Proceedings of Engineering and Technology Innovation, vol. 34, 2026, pp. 33-46

Comparative Evaluation of Deep Learning Models for Small-Scale Autonomous Driving Under Embedded System Constraints

Korawit Krajangpan[1], Kittinan Petsri[2,*]

[1]College of Industrial Technology, King Mongkut's University of Technology North Bangkok, Bangkok, Thailand

[2]Faculty of Technical Education, King Mongkut's University of Technology North Bangkok, Bangkok, Thailand

Received 07 March 2026; received in revised form 03 July 2026; accepted 08 July 2026

DOI: https://doi.org/10.46604/peti.2026.16240

Abstract

This study compares vision-based autonomous driving models for small-scale vehicles under embedded resource constraints. Three architectures—a linear model, a categorical model, and a convolutional neural network (CNN) 3D Model—are evaluated on the DonkeyCar platform. Training data comprise over 23,000 front-facing images with control commands, which are collected through joystick-based teleoperation. Model performance is benchmarked using quantitative metrics (mean absolute error (MAE), root mean squared error (RMSE), coefficient of determination (R^2)) and driving tests using Raspberry Pi 4. The CNN 3D model achieves the highest prediction accuracy, whereas the linear model provides more stable driving with lower computational cost. The findings indicate that model performance and hardware limits must be considered when selecting models for embedded applications: the linear model is suitable for resource-constrained platforms requiring real-time stability, the categorical model suits discrete decision-making, and the CNN 3D model supports accuracy-prioritized tasks with greater computing capacity.

Keywords: autonomous driving, deep learning, model evaluation, embedded systems

1. Introduction

End-to-end vision-based autonomous driving has attracted increasing research attention over the past decade. This trend has been driven by advances in deep learning that enable direct mapping from camera images to vehicle control commands without relying on hand-crafted features [1-4]. This paradigm demonstrates promising capability in handling complex driving tasks, including lane following, cornering, and speed adaptation in dynamically changing environments [1-2]. Consequently, end-to-end learning emerges as a widely studied alternative to traditional modular perception–planning–control pipelines.

Despite these advances, model evaluation in many existing studies remains largely focused on offline predictive metrics, such as mean absolute error (MAE), root mean squared error (RMSE), and the coefficient of determination (R^2). These metrics provide useful insight into prediction accuracy on held-out datasets. However, they do not fully capture closed-loop driving behavior when models are deployed as real-time controllers, particularly on embedded platforms with limited computational, memory, and thermal resources [1-2]. In practical operation, small prediction errors can accumulate over time in closed-loop control, resulting in oscillatory steering behavior, unstable speed profiles, or eventual track departure. This discrepancy reflects the gap between offline prediction accuracy and closed-loop driving behavior [1-2].

To better understand this issue under realistic deployment conditions, recent studies increasingly adopt small-scale autonomous driving platforms as experimental testbeds [5]. Such platforms offer a cost-effective and safe environment for evaluating end-to-end driving models while explicitly considering resource constraints that are representative of real embedded

* Corresponding author. E-mail address: kittinan.p@fte.kmutnb.ac.th

systems. In this work, three vision-based driving models with different levels of complexity—a linear model, a categorical model, and a CNN 3D model—are developed and trained using the DonkeyCar simulation environment. The trained models are subsequently deployed and evaluated on a physical prototype vehicle equipped with a Raspberry Pi 4, which serves as a widely used embedded computing platform for real-time autonomous driving applications.

To address these limitations, the study is threefold. First, a systematic comparison of vision-based driving models with varying architectural complexity is conducted under identical datasets and evaluation protocols, combining conventional offline metrics with closed-loop autonomous driving tests. Second, the mismatch between offline prediction accuracy and real-time driving stability is examined to highlight the behavioral gap that arises during embedded deployment [1-2]. Third, practical guidance for resource-aware model selection is provided through detailed profiling of CPU utilization, memory usage, and device temperature on the Raspberry Pi 4 platform, drawing on established perspectives in embedded and edge AI research.

The remainder of this paper is organized as follows. Section 2 reviews related work on end-to-end autonomous driving, spatiotemporal modeling, and embedded deployment. Section 3 describes the system architecture, model designs, and experimental setup. Section 4 presents the evaluation results and discussion. Finally, Section 5 concludes the paper and outlines directions for future work.

2. Related Work

Research on vision-based autonomous driving has expanded rapidly, particularly within the end-to-end learning paradigm, which directly learns control policies from camera-based visual inputs [1-4]. Existing studies can be broadly grouped into three main research directions: (i) end-to-end autonomous driving surveys and methodological studies, (ii) spatiotemporal modeling, and (iii) embedded system constraints and edge deployment.

