Dhirubhai Ambani
Institute of Information and Communication Technology

Computer Vision Project Report

Object Detection and Localization using Faster R-CNN and Single Shot MultiBox Detector (SSD)

Name	Student Id
Rajat Sharma	201811045
Ruchi Chourasia	201811076

ABSTRACT

The task of object detection is to identify "what" objects are inside of an image and "where" they are. Given an input image, the algorithm outputs a list of objects, each associated with a class label and location (usually in the form of bounding box coordinates). In practice, only limited types of objects of interests are considered and the rest of the image should be recognized as object-less background.

This project aims to build a model that can detect and localize specific object(s) in an image. Pascal Visual Object Classes(VOC) dataset will be used for training, validation, and testing of the model. There are various methods for object detection like YOLO, SSD, Faster-RCNN, etc. We will be implementing two algorithms SSD and Faster-RCNN and compare the results on the basis of Intersection over Union (IoU) and Mean Average Precision (mAP) of these two algorithms on the same dataset.

APPROACH

VGG16 (Base Model)

- VGG16 (also called OxfordNet) is a convolutional neural network architecture named after the Visual Geometry Group from Oxford, who developed it. It was used to win the ILSVR (ImageNet) competition in 2014.

- In VGG16 instead of having a large number of hyper-parameter it focused on having convolution layers of 3x3 filter with a stride 1 and always used the same padding and max pool layer of 2x2 filter of stride 2.

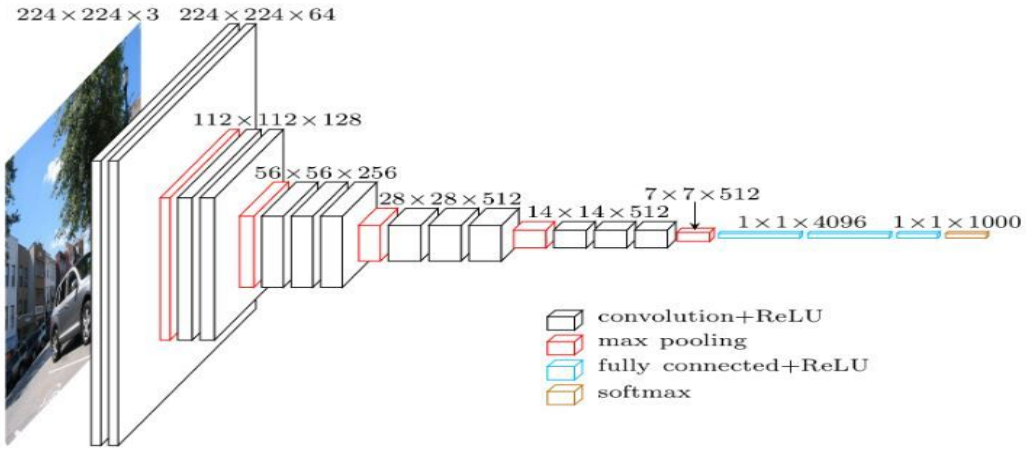


Fig.1 VGG16 Architecture

- It follows this arrangement of convolution and max pool layers consistently throughout the whole architecture. In the end, it has 2 FC(fully connected layers) followed by a softmax for output.
- The 16 in VGG16 refers to it has 16 layers that have weights. This network is a pretty large network and it has about 138 million (approx) parameters.
- It is used for the classification of an image as a whole.

Note: We have used VGG16 as our base model for the implementation of both of the models i.e. Faster R-CNN and SSD300

