Comparative study of Faster R-CNN and YOLOv2 and its modified version on images and videos

Harshit Aggarwal[1], Gaurav Shokeen[2], Rahul Aggarwal[3]
Mrs. Sudha Narang[4]
[1,2,3]Students of Maharaja Agrasen Institute of Technology
[4]Assistant Professor Computer Science and Engineering Department

Abstract

In this paper we use CNNs to detect cars in a scene for applications in self driving cars and mobile object tracking applications and do a comparative study of the results of two different algorithms. We try to use algorithm YOLOv2 and Faster R-CNN to perform this task. For effectively detecting small sized cars, we divide the input images into dense grids to obtain finer feature maps.
YOLOv2 outperforms all the other methods in both speed and detection.

1. INTRODUCTION

Driverless cars are the most important invention in the next coming years. It has great impact on the way humans function outside of living spaces. Deep learning is a field which has come a long way in bringing this invention come close to reality.
However, significant work has been made in the last few years on object detection with convolutional neural networks(CNNs). In this paper, we apply CNNs to object detection in images and try it on videos.
We tried to reduce the computational requirement for this task by bringing some changes to the YOLOv2 model. We then did a comparative study on the results by our model and other benchmark models. We use dataset provided by drive.ai to coursera. We use faster
R-CNN and YOLO to detect and classify the objects in each image as benchmark results. We apply different image processing techniques to experiment with the models and output. The output of our detector is the processed photo or
video. We identify the objects in the images to draw a bounding box around them. We can also do some pixel coloring of the box area to show the objects in the image and increase the ease to view the results.

To accomplish our goal, we must address several challenges. The pipeline of object detection generally uses hand-crafted features to extract regions, and then combines classifiers to filter out the negatives.

Nowadays, deep convolutional networks are applied to image recognition and object detection, becoming increasingly fast and accurate. Convolutional neural networks (CNNs) can learn features from many samples without preprocessing, which avoids the design difficulty of hand-crafted features and learns more generalized features. CNN has already been presented as a classifier in machine learning and has been used in many different fields.
We describe how we address these challenges in the following paper.

2. RELATED WORK

Object detection is one of the main goal of computer vision and there are many algorithms that have presented novel ways of performing the task. We do a comparative study on the YOLOv2 and Faster R-CNN. YOLOv1 [1,2] divided the image into a grid with each grid cell predicting bounding boxes, confidence scores as well as conditional class probability predictions for each cell. One Bounding box could only predict one class which was prone to errors as there could be more than one car in a given bounding box. YOLOv1 suffered from errors when there are many small objects inside one bounding box [2].
YOLOv1[2] also faced a variety of problems as compared to various state-of-the-art detection systems. Further Error analysis of YOLOv1 and Fast R-CNN [5,2] showed that YOLOv1 makes more number of bounding and localization errors. YOLO

also had lower recall compared to region proposal algorithms. YOLOv1 has higher recall and localization as compared to earlier methods. Faster R-CNNs [6] work a lot better as compared to Fast R-CNNs. Introduction of Region Proposal Network (RPN) reduces the proposal time in Faster R-CNNs.
Clearly, YOLOv1[2,3] performed a lot faster compared to the other methods. YOLOv1 detection is suffering about 10 maP in detection. Whereas YOLOv2[1] outperforms all the methods in both speed and detection. At 67 FPS, YOLOv2 gets 76.8 mAP on VOC 2007.[1] At 40 FPS, YOLOv2 gets 78.6 mAP (Mean Average Precision).Faster R-CNN uses VGG-16 model that achieves state-of-the-art object detection accuracy on PASCAL VOC 2007, 2012, and MS COCO datasets with only 300 proposals per image. [1,2, 5, 6]. The two algorithms have a tradeoff between speed and accuracy, so each algorithm has specific applications it is better suited for. Below graph and table shows comparisons of various algorithms' speed on VOC 2007 dataset and VOC 2007 + 2012 dataset respectively [1,3]:

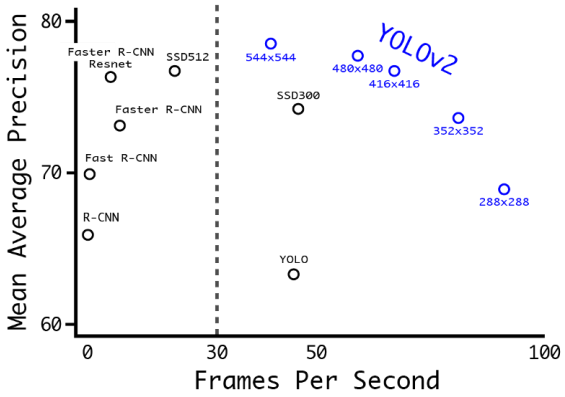

Fig:1 Comparison of various detection algorithms on average precision with respect to frames per second

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	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:1 Benchmarked results various detection algorithms

algorithm by self-learning when faced with the various problems.

