|  |
| --- |
|  |
| CAPSTONE PROJECT - Pneumonia Detection Challenge |
| FINAL REPORT |
|  |
| |  |  |  |  | | --- | --- | --- | --- | | **REVISION HISTORY** | | | | | **S. No** | **Author** | **Remarks** | **Modified Date** | | 2. | Karthikeyan, Jagan, Rajani, Ricky, Abhi | V.1.01 | 03/11/2020 | |
|  |

|  |
| --- |
|  |

Table of Contents

[1. Problem Statement:- 3](#_Toc34870525)

[ Current Scenario: 3](#_Toc34870526)

[ Opportunity for Improvement: 3](#_Toc34870527)

[2. Data Description: 3](#_Toc34870528)

[ File descriptions 3](#_Toc34870529)

[ Data fields 3](#_Toc34870530)

[3. EDA Analysis and Pre- Processing :- 4](#_Toc34870531)

[ EDA Conclusion: 11](#_Toc34870532)

[ Pre-Processing: 12](#_Toc34870533)

[4. Models and Model Building :- 12](#_Toc34870534)

[ Mask R-CNN Model : 12](#_Toc34870535)

[ MobileNet Model: 15](#_Toc34870536)

[ YOLO Model - YOU ONLY LOOK ONCE 18](#_Toc34870537)

[ INCEPTION NETWORK MODEL 22](#_Toc34870538)

[ SSD MODEL – SINGLE SHOT MULTIBOX DETECTOR 25](#_Toc34870539)

[5. Models Performance Improvement: - 28](#_Toc34870540)

# Problem Statement: -

### Current Scenario:

In current scenario, highly trained specialist requires review of CXR to detect Pneumonia. However, the diagnosis of pneumonia on CXR is complicated because of several other conditions in the lungs such as fluid overload (pulmonary edema), bleeding, volume loss (atelectasis or collapse), lung cancer, or post-radiation or surgical changes. Outside of the lungs, fluid in the pleural space (pleural effusion) also appears as increased opacity on CXR.

### Opportunity for Improvement:

We need to detect Inflammation of the lungs, for this we need to build an algorithm to detect a visual signal for pneumonia in medical images to automatically locate lung opacities on chest radiographs.

# Data Description:

The training data is provided as a set of PATIENT IDs and bounding boxes. Bounding boxes are defined as follows: x-min y-min width height

There is also a binary target column, Target, indicating pneumonia or non-pneumonia.

There may be multiple rows per PATIENT IDs.

Samples without bounding boxes are negative and contain no definitive evidence of pneumonia. Samples with bounding boxes indicate evidence of pneumonia.

When making predictions, competitors should predict as many bounding boxes as they feel are necessary, in the format: confidence x-min y-min width height

### File descriptions

* stage\_2\_train.csv - The training set, contains PATIENT IDs and bounding box / target information.
* stage\_2\_sample\_submission.csv - A sample submission file in the correct format. Contains PATIENT IDs for the test set. Note that the sample submission contains one box per image, but there is no limit to the number of bounding boxes that can be assigned to a given image.
* stage\_2\_detailed\_class\_info.csv - Provides detailed information about the type of positive or negative class for each image.

### Data fields

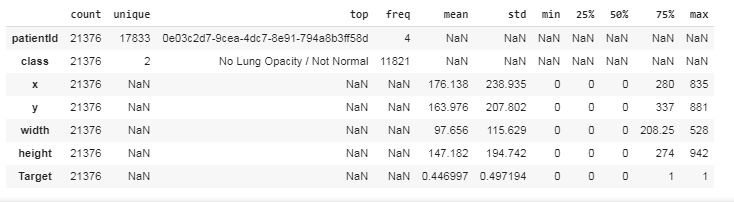
* patientId \_- A patientId. Each patientId corresponds to a unique image.
* x\_ - the upper-left x coordinate of the bounding box.
* y\_ - the upper-left y coordinate of the bounding box.
* width\_ - the width of the bounding box.
* height\_ - the height of the bounding box.
* Target\_ - the binary Target, indicating whether this sample has evidence of pneumonia.

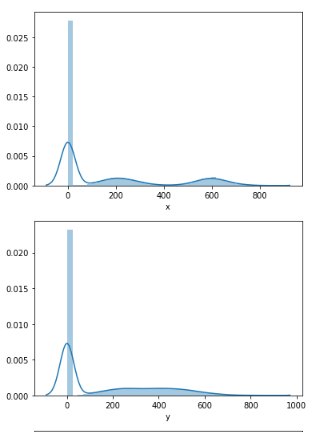
# EDA Analysis and Pre- Processing: -

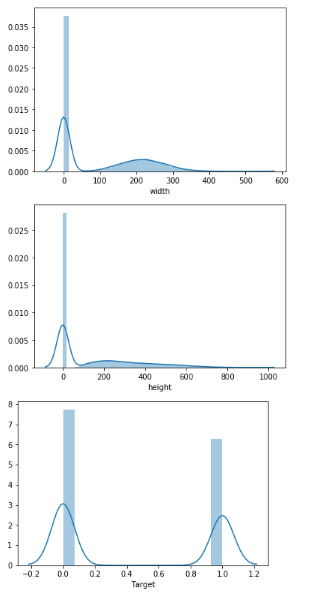
The following analysis/observations were made on the data:-

* **Skewness Check on basic data:**

**Observation:** All the features are Right Skewed





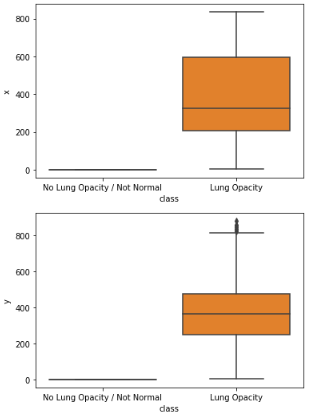


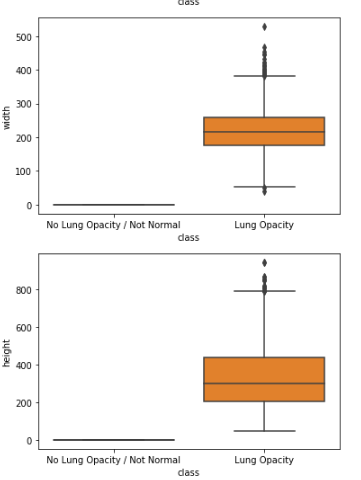
* **Outliers on basic data (By splitting a dataframe into 2, Normal cases and Pneumonic cases) :**

**Observation:**

From the below box plot we can see outliers in most of the cases, especially in y, width, height. If we see from our basic describe command y max --> 881, width max --> 528, height max --> 942

There are 4 outliers, in bounding box coordinates [We are going to keep this as it is]

