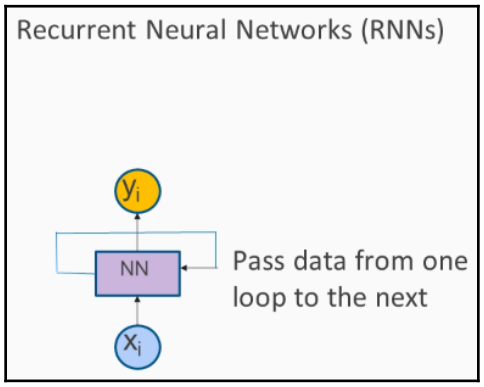
In [7]:
```
#Here's a nice way to watch a video within the notebook--will be useful
# when we want to work with videos
import io
import base64
from IPython.display import HTML

def playvideo(filename):
    video = io.open(filename, 'r+b').read()
    encoded = base64.b64encode(video)
    return HTML(data='''<video alt="test" controls>
                    <source src="data:video/mp4;base64,{0}" type="video/mp4"/>
                </video>'''.format(encoded.decode('ascii')))
```

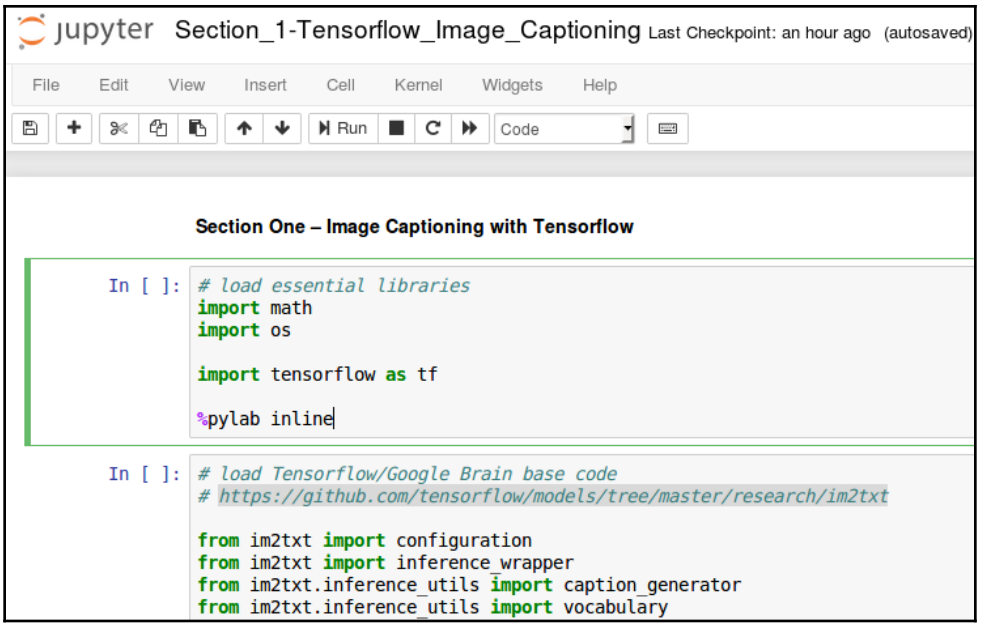
Chapter 02: Image Captioning with TensorFlow

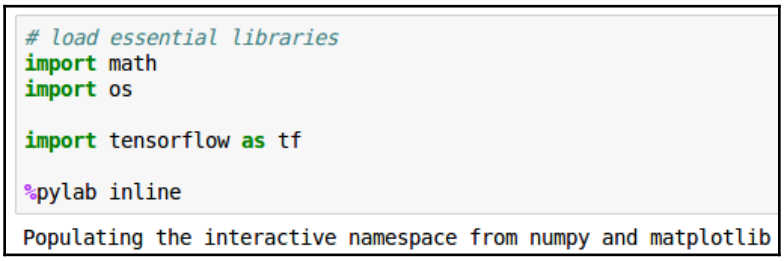


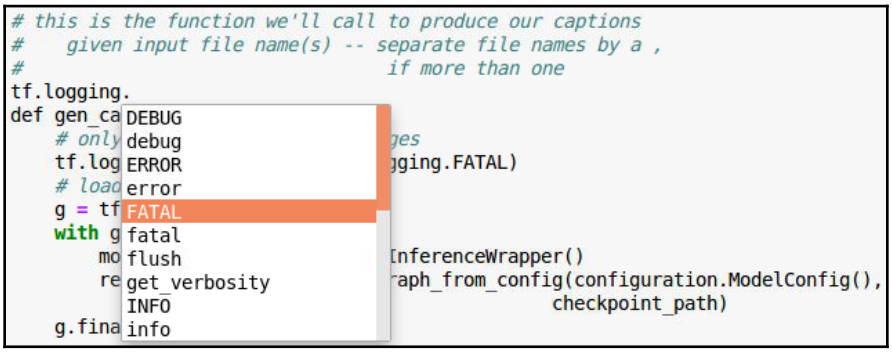

Captioning
A boy riding on a bike
A man is snowboarding
A woman with a dog


Convolutional Neural Networks (CNNs)
Input layer (image pixel values)
Output layer: classifier
Hidden layers (image features)


Recurrent Neural Networks (RNNs)
y_i
NN
x_i
Pass data from one loop to the next

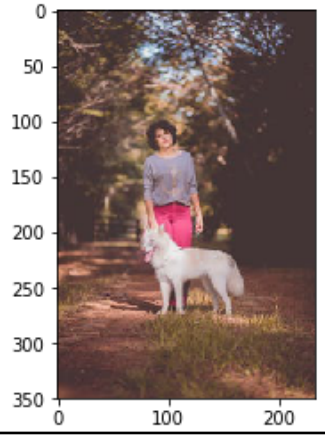

jupyter Section_1-Tensorflow_Image_Captioning Last Checkpoint: an hour ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run Code
Section One – Image Captioning with Tensorflow
In [ ]:
load essential libraries
import math
import os
import tensorflow as tf
%pylab inline
In [ ]:
load Tensorflow/Google Brain base code
https://github.com/tensorflow/models/tree/master/research/im2txt
from im2txt import configuration
from im2txt import inference_wrapper
from im2txt.inference_utils import caption_generator
from im2txt.inference_utils import vocabulary


load essential libraries
import math
import os
import tensorflow as tf
%pylab inline
Populating the interactive namespace from numpy and matplotlib


this is the function we'll call to produce our captions
given input file name(s) -- separate file names by a ,
if more than one
tf.logging.
def gen_ca
only
tf.log
load
g = tf
with g
mo
re
g.fina
DEBUG
debug
ERROR
error
FATAL
fatal
flush
get_verbosity
INFO
info
ges
gging.FATAL)
InferenceWrapper()
raph_from_config(configuration.ModelConfig(),
checkpoint_path)

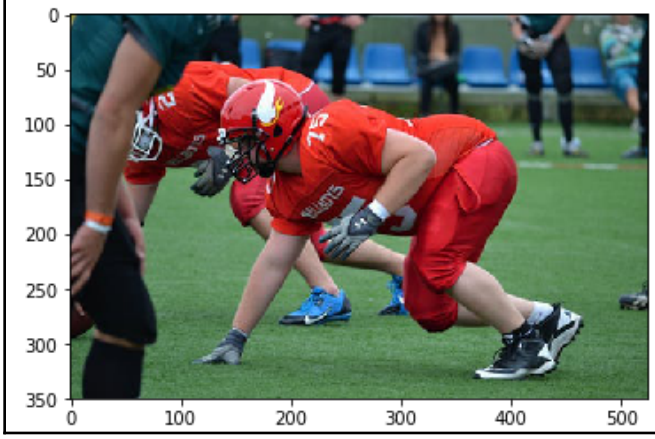
```
testfile = 'test_images/dog.jpeg'

figure()
imshow(imread(testfile))

capts = gen_caption(testfile)
```

```
Captions for image dog.jpeg:
  0) a woman and a dog are on a bench . (p=0.000024)
  1) a woman and a dog are standing on the grass . (p=0.000014)
  2) a woman and a dog are standing on a bench . (p=0.000012)
```



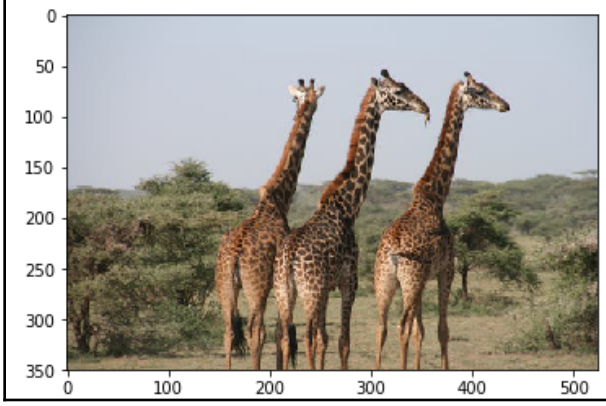
```
testfile = 'test_images/football.jpeg'

figure()
imshow(imread(testfile))

capts = gen_caption(testfile)
```

```
Captions for image football.jpeg:
  0) a couple of men playing a game of frisbee . (p=0.001167)
  1) a couple of men playing a game of football . (p=0.001053)
  2) a couple of men playing a game of soccer . (p=0.000807)
```



```
testfile = 'test_images/giraffes.jpeg'

figure()
imshow(imread(testfile))

capts = gen_caption(testfile)
```

```
Captions for image giraffes.jpeg:
  0) a group of giraffe standing next to each other . (p=0.002270)
  1) a group of giraffes are standing in a field (p=0.000959)
  2) a group of giraffe standing next to each other on a field . (p=0.000744)
```

