**Abstract :-** This report contains all the details about the internship project guided by Jignesh Bhatt sir, which includes the tasks, approach and results achieved by the team as well as individual contribution to the project.

**Task :-** Our task was to build a cognitive machine learning model capable of detecting and analysing daily human behaviour in a confined space like a work office. Our Objective was to predict accurately the different activities a person is performing on a daily basis in real time.

**Approaches and Theory :-**

**Phase 1 :-**

We tried to implement the project by dividing an activity into various tasks, for example, drinking water is an activity which is comprised of a water bottle, a person and a particular posture, Thus if we can detect a bottle, a person and the correct posture we can formulate that the person is drinking water.

**Yolo Algorithm :-**

There are various object detection algorithms available for fast and accurate object detection. However, YOLOv3 is one of the fastest and most accurate algorithm capable of detecting at 45 fps and over 80 classes. It is trained over COCO dataset.

**Working :-**

YOLO algorithm employs convolutional neural networks (CNN) to detect objects in real-time. As the name suggests(“You look only once”), the algorithm requires only a single forward propagation through a neural network to detect objects.

This means that prediction in the entire image is done in a single algorithm run. The CNN is used to predict various class probabilities and bounding boxes simultaneously.

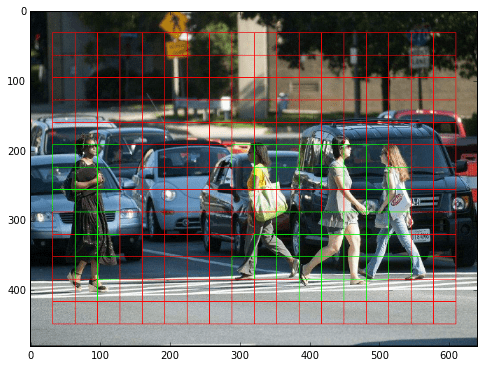
The YOLO algorithm consists of various variants. Some of the common ones include tiny YOLO and YOLOv3.

YOLO algorithm works using the following three techniques:

* Residual blocks
* Bounding box regression
* Intersection Over Union (IOU)

**Residual blocks**

First, the image is divided into various grids. Each grid has a dimension of S x S. The following image shows how an input image is divided into grids.



In the image above, there are many grid cells of equal dimension. Every grid cell will detect objects that appear within them. For example, if an object center appears within a certain grid cell, then this cell will be responsible for detecting it.

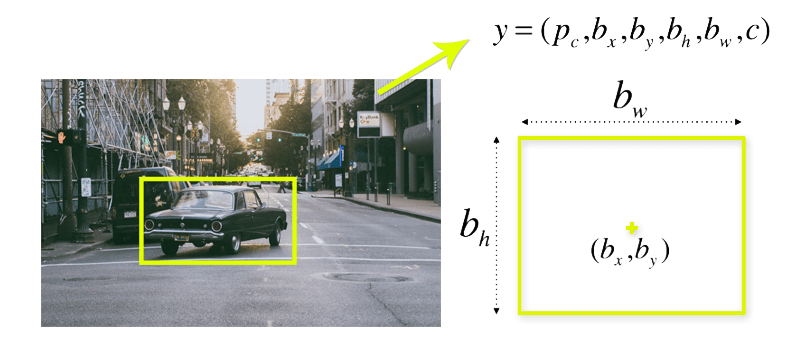
**Bounding box regression**

A bounding box is an outline that highlights an object in an image.

Every bounding box in the image consists of the following attributes:

* Width (bw)
* Height (bh)
* Class (for example, person, car, traffic light, etc.)- This is represented by the letter c.
* Bounding box center (bx,by)

The following image shows an example of a bounding box. The bounding box has been represented by a yellow outline.



