Object detection: Comparison of VGG16 and SSD

By Sabhatina Selvam

ECE 539 Project

Abstract

Image classification is a standard problem of classifying images into a category. Labelling objects in the image in this case, consists of a feature extraction stage followed by a classification model. The advent of Convolution Neural Networks (CNNs) has paved way for high-level feature extraction from raw images with astounding results in object classification. VGG16 architecture extrapolated the idea of greater depth in layers in the ILSVRC 2014. The model stood first for the object localization task and second for the object classification task. A slightly different objective pertaining to object detection combines the classification task with the localization task, thus laying the foundation for identifying multiple relevant objects in a single image. Unlike classification models, these are more accurate and have specific applications. Some of the real time applications call for high computational speed achieved by some single shot methods of object detection. Single Shot Multi-box detector is a state-of-the-art multiple-object detector that operates with a single feed forward pass and parallelly predicts bounding boxes as well as classification scores. In this project, I have discussed two state-of-the-art object detectors and compared their performances. I have attempted at matching my results with the state-of-the-art claims. My VGG16 model has regression layers for predicting bounding boxes after feature extraction and SSD has a single feed-forward network that parallelly predicts bounding boxes and confidence scores in different scales per feature map location. These prediction networks have been trained on PASCAL VOC dataset for VGG16, and PASCAL VOC plus COCO dataset for SSD. I have been able to achieve close resemblance to the state-of-the-art results as discussed in the last section.

Motivation

Comparing different object detection algorithms is difficult as the parameters under consideration can differ for different kind of applications. I have compared some of the primary parameters that differentiate models, for better understanding. In real-life applications, we make choices to balance speed and accuracy. The intricacies behind the choice of a detector lie in the following less explored parameters:

- Feature extractor
- Input image resolution
- IOU threshold
- Number of predictions
- Choice of localization loss

- Training dataset and configuration
- Deep learning software platform used

I have compiled results for these parameters in this project for firstly, a dense network performing regression with VGG16 base and secondly, a Single Shot multi-box Detector (SSD).

Introduction

Object detection in one of the fundamental problems in the field of artificial intelligence with applications in robotics, automation, and human-computer interaction. The aim is to track an arbitrary object in consecutive frames of a video segment by localizing it inside bounding boxes. The most common representation of these bounding boxes is in terms of the top-left and bottom-right coordinates in the frame with respect to the origin of each imaginary image grid.

Existing state-of-the-art object detectors follow similar hypothesis: construct ground truth boxes, extract features or resample pixels, and follow it up by a robust classifier. The current leading results on PASCAL VOC, COCO, and ILSVRC detection are all based on the same principle albeit with deeper feature extractors such as Resnets, Inceptionv3. The VGG16 model secured the first position in ILSRVC for object localization and its accuracy for predicting the location of these boxes is unquestionably high [1]. Nevertheless, trade-offs between accuracy and computation-intensity is obvious and raises the need for faster approaches. In this project, the VGG16 model has been trained on pre-trained weights on ImageNet for feature extraction. This transfer learning model is advantageous as it escapes the necessity to train the model on a large-scale dataset like ImageNet. Even though the weights have been tuned for classification task predominantly, the high-level features are useful for bounding box predictions also. It has a major limitation of being very slow for real-time applications. Often detection speed for these approaches is measured in seconds per frame (SPF). The VGG16 model can be summarized as below:

- The lower layers of VGG16 are trained on pre-trained weights to extract high-level features for fixed size input images.
- Extracted features are fed to the dense network with output as scaled bounding box coordinates. The ground truth boxes are scaled by the aspect ratio of each individual image.
- No region proposals or ROI-pooling layers involved in the training. The network learns to localize any arbitrary object from the ground truth box coordinates alone.

There are various network implementations for faster trackers by tweaking each stage of the detection pipeline. But the increase in speed comes at the cost of the deprecating accuracy. SSD overcomes this limitation as it does not resample pixels or features for ground truth bounding boxes and is as accurate as the approaches that do. SSD discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location[2]. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. Additionally, the network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes. I have implemented the state-of-the-art SSD300 object detector. The SSD network can be summarized as below:

- SSD is faster and more accurate than the previous state-of-the-art single shot detectors. It balances the speed vs accuracy trade-offs that opens a wide variety of applications in computer vision.
- A small set of convolutional filters predict class scores and box offsets for fixed-size bounding boxes from the feature maps. No explicit prediction of bounding boxes like R-CNNs is involved.
- These work well with low input resolution which bolsters the speed vs accuracy trade-off further.
- In this project, I have evaluated on PASCAL VOC and COCO dataset with varying input sizes and compared with the recent state-of-the-art approaches.

Model

1. VGG16 Architecture

The input to the Convolutional Network is a fixed-size 224 X 224 X 3 image. The pre-processing step subtracts the mean RGB value from each pixel. The image is passed through a stack of convolutional layers with 3 X 3 receptive fields (smallest size that accommodates a pixel shift). In one of the layers, a 1 X 1 convolutional filter linearly transforms the input channels. The stride is fixed to 1 pixel and padding is such that the spatial resolution is preserved after convolution. 5 max-pooling layers are performed over a 2 X 2 pixel window, with stride 2. I have used the pre-trained weights for VGG16 trained model on ImageNet dataset to extract features. The feature for each image is a tensor of 7 X 7 X 512 dimension. Fig 1 represents the architecture of the convolutional layers in VGG16.

