Experimental Evaluation of Lane Detection Models in Miniature Autonomous Vehicles

Rachel Fanti Coelho Lima[a], Michele Segata[b], Muhammad Azfar Yaqub[a], Antonio Liotta[a]
[a]Free University of Bozen-Bolzano, Italy
[b]University of Trento, Italy
rachelfanti@gmail.com, michele.segata@unitn.it, {myaqub, antonio.liotta}@unibz.it

Abstract

Lane detection is a critical component of autonomous vehicles, and essential for ensuring their safe and efficient navigation. In recent years, several models have been proposed to improve the accuracy and performance of this activity, however, little is known about practical real-time implementations of these models in small-scale vehicles and low-cost devices, with limited computing and memory resources. This work presents an overview of the models and techniques used for lane recognition in autonomous vehicles. Additionally, an experimental evaluation was conducted for testing different models for lane detection in real-time on miniature cars, identifying the best-performing models. Three methods were analysed: (1) a model using traditional image processing techniques, (2) an end-to-end deep learning (DL) model proposed by Nvidia, which directly predicts angles, and (3) an ultra-fast lane detection DL model, which predicts lane boundaries. In summary, the results demonstrate the adequacy of DL models for addressing the lane detection recognition, even on low-cost devices, as long as minimum capacity and memory requirements are met. It was observed that the ultra-fast lane detection DL model outperformed the other 2 methods with predicting accuracy of 95% compared to 86% and 92% by traditional and deep learning method by Nvidia, respectively.

1 Introduction

The autonomous vehicle (AV) industry has attracted significant investment from manufacturers and technology companies, with approximately $206 billion spent between 2010 and 2020 on AVs and smart mobility technologies [4]. However, critical steps need to be overcome before the adoption of completely autonomous vehicles. It is necessary to increase the reliability of current systems, develop and implement appropriate regulations, review the traffic code, and define liability for road accidents. Furthermore, the success of automated vehicle technology depends on the strategies adopted and collaboration between different disciplines and stakeholders. Questions on how to deal with the coexistence of human-driven and self-driving cars, how to deal with security issues, such as hackers and criminal actions on the system, and how to prevent significant harm to life and property must be evaluated.

While these challenges have yet to be overcome, advanced driver assistance systems (ADAS) have already been made available to consumers. These features offer several functionalities, such as lane centering, automatic lane changes, adaptive cruise control, semi-autonomous navigation for specific lanes, self-parking, and smart summon. It is estimated that in the last decade $36 billion were invested in ADAS components [4].

The edge-cloud computing continuum that is explored in various research areas, can play a crucial role in addressing these challenges and advancing the capabilities of autonomous vehicles [10, 5]. The performance of the vehicular networks and ADAS in autonomous vehicles, can be further enhanced by utilizing the edge resources and cloud computing. Cloud computing support data analysis and complex tasks which require substantial resources while Edge computing allows for real-time data processing within the edge devices (vehicles). Moreover, the continuum provides management/coexistence between autonomous vehicles and human-drivers and addresses security concerns.

Lane detection and motion planning play a relevant role in the performance of ADAS functionalities, being crucial for the autonomy of these cars. It allows autonomous vehicles and robots to understand their position and road boundaries while navigating safely and efficiently. Miniature car prototypes have been designed and implemented by researchers in academia and industry to test autonomous driving, enabling the evaluation of new products and systems under more realistic scenarios, with smaller budgets and in a safer environment. Moreover, the use of low-cost devices has led to an increase in the development of new technologies and applications, which include lane recognition in miniature cars, in robots for cargo handling, inspection, or maintenance activities; in traffic lights and radars for traffic control systems in simpler autonomous vehicles such as bicycles and scooters and in-car warning frameworks. Such systems can contribute to more affordable and accessible autonomous mobility and

Figure 1. The self-driving robotic car and the road lane

road safety systems, leading to new possibilities for innovation in this area.

This work aims to investigate the current state of the field of lane detection models for miniature cars or low-cost devices. It will analyse practical experiments that have been conducted in the field, including the methods, devices, hardware, systems, and metrics that have been used. The study will also test three lane detection models for use in real-time applications and evaluate their performance. Each model will be analyzed and optimized to enhance its robustness and accuracy in providing results.

