Seminar: Technology of Autonomous Vehicles

Topic 3: Traffic Sign Detection and Classification

Mahmoud Maan, Hanju Chen

Contents

Abstract

This paper investigates traffic sign classification and target detection, a fundamental task in autonomous vehicle perception. Autonomous vehicles need to accurately detect traffic signs and accurately recognise their content in complex driving environments. In real traffic scenarios, different factors such as motion jitter and weather conditions interfere. The data needs to be pre-processed to highlight the image information and then trained with convolutional neural networks for classification. For object detection we used the YOLOv5 network for training. We obtained a classification accuracy of 97.41% and a detection accuracy of 98.02%. Finally, the two functional modules were combined to achieve the entire detection task.

1 Introduction

Autonomous vehicles [1] have become a hot topic in recent years as interest in self-driving cars has exploded with the rise of the smart urban mobility [2]. Self-driving vehicles have the potential to replace conventional transport, and the advancement and implementation of this technology may be able to reduce the number of road accidents, as well as cause new disruptions. So it needs to

be viewed and analysed rationally. The logical architecture of an autonomous vehicle can large be separated into a layer for perception (“see”), a layer of planning (“think”), and a layer for control of the actuators (“act”) [3]. An overview of such an architecture is shown in Figure 1.

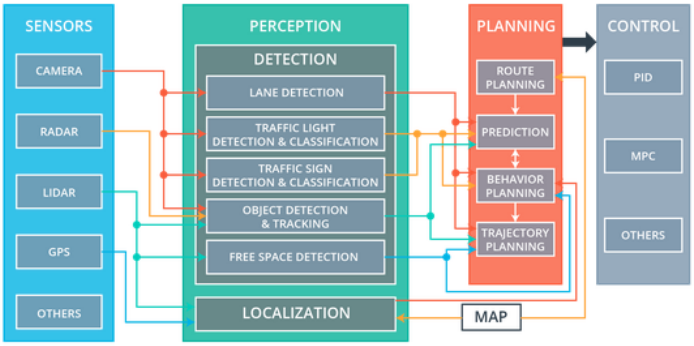


Figure 1: Autonomous vehicle functionality architecture. Source: Udacity Self-driving program.

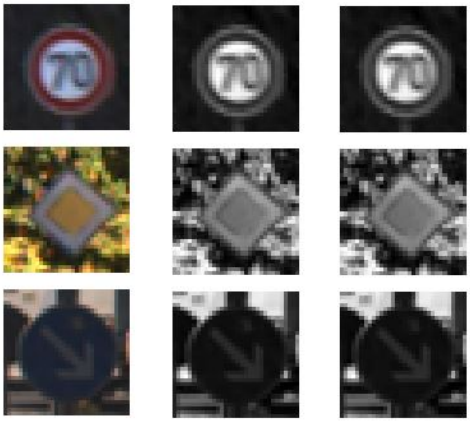
According to Figure 1, the autonomous driving architecture of a vehicle can be divided into four parts: scanning, perception, planning and control. These four parts cooperate with each other to control the operation of the vehicle. In this article, our work is mainly responsible for the sensing part. Vehicle perception is the acquisition of information from the outside world, which needs to be extracted and fed back to the vehicle in real time, which can involve a variety of factors such as traffic conditions, different weather and lighting conditions. Our main task is to detect and recognise traffic signs, as the vehicle needs to act on the information provided by them.
The work can be divided into two parts: 1) detection of traffic signs and 2) recognition of traffic sign information. This involves related aspects such as computer image processing, target detection algorithms and classification algorithms. Then the ’detection’ and ’classification’ functions are combined. This allows the vehicle to work in the sequence from ’detecting traffic signs’ to ’identifying traffic sign information’.