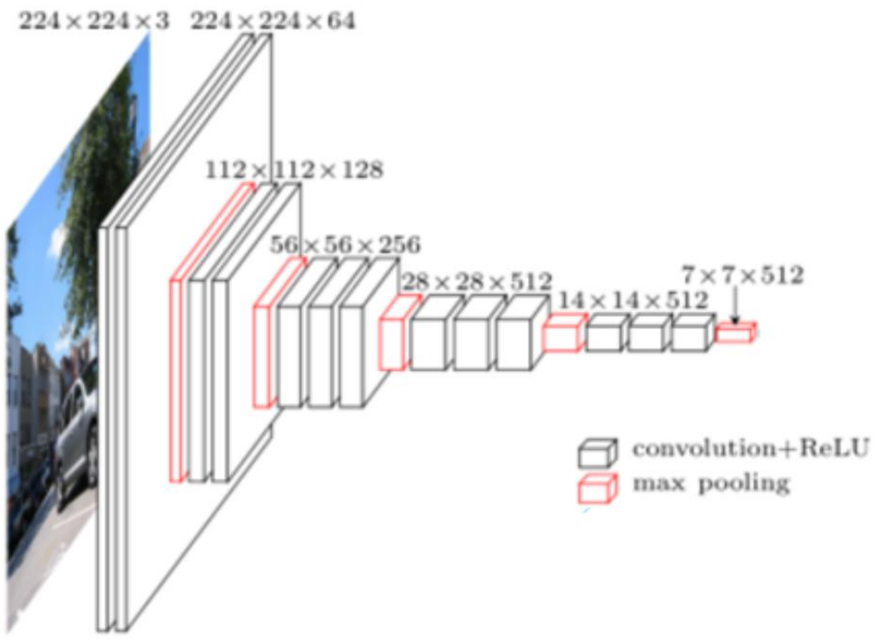


Figure 1. VGG16 Architecture

The final layer is the output tensor used as the feature for the regression layers further as shown in Fig 2. The stack of convolutional filters is followed by 5 Fully Connected (FC) – layers. The first layer has 4096 neurons, second layer has 1024 neurons, third layer has 512 neurons, fourth layer has 100 neurons and fifth layer (output layer) has 4 neurons, each of which outputs one of the bounding box coordinates. All hidden layers are equipped with the rectification (Leaky Relu) non-linearity. Drop-out layers in between ensure that the model does not over-fit. The inference behind using the pre-trained weights is shown in Fig 3. Even though the weights are tuned to the classification model in standard VGG16 network, the high-level features are invariably advantageous for bounding box regression.

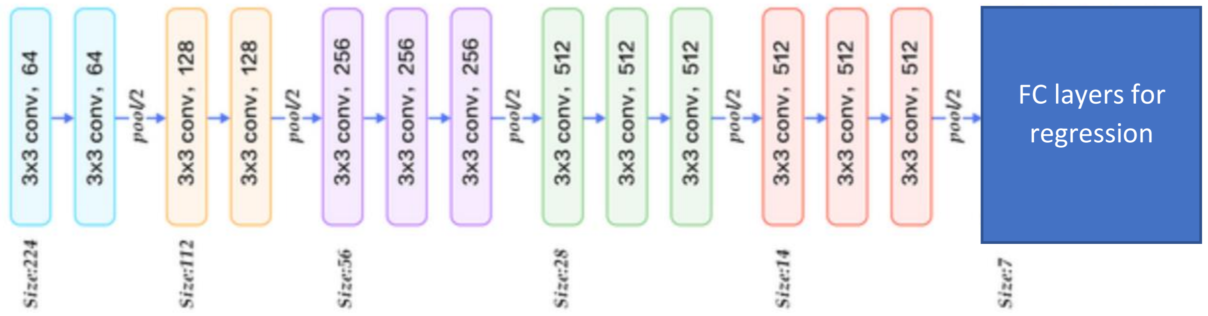


Figure 2. VGG16 convolutional layers with regression model on top

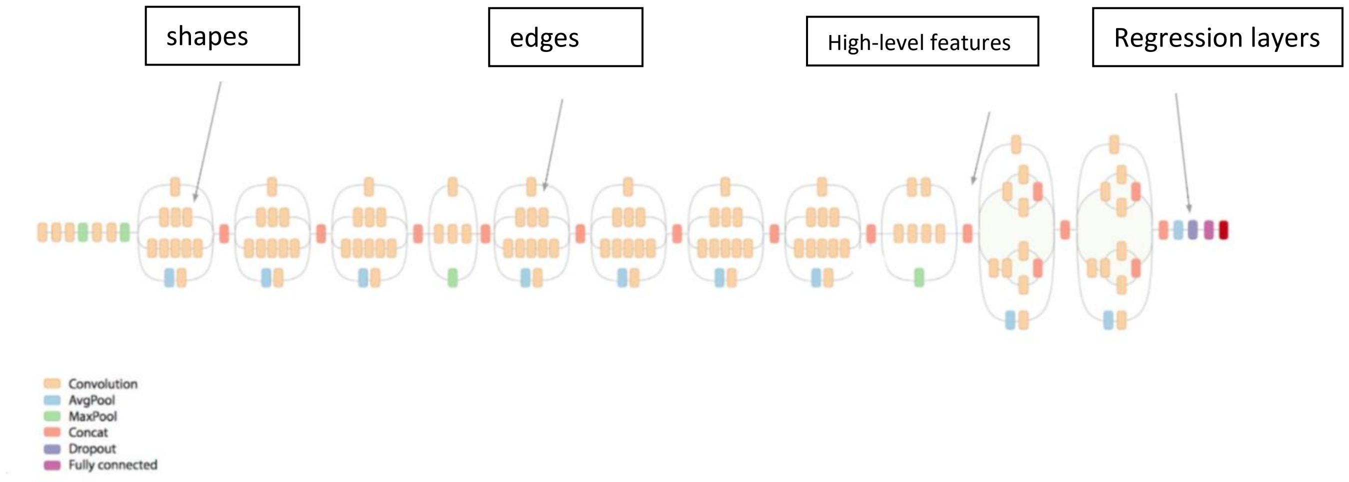


Figure 3. Transfer Learning Feature extraction inference for VGG16

An example of the transfer learning model for classification task using VGG16 is shown in Fig 4. As the network progresses deeper through the layers, the dimensionality decreases and only the relevant parts of the image are retained as features.

