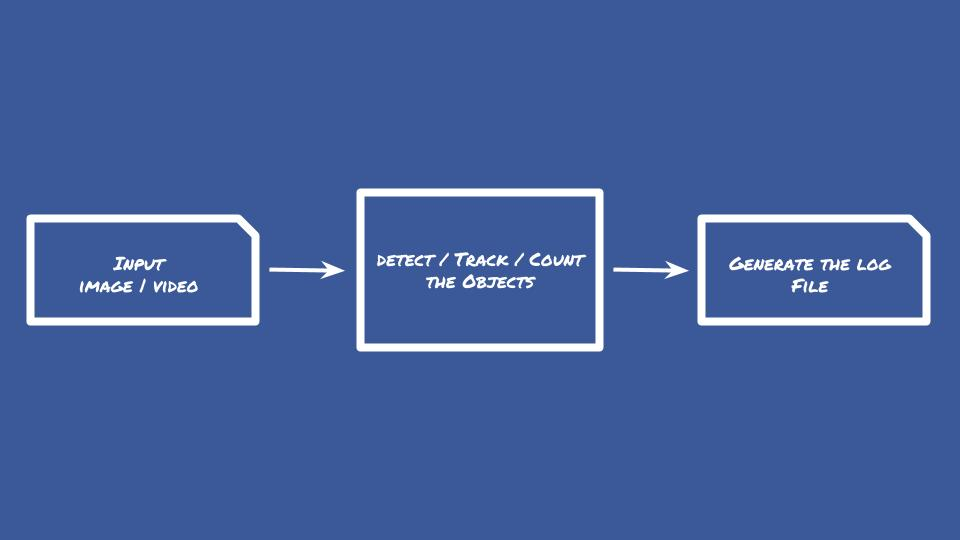
**TensorFlow Object Counting API**

The TensorFlow Object Counting API is an open source framework built on top of TensorFlow and Keras that makes it easy to develop object counting systems!

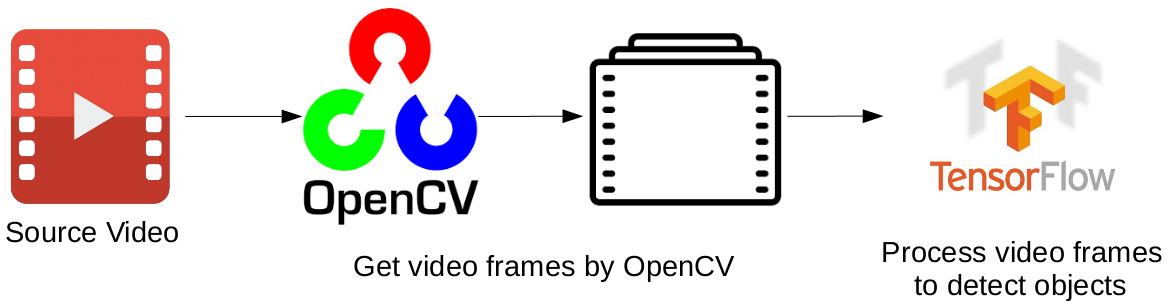


Object detection and classification have been developed on top of TensorFlow Object Detection API.

Object colour prediction has been developed using OpenCV via K-Nearest Neighbours Machine Learning Classification Algorithm is Trained Colour Histogram Features.

TensorFlow™ is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them.

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.



Source video is read frame by frame with OpenCV. Each frame is processed by the "SSD with Mobilenet" model developed on TensorFlow. This is a loop that continues working till reaching the end of the video. The main pipeline of the tracker is given at the above Figure.

Features of API -

1. Detect just the targeted objects
2. Detect all the objects
3. Count just the targeted objects
4. Count all the objects
5. Predict colour of the targeted objects
6. Predict colour of all the objects
7. Predict speed of the targeted objects
8. Predict speed of all the objects
9. Print out the detection-counting result in a .csv file as an analysis report
10. Save and store detected objects as new images under detected\_object folder
11. Select, download and use state of the art models that are trained by Google Brain Team
12. Use your own trained models or a fine-tuned model to detect specific object/s
13. Save detection and counting results as a new video or show detection and counting results in real time
14. Process images or videos depending on your requirements

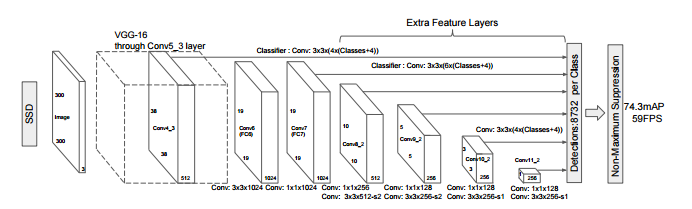
Advantages of API -

1. Lightweight, runs in real-time
2. Scalable and well-designed framework, easy usage
3. Gets "Pythonic Approach" advantages
4. It supports REST Architecture and RESTful Web Services

