SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

Vision-Based Multi-Stages Lane Detection Algorithm

Fayez Saeed Faizi[1,2]* and Ahmed Khorsheed Al-sulaifanie[1]

[1]Department of Electrical and Computer Engineering, College of Engineering, University of Duhok, Zakho Road - Duhok, Iraq
[2]Department of Energy Engineering, Technical College of Engineering, Duhok Polytechnic University, 61 Zakho Road, 1006 Mazi Qr - Duhok, Iraq

ABSTRACT

Lane detection is an essential task for autonomous vehicles. Deep learning-based lane detection methods are leading development in this sector. This paper proposes an algorithm named Deep Learning-based Lane Detection (DLbLD), a Convolutional Neural Network (CNN)-based lane detection algorithm. The presented paradigm deploys CNN to detect line features in the image block, predict a point on the lane line part, and project all the detected points for each frame into one-dimensional form before applying K-mean clustering to assign points to related lane lines. Extensive tests on different benchmarks were done to evaluate the performance of the proposed algorithm. The results demonstrate that the introduced DLbLD scheme achieves state-of-the-art performance, where F1 scores of 97.19 and 79.02 have been recorded for TuSimple and CU-Lane benchmarks, respectively. Nevertheless, results indicate the high accuracy of the proposed algorithm.

Keywords: Autonomous cars; deep learning; lane detection; vision-based CNN

INTRODUCTION

Autonomous driving has received much attention in recent years, which has led to rapid development in this technology. Artificial intelligence is increasingly utilized and considered the leading technique for self-driving and advanced driver-assistant systems (Tabelini et al., 2021a). For these systems to work efficiently, lane detection is an essential problem that needs to be addressed. It has a pivotal impact on the driving system's performance and accuracy. It is also important in applications such as lane-keeping assistants, driving route planning, and real-time vehicle positioning (Al-Jarrah et al., 2018).

ARTICLE INFO

Article history:
Received: 13 April 2023
Accepted: 20 November 2023
Published: 22 July 2024

DOI: https://doi.org/10.47836/pjst.32.3.19

E-mail addresses:
fayez.faizi@dpu.edu.krd (Fayez Saeed Faizi)
ahmed.khorsheed@uod.ac (Ahmed Khorsheed Al-sulaifanie)
*Corresponding author

ISSN: 0128-7680
e-ISSN: 2231-8526

Conventional lane detection algorithms mainly extract features through hand-crafted operators (Aly, 2008; Jiang et al., 2009; Kim, 2008), followed by fitting a line via methods such as random sampling consensus (Kim, 2008) and Hough transforms (Liu et al., 2010). Nevertheless, such methods lack accuracy in the actual environment because they depend on low-level hand-crafted features (Ko et al., 2022). Conversely, deep learning was deployed in recent studies, which resolved some shortcomings of traditional methods and gave better results (Abualsaud et al., 2021). The Deep Learning (DL)-based methods showed better accuracy than traditional methods, resulting in efficient process handling by deep learning. In addition, the generalization option provided by AI-based methods is missing in traditional algorithms because the conventional systems depend on hand-crafting low-level feature extraction. Segmentation was the earliest method proposed for lane detection through deep learning. Other paradigms have been presented lately, such as row-wise detection, anchored-based, and parametric prediction methods (Tabelini et al., 2021a). Deep learning-based lane detection methods have made significant progress and present robust algorithms; however, many challenges must be addressed.

Most lane detection methods depend on predicting points on the lane line and fitting a line to these points. Detecting and then aggregating points into line(s) is challenging. According to recent references in the lane detection field, deep learning-based algorithms show significant results from accuracy, speed, and total efficiency points of view. The fast development of AI technology helps to present reliable self-driving systems. Nevertheless, these methods still face some challenges. The key point challenge is recognizing the lane line(s) and determining points on that line(s) that can be utilized later for different purposes, including steering angle prediction, speed control, or a complete self-driving system (Ko et al., 2022; Tabelini et al., 2021a). Road condition and light intensity play essential roles in the vision-based detection process because they make the features difference to distinguish between lane line and background very small. Rain on the road, shadows on the lane line, dazzle, or dim light are common issues the system needs to face (Zang et al., 2018). Assigning the predicted points to the related lines is another challenging problem, especially because the lanes' shape and number change over the road (Neven et al., 2018; Tang et al., 2021).

This study proposes a new lane detection algorithm named DLbLD to overcome the mentioned challenges. The high learning ability of CNN is associated with a new network architecture to predict the presence of lane line parts in the incoming image block. The proposed network can recognize the features that lead to a binary classification of the incoming block as Lane or Background. A Gaussian filter eliminates pepper and salt noise in determining a point on the lane line for each block labeled Lane from the last stage. The DLbLD algorithm projects all the predicted points from the previous step into a one-dimensional form before applying the K-mean machine learning (ML) clustering method to improve assigning the detected points to the related lane line.

Different Lane detection methods have been presented in recent years; these methods can be divided into four types according to the strategy used in describing line shapes: segmentation-based methods, parametric prediction methods, row-wise detection methods, and anchor-based methods (Abualsaud et al., 2021; Tabelini et al., 2021a).