(1) End-to-end autonomous driving research: a substantial body of survey literature has reviewed the evolution of end-to-end autonomous driving, covering architectural developments, learning paradigms such as imitation learning and reinforcement learning, and common evaluation practices [1-4], [6-7]. In particular, prior studies provided a comprehensive overview of how driving policies are learned across different environments and task settings [1-2]. Other works emphasized challenges related to safety and generalization [3-4, 6]. These studies highlight that offline accuracy alone is insufficient for assessing real driving performance, motivating the need for evaluation frameworks that consider control behavior in closed-loop settings. More recently, generative world models have emerged as a promising direction, enabling models to synthesize realistic future driving scenarios [6].

(2) Spatiotemporal models and temporal learning: to improve temporal consistency, many studies incorporate sequential visual information rather than relying on single-frame inputs. Common approaches include three-dimensional convolutional neural networks (3D CNNs), recurrent neural networks (RNNs), and long short-term memory (LSTM) networks that model temporal dependencies [8-11]. Empirical results reported in these works indicate that spatiotemporal models often outperform single-image-based approaches, particularly in scenarios involving sharp turns or dynamic speed adaptation [8, 10]. In addition, several studies have focused on steering angle prediction using CNN–LSTM combinations and optimized regression frameworks, which have demonstrated improved prediction accuracy [12-14]. However, most of these evaluations were conducted in offline settings or on high-performance computing platforms, which differ substantially from the constraints of embedded systems used in small-scale autonomous vehicles. Moreover, spatiotemporal models—especially 3D CNNs and transformer-based architectures—generally require higher computational complexity, memory, and inference latency. These characteristics pose significant challenges for deployment on resource-constrained embedded devices without careful optimization [15-18]. As a result, many studies demonstrate strong numerical performance but provide limited analysis of practical deployability under realistic embedded conditions.

(3) Embedded system constraints and edge deployment: recent research increasingly addresses the deployment of deep learning models on embedded platforms. Survey and review papers analyze trade-offs between model accuracy and constraints such as processing speed, memory usage, energy consumption, and thermal limits [14-17]. These works also discuss optimization techniques, including model compression, lightweight architectures, and hardware-aware design, as well as the role of accelerators and on-device AI frameworks [15-17].

Despite this progress, much of the embedded AI literature focuses on perception tasks such as image classification or object detection rather than end-to-end autonomous driving [19]. Practical deployment studies on commonly used embedded boards further demonstrate that runtime environments and inference frameworks can significantly influence real-time performance, underscoring the importance of explicit resource profiling during deployment [19]. Consequently, studies that directly compare multiple end-to-end driving model architectures under identical datasets and embedded constraints remain relatively limited.

In addition to the three main directions above, recent work has addressed motion planning for autonomous electric vehicles operating in dynamic environments. For instance, approaches based on sine-resistance networks have demonstrated effective trajectory generation under complex obstacle configurations, highlighting the importance of integrating robust planning algorithms with perception systems [20]. While such methods focus on higher-level path planning rather than end-to-end pixel-to-control learning, they underscore the broader challenge of ensuring safe and stable vehicle behavior under real-world dynamic conditions—a challenge that is equally relevant to the embedded deployment context examined in the present study.

In summary, while prior research has established the feasibility of end-to-end autonomous driving and spatiotemporal learning, there remains a lack of holistic evaluation frameworks that jointly consider prediction accuracy, closed-loop control behavior, and deployability on embedded platforms. This gap directly motivates the evaluation design adopted in the present study.

3. Procedure

As illustrated in Fig. 1, the study proceeds through five main phases. The first phase focuses on preparing the tools required for developing the prototype system. This preparation is divided into two components: hardware and software.

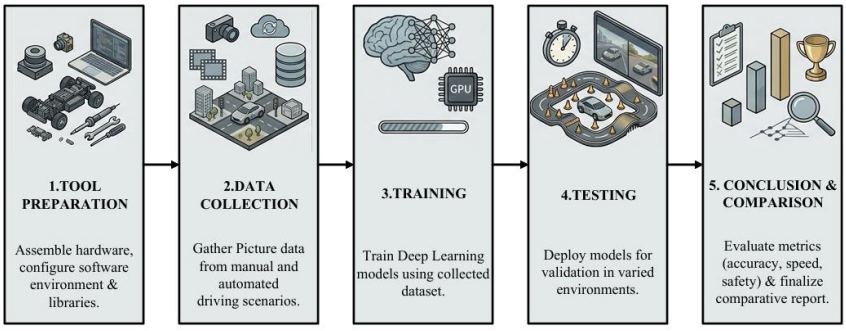


Fig. 1 Research methodology

On the hardware side, the researchers utilize a small-scale vehicle simulation platform along with a simulated test track to evaluate the performance of the autonomous driving models. On the software side, Raspberry Pi OS and the DonkeyCar platform are installed on a Raspberry Pi board. Simultaneously, the development environment on a personal computer is configured for model training, including Python, TensorFlow, and other tools required for data processing and model training within the DonkeyCar framework. Supporting software, such as Visual Studio Code, SSH connection tools, and image management utilities, are also installed to ensure an efficient development and testing workflow and to minimize potential errors in subsequent stages.

