EAI Endorsed Transactions
on Industrial Networks and Intelligent Systems Research Article EAI.EU

Using Deep Neural Networks to Classify Symbolic Road Markings for Autonomous Vehicles

Daniel Suarez-Mash[1,2], Arfan Ghani[*3], Chan H. See[4], Simeon Keates[5], and Hongnian Yu[4]

[1] School of Computing, Electronics and Maths, Coventry University, CV1 5FB, UK
[2] Data Scientist, UK Home Office, Sheffield, S3 8NU, UK (email: daniel.suarez.mash@gmail.com)
[3] School of Engineering, American University of Ras al Khaimah, United Arab Emirates (Arfan.ghani@aurak.ac.ae)
[4] School of Engineering and the Built Environment, Edinburgh Napier University, Edinburgh, Scotland, EH10 5DT, UK (email: c.see@napier.ac.uk, h.yu@napier.ac.uk)
[5] University of Chichester, College Lane, Chichester, West Sussex, PO19 6PE, UK (email: s.keates@chi.ac.uk)

Abstract

To make autonomous cars as safe as feasible for all road users, it is essential to interpret as many sources of trustworthy information as possible. There has been substantial research into interpreting objects such as traffic lights and pedestrian information, however, less attention has been paid to the Symbolic Road Markings (SRMs). SRMs are essential information that needs to be interpreted by autonomous vehicles, hence, this case study presents a comprehensive model primarily focused on classifying painted symbolic road markings by using a region of interest (ROI) detector and a deep convolutional neural network (DCNN). This two-stage model has been trained and tested using an extensive public dataset. The two-stage model investigated in this research includes SRM classification by using Hough lines where features were extracted and the CNN model was trained and tested. An ROI detector is presented that crops and segments the road lane to eliminate non-essential features of the image. The investigated model is robust, achieving up to 92.96 percent accuracy with 26.07 and 40.1 frames per second (FPS) using ROI scaled and raw images, respectively.

Keywords: convolutional neural networks, symbol road marking, autonomous cars, intelligent systems, system design, embedded systems

Received on 23 February 2022, accepted on 16 May 2022, published on 16 May 2022

doi: 10.4108/eetinis.v9i31.985

1. Introduction

Symbolic Road Markings (SRMs) convey instructions that are essential components of all modern highways that enable road users to identify, understand and obey their respective highway codes. Studies have shown that detecting or classifying SRMs can greatly enhance vehicle localisation for autonomous vehicles [1]. Localisation enhances the capacity for road users to read the road ahead and although humans are competent at this task, autonomous vehicles have yet to grasp this capability fully. Previous research in classifying SRMs typically involves methods that can be split into two main categories [1][2]. The first involves explicitly specifying features such as colour and edges and using machine learning to extract those features in images. The second uses deep learning to detect more abstract and complex features.

Deep learning methods can be further split into two categories: two-stage such as R-CNN [2], Fast R-CNN [3], Faster R-CNN [4], Feature Pyramid Networks (FPN) [5] and mask R-CNN [6], whereas single-stage models include Single Shot Detector (SSD) [7] and the You Only Look Once (YOLO) [8] model. Two-stage models consist of a region proposal unit that pre-processes images in preparation for processing by deep learning architectures. The latest research in the classification and detection of SRMs has begun to involve deep learning.

Retrospectively, a much lower amount of research has been conducted when compared with research into detecting pedestrians, vehicles, lane markings and traffic lights. This is mainly due to the lack of ground truth data available for SRMs. Examples of datasets that have

*Corresponding author. Email: Ghani_786@yahoo.com, Arfan.ghani@aurak.ac.ae

emerged within the past 10 years include Cambridge [10], Daimler [11], Malaga [12], and KITTI [13].

Authors in [14] presented an SRM detection model and a fully annotated dataset based on Maximally Stable Extremal Regions (MSER) feature matching. It produced a competitive accuracy rate of 90.1% and a computational efficiency meant that special GPU hardware was not required. In the dataset used in [14], there are many more left-turn images than any other class. This dataset bias is discussed thoroughly later in that paper. Nevertheless, the provision of annotation and labelling has led several papers, some of which are discussed in this section, to use this dataset as a benchmark for their models. Therefore, for retrospective analysis, the same dataset was used in this paper which includes 1443 annotated images from roads in the United States as shown in Table 1.

Feature matching learning method MSER was used to search for previously unseen imagery [1] which was based on blob detection. If a feature match was found, then the corresponding class was attached to the image. Since MSER features are based on regions, this model is also able to provide a bounding box to each classification in an image. However, perspective problems may arise when SRMs are further away or closer to the camera, causing them to appear to have the same shape, but different sizes.