The experiment was conducted in the following manner: first, a study was conducted to investigate the current state of the art in the field and understand the practical solutions adopted, as well as their challenges and limitations. Based on this mapping, three lane detection algorithms were selected for empirical evaluation using miniature cars (Figure 1).

Thus, the remainder of this research can be divided into several sections. An overview of relevant works in the area is provided in section 2. Section 3 describes the methodologies and the experimental setup used for model evaluation. It also describes the steps taken to ensure accuracy and reliability of the results. The analysis of each model is presented in section 4, while section 5 summarizes the major conclusions and presents some outlooks for further work.

2 Related Work

Most of the studies apply classical image processing methods to identify lane segments [12, 9]. However, these methods do not respond well to scenes with significant lighting and contrast variations, shadows, and obstructions [13]. Furthermore, these models rely on a combination of techniques and configurations, which vary with the diversity of scenarios, demonstrating scalability issues.

Several Deep Learning (DL) models for lane detection have been proposed to address these problems [8, 15], including architectures such as Convolutional Neural Networks (CNNs) [1, 11], Long Short Term Memory (LSTMs) [17], and generative adversarial networks (GANs) [2, 7].

New deep-learning models have increasingly been proposed to enhance lane detection accuracy and performance, as observed on the "Papers with Code" website. Among their best-performing benchmark models, are: (1) the Cross-Layer Refinement Network for Lane Detection (CLRNet), which utilizes both high and low-level features for lane detection and uses Residual Neural Network (ResNet) or Deep Layer Aggregation (DLA) as pre-trained backbones [16]; (2) the SCNN_UNet_ConvLSTM2, a hybrid spatial-temporal deep learning architecture for lane detection; (3) the CondLaneNet, a top-to-down lane detection framework based on conditional convolution [6]; and (4) YOLOPv2, a multi-task learning network to simultaneously perform the task of traffic object detection, drivable road area segmentation and lane detection [3].

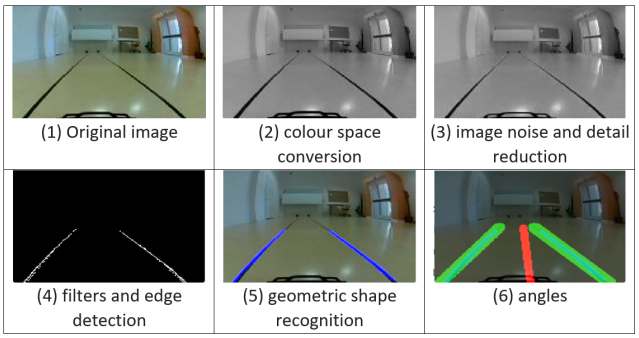

Figure 2. Model 1, the Traditional Model, which uses image processing techniques for feature extraction

However, usually, to apply in real-time applications, these models require to run in high-end GPU platforms. Little is known about DL lane detection algorithms in embedded systems with limited computing and memory resources. Furthermore, DL models in miniature cars can require data and labels from similar environments for training, representing the correct steering angles or road markings for the images. However, these labels must be produced manually or semi-manually, which is a time-consuming task.

An important observation obtained from the mapped studies with practical lane detection implementations in low-cost devices is that comparisons among models and results of these studies is challenging due to the different methods employed, variation in accuracy calculation criteria, and a variety of hardware used. This reinforces the importance of this study, which provides a thorough comparison of different models, isolating the hardware effects, which serves as a valuable reference for selecting a solution for a specific device.

3 Lane Detection Models

This section describes the selected three different real-time lane detection models and the experimental setup used to evaluate their accuracy.

3.1 A traditional model

This model uses feature extraction techniques to detect lanes on the road. It employs traditional image processing techniques to recognize lanes, which are among the most widely used methods for this task. The reason for choosing this model is that these techniques are well-established and have been shown to be effective in lane detection.

This model employs a variety of feature extraction techniques to detect lane segments including color space conversion, image noise and detail reduction, filters, edge detection, geometric shape recognition, and polygon reconstruction. Additionally, the model uses trigonometry to calculate steering angles. Photos of each step in the process were taken and are presented in Figure 2.