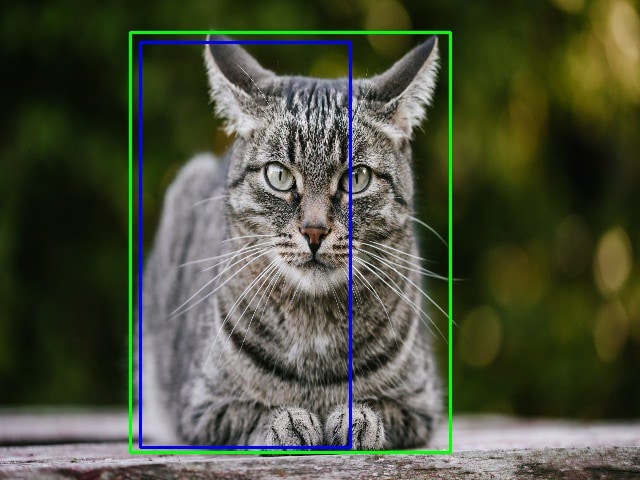
YOLO uses a single bounding box regression to predict the height, width, center, and class of objects. In the image above, represents the probability of an object appearing in the bounding box.

**Intersection over union (IOU)**

Intersection over union (IOU) is a phenomenon in object detection that describes how boxes overlap. YOLO uses IOU to provide an output box that surrounds the objects perfectly.

Each grid cell is responsible for predicting the bounding boxes and their confidence scores. The IOU is equal to 1 if the predicted bounding box is the same as the real box. This mechanism eliminates bounding boxes that are not equal to the real box.

The following image provides a simple example of how IOU works.



In the image above, there are two bounding boxes, one in green and the other one in blue. The blue box is the predicted box while the green box is the real box. YOLO ensures that the two bounding boxes are equal.

Drawbacks :- Even if the Yolo algorithm was fast and accurate, it could not perform well without a GPU and could not detect on real time. It could only provide a speed of 2 fps compared to much needed 30fps.

**Phase 2 :-**

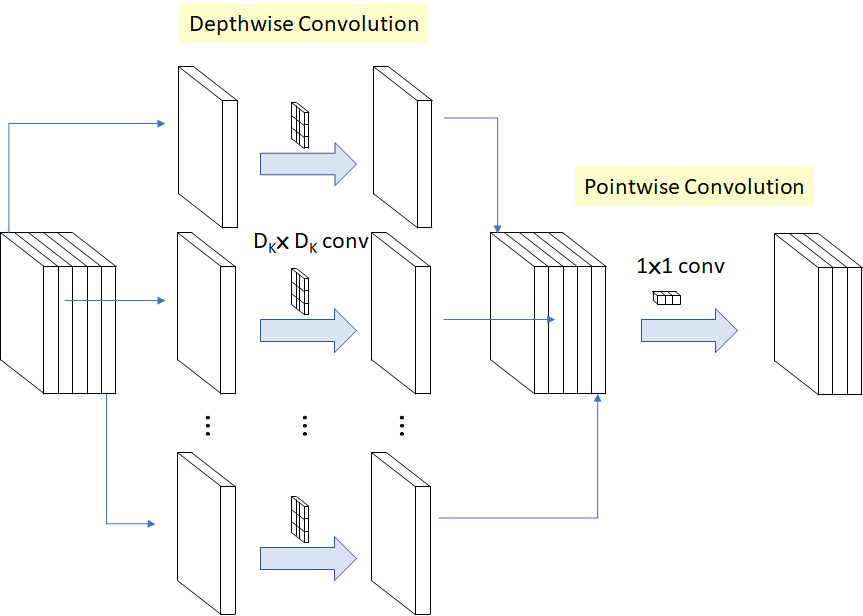
We used a pre-built architecture, specially designed by tensorflow for faster object detection. It compromises accuracy for speed, however, it yielded great results.

