Real-time Object Detection
CS 229 Course Project

Zibo Gong[1], Tianchang He[1], and Ziyi Yang[1]

[1]Department of Electrical Engineering, Stanford University

December 17, 2016

Abstract

Objection detection is a key problem in computer vision. We report our work on object detection using neural network and other computer vision features. We use Faster Region-based Convolutional Neural Network method (Faster R-CNN) for detection and then match the object with features from both neural network and features like histograms of gradients (HoG). We are able to achieve real-time performance and satisfactory matching results.

1 Introduction

Object detection is a challenging and exciting task in Computer Vision. Detection can be difficult since there are all kinds of variations in orientation, lighting, background and occlusion that can result in completely different images of the very same object. Now with the advance of deep learning and neural network, we can finally tackle such problems without coming up with various heuristics real-time.

We installed and trained the Faster R-CNN model 2 on Caffe deep learning framework. The Faster RCNN is a region-based detection neural networks method. We firstly used a region proposal network (RPN) to generate detection proposals. Then we employed the same network structure to Fast R-CNN to classify the object and modified the bounding box. Furthermore we extracted feature from object detected via algorithm developed by us. At last we matched object detected with ones stored in database.

2 Related Work

In 2012, Krizhevsky and et al.[1] trained a deep convolutional neural network to classify images in LSVRC-2010 ImageNet into 1000 kinds of classes with much better precision than previous work, which marked the beginning of usage of deep learning in computer vision. In 2014, Jia and et al.[2] created a clean and modifiable deep learning framework: Caffe.

In 2015, Ren et al.[3] proposed faster Region Proposal Network. The network shared convolutional features through the image, which led to almost cost-free regional proposals. Besides faster R-CNN, there are many other approaches to improve CNN's performance. More recent approaches such like YOLO [4] and SSD [5] directly put the whole image into the neural network and get the predicted boxes with score. Their running time is further reduced compared to faster R-CNN. You only look once (YOLO) applies a single neural network to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. Single Shot Multi-Box Detector(SSD) discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location.

3 Dataset and Features

For general object detection, we used SUN2012 database by MIT SAIL lab, which contains images with objects labeled by polygons, as shown in Fig.1. For each category of object, we extracted the

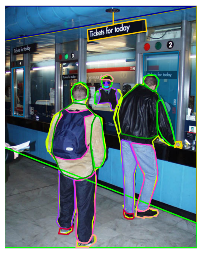

Figure 1: One database example.

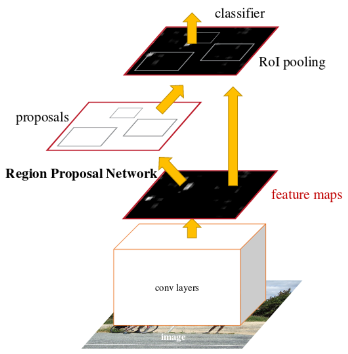

Figure 2: Illustration of Faster R-CNN training process.

bounding box for object detection and further classification tasks. For training, we divided the images of 3 categories to build up database. Also, we split the whole dataset into training set (90%) and testing set (10%). What's more, we tested out detecting automobiles in different perspective. The images of cars are from the KITTI database, which contains about 14,000 images. We chose the fully visible and partly occluded automobiles for training and testing. The KITTI database also provides the observation angle of each object. We used them to label the perspective (front, back, left, right) of automobiles.

4 Methods

4.1 Faster Region-based Convolutional Neural Network

We trained the Faster Region-based Convolutional Neural Network (Faster R-CNN) model on Caffe deep learning framework by Python language. The Faster R-CNN is a region based detection method. It firstly used a region proposal network (RPN) to generate detection proposals, then used the same network structure as Fast R-CNN to classify object and modify the bounding box. The strategy is visualized in 2.

The training strategy of this model was as follows: First, we trained the RPN end-to-end by back-propagation and stochastic gradient descent. We chose the ZF and VGG16 net to extract features for the RPN and all the layers are initialized by a pre-trained model for ImageNet classification. We chose the learning rate 0.0001 for 40k mini-batches, a momentum of 0.9 and a weight decay of 0.0005. Second, we trained a separate detection network by Fast R-CNN using the proposals generated by the step 1 RPN for 20k mini-batches. This detection network was also initialized by a pre-trained model. Now the two networks did not share convolutional layers. Third, we used the detector network to initialize RPN traning for 40k min-batches, but we fixed the shared convolutional layers. At last, we kept the shared convolutional layers fixed and fine-tune the unique layers of Fast R-CNN for 20 k mini-batches.