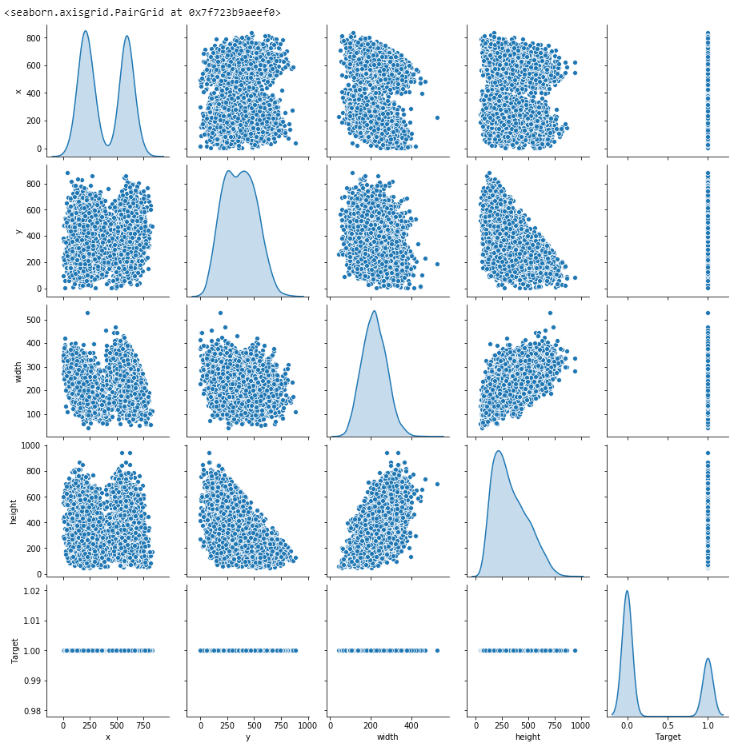




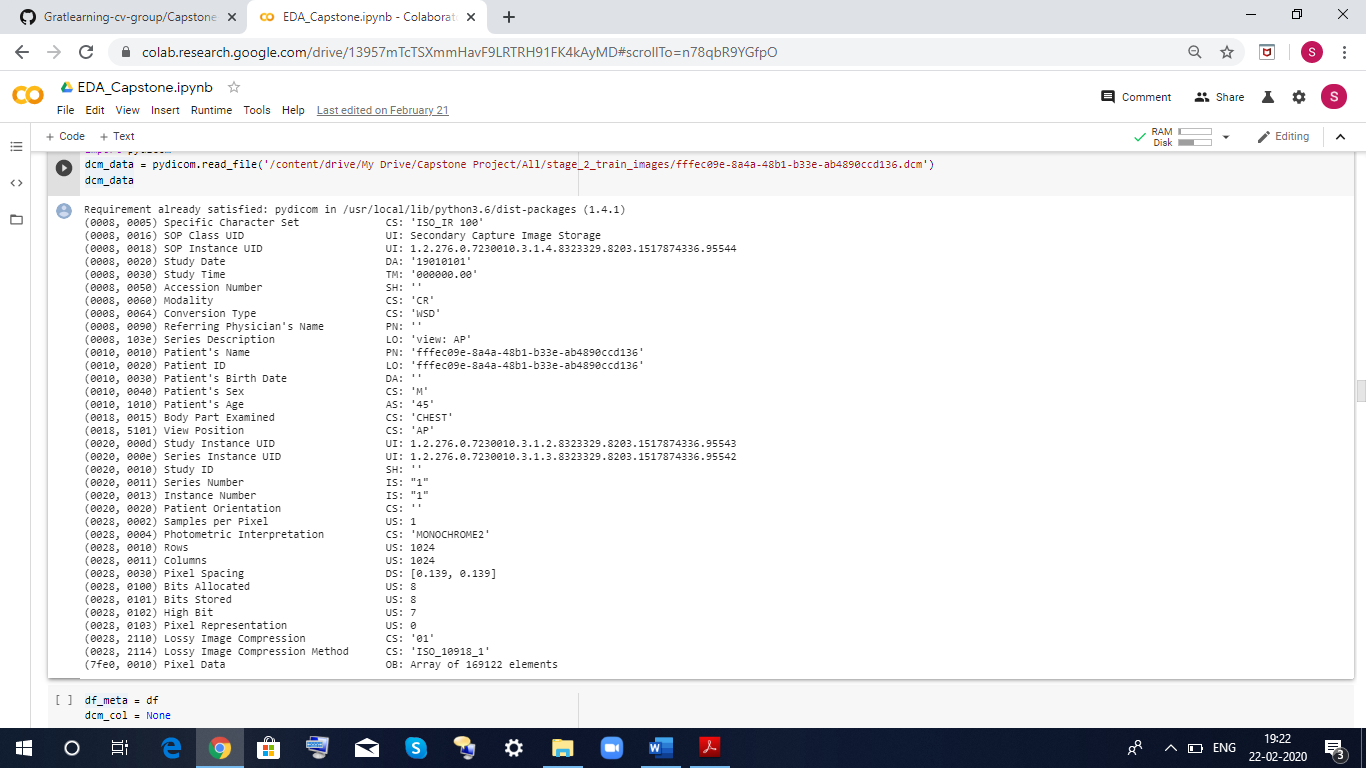
* **Correlation between features:**

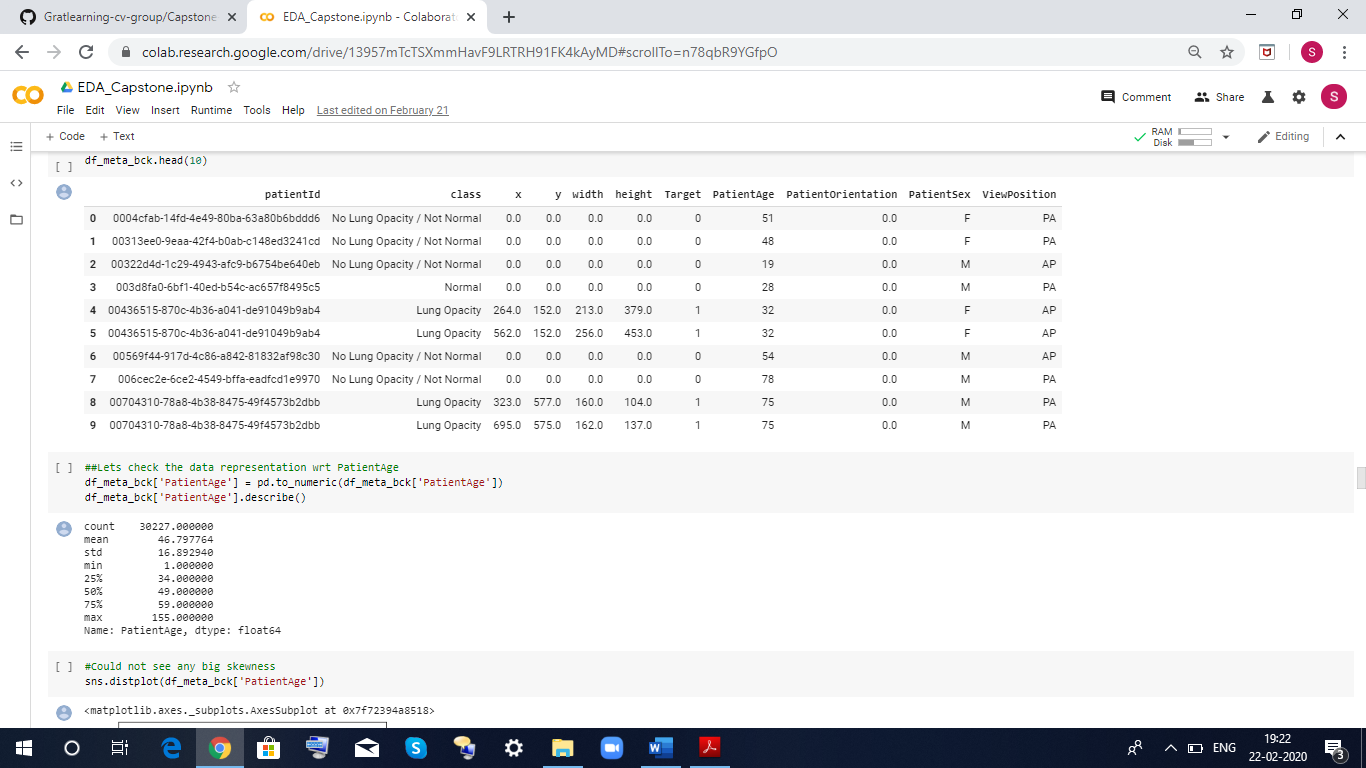
**Observation:**

We could not see linear relatioship between features, but could observesome kind of relationship between height and width / height and weight, y axis and height



* Took dcm data along with basic data, kept only those features which are related to the patients and checked for correlations, between these features

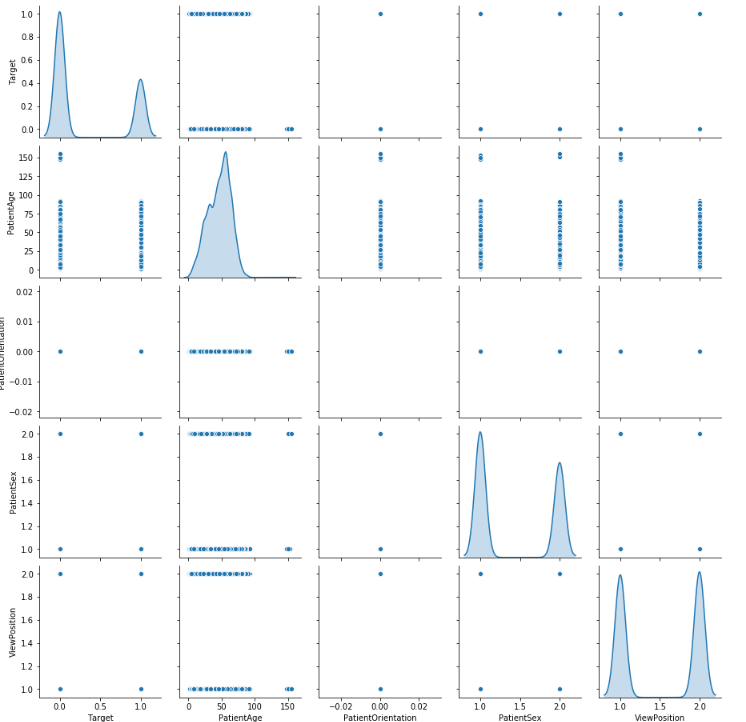




* **Correlation between basic data cum dcm data as below:**

**Observation:**

We could see clearly outliers in Patients Age, Patient Sex, View Position influencing Target i.e. whether the person is having lung opacity or not



* **Checked for cases pertaining to AP and PA**

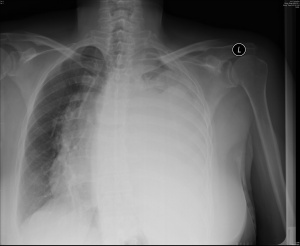
Overview of what is AP and what is PA?