2 Methods and Algorithms

2.1 Classification

2.1.1 Dataset and preprocessing

In the classification part we trained the model using German Traffic Sign Recognition Benchmark (GTSRB) [4]. The total amount of data is more than 50,000 images of traffic signs with 43 different classes, the dataset is divided to sub-datasets with 39209 RGB images and 12630 RGB images for training and testing, respectively. All the images are in different shapes and sizes. In order to validate the model performance during training and to be able to tune the hyperparameters of the model, the training data is divided into sub-datasets for training and validation in proportion 70% and 30%, respectively, and to achieve the best possible results, we need to make the data clean before feeding it into the model.
We applied some of data preprocessing techniques: the first technique is image resize, in this technique we change the size of input image to make all images in one size as in Convolutional Neural Networks it is necessary that all input images are in the same size. The second technique is image color conversion: in this technique we convert the RGB colored image to grayscale image. It reduces training time of the model and also the complexity of the network. This means that the model will



(a) original (b) grayscale (c) normalize

Figure 2: samples after and before preprocessing.

work with just one input channel instead of three. The third technique is image normalization: in this technique, the process is to change the pixel intensity values which represented between 0 and 255 to values between 0 and 1. This allows faster convergence during training. The final technique is data shuffling: in this technique, we shuffle the data in order to achieve that every image creates an independent effect to the model without being biased by the other data images. The final image shape before feeding it into the model is (32x32x1). Figure 2 shows random samples from the data before and after applying the second and the third techniques.

2.1.2 Convolutional Neural Networks

We begin developing the deep neural network after preprocessing the input images. Convolutional Neural Networks also known as ConvNets or CNN's are used in this project. Because of its exceptional ability to extract important and distinctive features from images for each object in an image. CNNs are the go-to model for image processing tasks. They are most effective in face recognition, object detection, image classification and more. They are able to do so because of the convolutional layers, which employ convolutional operations.

A convolutional layer uses a filter to apply a convolutional operation and it is used to extract high-level features from the input images such as edges. The first convolutional layer is used to collect low-level features such as edges, color and orientation. Adding more layers allows the architecture to adapt to high-level features as well. By sliding the filter over the input input we perform a convolution. It is an element-wise matrix multiplication done at each location. As a result we get a feature map.

We took the inspiration of LeNet-5 [5] CNN architecture to build our classification model. We firstly trained the dataset on LeNet and we got results, which imply that the model is not performing effectively and it seems to have overfitted the data. In order to improve the model performance some additional modifications and fine-tuning techniques done on the network. The first modification for potentially improving the accuracy is increasing the number of filters of the convolutional layers. This helps the network extract more features from the images and it already resulted improved ac-

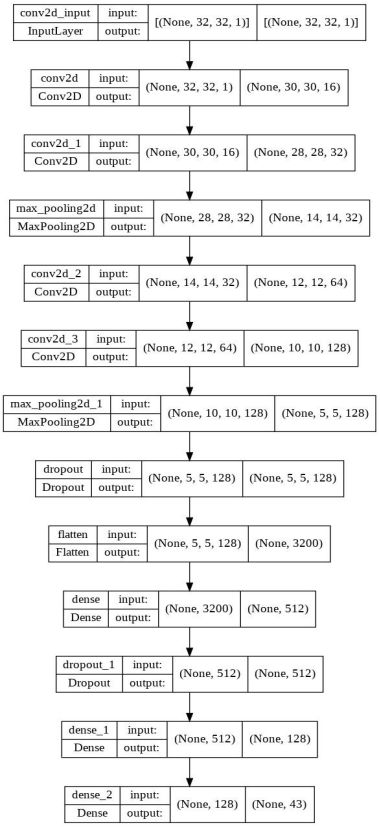


Figure 3: Self-modified CNN architecture.

curacy. Another modification that was effective in our case is adding two extra convolutional layers. More layers means more extracted features from images. This can lead to overfitting. However, to avoid that, we used a regularization technique called dropout [6]. Dropout layers are an effective way of preventing overfitting. It is a technique in which a group of neurons is ignored at random during training. We added another two dropout layers inside our model. Figure 3 and 4 show the final self-modified CNN architecture.