Table 1. Dataset breakdown

No.	Value	Count	Percentage
1	left-turn	705	48.8565
2	35	112	7.7616
3	right-turn	101	6.9993
4	rail	90	6.2370
5	forward	80	5.5440
6	40	69	4.7817
7	xing	64	4.4352
8	ped	54	3.7422
9	stop	49	3.3957
10	bike	41	2.8413
11	25	15	1.0395
12	forward&right	13	0.9009
13	yield	7	0.4851
14	X-crossing	6	0.4158
15	clear	6	0.4158
16	forward&left	6	0.4158
17	keep	6	0.4158
18	hump	3	0.2079
19	school	3	0.2079
20	stripe	3	0.2079
21	30	2	0.1386
22	slow	2	0.1386
23	speed	2	0.1386
24	car	1	0.0693
25	diamond	1	0.0693
26	lane	1	0.0693
27	pool	1	0.0693

The authors in [15] proposed a two-stage deep learning model aimed at detecting SRMs. The first stage consisted of an adaptive Region of Interest (ROI) detector that uses vanishing point detection to create an ROI image. These images are then fed into a deep CNN based on RetinaNet [16] for both training and testing. Three datasets were used in this research, namely the Cambridge, Daimler and Malaga urban datasets. The ROI detector used a Line Segment Detector (LSD) [17] together with CannyLines [18] to calculate a vanishing point. The architecture consisted solely of a lightweight CNN, without a region proposal system. Tests were undertaken on five iterations of deep CNN based on LeNet [19], with each containing a different number of convolutional and pooling layers. The dataset used to train the CNNs in this model was the same as used in this paper, consisting of 1443 images and 27 classes of SRM. However, data augmentation in the form of in-plane rotation was implemented in [9]15] which led to a total number of 20,479 images, as shown in Table 2.

Despite the significance of data augmentation, Table 2 still shows a large bias between classes in the augmented dataset where *left-turn'* is still the most represented class in the dataset. There are almost 6 times more *'left-turn'* images compared with the next most represented class *'35'*. This bias is measured and analysed in the following sections.

Table 2. Dataset resulting from data augmentation

Road Marking Class	Number of instances
STOP	674
LEFT	10433
RIGHT	1600
RAIL	1419
35	1793
FORWARD	1118
BIKE	590
40	1067
PED	799
XING	986
Total	20479

Authors in [20] created a real-time traffic light detector using a heuristic ROI detector followed by a lightweight CNN. The authors in [20] used three techniques in CNN architecture to reduce computational complexity. Small kernels of 1×1 and 3×3 were used to reduce the number of parameters in each layer. The CNN architecture also used stacked convolutional layers in contrast to the typical convolutional pooling modules seen in other research.

Inspired by the research published in [14] [15] [20], this paper demonstrates a novel classification method for painted symbolic road markings by using a region of interest (ROI) detector and a deep convolutional neural network (DCNN). The technique involves reducing the size of images before using Hough lines to achieve vanishing point calculation and image segmentation, which offers a unique advantage of classification accuracy and computational efficiency essential for real-world applications, such as SRM classification for autonomous cars.

The aims of this paper are categorised into three parts. Firstly, to create a region of interest (ROI) detector. Secondly, to join the ROI detector with a deep convolutional neural network. Finally, to ensure the completed model is computationally lightweight and accurate for real-time application.

2. Design Methodology

This research primarily focuses on investigating a model that is efficient in terms of Frames per Second (FPS) with improved classification accuracy. This paper used embedded hardware with a single CPU, 3 GHz, 6-Core Intel i5. Although this CPU has multiple cores, it does not offer the same capabilities as GPU.

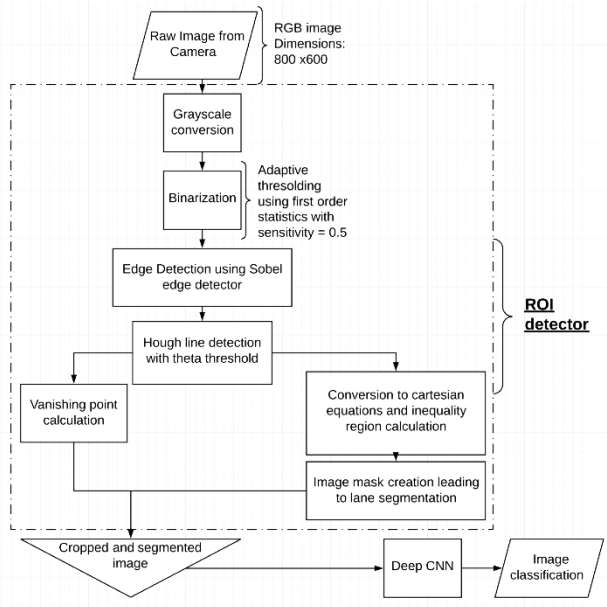


Figure 1. Model Architecture

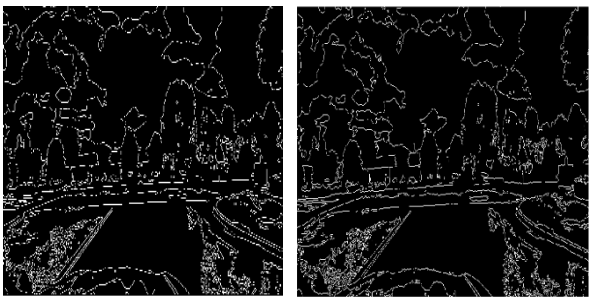


Figure 2. Sobel (left) vs Canny (right) edge detectors

The model workflow can be seen in Figure 1. The topology behind this architecture is to eliminate non-essential information which would reduce classification accuracy and speed by the CNN. The first pre-processing stage of the ROI detector is to apply a bottom-up vertical crop of 150 pixels to the image before the grayscale conversion. This section of the images contains only the bonnet of the camera vehicle. Therefore, this presents a good option to reduce the load on the rest of ROI detector as it has fewer pixels to work on, leading to reduced code run-time as evidenced by [9].

