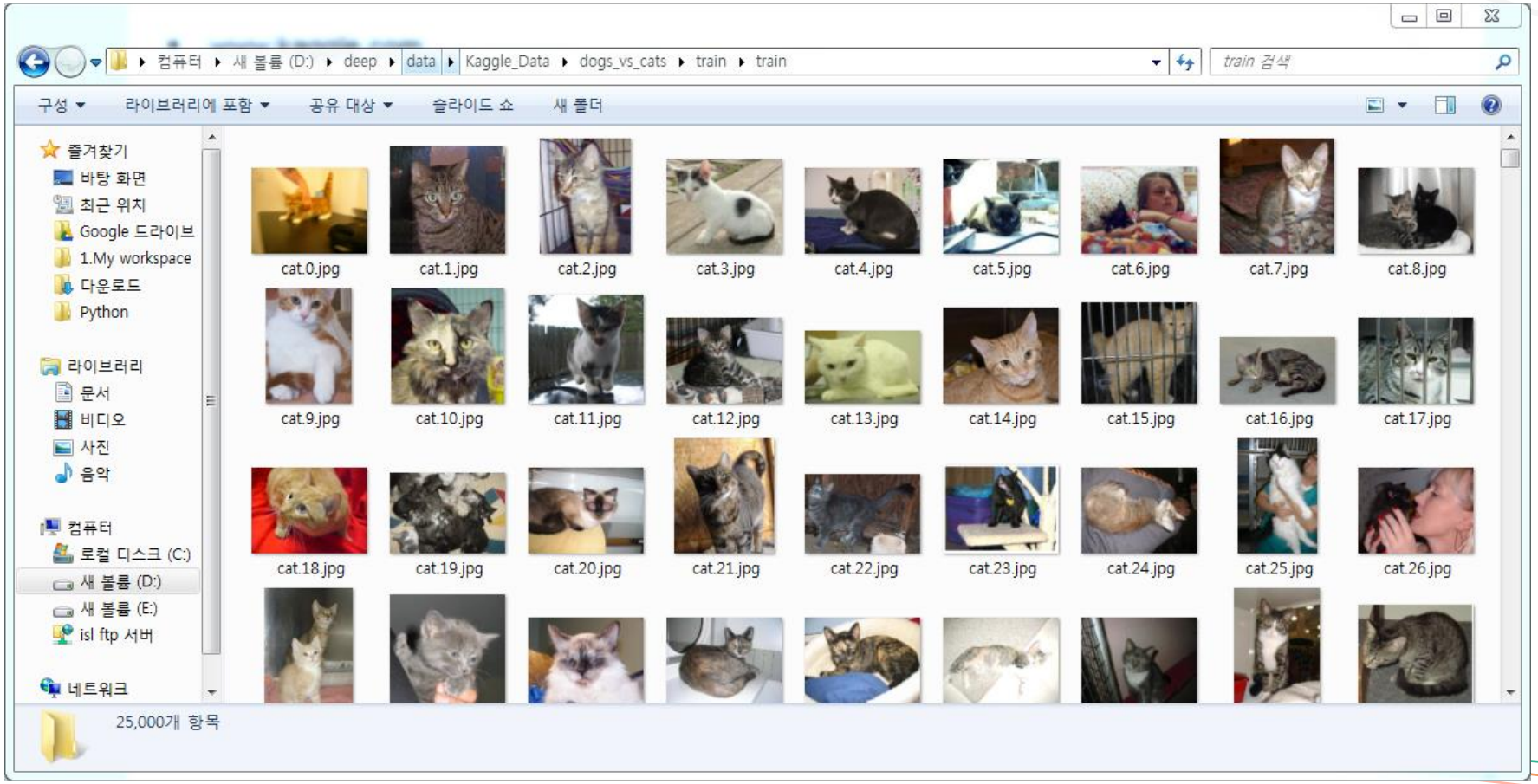
Object Detection

전현호

Contents

Dogs vs. Cats

- 각각 12,500장의 학습 데이터 (총 25,000장)
- 12,500장의 테스트 데이터



Dogs vs. Cats

In [1]:
```
import cv2
import numpy as np
import os
from random import shuffle
from tqdm import tqdm
```

In [2]:
```
TRAIN_DIR = 'D:/deep/data/Kaggle_Data/dogs_vs_cats/train/train'
TEST_DIR = 'D:/deep/data/Kaggle_Data/dogs_vs_cats/test/test'
IMG_SIZE = 50
LR = 1e-3

MODEL_NAME = 'dogsvscats-{}-{}.model'.format(LR, '5conv')
```