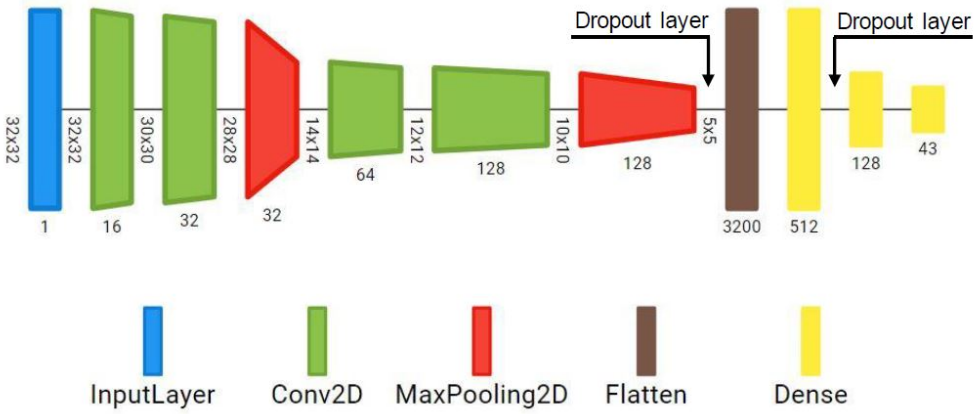


Figure 4: Self-modified CNN architecture. Generated by Net2Vis [7] .

2.1.3 Model evaluation

Model evaluation is the process of using different evaluation metrics to understand a machine learning model's performance as well as its strengths and weaknesses. Since the output of classification models is discrete, we need a metric that compares those discrete classes. Classification Metrics evaluate a model's performance and indicate how good or bad the classification is. There are four possible outcomes when making classification predictions. True positive (TP), true negative (TN), false positive (FP) and false negative (FN). True positive (TP) is when the model predicts a label for a class and the label actually does belong to that class. True negative (TN) is when the model predicts a label, which does not belong for a class and the label actually does not belong to that class. False positive (FP) is when the model predicts a label for a class and the label does not belong to that class. False negative (FN) is when the model predicts a label, which does not belong for a class and the label actually does belong to that class. We used accuracy to evaluate the classification model performance. Accuracy is one of the main metrics used to evaluate a classification model and it is defined as the percentage of correct predictions for the test data. It can be calculated by dividing the number of correct predictions by the total number of predictions as shown in equation (1).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

2.2 Detection

In the detection part, we used the German Traffic Sign Detection Benchmark (GTSDB) [8] to train the model. GTSDB consists of 900 image files with natural traffic scenes of 1360×800 size and text file gt.txt containing the ground truth for all traffic signs in the images.Unlike GTSRB, which divides the data into 43 classes, GTSDB divides the 43 different traffic signs into 4 classes(prohibitory, danger, other, mandatory), which is used to reduce the model training time. The training data was divided into

sub-datasets for training and validation respectively. The proportions were split into 80% and 20%. The training data consisted of image files, and corresponding txt files of the labels of the image files. In the raw data, gt.txt file contains all the ground truth information for the 900 images of the traffic signs. The text file contains lines of the form: [Image number], [left column], [top row], [right column], [bottom row], [class ID], for each traffic sign in the dataset. The first field refers to the image file in which the traffic sign is located. Fields 2 to 5 describe the region of interest (ROI) in this image. Finally, the ClassID is an integer that represents the type of traffic sign. This is a different format to the YOLO label data set, which has a label format of: [object-class], [x], [y], [width], [height]. [object-class] indicates the class number of the object(0-3). [x] [y] is the centre of the ROI rectangle. [width] [height] is the width and height of the ROI rectangle. So before we can proceed with training, the We need to pre-process the gt.txt file to make it conform to the YOLO label format. The final data is turned into a YOLO compliant format, a JPG image, and the same named txt label file. We used a YOLOv5 network [9] to train the GTSDB data. YOLO is a fast open source object detection model that is more powerful and stable compared to other networks of the same size, and is an end-to-end neural network that can predict the classes and bounding boxes of objects. After the data pre-processing was completed, we used the pre-training weights from YOLOv5 to train our data. The YOLO network consists of three main components:

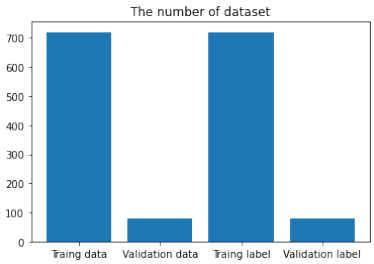


Figure 5: GTSDB training dataset and validation dataset.

- **Backbone:** A convolutional neural network that aggregates and forms image features at different image fine-grains.
- **Neck:** A series of network layers that mix and combine image features and pass image features to the prediction layer.
- **Head:** Prediction of image features, generating bounding boxes and and predicting categories.