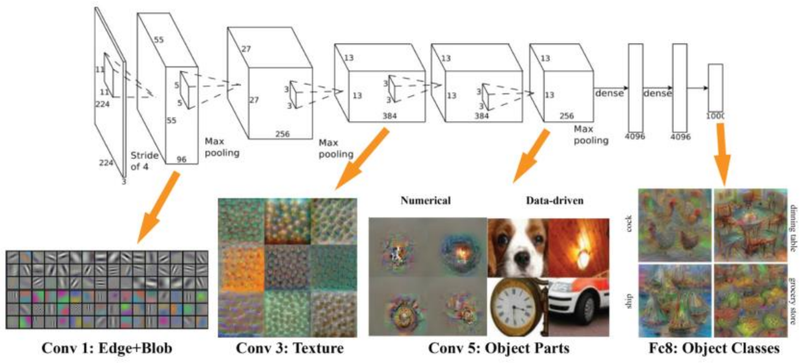


Figure 4. An example of transfer learning model for object classification

Loss functions for a regression model are commonly Mean Squared Error, Mean Absolute Error, Log Cosh, Huber, and Quantile. MAE loss is useful when training data has lots of outliers but has uniform gradient for all error values. Huber loss is more robust to outliers than MAE and MSE. Log-cosh is the logarithm of the hyperbolic cosine of the prediction error. It has the advantages of MSE and Huber loss and at the same time is twice differentiable everywhere, unlike Huber loss. As expected, for my experiments, Log-cosh loss gave maximum accuracy. (Figure 5)

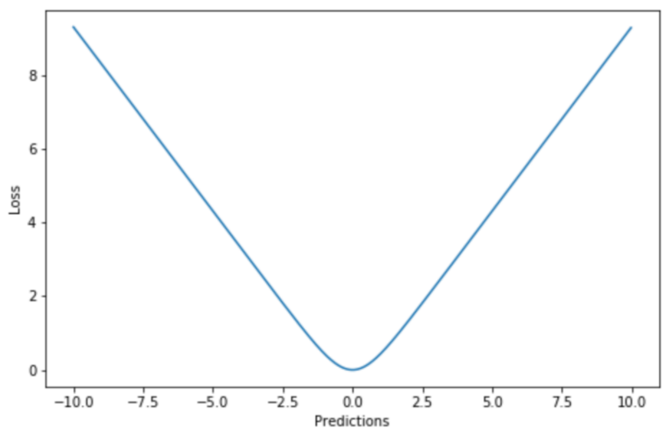


Figure 5. Log-cosh loss function

Choice of optimizer between Stochastic Gradient Descent (SGD) and RMSprop did not improve the accuracy by a significant amount given that the configurations of these optimizers ie. learning rate, momentum, and decay value remain the same. A low learning rate of 0.001, momentum value of 0.9, and decay of 1e-6 gave the highest validation set accuracy.

2. SSD Architecture

The SSD framework and training methodology is described in this section. Fig 5 describes the evaluation of the default boxes of different aspect ratios at each location of feature maps with different scales (e.g 8 X 8 and 4 X 4 in b and c). For each default box, confidence scores and offsets are predicted. While training, these default boxes are matched with the ground truth boxes. In Fig 6, two default boxes with the cat and dog, are treated as positives and the rest as negatives. The total loss becomes the weighted sum of localization loss (Smooth l1) and confidence loss (Softmax).

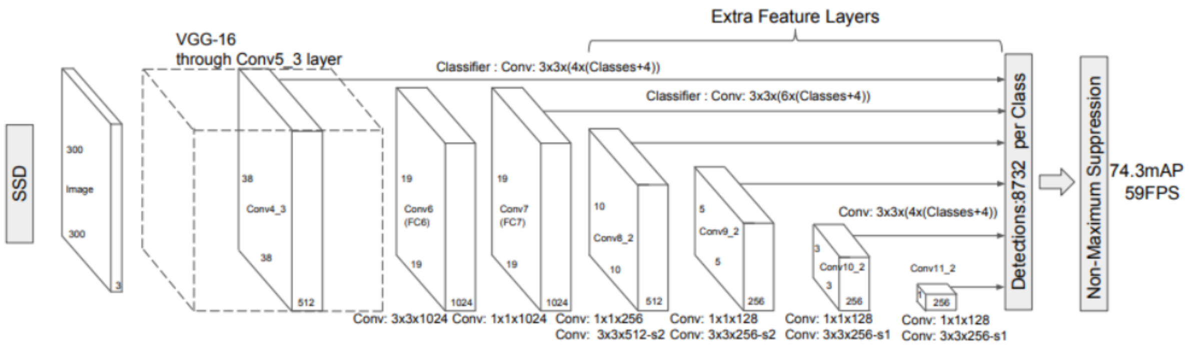


Figure 5. Architecture of SSD

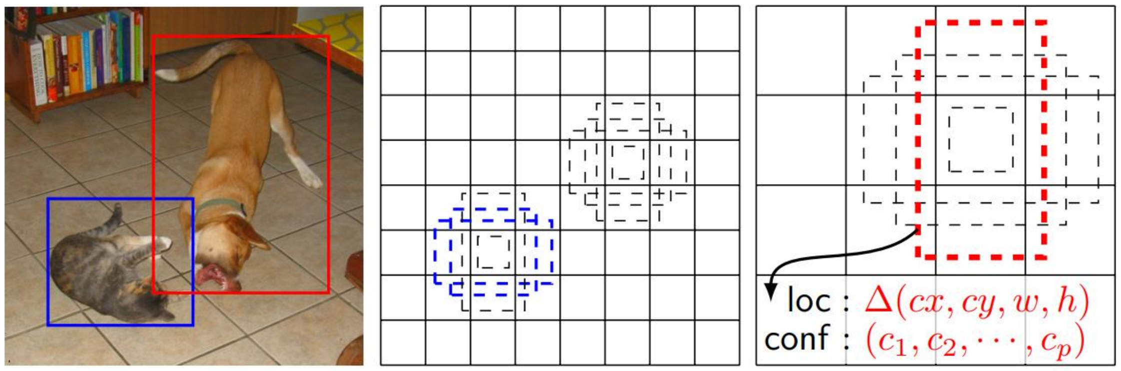

loc : $\Delta(cx, cy, w, h)$
conf : $(c_1, c_2, \cdots, c_p)$

Figure 6. Image with ground truth boxes, 8 X 8 feature map, 4 X 4 feature map

SSD is based on a feed-forward convolutional network predicting fixed size bounding boxes and scores for the presence of object classes in those boxes, followed by non-maximum suppression step to filter the final detections. Figure 7 shows the Pascal VOC image of a cat before the maximum suppression step. Multiple bounding boxes with different class-scores are predicted.

