**YOLOV3-CUSTOM-OBJECT-DETECTION**

**STEPS:**

1. Install anaconda or pip with python3.5

ANACONDA: <https://www.anaconda.com/distribution/>

Pip installation: <https://www.makeuseof.com/tag/install-pip-for-python/>

1. Install darknet framework by cloning in from github through commands in terminal and run the makefile. DARKNET:<https://pjreddie.com/darknet/install/>

**Commands are:**

git clone https://github.com/pjreddie/darknet.git

cd darknet

make

**after running the make command the output should be like this and after that run the darknet framework**

mkdir -p obj

gcc -I/usr/local/cuda/include/ -Wall -Wfatal-errors -Ofast....

gcc -I/usr/local/cuda/include/ -Wall -Wfatal-errors -Ofast....

gcc -I/usr/local/cuda/include/ -Wall -Wfatal-errors -Ofast....

.....

gcc -I/usr/local/cuda/include/ -Wall -Wfatal-errors -Ofast -lm....

**Command for running darknet framework**

./darknet

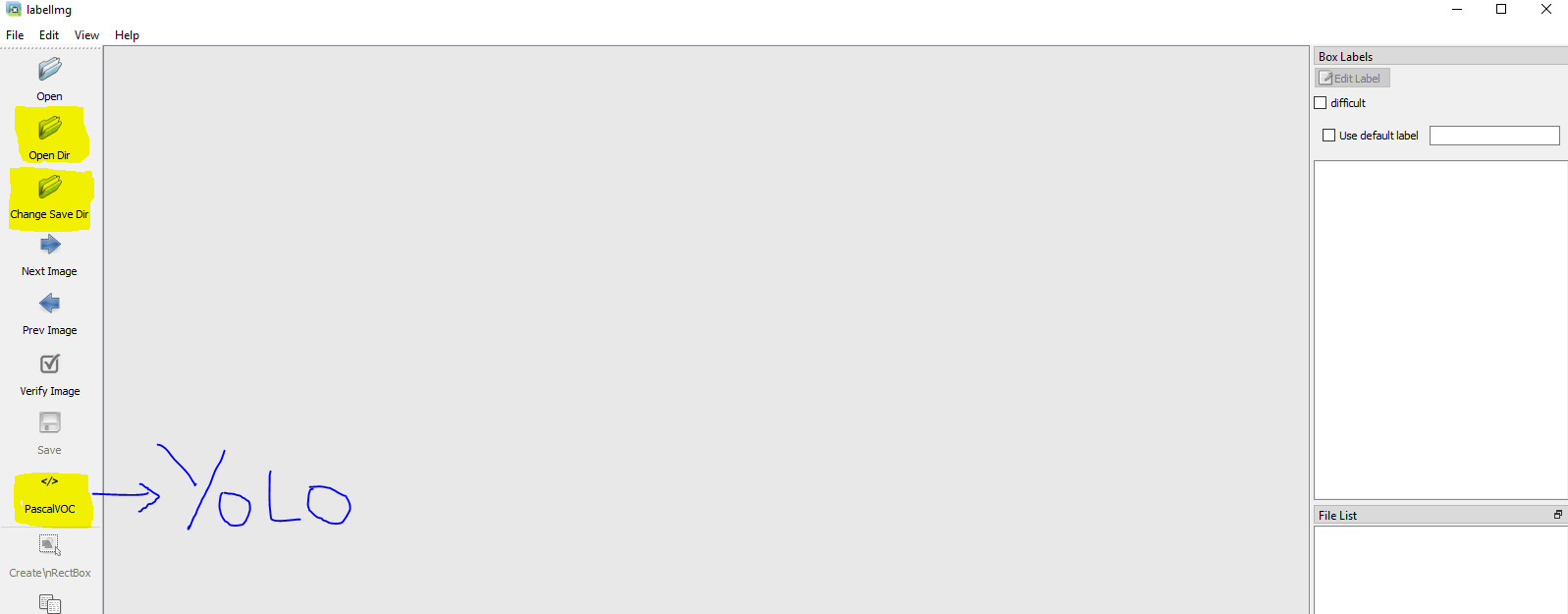
**3)**To run the object detection algorithm we need data and labels. Data can be collected from google by using **BULK IMAGE DOWNLOADER** that is available as an extension in google chrome. Once the data is collected we need to label it. This can be done by using **labelImg** software