The overview of YOLOv5 network structure is shown in Fig. 6. The important modules in YOLOv5 include BottleneckCSP, SPP (spatial pyramid pooling) and PANET. BottlenneckCSP is divided into two parts, Bottlenneck and CSP. Bottlenneck is a classical residual structure, first a 1×1 convolution layer, then a 3×3 convolution layer, and finally the initial input is added by the residual structure. The CSP divides the original input into two branches, each with a convolution operation that halves the number of channels, then branch one with a Bottlenneck $\times$ N operation, then concat branch one and branch two, so that the input and output of the Bottlenneck CSP are the same size, in order to allow the model to learn more features. YOLOv5 uses a lot of use of Convolutional Neural Networks (CNNs), in general structurally determined CNNs are usually fed with fixed size images for training and testing. So when faced with images of different sizes, they need to be changed to the same size after a series of operations such as cropping, or scaling. This results in a degradation of the accuracy of the recognition detection, and so the SPP technique is used, which allows us to input images of any size without the need to pre-process the image size in advance, allowing us to retain the maximum information of the original image. PANET is based on the Mask R-CNN and FPN frameworks and has the ability to enhance information propagation and accurately retain spatial information, which helps in the proper positioning of pixels to form masks.

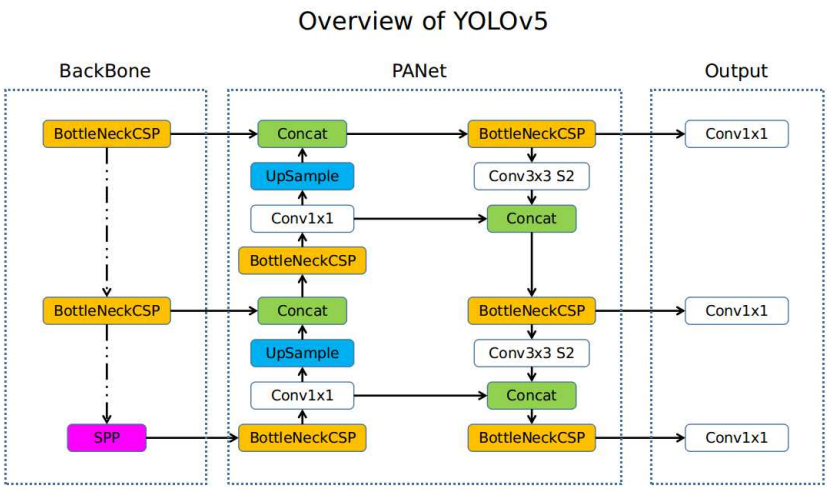


Figure 6: YOLOv5 Network Overview [10].

2.3 Combination of Classification and Detection Algorithms

Generally, the objective from this project is to detect and classify 43 traffic signs, but it is challenging to train YOLO to detect and classify 43 classes as it will cause lack in mean average precision (mAP) because of the small number of GTSDB dataset. To solve this problem and to be able to detect and classify all 43 classes, an approach [11] is used. The approach is to merge our models (YOLO model and classification CNN model). It is a two-step approach. The first step is to get the bounding box (localization) of traffic signs on images from the detector (YOLO) and then crop the detected signs based on the coordinates of bounding boxes. The second step is to feed the cropped part (traffic sign) as an input to the classifier. Figure 7 shows a flowchart about the method.

