Traffic Sign Detection Based on Faster R-CNN

Jiaxi Nie Yuan Cheng Rui Lan
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
{nie9, ycheng35, ruilan2}@illinois.edu

Abstract

Autonomous driving technology has been a key aspect of artificial intelligence throughout the past three decades and has successfully commercialized recently. Object detection has been on the center of the stage as a critical part of autonomous driving and other traffic-related applications. People have striven to detect and analyze road conditions to ensure safe and effective driving, focusing on recognizing pedestrians, automobiles, roads and traffic signs, etc. *Our project focuses on detecting traffic signs from provided images by using deep learning techniques. We perform Faster Region Based Convolutional Neural Network [1] (Faster R-CNN) to localize and recognize different traffic signs on The German Traffic Sign Detection Benchmark (GTSDB) dataset [2]. We show our Faster R-CNN is useful in detecting the traffic signs accurately and efficiently.*

1. Background

Thanks to the fast-growing Artificial Intelligence (AI) technology, many AI applications have been proposed on the stage to improve peoples lives in different aspects, and one of which is the autonomous driving technology. As one of the most useful and cutting-edge applications, leading technology companies try to invest and improve the self-driving car industry. Detecting and recognizing objects, such as traffic sign, are always one of the most crucial parts of this field. Also, to enable the real-time object detection, the algorithm must be highly efficient during the inference. However, traditional object detection algorithms, such as sliding window and support vector machine (SVM), requires extended testing time. The computational expensive of feature extraction and proposal generation in those traditional algorithms prevent them from applying to the self-driving car applications. The convolutional neural network (CNN) approaches, on the other hand, enable us to feed the raw images directly into the network which saves time during testing.

The CNN approaches of object detection algorithms come from two major groups: region proposal based algorithms like R-CNN [3], Fast R-CNN [4], and Faster R-CNN [1], and regression based algorithms like You Only Look Once (YOLO) [5] and Short MultiBox Detector (SSD) [6]. In this project, we mainly focus on region proposal based algorithms.

R-CNN. The Region-based convolutional neural networks (R-CNN) was first proposed in 2014. With three stages, the R-CNN algorithm first uses deep neural network method to solve object detection problem. The R-CNN model first generates region proposals using Selective Search and then feeds the proposals into different CNNs associated with region proposal to extract features. The features are classified by using SVMs. Although the R-CNN gives a good testing accuracy, the training and testing are expensive in respect of time.

Fast R-CNN. Later, in 2015, Fast R-CNN [4] comes to the stage. The Fast R-CNN algorithm [4] first uses a CNN to extract features of the whole input image. Then, the Selective Search method is applied on the extracted feature maps to generate proposals. The reshaped features of the proposals which called Region of Interest (RoI) are feed into serval fully connected layers to produce outputs. The outputs contain the classification score which indicates the class labels and the bounding boxes which represent the location of the objects. However, as the long running time of getting region proposals in Fast R-CNN, we need further improvements to enable real-time object detection.

Unlike R-CNN and Fast R-CNN, Faster R-CNN, which first proposed by Ren et al. at 2016 [1], develops intelligence in generating region proposal which solves the test-time computational bottleneck in the object detection algorithms. In this report, we show how can deep learning algorithm, specifically Faster R-CNN, can be applied to detect traffic signs.

In this project, we use the German traffic sign Detection benchmark [2] (GTSDB). In this dataset, there are 900 images with natural traffic scenes (600 training images and 300 evaluation images). For each image, there are several numbers of traffic signs (zero to six). The size of each image is 1360×800 pixels, which comes with annotations specifying the coordinates of the traffic signs and corresponding class. Sizes of traffic signs in this dataset vary from 16×16 to 128×128 pixels. There are total 43 types of traffic signs. We count the background as the 44th class, and thus, there are 44 classes in this dataset.

2. Methods

2.1. Faster R-CNN

The model we use in this project is Faster R-CNN. There are four main parts in this model which are shown in Figure 1.

Base Net. The Faster R-CNN first contains a CNN which extracts the feature maps from the raw images. The input of the CNN is the training image, and the output of the CNN is the feature map corresponding to the input image. Ren *et al*. [1] implement the Zeiler and Fergus model [7] (ZF), which includes five convolutional layers, and the Simonyan and Zisserman model [8] (VGG-16), which contains 13 convolutional layers in their original paper. While recently, there are more robust CNN models, such as Resnet [9], available for this step which gives us the flexibility to choose a suitable one. However, we believe our dataset is relatively small compared to the datasets Faster R-CNN was initially designed for, e.g., PASCAL VOC [10] and MS COCO [11]. As a result, we construct smaller neural networks to prevent potential overfit. We train and test three different models in this project: (1) the simple 2-layer Fully Convolutional model (FCnet), (2) the Zeiler and Fergus model (ZFnet) [7], (3) the Simonyan and Zisserman model (VGG-16) [8]. In our experiments, the mean average precision (mAP) is used to compare the accuracy of these three different base net models.