The computational requirements are very high, and the process requires large amounts of resources. Once the model is trained then also sometimes amount resource required is high. However smaller and lighter implementation are possible and required resources decreases by large amount as compared to training phase. The amount of data is a key factor for deep learning-based methods. The papers that doesn't use deep learning algorithms used to have better accuracy as compared to deep learning. But recent advancements have proved that deep learning methods are more robust and can be more accurate in specific circumstances.

3. METHODS AND IMPLEMENTATION

Convolutional neural networks (CNNs) have exhibited huge success in image recognition. Several detection networks have been adapted from image recognition networks and further advanced. In this paper, we build on YOLOv2[2,4] to construct a single convolutional neural network for car detection.

3.1 Object detection:

This step involves using Convolutional Neural Networks to detect the object inside images. We apply three different models: YOLO and Faster R-CNN, our model.

3.1.1 The Network Architecture of YOLOv2

The full network is shown below:

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Table:2 YOLOv2 architecture

3.1.2 The Network Architecture of Faster-RCNN

The architecture of Faster R-CNN [6] is complex because it has several moving parts. We'll start with a high-level overview, and then go over the details for each of the components.

It all starts with an image, from which we want to obtain:

- a list of bounding boxes.
- a label assigned to each bounding box.
- a probability for each label and bounding box.

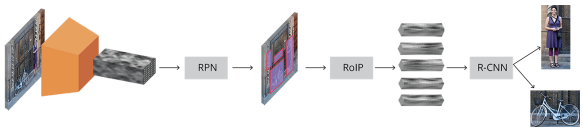

Fig 2: Faster R-CNN architecture

Fig:3 VGG16 architecture

3.1.3 Fine Tuned model with modifications

We tried several network models according to the a fore mentioned CNNs, and finally propose a network model in this paper, as shown in Table 1. We found that the networks of YOLOv2[2] and Faster R-CNN [6] pay more attention on the top layers because they are pre-trained on several objects. Compared to Faster R-CNN, YOLOv2 simplifies the pipeline of the process and has better mAP.

Our Model was designed to make full use of the channels information by 1×1 convolutional layers. We attempted to add a 1×1 convolutional layer followed by another 1×1 convolutional layer to learn more features via cross channels. But this didn't result in better mAP for us.

We decreased the repeated convolutional layers in the top layers and designed the network with a fine grid in the model. This reduced the number of parameters and reduce the running time. We know that the feature maps in the intermediate layers contain specific information.

3.2. Loss Functions

After putting the complete model together, we end up with 2 different losses. We have the trainable layers in Faster R-CNN and YOLOv2[1,6] and we also have the base network which we have trained (fine-tuned) with some modifications.

L2 regularization has been used along with individual algorithms having different regularizations of their own.

3.3 Dataset

The dataset is provided by Drive.ai, a company building the brains of self-driving vehicles. The dataset has been provided to coursera which we have also used.

To collect data, they've mounted a camera to the hood (meaning the front) of the car, which takes pictures of the road ahead every few seconds while the driver drives around.

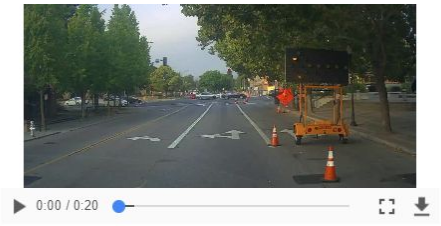

Fig: 4 Dataset collection using camera on top of a car in Silicon Valley

3.4 Hardware

We have used Coursera GPUs and Tesla K-80 GPU provided by Google through Google Cloud.
We have run the tests on the same GPUs.The RAM required was 52 GB.
We also used Google Colab which provided free GPUs to run an instance.
The training and testing phase both required these GPUs.

3.5 Region Proposals and thresholds

In Faster R-CNN [6,7], it computes the object scores in a region proposal and thresholds the object detection. In our implementation we varied the threshold from 0.3-0.6 and found it performed best around 0.5.
We use Region proposal network to identify the regions in the image. After applying NMS, we keep the top N proposals sorted by score. In the paper N=2000 is used, but it is possible to lower that number to as little as 50 and still get quite good results.
One of the advantages of using only the RPN [4,5,6] is the gain in speed both in training and prediction. Since the RPN is a very simple network which only uses convolutional layers, the prediction time can be faster than using the classification base network.RPN due to its simplicity

4. IMAGE PROCESSING

We have implemented a gaussian filter followed by randomly selecting pixels to enlarge into circles, followed by another gaussian layer. The three stages of filtering soften the edges between the unfiltered objects and the background.
We also used some unsupervised models to produce noise in the image. We used an Autoencoder to get some noisy images in which it is difficult to see the noise. We tried to finetune the models on these images to create a larger dataset and use it as a form of regularization also.

