Steps to train YOLO V2 on custom dataset

Step- 1: Data collection

Get your data into your machine:

First step in training yolo for your custom dataset is to collect the data.

* There are many yearly computer vision challenge websites where you can find relevant annotated data you are looking for.

The datasets are usually available in form of .json, text files which has information about images and annotations and then you need to download the data by writing your script or sometimes you can find the images itself with annotation file in xml, txt etc. files.

* Data scraping can also help you to get data from google, Instagram etc.

If you did not find any dataset using the above two methods:

Create your own dataset:

I believe you will have to create your own dataset and annotate it:

* Take a camera – you can take still images or videos.

Still images can be annotated one by one on the other hand you can convert the videos into frame using below openCV functions:

capture = cv2.VideoCapture(0)

ret, frame = capture.read()

image\_name = '/complete\_path\_to\_save\_images/'+current\_date\_time+'.jpg'

cv2.imwrite(image\_name, frame)

Annotation/Open source Labelling tool for yolo:

Use this tool BBox-Label-Tool - <https://github.com/puzzledqs/BBox-Label-Tool>

And label your data manually.

After annotating the images, you will have values like coordinate values like:

(xmin, ymin) = Leftmost x and y coordinate values of bounding box.

(xmax, ymax) = Rightmost x and y coordinate values of bounding box.

Increase your dataset:

* This repo <https://github.com/jrosebr1/imutils> ,provides functions like rotation, resize, scale, translation. Use these function on the available data/images you have created or downloaded and increase your dataset.
* You may also try to change the background of the images you have and insert a new background.
* You can insert noises and distortions like barrel, pincushion, mustache distortion using interpolation and increase your dataset.

Step – 2: Data cleaning and Balancing

There is no doubt that yolo is the state of art but the data that you choose to give for training determines the precision, recall, map etc. that you expect to be good after so much of time and expense spent on training.

Choose your data properly to save yourself from issues like overfitting, under fitting and alien detection about which you have no idea.

To understand about overfitting, under fitting, confusion matrix and other error matrices , I recommend this wonderful video by Luis serrano -<https://www.youtube.com/watch?v=e2vurxnd124>

6 very important rules to clean and balance your data before using it to train your model:

1. Do not use duplicate images it leads to overfitting.
2. Try to keep same number of annotation/labels per class.
3. Use variety of images not just very clear and clean images, chose images which has variations of your data like a mixture of clean, bit noisy, up scaled, down scaled, occluded images to train.
4. Practice a habit of noting down concise each class data statistics used for training by making a table, which can contain the number of images and annotation per class before each training.
5. Increase your dataset because deep learning is a giant the more you feed the stronger and better it performs.
6. Always check your annotation/label file before using it for training.

Write your cross check code, make a light weight gui by using function of openCV imshow() and draw shapes like rectangles using cv2.rectangle() to cross check whatever annotation/label data you wrote for each image in the text file was correctly written on not.

Step 3- Structuring the data to train yolo

This article is dedicated to train yolo (you only look once) so let us understand how the images and its labels must be structured so that we can successfully start the training.

Data collection, cleaning and balancing are the most important steps before you even take efforts to structure your data to train a model.

Let’s suppose you have only one image “Image.jpg” and only one class “humanface”.

Let us see the steps involved to train yolo for our first class humanface.

So let us start structuring data.

Yolo expects two files in two different folders, make two folders one to keep all your images and another to keep the annotation files which are supposed to be text files for that image.

For an image “image.jpg” you need an annotation file “image.txt”

Now the question is what will you have in your image.txt files?

Answer – <label> <x> <y> <width> <height>

Label – since humanface is our first class so the label must be “0”

X – x coordinate for center of the bounding box with respect to image dimension.

Y – y coordinate for center of your bounding box with respect to image dimension.

Width – width of the bounding box with respect to image dimension.

Height – height of the bounding box with respect to image dimension.

To get this kind of data use the (xmin, ymin) = Leftmost x and y coordinate values of bounding box, (xmax, ymax) = Rightmost x and y coordinate values of bounding box and feed it to this script <https://github.com/Guanghan/darknet/blob/master/scripts/convert.py>

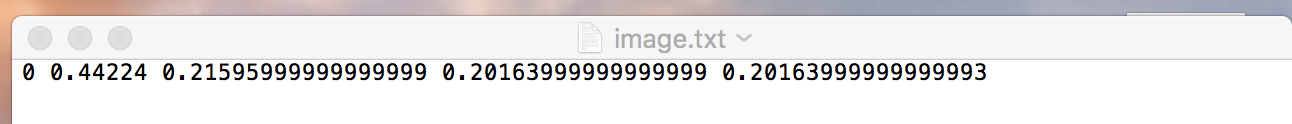
I request you to read and understand the code convert.py and take the function convert() from it write your own and generate text files corresponding to each image. I did the same.