From a Segmentation point of view, the predictions are made on a pixel level, where each pixel is marked as a Lane or Background. Unlike a general segmentation process, instance-level discrimination is needed for the lane detection task. Pan et al. (2018) proposed the SCNN paradigm, which deploys the multi-class classification technique for instance discrimination. SCNN performs efficiently in long, thin structures but at the cost of speed. It is slow and achieves 7.5 frames per second (FPS), limiting its real-time application. A large backbone is an essential cause of low-speed performance. Also, Hou et al. (2019) present a self-attention distillation (SAD) scheme to obtain rich contextual information from a frame, where a high-level feature map is used to build the low-level feature maps. However, to generate the segmentation map, a post-processing phase is required to translate it into different lane lines. At the same time, Lee and Kim (2022) propose a multi-task learning paradigm, including a lane detection function. Their experiments illustrate promising results with a speed of 37 FPS. All the experiments were run on TITAN RTX NVIDIA, an expensive and high-powered GPU.

On the other hand, the Anchor-based methods overcome the problems that segmentation-based modules face (Liu, Chen et al., 2021). Anchored methods deploy a top-to-down methodology and determine the affined coordinates by optimizing the lane shape. In their work, Tabelini et al. (2021a) proposed an algorithm with slender anchors because the normal box anchor used in object detection does not suit their paradigm. With the predefined anchors, the proposed method limits the effect of the no-vision-clue issue and emphasizes instance discrimination ability. Another lane detection framework was presented by Xu et al. (2022) that shows a high-speed performance due to its light backbone network size. The pipeline addresses the curve lane issue by capturing accurate curve information and long-ranged coherence while preserving an efficient computational division. Nevertheless, fixed anchor shapes have described lane line shapes with a low degree of freedom.

From the Row-Wise detection method perspective, the input image as grid divisions is used to predict the location of the cell that contains a lane line part. Only the cell with the highest probability is selected on each row, which is why this process must be repeated for every lane in the image. As in segmentation-based methods, row-wise detection-based methods require a post-processing step to aggregate the detected cells into a line. Yoo et al. (2020) proposed an idea to avoid dense prediction at the pixel level. Their results show promising efficiency after testing the proposed algorithm on two datasets. Philion (2019) and Qin et al. (2020) present other schemes to achieve higher speed. They reach 90 FPS and more than 310 FPS, respectively, but at the cost of losing some accuracy.

The above methods are based on modules that predict points on the lane line and then, through post-processing, regress these points into line(s). The algorithm directly outputs a curve equation representing a lane line using Parametric prediction methods. Tabelini et al. (2021b) and Liu, Yuan et al. (2021) proposed different lane detection algorithms by directly predicting quadratic or cubic polynomials representing different lane lines in the image. Both methods show a high-speed performance with 115 FPS and 420 FPS, respectively. However, the parametric prediction-based paradigms did not overtop other methods from an accuracy perspective. Generally, methods based on the Parametric prediction method show a high FPS with lower accuracy than other methods.

In contrast to what was presented before, our study makes the following contributions. First, the detection of lane features in challenging environmental conditions should be improved, and the incoming image blocks should be classified into Lane or Background groups. Second, each Lane labeled block is represented with a single coordinate representing a pixel on the line. Third, the detected points are categorized into the related lane lines.

PROPOSED METHOD

Deep learning is an efficient method of recognizing patterns after it is trained (Chen & Xiang, 2022). The proposed algorithm combines CNN, Gaussian filter, ML clustering, and curve fitting to detect a lane line. Figure 1 shows the flow diagram of the proposed algorithm applied in each incoming frame.

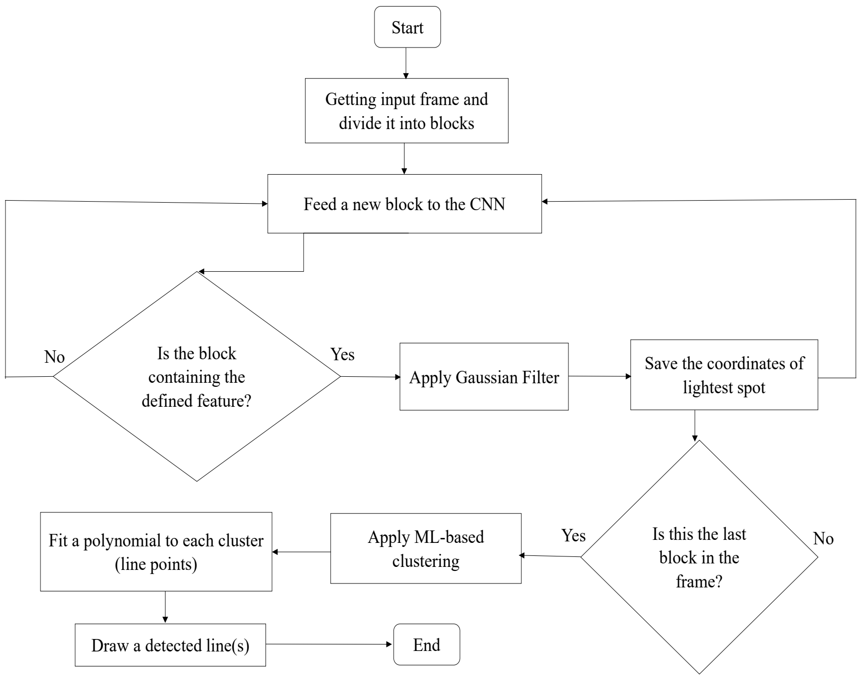

Start
Getting input frame and divide it into blocks
Feed a new block to the CNN
Is the block containing the defined feature?
No
Yes
Apply Gaussian Filter
Save the coordinates of lightest spot
Is this the last block in the frame?
Yes
No
Apply ML-based clustering
Fit a polynomial to each cluster (line points)
Draw a detected line(s)
End

Figure 1. General flow diagram of the DLbLD algorithm

Network Architecture

TensorFlow (Abadi et al., 2016) and Keras (Chollet, n.d.) are used to build the deep learning network of the proposed algorithm. The DLbLD network consists of three convolutional layers, each having a rectified linear unit (ReLU) activation function. The last two are followed by a Max-pooling size 2 x 2 layer. The depth of the first layer is 16, the second is 32, and the last is 128. All utilize a 3 x 3 kernel size. Next comes a flattened layer, followed by the dense layer with a depth of 128 and ReLU activation function, a 50% dropout layer, and a final dense layer with a single output and a Sigmoid activation function. Figure 2 illustrates the architecture of the network architecture used in this work.