In [3]:
```
def label_img(img):
    word_label = img.split('.')[-3]
    if word_label == 'cat': return [1,0]
    elif word_label == 'dog': return [0,1]
```

In [4]:
```
def create_train_data():
    training_data = []
    for img in tqdm(os.listdir(TRAIN_DIR)):
        label = label_img(img)
        path = os.path.join(TRAIN_DIR,img)
        img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        training_data.append([np.array(img),np.array(label)])
    shuffle(training_data)
    np.save('train_data.npy', training_data)
    return training_data
```

Dogs vs. Cats

In [5]:
```
def process_test_data():
    testing_data = []
    for img in tqdm(os.listdir(TEST_DIR)):
        path = os.path.join(TEST_DIR,img)
        img_num = img.split('.')[0]
        img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        testing_data.append([np.array(img), img_num])

    shuffle(testing_data)
    np.save('test_data.npy', testing_data)
    return testing_data
```

In [6]:
```
#train_data = create_train_data()
# If you have already created the dataset:
train_data = np.load('train_data.npy')
```

Dogs vs. Cats

In [7]:

```
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression

import tensorflow as tf
tf.reset_default_graph()

convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 128, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = fully_connected(convnet, 1024, activation='relu')
convnet = dropout(convnet, 0.8)

convnet = fully_connected(convnet, 2, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=LR, loss='categorical_crossentropy', name='targets')

model = tflearn.DNN(convnet, tensorboard_dir='log')
```

Dogs vs. Cats

- 비교

```
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression

import tensorflow as tf
tf.reset_default_graph()

convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 128, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = fully_connected(convnet, 1024, activation='relu')
convnet = dropout(convnet, 0.8)

convnet = fully_connected(convnet, 2, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=LR, loss='categorical_crossentropy', name='targets')

model = tflearn.DNN(convnet, tensorboard_dir='log')
```

TF-learn
5conv net

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("/tmp/data/", one_hot = True)

n_nodes_hl1 = 500
n_nodes_hl2 = 500
n_nodes_hl3 = 500

n_classes = 10
batch_size = 100

x = tf.placeholder('float', [None, 784])
y = tf.placeholder('float')


hidden_1_layer = {'weights':tf.Variable(tf.random_normal([784, n_nodes_hl1])),
                  'biases':tf.Variable(tf.random_normal([n_nodes_hl1]))}

hidden_2_layer = {'weights':tf.Variable(tf.random_normal([n_nodes_hl1, n_nodes_hl2])),
                  'biases':tf.Variable(tf.random_normal([n_nodes_hl2]))}

hidden_3_layer = {'weights':tf.Variable(tf.random_normal([n_nodes_hl2, n_nodes_hl3])),
                  'biases':tf.Variable(tf.random_normal([n_nodes_hl3]))}

output_layer = {'weights':tf.Variable(tf.random_normal([n_nodes_hl3, n_classes])),
                'biases':tf.Variable(tf.random_normal([n_classes])),}


l1 = tf.add(tf.matmul(x,hidden_1_layer['weights']), hidden_1_layer['biases'])
l1 = tf.nn.relu(l1)

l2 = tf.add(tf.matmul(l1,hidden_2_layer['weights']), hidden_2_layer['biases'])
l2 = tf.nn.relu(l2)

l3 = tf.add(tf.matmul(l2,hidden_3_layer['weights']), hidden_3_layer['biases'])
l3 = tf.nn.relu(l3)

output = tf.matmul(l3,output_layer['weights']) + output_layer['biases']


prediction = output
cost = tf.reduce_mean( tf.nn.softmax_cross_entropy_with_logits(prediction,y) )
optimizer = tf.train.AdamOptimizer().minimize(cost)

hm_epochs = 10
sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

Tensorflow
3 hidden layer

+Session ~~~

Dogs vs. Cats

In [8]:
```
if os.path.exists('{}.meta'.format(MODEL_NAME)):
    model.load(MODEL_NAME)
    print('model loaded!')
```

```
model loaded!
```