2.1. Edge Detection

Tests were run to distinguish the quality of both Sobel and Canny edge detectors for this application. Both algorithms have proven to be the most efficient amongst others; Canny is particularly useful for pattern matching in noisy images whereas Sobel is best as heavy data transfer of video and images. This was backed up when it was concluded that Sobel enabled the model to achieve higher FPS compare to its counterpart whilst retaining the same accuracy levels. It took 4.3 and 33.8 ms respectively to binarize the image in Figure 2 using Sobel and Canny detectors respectively. This is an 82% improvement in speed over the Canny edge detector. During real-time SRM classification of 412 images, the total time savings in code run-time averaged 6 seconds over 10 code executions. This is a substantial difference and the biggest reason Sobel was chosen over Canny.

Figure 3. Raw image from training data

2.2. Hough Line Detection

In this paper, Hough lines were detected to segment the lane and calculate a vanishing point. The segmentation of the lane is important as it removes image content unlikely to contain SRMs. As shown in Figure 3, the building, trees, sky and grass do not provide either insight or indication as to what SRM is present on the road.

The first stage in detecting Hough lines is to apply the Hough transform. This involves a voting procedure whereby each pixel on a line vote for that line. The Hough transform does not make use of the traditional straight-line equation form. Instead, it uses the Hesse normal form as shown in Figure 4 and equation (1). In this space, a straight line is defined using the following general equation:

$$\rho = xcos(\theta) + ysin(\theta) \qquad (1)$$

Where ρ and θ denote the distance from the origin perpendicular to the line and angle from the horizontal x-axis respectively. This is different to the general equation: $y=mx + c$, where m and c denote the gradient and y-axis intercept of the line respectively. The reason for not using this form is because vertical lines would cause m to be infinitely large. A thresholds for ρ and θ can be specified; as lane markings are never vertical nor horizontal, a θ threshold was defined and is shown below:

$$25 \leq \theta \leq 75 \quad \& \quad -25 \geq \theta \geq -75 \qquad (2)$$

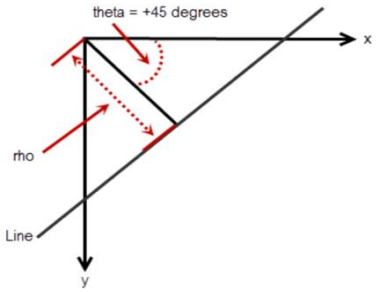


Figure 4. Hough plan

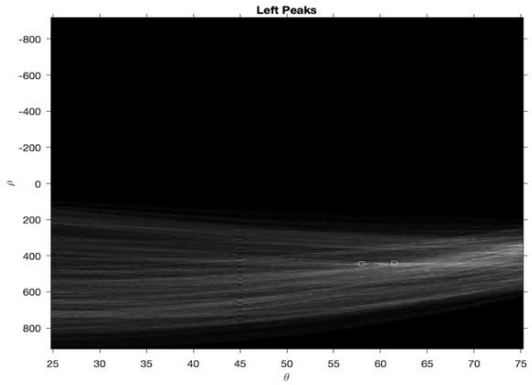


Figure 5. Peaks in the Hough plane from the left side of the road

These inequalities ensure that realistic lane lines, from both the left- and right-hand sides of the road, are detected. As there are two inequalities, two Hough transforms are applied to the image with either inequality. To save computational weight, the inequalities above in equation (2) are discretised in steps of 0.5.

After the Hough transform has been applied, Hough peaks are determined which are the Hough lines with the most votes. The more votes a line receives, the more predominant the line. In this model, the two lines within both inequalities with the most votes are selected and their parameters sorted into arrays. Two lines were chosen to ensure there is a redundant line in case one of the lines is inaccurate of the lane marking. Figure 5 shows the Hough plane and the small squares which denote the two pairs of ρ and θ with the most votes from the left-hand side of the road.

Figure 6 shows the four lines plotted visibly on an image from the dataset. The green lines are the same ones marked in squares in Figure 5.