* Posterior - Anterior (PA). This is the most common and preferred type of chest X-Ray. Posterior - anterior refers to the direction of the X-Ray beam travel.; i e. X-Ray beams hit the posterior part of the chest before the anterior part. To obtain the image, the patient is asked to stand with their chest against the film, to hold their arms up or to the sides and roll their shoulders forward. The X-ray technician may then ask the patient to take few deep breaths and hold it for a couple of seconds. This techniques of holding the breath generally helps to get a clear picture of the heart and lungs on the image.

[](https://www.physio-pedia.com/File:Chest_X-ray_2346.jpg)

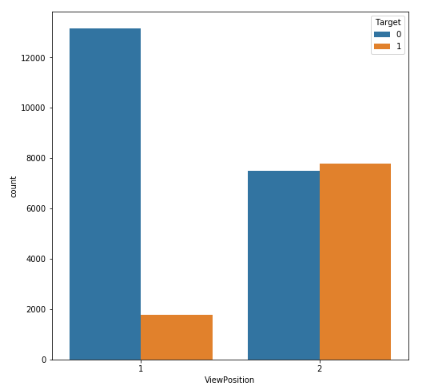
PA Chest X-Ray

* Anterior - posterior (AP): This type of chest X-Ray is generally less preferred because the image of the heart and mediastinum is less clear and focused in this projection. To obtain AP image, the patient is asked to stand with their back against the film. If the patient is unable to stand, an AP image can also be taken with the patient sitting or supine on the bed.

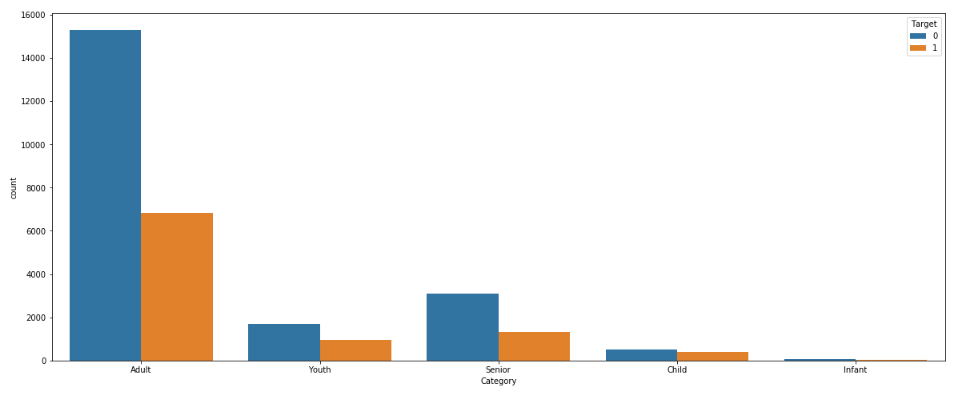
[](https://www.physio-pedia.com/File:AP_Chest_X-Ray.jpg)

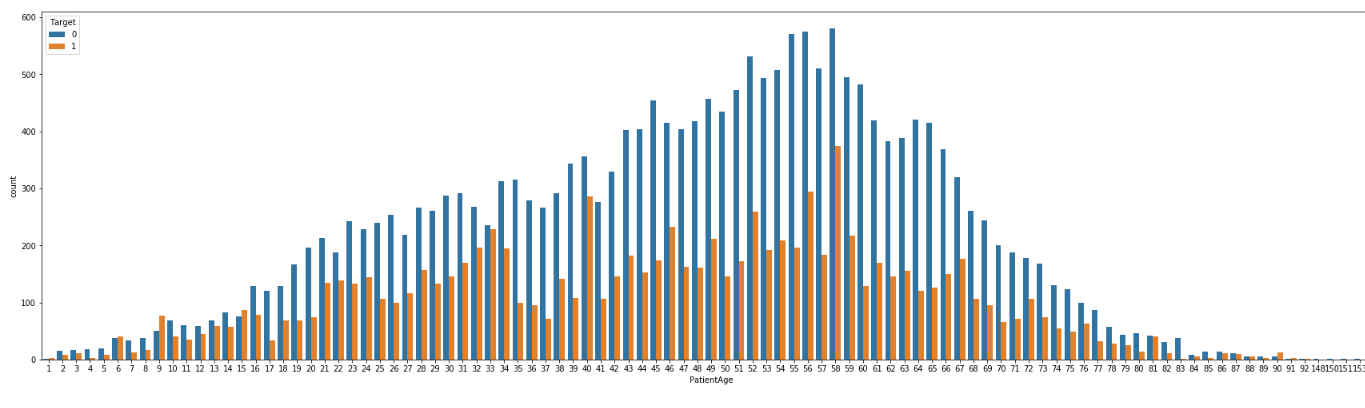
AP Chest X-Ray

* Count Plot with respect to Target



* Categorized the data age wise and analysed in which age group pneumonia reported high





### EDA Conclusion:

* 1. From Meta data 'PatientAge', 'PatientSex', 'ViewPosition' features can be useful in predicting the Target
  2. Could not get information wrt bounding box Coordinates
  3. There is some skewness between lung opacity cases and true Normal cases [If we exclude 'No Lung Opacity/Normal cases']
  4. 5 rows (5 patients) have been dropped as obvious outliers.

### Pre-Processing:

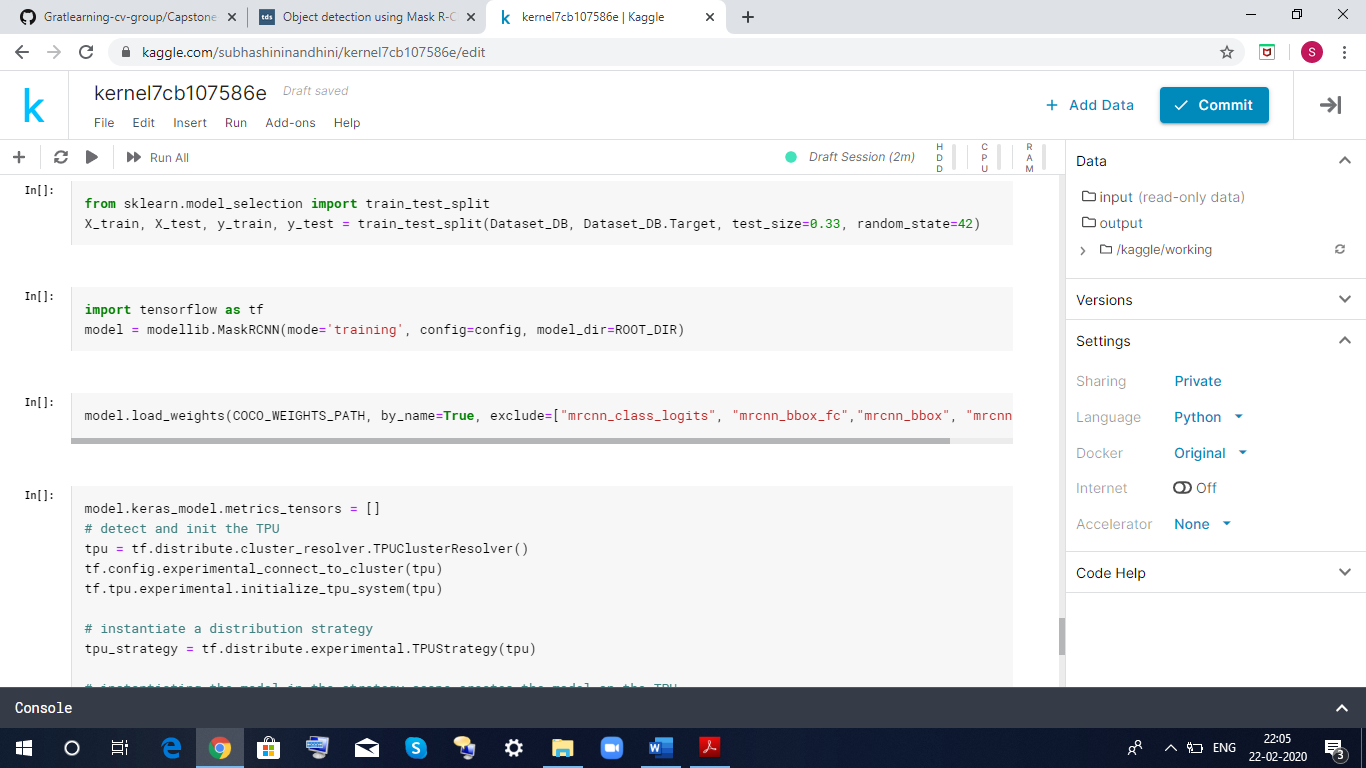
Following steps were performed for Data Pre-Processing: -