Upon completing the hardware and software configuration, the study enters the second phase, which involves the collection of training data for the autonomous driving models. This process begins with configuring the operating system on the Raspberry Pi board to ensure that the prototype vehicle can receive motion control commands accurately and continuously. A joystick (Joy Controller) is selected as the control interface due to its high precision and real-time responsiveness; these attributes render it well-suited for capturing human driving behavior, which serves as the reference data for subsequent model learning.

In the second phase, data collection takes place within the simulated track environment. The researchers manually control the vehicle for approximately 17 minutes, completing around 30 laps of the track. This process yields over 23,000 images captured from the vehicle's front-facing camera, together with motor control commands, including steering angles and throttle levels. These data are recorded in javascript object notation (JSON) format, incorporating timestamps and session information. In addition, overhead images are recorded to verify the actual driving trajectories for each lap. These three data sources—front camera images, control command data, and overhead images—play a crucial role in constructing a comprehensive dataset that captures diverse driving behaviors and track characteristics. This contributes to improved model training accuracy and enhances the ability of the models to handle real-world driving scenarios more effectively.

Following data collection, the next stage involves the development and training of artificial intelligence models for autonomous driving. An overview of the model development and training process is presented in Fig. 2, which consists of five key steps: data collection, data preparation and preprocessing, model development, performance evaluation, and deployment readiness. These five steps are interconnected through an iterative process, allowing continuous refinement and optimization of the models to achieve the highest possible accuracy.

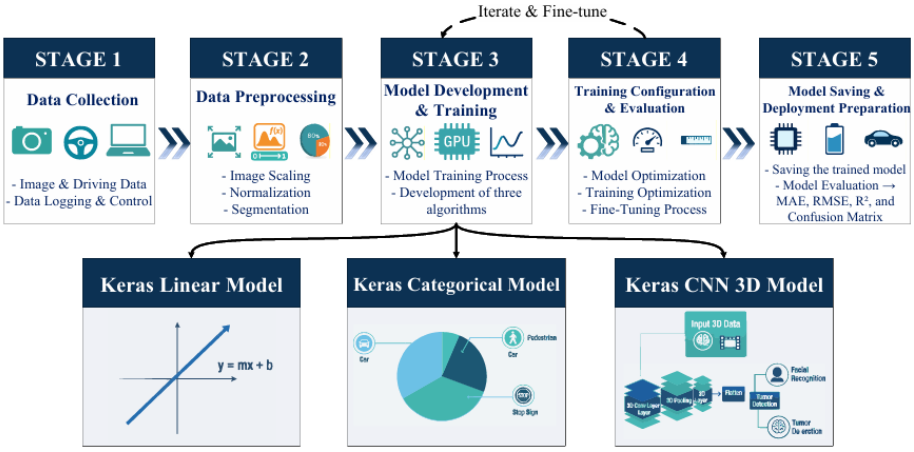


Fig. 2 System development process

In Phase 3, the study designs model architectures and selects appropriate learning parameters to enable the autonomous vehicle to learn driving behavior effectively. Three different models are developed: the linear model, the categorical model, and the CNN 3D model. Each model employs a distinct architecture aligned with its specific data processing objectives. Figs. 3-5 illustrate the systematic development workflow of each model type, enabling a clear comparison of performance across architectures with different levels of complexity.

(1) Linear model: as illustrated in Fig. 3, this linear model consists of a sequential architecture beginning with an input layer that receives images with a resolution of 120 × 160 pixels. The input data are processed through Conv2D layers to extract basic visual features, followed by a Flatten layer that transforms the feature maps into a one-dimensional representation. The resulting features are then passed through fully connected Dense layers before reaching the output layer, which predicts continuous control values such as steering angle and throttle levels. Owing to its low architectural complexity, this model serves as a suitable baseline for evaluating how increased model complexity in more advanced architectures influences prediction accuracy.