Follow the steps in github to download it for the respective OS:

<https://github.com/tzutalin/labelImg> (or)

Downloads list for windows: <https://tzutalin.github.io/labelImg/>

**4)** Once the labelImg is downloaded and installed start it either from terminal or by manually. To start through terminal type the command **labelImg** in terminal



Click on the **open dir** to load the **images folder**. Now click on the **change** **save dir** to save the annotations in the **labels folder.** Both the folders should be available in the same directory. Now change the **pascalVOC**  format to **yolo** format. And then press **W** to start the bounding boxes. After each image is done with the bounding boxes save the image with **ctrl+s** to save the annotations in the **labels** folder. Repeat this until every image is labelled. Every image is found in the **File list** in **right-bottom.**

**5)**Once the images are labeled move the images and the labels to the **data** folder in **darknet** directory.

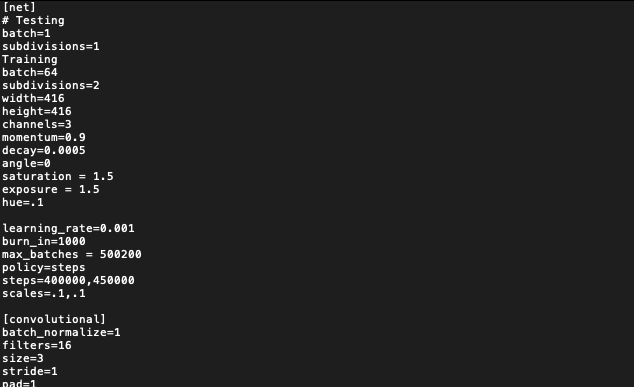
**6)**These images and labels need to be split into train and test files. For that run the **generate.py** by placing it in the darknet directory. The python file is available in my github. Command for running: **python generate.py** once we have moved to the darknet directory in cmd/bash. **(NOTE: In generate.py folder, the path of the images must be specified properly. If images and labels are properly placed in data folder of darknet directory and python file is placed In darknet directory no need to change the path)**

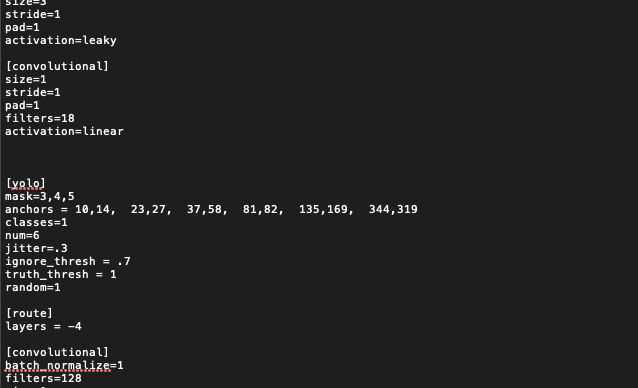
**7)** Train and test files will be generated automatically in the darknet directory. Create a new folder named **custom** in darknet directory and move the train test files to the **custom** folder.

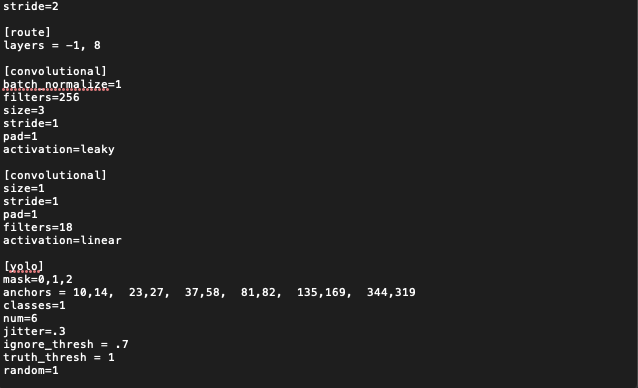
**8)** now we need to create **cfg** file of **yolo-tiny v3.** For simplicity I have placed the **cfg** file in the github repository. Orelse if need to be done manually go the **cfg** folder in darknet directory click on **yolov3-tiny.cfg** and do the specified modifications.

**MODIFICATIONS:**

* Uncomment the lines 5,6, and 7 and change the training batch to 64 and subdivisions to 2.
* Change the number of filters for convolutional layer "[convolution]" just before every yolo output "[yolo]" such that the number of filters= #anchors x (5 + #ofclasses)= 3x(5+1)= 18. The number 5 is the count of parameters center\_x, center\_y, width, height, and objectness Score. So, change the lines 127 and 171 to "filters=18"
* For every yolo layer [yolo] change the number of classes to 1 as in lines 135 and 177.