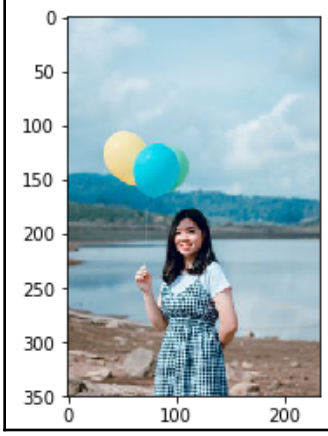


```
testfile = 'test_images/headphones.jpeg'

figure()
imshow(imread(testfile))

capts = gen_caption(testfile)
```

```
Captions for image headphones.jpeg:
  0) a woman holding a cell phone in her hand . (p=0.002533)
  1) a woman holding a cell phone up to her ear . (p=0.002143)
  2) a woman holding a cell phone in her hands . (p=0.001198)
```

```
testfile = 'test_images/ballons.jpeg'

figure()
imshow(imread(testfile))

capts = gen_caption(testfile)
```

```
Captions for image ballons.jpeg:
  0) a woman standing on a beach flying a kite . (p=0.004793)
  1) a woman is flying a kite on the beach . (p=0.003442)
  2) a young girl flying a kite on a beach . (p=0.002200)
```

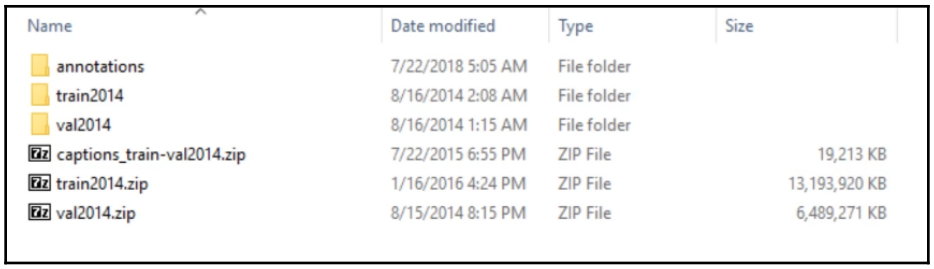


```
input_files = 'test_images/ballons.jpeg,test_images/bike.jpeg,test_images/dog.jpeg,test_images/fireworks.jpeg,test_images/

capts = gen_caption(input_files)
```

```
Captions for image ballons.jpeg:
  0) a woman standing on a beach flying a kite . (p=0.004793)
  1) a woman is flying a kite on the beach . (p=0.003442)
  2) a young girl flying a kite on a beach . (p=0.002200)
Captions for image bike.jpeg:
  0) a man riding a bike down a dirt road . (p=0.003308)
  1) a man riding a bike down a road . (p=0.002444)
  2) a man riding a bike down a road next to a forest . (p=0.000610)
Captions for image dog.jpeg:
  0) a woman and a dog are on a bench . (p=0.000024)
  1) a woman and a dog are standing on the grass . (p=0.000014)
  2) a woman and a dog are standing on a bench . (p=0.000012)
Captions for image fireworks.jpeg:
  0) a view of a tree in a park . (p=0.000004)
  1) a view of a tree in the middle of a park . (p=0.000003)
  2) a view of a tree in the middle of a city . (p=0.000003)
Captions for image football.jpeg:
  0) a couple of men playing a game of frisbee . (p=0.001167)
  1) a couple of men playing a game of football . (p=0.001053)
  2) a couple of men playing a game of soccer . (p=0.000807)
Captions for image giraffes.jpeg:
  0) a group of giraffe standing next to each other . (p=0.002270)
  1) a group of giraffes are standing in a field (p=0.000959)
  2) a group of giraffe standing next to each other on a field . (p=0.000744)
Captions for image headphones.jpeg:
  0) a woman holding a cell phone in her hand . (p=0.002533)
  1) a woman holding a cell phone up to her ear . (p=0.002143)
  2) a woman holding a cell phone in her hands . (p=0.001198)
Captions for image laughing.jpeg:
  0) a group of people standing next to each other . (p=0.003400)
  1) a man and a woman standing next to each other . (p=0.002685)
  2) a group of people standing in a room . (p=0.001571)
Captions for image objects.jpeg:
  0) a nc of items that are on a table . (p=0.000309)
  1) a nc of items that are sitting on a table . (p=0.000188)
  2) a nc of cellphones that are on a table . (p=0.000186)
Captions for image snowboard.jpeg:
  0) a man flying through the air while riding a snowboard . (p=0.028471)
  1) a man flying through the air while riding skis . (p=0.005110)
  2) a man flying through the air on top of a snowboard . (p=0.001624)
```

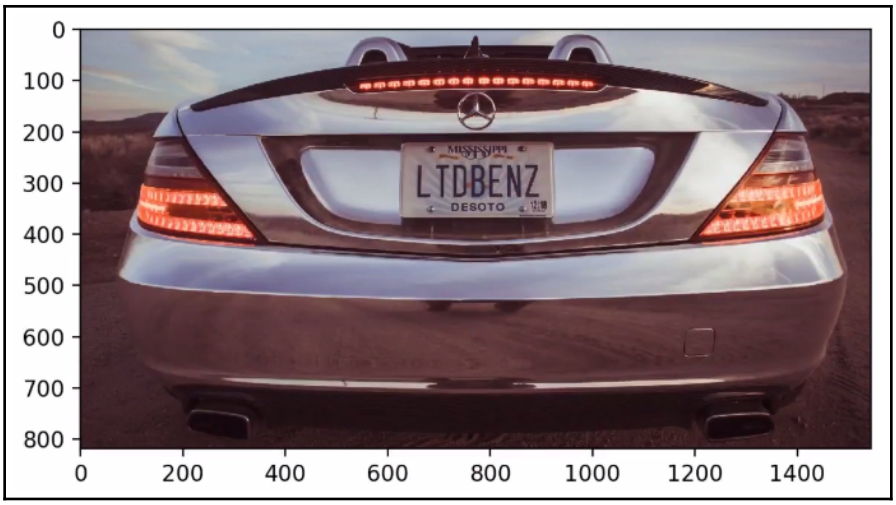
In [15]:
```
# First download pretrained Inception (v3) model

import webbrowser
webbrowser.open("http://download.tensorflow.org/models/inception_v3_2016_08_28.tar.gz")

# Completely unzip tar.gz file to get inception_v3.ckpt,
# --recommend storing in im2txt/data directory
```

Out[15]: True



Name	Date modified	Type	Size
annotations	7/22/2018 5:05 AM	File folder	
train2014	8/16/2014 2:08 AM	File folder	
val2014	8/16/2014 1:15 AM	File folder	
captions_train-val2014.zip	7/22/2015 6:55 PM	ZIP File	19,213 KB
train2014.zip	1/16/2016 4:24 PM	ZIP File	13,193,920 KB
val2014.zip	8/15/2014 8:15 PM	ZIP File	6,489,271 KB

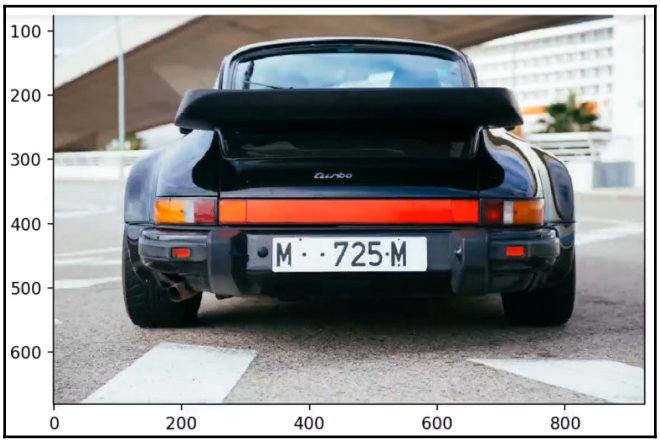
Chapter 03:Reading License Plates with OpenCV