In [9]:
```
train = train_data[:-500]
test = train_data[-500:]
```

In [10]:
```
print(train.shape)
X = np.array([i[0] for i in train]).reshape(-1,IMG_SIZE,IMG_SIZE,1)
Y = [i[1] for i in train]

test_x = np.array([i[0] for i in test]).reshape(-1,IMG_SIZE,IMG_SIZE,1)
test_y = [i[1] for i in test]
```

Validation set 생성

```
(24500, 2)
```

In [11]:
```
model.fit({'input': X}, {'targets': Y}, n_epoch=30, validation_set=({'input': test_x}, {'targets': test_y}),
    snapshot_step=500, show_metric=True, run_id=MODEL_NAME)
#tensorboard --logdir=foo:d:#deep#log
```

```
Training Step: 11489  | total loss: 0.07476 | time: 6.873s
| Adam | epoch: 030 | loss: 0.07476 - acc: 0.9727 -- iter: 24448/24500
Training Step: 11490  | total loss: 0.07033 | time: 7.907s
| Adam | epoch: 030 | loss: 0.07033 - acc: 0.9738 | val_loss: 1.51778 - val_acc: 0.7540 -- iter: 24500/24500
--
```

In [12]:
```
model.save(MODEL_NAME)
```

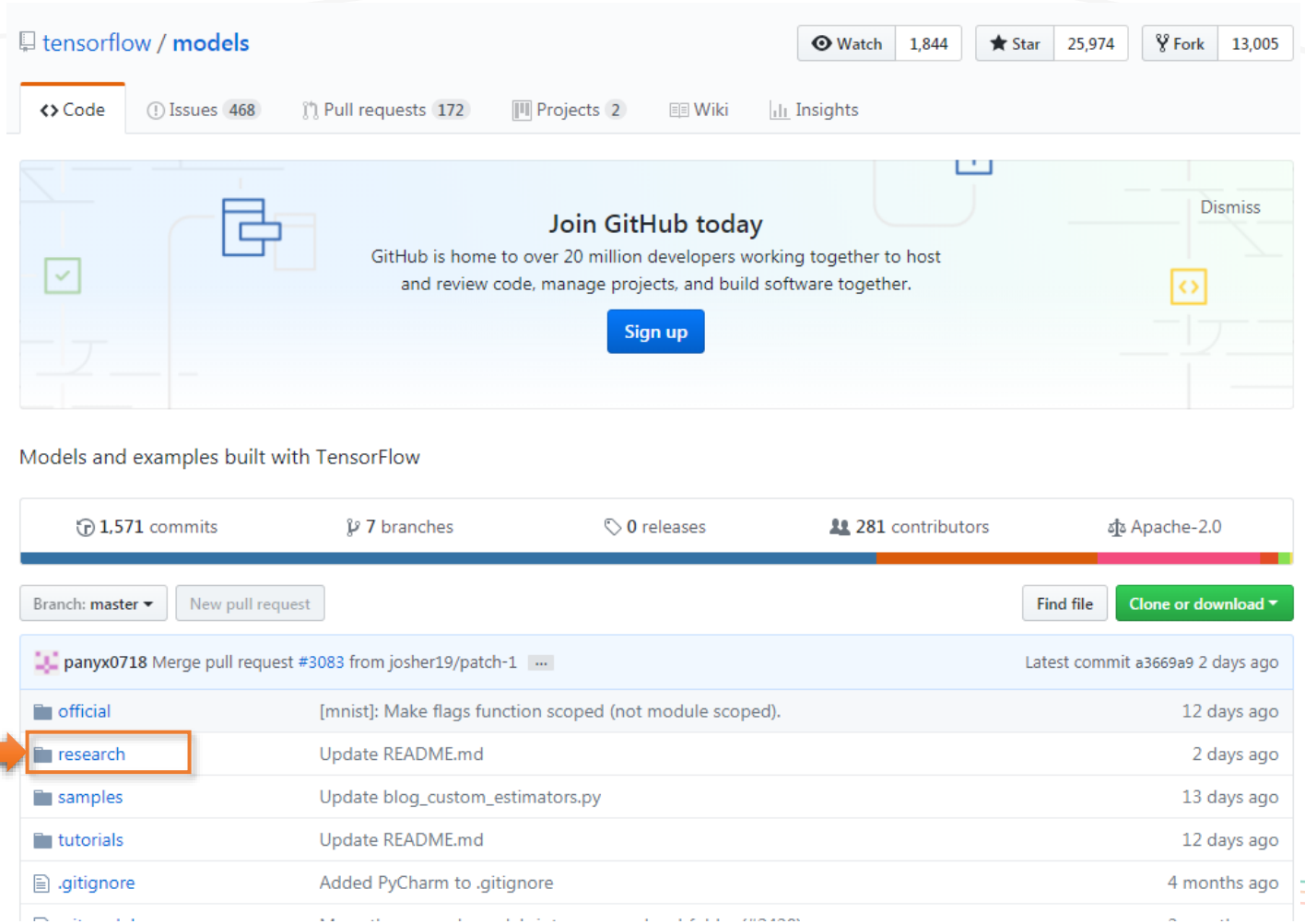
```
INFO:tensorflow:D:#deep#dogsvscats-0.001-5conv-basic-video.model is not in all_model_checkpoint_paths. Manually adding it.
```

