MINOR PROJECT EVALUATION - Phase 1

Implementation of Object Detection Algorithm-SSD

Faculty: Dr. Jaidhar Sir

Semester: 6th sem B.Tech

Department of Information Technology,

National Institute of Technology Karnataka, Surathkal

Mangalore, India.

Submitted by:

Vaishnavi Mishra(16IT145)
Sony Bachina(16IT208)
Vidhi Kothari (16IT223)

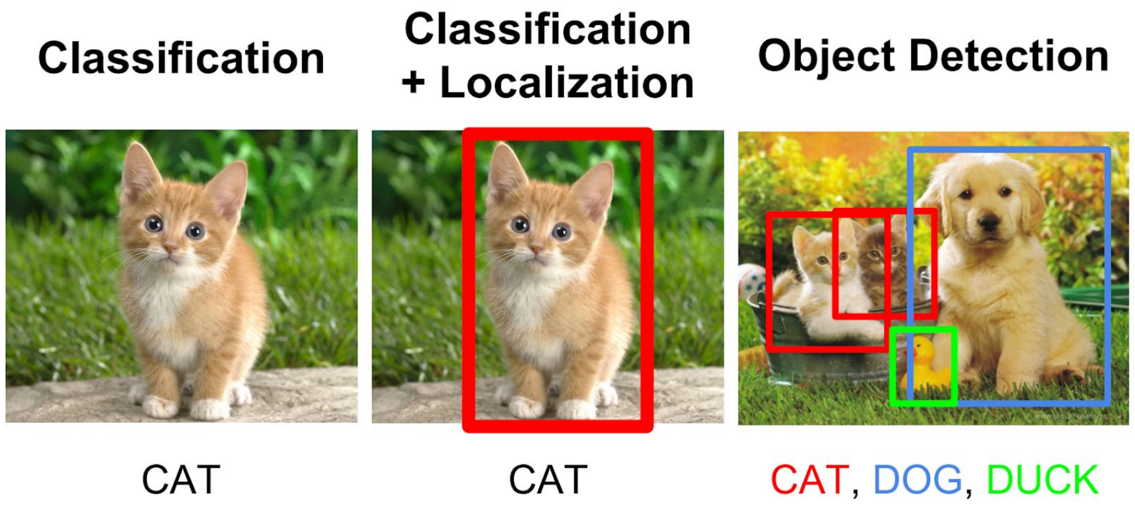
Abstract

Efficient and accurate object detection has been an important topic in the advancement of computer vision systems. With the advent of deep learning techniques, the accuracy for object detection has increased drastically. The project aims to incorporate a state-of-the-art technique for object detection with the goal of achieving high accuracy with real-time performance. A major challenge in many of the object detection systems is the dependency on other computer vision techniques for helping the deep learning based approach, which leads to slow and non-optimal performance. In this project, we use a completely deep learning based approach to solve the problem of object detection in an end-to-end fashion. Our network is trained on the dataset (PASCAL VOC 2012), on which an object detection challenge is conducted annually. The resulting system is fast and accurate, thus aiding those applications which require object detection.

Introduction

The area chosen for research was that of object detection, due to the varied and widespread application of those techniques to aid in the resolution of various problems of our day.

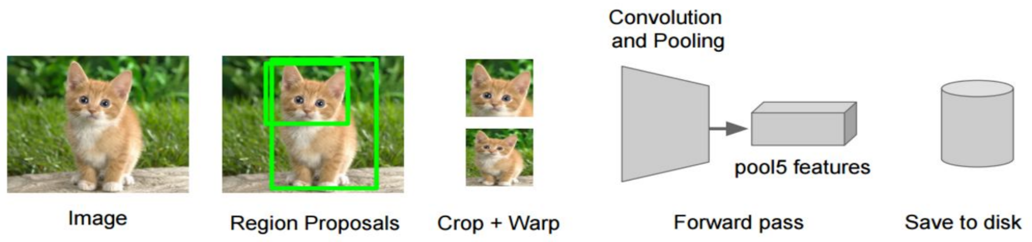
However, this important task does not come without it’s set of challenges. Object detection involves both classification and localization, i.e predicting the class of the image and bounding it. In this case, the input to the system will be image, and the output will be a bounding box corresponding to all the objects in the image, along with the class of object in each box.