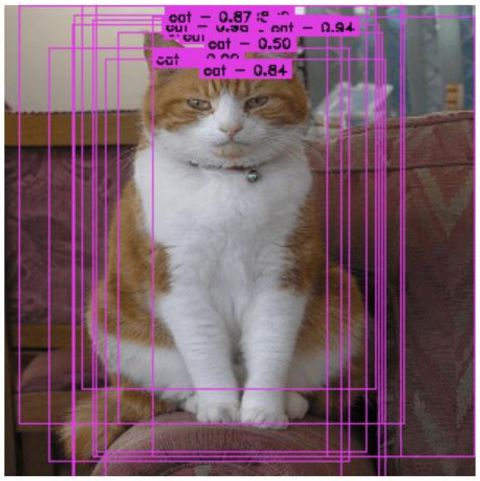


Figure 7: PASCAL VPC cat image with multiple bounding box predictions with different class scores

The primary layers consist of the convolutional network (for e.g VGG16), but the auxiliary layers have the following features:

- Multi-scale feature maps: The convolutional layers progressively decrease in size and allow predictions at multiple scales unlike YOLO that operates on a single scale feature map. Fig 8 illustrates the architecture of multi-scale convolutional prediction of the location and confidences of Multibox.

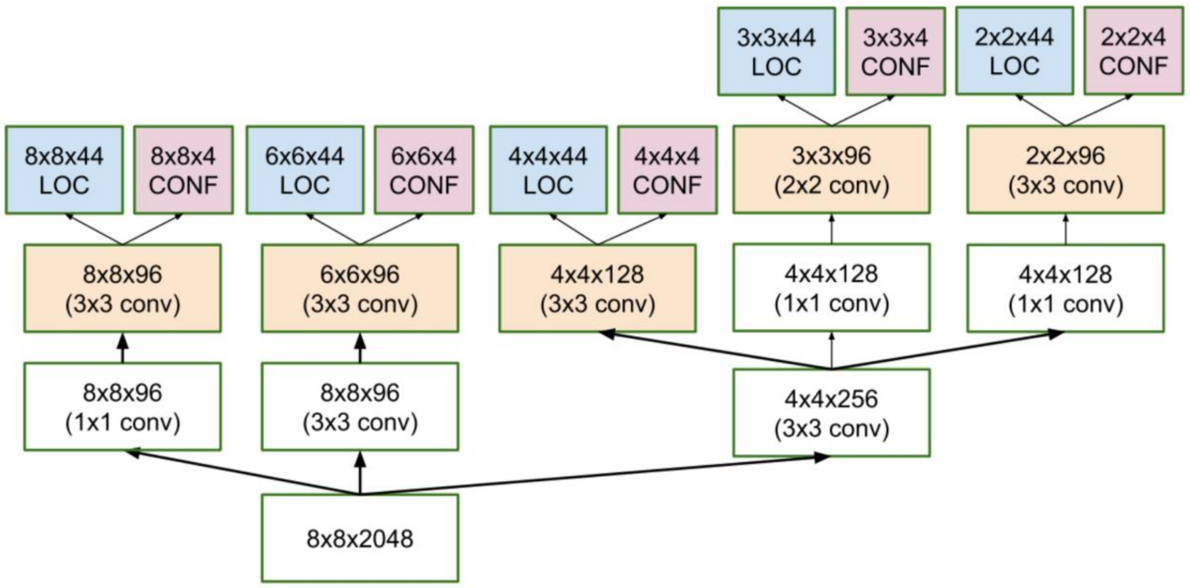


Figure 8. Architecture of multi-scale convolutional prediction of the location and confidences of Multibox

- Convolutional predictors: Each feature layer produces a fixed set of predictions for shape offsets and class scores of a category. For a feature layer of size m × n with p channels, the basic element for predicting parameters of a potential detection is a 3 × 3 × p small kernel that produces either a score for a category, or a shape offset relative to the default box coordinates.
- Default boxes and aspect ratios: A set of fixed-size boxes corresponds to each feature map at the top of the network. The relative offsets remain the same as the boxes tile the feature map in convolutional manner. Specifically, for each box out of k at a given location, c class scores and 4 relative offsets are found resulting in (c+4)k filters applied around each location in the feature map. Total of (c+4)kmn outputs for a m x n feature map.[2]

Training

The key difference between training SSD and training a typical detector that uses regression models or region proposals, is that ground truth information needs to be assigned to specific outputs in the fixed set of detector outputs. After finding these outputs, the loss function and bac propagation are applied end-to-end. Following steps summarize the training process of SSD:

- Matching strategy: For training the network to determine which ground truth box corresponds to the default boxes, a matching strategy is required. Jaccard overlap is the measure that is used to determine similarity. A Jaccard overlap greater than a threshold (0.5) signifies that the default box is close to the ground truth box. This is the goal for the network to learn. In case of multiple overlaps, it picks the one with maximum overlap. The image has multiple predictions before the maximum suppression stage as shown in Fig 9.

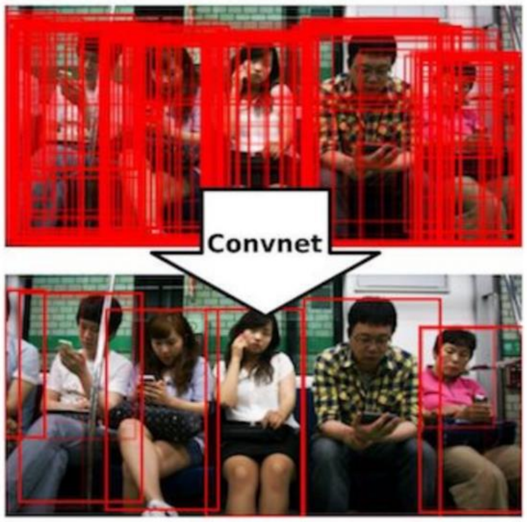


Figure 9. Output before maximum suppression and after maximum suppression.

