Traffic Sign Detection with Convolutional Neural Networks

Evan Peng[1(✉)], Feng Chen[2], and Xinkai Song[2]

[1] Taipei American School, Taipei, Taiwan
evanpl7ll2l4l@tas.tw
[2] Department of Automation, Tsinghua University, Beijing, China
chenfeng@mail.tsinghua.edu.cn, sxk0l8@l63.com

Abstract. This research focuses on improving traffic sign detection in cars using Convoluted Neural Networks (CNN), with images from the German Traffic Sign database. In order to generate more accurate detection results of traffic signs, different algorithms were used to generate the detection and classification tasks. The Faster Region Based Convolutional Neural Network (Faster R-CNN) and You Only Look Once networks were compared beforehand to determine which CNN to use. The Faster R-CNN was decided upon based off of previous results, then used to generate the classification and detection tasks. Pre-training weights were made using Caffe based off of the German Traffic Sign Recognition Benchmark database. Different methods of generation of training data were then used and compared. The Faster R-CNN network was used to create a classification task based off the images from the self-generated training images, which was tested against the German Traffic Sign Detection Benchmark database.

Keywords: Traffic sign detection · Traffic sign classification · Automation · R-CNN · Convolutional · Neural networks · Faster R-CNN · Fast R-CNN

1 Introduction

The automation of cars is becoming a more and more prominent subject in the current progression of artificial intelligence today. Companies such as Google, Uber, and Tesla have all announced plans of developing autonomous cars, with Tesla already having fully function autonomation capabilities on all 3 of their current models [1], available for consumers today. Laws are also being shifted in order to accommodate autonomous cars. In order to ensure that autonomous vehicles do not endanger the lives of humans on the road, the accurate detection and classification of traffic signs is vital.

For this research paper, in pertinence to the dataset used, traffic sign detection will be defined as the detection and classification of traffic signs, while traffic sign classification will be defined as solely the classification of traffic signs. Ensuring that the divide between the two is defined is necessary for this paper so results for detection do not get confused with the results of classification. Traffic sign detection is done by determining and detecting the area in which the traffic sign could potentially be located. Afterwards, the area is then classified to determine what the traffic sign represents. In

F. Sun et al. (Eds.): ICCSIP 2016, CCIS 710, pp. 214–224, 2017.
DOI: 10.1007/978-981-10-5230-9_24

this paper, 43 various traffic signs were used from the German Traffic Sign Recognition Benchmark (GTSRB) [2] for the classification of the traffic signs (Fig. 1).

Fig. 1. All 43 GTSRB signs

These signs present difficulty in the detection and classification of such images as many of them are quite similar to each other. Difficulty is further increased as the signs are detected from further distances, which would result in a higher chance of confusing the signs with other signs during the detection and classification task. Lighting and weather conditions can also affect the detection and classification of such images, which again may result in either mis-classification or non-detection such signs. Speed is also an issue that must be acknowledged in the detection such traffic signs. When an autonomous car is operating under real time circumstances the car must be able to accurately detect and classify the traffic sign as soon as possible to minimize the risk of danger (Fig. 2).

Fig. 2. An example of difficult detection circumstances (the sign is designated by the red box) (Color figure online)

In order to account for the above issues, the neural network model that is used must be able to efficiently detect images while also having high classification accuracy and detection/classification speed rates. Convolutional Neural Networks (CNN) are the most widely-used model of neural networks in image-based detection. Over 50,000 various traffic sign images from the GTSRB of various noise levels were used to generate training data for the German Traffic Sign Detection Benchmark (GTSDB) [2] test. After examination and light testing of various versions of different CNN implementations, the Faster Region Based Convolutional Neural Network (Faster R-CNN) was decided upon and used to generate the classification and detection tasks. After having viewed the testing results from Tim Dettmers of various Graphics Processing Units (GPU) [3] and his analysis of hardware requirements for deep learning [4], a NVIDIA GTX 1070 8 GB GPU, Intel Core i5 6500 Skylake Processor, and 16 GB of RAM were used to perform the training, detection, and classification tasks.

2 Related Works

There are not many modern papers regarding the usage of neural networks besides for CNNs for the application of image based detection. This is due to the high performance that CNNs present when image based detection tasks are needed. The CNN is modeled after a study by D. H. Hubel and T.N Wiesel in their paper on the monkey striate cortex [5], in which Hubel and Wiesel found that the visual cortex of a monkey contains receptive fields that detect light in overlapping sub-regions. The usage of CNNs grew especially prominent in image based detection with the LeNet-5, a 7 level convolutional network by Yann LeCun et al. [6], with focus around the detection of handwritten digits. On the Mixed National Institute of Standards and Technology (MNIST) database, a CNN based model resulted from Dan Ciregan et al. [7] achieved the highest accuracy to date, with a 99.77% accuracy rate.