**9)** once the cfg file is created move it to the custom folder of darknet directory. Now we need to create objects.names and trainer.data files in the custom folder.In objects.names file type the name of the class that is to be detected. For eg: “softdrinks”. In trainer.data place the following lines.

classes= 1

train = custom/train.txt

valid = custom/test.txt

names = custom/objects.names

backup = backup/

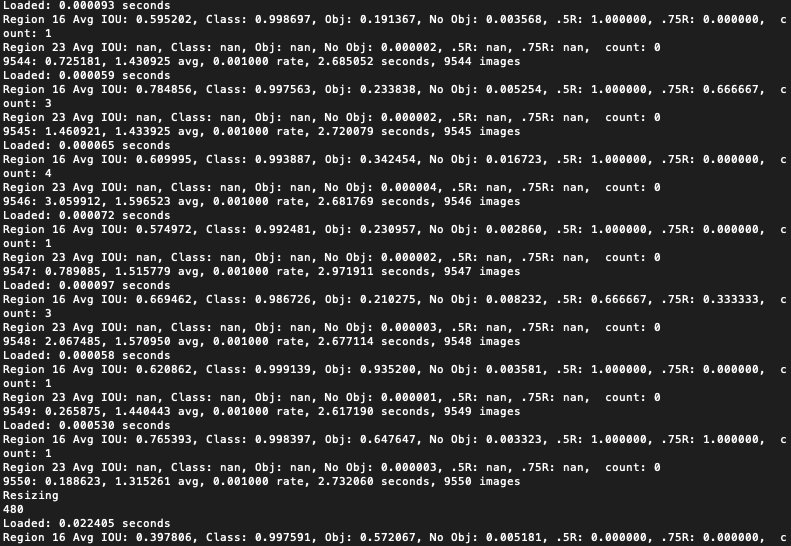
save them in the custom folder.

**10)** now weights for training the data needs to be downloaded and placed in the darknet directory … weights: <https://pjreddie.com/media/files/darknet53.conv.74>

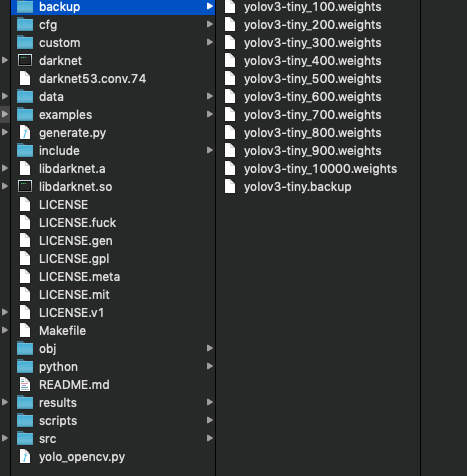
**11)**  once the weights are downloaded and placed in the main directory of darknet. Run the following command through the terminal in darknet directory:

./darknet detector train custom/trainer.data custom/yolov3-tiny.cfg darknet53.conv.74

This command will start the training process.



**12)**The weights will be stored in the backup folder every 100 iterations till 900 and after only at 10000 iteration weights will be stored.

****

**13)** The avg loss is best when it is at 0.06 app. It is good when it is less than 2.00 app.

**14)** Install opencv package in the System along with their dependencies in a new conda environment by executing the following commands. (Note: Dependency resolving is not mentioned here, it varies based on the OS. Here only conda env creation and opencv installation is mentioned.)

1. Create conda environment with this command

conda create --name opencv python=3.5

1. Activate opencv environment with this command

source activate opencv

1. Install OpenCV with this command

conda install -c conda-forge opencv

or this one

conda install -c conda-forge/label/broken opencv

**15)** Run the following command to run the final step of the yolov3 algorithm. Command: **“python yolo\_opencv.py -c /path/to/yolov3-tiny.cfg -w /path/to/yolov3-tiny\_finally.weights -cl /path/to/objects.names”**

Specify the paths for the above commands and run the command in darknet framework…For eg**: “python yolo\_opencv.py -c custom/yolov3-tiny.cfg -w backup/yolov3-tiny\_10000.backup -cl custom/objects.names”**