The architecture used two components :-

1. Mobile net base architecture
2. Multiple SSDs for object detection.

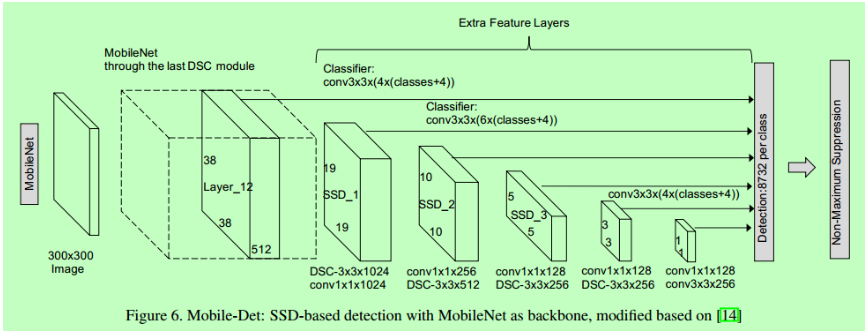
**Mobile Net architecture :-**

It is a predefined and refined architecture specially designed to provide a speed of 25 fps using Covolutional Neural Networks, by trading off accuracy. It uses depthwise separable convolution which is a depthwise convolution followed by a point wise convolution.



**SSD(Single Shot detector Algorithm):-**

The SSD architecture is a single convolution network that learns to predict bounding box locations and classify these locations in one pass. Hence, SSD can be trained end-to-end. The SSD network consists of base architecture (MobileNet in this case) followed by several convolution layers:



By using SSD, we only need to **take one single shot to detect multiple objects within the image**, while regional proposal network (RPN) based approaches such as R-CNN series that need two shots, one for generating region proposals, one for detecting the object of each proposal. Thus, SSD is much faster compared with two-shot RPN-based approaches.

**Tensorflow Object Detection Api and dataset :-**

Tensorflow has created various state of the art models for the purpose of object detection. Thus we used one of those models named **‘ssd\_mobilenet\_v2\_fpnlite\_320x320\_coco17\_tpu-8’.** We used this prebuilt architecture on the dataset of “<https://homepages.inf.ed.ac.uk/rbf/OFFICEDATA/>”. The dataset is over 30 GB and has more than 5 lakh frames which contains data for over 20 days of an office and 4 labels including :-

0 --- Room is empty (the position values are also 0)

1 --- Person is standing/walking

2 --- Person is sitting

3 --- Two or three people are talking to each other

4 --- Person in room has fallen

However, it did not suit our needs as :-

1. Very few labels
2. Labels do not provide a behavioural aspect
3. Oversized dataset required too much time to train.

So a new dataset was created manually :-

1. We collected images of ourselves doing daily office work.
2. The images were then labelled using labelimg, a python package specially efficient in labelling.
3. The labelled images and their XML files were then separated into train and test category.
4. The dataset contained 6 labels:-
   1. Working
   2. Standing
   3. Drinking Water
   4. Reading
   5. Tensed
   6. Unconscious

Advantages :-

1. Not only we created labels perfect for behavioural aspects, but it greatly reduced training time and increased accuracy.
2. We could perform real time activity detection as the model was trained especially for us.

**Implementation :-**

1. Used the model architecture from tensorflow object detection:-
   1. Tensorflow has a heap of models for the purpose of object detection. However there is a tradeoff between speed and accuracy on the models, the higher the speed, the lower the accuracy. So we chose a model concentrated in speed. i.e. **‘ssd\_mobilenet\_v2\_fpnlite\_320x320\_coco17\_tpu-8’.**
   2. The model converted images to 320x320, used them for detection, then decompressed the images for visualization.
   3. The model supported image Augmentation.
   4. Used the object\_detection api to implement the model.
2. Changed the config file according to our needs.
   1. The config file contained paths to be configured i.e. the label path, the tf record path and the number of classes.
   2. The config file also contained the checkpoint path necessaryto implement the moel with minimum loss percentage.
3. Generated the dataset.
   1. 30 different images under 6 categories were taken which amounts to 180 images.
   2. 26 images from each category were transported to train set
   3. 4 images from each category ere transported to test set.
4. Labelled the dataset
   1. LabelImg is a python software used to Label Images
   2. The images were labelled according to their category.
5. Generated tensorflow records from the labelled images so as to increase speed.
   1. A script was used to convert the images and xml Files into tensorflow records.
   2. The tf records helped speed up training process exponentially.
6. Trained the model using tf records.
   1. The model was trained for 20,000 steps achieveing a low loss function.
   2. The model was exported into a checkpoint and used for real time object detection.
7. Applied the model for real time activity detection.
   1. The checkpoint was accessed and along with OpenCV, a 30fps detection rate was achieved over 6 different labels.