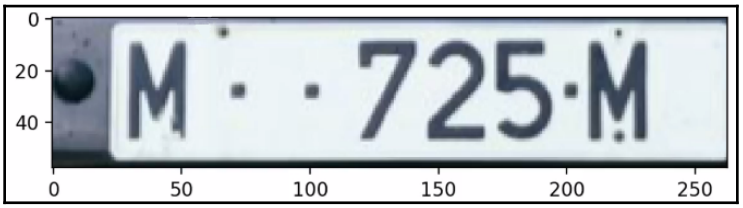

LTDBENZ
DESOTO

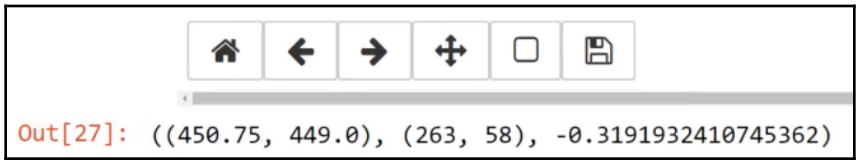

MISSISSIPPI
LTDBENZ
DESOTO
12 18

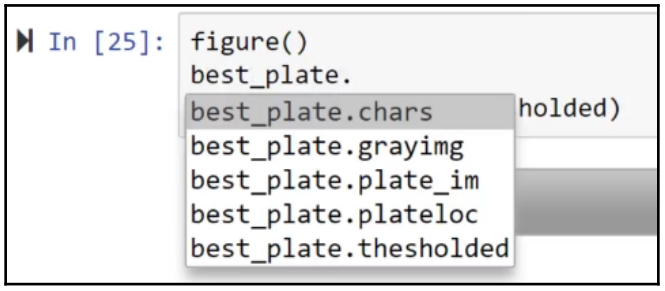

LTDBENZ

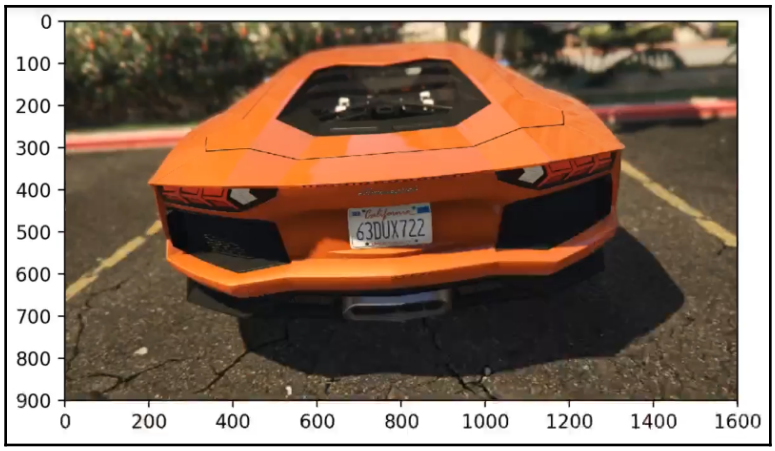
```
License plate read: M725M
```



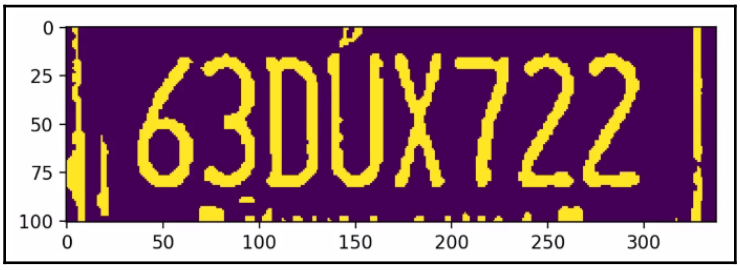


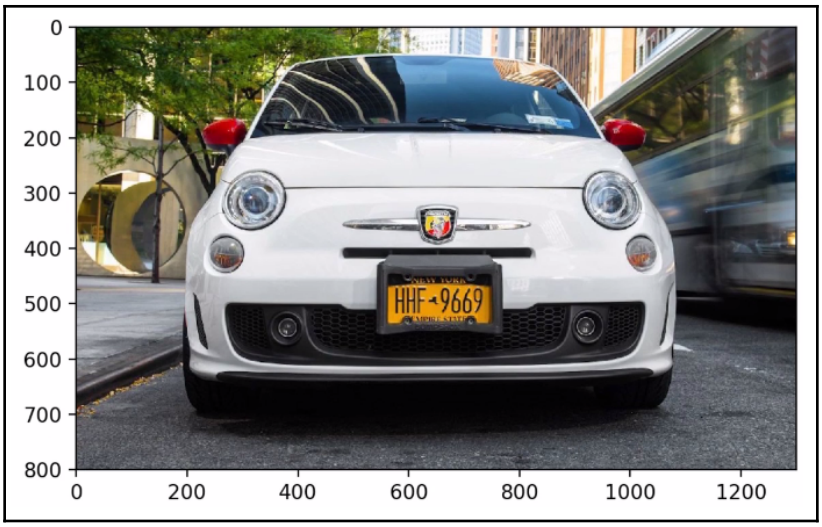




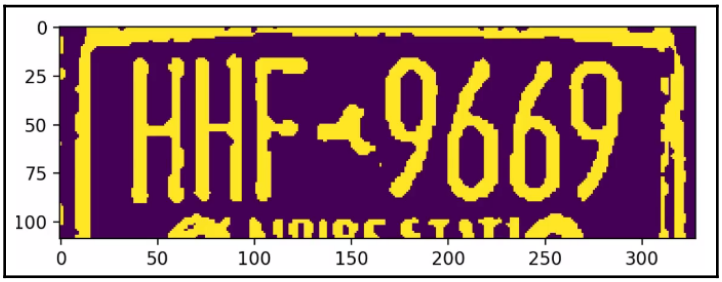

0
100
200
300
400
500
600
700
800
900
0
200
400
600
800
1000
1200
1400
1600
63DUX722


425
450
475
500
525
550
575
650
700
750
800
850
900
SEP
California
2015
63DUX722
dmv.ca.gov

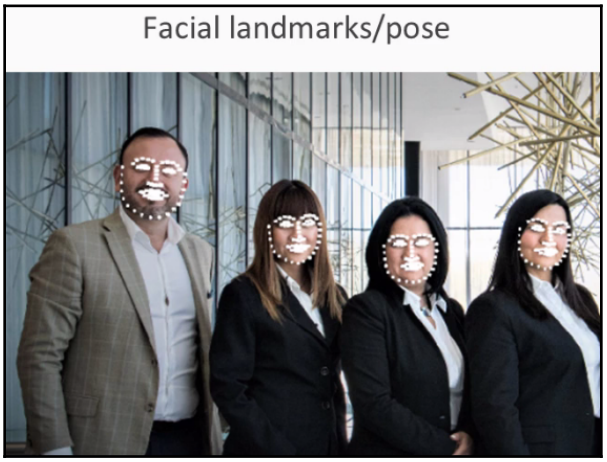

0
25
50
75
100
0
50
100
150
200
250
300
63DUX722



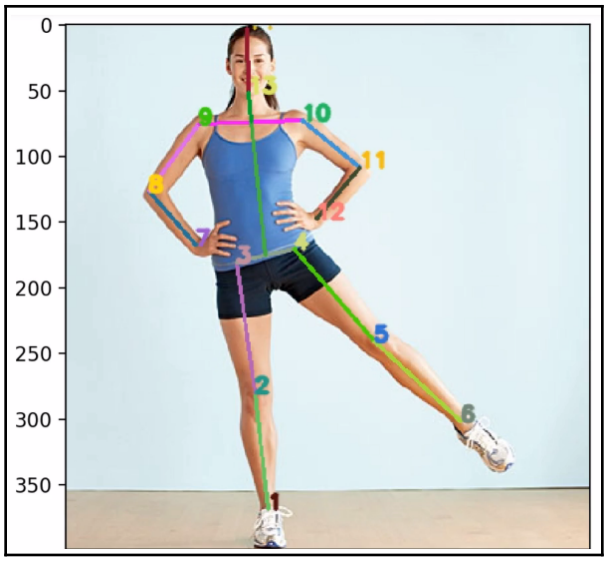

```
License plate read: HHF9669
```



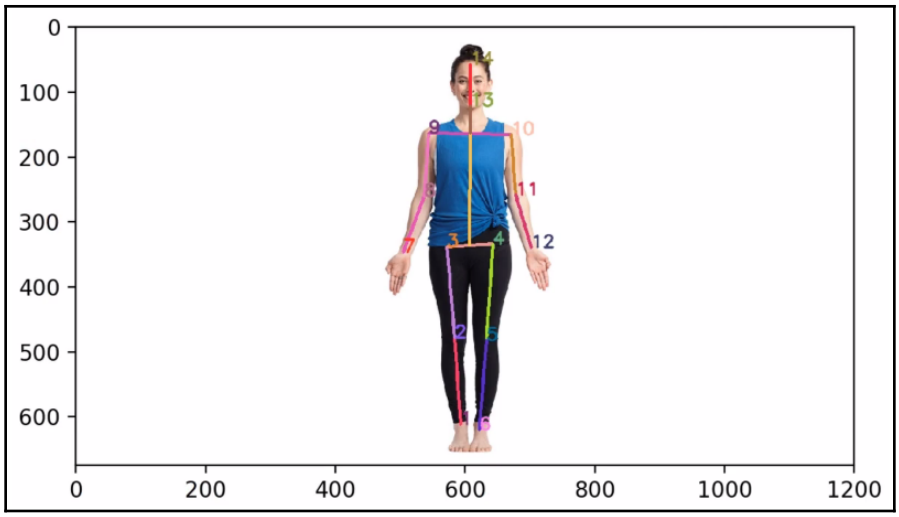
Chapter 04:Human Pose Estimation with TensorFlow

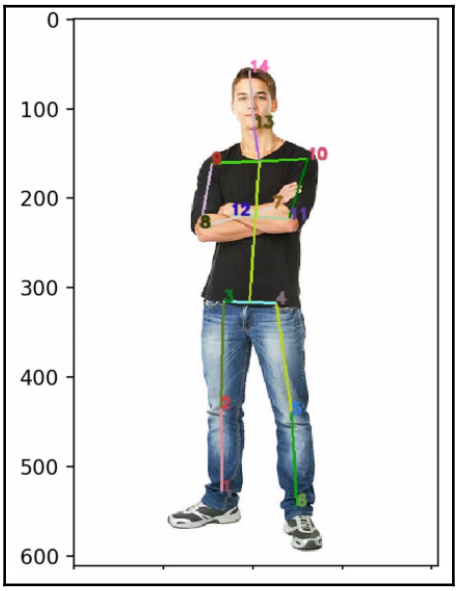


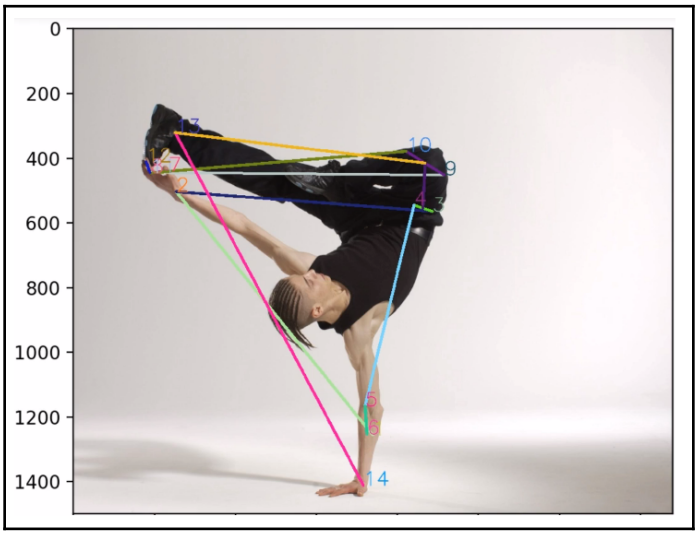
```
array([[154.21713603, 369.49524975,   0.97495675],
       [144.66004324, 280.38701272,   0.97836888],
       [130.74821067, 181.00517952,   0.89270324],
       [174.04091334, 171.14334095,   0.88625103],
       [235.90207493, 241.08008146,   0.97764331],
       [301.39780343, 302.21447849,   0.92958534],
       [ 99.43290633, 168.43356824,   0.98397946],
       [ 63.96742463, 127.38911414,   0.99213421],
       [101.82382166,  76.10697242,   0.99111789],
       [180.59155178,  73.29837465,   0.99223047],
       [224.60096455, 109.43770075,   0.99714881],
       [191.66644788, 148.67574054,   0.99290884],
       [139.69199833,  52.24837098,   0.9979133 ],
       [138.82638311,   3.14040202,   0.98071539]])
```