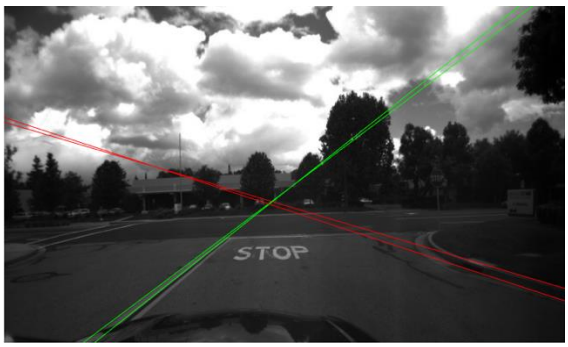


Figure 6. Hough lines plotted on image

To calculate the vanishing point and image mask, it is necessary to rearrange each line into $y = mx + c$ form. The following equation (3) can be obtained by rearranging the general Hesse normal form as follows:

$$y = -x\left(\frac{\cos(\theta)}{\sin(\theta)}\right) + \frac{\rho}{\sin(\theta)} \qquad (3)$$

By using this equation, it is clear that the gradient (m) and y-intercept (c) of each line is $-\left(\frac{\cos(\theta)}{\sin(\theta)}\right)$ and $\frac{\rho}{\sin(\theta)}$ respectively. However, the gradient was calculated by using equation 4.

$$m = \frac{y(800)-y(0)}{800} \qquad (4)$$

Where $y(800)$ and $y(0)$ are calculated using equation (3). The y-intercept is also taken to be $y(0)$. Number 800 is significant because it is the largest x-value in the image.

To calculate intersections, the coefficients of x and y and the y-intercept values are stored in arrays that mimic the form $-mx + y = c$. An example of this is shown below for the first line from the left- hand side:

$$left\ line\ 1\ coefficients = \begin{pmatrix} -m1_l & 1 \end{pmatrix} \qquad (5)$$
$$left\ line\ 1\ y-intercept = \begin{pmatrix} c1_l \end{pmatrix}$$

Where 1_l denotes line 1 from the left- hand side respectively.

The vanishing point is calculated as being the average mean intersection between the 2 pairs of Hough lines. The intersections are calculated using matrix operations. An example of a system of simultaneous equations between one left- and one right- hand side line is shown below in (6).

$$\begin{pmatrix} -m1_l & 1 \\ -m1_r & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} c1_l \\ c1_r \end{pmatrix} \qquad (6)$$

Where the intersection is calculated as:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} m1_l & 1 \\ m1_r & 1 \end{pmatrix} \setminus \begin{pmatrix} c1_l \\ c1_r \end{pmatrix} \qquad (7)$$

Figure 7 shows the Hough lines and their intersections between all lines in an image from the dataset.

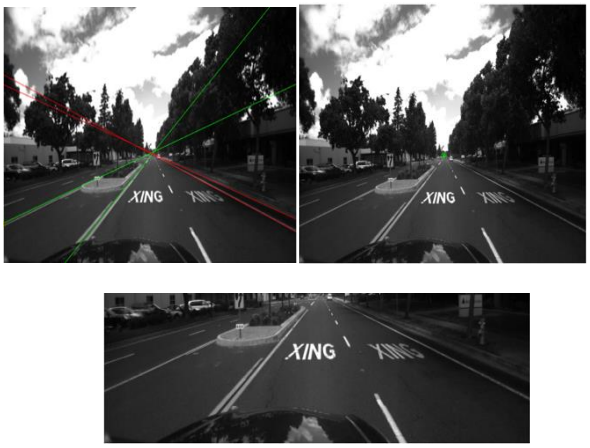


Figure 7. Hough lines (left), intersections (right), cropped image (bottom)

The y-coordinate of the vanishing point is then considered the maximum vertical height of the image and consequently used to crop the image. The result of this process can be seen in the bottom image of Figure 7.

To segment the lane from the rest of the image, a mask was placed on the image. One line from each side of the road is chosen and two inequalities are defined using the following principle:

$$-mx + y \geq c \qquad (8)$$

Any pixels which satisfy this inequality are set to 0 since they are above one of the lines. A mask such as the one shown in the left image of Figure 8 is placed on the image, which results in the image on the right. The overall effect of the vanishing point calculation and image segmentation is shown in Figure 9.

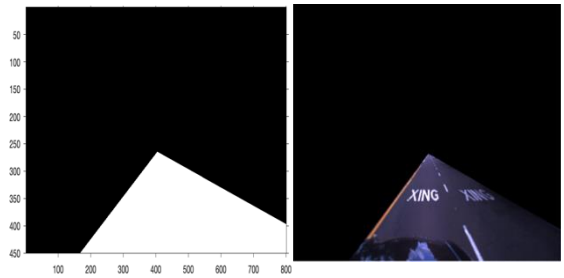


Figure 8. Image mask (left), segmented image (right)