Faster R-CNN

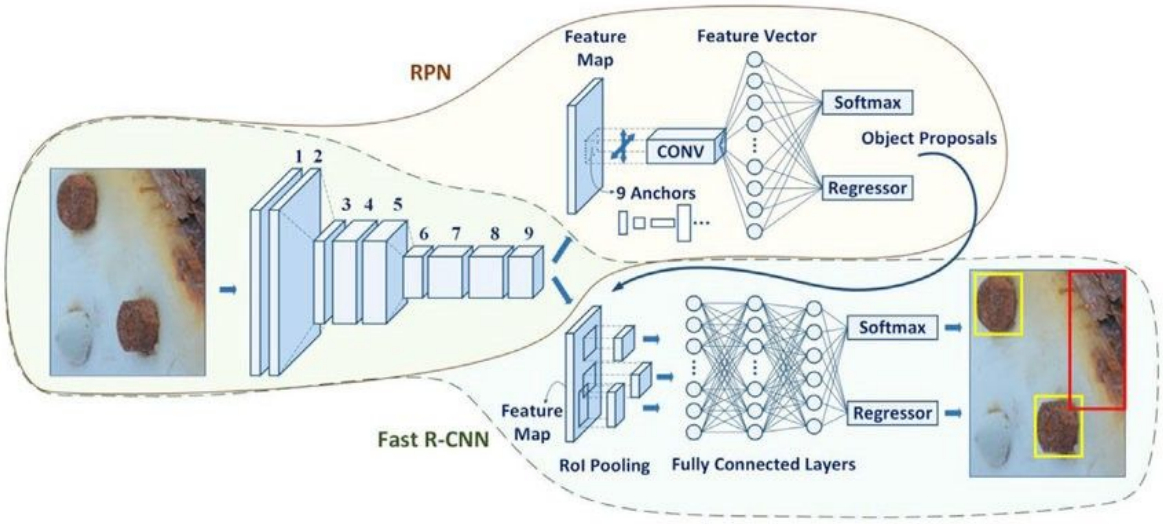


Fig.2 Architecture of Faster R-CNN

Faster R-CNN is composed of 3 parts -

1. **Convolution Layer** used to extract the appropriate features from the images. VGG-16 pre-trained model is used and output from conv5/conv5_1 layer is taken since we do not require the classification feature from the pre-trained model.
   In convolution, the initial layers are usually learn edges and after that more complex features are learned. The width and height of the feature map decrease because of the pooling applied between convolutional layers and the depth increases based on the number of filters the convolutional layer learns.

2. **Region Proposal Network**
   Anchors are fixed bounding boxes that are placed throughout the image with different sizes. An anchor is a box. In Faster R-CNN, there are 9 anchors at a position of an image with anchor scale [8,16,32] and anchor ratios [0.5,1,2].
   The RPN takes all the anchors and outputs a set of good proposals for objects. It does this by having two different outputs for each of the anchors. The first output is for checking that it is an object or a background. The second output is the bounding box regression for adjusting the anchors to better fit the object it's predicting.

   The **RPN is implemented** efficiently in a fully convolutional way, using the convolutional feature map returned by the base network as an input. First, we use a convolutional layer with 512 channels and 3x3 kernel size and then we have two parallel convolutional layers using a 1x11*x*1 kernel, whose number of channels depends on the number of anchors per point. For the classification layer, we output two predictions per anchor: the score of it being background (not an object) and the score of it being foreground (an actual object). For the regression, or bounding box adjustment layer, we output 4 predictions: the deltas Δxcenter,Δycenter,Δwidth,Δheight which we will apply to the anchors to get the final proposals.

   Non-Maximum Suppression (NMS) is used to solve the issue of duplicate proposals. It takes the list of proposals sorted by score and iterate over the sorted list, discarding those proposals that have an IoU larger than some predefined threshold (0.5) with a proposal that has a higher score. After applying NMS, we keep the top N proposals sorted by score.

3. **Detection Network**
   After the RPN, a bunch of object proposals with no class assigned to them is available. Now classification of bounding boxes into the desired categories is to be done.
   The simplest approach would be to take each proposal, crop it, and then pass it through the pre-trained base network. Then, we can use the extracted features as input for a vanilla image classifier. The main problem is that running the computations for all the 2000 proposals is really inefficient and slow.

Faster R-CNN tries to solve this problem by reusing the existing convolutional feature map. This is done by extracting fixed-sized feature maps for each proposal using region of interest pooling. Fixed size feature maps are needed for the R-CNN in order to classify them into a fixed number of classes.