NOTE:

All your integer values will be converted into float and NONE OF THE VALUES IN YOUR TEXT FILE SHOULD BE LESS THAN 0 AND GREATER THAN 1.

If the above conditions are not true, check your math’s to convert, spend some time.

So you “image.jpg” will have the rgb image data and “image.txt” will have data like

where the label = 0, x =0.442, y= 0.215959999, width = 0.201639, height = 0.2016

In same way for every image we need to make a corresponding text file with the same name as the name of image.

The above topics you just read is important to get introduced to how data should be structured for training yolo.

Step -3 Understand files needed to train yolo

You would need to understand these six files to run yolo training.

1. variable.names
2. variable.data
3. variable.cfg
4. variable.weights
5. train.txt
6. test.txt

Here variable can be any name, could be yolo, voc, coco, x, y, z.

Now, is the time to get yolo running on your computer.

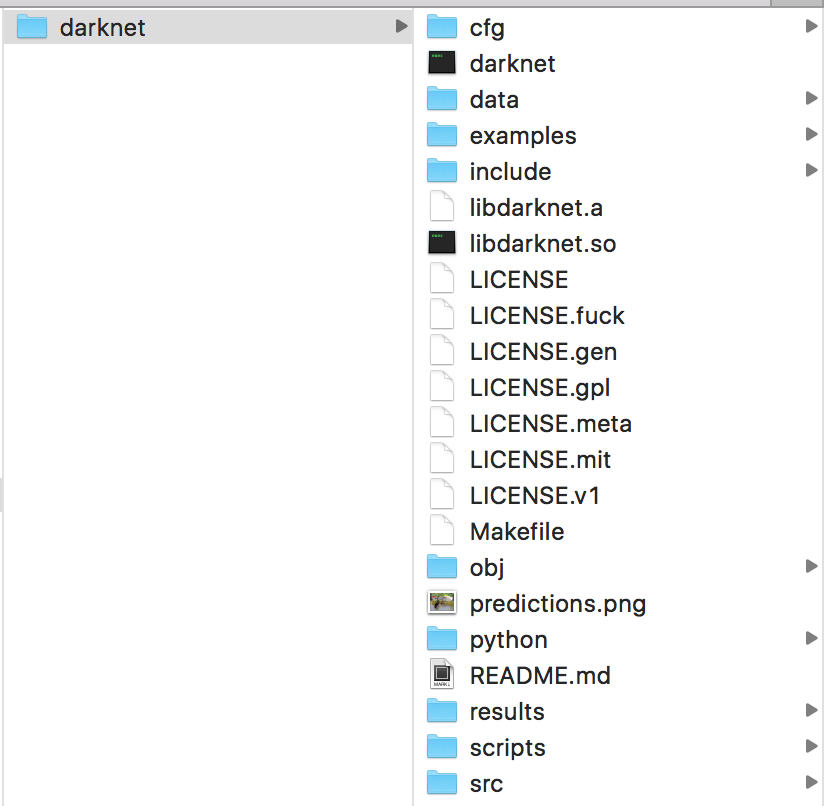
<https://pjreddie.com/darknet/yolo/> is the official website of yolo.

Yolo is made on the framework called as darknet.

Step 3(a): variable.cfg and variable.weights

Clone darknet repo from official website, use the below command on your command window.

Click on the link and git clone https://github.com/pjreddie/darknet



This is how your file look like.

Now go to the darknet directory,

cd darknet

Then make the Makefile by using the command make.

make

In darknet folder you will find a cfg folder which has all the configuration files trained on different dataset.

We will be using “yolo.cfg”. For now, do not bother about what is in that.