Stage	Function
Colour space conversion	cv2.cvtColor, cv2.COLOR_BGR2GRAY
Image noise and detail reduction	{cv2.morphologyEx, } cv2.MORPH_HITMISS, cv2.MORPH_DILATE, cv2.threshold, cv2.THRESH_OTSU, cv2.adaptiveThreshold, cv2.ADAPTIVE_- THRESH_GAUSSIAN_C
Filters and edge detection	cv2.divide, cv2.bitwise_not, A customized function to define the region of interest
Geometric shape recognition	cv2.HoughLinesP A customized function to detect left and right segments
Angles	A customized function to detect angles and avoids abrupt variations of them

Table 1. Functions used in the Traditional Model Algorithm

To build the model, several functions were analysed to identify which ones were most effective in extracting the image features and detecting the lanes for the controlled environment of the experiment. Table 3.1 presents the most effective functions employed in each stage.

3.2 An end-to-end model

This model proposed was by proposed by Nvidia [1] which uses a CNN to directly predict the angle of the car. This model receives an input image of 66x200 pixels and executes a regression task through a CNN, which consists of 9 layers (one normalization layer, 5 convolutional layers, and 3 fully connected layers). The first three convolutional layers use a 2×2 stride and a 5×5 kernel, while the final two convolutional layers are non-strided convolution with a 3×3 kernel size (Figure 3).

In this model the system learns the car's steering angle directly, without the need to first detect the lane and plan the route at hand. This was one of the reasons for choosing this model. It is a straightforward solution, which streamlines the steps and maybe leads to better performance in simpler systems, which is the case. Additionally, this was the most commonly referenced machine learning model in the mapping study conducted.

To pre-process the images before running the model, the following steps were followed as suggested by [14]:

- The top half of the images were removed, as they were not relevant for lane following.
- The images were converted to the YUV colour space.
- Denoising was performed, using *Gaussian Blur*, removing noise from the data.
- Images were resized to (200, 66, 3) and normalized (divided by 255).

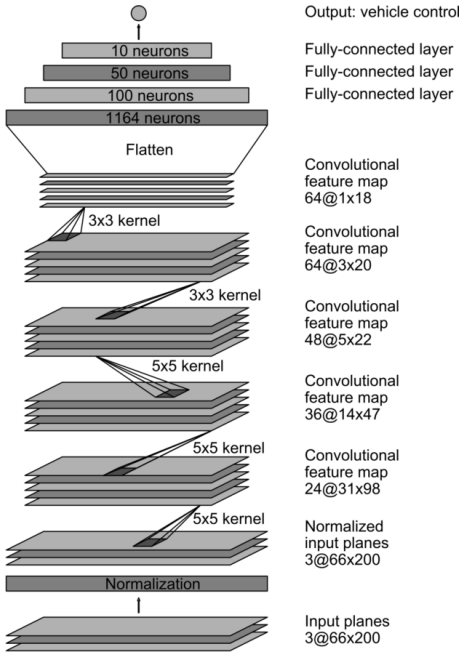

Figure 3. Model 2, the End-to-end Model, a CNN architecture, which directly outputs the steering angle of the car [1]

3.3 An ultra-fast model

The ultra-fast Lane detection model [11] is a CNN model which uses Resnet 18 as the backbone. The architecture contains residual blocks with feature extractors (blue box); an auxiliary branch with a segmentation task used only for training (orange box); and a module which selects predefined cells in rows and executes a classification task (instead of segmenting each pixel of the image), and thus, reducing the computational cost (green box) (Figure 4).

In this method, the images are arranged in a grid pattern and the lanes are represented as locations on predefined rows, referred to as row anchors, along the y-axis, and in cells along the x-axis. The method predicts the probability of selecting gridding cells for the i-th lane, j-th row anchor.

To pre-process the images before running the model, the following steps were followed, as suggested by [11]: (1) Images are resized to 288 pixels of height and 800 pixels of width, in RGB format. (2) Transformed to tensor. (3) Normalized using the mean and standard deviation of Imagenet.

The reason for selecting this model is its high processing speed and low computational cost, which allows it to be trained and tested on a standard machine. Moreover, it can be used in demanding scenarios such as those involving road obstructions and curved roads. This is due to the model's unique characteristics and its loss calculation formula, which incorporates both global and local features.