Dogs vs. Cats

In [13]:
```
import matplotlib.pyplot as plt

#if you dont have this file yet
#test_data = process_test_data()
#if you already have it
test_data = np.load('test_data.npy')

fig = plt.figure()

for num, data, in enumerate(test_data[:12]):
    # cat : [1,0]
    # dog : [0,1]

    img_num = data[1]
    img_data = data[0]

    y = fig.add_subplot(3, 4, num+1)
    orig = img_data
    data = img_data.reshape(IMG_SIZE, IMG_SIZE, 1)

    model_out = model.predict([data])[0]

    if np.argmax(model_out) == 1: str_label = 'Dog'
    else: str_label = 'Cat'

    y.imshow(orig, cmap = 'gray')
    plt.title(str_label)
    y.axes.get_xaxis().set_visible(False)
    y.axes.get_yaxis().set_visible(False)
plt.show()
```

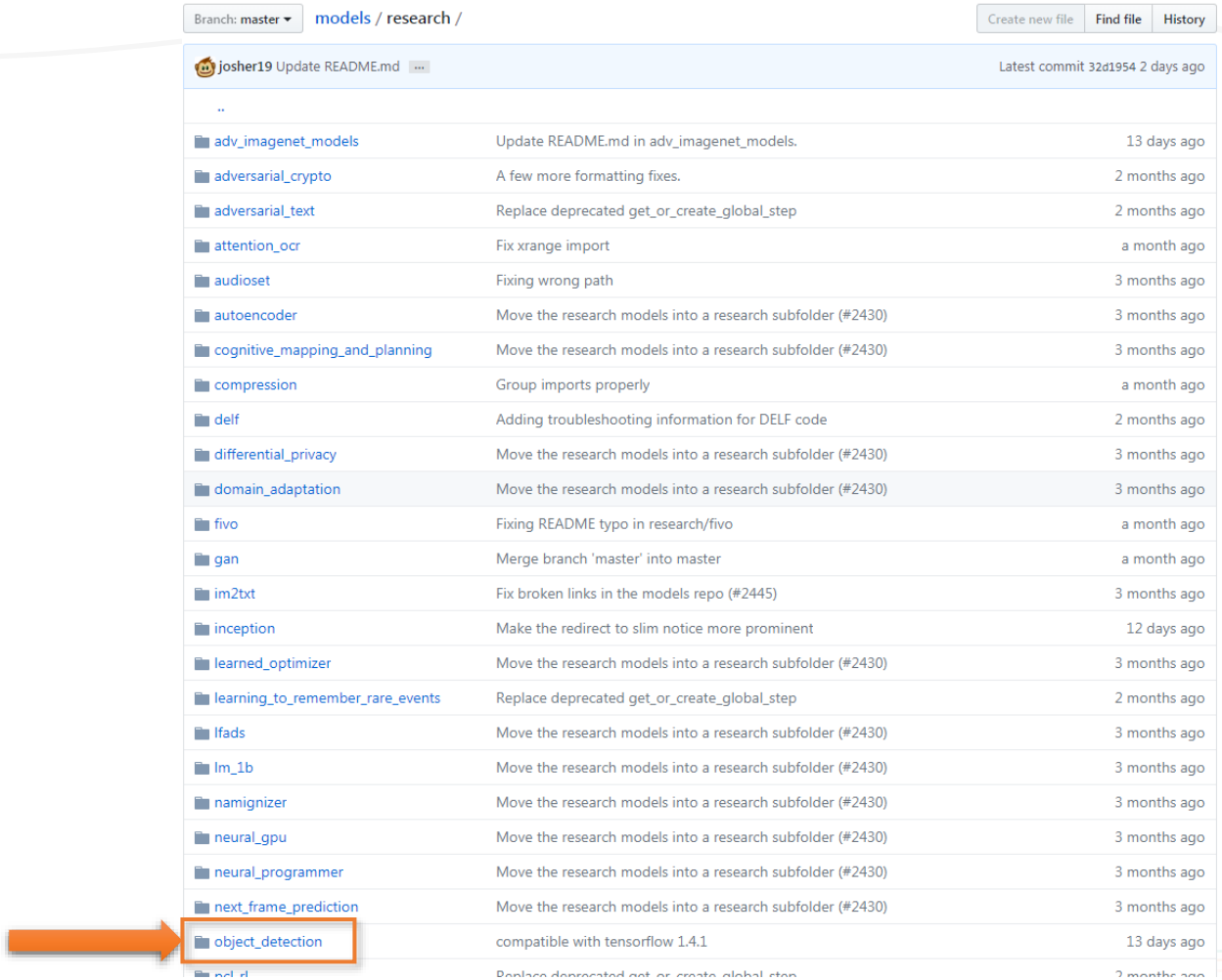
Object detection

- Tensorflow object detection API (https://github.com/tensorflow/models)



Object detection

- Tensorflow object detection API (https://github.com/tensorflow/models)



Branch: master ▾ models / research / Create new file Find file History

josher19 Update README.md ... Latest commit 32d1954 2 days ago

..		
adv_imagenet_models	Update README.md in adv_imagenet_models.	13 days ago
adversarial_crypto	A few more formatting fixes.	2 months ago
adversarial_text	Replace deprecated get_or_create_global_step	2 months ago
attention_ocr	Fix xrange import	a month ago
audioset	Fixing wrong path	3 months ago
autoencoder	Move the research models into a research subfolder (#2430)	3 months ago
cognitive_mapping_and_planning	Move the research models into a research subfolder (#2430)	3 months ago
compression	Group imports properly	a month ago
delf	Adding troubleshooting information for DELF code	2 months ago
differential_privacy	Move the research models into a research subfolder (#2430)	3 months ago
domain_adaptation	Move the research models into a research subfolder (#2430)	3 months ago
fivo	Fixing README typo in research/fivo	a month ago
gan	Merge branch 'master' into master	a month ago
im2txt	Fix broken links in the models repo (#2445)	3 months ago
inception	Make the redirect to slim notice more prominent	12 days ago