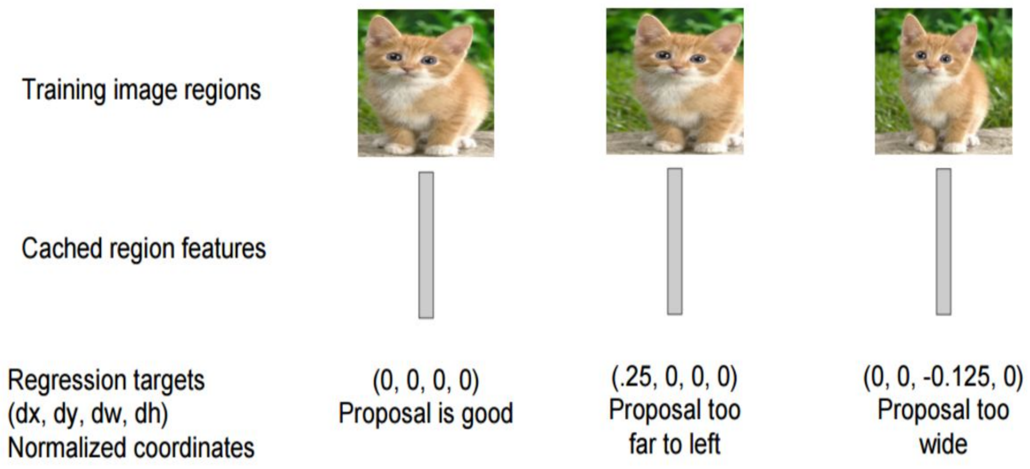


Literature Survey

There has been a lot of work in object detection using traditional computer vision techniques (sliding windows, deformable part models). However, they lack the accuracy of deep learning based techniques. Among the deep learning based techniques, two broad class of methods are prevalent: two-stage detection (R-CNN, Fast R-CNN, Faster R-CNN) and unified detection (Yolo, SSD). The major concepts involved in these techniques have been explained below:

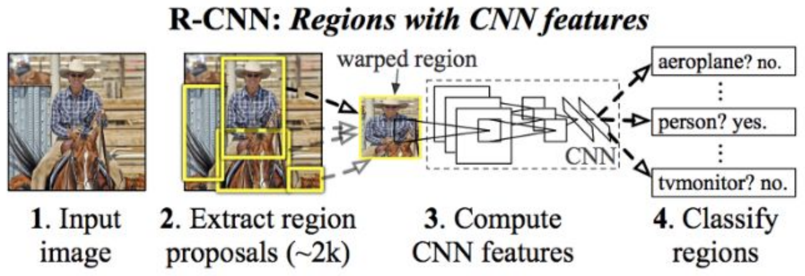
1) **Two-stage detection**: In this case, the proposals are extracted using some other computer vision technique and then resized to fixed input for the classification network, which acts as a feature extractor. Then an SVM is trained to classify between object and background (one SVM for each class). Also, a bounding box regressor is trained that outputs some correction (offsets) for proposal boxes. These methods are very accurate but are computationally expensive.




Training image regions
Cached region features
Regression targets (dx, dy, dw, dh) Normalized coordinates
(0, 0, 0, 0) Proposal is good
(.25, 0, 0, 0) Proposal too far to left
(0, 0, -0.125, 0) Proposal too wide

A brief overview of algorithms:

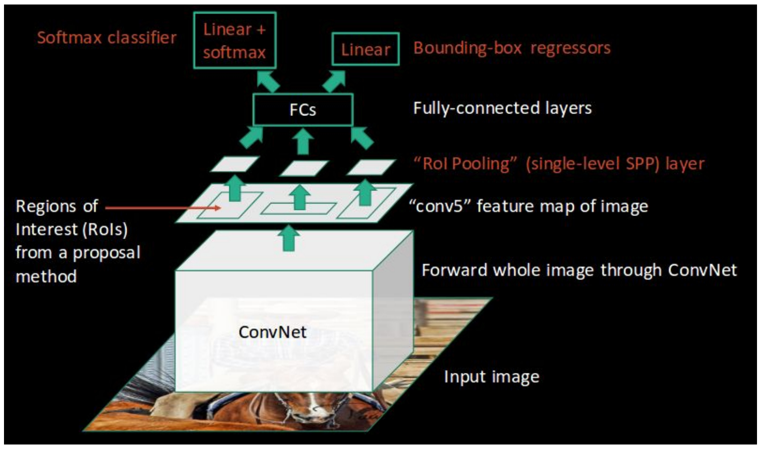
1. **R-CNN**: To overcome the problem of selecting a huge number of regions, Ross Girshick, proposed a method where we use selective search to extract just 2000 regions from the image and he called them region proposals. Therefore, now, instead of trying to classify a huge number of regions, you can just work with 2000 regions. These 2000 region proposals are generated using the selective search algorithm which is written below.