Object Detection with Neural Networks. A CNN, GoogLeNet from Christian Szegedy et al. of Google [8], the University of North Carolina, and the University of Michigan scored a 43.9% detection rate with a classification error of 6.66% on the ImageNet Large Scale Visual Recognition Challenge database, which won the competition in 2014. The ImageNet Challenge database consisted of several millions of images of various objects and scenes. The GoogLeNet consisted of around 100 layers, going 22 layers deep with parameters, and 27 layers deep when counting pooling. The team behind GoogLeNet proposed a CNN called Inception which focused on the stacking of modules upon each other, similar to the Faster R-CNN method, which will be discussed later. The difference between the 2 being the Inception region classifier compared to the R-CNN classifier.

Traffic Sign Recognition with GTSRB. For the GTSRB database, work had been done by Dan Cireşan et al. [9] in which a 9 layer single CNN was proposed, with 7 hidden layers. It consisted of the input layer, 3 convolutional layers with another 3 max pooling layers, followed by 2 fully connected layers. For training data, they had just cropped down the images of testing data to be the same size, then plain-fed the images through the CNN. With just the usage of a CNN, they achieved a 98.73% recognition rate.

Using a Multi-Layer Perceptron (MLP) and a CNN they achieved a 99.15% recognition rate. However due to the usage of the MLP, the error rate increased by 4%.

Convolutional Neural Networks (CNN). CNNs operate similarly to a standard Neural Network (NN); however, they take in images better than a standard NN would be able to. A CNN takes in images and moves the images through the input layer, creating a 3D volume of neurons with dimensions of width, height, and color channels/depth, which is then output to the next 3D neuron layer. The input layer will have a 3rd dimension of value 3 (representing the 3 color channels of R, G, B). However further outputs from the hidden layers will have a different 3rd dimension value depending on the amount of filters used in the convolution layer (Fig. 3).

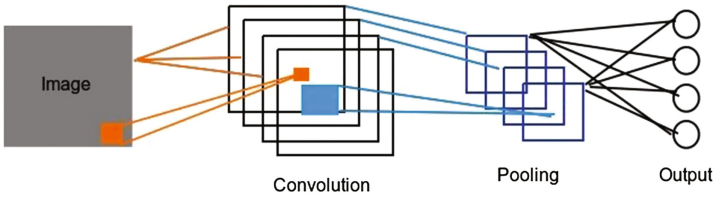

Fig. 3. Example CNN

An issue that is displayed with larger images is the issue of Fully Connected Networks, in which learning features which span the entire image would computationally take a very long time to process if every neuron was connected to every neuron. In order to solve this, for larger images Locally Connected Networks are used instead, and Fully Connected layers are placed at the end in order to compute the final class scores. This consists of only small subsets of the input to be sent to each neuron, and ensures that computation moves along speedily. This is more similar to how the visual cortex is used in which receptive fields only respond to certain areas rather than the entire area.

In order to process the image after the Convolutional layer, as per standard NN, an activation layer is needed. It is standard practice to use the Rectifier Linear Unit (ReLU) activation layer in NN and CNN due to not changing gradient when passing back through Back-Propagation (BPP) during training, which otherwise would result in the vanishing gradient problem. The vanishing gradient problem occurs when the activation layer has a changing gradient, such as with a sigmoid activation layer.

The Sigmoid function presents a changing gradient while, as shown in Fig. 4, the ReLU function does not. With the backpropagation algorithm, when the gradients are multiplied together, if the gradient is changing, the resulting product will begin to exponentially decrease, resulting in a 0 output, which will then result in a loss of accuracy/stagnation in cost. Thus, the ReLU function is the best choice for the CNN and most NN in general. There might follow many convolutional layers and ReLU layers followed by each other.

