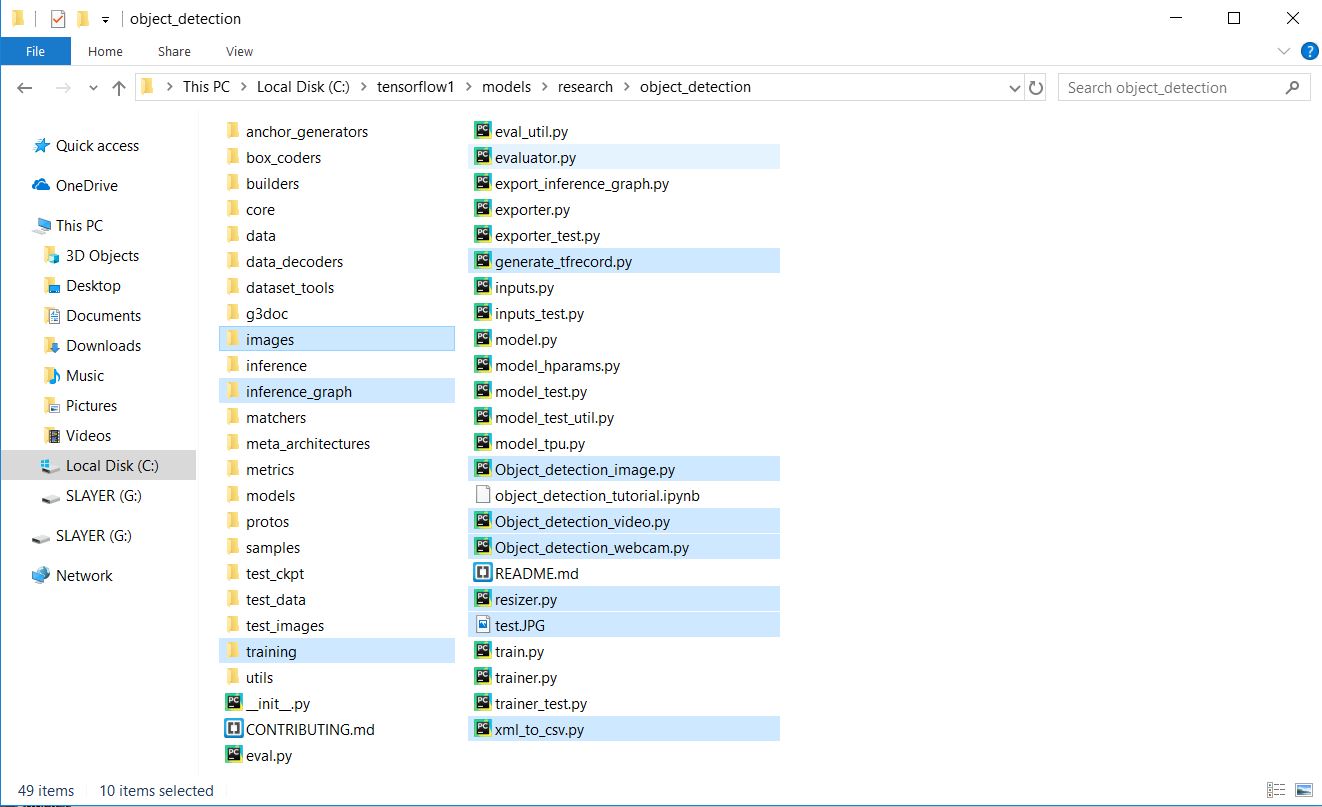
**1. Tensorflow Object Detection Kurulumu ve Anaconda Virtual Environment Oluşturma**

1. TensorFlow Object Detection API'ı indirin. <https://github.com/tensorflow/models>
2. C'de tensorflow1 isminde klasör oluşturup indirdiğiniz models klasörünü bu klasörün içine atın. models-master ismini basitleştirmek için models yapabilirsiniz. Object detection klasörünün yolu bu olmalı: C:\tensorflow1\models\research\object\_detection
3. Model zoo'dan Faster-RCNN-Inception-V2-COCO'yu indirin. tar.gz dosyasını object detection klasörüne çıkartın. <https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md>
4. Bu dersin kaynaklarında sizlerle paylaştığım .rar dosyasını indirin. .rar dosyasını object detection klasörüne çıkartın ve değiştirilsin mi diyen her dosyaya değiştirilsin deyin. Objects detection klasörünün içeriği aşağıdaki gibi olması lazım:



NOT: Eğer benden farklı bir dataseti üzerinde nesne tanıma yapmak istiyorsanız aşağıdaki klasörlerin içerisindeki dosyaları silin. (Klasörleri silmeyin)

* \object\_detection\images\train ve \object\_detection\images\test içerisindeki her şey.
* \object\_detection\images içerisindeki “test\_labels.csv” ve “train\_labels.csv”
* \object\_detection\training içerisindeki her şey.

**ANACONDA VIRTUAL ENVIRONMENT KURULUMU**

Komut penceresini yönetici olarak açın. Virtual environment oluşturmak için aşağıdaki komutu yazın:

C:\> conda create -n tensorflow1 pip

Virtual environment'ı aktifleştirmek için aşağıdaki komutu kullanın:

C:\> activate tensorflow1

Virtual environment aktifleştiyse parantez içerisinde (tensorflow1) göreceksiniz. Bu virtual environment üzerinde tek tek aşağıdaki tüm kütüphaneleri kurun:

(tensorflow1) C:\> pip install --ignore-installed --upgrade tensorflow-gpu

(tensorflow1) C:\> conda install -c anaconda protobuf

(tensorflow1) C:\> pip install pillow

(tensorflow1) C:\> pip install lxml

(tensorflow1) C:\> pip install jupyter

(tensorflow1) C:\> pip install matplotlib

(tensorflow1) C:\> pip install pandas

(tensorflow1) C:\> pip install opencv-python

NOT: Tensorflow'u CPU için kuracaksanız pip install'dan GPU'yu silin.

**PYTHONPATH AYARLAMA**

Aşağıdaki satırlarla PYTHONPATH oluşturun.

(tensorflow1) C:\> set PYTHONPATH=C:\tensorflow1\models;C:\tensorflow1\models\research;C:\tensorflow1\models\research\slim

NOT: tensorflow1'i her etkinleştirdiğinizde PYTHONPATH'i yeniden ayarlamanız lazım. O yüzden komut penceresini her kapatıp tekrar açışınızda bu kısmı tekrarlamanız gerekiyor.

**PROTOBUF'LARI DERLEME**

Protobuf'ları derlemek için research klasörü içinde olmanız gerekiyor. Aşağıdaki satırlar ile protobuf'ları derleyin:

(tensorflow1) C:\> cd C:\tensorflow1\models\research

protoc --python\_out=. .\object\_detection\protos\anchor\_generator.proto .\object\_detection\protos\argmax\_matcher.proto .\object\_detection\protos\bipartite\_matcher.proto .\object\_detection\protos\box\_coder.proto .\object\_detection\protos\box\_predictor.proto .\object\_detection\protos\eval.proto .\object\_detection\protos\faster\_rcnn.proto .\object\_detection\protos\faster\_rcnn\_box\_coder.proto .\object\_detection\protos\grid\_anchor\_generator.proto .\object\_detection\protos\hyperparams.proto .\object\_detection\protos\image\_resizer.proto .\object\_detection\protos\input\_reader.proto .\object\_detection\protos\losses.proto .\object\_detection\protos\matcher.proto .\object\_detection\protos\mean\_stddev\_box\_coder.proto .\object\_detection\protos\model.proto .\object\_detection\protos\optimizer.proto .\object\_detection\protos\pipeline.proto .\object\_detection\protos\post\_processing.proto .\object\_detection\protos\preprocessor.proto .\object\_detection\protos\region\_similarity\_calculator.proto .\object\_detection\protos\square\_box\_coder.proto .\object\_detection\protos\ssd.proto .\object\_detection\protos\ssd\_anchor\_generator.proto .\object\_detection\protos\string\_int\_label\_map.proto .\object\_detection\protos\train.proto .\object\_detection\protos\keypoint\_box\_coder.proto .\object\_detection\protos\multiscale\_anchor\_generator.proto