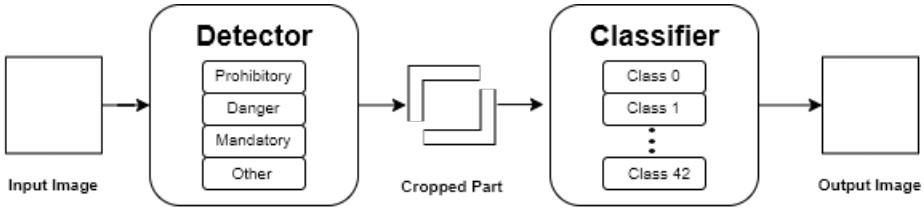
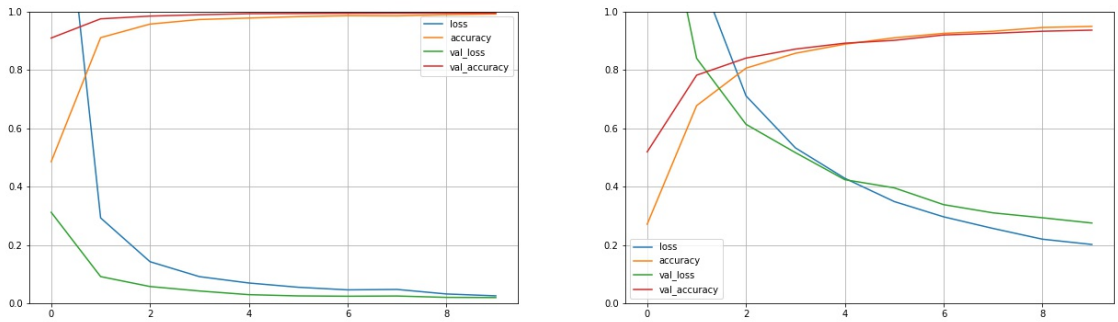


Figure 7: Flowchart of the combination.

3 Results

3.1 Classification



(a) Modified model accuracy and loss

(b) LeNet model accuracy and loss

Figure 8: Classification CNN accuracy and loss.

After data preprocessing and model modifications we saw very good improvement in accuracy and preventing in overfitting, as the model reached 99.08% training accuracy and 97.41% testing accuracy. Figure 8 shows the accuracy and loss for both LeNet and the modified model. Figure 9 shows random test images samples with true and predicted classes.