- Training loss: Since SSD tracks multiple objects, the Multibox objective is extended to handle multiple object categories. Let $x_{ij}^{p} = \{1,0\}$ be an indicator for matching the i-th default box to the j-th ground truth box of category p. In the matching strategy above, we can have $\sum_i x_{ij}^{p} = \{1,0\} \geq 1$. The overall objective loss function is a weighted sum of the localization loss (loc) and the confidence loss (conf): $L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g))$,

  where N is the number of matched default boxes. [2]

  The localization loss is smoothl1 loss with categorical cross-entropy and the confidence loss is softmax loss. The localization loss is used to regress the offsets for the center of the default bounding box and for its width and height.
- Choosing scales: By choosing the feature maps from several different layers in a single network for prediction, different object sizes are handled.
- Hard negative mining: This step is extremely crucial for optimization and stability in training. Usually the number of negative default boxes is large, especially if number of possible default boxes is large. This step sorts the predicted boxes in the order of their confidence scores and picks the top ones so that the positive: negative ratio is at most 3:1. An example of hard negative mining is shown in Fig 10.

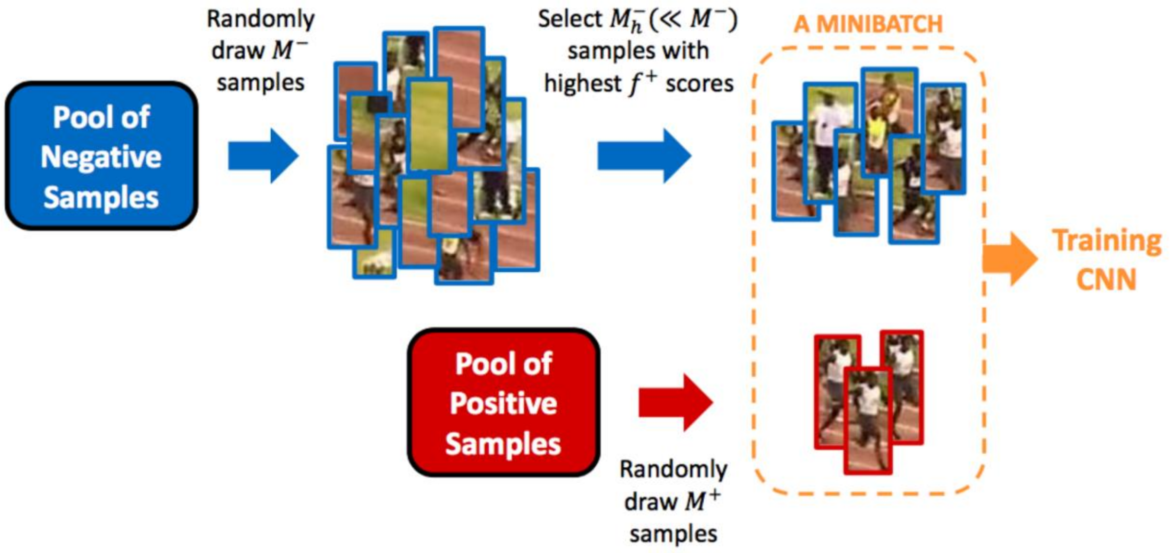

Pool of Negative Samples
Randomly draw M^- samples
Select $M_h^- (\ll M^-)$ samples with highest f^+ scores
A MINIBATCH
Training CNN
Pool of Positive Samples
Randomly draw M^+ samples

Figure 10. Hard negative mining example

- Data augmentation: Another unique feature of this method is data augmentation, that makes the model very robust. Random sampling by one of the following options:
  1. Use entire image
  2. Sample a patch with Jaccard overlap of 0.1,0.3,0.5, 0.7 or 0.9
  3. Randomly sample a patch

Experimental Results:

Vgg16 model

Pascal VOC 2007 and 2012 jpeg images have been used to train the VGG16 model. The training set is partitioned into 80:20 ratio sets, to make the validation set. The ground truth boxes are read from the annotations file along with the filename, size, object name, and difficulty attribute. The box coordinates are scaled by the width or height of the image, depending on whether they are ‘x’ or ‘y’ coordinates. The scaling step greatly improves the performance as the network must now learn a smaller range of values between 0 and 1. The stack of pre-trained convolutional filters in VGG16 is used to extract features. These feature sets are passed to the FC-layers with the following configuration (Fig 11):

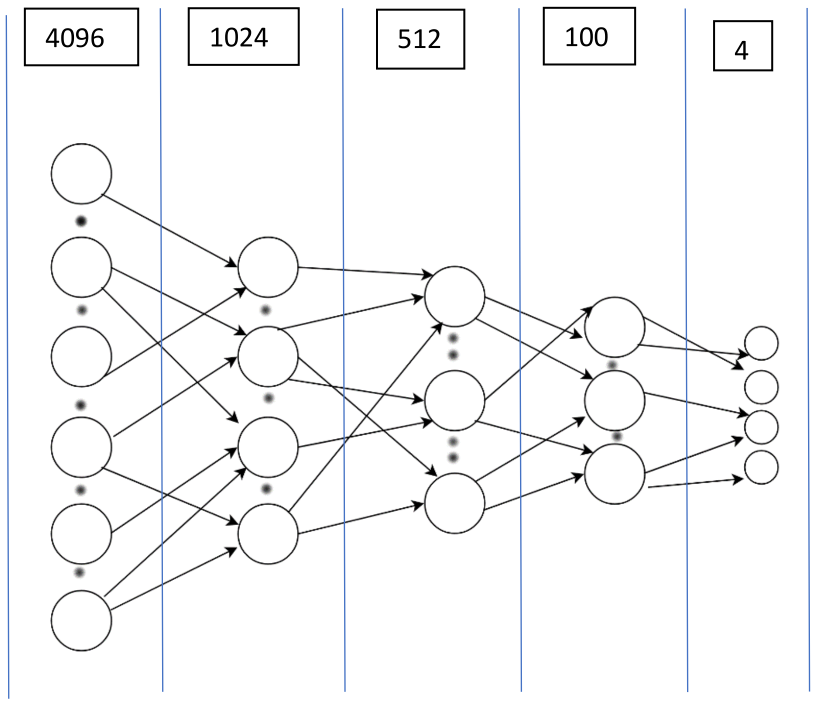


Figure 11. FC-layers in VGG16 regression model

Table 1 shows the activation functions in each of these layers.