The local feature approach is a way to identify and extract relevant information from images focuses on their small sections. For example, for training, the model performs an auxiliary task and compute the loss formula, segmenting each pixel of the lanes.

On the other hand, another branch used for training and

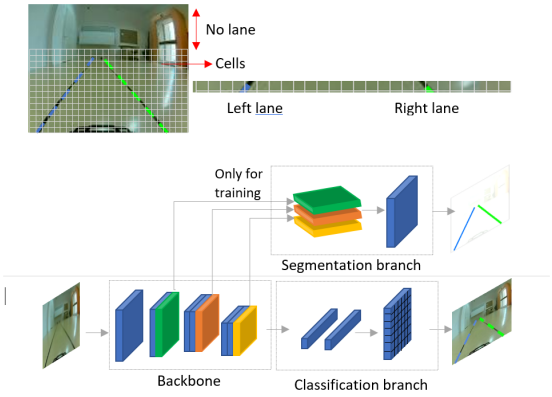

Figure 4. Model 3, the Ultra-fast Model, a CNN architecture, which 2 branches, one for auxiliary segmentation and another for classification

testing performs a classification task, applying a global feature approach. Instead of analyzing each individual pixel, this approach selects the location of the lanes on a grid, which reduces the computational cost compared to the local feature approach and uses a broader understanding of the entire image. This seeks to ensure the continuity of lanes and the definition of their shape.

3.4 Experimental Research

The main objective of this experimental research is to compare three different real-time lane detection models for miniature cars and low-cost devices presenting their performance, with respect to their accuracy, memory consumption and processing rate. This study provide insights into the relationship among accuracy, memory, and FPS to help select the best model for specific needs and constraints.

To train the DL models, this research utilized an approach for automatically estimating the labels, thereby streamlining the process and saving time by using the results of the traditional model to train the other two deep learning models.

The experiments were conducted using a miniature car on a designed track in a controlled environment (Figure 1). The model tests were performed on a laptop in a environment simulating the same characteristics of this setting in which the data were acquired.

The self-driven miniature car was built by students and researchers from the Free University of Bozen-Bolzano, following the recommendations suggested by Donkey Car, an open source self-driving car platform for small scale cars. The main components of the car are presented in Figure 5.

Each model was developed using Python and trained and tested using the image sets. To ensure smooth integration with the other functionalities of the Raspberry Pi and real-time processing of individual images, a dedicated code was written for each model.

3.5 Data collection

To develop model (1), pictures were captured by positioning the car in different road locations . Using this model, the vehicle was driven in real settings capturing numerous pictures, of which 6692 were selected (4679 for training and 2013 for tests). These pictures, in addition to the results of

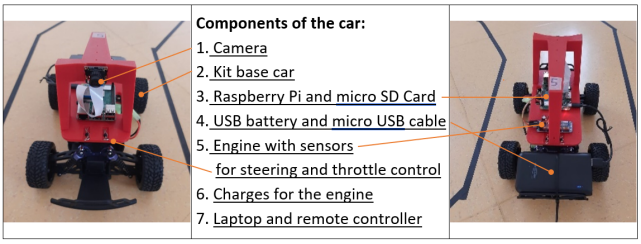

Figure 5. Components of the miniature car

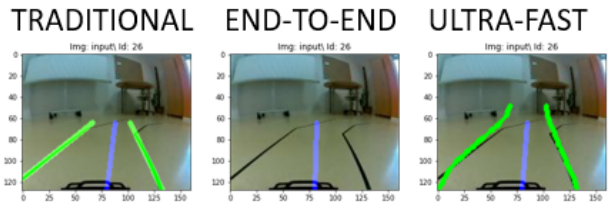

Figure 6. Visualization of the predictions

model (1) were then used as input and labels for predicting steering angles and lines for models (2) and (3). Images and labels that were not correctly predicted in the training of the first model were removed from the training of the other two models.