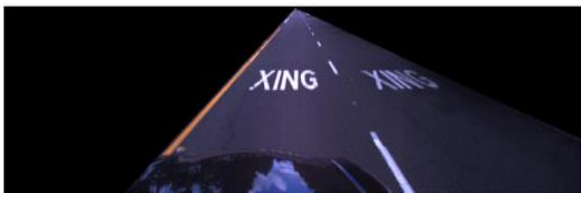


Figure 9. ROI detector output

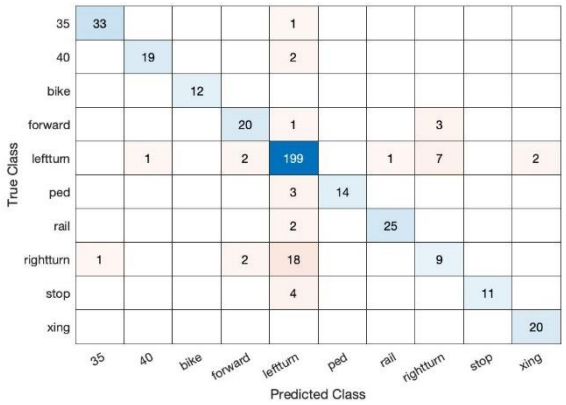


Figure 10. Colour CNN classification matrix

3. Deep Convolutional Neural Network (DCNN) Architecture and Testing

The CNNs used in this paper are AlexNet [21], SqueezeNet [22] and GoogleNet [23]. Their architectures are composed of 25, 68 and 144 layers respectively. All networks have an input size requirement of [227, 277, 3] and so each image that is fed into the CNNs needs to be resized to fulfil this requirement. In total, 5 deep CNN networks were used as listed in Table 3 where various networks were trained using images from the top 10 most represented classes.

In the case of AlexNet and GoogleNet, the fully connected layer, by default, outputs 1000 classes. Therefore, in AlexNet and GoogleNet based networks shown in Table 2, this layer was replaced with a fully connected layer with only 10 outputs. A different procedure was needed to adapt SqueezeNet to this application since it has a different architecture. Since SqueezeNet's last learnable layer is a convolutional layer and not fully connected, it is the final convolutional layer and classification layer which need to be replaced. This was done interactively by using the Deep Network Designer application.

To train the CNNs, the weights were randomly initialised and 'adam' was used as the learning method by which the CNN change its parameters to minimise its cost function. Adam computes individual adaptive learning rates for different parameters in the neural network which results in a faster learning algorithm when compared with the SGD (Stochastic Gradient Descent) learning method. The learning rate was specified as 0.0001 to ensure that the loss function can be truly minimised. If the learning rate was too large, it would cause gradient descent to overshoot the parameters needed to minimise the loss function.

The dataset was split into 60%, 30% and 10% for training, testing and validation respectively. This was done using the splitEachLabel function in MATLAB whilst specifying the randomized option which ensures the images are randomly allocated. By using the 'include' parameter, the algorithm was able to complete all testing using only the 10 most represented classes in the dataset.

	1	2	3	4	5	6	7	8	9	10
1	0.0001	0.0002	0.0001	0.0012	0.9777	0.0002	0.0004	0.0199	0.0001	0.0001

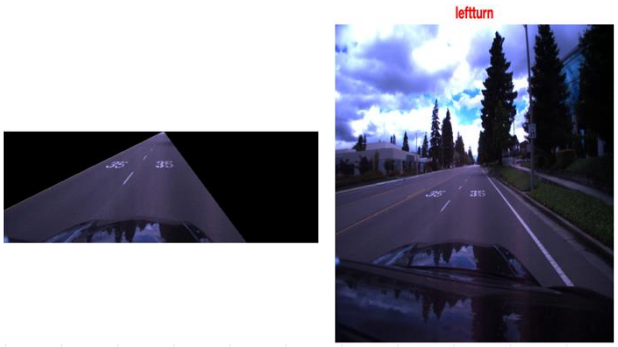
	1	2	3	4	5	6	7	8	9	10
1	0.0009	0.0019	0.0008	0.0028	0.4741	0.0009	0.0008	0.5164	0.0004	0.0011

	1
1	'35'
2	'40'
3	'bike'
4	'forward'
5	'leftturn'
6	'ped'
7	'rail'
8	'rightturn'
9	'stop'
10	'xing'

Figure 11. Scores for training image with left and right arrow, Colour CNN (top) vs Raw CNN (bottom) & Class number (right)

Table 3. CNN names and descriptions

CNN name	Description
Grayscale CNN	AlexNet-based CNN trained using grayscale segmented images.
Colour CNN	AlexNet-based CNN trained using colour segmented images.
Raw CNN	AlexNet-based CNN trained using raw images.
Colour CNN_SqueezeNet	SqueezeNet-based CNN trained using colour segmented images.
Colour CNN_GoogleNet	GoogleNet- based CNN trained using colour segmented images.



	1	2	3	4	5	6	7	8	9	10
1	0.0317	0.3000	0.0000	0.0000	0.6675	0.0006	0.0000	0.0000	0.0000	0.0000

Figure 12. Image segmentation and prediction scores

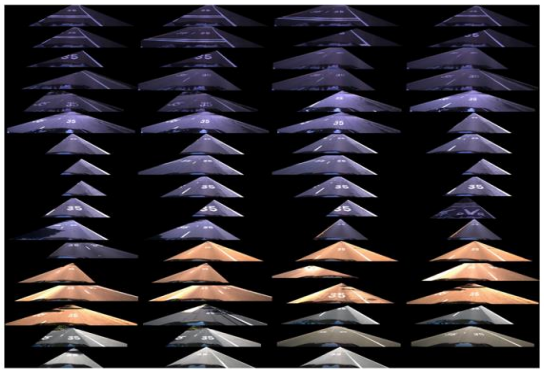


Figure 13. Montage of segmented training data for class '35' used for Colour CNN

To develop a successful deep learning model, a large dataset was a requirement for CNNs. The dataset was obtained from the research presented in [14] and downloaded from http://www.ananth.in/RoadMarkingDetection.html. It was chosen due to its size of manually labelled 1443 images organised into 27 classes of SRM. Figure 14 shows the training graph of ROI RESIZED + Colour CNN. It shows the training accuracy (blue graph) alongside the training loss (orange graph). The dotted black line shows the validation performance. Whereas the dataset was split into 60%, 30% and 10% for training, testing and validation respectively.