R-CNN

Problems of R-CNN:
- It still takes a huge amount of time to train the network as you would have to classify 2000 region proposals per image.
- It cannot be implemented in real time as it takes around 47(40-50) seconds for each test image.
- The selective search algorithm is a fixed algorithm. Therefore, no learning is happening at that stage. This could lead to the generation of bad candidate region proposals.

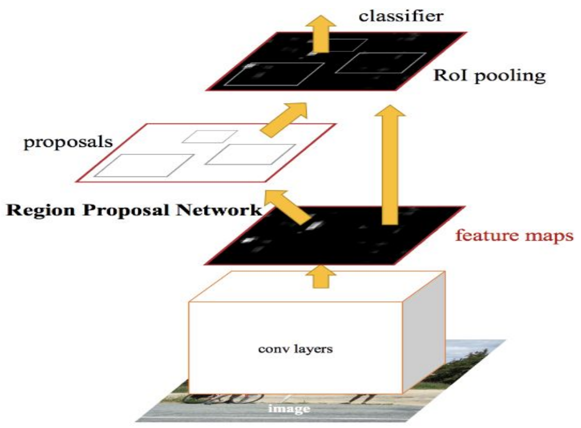
2. **Fast R-CNN:** The approach is similar to the R-CNN algorithm. But, instead of feeding the region proposals to the CNN, we feed the input image to the CNN to generate a convolutional feature map. From the convolutional feature map, we identify the region of proposals and warp them into squares and by using an RoI pooling layer we reshape them into a fixed size so that it can be fed into a fully connected layer. From the RoI feature vector, we use a softmax layer to predict the class of the proposed region and also the offset values for the bounding box. The convolution operation is done only once per image and a feature map is generated from it.


Softmax classifier
Linear + softmax
Linear
Bounding-box regressors
FCs
Fully-connected layers
"RoI Pooling" (single-level SPP) layer
Regions of Interest (RoIs) from a proposal method
"conv5" feature map of image
Forward whole image through ConvNet
ConvNet
Input image

Problem with Fast R-CNN:

- Fast R-CNN during testing time, including region proposals(again selective search), slows down the algorithm significantly when compared to not using region proposals. It takes 2sec to classify each image.

3. **Faster R-CNN:** Faster R-CNN is the modified version of Fast R-CNN. The major difference between them is that Fast R-CNN uses selective search for generating Regions of Interest, while Faster R-CNN uses "Region Proposal Network", aka RPN. RPN takes image feature maps as an input and generates a set of object proposals, each with an objectness score as output.



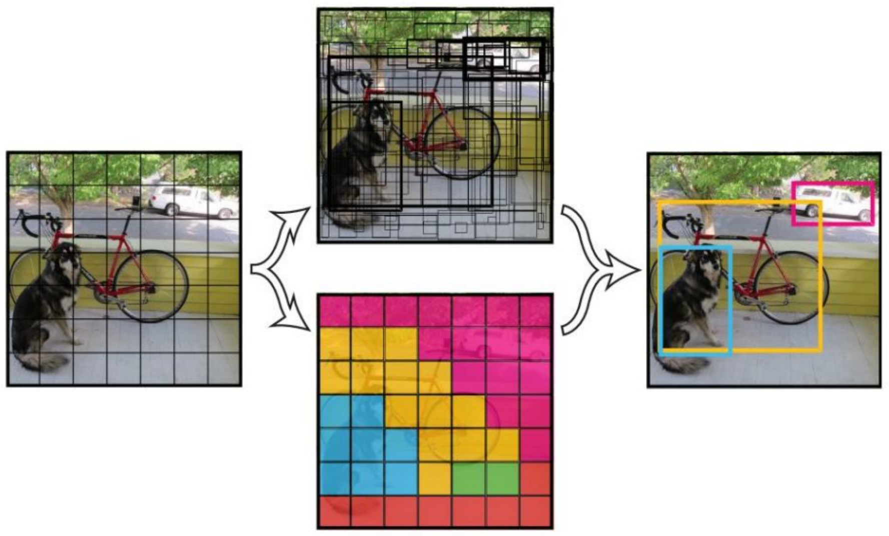
Problems with Faster R-CNN:

- The algorithm requires many passes through a single image to extract all the objects

- As there are different systems working one after the other, the performance of the systems further ahead depends on how the previous systems performed

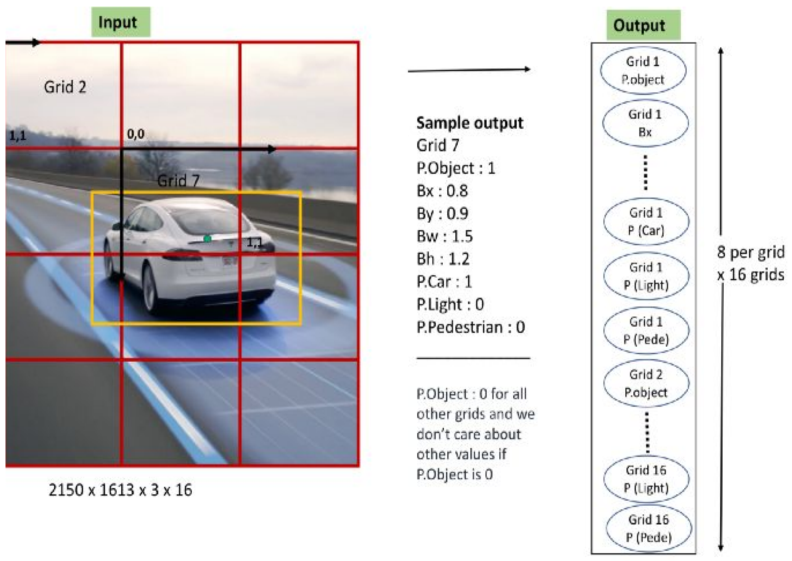
2) **Unified Process:** The difference here is that instead of producing proposals, pre-define a set of boxes to look for objects. Using convolutional feature maps from later layers of the network, run another network over these feature maps to predict class scores and bounding box offsets. The steps are mentioned below:

   1. Train a CNN with regression and classification objective.
   2. Gather activation from later layers to infer classification and location with a fully connected or convolutional layer.
   3. During training, use Jaccard distance to relate predictions with the ground truth.
   4. During inference, use non-maxima suppression to filter multiple boxes around the same object.



A brief overview of algorithms:

1. **YOLO(You Only Look Once):** The idea is to divide the image into multiple grids. Then we change the label of our data such that we implement both localization and classification algorithm for each grid cell.



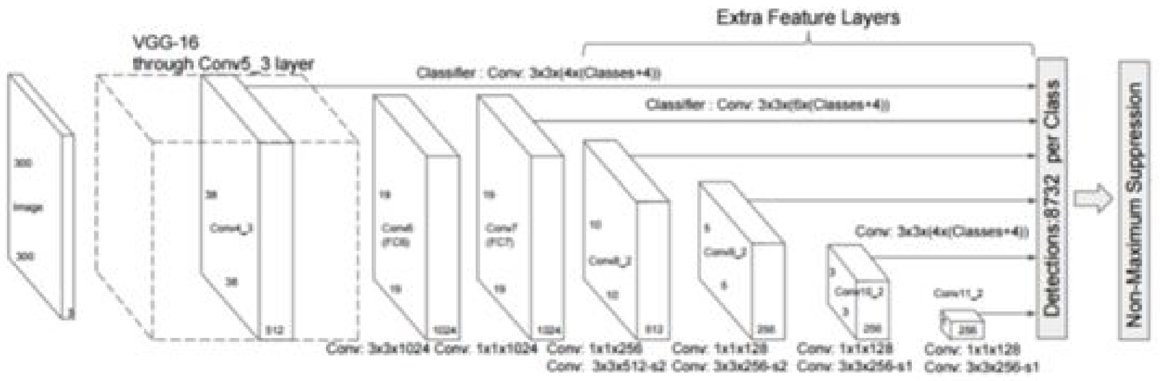
It is much more accurate and faster than the sliding window algorithm. As of today, there are multiple versions of pre-trained YOLO models available. The latest YOLO paper is: “YOLO9000: Better, Faster, Stronger”. The model is trained on 9000 classes.