Two sets of photos were taken of a car driving on a track at different times. To enhance the models' ability to handle challenging scenarios, a diverse set of photos was also included in the experiments, with varied lighting conditions (i.e. sunlight, shadows, darkness) and track geometries (i.e. sharper curves, straight lines, only one track lane visible). To remove redundant images from the dataset, steps were taken before randomly splitting it into training and testing sets. Data augmentation techniques, such as zoom, width and height shift, changes in brightness, and rotation, were used to train the machine learning models and increase their robustness, while avoiding over-fitting.

4 Results and Discussions

A significant emphasis was placed on the analysis of the accuracy because it is a crucial factor in determining the effectiveness and reliability of the models, directly impacting their ability to perform the intended task and deliver accurate results. Hence, it is important to evaluate and compare the accuracy of the models in a thorough and systematic manner.

These are the visualization of the predictions for the 3 models (Figure 6).

A visual inspection was carried out on all the images predicted in the test set for the three models. To avoid bias a line was drawn at one-third of the image's height to check if the car would stay on the track. Even if the prediction was not perfect, it was assumed that the car would have time to adjust its direction in the next frame (Figure 7).

Based on the results presented in table 2, the following considerations can be made:

- All three models performed satisfactorily, with the lowest performing model achieving an average accuracy

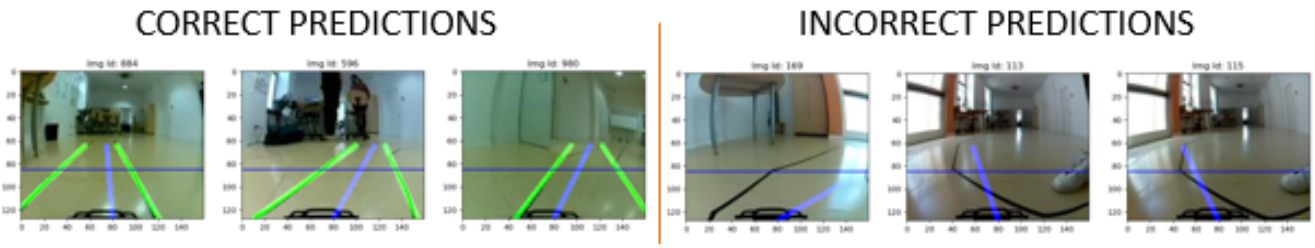

Figure 7. Examples of correct and incorrect predictions for one model

Group	Scenarios	Nº Pict.	Nº Pict. Correct Angle			% Pict. Correct Angle		
			Trad.	End	Ultra	Trad.	End	Ultra
(1)	Dataset 1	680	631	658	680	93%	97%	100%
(2)	Dataset 2	530	404	507	530	76%	96%	100%
(a)	Sunlight incidence	94	94	83	94	100%	88%	100%
(b)	Lighting variance	84	60	76	84	71%	90%	100%
(c)	Darkness	150	134	131	147	89%	87%	98%
(d)	Curves	300	250	257	266	83%	86%	89%
(e)	Diagonal straight	23	18	22	23	78%	96%	100%
(f)	Car off centre line	120	118	82	119	98%	68%	99%
(g)	One line visible	20	12	8	14	60%	40%	70%
		2013	**1731**	**1834**	**1969**	**86%**	**92%**	**95%**

Table 2. Correct angle prediction rates among the different models and scenarios

rate of 86% in determining the correct angles.

- In general, the Traditional Model performed worse than the others. During testing, it was noted that this model struggled in certain conditions where there was variation in illumination, causing the car to lose control at the same track marker on each lap.
- The Ultra-fast Model demonstrated superior accuracy across all testing conditions, while the End-to-end Model exhibited inferior performance in certain situations, including instances of bright sunlight, darkness, off-center vehicle position, or absence of visible lines. To enhance the model's performance in these challenging scenarios, additional training and increased data sampling are recommended.
- The End-to-end Model, which directly estimates steering angles, showed higher autonomy than the Traditional Model, despite being trained on data produced by the Traditional Model. Additionally, the End-to-end Model is more sensitive to the distribution of input images and features compared to the Ultra-fast Model.
- The Ultra-fast Model is more complex, but even with training labels generated by the first model, it was able to achieve better accuracy in determining angles. The automatic label generation did not compromise the identification of the main direction line.