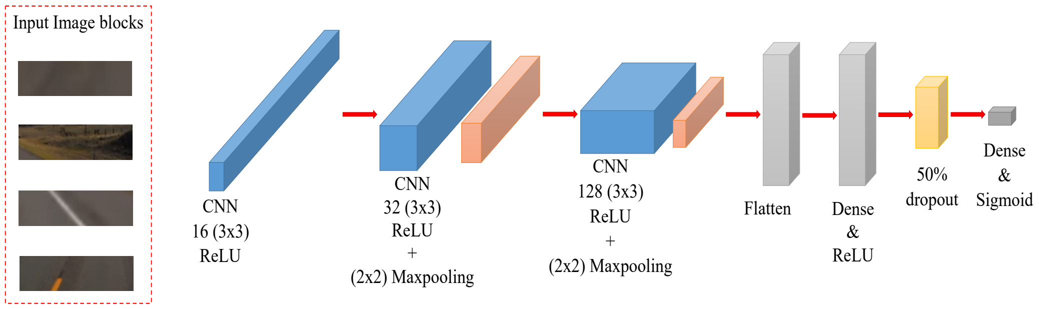


Figure 2. Proposed network architecture

First, the image is divided into smaller blocks, each having a 20-pixel height and 80-pixel width, as shown in the left part of Figure 2. The two upper blocks represent a sample of Background images, while the lower two are image blocks with a part of the Lane line. The network receives an RGB image block with a 20 x 80 x 3 dimension as an input—obtained from a camera located on the vehicle—and divided into blocks. Then, each convolution layer extracts features from the input image, resulting in a high-dimensional feature map of height x width x channel dimension. The feature map is flattened, fully connected, and 50% dropped. It outputs a single binary value indicating whether the input image block has the required feature (lane line part).

The network used in this work is a modified form of the common network used by He et al. (2016) and Zang et al. (2018). The version deployed here contains different layers with different parameters and outputs a binary decision. It contrasts with other works that result in Detection Loss or other required features depending on the system task. Such networks have a relatively simple architecture and still perform efficiently in feature extraction and recognizing the required characteristics (He et al., 2016; Tang et al., 2021).

In contrast to the work presented by Liu, Yuan et al. (2021), who used the Transformer encoder-decoder network, the idea behind using such common architecture is to show the power of the proposed idea. The incoming image is divided into small blocks and fed to a trained CNN that searches for one particular matter—the existence of a lane line part.

The network searches for features that indicate whether the incoming block contains a lane line part. Based on the network's prediction, the system decides whether to discard the incoming block if it is a background image block with no lane line or pass it to the next phase in the algorithm if it contains a lane line part.

Identifying Points on the Line

The network output represents a label that indicates if the input image block contains lane line features. The block labeled with a binary value of "1" indicates the presence of a line segment, while the "0" label refers to the absence of a line segment in that block (image part). The system stores the coordinates of the top-left corner of the block labeled "1" as it represents (with the offset 80 x 20) a specific block that feeds to the network. But the coordinates of a point on the line (PO) inside the block that belongs to group "1" are needed. The idea utilized in this work to detect a single point on the labeled block located on the lane line part is considering the intensity difference between the lane color and the background. It is known that the lane line color is either white or yellow, while the background (road) is black. In digital images, the lighter colors have a higher intensity than the darker ones. The pixels of the line part have a higher intensity value than the background pixels so that the PO can be calculated as the maximum intensity in the block. Nevertheless, not all the incoming blocks are clean, but they are distorted by a noise similar to pepper and salt noise. In this case, the salt noise point is selected as a PO. The proposed algorithm utilizes the Gaussian blur filter of 5 x 5 to solve this issue and get the real point coordinates on the line. The blur kernel applied to the block labeled "1" is generated using the 2D Gaussian equation shown in Equation 1:

$$G_{(x,y)} = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad [1]$$

where x and y specify the distance from the center point (0,0). For the kernel used in this work, x and y range from -5 to 5. σ is the standard deviation, indicating how significantly the pixels around the center pixel affect the computation result. Figure 3 illustrates the effect of deploying this filter.

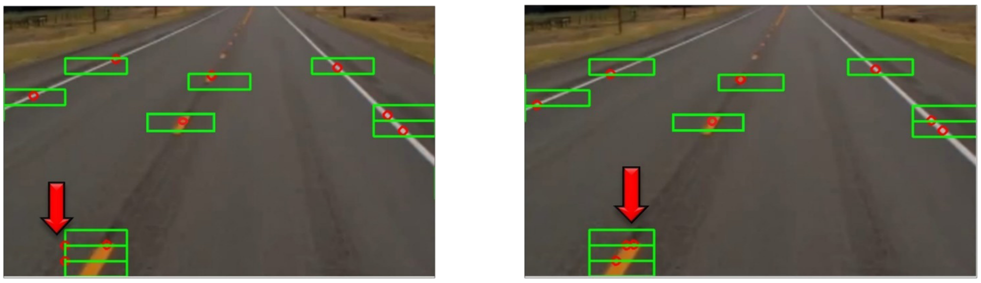

(a) (b)

Figure 3. Location of PO, (a) without using Gaussian kernel, (b) with the use of Gaussian kernel

In Figure 3(a), the two down-left blocks give the wrong PO points due to the presence of a salt noise. In Figure 3(b), the system determines the correct PO points on the line after deploying the Gaussian blur filter.