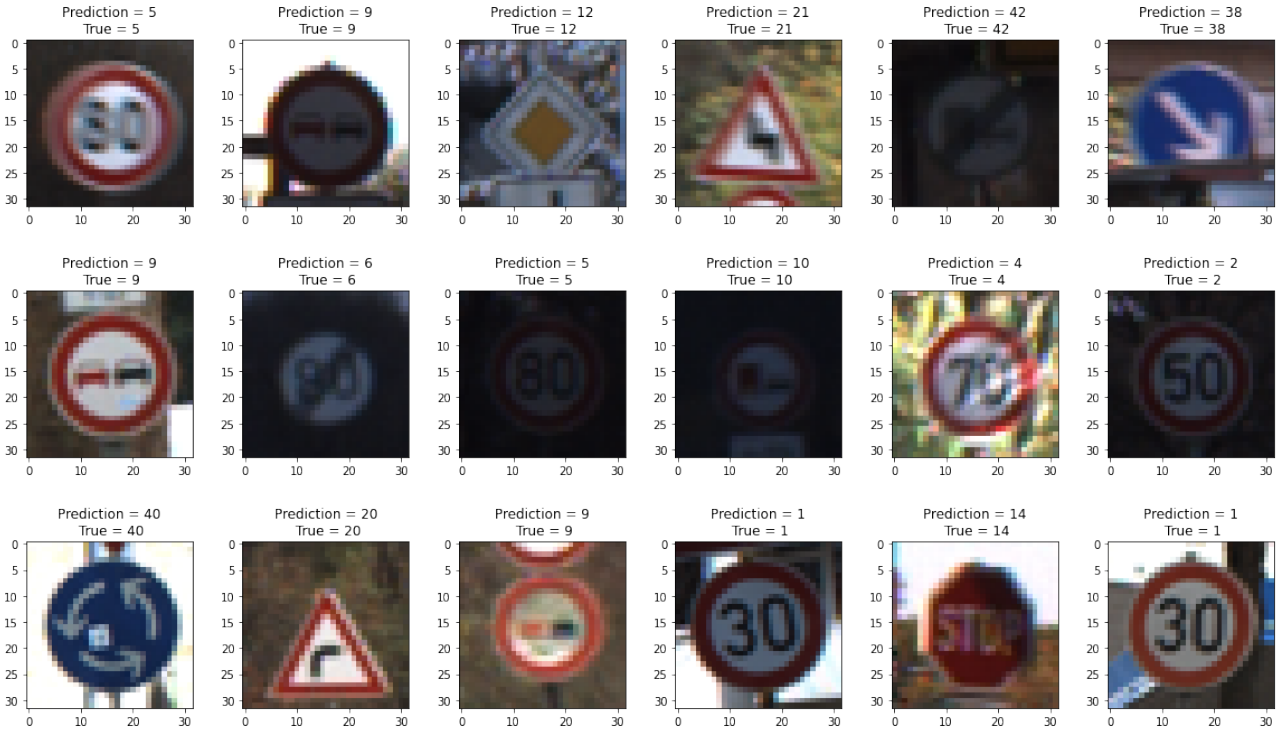


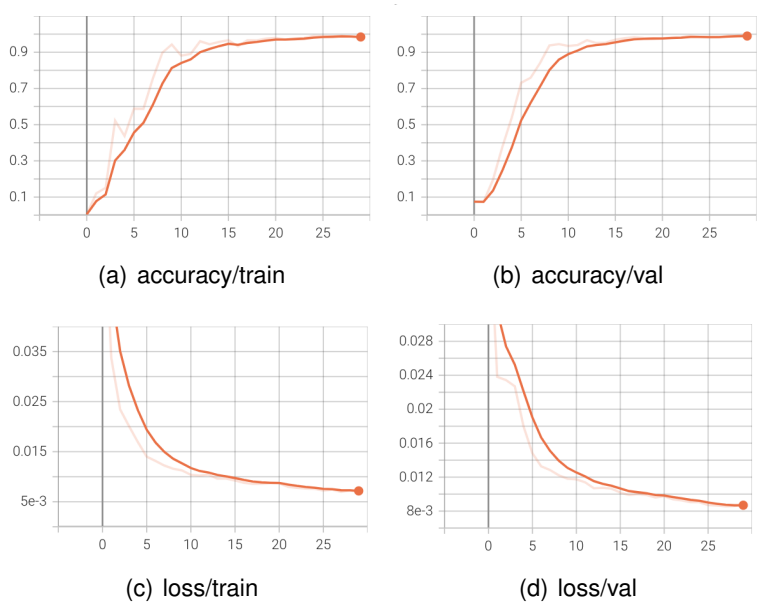
Figure 9: Samples from resulted test images with true and predicted classes.

3.2 Detection

In the target detection task, GTSRB has 43 classes and GTSDB has four classes. So we experimented with the two cases separately and obtained good performance.

Prohibitory	Danger	Mandatory	Other
Speed limit (20km/h)	Right-of-way at intersection	Turn right ahead	End of speed limit (80km/h)
Speed limit (30km/h)	General caution	Turn left ahead	Priority road
Speed limit (50km/h)	Dangerous curve left	Ahead only	Yield
Speed limit (60km/h)	Dangerous curve right	Go straight or right	Stop
Speed limit (70km/h)	Double curve	Go straight or left	No entry
Speed limit (80km/h)	Bumpy road	Keep right	End speed + passing limits
Speed limit (100km/h)	Slippery road	Keep left	End of no passing
Speed limit (120km/h)	Road narrows on the right	Roundabout mandatory	End no passing vehicle over 3.5 tons
No passing	Road work	-	-
No passing vehicle over 3.5 tons	Traffic signals	-	-
No vehicles	Pedestrians	-	-
Vehicle over 3.5 tons prohibited	Children crossing	-	-
-	Bicycles crossing	-	-
-	Beware of ice/snow	-	-
-	Wild animals crossing	-	-

Table 1: The content of four classes in GTSDB.



(a) accuracy/train (b) accuracy/val

(c) loss/train (d) loss/val

Figure 10: Detection accuracy and loss. Gray-red lines are for the baseline network, while red lines are for the final network.

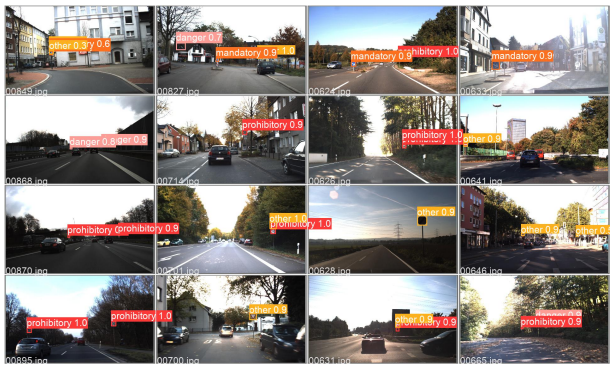
3.2.1 4 Classes Detection

For the four classes of the GTSDB dataset, the 43 classes of the GTSRB dataset were actually assigned to four classes('prohibitory', 'danger', 'mandatory', 'other') according to the meaning and characteristics of the traffic signs. See the table 1 for details.