Region-based convolutional neural network (R-CNN) is the final step in Faster R-CNN. Extracting features for each of the proposals (via RoI Pooling), need to use these features for classification. R-CNN tries to mimic the final stages of classification CNNs where a fully-connected layer is used to output a score for each possible object class.
R-CNN has two different goals:

- Classify proposals into one of the classes, plus a background class (for removing bad proposals).
- Better adjust the bounding box for the proposal according to the predicted class.

RPN Loss Function:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

Fig.3 RPN Loss Function

The first term is the classification loss over 2 classes (object or not). The second term is the regression loss of bounding boxes only when there is an object.
Thus, RPN network is to pre-check which location contains object. And the corresponding locations and bounding boxes will pass to detection network for detecting the object class and returning the bounding box of that object.

SSD-300

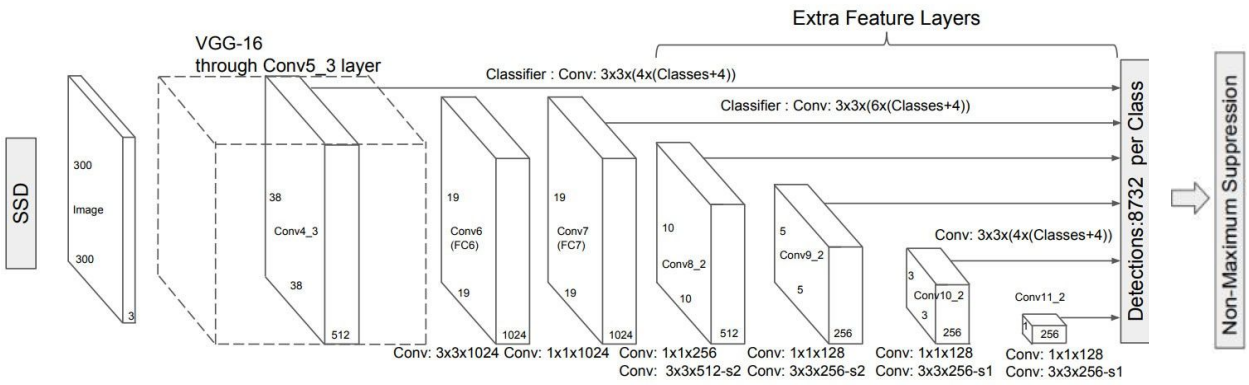


Fig.4 SSD300

Multibox

Multibox is an object detection technique where a prediction consists of two components.

- Coordinates of a box that may or may not contain an object (regression task).
- Scores for various object types for this box, including a background class(classification task).

Single Shot Detector (SSD)

The SSD is a purely convolutional neural network (CNN) that we can organize into three parts.

- **Base convolutions** derived from an existing image classification architecture it will provide lower-level feature maps.
- **Auxiliary convolutions** added on top of the base network that will provide higher-level feature maps.
- **Prediction convolutions** that will locate and identify objects in these feature maps.

Base Convolutions – part 1

Use the pretrained VGG-16 architecture as a base network, as this model has proven to work well with image classification and is already good at capturing the basic essence of an image. These convolutional features are also useful in object detection. There is an added advantage in using pertained layers on a reliable dataset as this is known as transfer learning, which means borrowing knowledge from a different but closely related task thus we have made progress even before starting.

Some changes are required to this pre-trained network for SSD 300 implementation for object detection and localization.

- The input image size should be 300, 300 for SSD300.
- In the 3rd pooling layer, use the mathematical ceiling function instead of the default floor function in determining output size as this will halves dimensions size. This is useful only if the dimensions of the previous feature map are odd and not even. For the input image size of 300, 300, the conv3_3 feature map will be of cross-section 75, 75, which is halved to 38, 38 instead of an inconvenient 37, 37.
- Modify the 5th pooling layer from a 2, 2 kernel and 2 strides to a 3, 3 kernel and 1 stride thus it no longer halves the dimensions of the feature map from the preceding convolutional layer.
- Remove the fully connected (i.e. classification) layers because they serve no purpose here and modify fc6 and fc7 into convolutional layers conv6 and conv7.