Figure 8 provides a comprehensive overview of the performance of the three models across various metrics, including accuracy, FPS, memory usage, and GPU requirements.It is important to note that the actual FPS may be limited when using a Raspberry Pi camera or any other equipment due to hardware constraints. During our testing, a maximum of 99 frames per second was achieved due to the limitations of the hardware.

The results showed that the Traditional Model had the best performance in terms of memory usage and processing time. The End-to-end Model also performed well in these criteria, however, the Ultra-fast Model had a less favorable outcome regarding processing time, obtaining a rate of 0.18 frames per second when running without GPU, but it achieved a good frame rate of 26 FPS when used with GPU.

5 Conclusions & Future Work

5.1 Concluding Remarks

The results of this study demonstrated the suitability of deep learning models for lane recognition tasks, with the two evaluated deep learning models achieving the highest levels of accuracy compared to the traditional method. The approach used for label generation proved to be effective.

Overall, the End-to-end model was relatively easy to implement and had fast processing speed, but its accuracy is highly influenced by the quality of the training data used. The distribution of the images and their features during training can strongly influence the accuracy of the algorithm. Therefore, it is crucial to ensure the training data used is representative of all types of relevant scenarios and actions.

The Traditional model achieved high accuracy without requiring training or labels and uses less memory than the other two model, but is less scalable. The Ultra-fast model achieved high accuracy, provided ample resources and time. However, this model required more GPU for appropriate processing time and higher FPS. DL methods require label generation, which can require extra time and effort if the labels are not readily available in a comparable setting. For instance, in this experiment, we had to employ the traditional

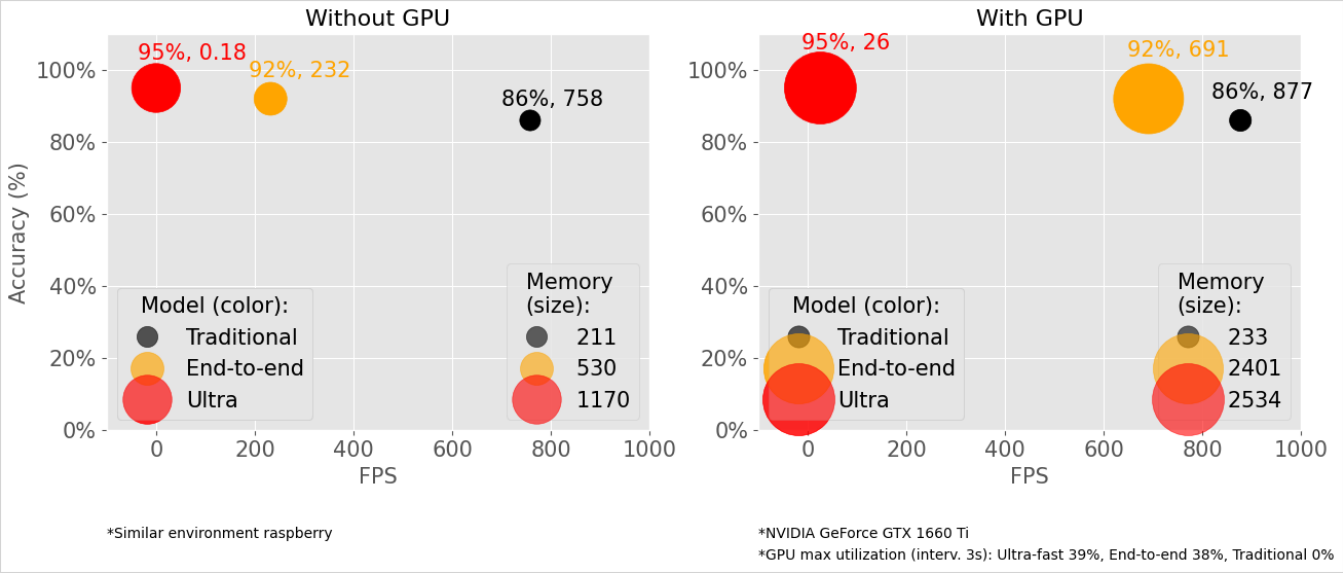

Figure 8. Comparison among the performance of the three models

model to generate the required labels.