FCN layer	Neurons	Activation
Layer 1	4096	Leaky Relu
Layer 2	1024	Leaky Relu
Layer 3	512	Leaky Relu
Layer 4	100	Leaky Relu
Layer 5	4	Linear

Table 1. VGG16 model regression layers

Input images are resized to 224 X 224 X 3 before feeding it into VGG16 network. The ground truth labels are the scaled bounding box coordinates. This regression network is fed with the extracted features and loss function minimized using stochastic gradient descent is Log-cosh. Activation functions for the first four layers are Leaky Relu and the last layer ha a Linear activation. Drop outs between the hidden layers prevent over-fitting. The stochastic gradient descent has a learning rate of 0.001, momentum of 0.9 and decay value of 1e-6. Advantage of using stochastic gradient descent is that it is faster than the standard gradient descent back propagation method. Nevertheless, it has the same expectation over loss as that incurred by taking all training examples and not just one training example for back propagation. Log-cosh loss handles outliers better than MSE and MAE and is smoother than Huber loss. Additionally, it is twice differentiable as discussed before.

The validation loss converges to a minimum after 15 epochs as is depicted in the Fig 12. Fig 13 shows that the accuracy reached its maximum after about 15 epochs too. The Jaccard overlap (Intersection Over Union(IOU)) for the predicted boxes is 0.65.

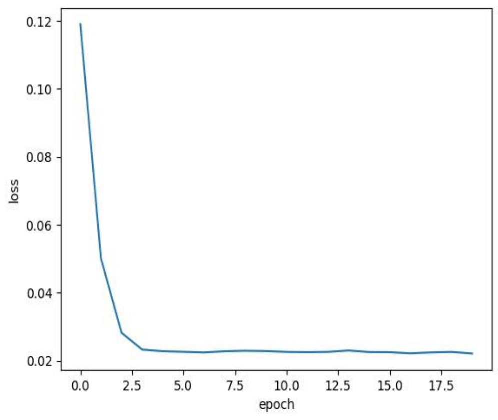


Figure 12. Validation loss vs epochs

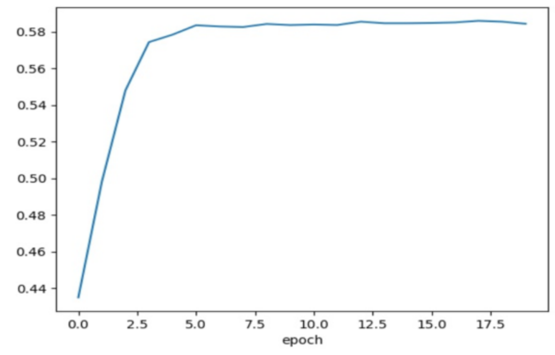


Figure 13. Accuracy vs epochs

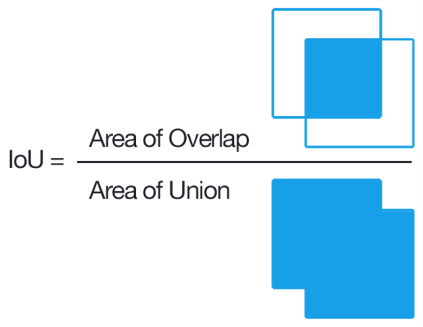


Figure 14, Jaccard Overlap representation

An interpretation of the IOU measure is shown in Fig 15 below:

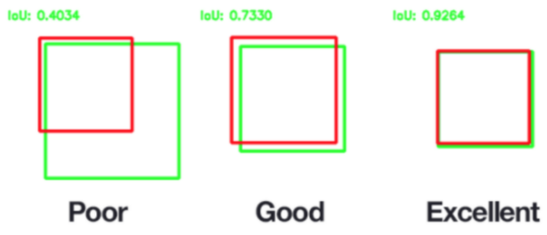


Figure 15. Different Jaccard Overlaps and their interpretation

Figure 16 show the results of training the VGG16 model with random initialization of weights. As we will see later, these trends are random and do not converge as smoothly as the network with pre-trained weights.

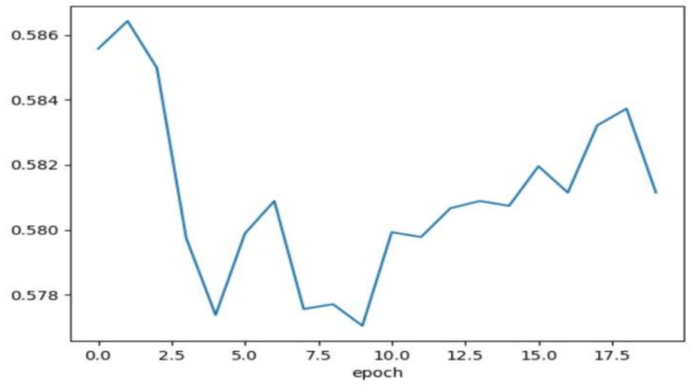
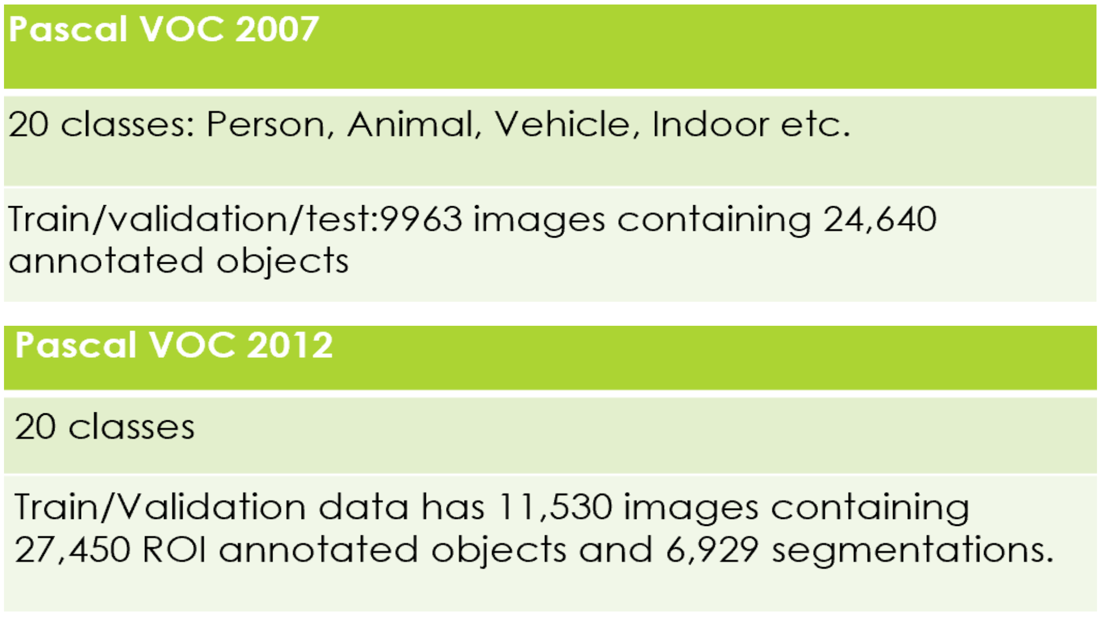