Bu satır ile protos klasöründeki her .proto dosyası derlenip .py dosyaları eklenecek. Gelecekte protos klasörüne daha fazla .proto dosyası eklenebilir. Eğer bu komutu çalıştırdığınızda /object\_detection/protos klasöründe derlenmemiş .proto dosyası görürseniz bu proto dosyasını ".\object\_detection\protos\DOSYAADI.proto" ile komuta ekleyebilirsiniz.

Aşağıdaki satırlar ile setup.py dosyasını çalıştırın. Önce build sonra install demeniz lazım.

(tensorflow1) C:\tensorflow1\models\research> python setup.py build

(tensorflow1) C:\tensorflow1\models\research> python setup.py install

Son olarak çalışıp çalışmadığını tutorial dosyasını açarak deneyebilirsiniz. Önce "cd object\_detection" diyerek object detection klasörüne gitmeniz lazım.

(tensorflow1) C:\tensorflow1\models\research\object\_detection> jupyter notebook object\_detection\_tutorial.ipynb

Jupyter notebook'ta deneme yaptıktan sonra Jupyter notebook'u komut penceresinde CTRL+C ile kapatabilirsiniz.

**2. Resim Toplayıp Etiketleme**

Kendi istediğiniz nesneleri eğitmek istiyorsanız öncelikle bir dataset oluşturmanız lazım. İnternetten veya kendi çektiğiniz foroğraflarla 200-300 kadar resim toplayın. Resimler farklı ortamlarda olursa daha güçlü bir model eğitebilirsiniz. Nesnenin sadece tek bir açıdan çekilmiş resimlerini kullanırsanız iyi sonuçlar almazsınız. Yani datasetinizde çeşitlilik bol olsun. Nesnenin bazı yerleri kapatılmış şekilde ve farklı ışıklandırmalarda resimleri toplayın.

Resimlerinizin boyutunu küçültün, çok büyük resimlerin eğitimi uzun sürecektir. 200kb'ı geçmemesine özen gösterin. Benim paylaştığım kodlarda resizer.py dosyası var bu dosyayı kullanarak resimlerinizin boyutunu ayarlayabilirsiniz.

Resimleri topladıktan sonra %20'sini \object\_detection\images\test klasörüne %80'ini \object\_detection\images\train klasörüne atın.

Resimleri uygun klasörlere attıktan sonra hepsini tek tek etiketlemeniz gerekiyor. Labellmg ile tüm resimlerdeki tanımak istediğiniz nesneleri tek tek seçerek etiketleme yapın.

Github: <https://github.com/tzutalin/labelImg>

Download: <https://www.dropbox.com/s/tq7zfrcwl44vxan/windows_v1.6.0.zip?dl=1>

Her etiketlediğiniz resim için .xml dosyası oluşturulacak. Bu .xml dosyaları tfrecords'a çevirilerek eğitim yapılacak. Test ve train klasörleri içerisindeki her resim için bir tane .xml dosyası olması lazım.

NOT: Etiketlerde Türkçe karakter olmasın.

**3. Eğitim Verisi Oluşturma**

Her resmi etiketledikten sonra TFRecord oluşturmak gerekiyor. Öncelikle etiket bilgisini barındıran .xml dosyalarını .csv dosyasına çevireceğiz. Bunu yapmak için komut penceresinde object\_detection klasöründe olmanız gerekiyor. Daha sonra aşağıdaki komutu çalıştırın. Images klasöründe 2 tane .csv dosyası oluşacak.

(tensorflow1) C:\tensorflow1\models\research\object\_detection> python xml\_to\_csv.py

.csv dosyaları oluşturulduktan sonra generate\_tfrecord.py dosyasını herhangi bir text editor ile açın. 31. satırda sınıf bilgilerinizi girin. Benim datasetimde 6 tane sınıf olduğu için 6 tane farklı ID atadım. Sınıf sayınıza göre bunları değiştirerek 1'den başlayarak sınıf sayınız kadar ID atayın.