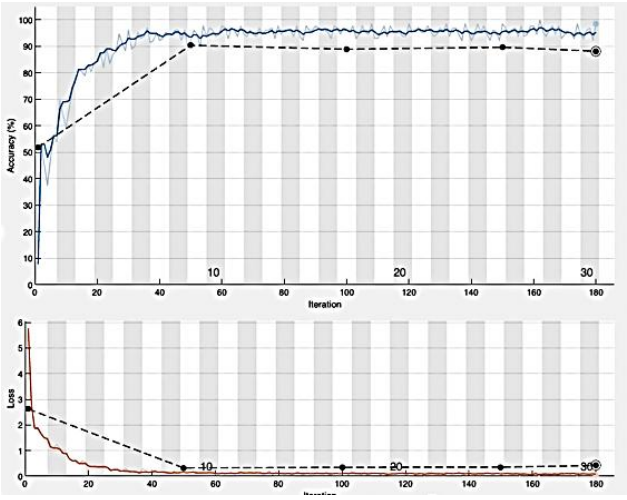


Figure 14. Training and test accuracy of ROI RESIZED + Colour CNN model

In this paper, multiple CNNs were developed and trained. Almost each CNN was trained using a different dataset to test what types of images worked best for image classification. For example, Colour CNN, Grayscale CNN and Raw CNN were trained using colour & grayscale segmented and raw images respectively. Table 4 shows the datasets and their corresponding CNNs.

Table 4. Datasets

Dataset	Trained CNN
Colour segmented dataset	Colour CNN Colour CNN_SqueezeNet Colour_CNN_GoogleNet
Raw images	Raw CNN
Grayscale segmented images	Grayscale CNN

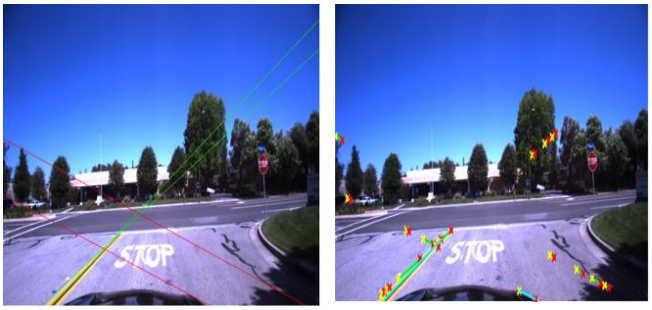


Figure 15. Hough lines on the image (left) and Hough line segments (right)

Figure 16. Line segments detected on image

To investigate and improve on the performance reported in the literature, accuracy rate and maximum supported frames per second were the two important factors for this study to assess the overall performance of the proposed models. All results shown in Table 5 are found using the test set consisting of 412 images. Table 5 shows the results of each model where the fastest ROI+CNN model presented in this paper is highlighted in bold achieved an accuracy rate of 92.96%. With respect to FPS, it is shown that the Raw CNN outperforms the other models reported in the literature. All accuracies and FPS figures are a mean average of 10 results.

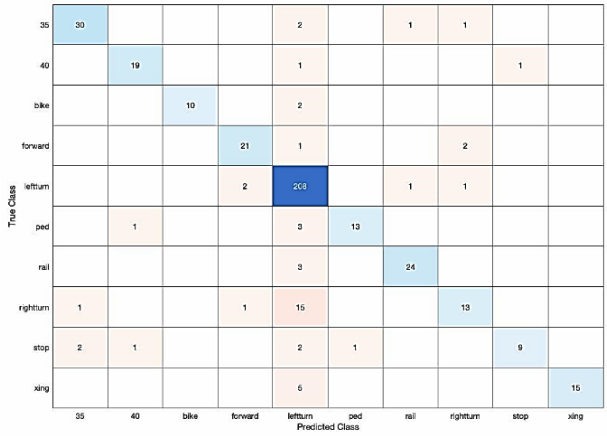


Figure 17. Confusion matrix for ROI_RESIZED detector + Colour CNN with top 10 classes

4. Discussions

Table 5 summarises the relevant performances retrospectively. In the dataset used, only the top 10 most represented classes were tested. The proposed work offers a unique perspective where not only SRMs were detected, rather bounding boxes were also provided along with retrospective classification accuracies.

The classification accuracy disparity in this case study is primarily due to the lack of true data augmentation to the dataset used. It is well established that CNNs perform better with more training samples. Hence, without this type of data augmentation, classes such as *bike* only contained 41 images. Not only is this not enough, but it is also far less than the most represented class, leftturn, which has 705 images. With regards to the models presented based on SqueezeNet and GoogleNet, the results are promising.