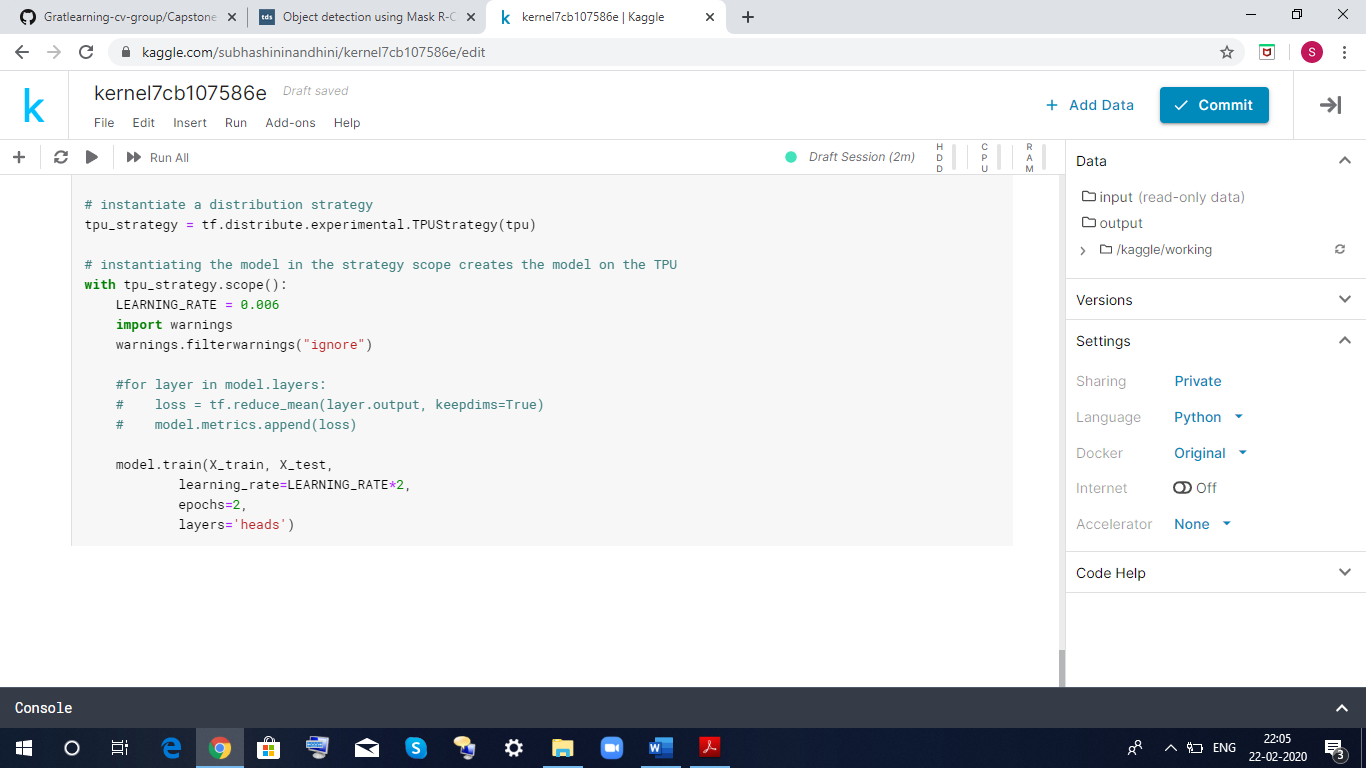
* Reading the given data in the Data frames.
* Merger the Labels with features based on Patient ID
* Get the annotation (Boundary box values) and merge it with image file path, features and Target based on the Patient ID.
* Set the Image Height and width to 1024.
* The final Dataset will have "patientId",  1:"class", 2:'x',3:'y', 4:'width', 5:'height', 6:'Target' 7:'path', 8:'annotation', 9:'original\_height', 10:'original\_width'.

# Models and Model Building: -

### Mask R-CNN Model:

We will be using Mask R-CNN model because Mask R-CNN uses anchor boxes to detect multiple objects, objects of different scales, and overlapping objects in an image. This improves the speed and efficiency for object detection. Mask R-CNN have a branch for classification and bounding box regression. It usesResNet101 architecture to extract features from image and Region Proposal Network (RPN) to generate Region of Interests (RoI).



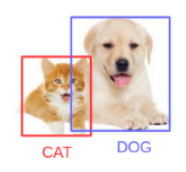


Mask R-CNN is basically an extension of Faster R-CNN. Faster R-CNN is widely used for object detection tasks. For a given image, it returns the class label and bounding box coordinates for each object in the image.

 So, let’s say you pass the following image:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/07/Screenshot-from-2019-07-18-15-51-45.png)

The Fast R-CNN model will return something like this:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/07/Screenshot-from-2019-07-18-15-52-17.png)

The Mask R-CNN framework is built on top of Faster R-CNN. So, for a given image, Mask R-CNN, in addition to the class label and bounding box coordinates for each object, will also return the object mask.

Let’s first quickly understand how Faster R-CNN works. This will help us grasp the intuition behind Mask R-CNN as well.

Faster R-CNN first uses a ConvNet to extract feature maps from the images

These feature maps are then passed through a Region Proposal Network (RPN) which returns the candidate bounding boxes

We then apply an RoI pooling layer on these candidate bounding boxes to bring all the candidates to the same size

And finally, the proposals are passed to a fully connected layer to classify and output the bounding boxes for objects

Once you understand how Faster R-CNN works, understanding Mask R-CNN will be very easy. So, let’s understand it step-by-step starting from the input to predicting the class label, bounding box, and object mask.

**Backbone Model**

Similar to the [ConvNet](https://www.analyticsvidhya.com/blog/2018/12/guide-convolutional-neural-network-cnn/?utm_source=blog&utm_medium=computer-vision-implementing-mask-r-cnn-image-segmentation" \t "_blank) that we use in Faster R-CNN to extract feature maps from the image, we use the ResNet 101 architecture to extract features from the images in Mask R-CNN. So, the first step is to take an image and extract features using the ResNet 101 architecture. These features act as an input for the next layer.

 Region Proposal Network (RPN)

Now, we take the feature maps obtained in the previous step and apply a region proposal network (RPM). This basically predicts if an object is present in that region (or not). In this step, we get those regions or feature maps which the model predicts contain some object.

 Region of Interest (RoI)

The regions obtained from the RPN might be of different shapes, right? Hence, we apply a pooling layer and convert all the regions to the same shape. Next, these regions are passed through a fully connected network so that the class label and bounding boxes are predicted.

Till this point, the steps are almost like how Faster R-CNN works. Now comes the difference between the two frameworks. In addition to this, Mask R-CNN also generates the segmentation mask.

For that, we first compute the region of interest so that the computation time can be reduced. For all the predicted regions, we compute the Intersection over Union (IoU) with the ground truth boxes. We can computer IoU like this:

IoU = Area of the intersection / Area of the union

**Now, only if the IoU is greater than or equal to 0.5, we consider that as a region of interest. Otherwise, we neglect that particular region. We do this for all the regions and then select only a set of regions for which the IoU is greater than 0.5.**

**Steps to implement Mask R-CNN: -**

It’s time to perform some image segmentation tasks! We will be using the Mask R-CNN Framework created by the Data scientists and researchers at Facebook AI Research (FAIR).

Let’s have a look at the steps which we will follow to perform image segmentation using Mask R-CNN.

**Step 1: Clone the repository**

First, we will clone the mask rcnn repository which has the architecture for Mask R-CNN. Use the following command to clone the repository:

git clone <https://github.com/matterport/Mask_RCNN.git>

Once this is done, we need to install the dependencies required by Mask R-CNN.