4.2 Conventional CV features

a) Color Histograms
After we found the bounding box that gave the outline of object, we could construct three histograms of the RGB values of all the pixels within the bounding box. The color of the object could be found combining the most frequent RGB values. Usually a clustering algorithm was applied to find the color, the color histogram is much more computationally efficient. The R, G and B value of color of object is the peak of each component histogram. Also since the image had already been segmented by the Faster R-CNN, the color histogram could achieve good performance with less computation expense.

b) Histogram of Oriented Gradients (HoG)
Histogram of Oriented Gradients (HoG) is a feature descriptor for object detection. It will build a histogram for localized portions of an image. The histogram is distributed along the angle of orientation (usually 8 directions) and the height is the magnitude of the gradient. In our project, usually the image of an object was divided into 16 bins (4 by 4) and the histogram of each bin was calculated. Then the total length of HOG feature vector would be $16 \times 8 = 128$.

c) Scale Invariant Feature Transform (SIFT) Descriptor
SIFT Descriptor is also a descriptor built from histogram of gradients, however, the SIFT descriptor has the advantage of being invariant to rotation, translation and resizing.
SIFT descriptor was used to extract keypoint from the image. We first applied Gaussian filter of different sizes and took their difference. The keypoints were local extrema across each layer of Difference of Gaussian (DoG). After determining a keypoint location and patch size, the dominant direction of gradient was decided and we rotated the patch to have its dominant direction aligned vertically. At last the same procedure of HoG could be carried out to assemble a 128-length feature vector.

4.3 Matching Algorithm

In our project, we used K-Nearest Neighbor (KNN) to match the extracted features with the database. The KNN algorithm is a basic unsupervised learning algorithm. It compares the incoming example with all the dataset and output the training example with the smallest distance from the example.
Note that here the distance function can be defined by multiple ways. The simplest way would be computing the Euclidean distance of the feature vectors. Another distance definition that is useful here is the Cosine distance, defined by

$$d(x_i, x_j) = 1 - \frac{x_i^T x_j}{|x_i||x_j|}. \tag{1}$$

In our project, we tried out both definitions of distances and use the one which works better and it is Cosine distance.

5 Results & Discussion

5.1 Object Detection

We successfully achieved detection of backpacks 3(a), towels 3(b), clocks, bottles, and automobile in different perspectives (front, back, left and right) 3(c). And the detection of automobile is with high detection precision, as sown in table 5.1:

Perspective	Average Precision
Front	81.57%
Back	86.40%
Left	87.56%
Right	79.31%

Table 1: Average precision of detecting cars in different perspective.

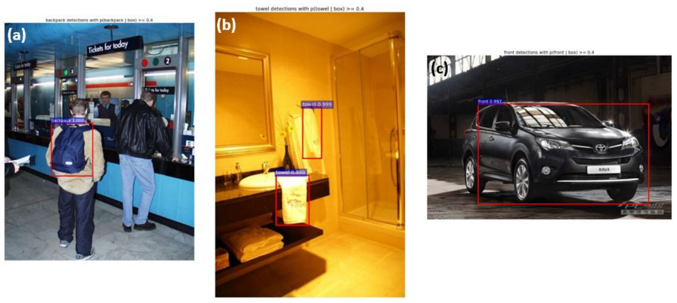

Figure 3: Object Detection.

In the poster session, we presented a demo of automobile detection. We took a video at parking lot and detected automobile in every frame. And detected automobile is bounded with box as shown in video. And the detection is in real-time.

5.2 Color Detection

We have successfully decided the color of object detected via algorithm mentioned above. Or to be more specific, the color of object in the bounding box. The result is shown in 4. 4[a] is the color detected, and 4[b] is the object. Judging from naked eyes, those two colors look very alike.