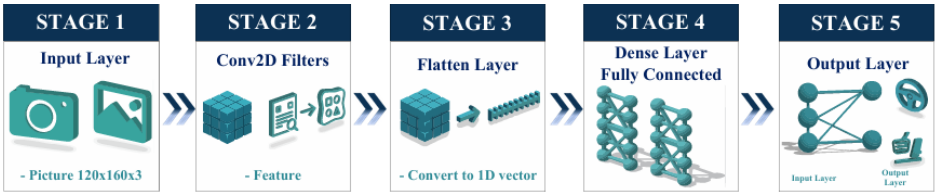


Fig. 3 Structure and processing stages of the linear model

(2) Categorical model: as illustrated in Fig. 4, this model adopts a classification-oriented architecture. Similar to the first model, it begins with an input layer; however, the input images are transformed into vector representations using a Flatten layer. The vectors are then processed through multiple Dense layers to learn discriminative data patterns. The model utilizes a Softmax function at the output layer, enabling the classification of driving behaviors—such as turning left, turning right, or moving straight—into predefined classes. One-hot encoding is applied to the training labels, allowing the system to learn directional behaviors from visual inputs in a categorical manner. This architecture is well-suited for decision-making tasks that require discrete behavior classification rather than continuous control prediction.

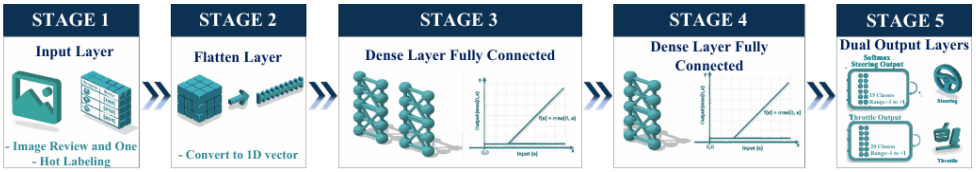


Fig. 4 Structure and processing stages of the categorical model

(3) CNN 3D model: as illustrated in Fig. 5, this is designed to process sequences of consecutive images—a sequence of five frames—to learn spatiotemporal relationships simultaneously. The architecture begins with Conv3D layers that extract deep motion-related features, followed by MaxPooling3D layers to reduce the dimensionality of the feature representations. The extracted features are then passed through multiple Dense layers to aggregate salient patterns. Finally, the model produces predictions through a dual output layer, which independently estimates steering angle and throttle values. This type of model offers high potential for learning dynamic driving behaviors, such as cornering and acceleration, by explicitly capturing temporal changes in visual inputs across successive time steps.

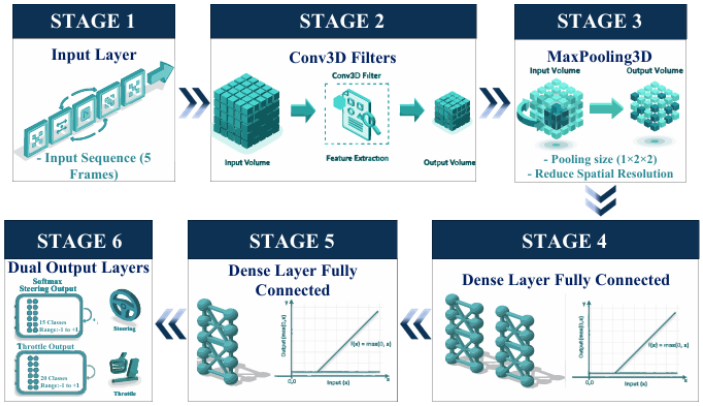


Fig. 5 Structure and processing stages of the CNN 3D model

All three models are trained using preprocessed image data and control command data, with preprocessing steps including image resizing, normalization, and dataset splitting. Training is performed using graphics processing unit (GPU) acceleration to achieve faster convergence and improved training stability. Upon completing the model design and architectural definition in Phase 3, the system development process proceeds to Phase 4: training configuration and evaluation, which aims to ensure stable learning behavior and alignment with the characteristics of autonomous driving tasks. In this phase, key training components are meticulously configured, including the selection of appropriate loss functions based on model type— such as MSE for continuous value prediction and Categorical Cross-Entropy for classification-based models— as well as the choice of optimizer, learning rate, number of training epochs, and batch size.

Throughout the training process, the learning progress of each model is continuously monitored using loss values and performance metrics on both training and validation datasets. This enables the assessment of learning trends and the detection of potential issues, such as overfitting or underfitting. The evaluation results obtained in this stage are subsequently used for parameter fine-tuning. This allows the models to effectively learn autonomous driving behaviors and to systematically assess the impact of architectural complexity on overall system performance prior to deployment.

Once the models achieve satisfactory performance through training and fine-tuning, the final step involves model saving and deployment preparation. The trained models are saved by preserving the model architectures, learned weights, and associated parameters, enabling reuse or further refinement without retraining from scratch. Furthermore, the compatibility of model input and output formats with the DonkeyCar control system is verified to ensure seamless integration into the autonomous driving decision-making pipeline. This stage also includes evaluating computational performance under real-time operation constraints, such as inference speed and response continuity, to ensure reliable deployment during testing while closely reflecting real-world operating conditions.