**Step 2: Install the dependencies**

Here is a list of all the dependencies for Mask R-CNN:

* numpy
* scipy
* Pillow
* cython
* matplotlib
* scikit-image
* tensorflow>=1.3.0
* keras>=2.0.8
* opencv-python
* h5py
* imgaug
* IPython

**You must install all these dependencies before using the Mask R-CNN framework.**

**Step 3: Download the pre-trained weights (trained on MS COCO)**

Next, we need to download the pretrained weights. These weights are obtained from a model that was trained on the MS COCO dataset. Once you have downloaded the weights, paste this file in the samples folder of the Mask RCNN repository that we cloned in step 1.

**Step 4: Predicting for our image**

1. Finally, we will use the Mask R-CNN architecture and the pretrained weights to generate predictions for our own images.
2. Once you’re done with these four steps. We will implement all these things in Python and then generate the masks along with the classes and bounding boxes for objects in our images.

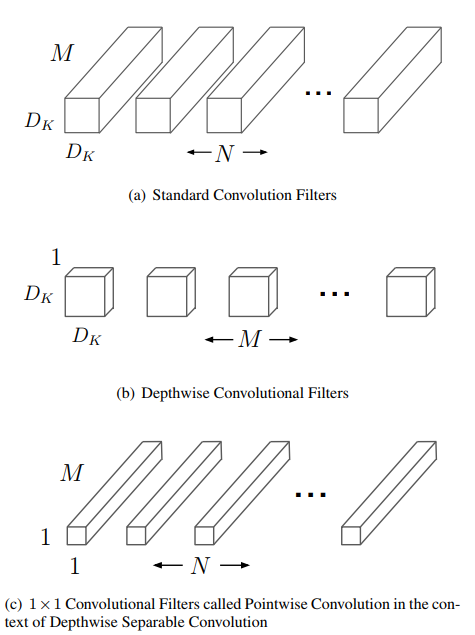
### MobileNet Model:

The second model which we will use is MobileNet and following are the reasons to use this model: -

* MobileNet are light weight deep neural networks and are based on a streamlined architecture that uses depth wise separable convolutions.
* MobileNet uses two simple global hyper parameters that efficiently trades off between accuracy and latency.
* Reduced network size - 17MB and Reduced number of parameters - 4.2 million.
* Faster in performance and are useful for mobile applications.
* Small, low-latency convolution neural network.

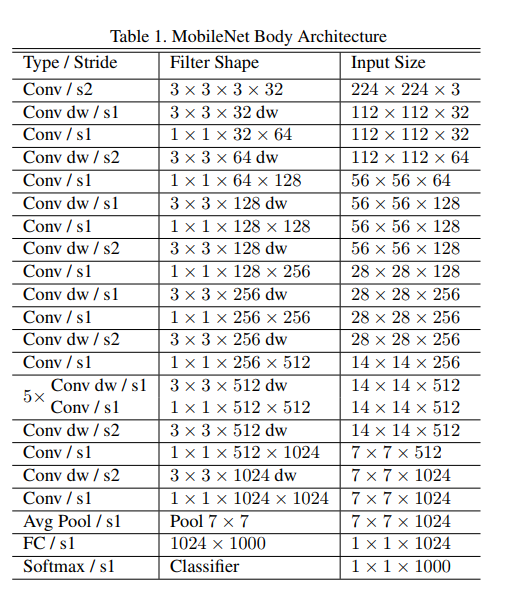
Mobilenet is a lightweight in its architecture. It uses depth wise separable convolutions which basically means it performs a single convolution on each colour channel rather than combining all three and flattening it. This has the effect of filtering the input channels. Or as the authors of the paper explain clearly:

“For MobileNets the depth wise convolution applies a single filter to each input channel. The pointwise convolution then applies a 1×1 convolution to combine the outputs the depth wise convolution. A standard convolution both filters and combines inputs into a new set of outputs in one step. The depthwise separable convolution splits this into two layers, a separate layer for filtering and a separate layer for combining. This factorization has the effect of drastically reducing computation and model size.”



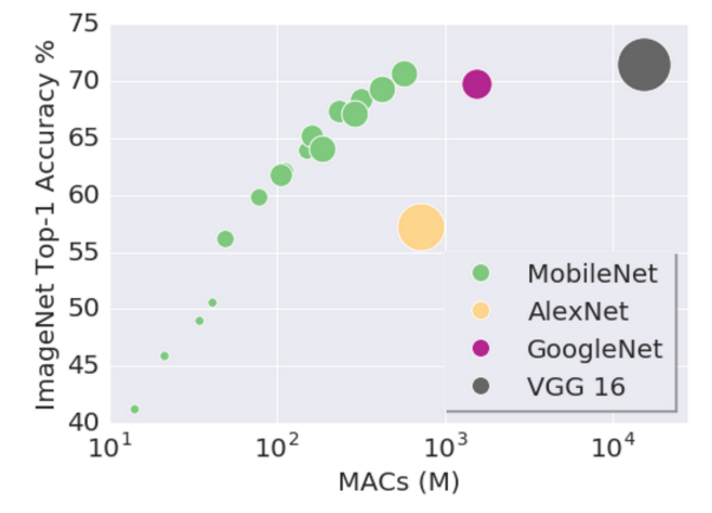
The overall architecture of the Mobile net is as follows, having 30 layers with

1. convolutional layer with stride 2
2. depth wise layer
3. pointwise layer that doubles the number of channels
4. depth wise layer with stride 2
5. pointwise layer that doubles the number of channels



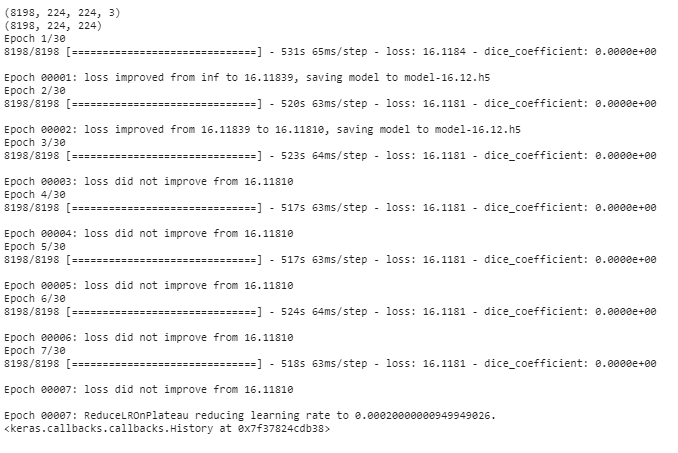
**Mobilenet full architecture**

It is also very low maintenance thus performing quite well with high speed. There are also many flavors of pre-trained models with the size of the network in memory and on disk being proportional to the number of parameters being used. The speed and power consumption of the network is proportional to the number of MACs (Multiply-Accumulates) which is a measure of the number of fused Multiplication and Addition operations.



**OUR IMPLEMENTATION**

**While Building the Mobile Net model Total parameters were 3,230,852, Trainable parameters were 3,208,964 and Non-trainable parameters were 21,888**



