Computer Vision
Final Report: Localized object detection with Convolutional Neural Networks

Bardia Doosti
Vijay Hareesh Avula

May 5, 2016

Abstract

Convolutional Neural Networks (CNNs) are the best tool to categorize the images. This classifier only labels the whole picture as a class and cannot localize the object in the picture. In this project, we want to implement an object detector using SIFT key points. After detecting the region of the object, we will classify the objects with a fine-tuned CNN.

1 Introduction

Multi-Object detection is one of the active fields of computer vision. The goal of this field is detecting all the objects of a given image.

![Figure 1: Input (left) and the objects that should be detected (right)](image)

Since the Convolutional Neural Networks can detect the objects with more than 90% of accuracy, we can use a fine-tuned neural network to detect an object. The problem of CNNs and all other classifiers is that they only can classify the whole image to an object and cannot localize that object in the image. In this project we are going to implement some techniques to detect the border of each object in the image and send these images to a well-known CNN and label that region.
2 Previous Works

The literature on object detection is vast and in this section we focus on the work carried out on the basis on Convolutional Neural Networks that have achieved a state-of-the-art performance on a number of image recognition benchmarks like VOC-2012 and ILSVRC-2012. We have specifically taken into account the popular Ross Girshik et al. Faster R-CNN [5] as our benchmark since it is claimed to be the fastest deep network that could recognise an object in a given image with a confidence. Girshiks Fast R-CNN [3] (previous version of Faster R-CNN) has been implemented on BVLCs Caffe tool\(^1\) leveraging its capability of neural network to extract regions out of images, computing CNN features out of it and Classifying the extracted regions. They have managed to achieve an average segmentation accuracy of 47.9% on the VOC 2011 test set and 58.5% mAP on the 2007 dataset. A similar contribution made by Dumitru et al. [1] on scalable object detection using deep neural networks was studied to consider a regression based approach towards classifying the regions of the image. However, in our approach, we have used the hierarchical clustering on the SIFT interest points (of course with two magic parameters to tune) to find out various regions and considered these clusters as individual regions as in the approach of Fast R-CNN to get a better accuracy.

Figure 2: A schematic graph of faster R-CNN algorithm

3 Object detection using Scale-invariant feature transform

In [4] David Lowe introduced the Scale-invariant feature transform (SIFT) and the idea of using this points for object detection. In this project we want to use these key-points, found by SIFT algorithms, to detect the objects. SIFT points detected on a test imaged are shown on figure 3. As we saw on many test images, the key-point around the objects are more dense than other parts of the image and this is the whole idea of this project; cluster the nearest points and ignore the others.\(^2\)

\(^1\)http://caffe.berkeleyvision.org/
\(^2\)The picture shown in figure 3 is adapted from Computer Vision assignment 2 sample pictures
As the number of SIFT points are much more greater than the number of objects in the picture we cannot make a window around all the SIFT points and detect them as the object.

3.1 Step 1: SIFT Threshold

In the first step we changed the edge threshold of the SIFT describers to get relatively small number of points. Although, this will reduce the number of key-points all over the image, the key-points around the objects are still dense. Also reducing the key-points saves a lot of time in the next (clustering) steps of the algorithm. The reduced SIFT points are shown in figure 4.
3.2 Step 2: Clustering

Due to the high number of key-points in the image, we cluster them and try to use these clusters as an intuition about the objects. Note that we don’t cluster the key-points based on their SIFT descriptors, because as it is shown in figure 5 it just clusters the point which their SIFT vector are very similar but what we want is the clustering based on the position of the key-points. So we clustered the key-points based on their $x$ and $y$ position.

![Figure 5: Key-points Clustered by $k$-means Algorithm based on the SIFT descriptors](image)

For this we compared the results of two well-known clustering algorithms; $k$-means and Hierarchical Clustering.

3.2.1 $k$-means Clustering

In the $k$-means algorithm first we pick $k$ points randomly as the $k$ cluster centers. Then we assign each point to the nearest cluster centers and then update the centers.

$k$-means did not result very well. Because we cannot determine which points in the cluster were closer and which points were farther. The other problem of this algorithm is that before running the algorithm we should know how many clusters do we have (how many objects in the picture) which is not possible. Also $k$ means uses all the key-points of the image, which is not good. Because we want to just merge the dense areas and not to use the key-points which are not relevant. The SIFT points clustered by $k$-means algorithm are shown in figure 6.

3.2.2 Hierarchical Clustering

In hierarchical clustering (Agglomerative or the "bottom up" approach) at first each point is a cluster. Then in each step we merge two nearest points as the new cluster. We continue the process of merging until we reached the best clustering (to our criteria) and then we stop the clustering (if we do not stop this process we end up with one cluster).

The most advantage of this algorithm is that we do not have to know the number of clusters before we run the algorithm and also we stop the clustering when clustering is not going to go better (by calculating the inter and intra distance of points on each clusters in
Figure 6: SIFT points clustered by $k$-means algorithm

each step). The result of this algorithm which is shown in figure 7 is acceptable and worths to use it in our detection part.