Region Proposal Network. The next important part in Faster R-CNN [1] is the Region Proposal Network (RPN) (shown in Figure 2), which contains one fully convolutional layer. The RPN is also the most important and innovative part of Faster R-CNN algorithm [1]. The RPN generates different proposals which may contain the object to be detected by using four coordinates as a bounding box. Besides the proposals, the RPN also gives the information about the confidence of each proposal as two classification scores. Although some of the proposals could be needless, the RPN is capable of finding out most of the object regions we are interested. Instead of the Selective Search method used by previous object detection algorithms (Fast R-CNN [4] and R-CNN [3]), the Faster R-CNN [1] uses RPN to learn how to generate proposals which increase the accuracy and efficiency of the overall detection process.

As shown in Figure 1, the input of the RPN is the feature maps generated by the base net, and the output of the RPN are positive proposals. Since the traffic signs are commonly

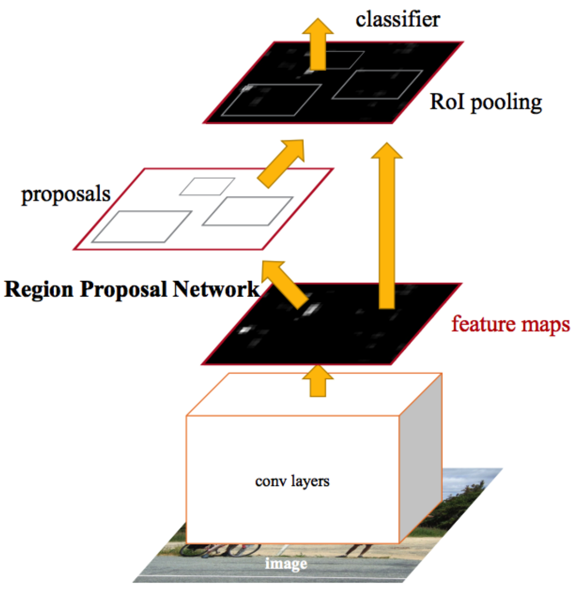

Figure 1. Faster R-CNN network [1].

in round or rectangular shape, we parameterize those proposals as boxes to locate different traffic signs in our RPN, and those reference boxes are called anchors in the original Faster R-CNN paper [1]. By comparing these anchors with the ground truth boxes, we get the positive proposals which may contain the traffic sign objects. For the detailed implementation, in the original Faster R-CNN paper, Ren *et al*. [1] choose the anchor sizes as [16, 32, and 64]. As there are very small and vague traffic signs in the GTSDB dataset [2], we add 8 as one of the anchor sizes to accommodate the nature of our dataset. We test both sets of anchor size [16, 32, 64] and [8, 16, 32, 64] to compare their accuracy. In addition, the shape of the boxes are in different scales which is determined by anchor ratio. We choose the anchor ratios to be [1:1, 1:2, 2:1] and [1:1] only to compare the accuracy. We believe the variance of anchor ratios can improve the performance since it can capture different traffic signs in different situations. We test the stride size to 4, 8, 16 to slide the anchor on the feature maps. The accuracy of different anchor configurations are tested and analyzed in this project.

To train RPN, the multi-task loss function is used in our project. As shown in Figure 2, the output of RPN contains two parts, i.e. classification loss and regression loss. The classification loss is a log loss of two classes which encodes the confidence of whether the region proposal is an object or not. The regression loss is smooth L1 loss which represents the four coordinates of the proposed bounding box. The loss function in a formal mathematical format is shown below [1]:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i L_{reg}(t_i, t_i^*) \quad (1)$$

Here, p_i is 1 when the RPN predicts the bounding box contains object and is 0 when the RPN predicts the bounding box contains no objects. p_i^* is the ground truth label. t_i and t_i^* devotes the location of the bounding boxes which are transformations of the actual four coordinates. λ is the balancing parameter which controls the weights of two losses. Specifically, the following equations shows how the relationship between t and the actual four coordinates, where x, y, w, and h represents the center of the bounding box and its height and width [1].