5. TRAINING AND TESTING

5.1 Training and Testing of Faster R-CNN

We implemented Faster R-CNN [5,6] trained on the drive.ai dataset first in CPU mode with the VGG-16 network with Python which had a very slow performance at around 6 seconds per image. Then on GPU mode we were able to get the timing down significantly. We tried training on ImageNet with few more classed, but it was very computationally expensive which we had to later stop. The weights of the VGG network on ImageNet were pre-trained.
We train using Stochastic Gradient Descent with momentum, setting the momentum value to 0.7. The learning rate starts at 0.0010 and then decreases to 0.00010 after 50K steps. This is one of the hyperparameters that usually matters the most.

5.2. Training and Testing of YOLO

We used the model pre-trained on the ImageNet 1000-class competition dataset from YOLOv2[1,2] to train our detection networks. The pre-trained model could reduce the training time obviously.
In the process of training, the initial value of the learning rate was 0.0001. We observed the downward trend of loss and changed the learning rate to 0.001 if the loss was found to be stable at a value greater than 1.

5.3. Training and Testing of our Modified model

We used pretrained YOLOv2[2,4] model from the official website of YOLO. The pretrained weights used in this exercise came from the same.
We used stochastic gradient descent with momentum to train(finetune) our modified model. We used the same momentum as in the YOLOv2 model. We tried using cyclic learning rate method to set the learning rate, but it didn't give any better result.

6.EVALUATION

The evaluation is done using the standard Mean Average Precision [1,2,4] (mAP) at some specific

IoU threshold (e.g. mAP@0.5). maP is the standard measure for comparing search algorithms. Mean average precision for a set of queries is the mean of the average precision scores for each query.

7. RESULTS

7.1. Experiments

All the training was performed on Nvidia Tesla K80 and Coursera GPUs.

Image Output Results from YOLO

. Fig:5 Output of an image to YOLOv2 model

7.2 Comparison of Faster R-CNN and YOLO

The bounding box on YOLO is more accurate than that of R-CNN on most cases. This could be mainly due to the modifications made to previous versions like introduction of anchor boxes. With regards to significance we see a clear trade-off here. In YOLO, we can get rid of the noise and other detected objects but also lose the accuracy sometimes.
YOLO was faster and more accurate than all the three models.
Faster R-CNN extracts its features through VGG16, but VGG-16 requires 30.69 billion floating point operations for a single pass over a single image at 224 × 224 resolution causing the Faster R-CNN to get very slow.
On our dataset the Faster R-CNN with VGG 16 had a mAP of 24.7 with region proposals equal to 80. We increased it to 100,200 and 300.It varied between 30.2,32.2 and 34.7. So, we were able to achieve more than 90 percent of mAP at region proposals equal to 100 as compared to 300 and 200.Faster R-CNN [6] consumed greater computational resources.
Our model gave mAP of 32.2 which was not much of an improvement. This could be due to less hyperparameter tuning. Also, the image preprocessing step might have regularized the model and resulted in high bias as compared to other models.
YOLOv2[1] performed better and gave mAP of 35.5. Still the results are far from benchmark which could be a result of high threshold and IOU which was set to 0.7.
The main reason behind the low mAP can be less computational resources. The preprocessing steps could also be modified to get better results.

7.3 Analysis

YOLO allowed to detect more objects in the image as compared to Faster R-CNN due to implementation of anchor boxes and non-max suppression. YOLO can thus allow faster processing time and as a result allowed us to playback frames in real time as they were being capture through a web camera. Moreover, in YOLO there are generally fewer bounding boxes found than that of Faster R-CNN, which can help us eliminate some of the non-significant picture subjects, and as a result further reduce our processing time.

8. CONCLUSION

We implemented state of the art techniques to implement object detection and compared the results with our modified model. We conclude through our experimentation on drive.ai dataset that YOLOv2 is faster and has more mAP than Faster R-CNN. We also achieved results that can be implemented in small mobile applications linked with cloud to perform object detection using mobile camera. We also found out that the increase in region proposals in Faster R-CNN resulted in better mAP.Our Future step would be to implement this on mobile application and use it for real world purposes like real world car games, car detection or other object detection tasks also. We would also want to achieve state of the art results on our dataset using YOLOv3 model and modify it to use in our applications.

9.REFERENCES

[1] Redmon, Joseph, et al.” YOLO9000: Better, Faster, Stronger.” arXiv preprint **arXiv:1612.08242**

[2] Redmon, Joseph, et al.” You only look once: Unified, real-time object detection.” arXiv preprint arXiv:1506.02640 (2015).

[3] Girshick, Ross, et al.” Rich feature hierarchies for accurate object detection and semantic segmentation.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

[5] Girshick, Ross. ” Fast r-CNN.” Proceedings of the IEEE International Conference on Computer Vision. 2015.

[6] Ren, Shaoqing, et al.” Faster R-CNN: Towards real time object detection with region proposal networks.” Advances in Neural Information Processing Systems. 2015.

[7] Kong, Yan, et al.” Measuring and Predicting Visual Importance of Similar Objects.” (2016).

[8] Berg, Alexander C., et al.” Understanding and predicting importance in images.” Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE, 2012.