Following the completion of the model development and training process, the research proceeds to the testing and evaluation phase, which systematically compares the performance of the models under identical conditions. The evaluation consists of two main components: quantitative evaluation using metrics—such as MAE, RMSE, and the R^2—to measure prediction accuracy and error magnitude; and practical evaluation, in which the trained models are deployed within the DonkeyCar simulation environment to assess driving stability, lane-following capability, and responsiveness to dynamic scenarios such as cornering and speed adjustment. The results obtained in this phase enable a clear comparison of the strengths and limitations of each model.

The final stage of the research process comprises the conclusion and discussion, in which the evaluation results are comparatively analyzed across the three model architectures. The analysis considers prediction accuracy, driving stability, suitability for real-time operation, and system resource utilization, allowing the identification of the most effective model for small-scale autonomous driving applications. The comparative findings provide insights to summarize the key contributions of the study and to propose guidelines for selecting appropriate models for practical deployment. Furthermore, the analysis establishes a foundation for future research, including architectural improvements and the extension of applications to more complex and challenging driving scenarios.

4. Results

The results obtained from the development and testing of the autonomous driving models, following the defined research methodology, are organized into three main components: (i) model training outcomes, (ii) model performance evaluation results, and (iii) operational testing results within a simulated environment. These components collectively reflect the performance of each model type across different evaluation dimensions.

The training outcomes for the linear model, the categorical model, and the CNN 3D model are presented in Table 1. This table summarizes training loss values, training time, and the number of training epochs, highlighting differences in model complexity and computational burden during training across the three architectures.

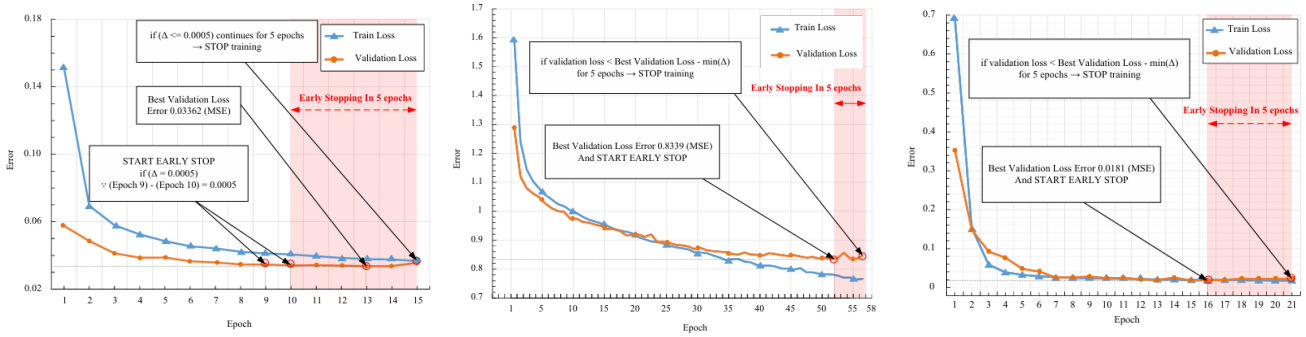
All models are trained on a personal computer equipped with a 13th Gen Intel® Core™ i7-13700KF processor (3.40 GHz), 16.0 GB of main memory (15.8 GB usable), and an NVIDIA GeForce RTX 3080 GPU with 10 GB of dedicated memory. The computational environment is executed via the Windows Subsystem for Linux (WSL) to support artificial intelligence libraries in a Linux-based framework. The hardware and software specifications directly influence training duration and resource utilization, which are critical factors in assessing the suitability of each model for real-world deployment.

Table 1 Summary of model training results

No.	Training metric	Model			Unit
		Linear	Categorical	CNN 3D	
1	Training loss (per model loss function)	0.03362 (MSE)	0.8339 (Cross-Entropy)	0.0181 (MSE)	-
2	Training time	9	20	10	minutes
3	Number of training epochs	15	57	21	epochs

Table 1 indicates that the CNN 3D model achieves the lowest training loss, whereas the linear model requires the shortest training time and the fewest training epochs. In contrast, the categorical model exhibits the highest training resource consumption when compared with the other models.

To provide a clearer understanding of the learning behavior of each model, the researchers present training loss curves illustrating the evolution of loss values over the training process. An early stopping mechanism (patience = 5 epochs) is employed to terminate training when the loss no longer decreases, and the point at which each model attains its minimum loss value is highlighted in Fig. 6.