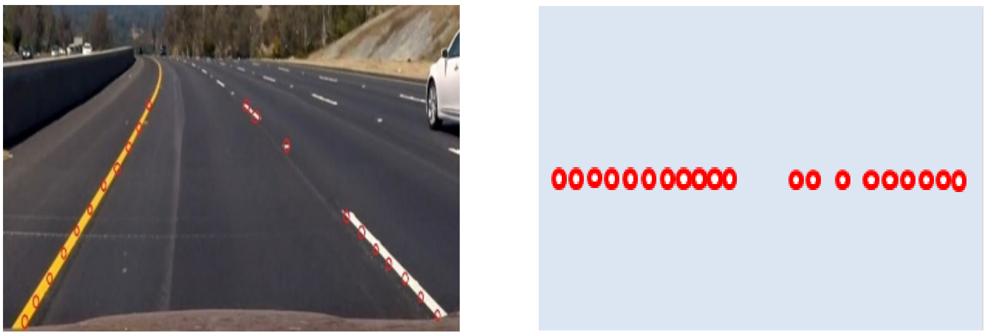
Assigning PO Points to the Related Lane Lines

All the detected points on the line POs are saved as raw data points for each video frame. These data must be classified according to the lane line they belong to, which is used to draw the polynomial that fits that set of points. Because the detected POs seem 'Scattered' on the image, it is essential to specify each PO to the related Lane line. Conventional methods that depend on simple 'Center point dividing' were used in other works to perform this task. Such methods try to anchor a point between the two-lane lines and assign the points located on the left of the anchored point to the left line and the remaining detected points to the right lane line. These methods are suffused from an accuracy perspective, which applies only to two-lane line cases. On the other hand, in this work, the well-known K-mean (Lloyd, 1982), the ML-based clustering method, is used to achieve this goal. Deploying K-mean clustering directly on the raw data assigns some points belonging to one line to another. The proposed method performs a pre-processing step, making the clustering process more accurate. The idea is to put all the detected points (the raw data points) on one y-level instead of being distributed over the y-axis (i.e., project the points into a one-dimensional form). Figure 4 shows this process, where, first, the system detects the points on the line. Then, y-values for all POs are set to the center y-value of the POs using Equations 2 and 3.

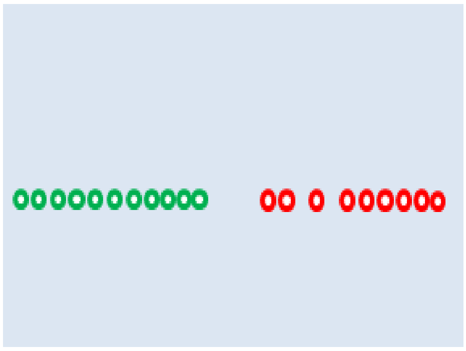
$$y_{center} = \frac{y_{\max} + y_{\min}}{2} \qquad [2]$$

$$PO_{new} = (x, y_{center}) \qquad [3]$$

K-mean clustering is applied to all new (POs), which produces an efficient grouping. Finally, return the original y-value to each PO.



(a) (b)


(c)

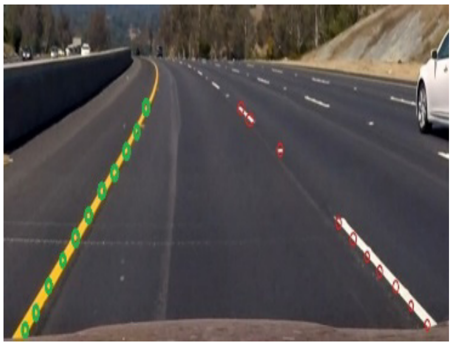

(d)

Figure 4. K-mean clustering, (a) detected pos, (b) new pos with new y-value, (c) deploying k-mean clustering, (d) returning original y-values and setting each PO to its line

The method explained in Figure 4 improved the clustering results significantly and put each PO into its correct lane line. The next step depends on this, so ensuring that each PO belongs to the correct group is important.

Creating the Lane Lines

After the clustering, the proposed algorithm fits all the POs belonging to a specific line via a quadratic polynomial shown in Equation 4.

$$f(x) = a_2x^2 + a_1x + a_0 \qquad [4]$$

Using Python (Rossum & Drake, 1995), the command *coef = np.poly1d(np.polyfit(x,y,2))* returns the coefficient of the quadratic equation that will be used in building each lane line, where x and y represent a list of x and y values for each line group, respectively. Finally, a line is drawn over each set of POs belonging to each line with a different color.

RESULTS AND DISCUSSION

Several experiments were performed using different benchmarks to evaluate the proposed algorithm. The results were compared to those of other recent works to gauge the impact of the proposed method.

Datasets

Two benchmarks, TuSimple (Pizzati et al., 2020) and CU-Lane (Pan et al., 2018), were utilized to evaluate the DLbLD algorithm. TuSimple is a widely used dataset consisting of 6,766 frames of highways with different weather conditions and daytimes. A one-second video of this benchmark is formed from 20 frames. The CU-Lane dataset is a larger and more challenging benchmark, consisting of 133,235 frames divided into nine categories for both urban and highway roads. The summary of these benchmarks is illustrated in Table 1.