Applications

a. Face detection uses object detection.
b. It has an important role to play in surveillance systems.
c. Cancer detection in medicine.
d. Used for people counting, it is used for analysing store performance or crowd statistics during festivals.
e. Apart from these object detection can be used for classifying images found online. Obscene images are usually filtered out using object detection.
f. Object detection is also used in industrial processes to identify products.

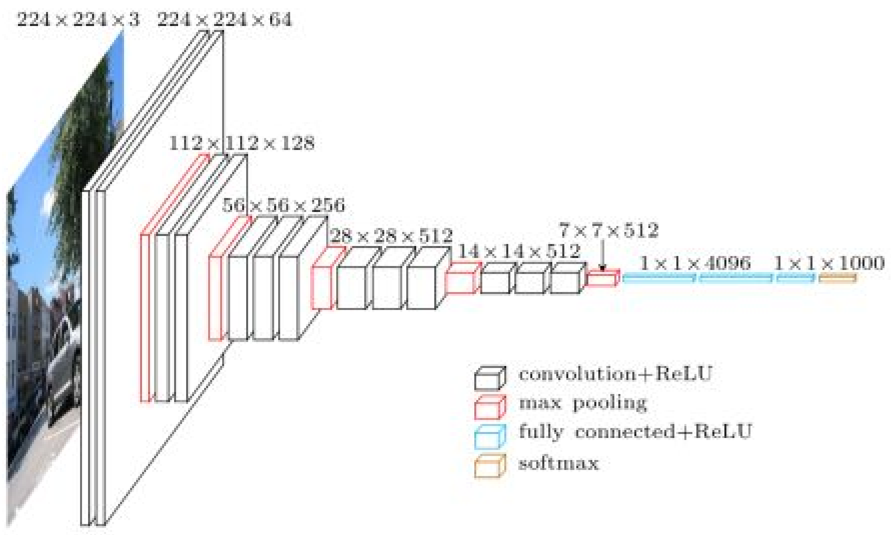
Research Methodology

The network used in this project is based on Single shot detection (SSD). The architecture is shown in Fig.



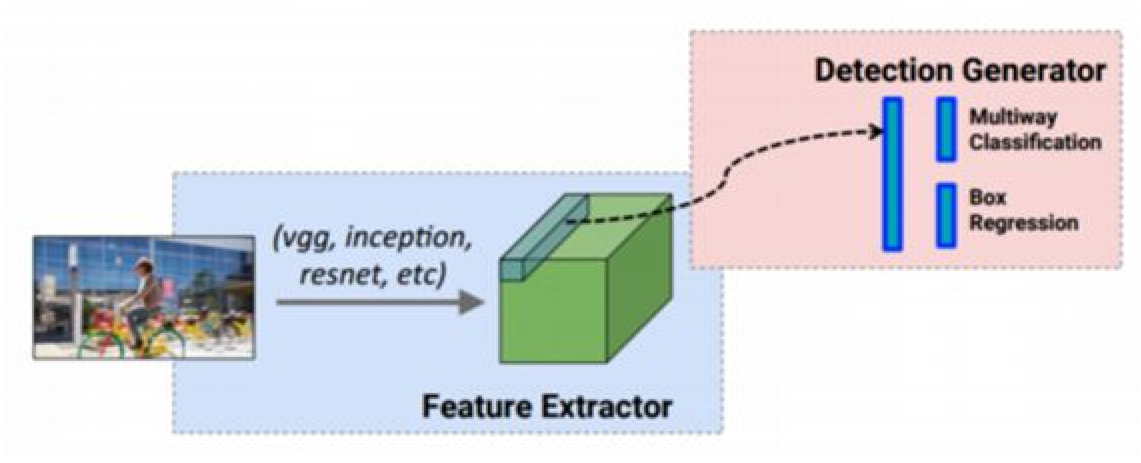
By using SSD, we only need to take one single shot to detect multiple objects within the image, while approaches such as R-CNN series need two shots, one for generating region proposals, one for detecting the object of each proposal. Thus, SSD is much faster compared with two-shot RPN-based approaches. SSD300 achieves 74.3% mAP at 59 FPS while SSD500 achieves 76.9% mAP at 22 FPS, which outperforms Faster R-CNN (73.2% mAP at 7 FPS) and YOLOv1 (63.4% mAP at 45 FPS).

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 to extract features at multiple scales and progressively decrease the size of the input to each subsequent layer.