0
50
100
150
200
250
300
350
10
11
12
5
2
6


0
100
200
300
400
500
600


0
100
200
300
400
500
600
0
200
400
600
800
1000
1200

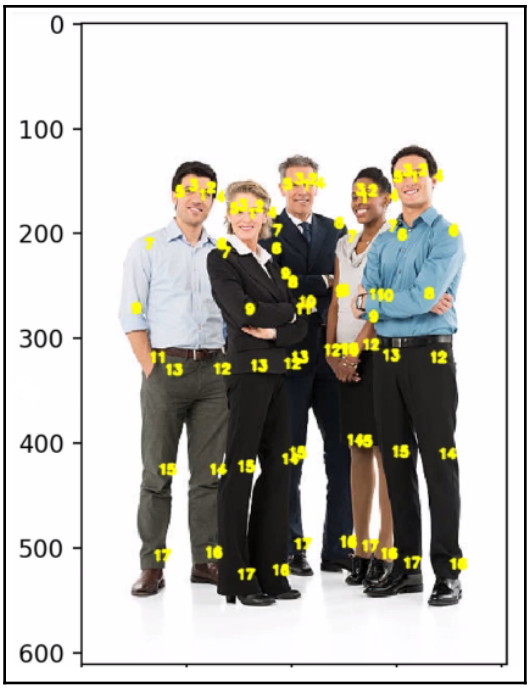

0
100
200
300
400
500
600

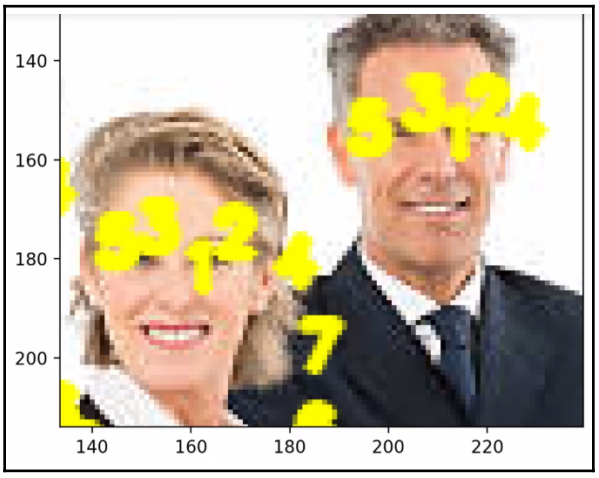



```
Populating the interactive namespace from numpy and matplotlib
```

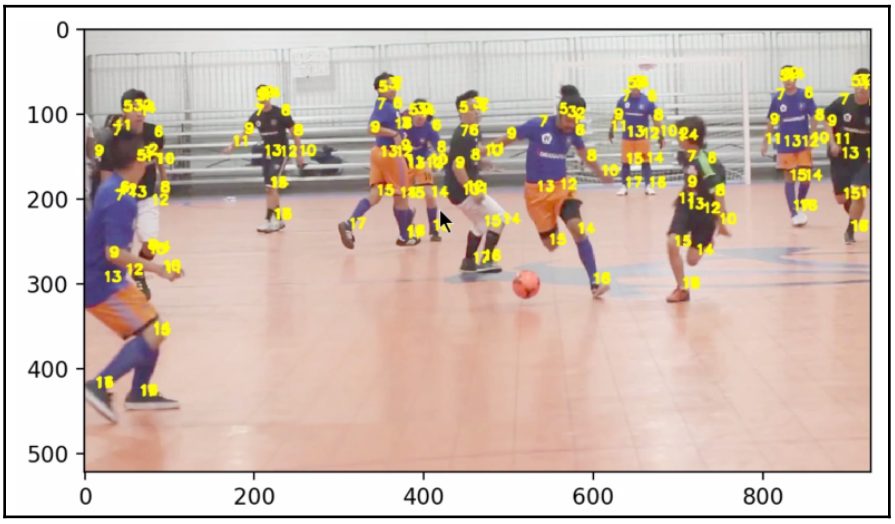
```
C:\Users\mrever\Anaconda3\lib\site-packages\IPython\core\magics\pylab.py:160:
UserWarning: pylab import has clobbered these variables: ['imread', 'imsave']
`%matplotlib` prevents importing * from pylab and numpy
  "\n`%matplotlib` prevents importing * from pylab and numpy"
```

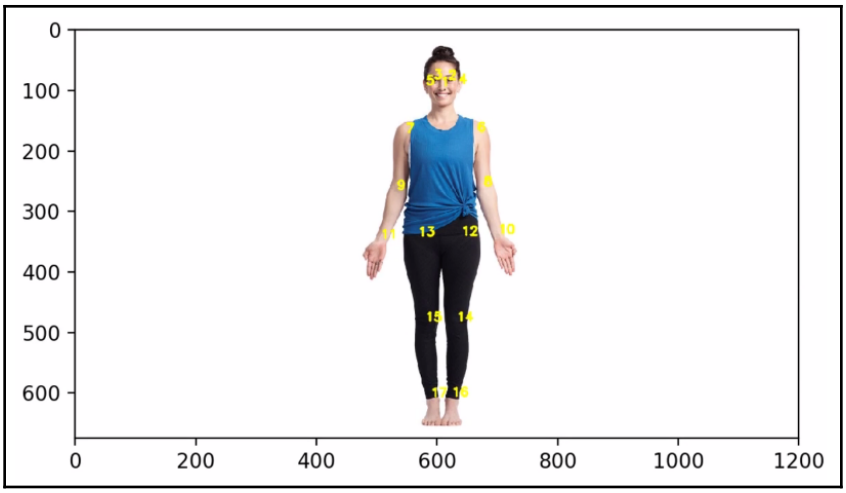
```
INFO:tensorflow:Restoring parameters from models/coco/coco-resnet-101
```


0
100
200
300
400
500
600


140
160
180
200
140
160
180
200
220


0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750
2000
2250


0
100
200
300
400
500
0
200
400
600
800



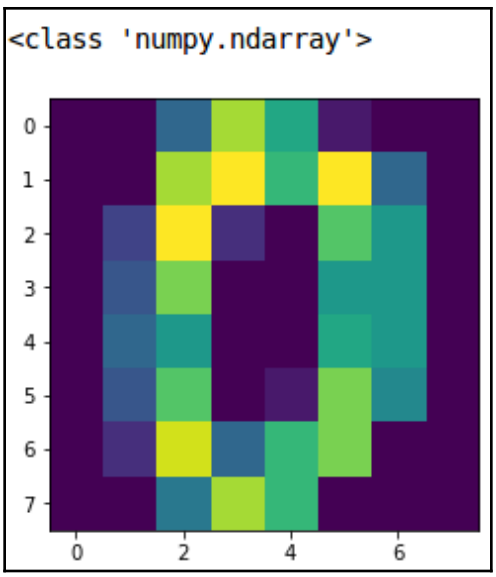
```
num_people:  1
```

Chapter 05:Handwritten Digit Recognition with scikit-learn and TensorFlow

```
#what kind of data do we already have?
from sklearn import datasets
digits=datasets.load_digits()

datasets.
example_
print(ty
plt.imsh
example_

#acquire
#http://

data_dir
```

```
base
california_housing
clear_data_home
covtype
data
descr
dump_svmlight_file
fetch_20newsgroups
fetch_20newsgroups_vectorized
fetch_california_housing
```

```
<class 'numpy.ndarray'>
```

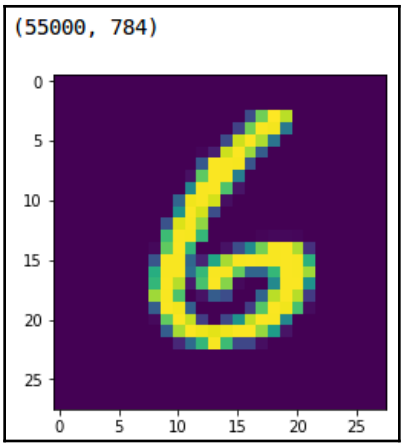


```
Out[2]: array([[ 0.],
               [ 0.],
               [ 5.],
               [13.],
               [ 9.],
               [ 1.],
               [ 0.],
               [ 0.],
               [ 0.],
               [ 0.],
               [13.],
               [15.],
               [10.],
               [15.],
               [ 5.],
               [ 0.],
               [ 0.],
               [ 3.],
               [15.],
               [ 2.],
               [ 0.],
               [11.],
               [ 8.],
               [ 0.],
               [ 0.],
               [ 4.],
               [12.],
```

```
#acquire standard MNIST handwritten digit data
#http://yann.lecun.com/exdb/mnist/

data_dir = '/tmp/tensorflow/mnist/input_data'
mnist = input_data.read_data_sets(data_dir, one_hot=True)
```

```
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
Extracting /tmp/tensorflow/mnist/input_data/train-images-idx3-ubyte.gz
Successfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.
Extracting /tmp/tensorflow/mnist/input_data/train-labels-idx1-ubyte.gz
Successfully downloaded t10k-images-idx3-ubyte.gz 1648877 bytes.
Extracting /tmp/tensorflow/mnist/input_data/t10k-images-idx3-ubyte.gz
Successfully downloaded t10k-labels-idx1-ubyte.gz 4542 bytes.
Extracting /tmp/tensorflow/mnist/input_data/t10k-labels-idx1-ubyte.gz
```