To understand the ROI detector models performances in terms of accuracy, the classification matrix was used to search for examples of misclassification. The classification matrix for Colour CNN can be seen in Figure 10. It was observed that the most misclassifications (18 in total) occur between *left-turn* and *right-turn* classes, i.e., when the true label is *right-turn*, but the predicted label was *left-turn*. In comparison, Figure 11 shows the scores for the Raw CNN when classifying an image from the training data. It shows a much more even prediction confidence between *right-turn* and *left-turn* classes. This is much more realistic and promising if this model was to be implemented in a real-world scenario. These confidence levels allow for future development of special scenarios whereby the CNN can output two classes to indicate that both arrows are present on the road. This therefore represents an area where the Raw CNN has more realistic conviction about its predictions in this scenario with multiple symbolic road markings. An unusual misclassification occurred when an image is segmented as intended but misclassified by Colour CNN. Raw CNN classifies this image correctly which makes this example a prime one to investigate the differences between the models.

As shown in the classification matrix, this misclassification only occurs three times between these two classes. By observing the scores in the bottom left image of Figure 12, the CNN was most confident about *leftturn* followed by *40* and finally *35*.

Table 5. Model Results

Model purpose	Model architecture	Accuracy rate (%)	Frames per second (FPS)	Hardware	Raw image resolution
SRM Classification *(Proposed)*	ROI detector + Grayscale CNN	87.38	23.37	Single CPU – 3 GHz 6-Core Intel Core i5	800 × 600
	ROI detector + Colour CNN	85.68	15.9		
	ROI_RESIZED detector + Colour CNN	**92.96**	26.07		
	Raw CNN	**91.26**	**40.1**		
	ROI detector + Colour CNN_SqueezeNet	63.33	28.03		
	ROI_RESIZED detector + Colour CNN_GoogleNet	69.9	24.41		
SRM Detection	Template matching using MSER features [14]	90.1	N/A	CPU	800 × 600
SRM Detection	ROI (vanishing point) + deep CNN [15]	96.9	18.87	NVIDIA Jetson TX2	Multiple: 960 × 720 1012 × 328 800 × 600

Using further investigation, those images are classified correctly due to the segmentation. This can be seen in the montage in Figure 13 which consists of the segmented training data for Colour CNN. During the further investigation into the accuracy disparity between raw and ROI detector models, it was found that some images were being segmented incorrectly. Despite being correctly classified by Colour CNN, certain images are not segmented correctly by the ROI detector which caused accuracy disparity since inconsistencies like these cause the CNN to learn inaccurate features which may not exist in other datasets such as the testing, validation or more importantly, real-time imagery.

Hough lines were not always representative of the edges of the lane as shown in Figure 15. By using the image on the right, the edges which triggered the detection of the left- hand side Hough lines are the side of the bonnet and the texture of the road. The right side of the vehicle bonnet, in particular, is primarily responsible for the detection of the disruptive Hough line which is the lower red line.

However, it is clear that there were no other reliable Hough lines that could have been chosen instead of this one. This can be deduced by the very small segments detected on the right-hand side compared with the left-hand side. This shows that there were not many options for the *houghpeaks* function to select. Figure 16 shows much longer line segments detected on certain images which later led to its accurate segmentation. This is what occurs in most images.

Despite the performance disparity of the standard ROI_detector, the modified ROI_RESIZED detector + Colour CNN outperformed the Raw CNN model by almost 2%. A confusion matrix for this improved model is shown in Figure 17. The confusion matrix shows much lower numbers of misclassification compared with Figure 10. This is partly because certain images were segmented better than before. For example, the segmentation of some images was improved by using the revised ROI detector as shown in Figure 18. The image reveals more of the *stop* SRM marking than the output from the other models presented in this paper.

Figure 18. Training data image improved segmentation by ROI_RESIZED detector

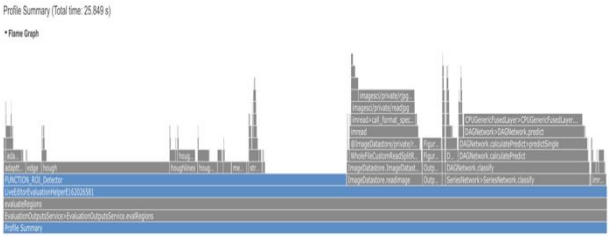


Figure 19. Flame graph for FPS test on 412 images using standard ROI detector

Furthermore, the code profiler was used to analyse and ascertain why the ROI-based models perform at a lower FPS. A screenshot of this is shown in Figure 19. The profiler ran the standard ROI detector and passed through 412 raw images from the Raw CNN test dataset. The blue and grey bars represent user-defined and Math Works functions respectively. The line taking the most time is the detection of Hough lines within the θ thresholds, which consumes a combined 34.2% of the time for the ROI detector. This is almost 6 times longer than the next most time-consuming line of code which is the calculation of an adaptive threshold for binarization. Given this analysis, it could be established that the runtime is strongly correlated to the number of pixels in the image. By reducing the number of pixels in the image, the voting space in the Hough transform will decrease and the inequality regions will decrease in size along with the time taken to calculate an adaptive threshold. This is led to the creation of a second ROI detector, ROI_RESIZED detector. This function changes the size of the input image to 227 × 227 at the start of the script. The rationale behind this was to use small input sizes of 96 × 96 and smaller for their CNNs. A smaller input size ensures that all functions now have fewer pixels to compute with. The results in Table 5 show a

significant increase of 38.85% in speed and FPS compared with the standard ROI detector using Colour CNN.