Now we will look into the next file that is weights file, weight files are usually not found in repo because it is huge in size so you can download “yolo.weights” from

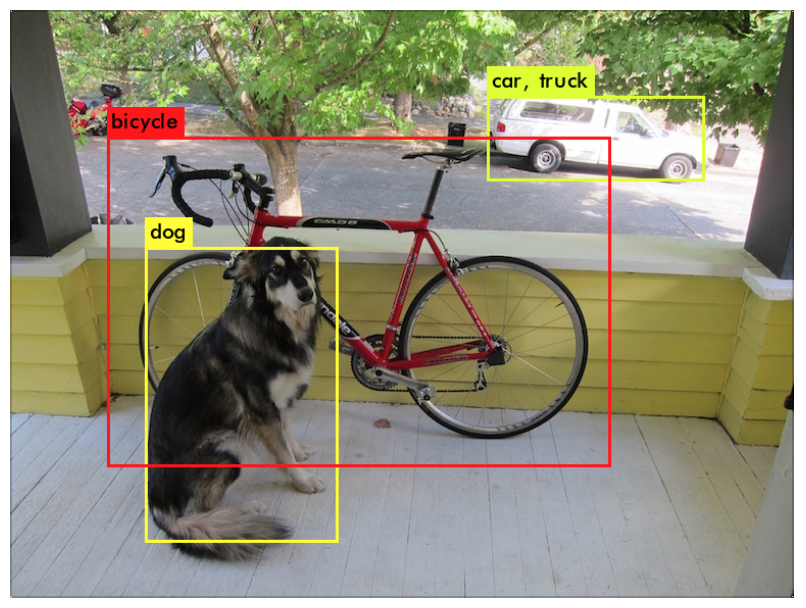
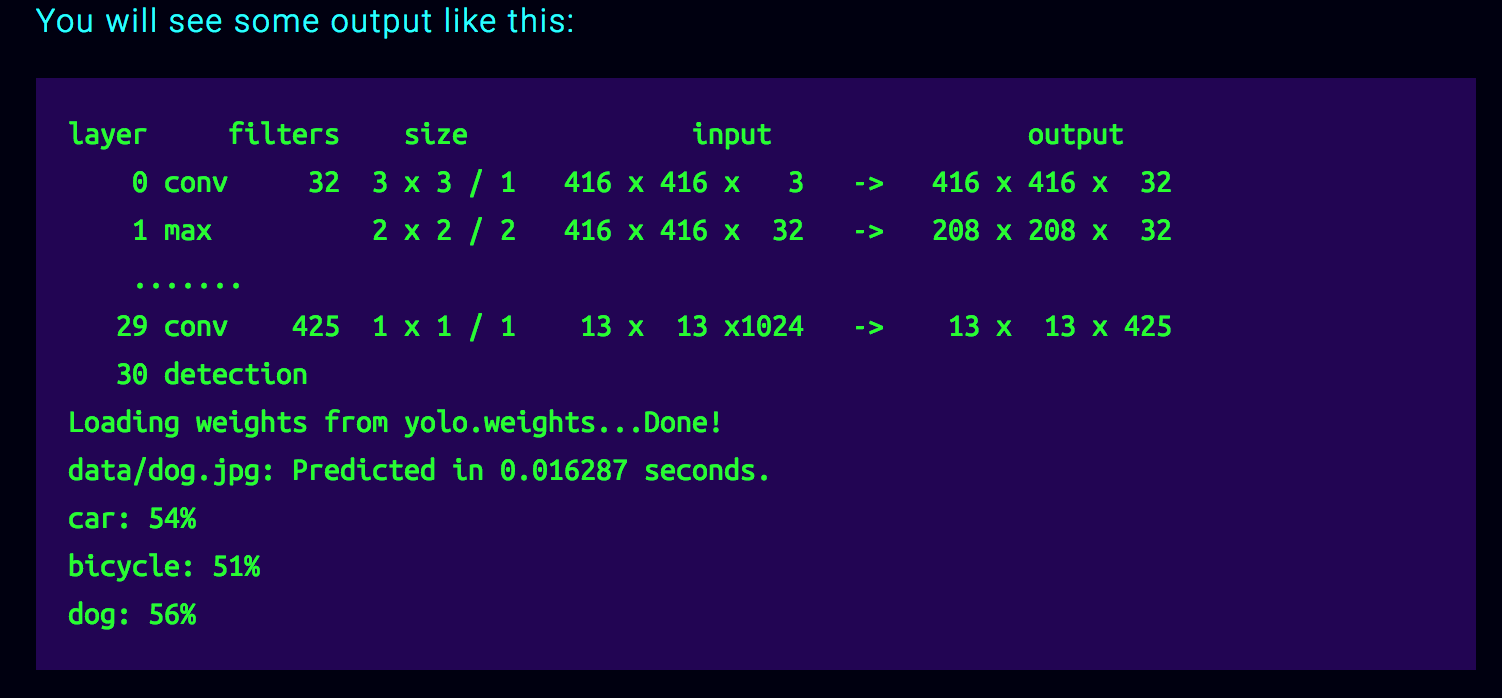
wget https://pjreddie.com/media/files/yolo.weights

once the weights and cfg file are in place use the below command on your cmd window.