learned_optimizer	Move the research models into a research subfolder (#2430)	3 months ago
learning_to_remember_rare_events	Replace deprecated get_or_create_global_step	2 months ago
lfads	Move the research models into a research subfolder (#2430)	3 months ago
lm_1b	Move the research models into a research subfolder (#2430)	3 months ago
namignizer	Move the research models into a research subfolder (#2430)	3 months ago
neural_gpu	Move the research models into a research subfolder (#2430)	3 months ago
neural_programmer	Move the research models into a research subfolder (#2430)	3 months ago
next_frame_prediction	Move the research models into a research subfolder (#2430)	3 months ago
object_detection	compatible with tensorflow 1.4.1	13 days ago

Object detection

- Tensorflow object detection API (https://github.com/tensorflow/models)

Tensorflow Object Detection API

Creating accurate machine learning models capable of localizing and identifying multiple objects in a single image remains a core challenge in computer vision. The TensorFlow Object Detection API is an open source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models. At Google we've certainly found this codebase to be useful for our computer vision needs, and we hope that you will as well.

Contributions to the codebase are welcome and we would love to hear back from you if you find this API useful. Finally if you use the Tensorflow Object Detection API for a research publication, please consider citing:

```
"Speed/accuracy trade-offs for modern convolutional object detectors."
Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z,
Song Y, Guadarrama S, Murphy K, CVPR 2017
```

Object detection

- Tensorflow object detection API (https://github.com/tensorflow/models)

jupyter object_detection_tutorial Last Checkpoint: 10/11/2017 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Markdown CellToolbar

Object Detection Demo

Welcome to the object detection inference walkthrough! This notebook will walk you step by step through the proces
objects in an image. Make sure to follow the installation instructions before you start.