The first three modifications are straightforward enough, but that last one is because at this stage feature map is required, not a flatten feature for the classification tasks.

Base Convolutions – part 2

In the ImageNet VGG-16 shown previously, which operates on images of size 224, 224, 3, it can be seen that the output of conv5_3 will be of size 7, 7, 512. Therefore

- fc6 with a flattened input size of 7 * 7 * 512 and output size of 4096 has parameters of dimensions 4096, 7 * 7 * 512. The equivalent convolutional layer conv6 has a 7, 7 kernel size and 4096 output channels, with reshaped parameters of dimensions 4096, 7, 7, 512.

- fc7 with an input size of 4096 (i.e. the output size of fc6) and an output size 4096 has parameters of dimensions 4096, 4096. The input could be considered as a 1, 1 image with 4096 input channels. The equivalent convolutional layer conv7 has a 1, 1 kernel size and 4096 output channels, with reshaped parameters of dimensions 4096, 1, 1, 4096.

It can be seen that conv6 has 4096 filters, each with dimensions 7, 7, 512, and conv7 has 4096 filters, each with dimensions 1, 1, 4096. These filters are large and computationally very expensive.

To solve this problem, reduce the number and the size of each filter by subsampling parameter from the converted convolution layers.

- conv6 will use 1024 filters, each with dimensions 3, 3, 512. Therefore, the parameters are subsampled from 4096, 7, 7, 512 to 1024, 3, 3, 512.

- conv7 will use 1024 filters, each with dimensions 1, 1, 1024. Therefore, the parameters are subsampled from 4096, 1, 1, 4096 to 1024, 1, 1, 1024.

Now, subsample the kernel by picking every mth parameter along a particular dimension, this process is known as decimation. After the decimation of the kernel, there are now holes in the kernel, therefore, make the kernel dilated. Dilation of 6 is used as 5th pooling layer now no longer halves the dimension of preceding feature map

Thus VGG16 is now modified.

Auxiliary Convolutions

Now add some more convolutional layers on top of our base network. These convolutions provide feature maps, each smaller than the previous one.

Introduce four convolutional blocks, each with two layers in which size reduction is caused by a stride of 2 in every second layer.

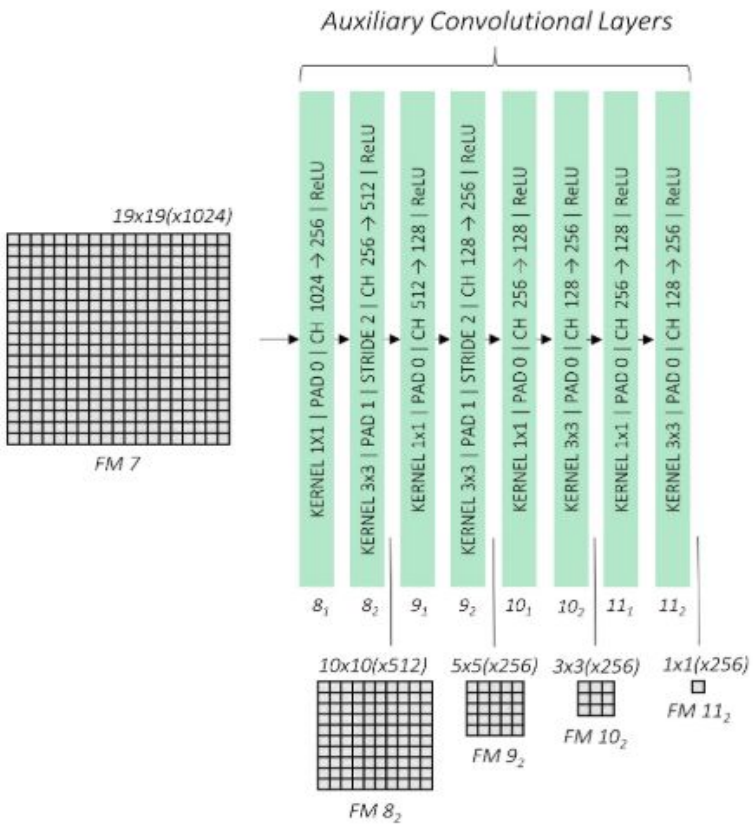


Fig: Auxiliary Convolution

Fig reference:
https://www.google.com/search?q=Auxiliary+convolution+of+SSD300&rlz=1C2RLNS_enIN736IN754&sxsrf=ACYBGNTfchG2hFqyhXSj4dhP9-4pILs_8A:1575270215524&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiOivzsspbmAhU8ILcAHXo9Dl0Q_AUoAXoECA8QAw&biw=1242&bih=553