Give absolute path to access the “yolo.cfg” and “yolo.weights” file.

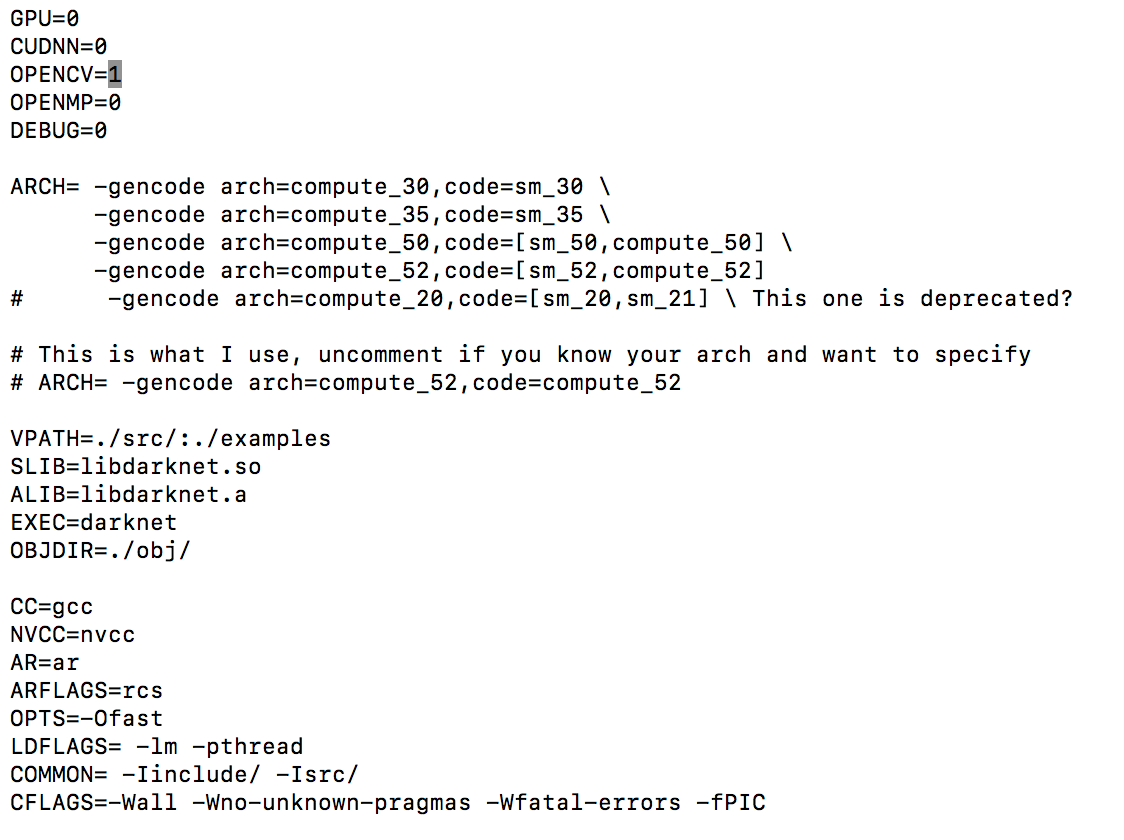
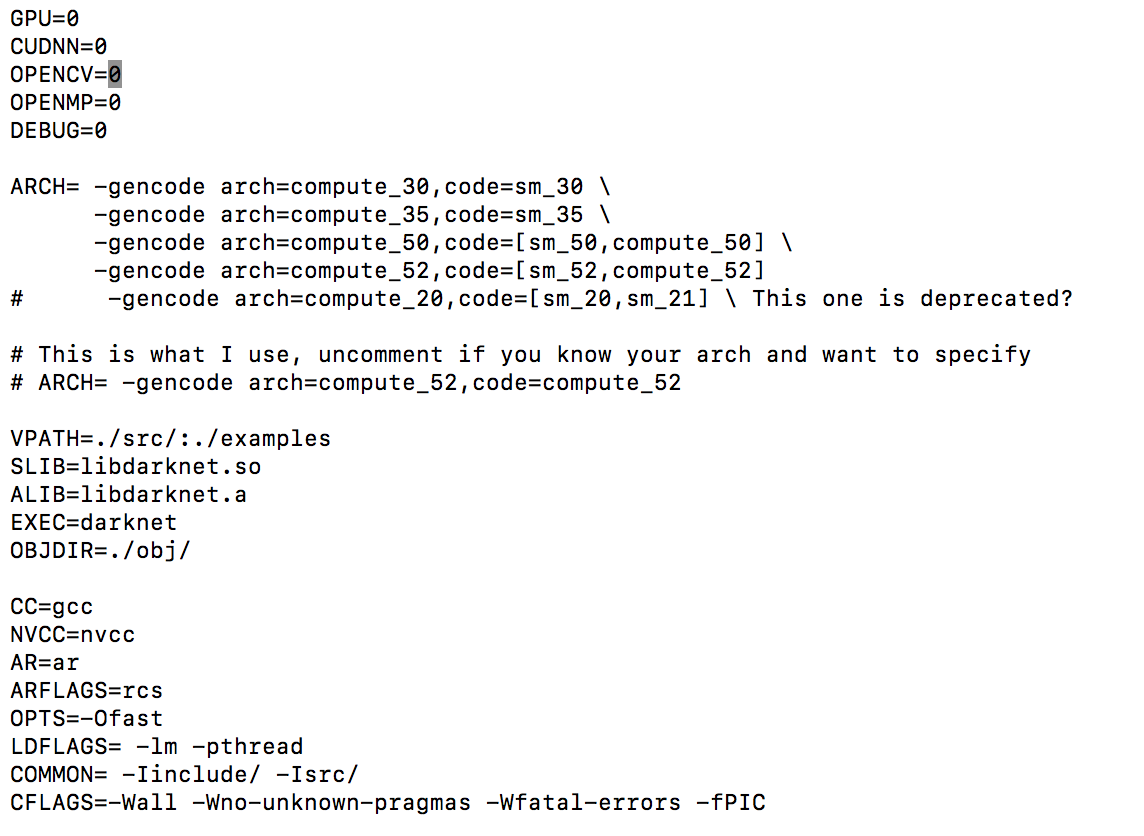
./darknet detect cfg/yolo.cfg cfg/yolo.weights data/dog.jpg