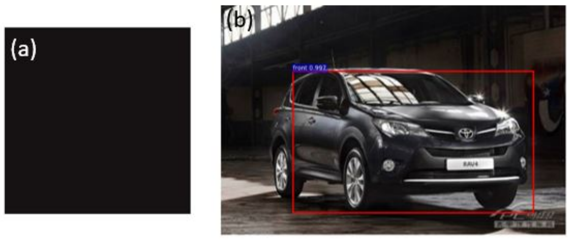

Figure 4: Color Detection.

5.3 Object Matching

First we picked the better distance definition used in kNN. We tested out two definitions of distance on a set of 400 test images of cars with k=10 in kNN. The features used here were RGB features and HoG features. The results are shown in Table 5.3. We can see that Euclidean distance gives better result, we then used Euclidean distance in further experimentation.

Then we tried out different combinations of features. We tested out the combinations on 1000 images of cars. The results are shown in Table 5.3. We can see that the fully connected layer is better than the HOG features. Also we find that using color as a supplementary feature would reduce the matching scope and improve the running time and matching precision. An example of matching result is shown in 5. Also, in the poster session, we showed in the video that after detection of automobile, we could successfully matched it with one car in our database (RAV 4 in our demo). And then researched corresponding model and color in car.com.

Distance Definition	Precision
Euclidean	70.5%
Cosine	66.3%

Table 2: Precision of different definition of distance.

Feature Combination	Average Precision
HoG	70.7%
HoG+Color	73.2%
Fully Connected Layer	77.6%
Fully Connected Layer + Color	78.1%

Table 3: Precision of different combinations of features.

5.4 Pipeline

The pipeline for our project goes as the following: The input image/video was fed into a trained neural network, which will output the object classification and bounding box. Then the features of the object (including the fully connected layer from CNN and other conventional CV features) are extracted and compared to the database. The final output is the closest match from the database.

Such a pipeline is not only able to tackle the general task of object detection and recognition, it can also be utilized by companies such as online shopping sites. Such companies have the man-power and computational power to build it into a mobile application, which can take real time image or video and match it on the server. It can also benefit from the fact that they have huge databases on objects.

6 Conclusion & Future Work

In conclusion, we trained Faster R-CNN to detect objects in real-time. And then we extracted features, for instance, color, HoG, SIFT descriptors, and result (the last layer of networks) given by faster R-CNN. Finally, we compared the object detected with those in our database and decided the matching one, based on the features extracted.

For future work, we can conduct more thorough test of our matching algorithm using larger dataset and more rigorous diagnostics of the algorithm. For example, we can plot the error rate vs the size of the dataset or plot the confusion table of the test set, etc.

We tried out feature combinations such as fully connected layer, RGB color and HOG, but more combinations of other features might be useful. Viable features including rectangle features and other features from the neural network. However, if we confine our attention to kNN, there are many other distance functions we can try out. Other distance definitions including the Manhattan distance, Histogram intersection distance and Chebyshev distance can be implemented with ease. Matching quality can be improved by better detection. We also recognized that the observation angle is very important for object matching. It's essential to do the "car face" alignment. We can apply the similar algorithm for face recognition in our project.

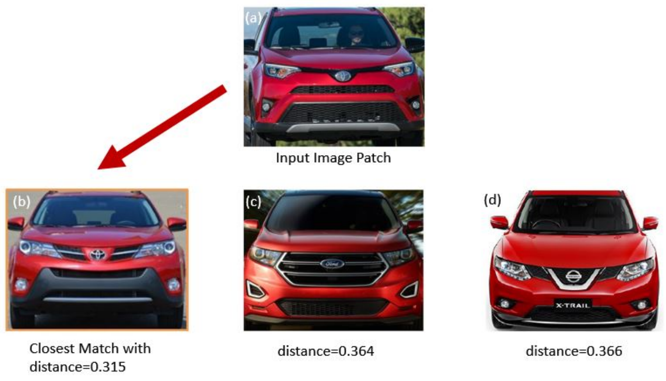

Figure 5: An example of matching result

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[2] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” *arXiv preprint arXiv:1408.5093*, 2014.

[3] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, pp. 91–99, 2015.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *arXiv preprint arXiv:1506.02640*, 2015.

[5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. Reed, “Ssd: Single shot multibox detector,” *arXiv preprint arXiv:1512.02325*, 2015.