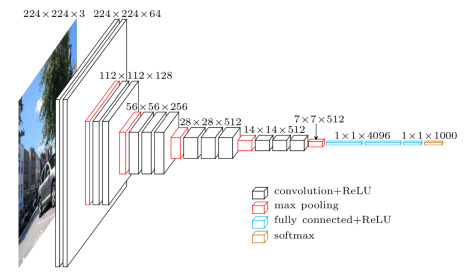
**SSD - Models**

The paper about SSD: Single Shot MultiBox Detector (by C. Szegedy et al.) was released at the end of November 2016 and reached new records in terms of performance and precision for object detection tasks, scoring over 74% mAP (mean Average Precision) at 59 frames per second on standard datasets such as PascalVOC and COCO. To better understand SSD, let’s start by explaining where the name of this architecture comes from:

1. Single Shot: this means that the tasks of object localization and classification are done in a single forward pass of the network
2. MultiBox: this is the name of a technique for bounding box regression developed by Szegedy et al. (we will briefly cover it shortly)
3. Detector: The network is an object detector that also classifies those detected objects

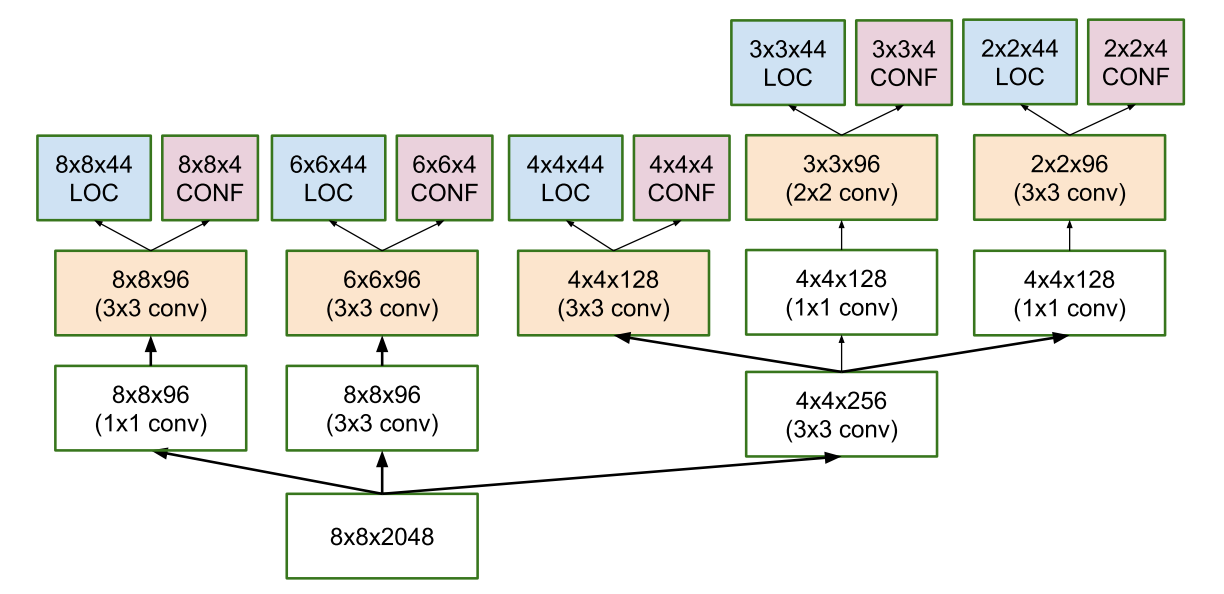
****

As you can see from the diagram above, SSD’s architecture builds on the venerable VGG-16 architecture, but discards the fully connected layers. The reason VGG-16 was used as the base network is because of its strong performance in high quality image classification tasks and its popularity for problems where transfer learning helps in improving results. Instead of the original VGG fully connected layers, a set of auxiliary convolutional layers (from conv6 onwards) were added, thus enabling the extraction of features at multiple scales and progressively decreasing the size of the input to each subsequent layer.



**MultiBox**

The bounding box regression technique of SSD is inspired by Szegedy’s work on MultiBox, a method for fast class-agnostic bounding box coordinate proposals. Interestingly, in the work done on MultiBox an Inception-style convolutional network is used. The 1x1 convolutions that you see below help in dimensionality reduction since the number of dimensions will go down (but “width” and “height” will remain the same).



MultiBox’s loss function also combined two critical components that made their way into SSD:

* Confidence Loss: this measures how confident the network is of the objectness of the computed bounding box. Categorical cross-entropy is used to compute this loss.
* Location Loss: this measures how far away the network’s predicted bounding boxes are from the ground truth ones from the training set. L2-Norm is used here.

Without delving too deep into the maths (read the paper if you are curious and want a more rigorous notation), the expression for the loss, which measures how far off our prediction “landed”, is thus:

**multibox\_loss = confidence\_loss + alpha \* location\_loss**

The alpha term helps us in balancing the contribution of the location loss. As usual in deep learning, the goal is to find the parameter values that most optimally reduce the loss function, thereby bringing our predictions closer to the ground truth.