**Project Report**

Submitted to Dhiraj Sangwan

Submitted by:

Jyoti Vishnoi- 2015A7PS0013P

Maitri Shastri-2014B2A70220P

Date – 29.4.2018.

**STEPS TO BE FOLLOWED WHEN WORKING WITH YOUR OWN DATASET YOLOv2 :-**

1. Clone Darknet github repository.

git clone [**https://github.com/AlexeyAB/darknet.git**](https://github.com/AlexeyAB/darknet.git)

1. Yolo requires labels.txt for each training image in the following format.

*[category number] [object center in X] [object center in Y] [object width in X] [object width in Y]*

1. There are several conversion scripts on github to change annotation format to that required by Darknet.
2. Create two txt files train.txt and test.txt having the names of the images used for training and testing.

Sample-train.txt file



1. Create obj.data file having the following content and save it in /cfg directory.

classes= 1

train = train.txt

valid = test.txt

names = obj.names

backup = backup/

It basically refers to the number of classes the training txt file name and the testing txt file name.

1. Create obj.names file. Every new category should be on a new line, its line number should match the category number in the *.txt* label files we created earlier.
2. Change yolo-model.cfg file as follows :-

(YOLOV2 model example)

Line 3: set batch=64, this means we will be using 64 images for every training step

Line 4: set subdivisions=8, the batch will be divided by 8 to decrease GPU VRAM requirements. If you have a powerful GPU with loads of VRAM, this number can be decreased, or batch could be increased. The training step will throw a CUDA out of memory error so you can adjust accordingly.

Line 244: set classes=1, the number of categories we want to detect

Line 237: set filters=(classes + 5)\*5 in our case filters=30

Line numbers may vary according to the cfg file

1. Train the model using the following command. Keep all configuration files, names file and .data file in /cfg directory

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

1. After running it we will have the .weights file for our use.

**CREATING MODEL AND LOADING WEIGHTS**

1. Run createmodel.py Add the name of configuration files , weight files and model name to be generated before running on lines 52 onwards.
2. Specific versions of libraries must be installed as written in YOLO-out notebook.
3. This will create the model and load weights in it and store it in the form of .h5 file that can be directly loaded.
4. Run YOLO-out jupyter notebook. This does the preprocessing of the image and generates bounding box coordinates and saves the output image as ‘outimage.txt’.
5. In[81] give the image name and path in imagePath variable.
6. In[85] post\_processing\_out(outputvector(should not be changed),name of the input image, threshold value). Name of the input image should be given and threshold value must be given.