Priors

Object prediction can be quite diverse as an object can occur at any position inside the image with any shape and size. So there can be infinite possibilities of occurrence of an object within an image, mathematically it is true but practically impossible to do so stick to just thousands of possibilities for convenience.

Priors are precalculated, fixed boxes that collectively represent this universe of probable and approximate box predictions. Priors are manually but carefully chosen based on the shapes and sizes of ground truth objects in our dataset. By placing these priors at every possible location in a feature map, we also account for variations in position.

Specify priors as:

- Priors are applied to various low-level and high-level feature maps i.e conv4_3, conv7, conv8_2, conv9_2, conv10_2, and conv11_2
- if a prior has a scale s, then its area is a square with side s. The largest feature map, conv4_3, will have priors with a scale of 0.1, i.e. 10% of the image's dimensions, while the rest have priors with scales linearly increasing from 0.2 to 0.9. Larger feature maps have priors with smaller scales thus it is ideal for detecting smaller objects.
- At each location on a feature map, there are priors of many different aspect ratios. All feature maps will have priors with ratios 1:1, 2:1, 1:2. The intermediate feature maps of conv7, conv8_2, and conv9_2 will also have priors with ratios 3:1, 1:3. All feature maps will have one extra prior with an aspect ratio of 1:1 and at a scale that is the geometric mean of the scales of the current and subsequent feature map.

The table below shows the number of prior calculated at each of these feature maps.

Feature Map From	**Feature Map Dimensions**	**Prior Scale**	**Aspect Ratios**	**Number of Priors per Position**	**Total Number of Priors on this Feature Map**
conv4_3	38, 38	0.1	1:1, 2:1, 1:2 + an extra prior	4	5776
conv7	19, 19	0.2	1:1, 2:1, 1:2, 3:1, 1:3 + an extra prior	6	2166
conv8_2	10, 10	0.375	1:1, 2:1, 1:2, 3:1, 1:3 + an extra prior	6	600
conv9_2	5, 5	0.55	1:1, 2:1, 1:2, 3:1, 1:3 + an extra prior	6	150
conv10_2	3, 3	0.725	1:1, 2:1, 1:2 + an extra prior	4	36
conv11_2	1, 1	0.9	1:1, 2:1, 1:2 + an extra prior	4	4
Grand Total	–	–	–	–	8732 priors

There are a total of 8732 priors defined for the SSD300.

Predictions via Priors

Earlier, It was said that use regression to find the coordinates of an object's bounding box. As the priors can't represent our final predicted boxes. Prior only represent possibilities for the prediction that means that use each prior as an approximate starting point then find out how much it needs to be adjusted to obtain a more exact prediction for a bounding box.

So if each predicted bounding box has a little deviation from a prior, and our goal is to calculate this deviation, we need a way to measure or quantify it.

Prediction convolutions

Six feature maps of various scales and granularity are generated, i.e those from conv4_3, conv7, conv8_2, conv9_2, conv10_2, and conv11_2.

Then, for each prior at each location on each feature map, predict:

- The offsets (g_c_x, g_c_y, g_w, g_h) for a bounding box.
- A set of n_classes scores for the bounding box, where n_classes represents the total number of object types (including a background class).

