**Documentation for Object detection from video using YOLOv3**

**Work:**

The main objective of this project is to identify objects from the video file and draw boxes around them. Here, I am going to pass a video file as the input and obtain the result with detected objects. The output is going to be live and will not be stored in any file.

**Project Structure:**

Following is the structure of the Project Directory:

Obj\_detection [Parent Folder]

│

│- .gitignore – Git ignore file

│- sample.avi – sample video file for testing obtained from internet

│- coco.names – A file containing object’s class names

│- video.py – python project file

│- yolov3.cfg – YOLO’s configuration file obtained from YOLO’s official site.

**|**- yolov3.weights – YOLO’s pre-trained model with 80 different classes obtained from YOLO’s site.

**Trained Model:**

I have used the pre-trained YOLOv3 model which can be downloaded from their official website to detect objects. The pre-trained model can identify 80 different objects. I have also provided a link to download the pre-trained model in the reference section **[0]**. We can also create our model using Darknet. Moreover creating our model takes a lot of time and more computation power. For detecting common everyday objects like Cars, Bikes, Bicycles, persons, Cats, Dogs and others using pre-trained model is the viable option.

**Algorithm:**

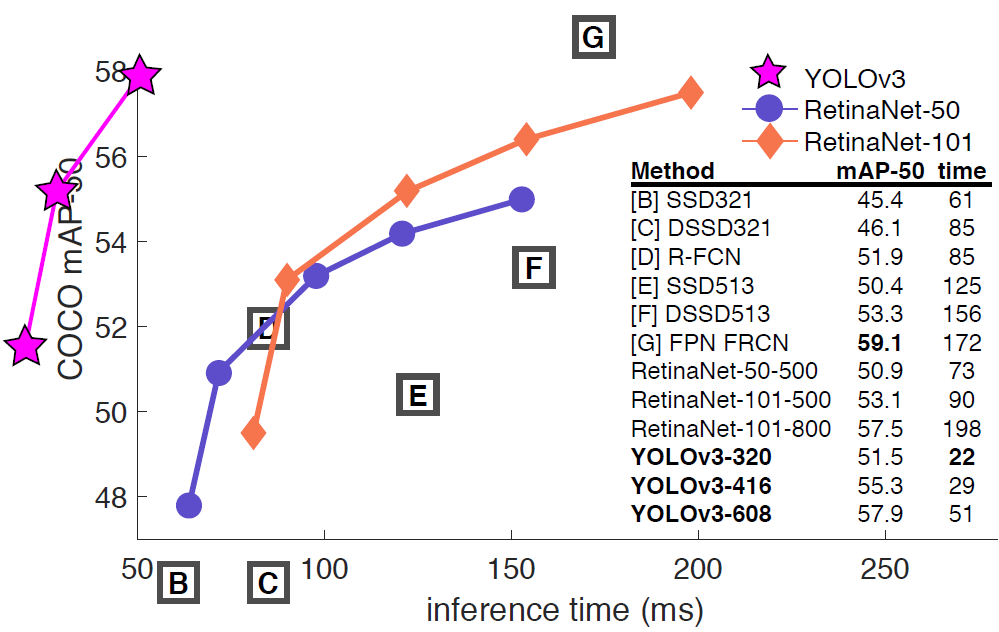
For this project, I have considered many object detection algorithms like RetinaNet, SSD, R-CNN and YOLOv3. After some surfing on the internet, I chose YOLOv3 because it is fast and highly accurate at the same time. You only look once, or YOLO is one of the faster object detection algorithms. Though it is no longer the most accurate object detection algorithm, it is a very good choice when you need real-time detection, without loss of too much accuracy. YOLO v3 uses a variant of Darknet (a C++ library), which originally has 53 layer network trained on Imagenet. For the task of detection, 53 more layers are stacked onto it, giving us a 106 layer fully convolutional underlying architecture for YOLO v3.

For more details on YOLO Algorithm see reference **[2]**

**Special Features:**

* Fast. Good for real-time processing.
* Predictions (object locations and classes) are made from one single network. Can be trained end-to-end to improve accuracy.
* YOLO is more generalized. It outperforms other methods when generalizing from natural images to other domains like artwork.
* YOLO detects one object per grid cell. It enforces spatial diversity in making predictions.

For the detailed performance of the YOLOv3, I’ve enclosed an image below which depicts the performance of all popular object detection algorithms along with YOLOv3. YOLOv3 outperformed all the other algorithms in terms of mAP(Mean Average Precision) with Time.



**Boundary of the model:**

YOLOv3 consists of convolution layers and is constructed of a deep network for improved accuracy. However, in general, it still has a lower accuracy than a two-stage detector using a region proposal stage. Therefore, YOLOv3 is unsuitable for autonomous driving applications, where high accuracy is required for dense road objects and small objects such as traffic signs and lights.

**References:**

[0] <https://pjreddie.com/darknet/yolo>

[1] <https://arxiv.org/abs/1804.02767>

[2] <https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b>

[3] <https://towardsdatascience.com/object-detection-using-yolov3-and-opencv-19ee0792a420>