As the number of images in our dataset totals 900, a small amount of training data is difficult to provide enough information for the model to learn, which may cause overfitting and make the model lack predictive power. Therefore, we used the pre-training weights of YOLOv5s to train the GTSDB dataset after pre-processing. The number of our training and validation sets were 593 and 148 respectively, and the images were both 1360$\times$800 in size and in RGB format.

We split the data into 12 batches and 30 epochs to speed up the training, so that we only need half an hour to complete the training. We obtained 98.02% and 99.13% accuracy for training and validation respectively (Figure 10). The training and validation losses were also reduced to 6.986e-3 and 8.7005e-3. After training, we also successfully predicted the location of the traffic signs in the images. See Figure 11. In addition to this, we have Precision-Recall curves for four classes in the GTSDB dataset. YOLOv5 found all traffic sign classes and the Precision/Recall were above 98%.

(a) labels



(b) predict

Figure 11: Pictures of ground truth and predicted values in four categories.

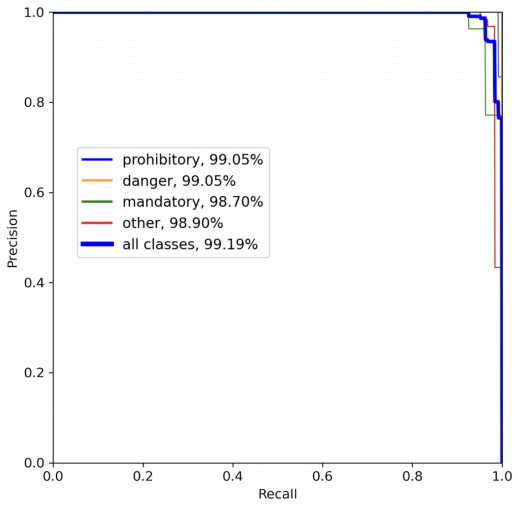


Figure 12: The Precision-Recall curves of the YOLOv5 module.

3.2.2 43 Classes Detection

As a result of the combination method, instead of just detect the traffic sign and classify it among 4 categories. The traffic signs have been detected and took a class among 43 different classes. As shown in figure 13, by applying only the detection algorithm the Yield (give way) traffic sign is detected and classified as ”other”. By applying the combination method the sign is detected and classified as ”Yield”. Figure 14 shows more result images from final algorithm.

((a)) Traffic sign classified as ”Other”

((b)) Traffic sign classified as ”Yield”

Figure 13: Test image sample after and before the combination of models.

Figure 14: Test images samples by the combination method.

4 Experimental Discussion

In the previous section we have successfully fulfilled the requirements for classification and detection and also presented the results we have obtained. In this section we will compare the results obtained in this experiment with those obtained in other studies.

4.1 Classification

The goal of traffic sign classification is to classify the detected traffic signs into specific subclasses. Convolutional neural networks have good results when faced with classification training. Classification experiments were also carried out in the research [12] and [13] for the GTSRB dataset. In [12],

they used a combination of CNN and HOG feature training, which won first place in the primary stage of the GTSRB competition. In [13], the method they use is based on a colour probability model and a colour HOG with a CNN for the classification of traffic signs. The computational speed is increased without compromising on accuracy. The accuracy of our method does not differ significantly from the results of other studies. We did not use HOG-based feature training, but simply greyed out the dataset in pre-training. It did not replace the images with colour channels to extract the features as in the other two studies. This may be the reason why the accuracy of our method is slightly lower than the other two methods.

Methods	[12]	[13]	modified model	LeNet-5
Accuracy	99.15%	97.75%	97.41%	85.12%

Table 2: The comparison between our classification module and others' research on GTSRB.