$$\begin{aligned} &t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a, \\ &t_w = \log(w/w_a), \quad t_h = \log(h/h_a), \\ &t_x^* = (x^* - x_a)/w_a, \quad t_y = (y^* - y_a)/h_a, \\ &t_w^* = \log(w^*/w_a), \quad t_h = \log(h^*/h_a), \end{aligned} \quad (2)$$

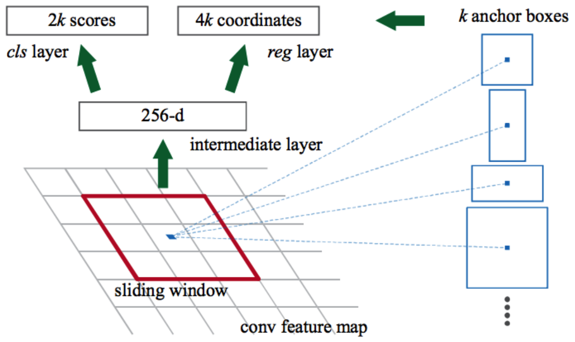

Figure 2. Region Proposal Network (RPN) in Faster R-CNN [1].

Region of Interest Pooling. The third part of Faster R-CNN is the Region of Interest (RoI) pooling algorithm. The purpose of the RoI pooling is to standardize different anchors in the same size, since the RPN generate several proposals with different sizes. To feed the region proposals into last convolutional net for classification, we have to standardize all the positive proposals into a fixed size. The adjustment of the proposals includes cropping and resizing. The modified features are sent to the classifier, which contains two fully-connected layers to generate the final classification decision.

Classifier. Lastly, there is a classifier which contains two separate fully-connected layers. Specifically, one is for classification of traffic sign type denoted as class number and confidence score, another is for regression of the box location denoted as 4 coordinates (x,y,w,h). This classifier takes the RoI pooling results as the input and generates predicted boxes and make the classification.

2.2. Dataset

The original model of Faster R-CNN is trained on PASCAL VOC 2007 dataset [10] and MS COCO dataset [11]. The VOC 2007 dataset contains three main sections: JPEG Images, annotation, and Image Set/Main. As the dataset we are going to use, GTSDB [2], has a different format, we further preprocess to make the dataset fit in the Faster R-CNN model [1]. The preprocessing we use includes resizing, format converting, and labeling. Furthermore, other data preprocessing techniques, such as normalization and augmentation, also be tried in this project to get a better generative accuracy.

2.3. Implementation

Although the original Faster R-CNN [1] is based on Caffe, there are a lot reproduced versions using other frameworks such as PyTorch and Tensorflow. In this project, we implement our Faster R-CNN based on a general open source Faster R-CNN code [12] in Keras and make changes to accommodate our dataset and enhance the performance. The computational resource is another important factor when doing a deep learning project. The Faster R-CNN model needs a lot of computational resources which requires GPU computing. In this project we use online GPU servers such as Google Could and Floydhub to train our models.

3. Experiments

As our traffic sign detection system is intended to be deployed on autopilot cars and perform real-time detection and classification within a very small amount of time with fairly high accuracy, we develop performance and accuracy-oriented experiments on the Faster R-CNN model Ren *et al.* [1]. Recognizing the nature of our project, detecting traffic signs, we adapt and experiment on the Faster R-CNN model Ren *et al.* [1] aiming to produce the highest recall on the detection and classification of the traffic signs. The image dataset, GTSDB dataset [2], contains 900 images and 43 labels. As described in the previous section, we experiment on various sets of configurations and models to explore and determine the performance on the dataset, which includes the detection rate and the speed of training and testing processes. We evaluate the performance of our models using mean Average Percentage (mAP) over all classes.

Our experiments of configurations are deployed in all three models, the simple 2-layer Fully Convolutional Net (FCnet), ZFnet and VGG-16. The mAP results comparing these three base networks are shown in Table 1. The stride of RPN is a crucial factor that affect the accuracy and speed of detection. Practically, varying the RPN stride result in drastic change in performance, in terms of both training and testing time. As we recognize that this stride also depends on the model structure and affect the number of anchors we produce, we vary the RPN stride from 4 to 16 while slightly changing the model structure by adding or omitting Max-Pooling layers correspondingly. The mAP results comparing three different stride values, i.e. 4, 8, and 16, are also shown in Table 1.

As opposed to the original Faster R-CNN training and used on PASCAL VOC [10] and MS COCO [11] for general object detection, our experiment decisions are made upon the fact that most traffic signs are relatively small in a scene and tend to have a square of round shape. Adapted from the original Faster R-CNN paper *et al*. [1], we vary the anchor size and anchor ratio to better accommodate our data. Specifically, we compare the the use of anchors with size 8 to the original settings, as well as the abandon of anchor ratios of 1:2 and 2:1, which would speed up the training and testing processes by decreasing the number of anchors. The mAP results comparing different anchor sizes and anchor ratios are shown in Table 1.