# TO-DO replace this with label map

def class\_text\_to\_int(row\_label):

if row\_label == 'dokuz':

return 1

elif row\_label == 'on':

return 2

elif row\_label == 'vale':

return 3

elif row\_label == 'kiz':

return 4

elif row\_label == 'papaz':

return 5

elif row\_label == 'as':

return 6

else:

None

Örnek olarak eğer sınıflarınız kedi, köpek, at ise yukarıdaki kodu aşağıdaki şekilde düzenlemeniz gerekir.

# TO-DO replace this with label map

def class\_text\_to\_int(row\_label):

if row\_label == 'kedi':

return 1

elif row\_label == 'kopek':

return 2

elif row\_label == 'at':

return 3

else:

None

Sınıflarınıza ID verdikten sonra aşağıdaki komutlar hem test hem train için TFRecord dosyası oluşturun. Komutları çalıştırdıktan sonra object\_detection klasöründe test.record ve train.record dosyaları yaratılacak.

python generate\_tfrecord.py --csv\_input=images\train\_labels.csv --image\_dir=images\train --output\_path=train.record

python generate\_tfrecord.py --csv\_input=images\test\_labels.csv --image\_dir=images\test --output\_path=test.record

**4. Label Map Oluşturma ve Eğitim Ayarları**

**LABEL MAP**

Label map modele hangi ID'de hangi sınıf var onu söyler. object\_detection/training klasörüne gidin ve yeni bir dosya oluşturun. Oluşturduğunuz dosyanın ismine "labelmap.pbtxt" verin (uzantısı .pbtxt olması lazım .txt vermeyin). Dosyayı text editor ile açıp aşağıdaki şekilde sınıf ve ID bilgilerini girin. Bu sınıf ID'leri kendi datasetinize göre düzenlemeniz gerekiyor.

item {

id: 1

name: 'dokuz'

}

item {

id: 2

name: 'on'

}

item {

id: 3

name: 'vale'

}

item {

id: 4

name: 'kiz'

}

item {

id: 5

name: 'papaz'

}

item {

id: 6

name: 'as'

}

Eğer sınıflarınız yukarıda belirttiğimiz şekilde kedi, köpek, at olsaydı bu dosyanın içeriği şu şekilde olmalıydı:

item {

id: 1

name: 'kedi'

}

item {

id: 2

name: 'kopek'

}

item {

id: 3

name: 'at'

}

**EĞİTİM AYARLARI**

Bu aşamada hangi model ve hangi parametreler kullanılacak belirliyoruz.

C:\tensorflow1\models\research\object\_detection\samples\configs klasörüne gidin

faster\_rcnn\_inception\_v2\_pets.config dosyasını kopyalayıp object\_detection\training klasörüne yapıştırın. (Hangi modeli kullanacaksanız o modelin config dosyasını yapıştırın)

Text editor ile bu dosyayı açın ve aşağıdaki değişiklikleri yapın.

NOT: Dosyaların yolu normal slash(/) ile olmalı ters slash(\) koyarsanız hata alırsınız. Windows'ta adres satırından kopyalama yaparken ters slash olarak kopyalar. Bu durumda tüm bu ters slashları düz slasha çevirmeniz lazım. Ayrıca çift tırnak(") işareti kullanın, tek(') kullanmayın.

* 9. satır: Buraya sınıf sayınızı yazın. Benim için bu 6. Kedi, köpek, at üzerinde tanıma yapıyorsanız 3 olmalı.
* 110. satır: fine\_tune\_checkpoint'e aşağıdaki satırı verin. (Hangi modeli kullanmak istiyorsanız onu verin)

fine\_tune\_checkpoint : "C:/tensorflow1/models/research/object\_detection/faster\_rcnn\_inception\_v2\_coco\_2018\_01\_28/model.ckpt"

* 126 ve 128. satır: train\_input\_reader içerisinde input\_path ve label\_map\_path belirleyin. Bunları aşağıdaki satırlardaki gibi değiştirin.