4.2 Detection

The goal of traffic sign detection is to find the location and size of traffic signs in a realistic scene. The most explicit image features are the shape and colour of the traffic sign. Target detection can be broadly divided into two main approaches: sliding window based and region of interest (ROI) based. [14] is the ROI-based approach to target detection. This method is divided into three steps. The first step is the colour conversion. The second step is shape matching. The third step is refinement of the ROI by which target detection is achieved. The [13] is based on the classification of GTSDB traffic signs into is based on the classification of GTSDB traffic signs into four classes, between which the shape and colour of the traffic signs have significant differences. The HOG features are computed on the probability map, so that the shape and colour information of the traffic signs can be fully utilised and the effect of the background can be suppressed to find the traffic sign area. The target detection method we use, YOLO, uses features from the entire image to make predictions about the bounding box. Unlike sliding window methods and regional proposal-based methods, YOLO can make use of full image information during training and prediction. The results of the YOLOv5 target detection we used were slightly better than the results of the remaining two studies. The possible reason for this lies in the extraction of image feature information. YOLO uses features from the whole image to predict the bounding box, perhaps extracting excess noise information, but not missing it. The remaining two methods, on the other hand, refine the image features before training. This method is able to reduce the training volume and improve the training speed. However, it causes information loss, making its prediction ability lower than YOLO.

	Prohibitory	Mandatory	Danger	Other
[13]	99.29%	96.74%	97.13%	-
[14]	100%	92%	98.85%	-
YOLOv5	99.05%	98.70%	99.05%	98.90%

Table 3: The comparison between our detection module and others' research on GTSDB.

5 Conclusions

The paper examines the problem of traffic sign detection and classification using GTSRB and GTSDB as datasets. A CNN with four convolutional layers is used to the classification problem, and YOLOv5 is used for the detection. Because of the large number of classes and little number of images in the GTSDB training dataset, it was suggested that the detection and classification algorithms that we used, better to be merged together into one distinct algorithm, which is able to detect and classify

43 different classes in images and videos. The final accuracy that achieved is 97.41% for the classification model and 98.02% for the detection model. The paper solves one of the most problems in ”perception” stack in autonomous vehicles, but in order to successfully implement the full function of an autonomous vehicle, several other components need to cooperate with each other to enable the vehicle drives itself in the environment with traffic rules.

References

[1] W. Schwarting, J. Alonso-Mora, and D. Rus, “Planning and decision-making for autonomous vehicles,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 187–210, 2018.

[2] L. Butler, T. Yigitcanlar, and A. Paz, “Smart urban mobility innovations: A comprehensive review and evaluation,” *Ieee Access*, vol. 8, pp. 196 034–196 049, 2020.

[3] M. Campbell, M. Egerstedt, J. P. How, and R. M. Murray, “Autonomous driving in urban environments: Approaches, lessons and challenges,” *Philosophical Transactions of the Royal Society A*, vol. 368, no. 1928, pp. 4649–4672, 2010.

[4] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, “Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition,” *Neural Networks*, no. 0, pp. –, 2012. [Online]. Available: `http://www.sciencedirect.com/science/article/pii/S0893608012000457`

[5] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: `http://jmlr.org/papers/v15/srivastava14a.html`

[7] A. Bäuerle, C. van Onzenoodt, and T. Ropinski, “Net2vis – a visual grammar for automatically generating publication-tailored cnn architecture visualizations,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 6, pp. 2980–2991, 2021.

[8] 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.

[9] R. Huang, J. Pedoeem, and C. Chen, “Yolo-lite: a real-time object detection algorithm optimized for non-gpu computers,” in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 2503–2510.

[10] seekFire, “Yolov5 overview,” `https://github.com/ultralytics/yolov5/issues/280`.

[11] V. N. Sichkar and S. A. Kolyubin, “Real time detection and classification of traffic signs based on yolo version 3 algorithm,” *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, vol. 20, pp. 418–424, 2020.

[12] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber, “A committee of neural networks for traffic sign classification,” in *The 2011 international joint conference on neural networks*. IEEE, 2011, pp. 1918–1921.

[13] Y. Yang, H. Luo, H. Xu, and F. Wu, “Towards real-time traffic sign detection and classification,” *IEEE Transactions on Intelligent transportation systems*, vol. 17, no. 7, pp. 2022–2031, 2015.

[14] M. Liang, M. Yuan, X. Hu, J. Li, and H. Liu, “Traffic sign detection by roi extraction and histogram features-based recognition,” in *The 2013 international joint conference on Neural networks (IJCNN)*. IEEE, 2013, pp. 1–8.