To accomplish this task in the simplest manner possible, two convolutional layers are required for each feature map

- **A localization prediction convolutional layer** with a 3,3 kernel operating at each location (i.e. with padding and stride of 1) with 4 filters for each prior present at the location. The 4 filters for a prior calculate the four encoded offsets (g_c_x, g_c_y, g_w, g_h) for the bounding box predicted from that prior.
- **A class prediction convolutional layer** with a 3, 3 kernel operating at each location (i.e. with padding and stride of 1) with n_classes filters for each prior present at the location. The n_classes filters for a prior calculate a set of n_classes scores for that prior.

Matching predictions to ground truths

The model learns anything by comparisons between what got predicted and the objects actually present in the image. Here prior helps in the same manner.

- Find the IOU(Jaccard overlaps) between the 8732 priors and N ground truth objects. This will be a tensor of size 8732, N.
- Match each of the 8732 priors to the object with which it has the greatest overlap.

- If a prior is matched with an object with a Jaccard overlap of less than 0.5, then it cannot be said to "contain" the object and is, therefore, a negative match. Considering we have thousands of priors, most priors test negative for an object.
- On the other hand, a small number of priors will actually overlap significantly (greater than 0.5) with an object thus containing objects. These are positive matches.
- Now that we have matched each of the 8732 priors to ground truth, we have, in effect, also matched the corresponding 8732 predictions to the ground truth.

Positively matched prediction with an object now have ground-truth coordinates that will be used as targets for localization, i.e. in the regression task.

All predictions have a ground truth label, which is either the type of object if a positive match or a background class if a negative match. These are used as targets for class prediction, i.e. the classification task.

Processing predictions

After training the model, it can be applied over images. However, the predictions are still in their raw form two tensors containing the offsets and class scores for 8732 priors. These need to be processed to get final bounding boxes with labels. The algorithm is as follows.

Algorithm:

On selecting candidates for each non-background class,

- Arrange candidates for this class in decreasing order of likelihood.
- Eliminate all candidates with Jaccard scores lesser than 0.5.
- Consider the next highest-scoring candidate still remaining in the pool. Eliminate all candidates with lesser scores that have a Jaccard overlap of more than 0.5 with this candidate.
- Repeat until you run through the entire sequence of candidates.

The end result is just a single box the best one for each object in the image.

DATASET

Pascal Visual Object Classes(VOC) 2007
The dataset contains a total of 9,963 annotated images. The data set is divided into 2 equal parts:

- **trainval**: The union of the **train** and **val** dataset.
- **test**: Test data.

There are 20 different classes of objects in the dataset. Each image contains at least one object of a given class.

The dataset has been split into 50:50 part for training/validation and testing. Also, the distribution of objects by class is approximately equal across the training/validation and testing data.

So, in total there are 9,963 images, containing 24,640 annotated objects.

For each object the following annotation is present:

EVALUATION PARAMETERS

Intersection over Union (IoU)

The ratio between the intersected area over the joined area for two bounding boxes is called as IoU.

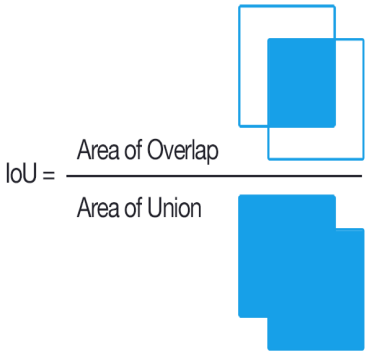


Fig. IoU Formula

Mean Average Precision (mAP)

Precision **P = TP / (TP + FP)**
Recall **R = TP / (TP + FN)**

Where,

TP: are the Bounding Boxes that have IoU with the ground truth is above 0.5
FP: Bounding boxes that the IoU with ground truth is below 0.5 also the BB that have IoU with a GT that has already been detected.
TN: there are not true negative, the image is expected to contain at least one object
FN: those images were the method failed to produce a Bounding boxes

Average Precision The general definition for the **Average Precision (AP) is finding the area under the precision-recall curve.**

$$\mathrm{AP} = \int_0^1 p(r)dr$$

Fig. AP Formula
AP for each class is calculated.