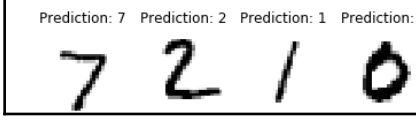
```
(55000, 784)
```

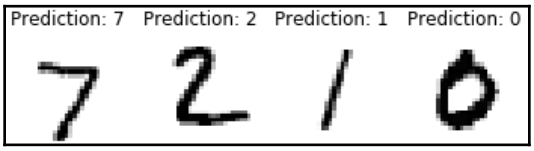


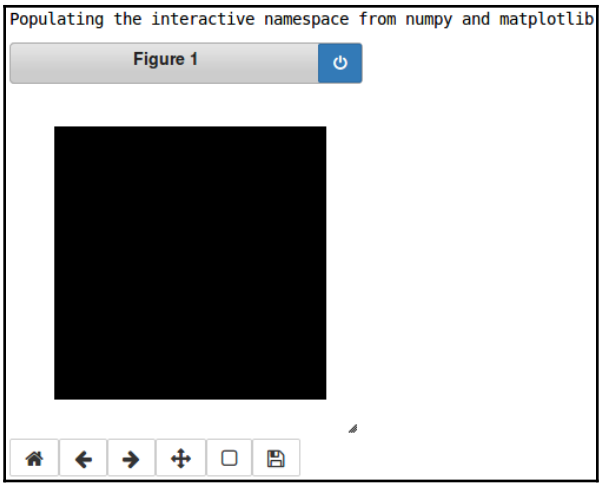

```
# Create a classifier: a support vector classifier
classifier = svm.SVC(gamma=0.001)
# Learn about gamma and other SVM parameters here:
# http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html
# Exercise:  Experiment with the parameters to see how they affect execution
#             time and accuracy

# Train the model -- we're only going to use the training data (and not
# the test data) to ensure that our model generalizes to unseen cases.
# This (training) is typically what takes the most computational time
# when doing machine learning.
classifier.fit(train_data, train_labels)
```

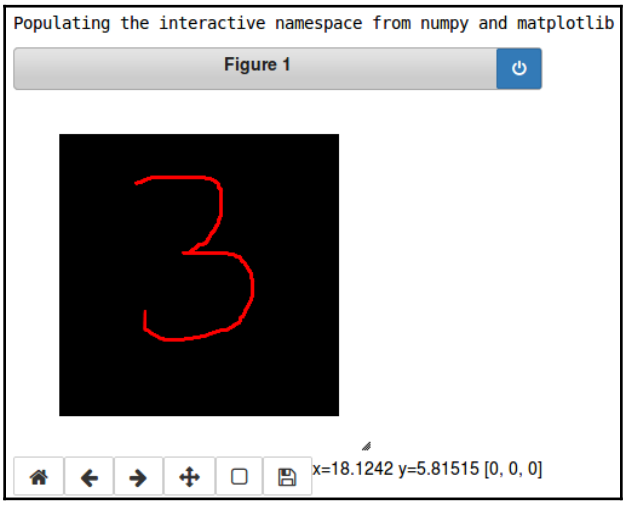
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

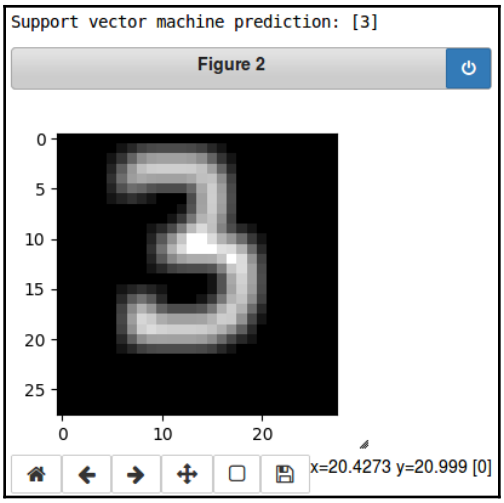
```
See https://en.wikipedia.org/wiki/Precision_and_recall to understand metric definitions
Classification report for classifier SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.96      0.99      0.97       980
           1       0.96      0.99      0.97      1135
           2       0.94      0.92      0.93      1032
           3       0.92      0.94      0.93      1010
           4       0.92      0.95      0.94       982
           5       0.93      0.90      0.91       892
           6       0.94      0.96      0.95       958
           7       0.95      0.93      0.94      1028
           8       0.93      0.91      0.92       974
           9       0.94      0.91      0.92      1009

   micro avg       0.94      0.94      0.94     10000
   macro avg       0.94      0.94      0.94     10000
weighted avg       0.94      0.94      0.94     10000


Confusion matrix:
[[ 967    0    1    0    0    5    5    1    1    0]
 [   0 1118    2    3    0    1    3    1    7    0]
 [   9    1  951   10   12    1   15    9   22    2]
 [   0    2   16  947    1   17    1   11   11    4]
 [   1    2    7    0  931    0    8    2    3   28]
 [   7    5    5   34    8  804   12    2   10    5]
 [   9    3    4    1    6   11  923    0    1    0]
 [   2   14   21    5    9    0    0  953    3   21]
 [   4    7    7   15    8   24   10    7  889    3]
 [  10    7    0   12   33    6    1   14    6  920]]
```






Populating the interactive namespace from numpy and matplotlib
Figure 1


Populating the interactive namespace from numpy and matplotlib
Figure 1
x=18.1242 y=5.81515 [0, 0, 0]


Support vector machine prediction: [3]
Figure 2
0
5
10
15
20
25
0
10
20
x=20.4273 y=20.999 [0]

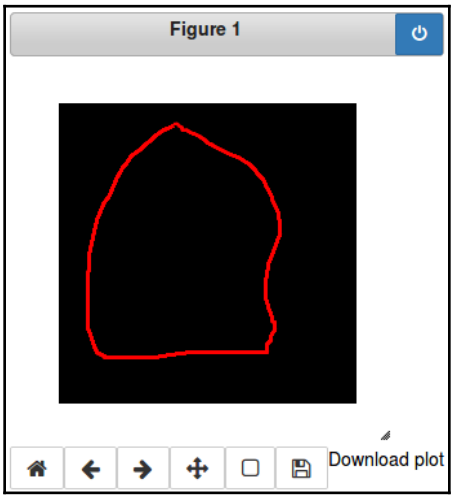
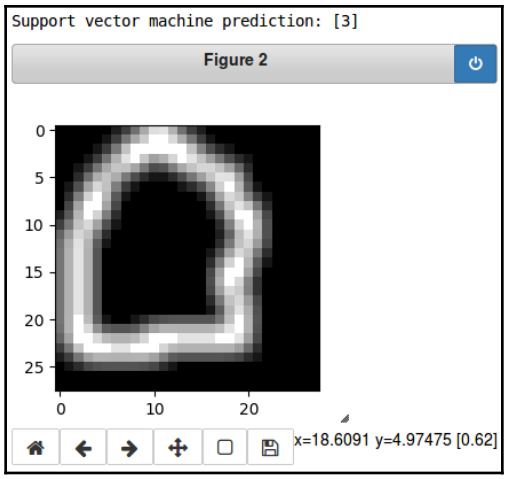
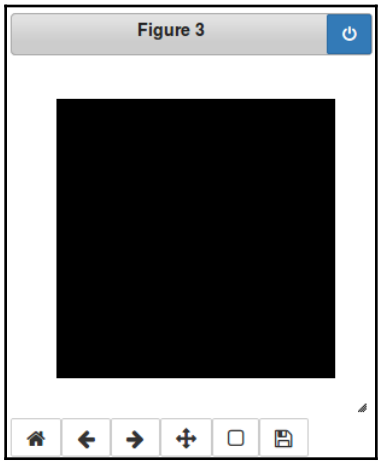


Figure 1
Download plot




```
Extracting /tmp/tensorflow/mnist/input_data/train-images-idx3-ubyte.gz
Extracting /tmp/tensorflow/mnist/input_data/train-labels-idx1-ubyte.gz
Extracting /tmp/tensorflow/mnist/input_data/t10k-images-idx3-ubyte.gz
Extracting /tmp/tensorflow/mnist/input_data/t10k-labels-idx1-ubyte.gz
WARNING:tensorflow:From <ipython-input-18-b2f1671e64b3>:16: softmax_cross_entropy_with_logits (from tensorflow.python.op
s.nn_ops) is deprecated and will be removed in a future version.
Instructions for updating:

Future major versions of TensorFlow will allow gradients to flow
into the labels input on backprop by default.

See tf.nn.softmax_cross_entropy_with_logits_v2.

Model accuracy: 0.918
```



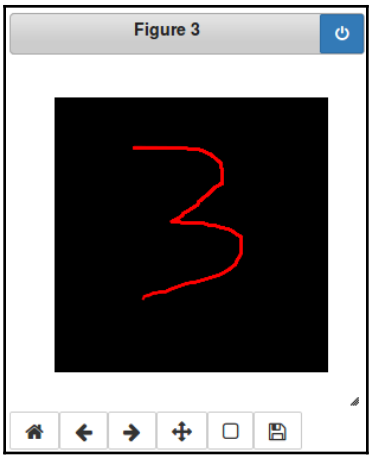
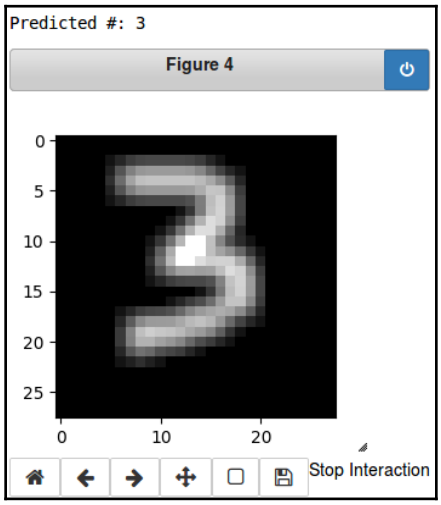