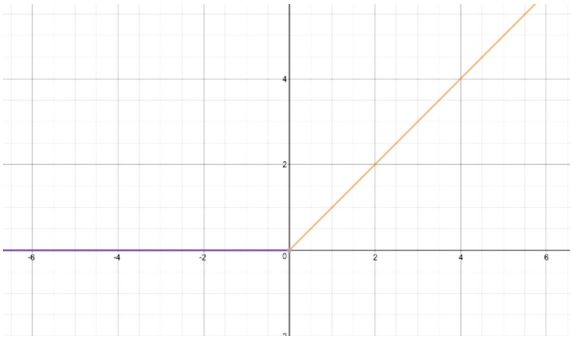

Fig. 4. ReLU function

After going through the variable amount of convolutional and ReLU layers, a Pooling layer will then follow to perform downsampling to create a volume with the same depth, but different height and width. After obtaining the features from the convolutional and ReLU layers, the features are pooled over a smaller region. In this case, the maximum value of each feature is taken to develop a single, smaller feature region. These pooling layers may also be inserted periodically between the convolutional and ReLU layers in larger CNN. Finally, a fully connected layer, which has full connections to all the activations from the previous layer, computes the final class scores and determines the resulting classification.

Faster Region Based CNN (Faster R-CNN). The Faster R-CNN differs from a standard CNN with its usage of Region Proposal Networks (RPN). RPNs are neural networks which focus on splitting the input image into around 2,000 different region proposals using Selective Search [10], which computes regions which have the highest probability of containing an object. The RPN in Faster R-CNN however does not use Selective Search to divide the regions, and is only present in R-CNN and Fast R-CNN. The region proposals are then warped into a single region, which is then fed through the CNN, resulting in a CNN known as Region Based Convolutional Neural Networks (R-CNN).

As shown in Fig. 5, the image first goes through the RPN and then through the CNN. While this resulted in more accurate results compared to a standard CNN, computation and detection time was very slow due to the multiple stages generated by the R-CNN with the multiple region proposals. Fast R-CNN was then created by Ross Girschick [11] in order to solve this issue by splitting the computation of the convolution layers between proposals and having the CNN run first and then generate the region proposals rather than have the Regions of Interest (RoI) feed into the CNN. Each object proposal from the CNN has a feature vector extracted by a RoI max pooling layer which then connects to the fully connected layers which outputs the classification outputs of a probability (softmax layer) and 4 values of the object class (bbox regressor) (Fig. 6).

Faster R-CNN was further created by Ross Girschick et al. [12] in order to account for the complex training and detection pipeline of Fast R-CNN. Faster R-CNN focused

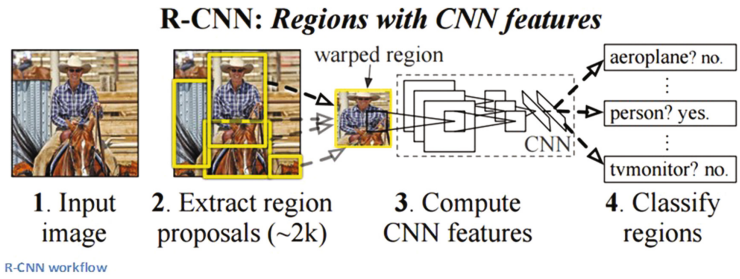

Fig. 5. Region based convolutional neural network [11]

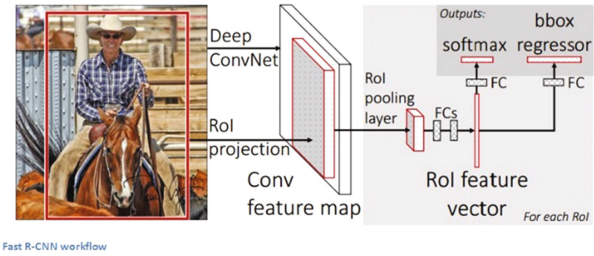

Fig. 6. Fast-RCNN [12]

on the usage of a CNN followed by a 4-step alternating training of the RPN and Fast-RCNN algorithms. The 4-step alternating training path consisted of first training an RPN off just the feature maps from the initial CNN, which already proves to be faster than training the RPN off the entire image. Using these proposals, a Fast R-CNN detection network is then trained, with pre-trained weights used to ensure learning occurs. In the third step, the RPN training is initialized based off the Fast R-CNN detections with any convolutional layers shared between the 2 fixed and only the unique RPN layers are adjusted. In the final step, the Fast R-CNN uses the proposals from the previous step RPN to initialize itself, in which the convolutional layers shared between the two are fixed once again, and unique layers are adjusted. Once again, with these two networks sharing the same convolutional layers to form a unified network, two outputs are generated with the classification of the object from a probability layer and the region of the object detected with the bbox regressor (Fig. 7).