As we intend to facilitate the performance of Traffic Sign Detection system, we show that accuracy is preserved and we can detect and classify traffic signs with a fairly high recall.

In Figure 3, we show a number of testing images with bounding boxed drawn to mark our detection of traffic signs and predicted class. The probability of the traffic sign belonging to the predicted class is also labeled by the bounding box.

4. Discussion

As we can see from the results in the previous section, our system can detect and classify traffic signs with a fairly high accuracy in certain configuration settings. In fact, among all experiments, the one with VGG-16, RPN stride of 8, anchor ratio of [1:1,1:2,2:1], and anchor size of [8, 16, 32, 64] outperforms other models in terms of accuracy, as evidenced by its mAP. This result is as expected since VGG-16 is a deep convolutional network that captures features in a much detailed fashion compared to FCnet and ZFnet. VGG-16 demonstrates this effect in our experiments, as evidenced by the second picture in Figure 3 where we are able to detect the signs in a dark lighting environment.

On the contrary, we are not able to detect any traffic signs in such lighting conditions in experiments where we have FCnet or ZFnet. Setting RPN stride to 8 and include 8 in the proposed anchor sizes achieve the tradeoff between accuracy and performance, in that most traffic signs tend to have a small size but larger than 8 pixels, while in the meantime we reduce the number of anchors proposed by half compared to the original setting. As we see in the picture on the top of Figure 3, we would not be able to detect the speed

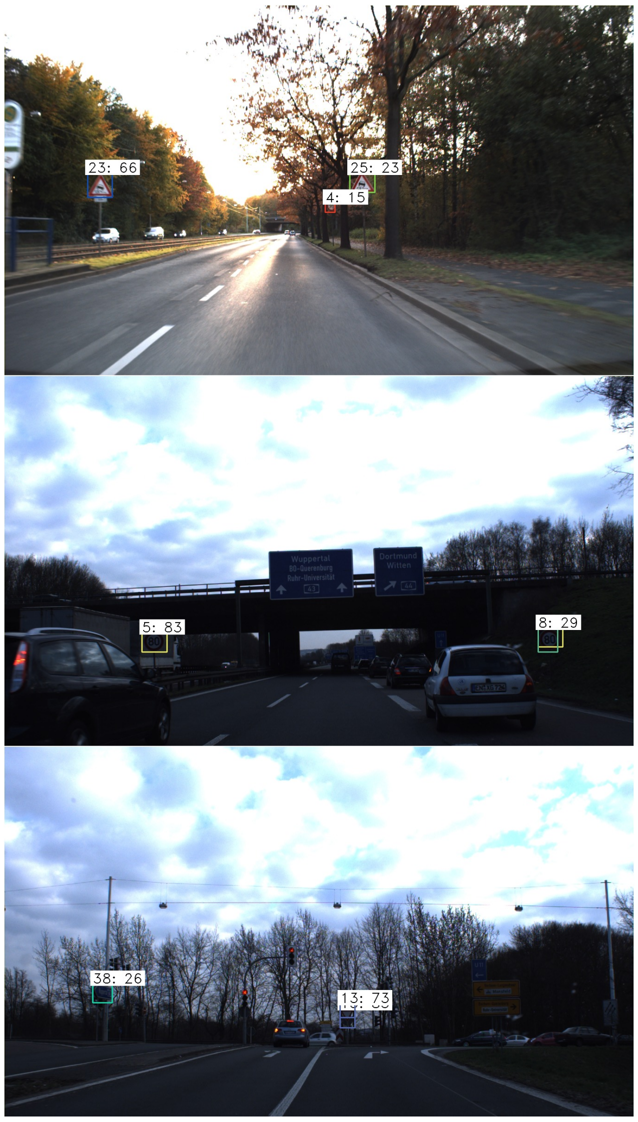

Figure 3. Testing results with bounding box, predicted class and probability using VGG-16 with RPN stride of 8, anchor size [8, 16, 32, 64], and anchor ratio [1:1].

limit 70 sign if we only have anchor size larger than 8. The reduction in anchor ratio selections also proves to speed up the training and testing process since it only proposes a third number of anchors as in the original paper, and in the meantime, our detection accuracy slightly decreases compared to the experiment where we use the full set of anchor ratios, i.e. 1-by-1, 1-by-2, 2-by-1.