once you run this, you would find an image predictions.png in your darknet folder.



If you want to view the results on screen instead of it getting saved in the folder.

Seeing output on screen requires openCV <https://opencv.org/> if you do not have openCV installed then download it and install it and open Makefile and change the OPENCV micro from 0 to 1.



Then run the command again. If your openCV is installed properly you will now see the results on screen.

./darknet detect cfg/yolo.cfg cfg/yolo.weights data/dog.jpg

Good to go.

So you saw two files yolo.cfg and yolo.weights now let us look into another files.

Now download PASCAL data.

wget https://pjreddie.com/media/files/VOCtrainval\_11-May-2012.tar

wget https://pjreddie.com/media/files/VOCtrainval\_06-Nov-2007.tar

wget https://pjreddie.com/media/files/VOCtest\_06-Nov-2007.tar

tar xf VOCtrainval\_11-May-2012.tar

tar xf VOCtrainval\_06-Nov-2007.tar

tar xf VOCtest\_06-Nov-2007.tar

There will now be a VOCdevkit/ subdirectory with all the VOC training data in it.

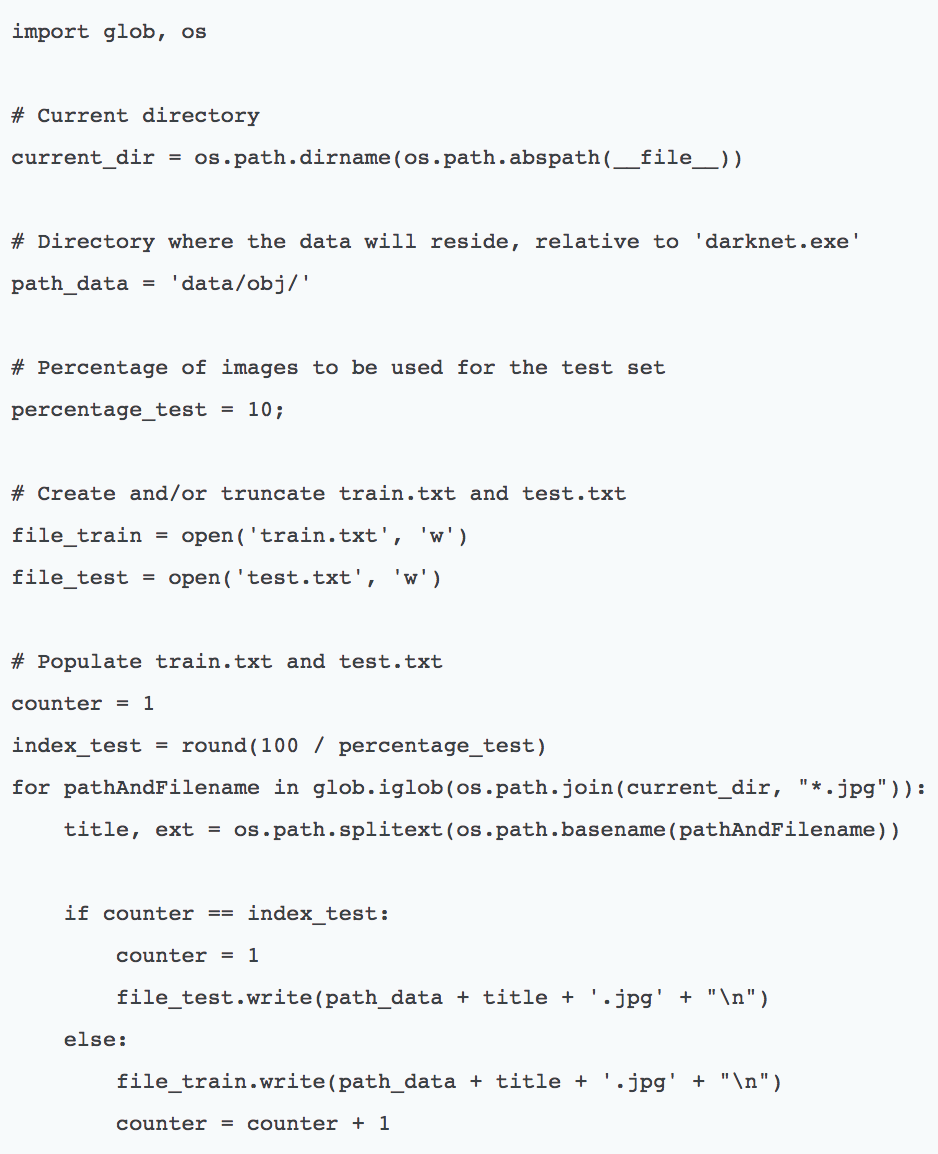
Take a backup of images and text files of JPEGImages and labels from darknet/VOCdevkit/VOC2007/JPEGImages and darknet/VOCdevkit/VOC2007/labels.

Go in darknet/VOCdevkit/VOC2007/JPEGImages🡪”image.jpg” and paste all your images that will be used for training.

Go to darknet/VOCdevkit/VOC2007/labels🡪”image.txt” and paste all the annotation text files corresponding to images you are pasting in JPEGImages folder for training.

Step -3(b): train.txt and test.txt

The two files train.txt and test.txt are will contain the path of your training data so let us use this code to generate our train and test text files.



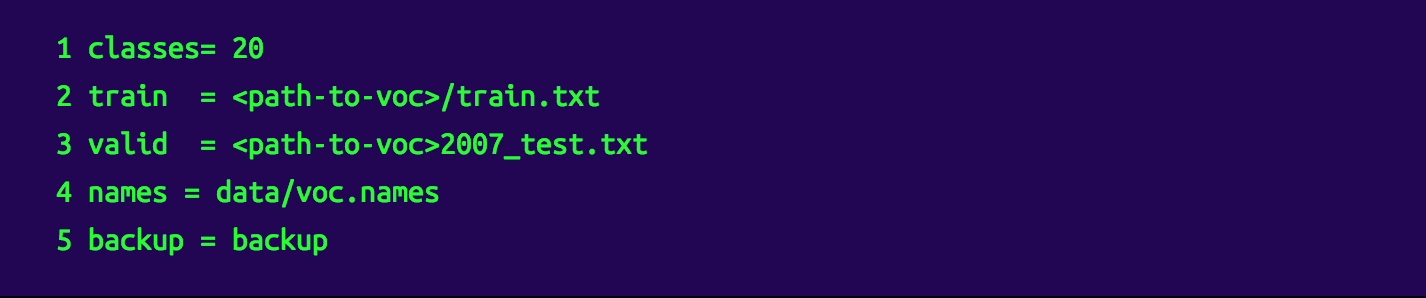
Run the above code and segregate your data into train and test. When you open your train.txt you will find path of images which will be used to train the model and test.txt you will find the path of your test data.

