International Journal of Computer Architecture and Mobility (ISSN 2319-9229)
Volume 7 -Issue 10, October 2019

A Review and a Comparative Study of Object Detection Algorithms

Sudhanshu Jaisani , Vrinda Diwan , Parth Katare
sudhanshujaisani38@gmail.com, diwanvrinda@gmail.com, p.katare98@gmail.com
Shri Vaishnav Vidyapeeth Vishwavidyalaya Indore-Ujjain Road, Indore – 453111, India
Indore,India

Abstract—**This paper aims to find the best possible combination of speed and accuracy while comparing different object detection algorithms that use convolutional neural networks to perform object detection. Models which are usually computationally expensive, achieve the best accuracy, but cannot be deployed in a simple setting with limited resources, whereas, faster models fail to achieve similar results compared to their bigger and more memory intensive counterparts. We discuss two ends of the spectrum here comparing three different models ,i.e Single Shot Detector (SSD), Faster R-CNN (Region - based Convolutional Neural Networks), R-FCN(Region based- Fully Convolutional Networks), where on one end we get a model which can be deployed on a mobile device because of its speed and another model which is at the cutting edge of performance when it comes to accuracy. These models are trained and their performance metrics tested on the COCO (common objects in context) dataset.**

INTRODUCTION

State of the art computer vision systems have involved a range of object detection models that use convolutional neural networks in their working. The deployment of these models like YOLO, SSD, R-FCN, R-CNN, etc depends on the kind of usage we want. Google photos has deployed models like SSD MobileNet which is known for its speed and isn't memory intensive, performance isn't the most important factor, but memory efficiency is. In case of self driving cars though, the requirement for an extremely accurate model is a priority, as these real time systems are performing tasks which can lead to a life and death situation in their surroundings. We need to evaluate these models using several evaluation metrics like mAP, memory requirement, testing time. We have only considered models which are quite similar in their architecture on a high level overview. The models that are considered are Faster R-CNN, R-FCN, SSD, these models have used sliding window style predictions and have only used a single CNN network. These models are responsible for the object detection part, whereas the feature extraction part is carried out by the different image classification models , the state of the art ones which have participated in the imagenet competitions. We refer to the object detection models as meta architectures and they are coupled with different feature extractors for eg(VGG, Resnet, MobileNet), to evaluate the various combinations we get through them. We would first describe the different meta architectures and then the different feature extractors, and then compare their combinations by testing them on the objects present in the COCO dataset. In this paper we have tried to perform this experiment in controlled conditions with same hardware conditions for comparison and run these tests multiple times so that they prove to be statistically consistent. We have used the deep learning library Tensorflow for our implementation. Previous experiments by others have been performed on caffe, and tried on other standard datasets like PASCAL VOC (Visual Object Classes). A few have fine tuned the detectors for their specific objects and noted the test times for different models. The main reason for using tensorflow was its portability and ease of use.

2. META ARCHITECTURE

Modern meta architectures all use CNN for object detection , let us have a look at the history of a few meta architectures that we are discussing, and have an in depth look at their inner workings

2.1 R-CNN

The R-CNN model was one of the first models to use convolutional neural networks for object detection. The goal of R-CNN is to take in an image, and correctly identify where the main objects (via a bounding box) in the image. But how do we find out where these bounding boxes are? R-CNN does what we might intuitively do as well -propose a bunch of boxes in the image and see if any of them actually correspond to an object. R-CNN creates these bounding boxes, or region proposals using a process called Selective Search. Selective search, looks at the image through windows of different sizes, and for each size tries to group together adjacent pixels by texture, color, or intensity to identify objects. Once the proposals are created, R-CNN warps the region to a standard square size and passes it through to a feature extractor or image classifier , which is a CNN. On the final layer of the CNN, R-CNN adds a Support Vector Machine (SVM) that simply classifies whether this is an object, and if yes, which object is it.

Limitations of R-CNN:

1.It still takes a huge amount of time to train the network as you would have to classify 2000 region proposals per image.

2.It cannot be implemented in real time as it takes around 47 seconds for each test image.

3.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.

2.2 Fast R-CNN

R-CNN works really well, but is really quite slow for a few simple reasons It requires a forward pass of the CNN for every single region proposal for every single image. It has to train three different models separately -the CNN to generate image features, the classifier that predicts the class, and the regression model to tighten the bounding boxes. This makes the pipeline extremely hard to train. Both these problems were solved in Fast R-CNN by the creator of R-CNN himself. For the forward pass of the CNN, Girshick realized that for each image, a lot of proposed regions for the image invariably overlapped causing us to run the same CNN computation again and again . His insight was simple — Why not run the CNN just once per image and then find a way to share that computation across the proposals? .This is exactly what Fast R-CNN does using a technique known as RoIPool (Region of Interest Pooling). At its core, RoI Pool shares the forward pass of a CNN for an image across its subregions. In the image above, notice how the CNN features for each region are obtained by selecting a corresponding region from the CNN's feature map. Then, the features in each region are pooled (usually using max pooling). So all it takes us is one pass of the original image. The second insight of Fast R-CNN is to jointly train the CNN, classifier, and bounding box regressor in a single model. Where earlier we had different models to extract image features (CNN), classify (SVM), and tighten bounding boxes (regressor),Fast R-CNN[4] instead used a single network to compute all three. Fast R-CNN replaced the SVM classifier with a softmax layer on top of the CNN to output a classification. It also added a linear regression layer parallel to the softmax layer to output bounding box coordinates. In this way, all the outputs needed came from one single network.