Table 1
Details of used datasets

Dataset	Train	Test	Validation	Road
TuSimple	3.63K	2.78K	0.36K	Highways
CU-Lane	88.88k	34.68k	9.68k	Highways and urban

Evaluation Method

F1 measurement was considered by applying the method used by Pan et al. (2018) to judge the algorithm's accuracy in detecting lane lines. Intersect of union (IoU) between prediction and the ground truth is calculated (for a fixed-line width) to determine whether the sample is a true positive (PT), false negative (FN), or false positive (FP). The F1 measure is evaluated according to Equation 7, which utilizes precision and recall, as in Equations 5 and 6.

$$Precision= \frac{TP}{TP+FP} \quad [5]$$

$$Recall = \frac{TP}{TP+FN} \quad [6]$$

$$F1 = \frac{2\ x\ Precision\ x\ Recall}{Precision\ +Recall} \quad [7]$$

For the TuSimple dataset tests, the accuracy metric is also considered, according to Equation 8, to evaluate the accuracy of the proposed method.

$$Accuracy = \frac{\sum_{frame} POs}{\sum_{frame} TPOs} \quad [8]$$

Where POs are the number of detected points on the line, and TPOs are the total number of points on the line for that frame.

Implementation Details

The proposed CNN was first trained with a dataset of 8,500 images, divided equally into two groups containing Lane Line and Background. The RMSprop with a learning rate inspired by the strategy Loshchilov and Hutter (2017) used, equal to 0.001, was an optimizer and utilized “binary_crossentropy” as a loss augment. The presented paradigm preserving data augmentation needs a long learning time, so we chose 200 training epochs for both datasets. Results recorded in this paper for comparison were performed using an NVIDIA GeForce GTX 1080 card.

Experiment Results

Before evaluating the proposed system, several experiments were done to attain the best possible results. A main aspect considered was the block-size image parts fed to CNN. Selecting the dimensions of the block has a significant impact on the results. A program was built that divided the incoming image into smaller blocks and fed it to the network to choose the block size with optimal prediction accuracy. This program divided the incoming image into blocks of 10 x 10 pixels. Then, it took all the probabilities for width and height—by an increasing step of 10 pixels—until it reached a block size of 120 x 120 pixels. It took 144 different block sizes. The system fed the blocks to the network at each probability and calculated the prediction accuracy. Figure 5 summarizes the results of this process.

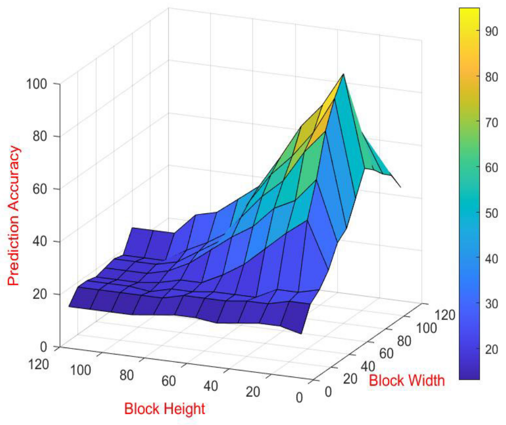


Figure 5. Accuracy for different block-size

It is clear from Figure 5 that a block size of width=80 pixels x height=20 pixels gives the highest prediction accuracy, which explains why these dimensions have been chosen in this work. The proposed algorithm was tested on both the TuSimple and CU-Lane benchmarks, and the results were compared to some of the recent methods proposed in this research, including Tabelini et al. (2021a), Hou et al. (2019), Pan et al. (2018), Philion (2019), Yoo et al. (2020) and Qin et al. (2020).

Using the TuSimple Dataset

The TuSimple dataset was used to evaluate the proposed algorithm in the first experiment set. The comparison results on deploying the TuSimple benchmark are illustrated in Table 2. The proposed algorithm achieved a new F1 score of 97.19 with a low FP of 2.0.

Table 2
Comparing DLbLD to different methods using the TuSimple dataset

Method	Accuracy	FP	FN	F1
DLbLD (ours)	96.56	2.0	3.52	97.19
SCNN (Pan et al., 2018)	96.53	6.17	1.8	95.97
ENet-SAD (Hou et al., 2019)	96.64	6.02	2.05	95.92
FastDraw (Philion, 2019)	95.2	7.6	4.5	93.92
UFAST-ResNet34 (Qin et al., 2020)	95.86	18.91	3.75	88.02

Table 2 *(Continue)*

Methods	Accuracy	FP	FN	F1
UFAST-ResNet18 (Qin et al., 2020)	95.82	19.05	3.92	87.87
LanATT-ResNet18 (Tabelini et al., 2021a)	95.57	3.56	3.01	96.71
LanATT-ResNet 34 (Tabelini et al., 2021a)	95.63	3.53	2.92	96.77
LanATT-ResNet122 (Tabelini et al., 2021a)	96.1	5.64	2.17	96.06
ERF-E2E (Yoo et al., 2020)	96.02	3.21	4.28	96.25
ERFNet-HESA (Lee et al., 2022)	96.01	3.29	4.58	-
Lane-Detection (Yao & Chen, 2022)	95.11	6.95	4.9	-
MLP (Yao et al., 2023)	94.36	5.5	5.6	-
Lane method (Xin et al., 2023)	95.92	2.41	4.29	96.64
HAM (Xie et al., 2023)	96.1	2.29	4.0	96.84
4-head self-attention (Shengli et al., 2023)	95.55	3.39	3.29	-
XR34 (Yanga et al., 2023)	**96.71**	2.82	3.24	-
Res (Yang et al., 2023)	**-**	-	-	97.15

Testing the proposed algorithm on the TuSimple benchmark shows the highest F1 score of (97.19) while the previous best result was registered by Tabelini et al. (2021a) with an F1 score of (96.77). Nevertheless, the presented method recorded one of the best accuracy scores after the one registered by Yanga et al. (2023) and then Hou et al. (2019) with a very small lack of (0.16 %) compared to the best accuracy recorded.