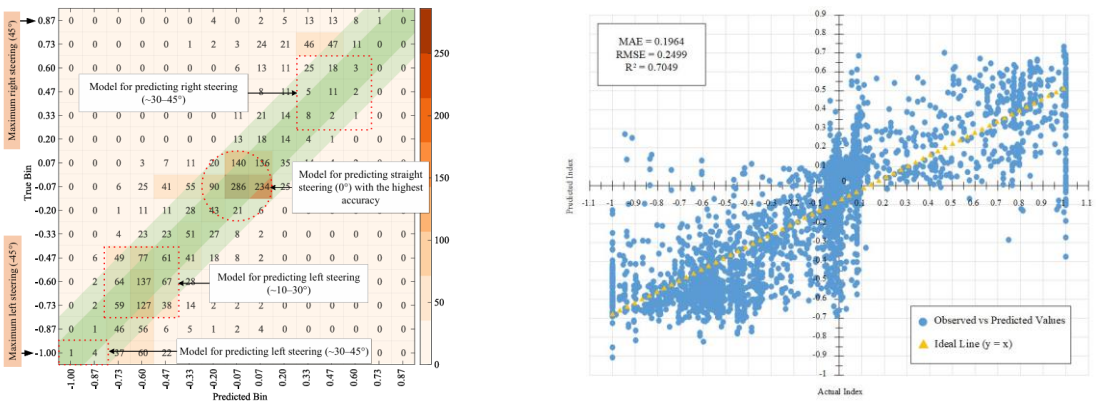


(a) Loss model linear (b) Loss model categorical (c) Loss model CNN 3D

Fig. 6 Illustration of the training loss curve and early stopping mechanism

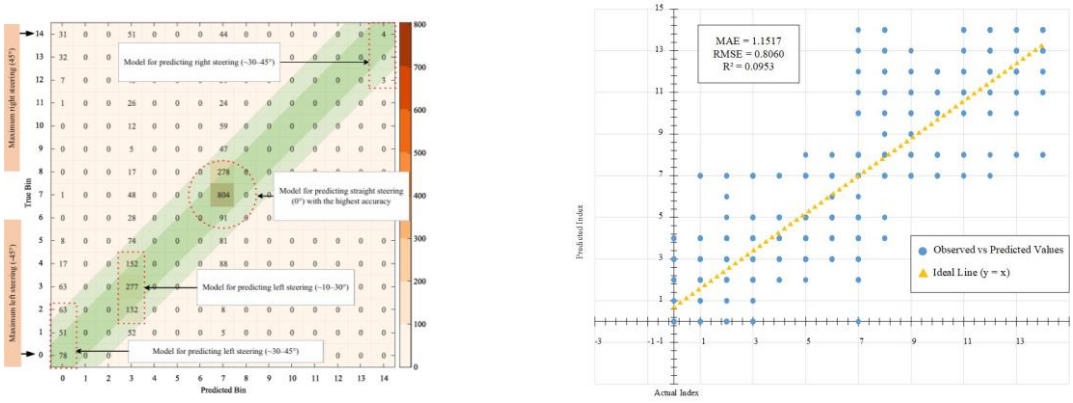
After completing the training of all three model architectures, the study conducts a performance evaluation to assess each model's generalization capability and prediction accuracy on unseen data. This evaluation aims to compare the performance of the different models under identical conditions and to examine the suitability of each architecture for practical deployment in autonomous driving systems.

The analysis utilizes statistical metrics appropriate to the nature of the outputs, including MAE, Error RMSE, R^2, and confusion matrices. The performance of each model is presented through comparative tables and graphical visualizations, as shown in Figs. 7-9, allowing clear observation of differences in prediction accuracy, stability, and error characteristics across the evaluated models.



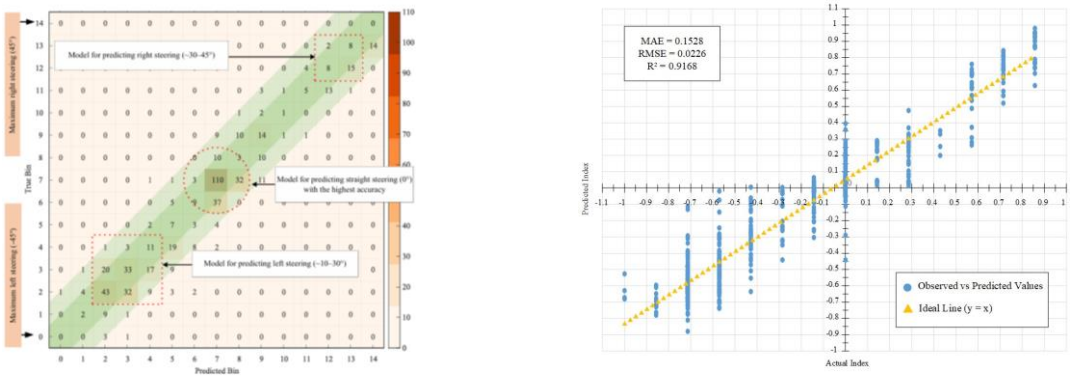
(a) Confusion matrix for linear steering (b) Linear actual vs predicted steering angle