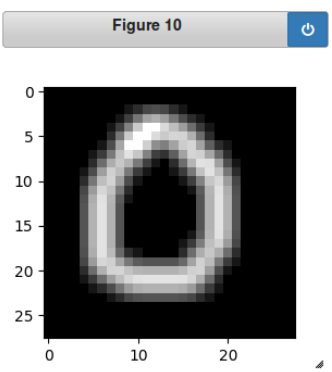
Figure 3


Predicted #: 3
Figure 4
0
5
10
15
20
25
0
10
20
Stop Interaction

```
Extracting /tmp/tensorflow/mnist/input_data/train-images-idx3-ubyte.gz
Extracting /tmp/tensorflow/mnist/input_data/train-labels-idx1-ubyte.gz
Extracting /tmp/tensorflow/mnist/input_data/t10k-images-idx3-ubyte.gz
Extracting /tmp/tensorflow/mnist/input_data/t10k-labels-idx1-ubyte.gz
Saving graph to: /tmp/tmpdr9rhjcj
step 0, training accuracy 0.1
step 100, training accuracy 0.86
step 200, training accuracy 0.92
step 300, training accuracy 0.94
step 400, training accuracy 0.94
step 500, training accuracy 0.92
step 600, training accuracy 0.9
step 700, training accuracy 0.88
step 800, training accuracy 0.92
step 900, training accuracy 0.96
```

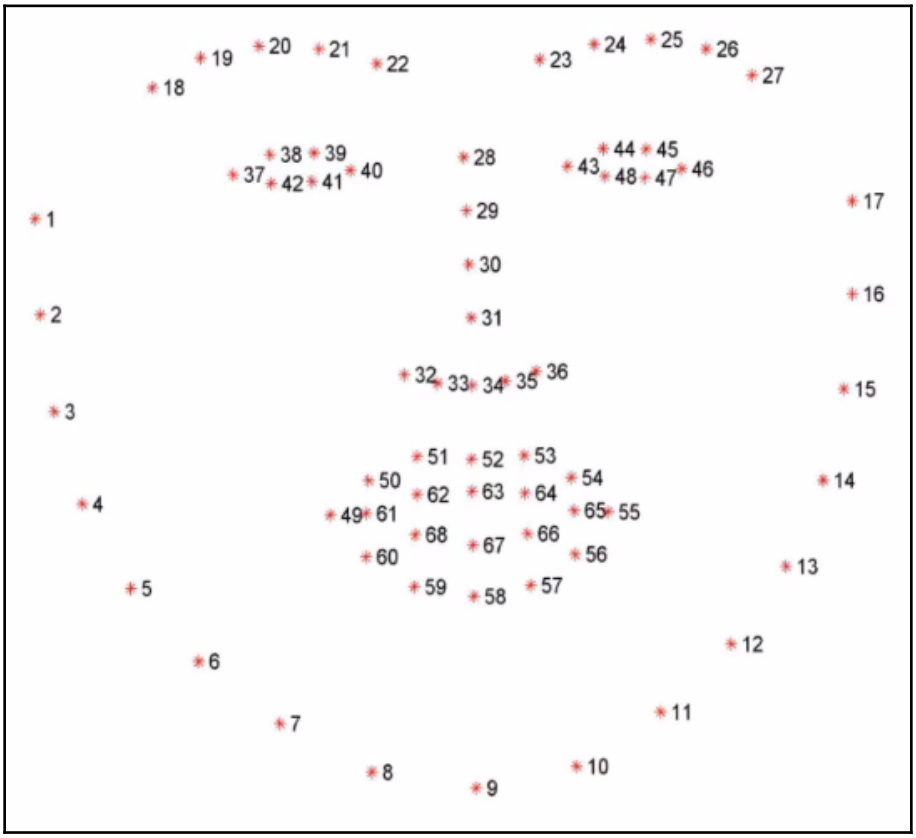
```
step 18500, training accuracy 1
step 18600, training accuracy 1
step 18700, training accuracy 1
step 18800, training accuracy 1
step 18900, training accuracy 1
step 19000, training accuracy 1
step 19100, training accuracy 1
step 19200, training accuracy 1
step 19300, training accuracy 1
step 19400, training accuracy 1
step 19500, training accuracy 1
step 19600, training accuracy 1
step 19700, training accuracy 1
step 19800, training accuracy 1
step 19900, training accuracy 1
test accuracy 0.9933
```

```
Predicted #: 0
```

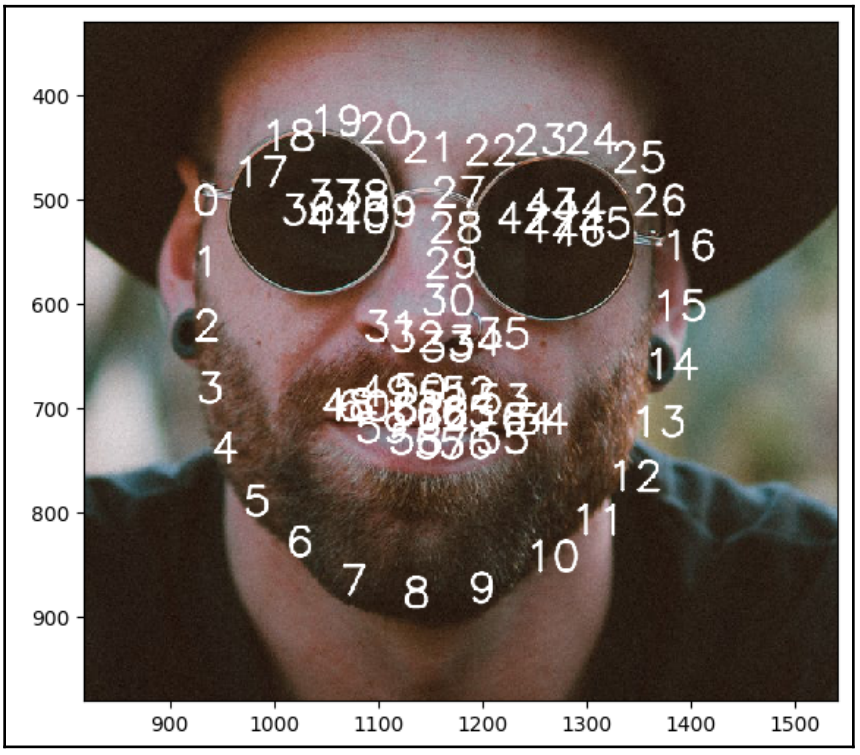


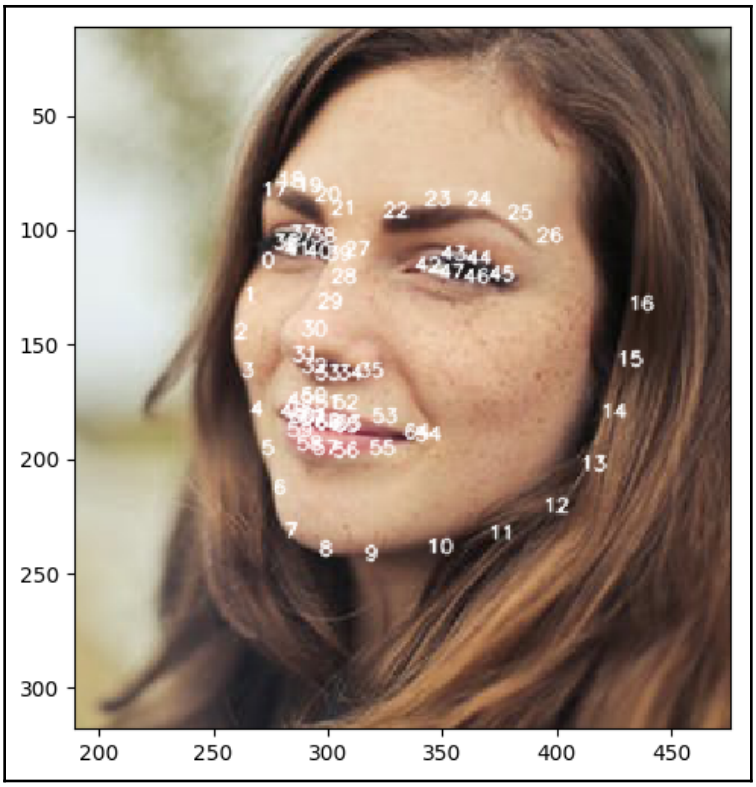

```
<matplotlib.image.AxesImage at 0x7f2ef9f859b0>
```

Chapter 06: Facial Feature Tracking and Classification with dlib

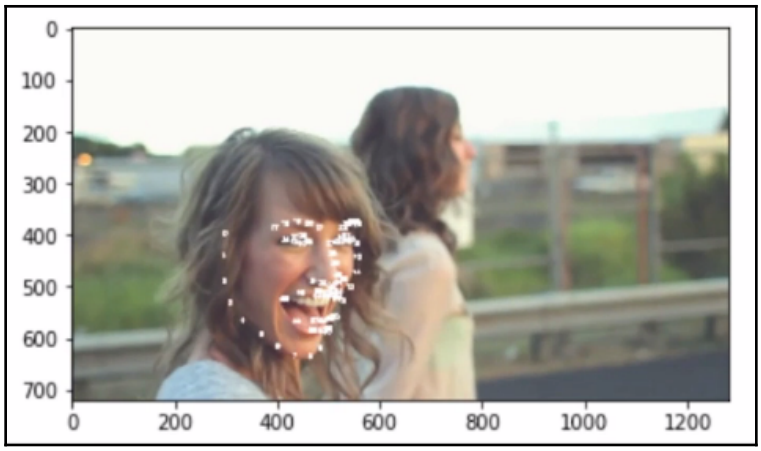


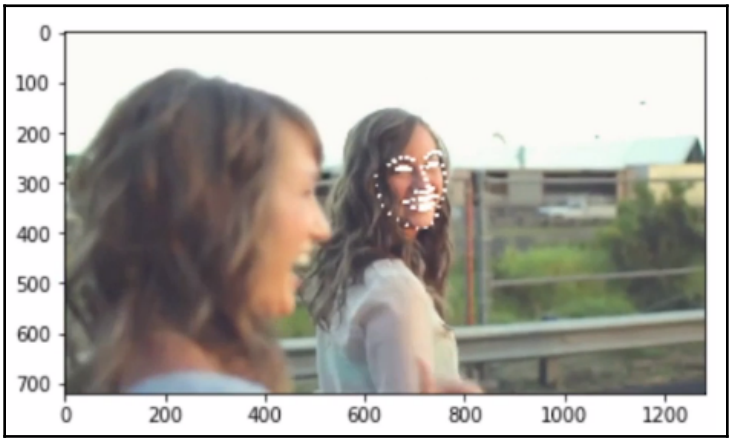

```
Populating the interactive namespace from numpy and matplotlib
/home/test/13293
```