To have more accurate detection, different layers of feature maps are also going through a small 3×3 convolution for object detection as shown in the SSD architecture diagram.

The SSD normally starts with a VGG model, which is converted to a fully convolutional network. Then we attach some extra convolutional layers, that help to handle bigger objects. The output at the VGG network is a 38x38 feature map (conv4_3). The added layers produce 19x19, 10x10, 5x5, 3x3, 1x1 feature maps. All these feature maps are used for predicting bounding boxes at various scales (later layers responsible for larger objects). Thus the overall idea of SSD is shown in Fig. Some of the activations are passed to the sub-network that acts as a classifier and a localizer.



When you use a neural network like SDD to predict multiple objects in a picture, the network is actually making thousands of predictions and only showing the ones that it decided were an object. The multiple predictions are output with the following format:

Prediction 1: (X, Y, Height, Width), Class

….

Prediction ~80,000: (X, Y, Height, Width), Class

Where the(X, Y, Height, Width) is called the “bounding box”, or box surrounding the objects. This box and the object class are labelled manually by human annotators. These bounding boxes are also known as anchors, as they create an anchor or ideal standard for what the object that is to be detected must look like.

Steps to create an anchor:

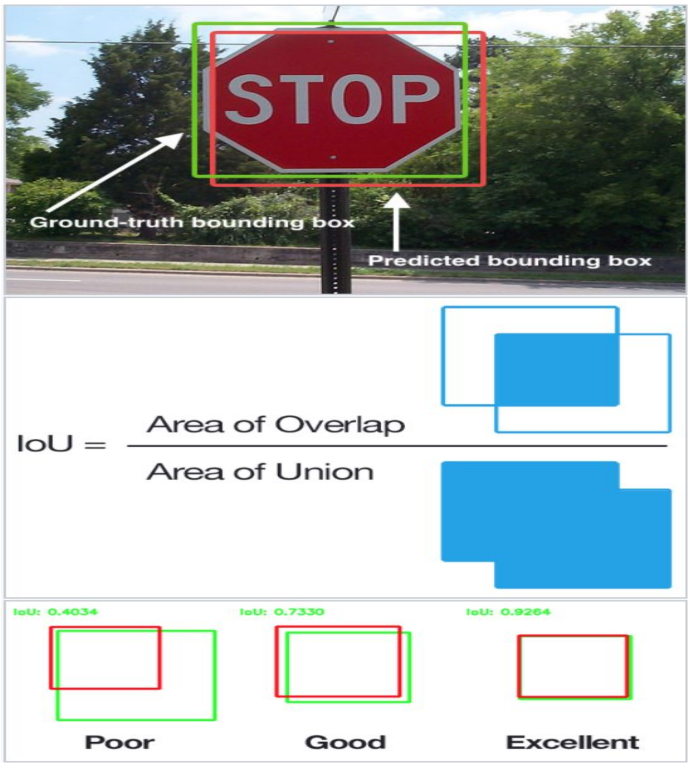
1. Create thousands of “anchor boxes” or “prior boxes” for each predictor that represent the ideal location, shape, and size of the object it specializes in predicting.

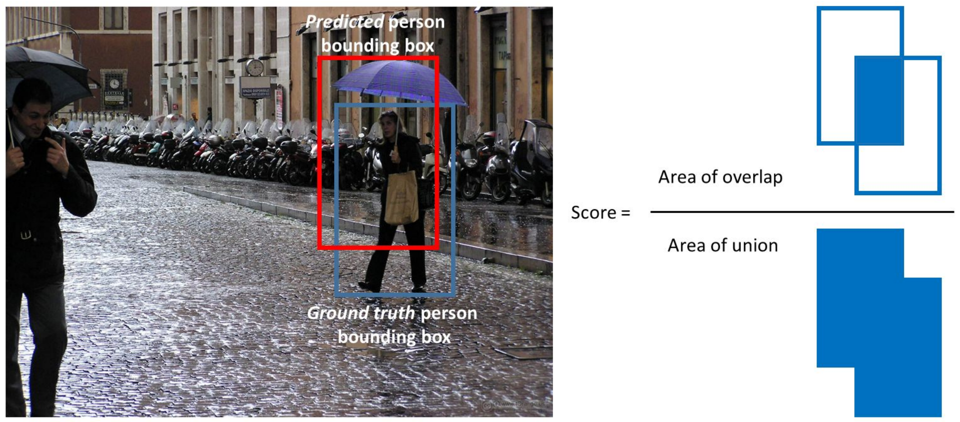
2. For each anchor box, calculate which object’s bounding box has the highest overlap divided by non-overlap. This is called Intersection Over Union or IOU.