Step 3(c): voc.data and voc.names

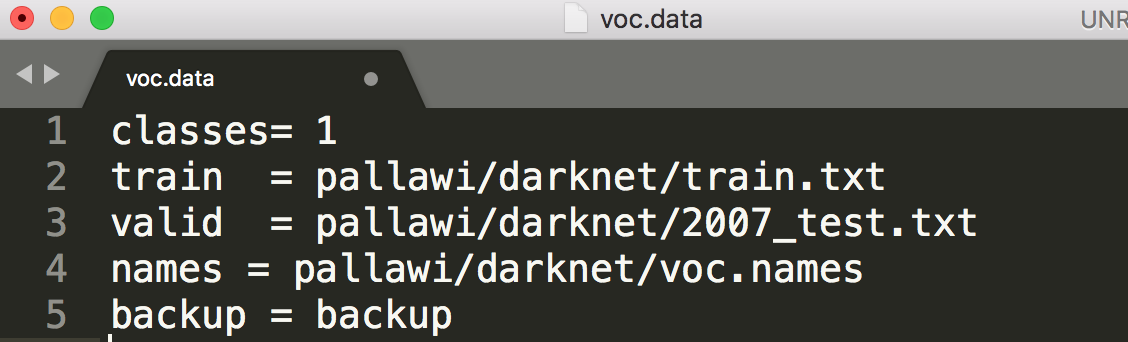
Go into darknet/cfg you will find voc.data. The below image shows the default structure of the voc.names file but now let us customize it according to our requirements. Since we are training for only one class so for us the classes will be classes = 1 and we will always practice giving absolute path of train and text file in my case I prefer keeping my train and text file in darknet/ folder. train = myname/darknet/train.txt and test =myname/darknet/2007\_test.txt

And names would be same names = data/voc.names and backup = backup,

Backup is the folder where your weight will be generated after every 100 iterations.



Training for only one class “humanface” your voc.data will look like.



voc.names

NOTE: In the annotation file “image.txt” we mentioned label as “0” for “humanface”.

While making voc.names file we will make sure, open your eyes wide open and write the class name “humanface” on zeroth line (line number 0).

RULE SAYS LABEL AND CLASS NAME LINE NUMBER SHOULD BE SAME.

If you are training for two classes in the same image “image.jpg”, “humanface” and “dog”, your annotation text file will have data like this:

image.txt

0 0.7876 0.878600 0.3422 0.56534

1 0.765342 0.52334 0.52314 0.89743526

And your “voc.names” file will be

voc.names

humanface

dog

In images.txt the first value corresponds to the line number in voc.names.

So you know now all the files and you have put them correctly in their proper places.

Step 3(d):

Download the pre trained weights trained on Imagenet data - weight file from:

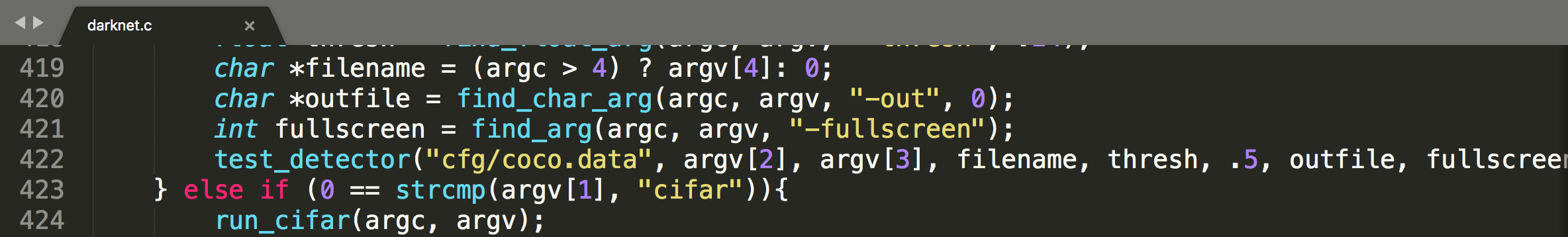
wget https://pjreddie.com/media/files/darknet19\_448.conv.23

#######MOST IMPORTANT FIX#######

Go to: darknet/examples/darknet.c

Line number 422 where test\_detector() is called.

Make changes in line 422 from coco.data to voc.data.