Fig. 7 Prediction performance of the linear model for steering angle estimation



(a) Confusion matrix for categorical steering (b) Categorical actual vs predicted steering angle

Fig. 8 Prediction performance of the categorical model for steering angle estimation



(a) Confusion matrix for CNN 3D steering (b) CNN 3D actual vs predicted steering angle

Fig. 9 Prediction performance of the CNN 3D model for steering angle estimation

The performance evaluation of the three model architectures reveals distinct strengths and limitations for each. The linear model predicts steering angles at an acceptable level, with moderate error values and an R^2 score indicating a reasonable ability to explain data variance; however, it shows limitations in predicting extreme steering angles, particularly during sharp turns. The categorical model demonstrates effective classification performance for steering angles near the central range. Its accuracy decreases for classes farther from zero, resulting in relatively low overall statistical consistency due to the discrete representation of steering angles.

In comparison, the CNN 3D model achieves the most outstanding evaluation results, exhibiting the lowest prediction errors and the highest R^2 values among all models. These findings indicate that the CNN 3D architecture effectively captures both spatial and temporal features from sequential image data. This capability leads to more accurate and stable steering angle predictions, highlighting its suitability for autonomous driving systems requiring smooth and continuous directional control.

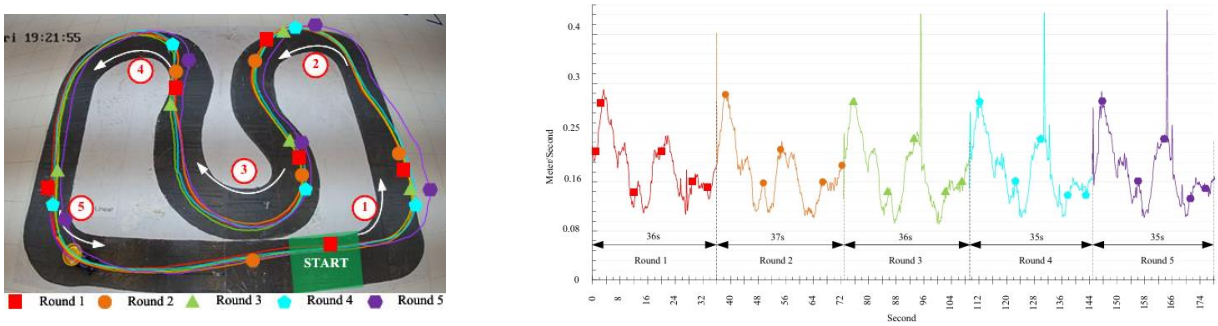
Beyond statistical performance evaluation, additional testing is conducted in a simulated environment to further assess each model's ability to control vehicle motion under conditions resembling real-world operation. This phase focuses on control behaviors such as steering adjustments, motion continuity, and driving stability when the models are deployed on the autonomous vehicle platform. The tests are carried out on a simulated track measuring 4.2 × 5.4 meters, designed with continuous curves and directional changes to challenge steering prediction performance. Each model is evaluated under identical conditions, completing five autonomous driving laps. An overhead camera is used to record vehicle motion during each trial, and the recorded videos are analyzed using Kinovea software to extract driving trajectories and vehicle speed over time. These data are subsequently used to analyze driving behavior, motion errors, and total test duration for each model.

Each model is evaluated over five autonomous driving laps under identical track and environmental conditions. This sample size is determined by the practical constraints of the overhead-camera recording and manual trajectory analysis pipeline using Kinovea software, which is time-intensive per trial. Although five laps represent a limited number of repetitions, the

evaluation conditions—fixed simulation parameters, identical starting positions, and a deterministic rendering environment—are designed to minimize inter-trial variability. The results reported in Table 2 are considered indicative, and the implications of this limitation are discussed further in Section 5.

This section presents the simulation testing results of the linear model, including visualizations of vehicle trajectories for each lap and time-series plots of speed variations. These results are used to analyze prediction accuracy, control consistency, and the practical limitations of the linear model when applied to realistic driving scenarios.