Figure 7: SIFT points clustered by hierarchical algorithm and region of each cluster

The criteria that we used to stop the hierarchical clustering is so naïve. We stop the algorithm whenever the cluster with maximum point has reached to the 20% of all the points we stop the algorithm. Also we get rid of the clusters which has less than 5 points in them, because we found out around the objects always there are more than 5 points. These are the numbers which we reached in the process of trial and error and resulted the best in our algorithm.
3.3 Step 3: Making the regions more accurate

After make a rectangle around the clusters we should make the regions more and more accurate. The algorithm that we started to implement was the Non-Maximum Suppression [6]. Due to the lack of time we could not get a good result in this part, but, it should be done in the future to make the borders more accurate.

3.4 Step 4: Labeling the objects

In this part we had to send the regions to a well-known CNN to label each region with a confidence. Among all the CNNs we found OverFeat [7] the most comfortable CNN to work with. OverFeat lists some labels with different confidences. In this project we just use the label with the highest confidence. Also if the confidence for a region is less than 0.2 we assume that region as a false positive and remove that region.

4 Implementation and Results

In this part we will list the details of the implementation and the show the results of our algorithm.

4.1 How to Run the Code

All of our code and result picture of our code is published in our project GitHub page.

Instructions to run the code:
The code is run on PASCAL VOC and assumes that it follows the corresponding devkit provided. So, before running the code,

1. Download and un-tar the PASCAL2007 devkit to the root folder i.e., bdoosti-vavula-final/ and you would find a folder named VOCdevkit in the root folder.

2. The code is already made for you to run but in order to remake, run make clean; make in order to generate the a2 binary.

3. Execution: ./a2 train.txt 0.5 20 for which you could find the results in the folder VOCdevkit/results/VOC2007/Segmentation/

4. For reference, the outputs generated for classes car, dog and aeroplane have been made available for viewing in the Our_Code/Results/ folder.

4.2 PASCAL VOC dataset

The PASCAL VOC is a supervised learning dataset for object detection and segmentation. It has been built for the PASCAL challenge. The goal of this challenge is to recognize objects from a number of visual object classes in realistic scenes (i.e. not pre-segmented objects) [2]. The twenty object classes that have been selected are:

- Person: person
- Animal: bird, cat, cow, dog, horse, sheep
- Vehicle: aeroplane, bicycle, boat, bus, car, motorbike, train
- Indoor: bottle, chair, dining table, potted plant, sofa, tv/monitor
4.3 Results

We run our algorithm on PASCAL VOC 2007 (because it had less pictures than PACAL VOC 2012). We tried hard to run the PASCAL VOC evaluator to get a quantitative result (like percentage of objects correctly localized) but we couldn’t run their code on our result and we had to subsample the tested images (20 images from three categories of cars, airplanes and dogs) and calculate the evaluator for each image by hand.

4.4 Evaluation

For the evaluation of our results we calculated the intersection of area detected by our algorithm and the ground truth area over union of area detected by our algorithm and the ground truth area. In other words

\[ r = \frac{\text{detected area} \cap \text{ground truth area}}{\text{detected area} \cup \text{ground truth area}} \]

It is evident that the higher \( r \) means the better the algorithm worked. As it is shown in the figure 4.4 the results of the algorithm on the dog category was acceptable and we calculated the ratio for the dog detected region and we got 61% which is a good result. The airplane category is the next category that we calculated the result. The airplane ratio was 38% and at last the category that our algorithm did not result well was the car category in which we got 19% accuracy of the regions.

4.5 The Advantages and Disadvantages of Our Algorithm Compared to Faster R-CNN

Although the Faster R-CNN’s regions are more accurate than ours and also because of being sensitive to the details we may have some false positive regions, our algorithm has some advantage to the Faster R-CNN.

First of all, this algorithm is faster than R-CNN (when you run both algorithms on CPU). Since the Faster R-CNN blindly does the all the filter convolution to the image and get the result from many layers of the CNN it takes a lot of time to perform this task. Also this blindly working makes their algorithm a black-box and this means that before running the algorithm nobody knows what will happen and how to improve the result for that image. But the process in our algorithm is completely clear and the flaws of each step are known and we know that what will happen for the image before running the algorithm and we know why that happen and how to improve it. Another issue in R-CNN is that we cannot be very certain to detect all the objects. As we see in the figure 4.5 our algorithm can be tuned to detect as much as objects we want. At last we should mention that both algorithms failed on detecting super macro images.

5 Future works

There are still many things that we should work on and make the algorithm more robust. First of all, we should implement the Non-Maximum Suppression algorithm to make our

\footnote{The First and second row images are adapted from http://www.geek.com/news/googles-driverless-cars-get-a-license-in-nevada-1488371/ and https://www.flickr.com/photos/41125886N03/7035526613}
regions more accurate. Our detection of the objects are now good but the not accurate regions lowers the ratio of our algorithm.

Also the stopping point of the algorithm still has some problem. As it is seen in the airplane category in some examples the algorithm ended up with two different part of the airplane and if it merged these two parts the ratio become very good.

References


Figure 8: The results of three classes of PASCAL VOC 2007 datasets: Dogs (left), Airplanes (Center) and Cars (right).
Figure 9: The results of our algorithm compared to the results of Faster R-CNN. The first row is an example of a lot of false negative of Faster R-CNN which much more than ours. Although their region is more accurate. The second row images shows an example that we detected almost every objects but Faster R-CNN did not. And the third row is an example with a lot of details that our algorithm failed and Faster R-CNN also did not do well. The different colors around the objects means different object detected not different label.