5. Conclusion

This paper investigated several CNN based SRM classification models where the proposed model focused on classifying painted symbolic road markings by using a region of interest (ROI) detector and a deep convolutional neural network (DCNN). This work also provided an in-depth analysis of the misclassification of images and aspects that leads to it. The model was demonstrated with the benchmark application and has shown promising performance and accuracy retrospectively. Reducing the size of images before using Hough lines to achieve vanishing point calculation and image segmentation is an energy-saving method that has not been attempted before in this application. More importantly, the best model shown in this paper strikes an important balance between accuracy and computational efficiency which is essential for real-world applications such as SRM classification for autonomous cars.

Acknowledgements.

The authors are thankful for the research funding (ENGR/007/22) provided by the American University of Ras al Khaimah, UAE.

References

[1] Ranganathan, D. Ilstrup and T. Wu, "Light-weight localization for vehicles using road markings," *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, 2013, pp. 921-927, doi: 10.1109/IROS.2013.6696460

[2] Girshick, R., Donahue, J., Darrell, T. and Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).

[3] R. Girshick, "Fast R-CNN", *Proc. IEEE Int. Conf. Comput*. Vis., pp. 1440-1448, Dec. 2015.

[4] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks", *IEEE Trans. Pattern Anal. Mach. Intell*., vol. 39, no. 6, pp. 1137-1149, Jun. 2017.

[5] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature pyramid networks for object detection", *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*., pp. 936-944, Jul. 2017

[6] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN", *Proc. IEEE Int. Conf. Comput*. Vis., pp. 2980-2988, Oct. 2017.

[7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, et al., "SSD: Single shot MultiBox detector", *Proc. Eur. Conf. Comput*. Vis., pp. 21-37, Oct. 2016.

[8] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement", arXiv:1804.02767v1, 2018, [online] Available: https://arxiv.org/abs/1804.02767.

[9] Girshick, R., 2015. Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

[10] Cambridge-Driving Labeled Video Database (CamVid), Oct. 2018, [online] Available: http://mi.eng.cam.ac.uk/research/projects/VideoRec/CamVid/.

[11] Daimler Urban Segmentation Dataset, Jan. 2019, [online] Available: http://www.6d-vision.com/scene-labeling.

[12] The Malaga Stereo and Laser Urban Data Set—MRPT, Oct. 2018, [online] Available: https://www.mrpt.org/MalagaUrbanDataset.

[13] A. Geiger, P. Lenz, C. Stiller and R. Urtasun, "Vision meets Robotics: The KITTI Dataset", *International Journal of Robotics Research*, vol. 32, no. 11, 2013. [online] Available: http://cvrr.ucsd.edu/vivachallenge/index.php/traffic-light/traffic-light-detection/.

[14] T. Wu and A. Ranganathan, "A practical system for road marking detection and recognition," *2012 IEEE Intelligent Vehicles Symposium*, Alcala de Henares, 2012, pp. 25-30, doi: 10.1109/IVS.2012.6232144.

[15] T. M. Hoang, S. H. Nam and K. R. Park, "Enhanced Detection and Recognition of Road Markings Based on Adaptive Region of Interest and Deep Learning," in *IEEE Access*, vol. 7, pp. 109817-109832, 2019, doi: 10.1109/ACCESS.2019.2933598.

[16] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal loss for dense object detection", *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 2999-3007, Oct. 2017.

[17] R. Grompone von Gioi, J. Jakubowicz, J. Morel and G. Randall, "LSD: A Fast Line Segment Detector with a False Detection Control," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 722-732, April 2010, doi: 10.1109/TPAMI.2008.300.

[18] X. Lu, J. Yao, K. Li and L. Li, "CannyLines: A parameter-free line segment detector," *2015 IEEE International Conference on Image Processing (ICIP)*, Quebec City, QC, 2015, pp. 507-511, doi: 10.1109/ICIP.2015.7350850.

[19] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient based learning applied to document recognition", *PIEEE*, vol. 86, no. 11, pp. 2278-2324, 1998.

[20] Ouyang, Z., Niu, J., Liu, Y. and Guizani, M., "Deep CNN-Based Real-Time Traffic Light Detector for Self-Driving Vehicles," *IEEE Transactions on Mobile Computing*, 19(2), 2020

[21] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet classification with deep convolutional neural networks", *Proc. Int. Conf. Neural Inf. Process. Syst*., pp. 1097-1105, 2012.

[22] Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J. and Keutzer, K., 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. arXiv preprint arXiv:1602.07360.

[23] C. Szegedy et al., "Going deeper with convolutions", Proc. 14th *Int. IEEE Conf. Intell. Transp. Syst*., pp. 1609-1615, 2014