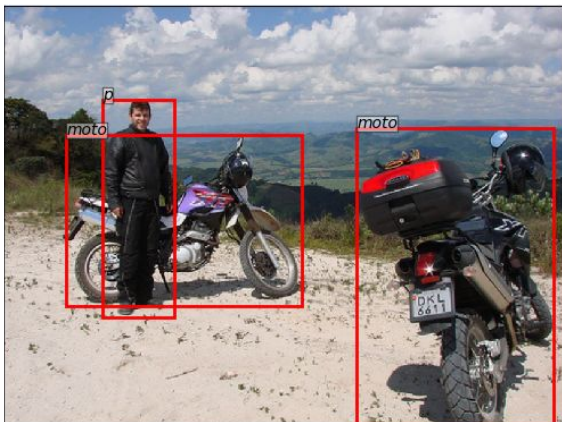
Mean Average Precision (mAP) is the average of AP of each class.

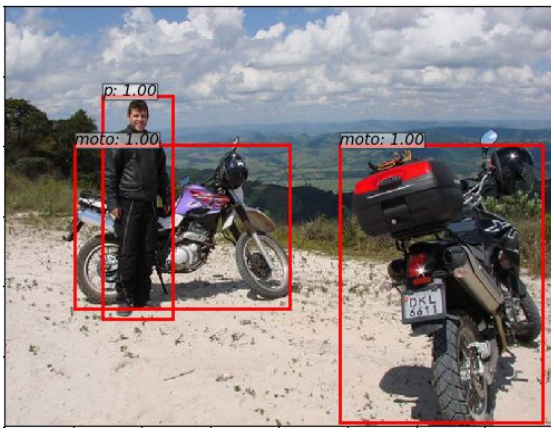
$$\mathrm{mAP} = \frac{1}{N}\sum_{i=1}^{N}\mathrm{AP}_i$$

Fig. mAP Formula

RESULTS

Results of Faster R-CNN





Ground Truth

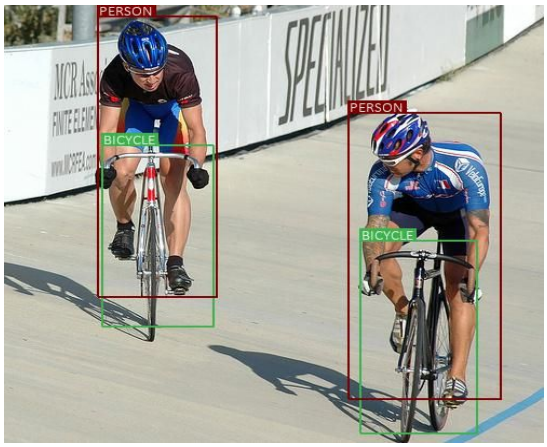
Result obtained

Ground Truth

Result obtained

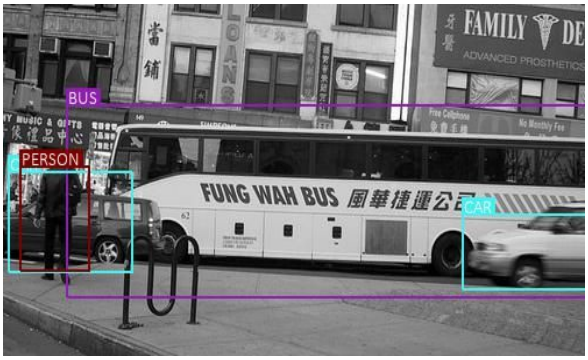
Results of SSD-300

Ground Truth



Result obtained

Ground Truth



Result obtained

Mean Average Precision Value

	Faster R-CNN	SSD-300
mAP	0.702	0.685

CONCLUSION

- SSD takes lesser time to train than Faster R-CNN.
- mAP value of Faster R-CNN is more than SSD.
- SSD has more classification error, since it uses the same bounding box to make multiple class prediction.

CODE GITHUB LINK

https://github.com/RajatSharma160996/Object-Detection-and-Localization-in-image

REFERENCES

[1] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu and Alexander C. Berg "SSD: Single Shot MultiBox Detector", *ECCV* 2016.

[2] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, VOL. 39, NO. 6, JUNE 2017