3. If the highest IOU is greater than 50%, tell the anchor box that it should detect the object that gave the highest IOU.

4. Otherwise, if the IOU is greater than 40%, tell the neural network that the true detection is ambiguous and not to learn from that example.

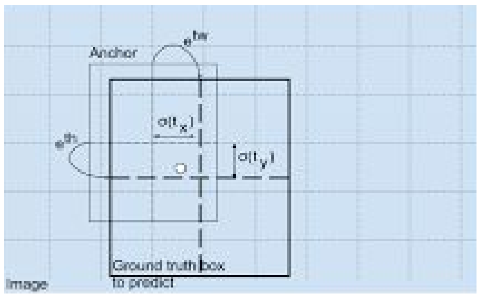
5. If the highest IOU is less than 40%, then the anchor box should predict that there is no object.




Predicted person bounding box
Ground truth person bounding box
Score =
Area of overlap
Area of union

An IoU of 0.5 is still not good enough but it does however provide a strong starting point for the bounding box regression algorithm — it is a much better strategy than starting the predictions with random coordinates! Therefore SSD starts with the priors as predictions and attempt to regress closer to the ground truth bounding boxes.
Anchors act as reference points on ground truth images as shown in Fig.. A model is trained to make two predictions for each anchor:

• A discrete class
• A continuous offset by which the anchor needs to be shifted to fit the ground-truth bounding box.



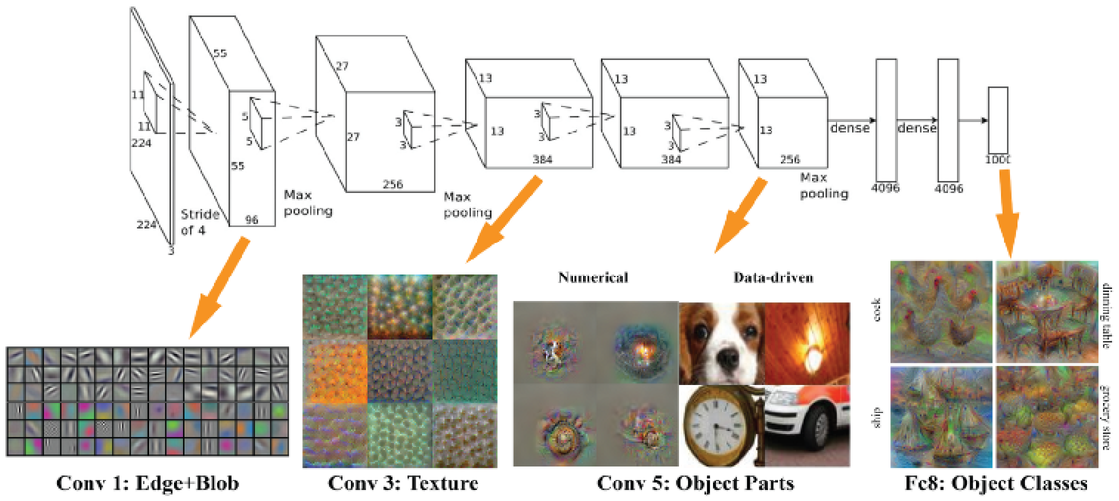
Predicted person bounding box
Ground truth person bounding box

Default Bounding Boxes

It is recommended to configure a varied set of default bounding boxes, of different scales and aspect ratios to ensure most objects could be captured. We have adopted this method.

Feature Maps

Features maps (i.e. the results of the convolutional blocks) are a representation of the dominant features of the image at different scales, therefore running SSD on multiple feature maps increases the likelihood of any object (large and small) to be eventually detected, localized and appropriately classified. This method too has been adopted. The image below shows how the network "sees" a given image across its feature maps:


224
224
3
11
11
Stride of 4
55
55
96
5
5
Max pooling
27
27
256
3
3
Max pooling
13
13
384
13
13
384
13
13
256
Max pooling
dense
dense
4096
4096
1000
Numerical
Data-driven
cock
dinning table
ship
grocery store
Conv 1: Edge+Blob
Conv 3: Texture
Conv 5: Object Parts
Fc8: Object Classes