### YOLO Model - YOU ONLY LOOK ONCE

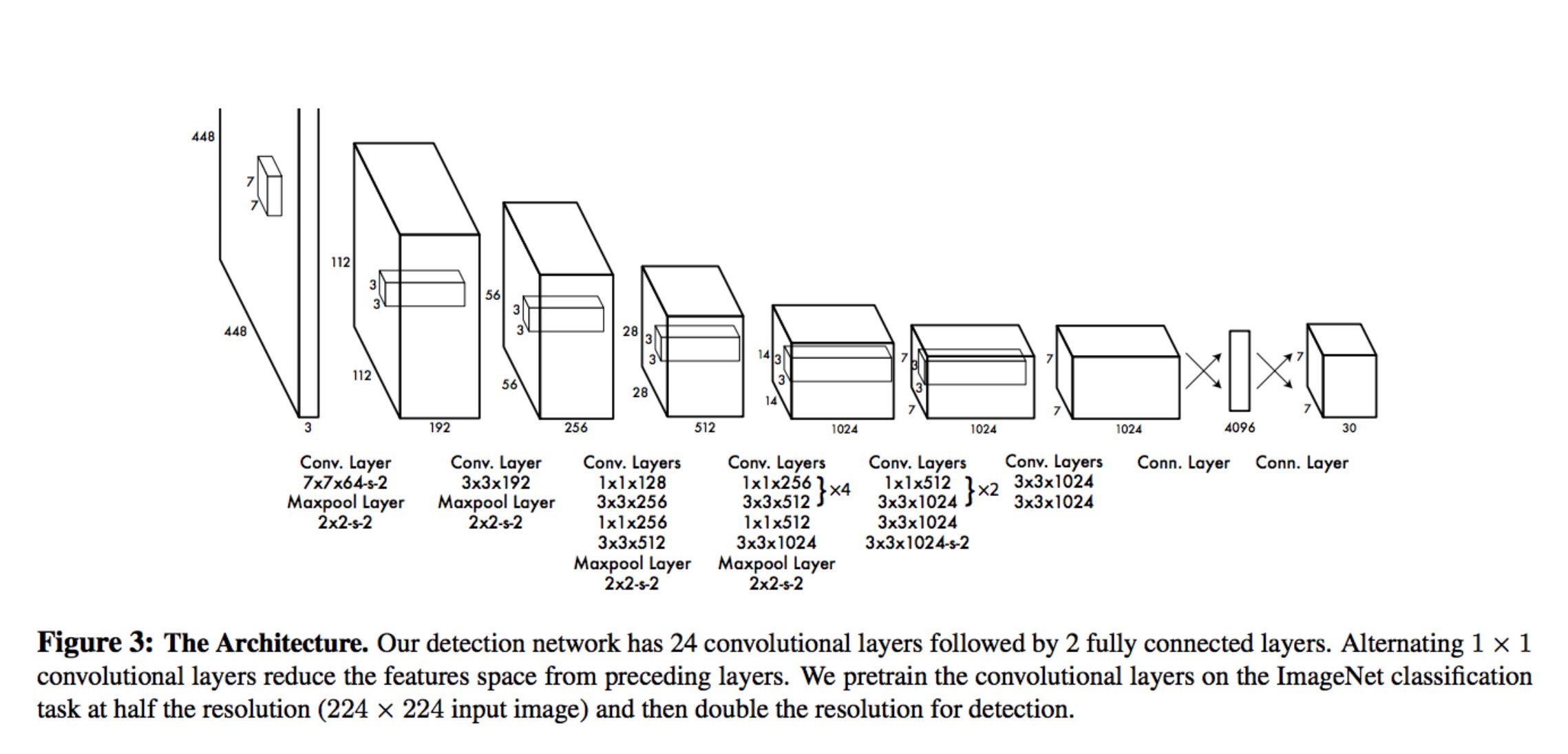
You Only Look Once (YOLO) is a state-of-the-art, real-time object detection system. On a Pascal Titan X it processes images at 30 FPS and has a map of 57.9% on COCO test-dev.

**HOW IT WORKS!!**

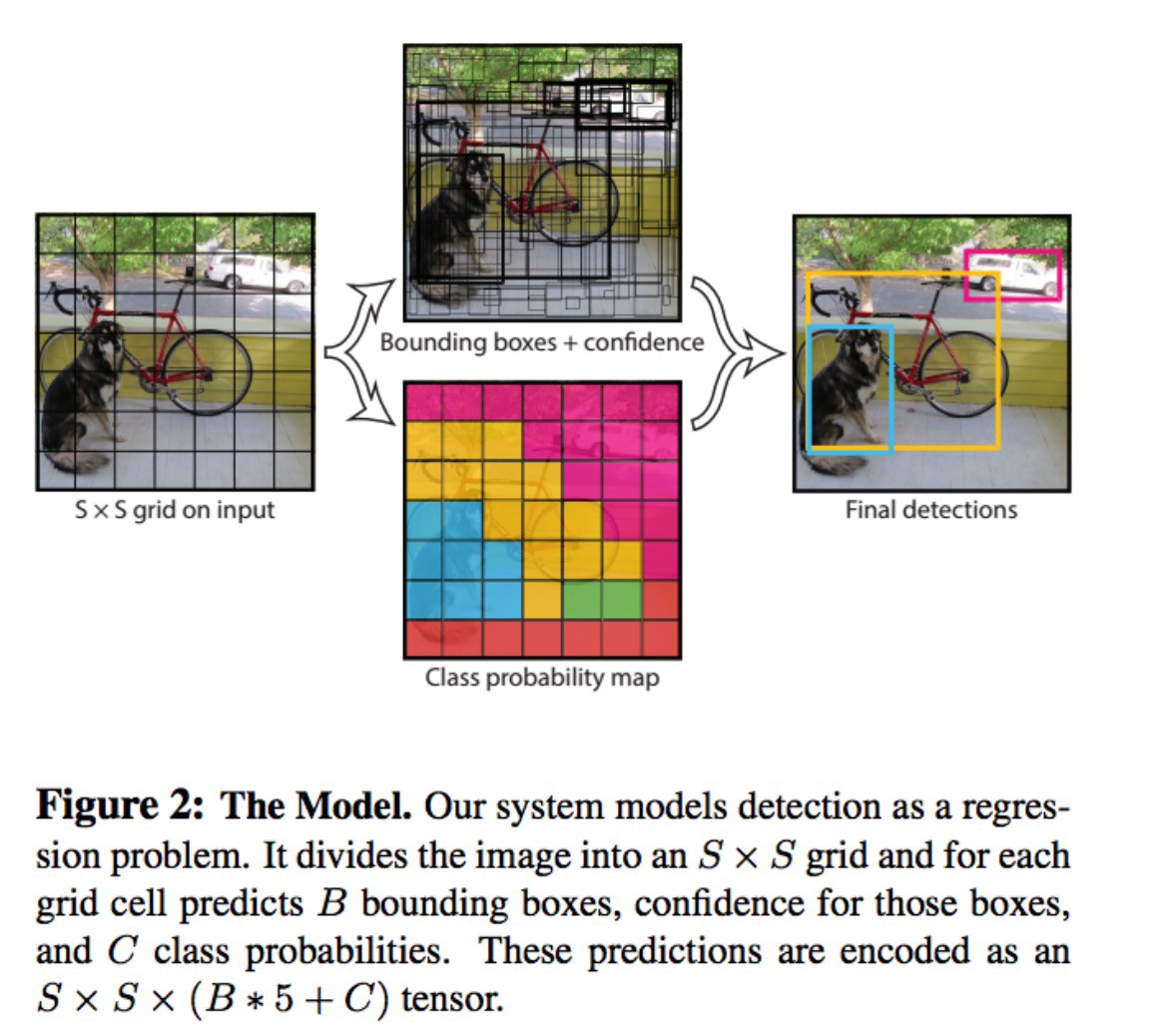
Prior detection systems repurpose classifiers or localizers to perform detection. They apply the model to an image at multiple locations and scales. High scoring regions of the image are considered detections. We use a totally different approach. We apply a single neural network to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

Model has several advantages over classifier-based systems. It looks at the whole image at test time, so its predictions are informed by global context in the image. It also makes predictions with a single network evaluation unlike systems like [R-CNN](https://github.com/rbgirshick/rcnn) which require thousands for a single image. This makes it extremely fast, more than 1000x faster than R-CNN and 100x faster than [Fast R-CNN](https://github.com/rbgirshick/fast-rcnn). See our [paper](https://pjreddie.com/media/files/papers/YOLOv3.pdf) for more details on the full system.

YOLO network has 24 convolutional layers followed by 2 fully connected layers. Instead of the inception modules used by GoogLeNet, we simply use 1 × 1 reduction layers followed by 3 × 3 convolutional layers. Fast YOLO uses a neural network with fewer convolutional layers (9 instead of 24) and fewer filters in those layers. Other than the size of the network, all training and testing parameters are the same between YOLO and Fast YOLO.



The Model detection as a regression problem. It divides the image into an S x S grid and for each grid cell predicts B bounding box, confidence for those boxes and C class probabilities. These predictions are encoded as an S x S x (B\*5 + C) tensor.