Using the CU-Lane Dataset

Considering seven categories, DLbLD achieved the best F1 score results in five scenarios, indicating the robustness of the proposed method under differing conditions. A new state-of-the-art F1 average total score of 79.02 has been achieved. Table 3 shows the results when utilizing the CU-Lane benchmark.

Table 3
Comparing DLbLD to different methods using the CU-Lane dataset

Method	Average	Normal	Crowd	Dazzle	Shadow	Arrow	Curve	Night
DLbLD (Ours)	**79.02**	**94.03**	70.4	**70.17**	**81.26**	84.75	**77.43**	**75.1**
SCNN (Pan et al., 2018)	71.47	90.60	69.70	58.50	66.90	84.10	64.40	66.10
ENet-SAD (Hou et al., 2019)	71.53	90.1	68.8	60.2	65.9	84	65.7	66

Table 2 *(Continue)*

Method	Average	Normal	Crowd	Dazzle	Shadow	Arrow	Curve	Night
FastDraw (Philion, 2019)	68.4	85.9	63.6	57	69.9	79.4	65.2	57.8
UFAST-ResNet34 (Qin et al., 2020)	73.09	90.7	70.2	59.5	69.3	85.7	69.5	66.7
UFAST-ResNet18 (Qin et al., 2020)	67.99	87.7	66	58.4	62.8	81	57.9	62.1
LanATT-ResNet18 (Tabelini et al., 2021a)	73.98	91.17	72.71	65.82	68.03	87.82	63.75	68.58
LanATT-ResNet 34 (Tabelini et al., 2021a)	76.94	92.14	75.03	66.47	78.15	88.38	67.72	70.72
LanATT-ResNet122 (Tabelini et al., 2021a)	76.4	91.74	**76.16**	69.47	76.31	86.29	64.05	70.81
ERF-E2E (Yoo et al., 2020)	75.46	91	73	64.5	74.1	85.8	71.9	67.9
ERFNet-HESA (Lee et al., 2022)	75.69	92	73.1	63.1	75.1	88.1	68.8	69.6
Lane-Detection (Yao & Chen, 2022)	68.64	88.5	67.6	62.4	62	81.2	56.9	61.9
MLP (Yao et al., 2023)	69.03	89.6	67.1	59.9	60.3	83.0	61.4	61.9
Lane method (Xin et al., 2023)	76.8	92.6	74.9	65.6	75.5	88.2	69.8	70.9
HAM (Xie et al., 2023)	76.47	92.1	74.1	63.1	77.1	88.3	69.3	71.3

Table 2 *(Continue)*

Method	Average	Normal	Crowd	Dazzle	Shadow	Arrow	Curve	Night
4-head self-attention (Shengli et al., 2023)	74.38	91.43	73.56	66.18	66.81	86.79	65.6	70.3
XR34 (Yanga et al., 2023)	78.52	92.83	75.96	69.48	77.86	**88.66**	71.14	73.74
Res (Yang et al., 2023)	74.39	91.3	70.0	61.4	72.1	84.6	74.8	66.5

The results on the CU-Lane dataset show the best F1 score in five scenarios out of seven, including the Normal, Dazzle, Shadow, Curve, and Night cases, with scores of 94.03, 70.17, 81.26, 77.43, and 75.1, respectively. The system can 'see' the road and analyze the image to accurately predict lane line location for all these scenarios. Although the proposed method recorded state-of-the-art results in the cost of the need for a GPU. On the other hand, in the Crowd scenario, the presented paradigm recorded a low F1 score because it could not see the road clearly, whereas other cars were blocking the lane lines. Knowing that the DLbLD algorithm was designed to extract lane features from an image block of the total image and indicate its coordinates to be used later in creating the lane line. So, being unable to see some lane parts blocked by other cars leads to a drop in the efficiency of the presented system.

In the seventh scenario that considered the Arrows case, the DLbLD algorithm registered a low F1 score; the reason behind that is to return to the design idea of the proposed scheme where it searches for lane parts in the incoming blocks and replaces them with a single point located on that line part to be utilized later in forming the related lane line, and not searching for Arrows drown on the road, as it is not designed or trained to do so. Results shown in Table 3 indicate that the proposed method is efficient in dealing with most scenarios when the camera can see the road; it also shows the superiority of this method compared to other works. Nevertheless, the introduced system's performance degrades when obstacles, such as cars in the crowd scenario, block the lane lines. In addition, the results depicted in Table 3 show a low F1 score in the arrow case due to the design of the DLbLD algorithm, which was designed to deal with lane lines and not recognize arrows on the road.

Deploying the proposed method on two standard benchmarks with different environmental conditions demonstrates its impact and superiority over the recently available methods in most environments. Even in challenging Night and Shadow conditions, the proposed algorithm illustrates an improvement of more than 6% and almost 4%, respectively, and a total average improvement of 2.7%. From a real-time performance perspective, the proposed system was tested to indicate this metric. First, an Intel® Core™

i7 Control Processing Unit (CPU) was used to run the system and process the incoming images. Results indicated a low FPS rate of 3.2. Another test used the NVIDIA GeForce GTX 1080 card. Real-time performance improved significantly and recorded 67.8 FPS. These results can be interpreted as the effect of using a graphic processing unit (GPU) that handles image processing better than a CPU (Qin et al., 2020). Table 4 compares other works from the FPS point of view.