You Only Look Once (YOLO). You Only Look Once is a CNN created by Joseph Redmon et al. [13] that works by resizing the image then running a convolutional network to increase training speed and detection speed, being a reported 100 times faster in training than Fast R-CNN as per the paper from Joseph Redmon. However, when tested against the Fast R-CNN, the Fast R-CNN managed to beat it heavily in terms of accuracy by 10.5% in terms of mean average precision regarding the detection and classification of objects on the VOC 2012 database. While the YOLO network is a lot faster than the Fast R-CNN, the speed of both is still quite fast, and the accuracy trade-off of YOLO does not seem worth it in the event of traffic signs detection, since a

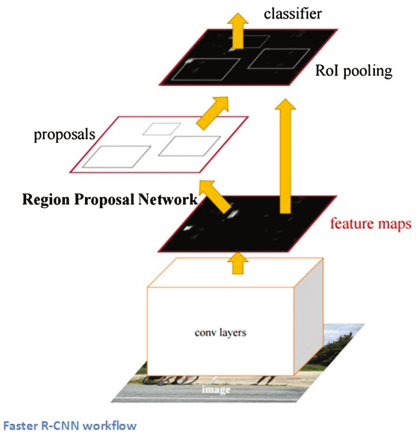

Fig. 7. Faster-RCNN [12]

car would still have some time to process the sign before it would need to act upon it, in which the Fast R-CNN time is still a fraction of the time needed.

3 Experiments

The Faster R-CNN was used to train the network. The network was trained, as previously mentioned, with a NVIDIA GTX 1070 8 GB GPU, Intel Core i5 6500 Skylake Processor, and 16 GB of RAM. The network was tested against the GTSDB database.

Training Process. First, the Faster R-CNN was trained in its original state with the pre-trained weights of the ImageNet weights, with only the GTSRB images as training. This resulted in low results, and a low mean average precision (mAP) of 24.19%.

This was probably a result of there being little background for the Faster R-CNN to detect, as the GTSRB images ranged from 34 $\times$ 34 to 127 $\times$ 127, and the GTSDB test images area 1360 $\times$ 800. A method was then used which just used the GTSDB training images. However, there were only 600 images, and a similar result was made when compared to the 300 testing images with a final mAP of 24.9%. This was probably due to the low amount of training images and lack of noise with the signs in the images that the GTSRB images have. In order to compensate for the both of this, a script was written which pasted GTSRB images over images from the Road/Lane Detection Evaluation 2013 (RLDE) by Jannick Fritsch and Tobias Kuehnl [13]. Annotation files were also generated. Figure 8 which displays an example training image generated is on the next page for easier viewing. Around 19,000 images were generated in total. This ensured that appropriate background was generated while also ensuring that noise is present for the traffic signs.

The Fast R-CNN and RPN networks for the 4 step alternating training process had 7 convolutional layers each to ensure higher accuracy. Pre trained weights were made

using Caffe off the GTSRB images to train off of instead of using the ImageNet weights for the Faster-RCNN algorithm.

Fig. 8. Example training image generated

Results and Efficiency. Across the 300 GTSDB testing images, a mAP of 77.21% was achieved. The 0 indexes present were not counted towards the mAP, as those classes were just not present in the GTSDB testing images. When testing with all 800 GTSDB images from the training and testing database, a mAP of 72.15% is achieved (Figs. 9 and 10).

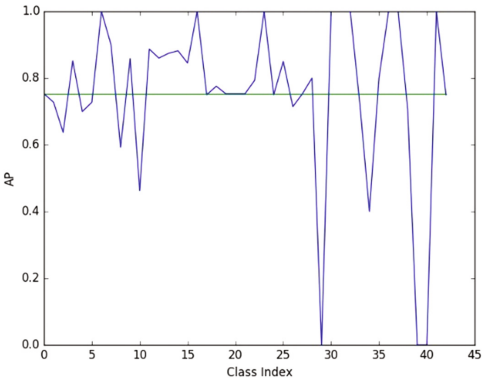

Fig. 9. 300 GTSDB images

Detection images on the GTSDB set will be shown on the last page for easier viewing (Fig. 11). In Fig. 12 the previous issue of distance and bad weather/lighting affecting the results of the detection is shown to have been resolved with this issue through the training data generation, as the 2 signs are placed quite a distance away from the car. While still being quite close together, the model was able to differentiate between these signs with a high accuracy of 99.7% and 94.5% accuracy for each sign respectively. While most of the detection results were quite high, there were some issues. Especially when considering the signs bluer in appearance, there tended to be an issue of accuracy. Figure 13 shows a blue sign with a lower accuracy present of 60.6%,