Imports

In [1]:
```
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
```

In [2]:
```
import tensorflow as tf
```

In [3]:
```
import zipfile

from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image
```

Env setup

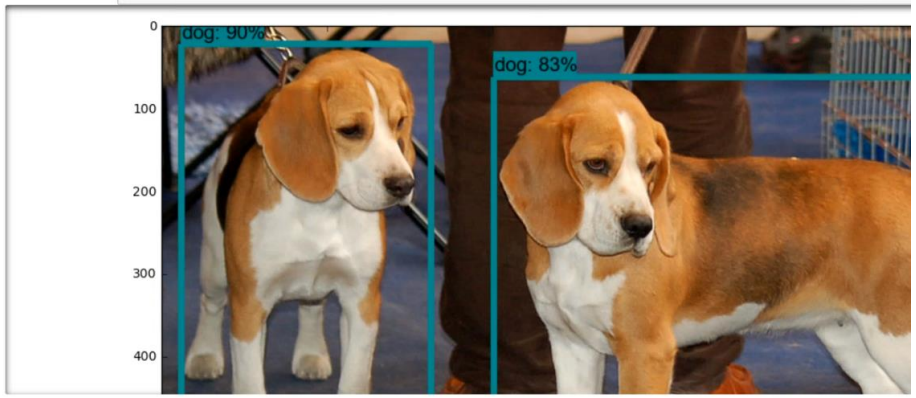
In [4]:
```
# This is needed to display the images.
%matplotlib inline

# This is needed since the notebook is stored in the object_detection folder.
sys.path.append("..")
```

jupyter object_detection_tutorial (unsaved changes)

File Edit View Insert Cell Kernel Help

Code CellToolbar

```
    line_thickness=8)
plt.figure(figsize=IMAGE_SIZE)
plt.imshow(image_np)
```



Object detection

- Adapting to video

In [1]:
```
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
```

In [2]:
```
import tensorflow as tf
```

In [2]:
```
import zipfile

from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image
```

In [3]:
```
import cv2
cap = cv2.VideoCapture(0)
```

Env setup

In [4]:

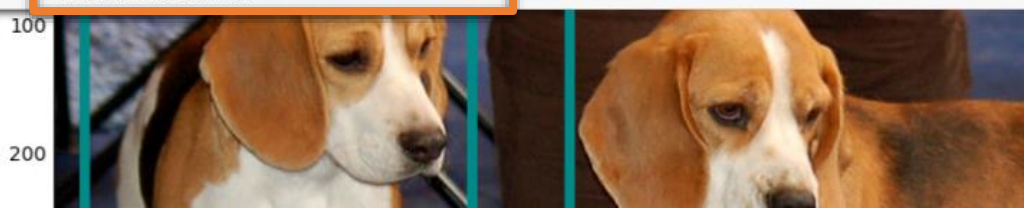

Object detection

- Adapting to video

```
In [14]: with detection_graph.as_default():
           with tf.Session(graph=detection_graph) as sess:
             # Definite input and output Tensors for detection_graph
             image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')
             # Each box represents a part of the image where a particular object was detected.
             detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')
             # Each score represent how level of confidence for each of the objects.
             # Score is shown on the result image, together with the class label.
             detection_scores = detection_graph.get_tensor_by_name('detection_scores:0')
             detection_classes = detection_graph.get_tensor_by_name('detection_classes:0')
             num_detections = detection_graph.get_tensor_by_name('num_detections:0')
             for image_path in TEST_IMAGE_PATHS:
               image = Image.open(image_path)
               # the array based representation of the image
               # result image with boxes and labels on it.
               image_np = load_image_into_numpy_array(image)
               # Expand dimensions since the model expects images to have shape: [1, None, None, 3]
               image_np_expanded = np.expand_dims(image_np, axis=0)
               # Actual detection.
               (boxes, scores, classes, num) = sess.run(
                   [detection_boxes, detection_scores, detection_classes, num_detections],
                   feed_dict={image_tensor: image_np_expanded})
               # Visualization of the results of a detection.
               vis_util.visualize_boxes_and_labels_on_image_array(
                   image_np,
                   np.squeeze(boxes),
                   np.squeeze(classes).astype(np.int32),
                   np.squeeze(scores),
                   category_index,
                   use_normalized_coordinates=True,
                   line_thickness=8)
               plt.figure(figsize=IMAGE_SIZE)
               plt.imshow(image_np)
```

While True:
    ret, image_np = cp.read()

cv2.imshow('object detection', cv2.resize(image_np, (800,600)))
if cv2.waitKey(25) & 0xFF == ord('q'):
    cv2.destroyAllWindows()
    break



Object detection

- Adapting to video

Q&A