Table 4
Comparing the speed of Dlbld with GPU to different methods

Method	DLbLD (Ours)	SCNN [12]	ENet-SAD [13]	UFAST-ResNet [19]	LanATT-ResNet18 [1]	ERF-E2E [17]
FPS	67.8	7.5	75	322.5	250	90.31

Table 4 indicates the performance of the proposed algorithm compared to other related works in terms of processing speed using the FPS measure. Although the presented system achieved a 67.8 FPS, which is not the highest in Table 4, it is still considered a very good result as it is more than enough for real-time applications. It is worth mentioning that the recorded FPS can be significantly improved by using a powerful GPU. The proposed paradigm is superior to other systems as a total trade-off between the recorded accuracy and the achieved speed.

CONCLUSION

This work presents a new CCN-based lane detection algorithm. The proposed system utilized a single vision sensor as an input; the camera takes video frames and feeds them to the system. The processing starts by dividing the incoming images into smaller blocks with 20x80 pixels. The blocks are presented to the proposed neural network to check the presence of a lane line segment. The image block will then be classified in a binary base; if the paradigm predicts that the block contains a part of a lane line, that block will be labeled as ‘1’; otherwise, it will be labeled as ‘0’. After applying the Gaussian blur filter, a point-on-line (PO) for each block in the frame is determined. K-mean clustering then assigns each PO of the projected points to the related lane line. The results of applying the proposed DLbLD algorithm to TuSimple and CU-Lane benchmarks indicate its robustness and superiority to recent works. Nevertheless, GPU is needed to speed up the process and give a better FPS rate.

ACKNOWLEDGEMENTS

This work was supported by the (UoD) and (DPU)/Ministry of Higher Education (MHE)/ Kurdistan Region Government (KRG)/Iraq. As a part of the higher studies program/Ph.D. plan for the studying years 2020–2023.

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., … & Zheng, X. (2016, November 2-4). *TensorFlow: A system for large-scale machine learning*. [Paper presentation]. Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI), Savannah, USA.

Abualsaud, H., Liu, S., Lu, D., Situ, K., Rangesh, A., & Trivedi, M. M. (2021). LaneAF: Robust multi-lane detection with affinity fields. *IEEE Robotics and Automation Letters*, *6*(4), 7477–7484. https://doi.org/10.1109/lra.2021.3098066

Al-Jarrah, R., Al-Jarrah, M., & Roth, H. (2018). A novel edge detection algorithm for mobile robot path planning. *Journal of Robotics*, *2018*, Article 1969834. https://doi.org/10.1155/2018/1969834

Aly, M. (2008, June 4-6). *Real time detection of lane markers in urban streets.* [Paper presentation]. IEEE Intelligent Vehicles Symposium, Eindhoven, Netherlands. https://doi.org/10.1109/IVS.2008.4621152

Chen, Y., & Xiang, Z. (2022). Lane mark detection with pre-aligned spatial-temporal attention. *Sensors*, *22*(3), Article 794. https://doi.org/10.3390/s22030794

Chollet, F. (n.d.). *Keras* [Pyhton]. Github.com. https://github.com/keras-team/keras

He, K., Zhang, X., Ren, S., & Sun, J. (2016, June 27-30). D*eep residual learning for image recognition.* [Paper presentation]. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, USA. https://doi.org/10.1109/cvpr.2016.90

Hou, Y., Ma, Z., Liu, C., & Loy, C. C. (2019, October 27-November 2). *Learning lightweight lane detection CNNs by self attention distillation.* [Paper presentation]. IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea. https://doi.org/10.1109/iccv.2019.00110

Jiang, R., Klette, R., Vaudrey, T., & Wang, S. (2009, September 2-4). N*ew lane model and distance transform for lane detection and tracking*. [Paper presentation]. Computer Analysis of Images and Patterns: 13th International Conference (CAIP), Munster, Germany. https://doi.org/10.1007/978-3-642-03767-2_127

Kim, Z. (2008). Robust lane detection and tracking in challenging scenarios. *IEEE Transactions on Intelligent Transportation Systems*, *9*(1), 16–26. https://doi.org/10.1109/TITS.2007.908582

Ko, Y., Lee, Y., Azam, S., Munir, F., Jeon, M., & Pedrycz, W. (2022). Key points estimation and point instance segmentation approach for lane detection. *IEEE Transactions on Intelligent Transportation Systems*, *23*(7), 8949–8958. https://doi.org/10.1109/tits.2021.3088488

Lee, D. G., & Kim, Y. K. (2022). Joint semantic understanding with a multilevel branch for driving perception. *Applied Sciences*, *12*(6), Article 2877. https://doi.org/10.3390/app12062877

Lee, M., Lee, J., Lee, D., Kim, W., Hwang, S., & Lee, S. (2022, January 3-8). *Robust lane detection via expanded self attention.* [Paper presentation]. IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), Waikoloa, USA.

Liu, G., Worgotter, F., & Markelic, I. (2010, June 21-24). *Combining statistical Hough transform and particle filter for robust lane detection and tracking*. [Paper presentation]. IEEE Intelligent Vehicles Symposium, La Jolla, USA. https://doi.org/10.1109/IVS.2010.5548021

Liu, L., Chen, X., Zhu, S., & Tan, P. (2021, October 10-17). *CondLaneNet: A top-to-down lane detection framework based on conditional convolution*. [Paper presentation]. IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, Canada. https://doi.org/10.1109/iccv48922.2021.00375

Liu, R., Yuan, Z., Liu, T., & Xiong, Z. (2021, January 3-8). *End-to-end lane shape prediction with transformers*. [Paper presentation]. IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), Waikoloa, USA. https://doi.org/10.1109/WACV48630.2021.00374

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, *28*(2), 129–137. https://doi.org/10.1109/TIT.1982.1056489

Loshchilov, I., & Hutter, F. (2017). *SGDR: Stochastic gradient descent with warm restarts*. arXiv. https://doi.org/10.48550/arXiv.1608.03983.