Figure 16. Accuracy vs epochs plot for training with random weight initialization of VGG16

The datasets used for training is described in the following section(Table 2). It is a heavy dataset that required training on Euler cluster using 4 NVIDEA GTX 1080 GPUs.



Pascal VOC 2007
20 classes: Person, Animal, Vehicle, Indoor etc.
Train/validation/test:9963 images containing 24,640 annotated objects

Pascal VOC 2012
20 classes
Train/Validation data has 11,530 images containing 27,450 ROI annotated objects and 6,929 segmentations.

Table 2. PASCAL VOC dataset

The results for the VGG16 is shown in the Table 3:

Input resolution	224 X 224 X 3
Feature extraction	VGG16
Regression loss	Log-cosh
Optimizer	SGD(lr=1e-2,momentum=0.9, decay=1e-6)
Average IOU	0.65
Mean Average Precision	60.3%
Epochs	50
Training set	27, 188 Jpeg images and annotations from Pascal VOC 2007 and 2012 dataset
Convergence time on GPU	~5 hours
Software platform	Keras with Tensorflow backend, Euler cluster for GPU access, 4 NVIDEA GTX 1080 GPUs.

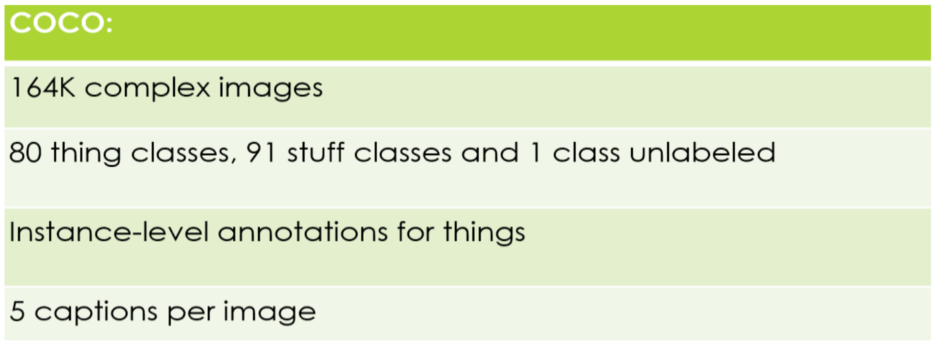
Table 3. Results for VGG16 training

Experiment 2:

SSD network is trained on Pascal VOC 2007 and 2012, and COCO dataset. The base network is VGG16, which is pre-trained on the ILSVRC CLS-LOC dataset. Fc6 and fc7 is converted into convolutional layers, parameters from fc6 and fc7 are subsampled, pool-5 is changed from 2-X 2-s2 to 3 x 3-s1, and trous algorithm is used to fill the "holes". [5] All drop-out layers are removed from fc8 layer. I fine tuned the resulting SGD with initial learning rate of 1e-3, momentum 0.9, 0.0005 weight decay, and batch size of 32. The full training and testing code are built on Caffe and is open source at: https://github.com/weiliu89/caffe/tree/ssd. MultiBox's loss function in SSD can be summarized as:

- Confidence Loss: 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: 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.

The datasets used for training is the Pascal VOC 2007+2012 dataset along with COCO dataset. The Pascal VOC dataste is described in the previous experiment. COCO dataset is described in Table 4. It is a heavy dataset that required training on Euler cluster using 4 NVIDEA GTX 1080 GPUs.



COCO:
164K complex images
80 thing classes, 91 stuff classes and 1 class unlabeled
Instance-level annotations for things
5 captions per image

Table 4. COCO dataset

The SSD results for training are in Table 5:

Input Resolution	512 X 512 X 3 for Pascal, 300 X 300 X 3 for COCO
Base feature extraction model	VGG16
Optimizer	SGD(learning rate=0.001,momentum=0.9, decay=0.00001)
Localization loss	Smoothl1
Confidence loss	Softmax
Hard negative mining ratio	3:1
IOU threshold	0.5
Training time	~10 hours
Training set	Pascal VOC + COCO
Mean Average Precision	75%
Mean IOU	0.85
Software platform used	PyTorch, Euler cluster for GPU access, 4 NVIDEA GTX 1080 GPUs

Table 5. Results for SSD training

The precision for each of the 20 classes in VOC dataset is depicted in Figure 15 below:

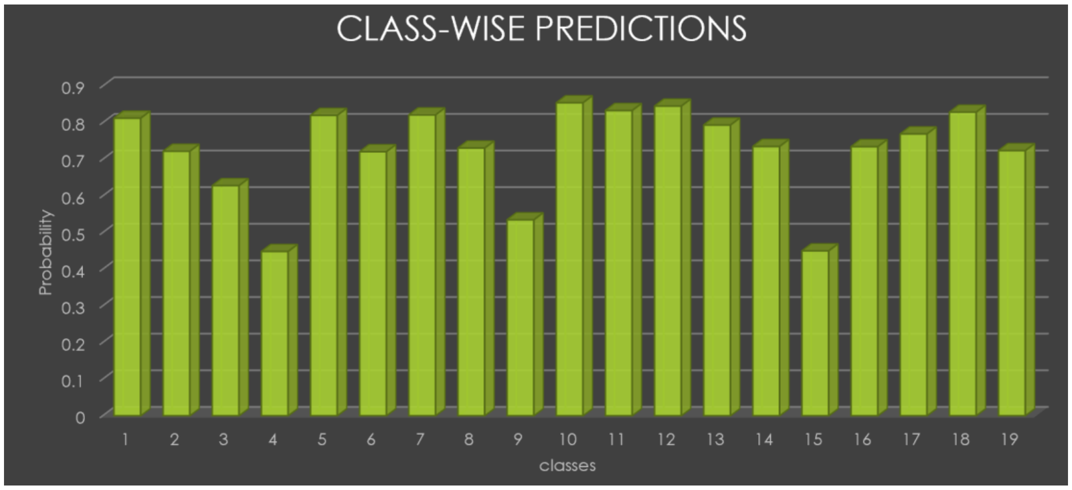


Figure 17. mAP for the 20 classes in Pascal VOC dataset

Total loss to be minimized by backpropagation for the network is a weighted sum of localization loss and confidence loss.