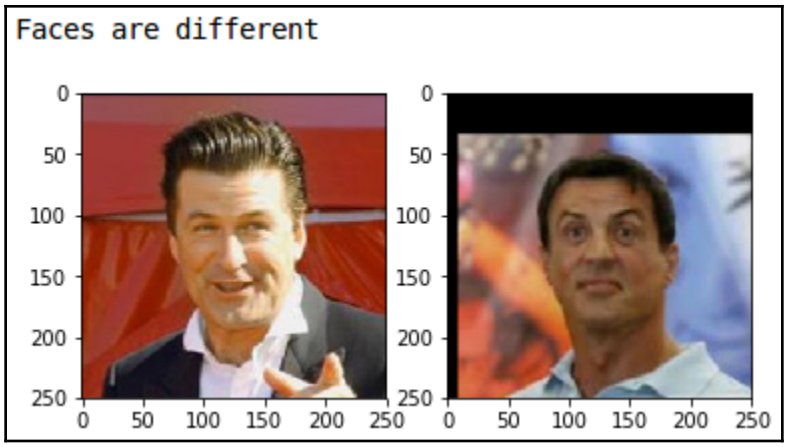


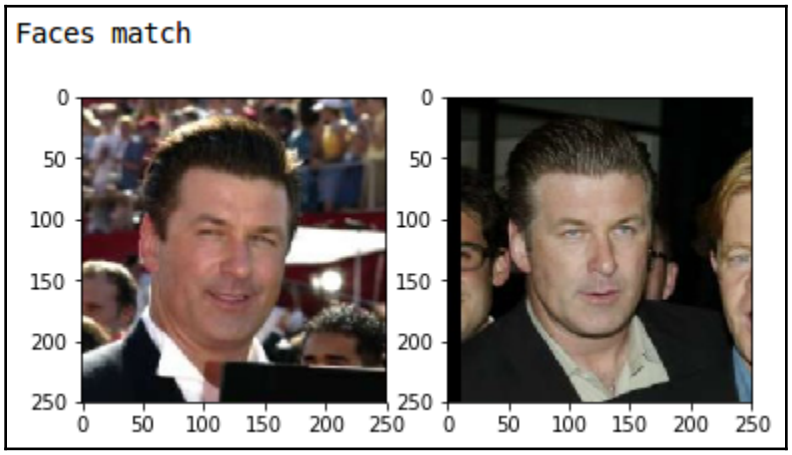

50
100
150
200
250
300
200
250
300
350
400
450

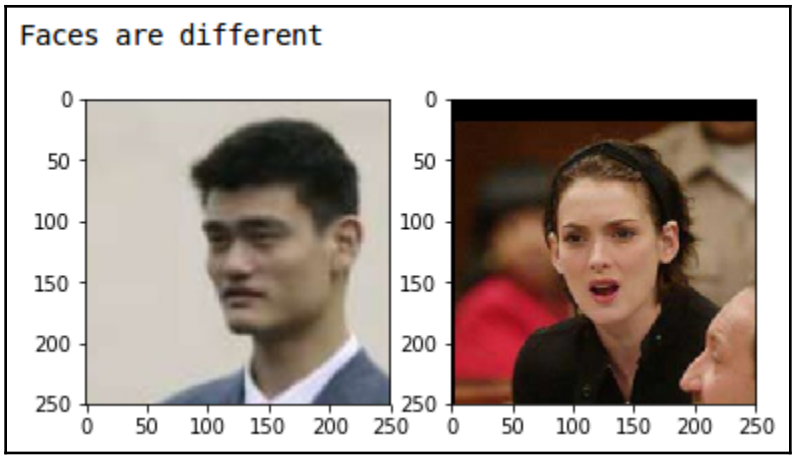

600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750
2000
2250


0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200


0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200


Faces are different
0
50
100
150
200
250
0
50
100
150
200
250
0
50
100
150
200
250
0
50
100
150
200
250


Faces match
0
50
100
150
200
250
0
50
100
150
200
250
0
50
100
150
200
250
0
50
100
150
200
250


Faces are different
0
50
100
150
200
250


Faces match
0
50
100
150
200
250

```
Type:           vector
String form:
-0.147528
           0.048263
           0.0316406
           -0.124176
           -0.0820134
           -0.0260086
           -0.0338606
           -0.10674
           0.166148
           -0.1133 <...> 79
           0.114875
           -0.0459535
           -0.0619625
           -0.120842
           -0.0212056
           -0.0608454
           -0.101526
           0.0224088
```

Chapter 07: Deep Learning Image Classification with TensorFlow

```
with tf.Session() as sess:
    print("a+b=" + str(sess.run(add, feed_dict={a: 2, b: 3})))
    print("a*b=" + str(sess.run(mul, feed_dict={a: 2, b: 3})))
```

```
a=2, b=3
a+b=5
a*b=6
a+b=5
```

```
#Matrix multiplication
matrix1 = tf.constant([[1., 2.],[9.0,3.14159]])
matrix2 = tf.constant([[3.],[4.]])

product = tf.matmul(matrix1, matrix2)

with tf.Session() as sess:
    result = sess.run(product)
    print(result)
```

```
[[11.     ]
 [39.56636]]
```

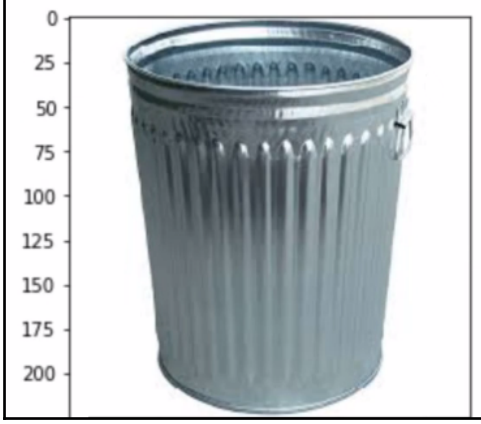
inception_1000_class_list.txt

```
{0: 'tench, Tinca tinca',
 1: 'goldfish, Carassius auratus',
 2: 'great white shark, white shark, man-eater, man-eating shark, Carcharodon carcharias',
 3: 'tiger shark, Galeocerdo cuvieri',
 4: 'hammerhead, hammerhead shark',
 5: 'electric ray, crampfish, numbfish, torpedo',
 6: 'stingray',
 7: 'cock',
 8: 'hen',
 9: 'ostrich, Struthio camelus',
 10: 'brambling, Fringilla montifringilla',
 11: 'goldfinch, Carduelis carduelis',
 12: 'house finch, linnet, Carpodacus mexicanus',
 13: 'junco, snowbird',
 14: 'indigo bunting, indigo finch, indigo bird, Passerina cyanea',
 15: 'robin, American robin, Turdus migratorius',
 16: 'bulbul',
 17: 'jay',
 18: 'magpie',
 19: 'chickadee',
 20: 'water ouzel, dipper',
 21: 'kite',
 22: 'bald eagle, American eagle, Haliaeetus leucocephalus',
 23: 'vulture',
 24: 'great grey owl, great gray owl, Strix nebulosa',
 25: 'European fire salamander, Salamandra salamandra',
 26: 'common newt, Triturus vulgaris',
 27: 'eft',
 28: 'spotted salamander, Ambystoma maculatum',
 29: 'axolotl, mud puppy, Ambystoma mexicanum',
 30: 'bullfrog, Rana catesbeiana',
```

```
giant panda, panda, panda bear, coon bear, Ailuropoda melanoleuca (score = 0.89
632)
```

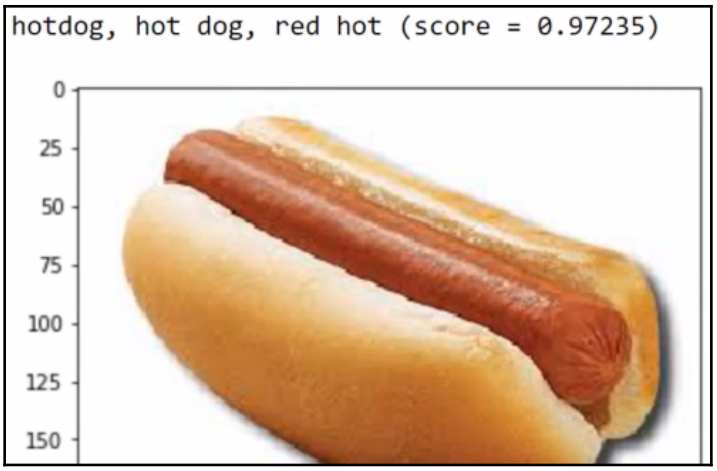
```
mountain bike, all-terrain bike, off-roader (score = 0.91317)
```

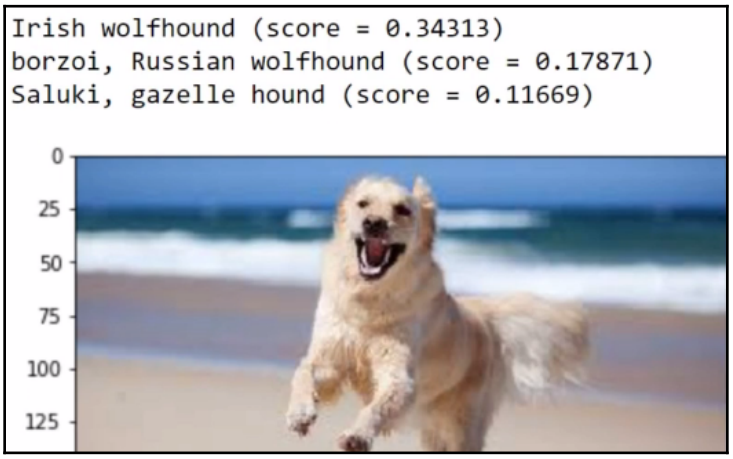

```
ashcan, trash can, garbage can, wastebin, ash bin, ash-bin, ashbin, dustbin, tr
ash barrel, trash bin (score = 0.66923)
```