**Biggest advantages:**

* Speed (45 frames per second — better than Realtime)
* Network understands generalized object representation (This allowed them to train the network on real world images and predictions on artwork was still fairly accurate).
* faster version (with smaller architecture) — 155 frames per sec but is less accurate.
* open source: <https://pjreddie.com/darknet/yolo/>

**Limitations of YOLO**

YOLO imposes strong spatial constraints on bounding box predictions since each grid cell only predicts two boxes and can only have one class. This spatial constraint limits the number of nearby objects that our model can predict. Our model struggles with small objects that appear in groups, such as flocks of birds. Since our model learns to predict bounding boxes from data, it struggles to generalize to objects in new or unusual aspect ratios or configurations. Our model also uses relatively coarse features for predicting bounding boxes since our architecture has multiple down sampling layers from the input image. Finally, while we train on a loss function that approximates detection performance, our loss function treats errors the same in small bounding boxes versus large bounding boxes. A small error in a large box is generally benign but a small error in a small box has a much greater effect on IOU. Our main source of error is incorrect localizations

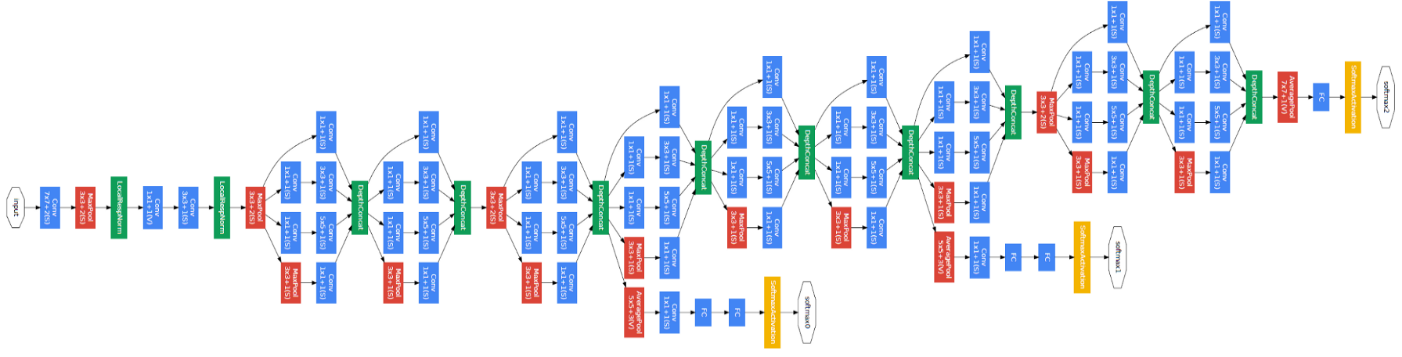
OUR IMPLEMENTATION

### INCEPTION NETWORK MODEL

**Inception network was once considered a state-of-the-art deep learning architecture (or model) for solving image recognition and detection problems**.

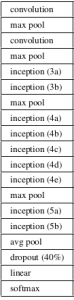
It put forward a breakthrough performance on the ImageNet Visual Recognition Challenge (in 2014), which is a reputed platform for benchmarking image recognition and detection algorithms. Along with this, it set off a ton of research in the creation of new deep learning architectures with innovative and impactful ideas.

We will go through the main ideas and suggestions propounded in the paper and try to grasp the techniques within.



**Inception Architecture**

The paper proposes a new type of architecture – GoogLeNet or Inception v1. It is basically a convolutional neural network (CNN) which is 27 layers deep. Below is the model summary:

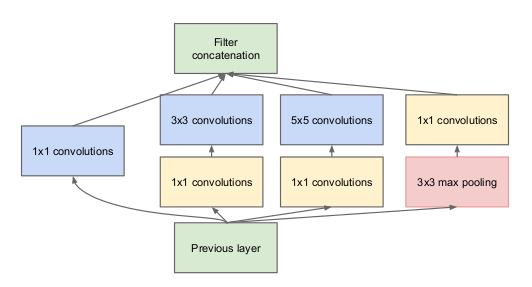


The above image that there is a layer called inception layer. The inception layer is the core concept of a sparsely connected architecture.

***“(Inception Layer) is a combination of all those layers (namely, 1×1 Convolutional layer, 3×3 Convolutional layer, 5×5 Convolutional layer) with their output filter banks concatenated into a single output vector forming the input of the next stage.”***

long with the above-mentioned layers, there are two major add-ons in the original inception layer:1×1 Convolutional layer before applying another layer, which is mainly used for dimensionality reduction

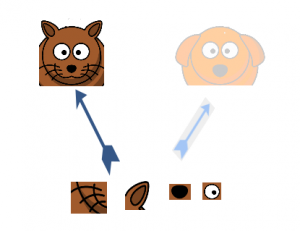
A parallel Max Pooling layer, which provides another option to the inception layer



**Inception Layer**

To understand the importance of the inception layer’s structure, the author calls on the Hebbian principle from human learning. This says that “neurons that fire together, wire together”. The author suggests that when creating a subsequent layer in a deep learning model, one should pay attention to the learnings of the previous layer.

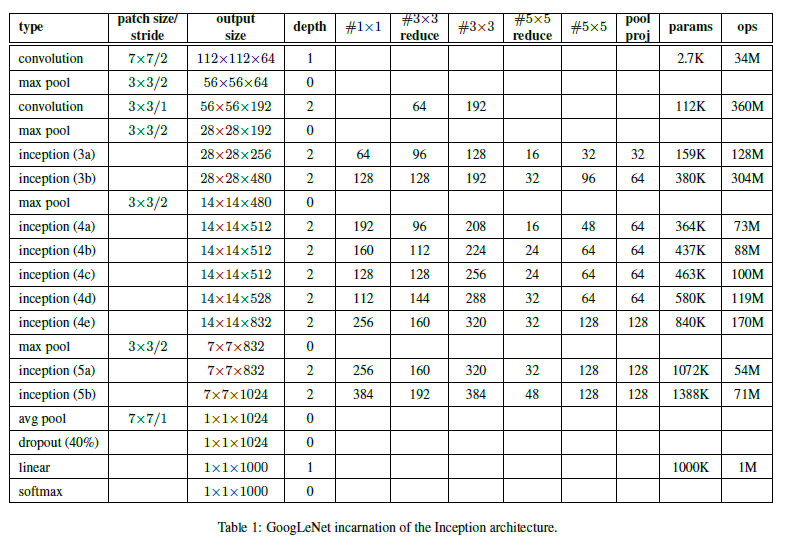
Suppose, for example, a layer in our deep learning model has learned to focus on individual parts of a face. The next layer of the network would probably focus on the overall face in the image to identify the different objects present there. Now to do this, the layer should have the appropriate filter sizes to detect different objects.



This is where the inception layer comes to the fore. It allows the internal layers to pick and choose which filter size will be relevant to learn the required information. So even if the size of the face in the image is different (as seen in the images below), the layer works accordingly to recognize the face. For the first image, it would probably take a higher filter size, while it’ll take a lower one for the second image.

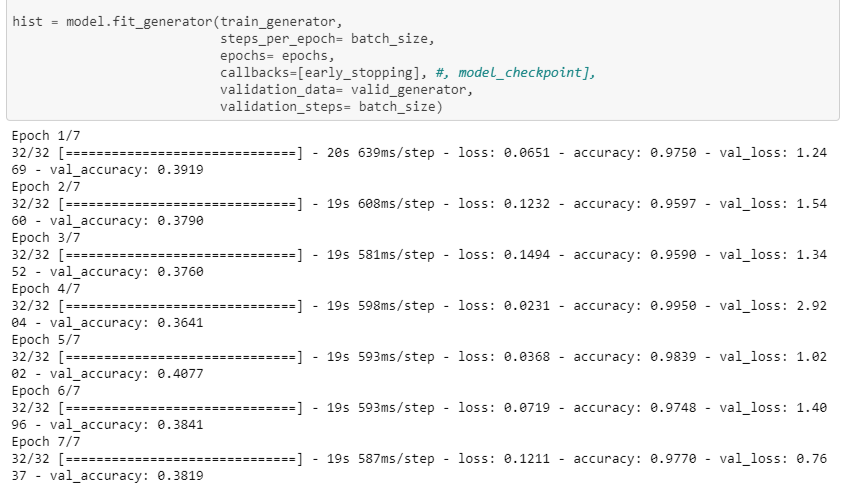


The overall architecture, with all the specifications, looks like this:



OUR IMPLEMENTATION

Accuracy Score for our Model Implemented:



### SSD MODEL – SINGLE SHOT MULTIBOX DETECTOR

SSD is designed for object detection in real-time. Faster R-CNN uses a region proposal network to create boundary boxes and utilizes those boxes to classify objects. While it is considered the start-of-the-art in accuracy, the whole process runs at 7 frames per second. Far below what a real-time processing need. SSD speeds up the process by eliminating the need of the region proposal network. To recover the drop-in accuracy, SSD applies a few improvements including multi-scale features and default boxes. These improvements allow SSD to match the Faster R-CNN’s accuracy using lower resolution images, which further pushes the speed higher. According to the following comparison, it achieves the real-time processing speed and even beats the accuracy of the Faster R-CNN. (Accuracy is measured as the mean average precision MAP: the precision of the predictions.)