multibox_loss = confidence_loss + alpha * location_loss

The *alpha* term helps 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 the predictions closer to the ground truth. The decrease in the multibox_loss is more pronounced if we view every 100 iterations compared to the one where we view only 100 iterations. The model converges to a minimum after about 40,000 iterations (Fig 18).

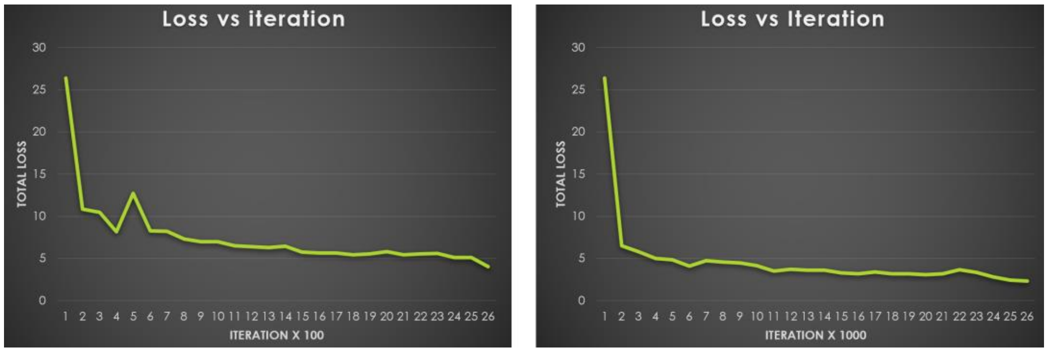
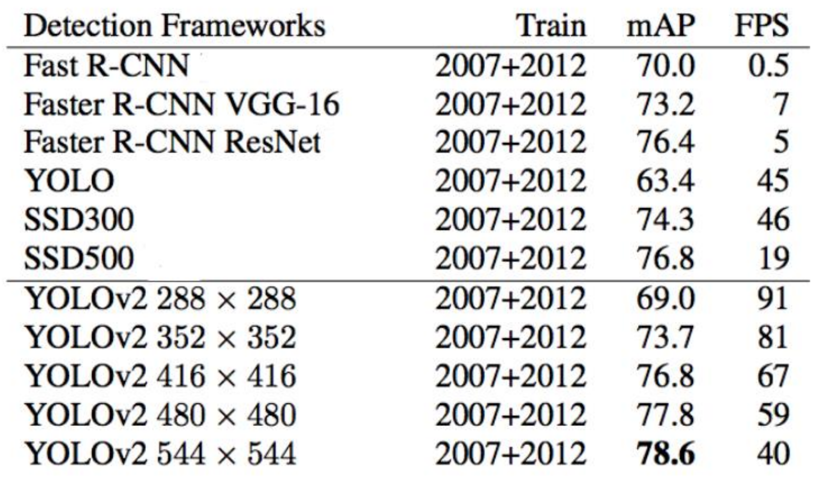


Figure 18. Loss vs number of iterations

Conclusion:

The state of the art for object detectors is shown in Table 6.



Detection Frameworks	Train	mAP	FPS
Fast R-CNN	2007+2012	70.0	0.5
Faster R-CNN VGG-16	2007+2012	73.2	7
Faster R-CNN ResNet	2007+2012	76.4	5
YOLO	2007+2012	63.4	45
SSD300	2007+2012	74.3	46
SSD500	2007+2012	76.8	19
YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40

Table 6. State-of-the-art results for training on Pascal VOC 2007 dataset [3]

Now, we can compare the VGG16 model and SSD model with the state-of-the-art. The VGG16 model has resemblance to the YOLO network as both have only convolutional layers and a regression model on top. The closeness in their accuracy is not surprising and it can be concluded that the model has closely reproduced state-of-the-art results. Some observations from the VGG16 model from the paper[1]:

- The transfer learning model learns to predict the bounding boxes without any region proposals or default boxes.
- 80% of the execution time is consumed in extracting features from the VGG16 convolutional layers. With a faster extraction network, the model will perform better.
- Real-time detection is extremely slow and all it gives is good accuracy for localization and classification.

The SSD network implemented in this project is the SSD300 variant and its accuracy correctly overlaps with the state-of-the-art results. Some additional observations from the SSD paper [2]:

- Accuracy in detection is better with more default boxes, with a negative impact on speed
- Almost 80% of the execution time is spent on the base VGG-16 network: With a faster extraction network, the model will perform better.

- SSD confuses objects with similar categories (e.g. animals). This is probably because locations are shared for multiple classes
- SSD results are not precise for smaller objects, as they may not appear across all feature maps. Increasing the input image resolution alleviates this problem but does not completely address it[4].

References:

1. Karen Simonyan, Andrew Zisserman, *Very deep neural networks for large scale image classification*, (Visual Geometry Group, Department of Engineering Science, University of Oxford).

2. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng- Yang Fu,Alexander C. Berg *SSD: Single Shot MultiBox Detector*, (Part of the Lecture Notes in Computer Science book series (LNCS, volume 9905))

3. Object detection: speed and accuracy comparison(Faster R-CNN, R- FCN, SSD, FPN, RetinaNet and YOLOv3), (Medium)

4. Understanding SSD Multibox- Real-time object detection in deep learning, (Towards data science)

5. Holschneider, M., Kronland-Martinet, R., Morlet, J., Tchamitchian, P.: A real-time algorithm for signal analysis with the help of the wavelet transform. In: Wavelets. Springer (1990) 286–297