Neven, D., De Brabandere, B., Georgoulis, S., Proesmans, M., & Van Gool, L. (2018, June 26-30). *Towards end-to-end lane detection: An instance segmentation approach*. [Paper presentation]. IEEE Intelligent Vehicles Symposium (IV), Changshu, China. https://doi.org/10.1109/ivs.2018.8500547

Pan, X., Shi, J., Luo, P., Wang, X., & Tang, X. (2018). Spatial as deep: Spatial CNN for Traffic scene understanding. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1), Article 7277. https://doi.org/10.1609/aaai.v32i1.12301

Philion, J. (2019, June 15-20). *Fastdraw: Addressing the long tail of lane detection by adapting a sequential prediction network*. [Paper presentation]. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, California. https://doi.org/10.1109/cvpr.2019.01185

Pizzati, F., Allodi, M., Barrera, A., & García, F. (2020). Lane detection and classification using cascaded CNNs. In R. Moreno-Diaz, F. Pichler & A. Quesada-Arencibia (Eds.) *Computer aided systems theory–EUROCAST 2019* (pp. 95–103). Springer. https://doi.org/10.1007/978-3-030-45096-0_12

Qin, Z., Wang, H., & Li, X. (2020). Ultra fast structure-aware deep lane detection. In A. Vedaldi, H. Bischof, T. Borx & J. Frahm (Eds.) *Computer vision–ECCV 2020* (pp. 276–291). Springer. https://doi.org/10.1007/978-3-030-58586-0_17

Rossum, G. V., & Drake, F. L. (1995). *Python reference manual*. Centrum voor Wiskunde en Informatica.

Shengli, F., Yuzhi, Z., & Xiaohui, B. (2023). Lane marker detection based on multihead self-attention. *Mobile Information Systems*, *2023*, Article 2075022. https://doi.org/10.1155/2023/2075022

Tabelini, L., Berriel, R., Paixão, T. M., Badue, C., De Souza, A. F., & Oliveira-Santos, T. (2021a, June 20-25). *Keep your eyes on the lane: Real-time attention-guided lane detection*. [Paper presentation]. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, Tennessee. https://doi.org/10.1109/cvpr46437.2021.00036

Tabelini, L., Berriel, R., Paixão, T. M., Badue, C., De Souza, A. F., & Oliveira-Santos, T. (2021b, January 10-15). *Polylanenet: Lane estimation via deep polynomial regression*. [Paper presentation]. 25th International Conference on Pattern Recognition (ICPR), Milan, Italy. https://doi.org/10.1109/icpr48806.2021.9412265

Tang, J., Li, S., & Liu, P. (2021). A review of lane detection methods based on deep learning. *Pattern Recognition*, *111*, Article 107623. https://doi.org/10.1016/j.patcog.2020.107623

Xie, T., Yin, M., Zhu, X., Sun, J., Meng, C., & Bei, S. (2023). A fast and robust lane detection via online re-parameterization and hybrid attention. *Sensors, 23*(19), Article 8285. https://doi.org/10.3390/s23198285

Xin, L., Yingping, H., & Zhenming, L. (2023). Axial attention-guided anchor classification lane detection. *Opto-Electronic Engineering*, *50*(7), Article 230079-1. https://doi.org/10.12086/oee.2023.230079

Xu, H., Wang, S., Cai, X., Zhang, W., Liang, X., & Li, Z. (2022). Curvelane-NAS: Unifying lane-sensitive architecture search and adaptive point blending. In A. Vedaldi, H. Bischof, T. Brox & J. Frahm (Eds.) *Computer vision–ECCV 2020* (pp. 689–704). Springer. https://doi.org/10.1007/978-3-030-58555-6_41

Yang, Q., Ma, Y., Li, L., Su, C., Gao, Y., Tao, J., Huang, Z., & Jiang, R. (2023). Lightweight lane line detection based on learnable cluster segmentation with self-attention mechanism. *IET Intelligent Transport Systems*, *17*(3), 522–533. https://doi.org/10.1049/itr2.12277

Yanga, J., Zhang, L., & Lu, H. (2023). Lane detection with versatile atrousformer and local semantic guidance. *Pattern Recognition*, *113*, Article 109053. https://doi.org/10.1016/j.patcog.2022.109053

Yao, X., Wang, Y., Wu, Y., He, G., & Luo, S. (2023). MLP-Based efficient convolutional neural network for lane detection. *IEEE Transactions on Vehicular Technology*, *72*(10), 12602-122614. https://doi.org/10.1109/TVT.2023.3275571

Yao, Z., & Chen, X. (2022). Efficient lane detection technique based on lightweight attention deep neural network. *Journal of Advanced Transportation*, *2022*, Article 5134437. https://doi.org/10.1155/2022/5134437

Yoo, S., Lee, H., Myeong, H., Yun, S., Park, H., Cho, J., & Kim, D. H. (2020, June 14-19). *End-to-end lane marker detection via row-wise classification*. [Paper presentation]. IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Seattle, USA. https://doi.org/10.1109/cvprw50498.2020.00511

Zang, J., Zhou, W., Zhang, G., & Duan, Z. (2018, November 12-15). *Traffic lane detection using fully convolutional neural network*. [Paper presentation]. *A*sia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Honolulu, Hawaii. https://doi.org/10.23919/APSIPA.2018.8659684