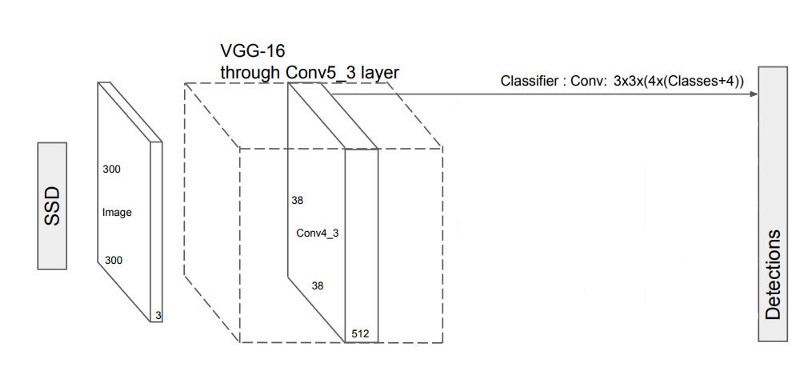


**SSD Object detection**

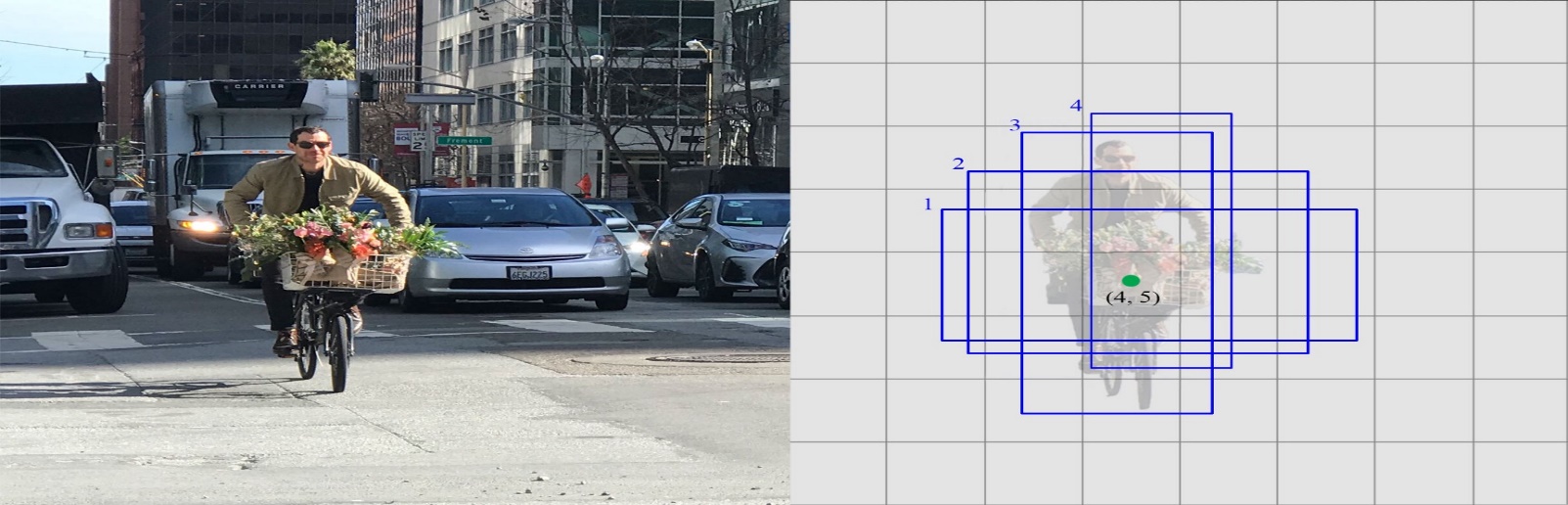


**The SSD object detection composes of 2 parts:**

* Extract feature maps, and
* Apply convolution filters to detect objects.

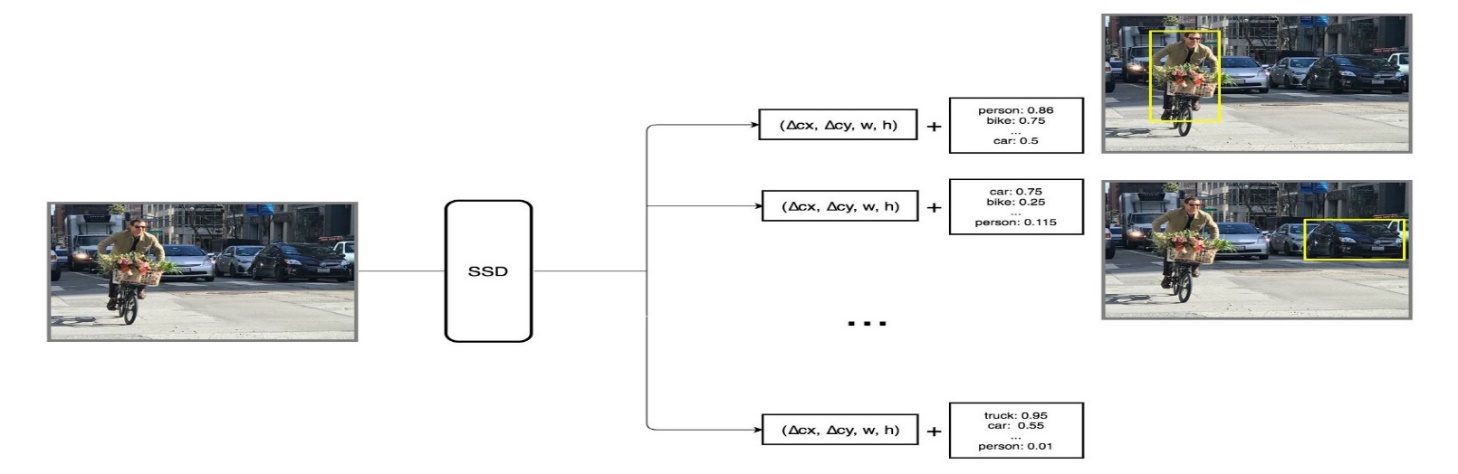


SSD uses VGG16 to extract feature maps. Then it detects objects using the Conv4\_3 layer. For illustration, we draw the Conv4\_3 to be 8 × 8 spatially (it should be 38 × 38). For each cell (also called location), it makes 4 object predictions.



**Left: the original image. Right: 4 predictions at each cell.**

Each prediction composes of a boundary box and 21 scores for each class (one extra class for no object), and we pick the highest score as the class for the bounded object. Conv4\_3 makes a total of 38 × 38 × 4 predictions: four predictions per cell regardless of the depth of the feature maps. As expected, many predictions contain no object. SSD reserves a class “0” to indicate it has no objects.



Each prediction includes a boundary box and 21 scores for 21 classes (one class for no object).

**Here are some key observations:**

* SSD performs worse than Faster R-CNN for small-scale objects. In SSD, small objects can only be detected in higher resolution layers (leftmost layers). But those layers contain low-level features, like edges or colour patches, that are less informative for classification.
* Accuracy increases with the number of default boundary boxes at the cost of speed.
* Multi-scale feature maps improve the detection of objects at different scale.
* Design better default boundary boxes will help accuracy.
* COCO dataset has smaller objects. To improve accuracy, use smaller default boxes (start with a smaller scale at 0.15).
* SSD has lower localization error comparing with R-CNN but more classification error dealing with similar categories. The higher classification errors are likely because we use the same boundary box to make multiple class predictions.
* SSD512 has better accuracy (2.5%) than SSD300 but run at 22 FPS instead of 59.

**SSD is a single-shot detector.** It has no delegated region proposal network and predicts the boundary boxes and the classes directly from feature maps in one single pass.

To improve accuracy, SSD introduces:

* small convolutional filters to predict object classes and offsets to default boundary boxes.
* separate filters for default boxes to handle the difference in aspect ratios.
* multi-scale feature maps for object detection.

SSD can be trained end-to-end for better accuracy. SSD makes more predictions and has a better coverage on location, scale and aspect ratios. With the improvements above, SSD can lower the input image resolution to 300 × 300 with a comparative accuracy performance. By removing the delegated region proposal and using lower resolution images, the model can run at real-time speed and still beats the accuracy of the state-of-the-art Faster R-CNN.

# Models Performance Improvement: -

Following methods will be used to improve the model performance.

* Tuning Hyper parameters of the model.
* Handel Overfitting and Underfitting problem.
* If required Image Data Augmentation will be used.

|  |  |  |  |
| --- | --- | --- | --- |
| Model Performed | Trained Set Count | Predicted Accuracy | Preferred Model |
| Mask R-CNN |  |  |  |
| Mobil Net |  |  |  |
| YOLO |  |  |  |
| Inception V3 – Google Net |  |  |  |
| SSD Model |  |  |  |

Final Report