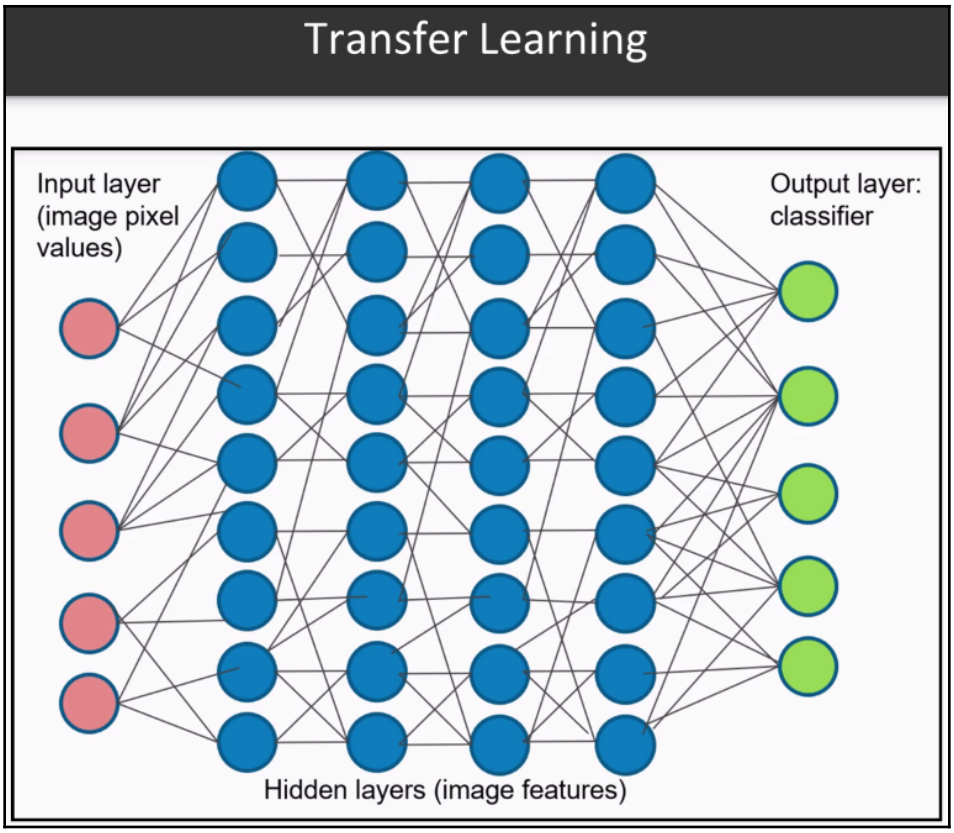



wood rabbit, cottontail, cottontail rabbit (score = 0.87354)
0
20
40
60
80
100
120


trombone (score = 0.99370)
0
20
40
60
80
100
120
140
160


hotdog, hot dog, red hot (score = 0.97235)
0
25
50
75
100
125
150


Irish wolfhound (score = 0.34313)
borzoi, Russian wolfhound (score = 0.17871)
Saluki, gazelle hound (score = 0.11669)
0
25
50
75
100
125



barbie	12/20/2018 11:17 ...	File folder
gi joe	12/20/2018 11:17 ...	File folder
my little pony	12/20/2018 11:17 ...	File folder
transformers	12/20/2018 11:21 ...	File folder

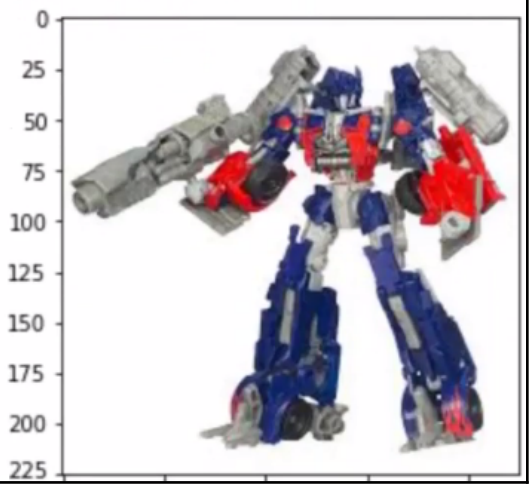
```
#pull the function from our custom retrain.py file
from retrain import retrain

#Now we'll train our model and generate our model/graph file 'output_graph.pb'
retrain('toy_images')
```

```
Retraining with images in directory: toy_images
Converted 378 variables to const ops.
```

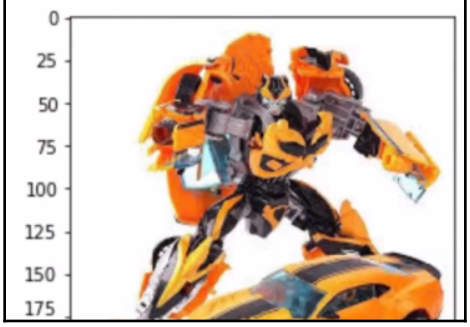
```
#Confirm that it worked
!ls *.pb

#should see file "output_graph.pb
```

```
classify_image_graph_def.pb
output_graph - Copy.pb
output_graph.pb
```

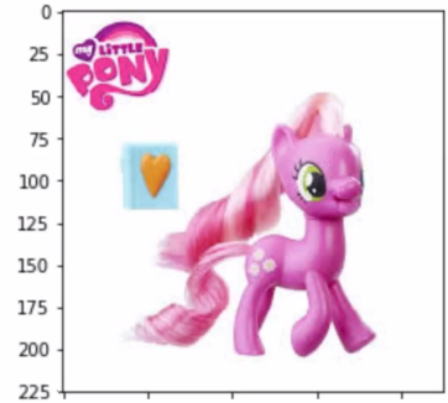
```
transformers 0.999356
gi joe 0.0005511869
barbie 4.677048e-05
my little pony 4.608043e-05
```



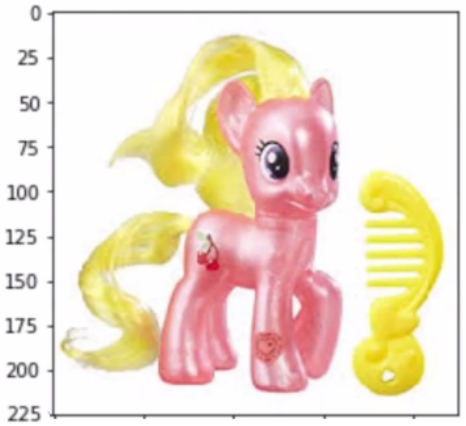
```
transformers 0.98937875
barbie 0.0064697447
gi joe 0.003952643
my little pony 0.00019881024
```



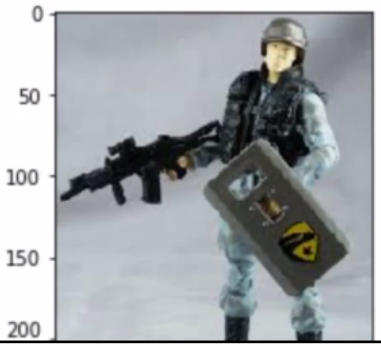

```
my little pony 0.99906
barbie 0.00081196585
transformers 8.224221e-05
gi joe 4.580088e-05
```

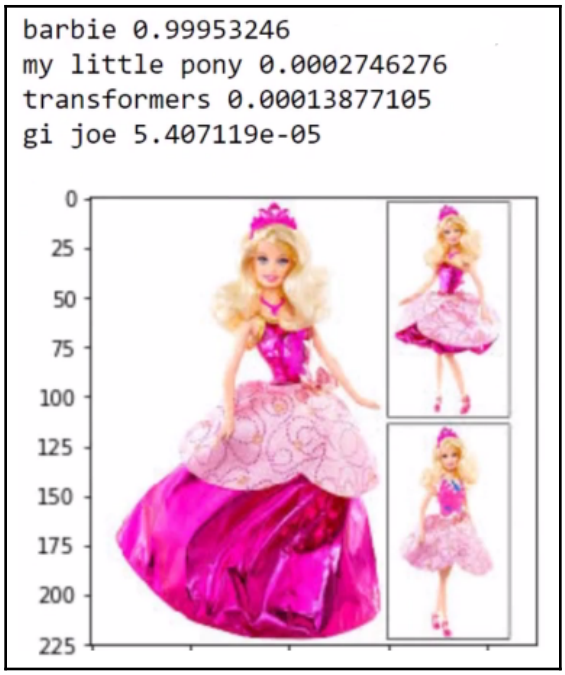


```
my little pony 0.9998729
barbie 0.00011279018
transformers 7.585655e-06
gi joe 6.6580324e-06
```




```
gi joe 0.9767354
transformers 0.020028114
barbie 0.0028766098
my little pony 0.00035978633
```






```
In [2]: print(tensorflow.Session())
2018-04-29 23:40:34.654575: I T:\src\github\tensorflow\tensorflow\core\platform\cpu_feature_guard.cc:140] Your CPU supports instructions that
this TensorFlow binary was not compiled to use: AVX2
2018-04-29 23:40:35.152570: I T:\src\github\tensorflow\tensorflow\core\common_runtime\gpu\gpu_device.cc:1356] Found device 0 with properties:

name: GeForce GTX 970M major: 5 minor: 2 memoryClockRate(GHz): 1.038
pciBusID: 0000:01:00.0
totalMemory: 6.00GiB freeMemory: 4.90GiB
```