Choosing the appropriate model application depends on several factors, including the available time and resources. The results of this study provide a useful guide for selecting the appropriate model based on the requirements of memory, GPU, accuracy, and processing time. In addition to these factors, it is important to consider other non-functional requirements such as the time required for code development and label generation.

5.2 Future Work

For future work, improvements for each algorithm studied are proposed: (1) Enhancing the Traditional Model's ability to detect lanes under varying lighting conditions. (2) Incorporating techniques to handle imbalanced data in the End-to-end Model. (3) Enhancing the Ultra-fast Model algorithm to reduce its memory consumption, and to increase processing speed.

Furthermore, It is recommended investigate alternative deep learning models to identify more efficient and straightforward methods.Finally, other proposals are: developing specialized models; new equipment's; backbones and platforms for low-cost devices.

6 References

[1] M. Bojarski et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[2] M. Ghafoorian, C. Nugteren, N. Baka, O. Booij, and M. Hofmann. El-gan: Embedding loss driven generative adversarial networks for lane detection. In *proceedings of the european conference on computer vision (ECCV) Workshops*, pages 0–0, 2018.

[3] C. Han et al. Yolopv2: Better, faster, stronger for panoptic driving perception. *arXiv preprint arXiv:2208.11434*, 2022.

[4] D. Holland-Letz, M. Kässer, B. Kloss, and T. Müller. Mobility's future: An investment reality check. `https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/mobilitys-future-an-investment-reality-check`, 2021. Accessed: 2023-01-16.

[5] Y. Hui, G. Zhao, C. Li, N. Cheng, Z. Yin, T. H. Luan, and X. Xiao. Digital twins enabled on-demand matching for multi-task federated learning in hetvnets. *IEEE Transactions on Vehicular Technology*, 72(2):2352–2364, 2023.

[6] L. Liu, X. Chen, S. Zhu, and P. Tan. Condlanenet: a top-to-down lane detection framework based on conditional convolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3773–3782, 2021.

[7] Y. Liu, J. Wang, Y. Li, C. Li, and W. Zhang. Lane-gan: A robust lane detection network for driver assistance system in high speed and complex road conditions. *Micromachines*, 13(5):716, 2022.

[8] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool. Towards end-to-end lane detection: an instance segmentation approach. In *2018 IEEE intelligent vehicles symposium (IV)*, pages 286–291. IEEE, 2018.

[9] G. S. Pannu, M. D. Ansari, and P. Gupta. Design and implementation of autonomous car using raspberry pi. *International Journal of Computer Applications*, 113(9), 2015.

[10] E. Peltonen, A. Sojan, and T. Päivärinta. Towards real-time learning for edge-cloud continuum with vehicular computing. In *IEEE 7th World Forum on Internet of Things (WF-IoT)*, pages 921–926, 2021.

[11] Z. Qin, H. Wang, and X. Li. Ultra fast structure-aware deep lane detection. In *European Conference on Computer Vision*, pages 276–291. Springer, 2020.

[12] J. Suto. Real-time lane line tracking algorithm to mini vehicles. *Transport and Telecommunication*, 22(4):461–470, 2021.

[13] J. Tang, S. Li, and P. Liu. A review of lane detection methods based on deep learning. *Pattern Recognition*, 111:107623, 2021.

[14] D. Tian. Deeppicar — part 5: Autonomous lane navigation via deep learning. `https://towardsdatascience.com/deeppicar-part-5-lane-following-via-deep-learning-d93acdce6110`, 2019. Accessed: 2023-01-16.

[15] D. Wu, M.-W. Liao, W.-T. Zhang, X.-G. Wang, X. Bai, W.-Q. Cheng, and W.-Y. Liu. Yolop: You only look once for panoptic driving perception. *Machine Intelligence Research*, pages 1–13, 2022.

[16] T. Zheng, Y. Huang, Y. Liu, W. Tang, Z. Yang, D. Cai, and X. He. Clrnet: Cross layer refinement network for lane detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 898–907, 2022.

[17] Q. Zou, H. Jiang, Q. Dai, Y. Yue, L. Chen, and Q. Wang. Robust lane detection from continuous driving scenes using deep neural networks. *IEEE transactions on vehicular technology*, 69(1):41–54, 2019.