Nevertheless, this configuration does produce false positive and incorrect classifications sometimes, such as the detection in the bottom picture in Figure 3. As a matter of fact, many detections in our testing images tend to be classified as keep right sign, which is mostly blue. This is a result of insufficient training, as our model only learns to clas-

Table 1. Detection results of Faster R-CNN using different CNN models (FCnet, ZFnet, VGG-16), with stride size (4,8,16), anchor size [8, 16, 32, 64], and anchor ratio [1:1, 1:2, 2:1].

anchor size	anchor ratio	RPN stride	CNN model	mAP(%)
[8,16,32,64]	[1:1]	16	VGG-16	80.50
[8,16,32,64]	[1:1]	4	ZFnet	91.76
[16,32,64]	[1:1]	16	VGG-16	95.30
[16,32,64]	[1:1]	8	ZFnet	85.59
[16,32,64]	[1:1]	8	VGG-16	**96.42**
[8,16,32,64]	[1:1]	8	ZFnet	81.66
[8,16,32,64]	[1:1]	8	VGG-16	90.48
[16,32,64]	[1:1]	4	FCnet	83.33
[16,32,64]	[1:1]	4	ZFnet	90.40
[8,16,32,64]	[1:1,1:2,2:1]	8	FCnet	93.32
[8,16,32,64]	[1:1,1:2,2:1]	8	ZFnet	94.11
[8,16,32,64]	[1:1,1:2,2:1]	8	VGG-16	**98.17**

sify based on blue color for specific signs. This is also expected since our training process is implemented on Google Cloud, where we only have only one GPU to carry out all experiments. Due to this limit in computational resources and time, we are only able to train our models in a limited number of epoch, 10 the most. Provided that we had more time and powerful computational resources, we expect our models to yield better detection accuracy and performance given sufficient training to our deep net.

Although our implementation can detect and classify more than % traffic signs correctly in our test data, the real-life situation can vary based on the road constructions, human activities or the weather changes. Therefore, it is necessary to further enhance our system such that it can be suitable for detection in more circumstances. Further improvement can be made on the RPN model, as it is one fully convolutional layer now. We can test different nets and different parameters in the RPN model so the anchors can be much more precise.

Furthermore, a more flexible and straightforward model Mask R-CNN [13] is proposed earlier to perform the object detection on the instance level. By adding the component pixel to pixel alignment on the base of RPN and RoI pooling, it can effectively reduce the complexity and increase the accuracy. We believe that we can try to apply this technique to assist our further improvements.

5. Conclusion

From the insights of R-CNN and Fast R-CNN, we build the model of Faster R-CNN and test our implementation based traffic sign detection. By using RPN and shared convolutional layers, it can generate region proposals nearly cost-free which enable the real-time object detection. By using the deeper convolutional neural network, such as ZF net and VGG-16, our result achieves high accuracy for the sign detection and classification. For the VGG-16 net, the mAP can reach 98.1%. However, there are still some vague signs which cannot be detected by our system. Considering the various situations in real life, we need to improve our design to achieve high accuracy as well as faster speed. We believe our method can be applied to the encouraging autonomous driving technology field.

6. Acknowledgements

This project is submitted as the final project for ECE 544NA (Pattern Recognition) at the University of Illinois at Urbana-Champaign. We thank Prof. Alexander Schwing for his guidance and help. We thank TAs Raymond Yeh, Teck Yian Lim, and Safa Messaoud for their work during the semester. As this project is trained on Google Cloud GPUs, we also want to express our gratitude to the course staff for the Google Cloud credits.

References

[1] 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 28* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 91–99, Curran Associates, Inc., 2015.

[2] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark," in *International Joint Conference on Neural Networks*, no. 1288, 2013.

[3] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation.," *CoRR*, vol. abs/1311.2524, 2013.

[4] R. Girshick, "Fast r-cnn," in *International Conference on Computer Vision (ICCV)*, 2015.

[5] J. Redmon and A. Farhadi, “Yolo9000: Better, faster, stronger,” *arXiv preprint arXiv:1612.08242*, 2016.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single shot multibox detector,” in *ECCV*, 2016.

[7] M. D. Zeiler and R. Fergus, *Visualizing and Understanding Convolutional Networks*, pp. 818–833. Cham: Springer International Publishing, 2014.

[8] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *ArXiv e-prints*, Sept. 2014.

[9] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778, 2016.

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results.” http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[11] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollr, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European Conference on Computer Vision (ECCV)*, (Zrich), 2014. Oral.

[12] Y. Henon, “keras-frcnn.” https://github.com/yhenon/keras-frcnn, 2017.

[13] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” *arXiv preprint arXiv:1703.06870*, 2017.