Put the weight file in your darknet/cfg/darknet19\_448.conv.23

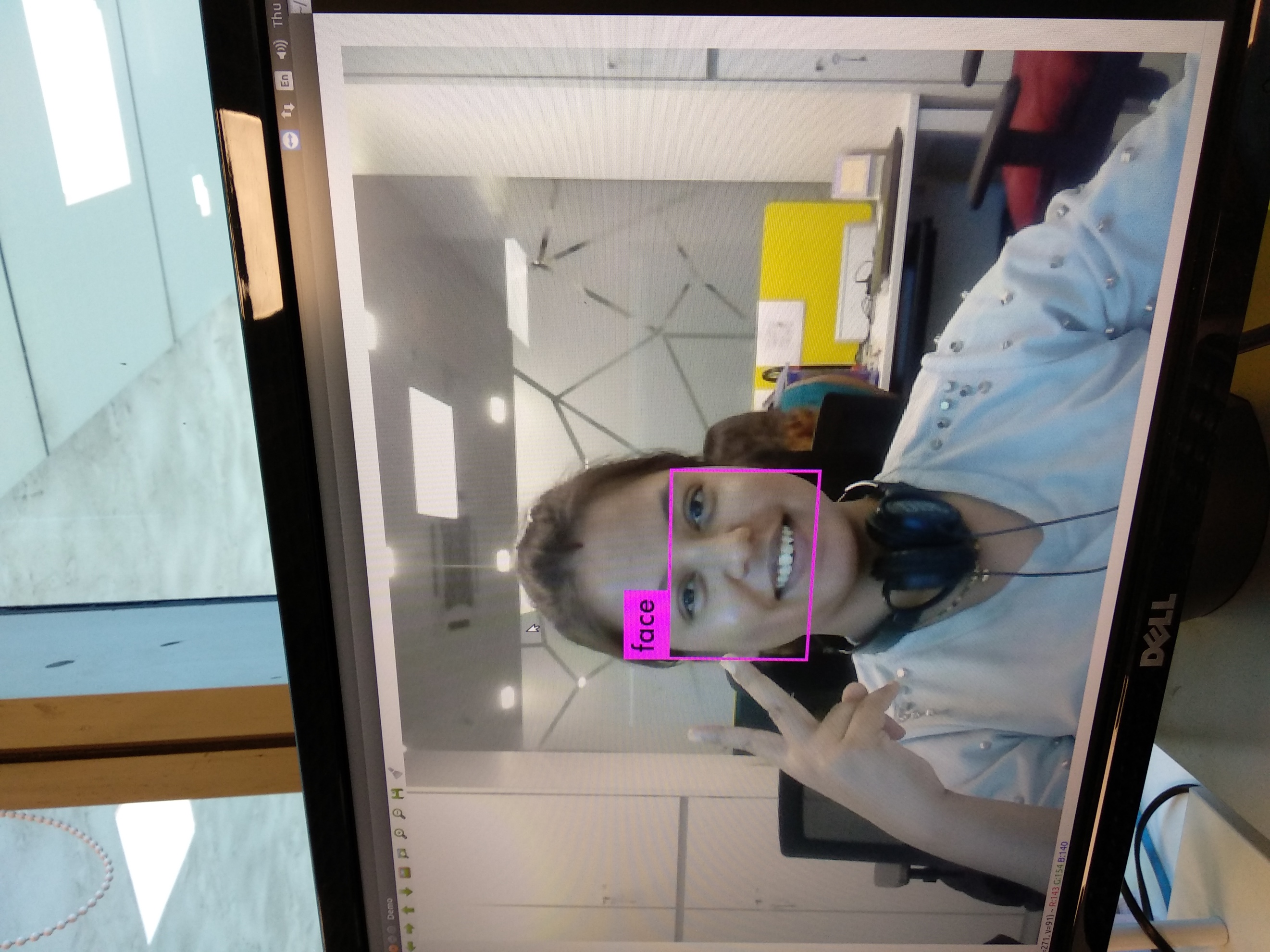
Give absolute path of “voc.data”, “yolo-voc.cfg”, “darknet19\_448.conv.23”

Now you can train! Run the command!

./darknet detector train cfg/voc.data cfg/yolo-voc.cfg darknet19\_448.conv.23

When I trained my first class it was for humanface, I referred this well explained blog , but before clicking onto this link I advise you to continue reading because there are few things we need to know before reading and then I suggest you read this blog, <https://timebutt.github.io/static/how-to-train-yolov2-to-detect-custom-objects/> to understand how to train yolo. Thanks to the author.

You got to pay attention on minute details and work smart to train and test yolo and the only way is to read, reflect and RUN!



Trained yolo successfully for single class.

Class -face