input\_path : "C:/tensorflow1/models/research/object\_detection/train.record"

label\_map\_path: "C:/tensorflow1/models/research/object\_detection/training/labelmap.pbtxt"

* 132. satır: Bu satırda num\_examples'a kaç tane test resminiz varsa onu vereceksiniz. images\test klasöründe kaç tane test resminiz olduğunu öğrenebilirsiniz. Ben de 67 tane var. O yüzden num\_examples: 67 yazmam gerekiyor.
* 140 ve 142. satır: eval\_input\_reader içerisinde input\_path ve label\_map\_path belirleyin. Bunları aşağıdaki satırlardaki gibi değiştirin.

input\_path : "C:/tensorflow1/models/research/object\_detection/test.record"

label\_map\_path: "C:/tensorflow1/models/research/object\_detection/training/labelmap.pbtxt"

Yaptığınız değişiklikleri kaydedin.

**5. Eğitimi Gerçekleştirme**

Eğitimi çalıştırmak için aşağıdaki komutu /object\_detection klasörü içerisinde çalıştırın.

python train.py --logtostderr --train\_dir=training/ --pipeline\_config\_path=training/faster\_rcnn\_inception\_v2\_pets.config

Eğitimin başlaması 30 saniye, 1 dakika kadar sürecektir. Adım adım eğitildiğini görüyorsanız eğitim yapılıyor demektir. Eğer bu noktada hata aldıysanız en altta sıkça karşılaşılan hatalar kısmını inceleyin.

Her adımda ekrana loss yazdırılacak. Loss'ı takip ederek dilediğiniz kadar eğitin. Benim tavsiyem en az 50 bin adım eğitmeniz yönünde. Datasetinize ve modelinize bağlı olarak daha az veya daha fazla eğitim yapmanız da gerekebilir. Loss'ı takip ederek bir noktadan sonra grafiğin düzleştiğini gözlemlerseniz eğitimi durdurabilirsiniz. Eğitimi durdurmak için Ctrl+C tuşlarına basın.Belli aralıklarla kaydediliyor kapattıktan sonra kaldığınız yerden eğitime devam edebilirsiniz.

Eğitimi Tensorboard'da takip etmek için yeni bir komut penceresi açıp aşağıdaki komutu çalıştırın:

(tensorflow1) C:\tensorflow1\models\research\object\_detection>tensorboard --logdir=training

Tensorboard'u açmak için verilen adrese gidebilirsiniz. Veya tarayıcınızda bu adrese gidebilirsiniz:

localhost:6006

Tensorboard'u kapatmak için Tensorboard'un açılı olduğu komut penceresinde Ctrl+C'ye basın.

**6. Inference Graph**

Yeterince eğittikten sonra modeli kullanabilmek için frozen inference graph oluşturacağız. Bunun için aşağıdaki komutu çalıştırabilirsiniz. Komutta yazan XXXX yerine training klasöründe .ckpt dosyalarından sayısı en büyük olanı yazın. Mesela training klasöründen en büyük sayı olarak " model.ckpt-55841" görüyorsanız XXXX yerine 55841 yazacaksınız.

python export\_inference\_graph.py --input\_type image\_tensor --pipeline\_config\_path training/faster\_rcnn\_inception\_v2\_pets.config --trained\_checkpoint\_prefix training/model.ckpt-XXXX --output\_directory inference\_graph

Bu çalıştıktan sonra \object\_detection\inference\_graph klasöründe nesne tanıma sınıflandırıcısı .pb dosyası oluşturacak.

**7. Programı Kullanma**

Nesne tanıma programınız artık kullanıma hazır durumda. Resim, video ve webcam için python scriptleri mevcuttur. Bu scriptleri çalıştırmadan önce açın ve NUM\_CLASSES değişkenine sınıf sayınızı atayın.

Object\_detection\_image.py'ı çalıştırdığınız zaman object\_detection dosyasında test1.jpg isimli resim üzerinde tanıma yapacak. Bu kodlarla istediğiniz gibi oynayarak nesne tanıma yapabilirsiniz.