Advantage of Fast R-CNN over R-CNN:
It is faster than R-CNN because we don't have to feed 2000 region proposals to the convolutional neural network every time. Instead, the convolution operation is done only once per image and a feature map is generated from it.

Drawback of Fast R-CNN:
The region proposals are still generated by selective search (SS)(as in the case of R-CNN) rather than using convolutional neural network (as in the case of Faster R-CNN). Due to this fact Fast R-CNN is slow in object detection as compared to Faster R-CNN.

2.3 Faster R-CNN
There was still one bottleneck with Fast R-CNN which had to be sorted, that was the region proposer. The very first step to detecting the locations of objects is generating a bunch of potential bounding boxes or regions of interest to test. In Fast R-CNN, these proposals were created using Selective Search, a fairly slow process that was found to be the bottleneck of the overall process. Faster R-CNN found a way to make the step of region proposal almost cost free. The insight of Faster R-CNN was that region proposals depended on features of the image that were already calculated with the forward pass of the CNN (first step of classification). So why not reuse those same CNN results for region proposals instead of running a separate selective search algorithm? Indeed, this is just what the Faster R-CNN team achieved. In this model, a single CNN is used to both carry out region proposals and classification. This way,only one CNN needs to be trained and we get region proposals almost for free. How the Regions are Generated? Let's take a moment to see how Faster R-CNN generates these region proposals from CNN features. Faster R-CNN adds a Fully Convolutional Network on top of the features of the CNN creating what's known as the Region Proposal Network.

Advantage of Faster R-CNN:
Since Faster CNN uses the same Convolutional Neural Network for generating regional proposals and object detection as well, It is much faster in object detection as compared to the previous two algorithms; Namely:R-CNN and Fast R-CNN.

2.4 R-FCN
With the increase in performance, Faster R-CNN was an order of magnitude swifter than its earlier counterpart fast R-CNN .But there was an issue of applying the region-specific component had to be applied several times in an image, this issue was resolved in R-FCN (Region based Fully Convolutional networks) where the computation required per image was reduced drastically, where instead of cropping features from the same layer where the crops are predicted, the crops are taken from last layer of features prior to predictions. The algorithm works faster than Faster R-CNN while achieving comparable accuracy

2.5 SSD(Single Shot Detector)
SSD works by converting discrete output spaces for bounding boxes into sets of default boxes for different aspect ratios and for every feature map location. During predictions, the model generates the scores in each default box for every object detected and scales the default box to fit the object shape. SSD is simpler than other networks as it performs all computations in a single network. SSD combines the predictions from numerous feature maps having different resolutions to handle objects of various sizes. It doesn't involve regional proposal generating or feature resampling like previous networks did, which makes it easy to train and integrate into systems where detection is required.

COMPARATIVE STUDY

Algorithm	Features	Prediction time / image	Limitations
CNN	Divides the image into multiple regions and then classify each region into various classes.	-	Needs a lot of regions to predict accurately and hence high computation time.
RCNN	Uses selective search to generate regions. Extracts around 2000 regions from each image.	40-50 seconds	High computation time as each region is passed to the CNN separately also it uses three different model for making predictions.
Fast RCNN	Each image is passed only once to the CNN and feature maps are extracted. Selective search is used on these maps to generate predictions. Combines all the three models used in RCNN together.	2 seconds	Selective search is slow and hence computation time is still high.
Faster RCNN	Replaces the selective search method with region proposal network which made the algorithm much faster.	0.2 seconds	Object proposal takes time and as there are different systems working one after the other, the performance of systems depends on how the previous system has performed.

CONCLUSION

We have put forward a comparative study of state of the art object detectors which use convolutional neural networks. We have addressed their issues and performance on a common hardware.We discovered that SSD performs much better with light weight feature extractors on bigger images, competing with the most accurate of models. We also found that fewer proposals increase speed without compromising much on the mAP scores.We wish to try different combinations and find better results and sweet spots which can be applied to specific use cases.

REFERENCES

[1] Antonio T, Contextual Priming for Object Detection, 2003. International Journal of Computer Vision, Volume 53, Number 2, 169-191.

[2] Paul V and Michael J, Robust Real-time Object Detection, 2001. Second International Workshop On Statistical And Computational Theories Of Vision – Modeling, Learning, Computing, And Sampling, Vancouver, Canada, July 13, 2001.