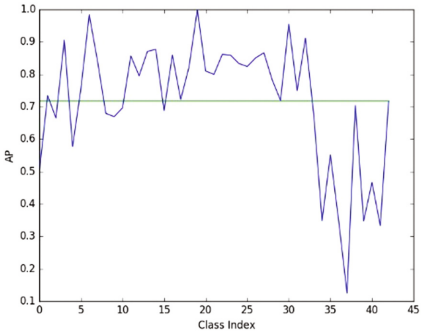

Fig. 10. 800 GTSDB images

showing how ineffective the model could be when detecting more "blue" signs. Occasionally the model also detected an area that was not a sign due to similar coloring (Fig. 14). This is an issue as this would result in a vehicle performing actions it should not be performing. This was potentially generated due to some of the signs in the RLDE dataset not being annotated as they were not generated from the GTSRB, and thus the model considered those background, resulting in awry detections. This was more prominent with images that displayed the color blue more.

The detection speed when the demo was run was a mere 0.2818 s per image on average, even with multiple signs. This demonstrates the high speed of the neural network, which is a goal that was set forward with the research of the application of Faster R-CNN on traffic signs. Earlier a potential issue that must be analyzed is the issue of speed, and YOLO was considered as a response to this issue. However, we see with the results from Faster R-CNN that speed is not an issue, and YOLO would only have resulted in potentially lower results.

Fig. 11. Red sign with 99.8% accuracy (Color figure online)

Fig. 12. Image of 2 signs with 99.7% and 94.5% accuracy respectively

Fig. 13. Image with blue sign detected with a lower 60.6% accuracy. (Color figure online)

Fig. 14. Failed detection with supposed accuracy of 89.6%

4 Conclusions

In this paper, we presented the usage of Faster R-CNN on the generation of detection and classification tasks. The detector proved to be very fast and had quite high results of around 90% accuracy for multiple images, but there were images where accuracy was quite low or detection was completely off due to potential issues regarding the

background of the RLDE dataset. In the future, road images should be used that do not include direct signs in the background that are not annotated. However, including images or colors that are similar to the color on signs in the background would ensure that the images which are similar to signs in the future are ignored.

Higher resolution pictures of Figs. 11, 12, 13 and 14 and additional detection images are available at http://imgur.com/a/MHlPY.

Acknowledgments and Institution. 我单位的置名格式: Department of Automation and CBICR Center, Tsinghua University; Beijing KLSBDPA Key Laboratory.

论文的资助项目: This work is supported in part by the National Natural Science Foundation of China under Grants 61671266, 61327902, and in part by the Research Project of Tsinghua University under Grant 20161080084, and in part by National High-tech Research and Development Plan under Grant 2015AA042306.

Acknowledgment must be given to the Tsinghua University Department of Automation, for supporting me and allowing me to work with them for this research, and Yan Qi for helping solve issues regarding implementation.

References

1. https://www.tesla.com/presskit/autopilot
2. Houben, S., Stalkamp, J., Salmen, J., Schilpsing, M., Igel, C.: Detection of traffic signs in real-world images: the German traffic sign detection benchmark. In: International Joint Conference on Neural Networks
3. Dettmers, T.: Which GPU for deep learning http://timdettmers.com/2014/08/14/which-gpu-for-deep-learning/
4. Dettmers, T.: A full hardware guide to deep learning http://timdettmers.com/2015/03/09/deep-learning-hardware-guide/
5. Hubel, D.H., Wiesel, T.N.: Receptive fields and functional architecture of monkey striate cortex. J. Physiol. **195**, 215–243 (1968)
6. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceeding of Institute of Electrical and Electronical Engineers (1998)
7. Ling, S., Wu, D., Li, X.: Learning deep and wide: a spectral method for learning deep networks. IEEE Trans. Neural Netw. Learn. Syst. **25**, 2303–2308 (2014)
8. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vincent, V., Andrew, R.:Going deeper with convolutions, computer vision foundation CVPR (2015)
9. Ciresan, D., Meier, U., Masci, J., Schmidhuber, J.: A committee of neural networks for traffic sign classification. In: Dalle Molle Institute for Artificial Intelligence (2011)
10. Uijlings, J.R.R., van de Sande, K.E.A.; Gevers, T., Smeulders, A.W.M.: Selective search for object recognition. In: IJCV Technical report (2012)
11. Girshick, R.: Fast R-CNN in arXiv:1504.08083
12. Girschik, R., Ren, S., He, K., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks arXiv:1506.01497
13. Redmon, J., Divvala, S., Girschik, R., Farhadi, A.: You only look once: unified, real-time object detection arXiv:1506.02640
14. Fritsch, J., Tobias, K.: Road/lane detection evaluation. In: Honda Research Institute Europe, Karisruhe Institute of Technology (2013)