The results illustrated in Fig. 10 indicate that the linear model is capable of maintaining consistent directional control, as the vehicle trajectories across all five laps closely resemble one another. At the same time, the speed–time graph demonstrates speed adjustment behavior that aligns with the characteristics of the track, showing deceleration in curved sections and gradual acceleration along straight segments. No abrupt changes in speed occur, which reflects smooth and stable driving control achieved by the model. Notably, the trajectory overlay in Fig. 10(a) shows a high degree of visual overlap across all five laps, providing evidence of consistent directional control despite the limited number of trials. The velocity profile in Fig. 10(b) exhibits a regular deceleration–acceleration pattern that corresponds directly to the curved and straight sections of the 4.2 × 5.4 m track. This observation confirms that the model's speed regulation is aligned with track geometry rather than being artifactual.

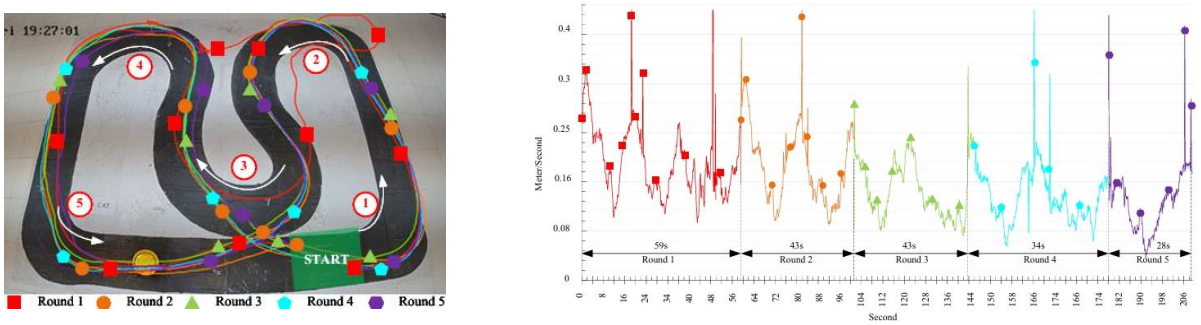


(a) Top-view trajectory of the simulated vehicle (b) Velocity variation during track navigation

Fig. 10 Performance visualization of the linear model with top-view trajectory tracking and velocity analysis

The results shown in Fig. 11 indicate that the categorical model exhibits inconsistency in directional control. The vehicle trajectories in several laps deviate from the track or follow incorrect paths, particularly during cornering sections. In addition, the speed–time graph reveals relatively high fluctuations, including periods of abnormally increased speed and intervals of significant deceleration, which correspond to control errors. These behaviors result in less smooth vehicle motion and longer overall test durations, highlighting the model's limitations in handling complex track conditions.

The trajectory divergence observed in Fig. 11(a) is most pronounced at the two sharpest cornering sections of the track, suggesting that the discrete softmax output of the categorical model introduces a steering lag that compounds in closed-loop operation. The velocity spikes visible in Fig. 11(b) coincide with these cornering events, indicating that throttle regulation is also disrupted when steering classification errors occur — a behavioral coupling that is absent in the continuous-output models.

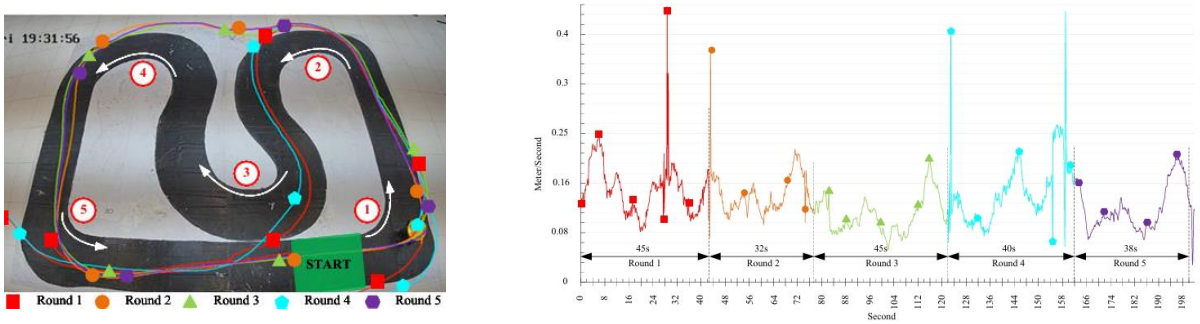


(a) Top-view trajectory of the simulated vehicle (b) Velocity variation during track navigation

Fig. 11 Performance visualization of the categorical model with top-view trajectory tracking and velocity analysis

The results presented in Fig. 12 show that the CNN 3D model exhibits trajectory deviations at several points, particularly in curved sections of the track, reflecting the challenges of making instantaneous control decisions based on sequential image data. The speed–time graph indicates that the vehicle operates mostly at relatively low speeds, with noticeable fluctuations in certain intervals. Occasional spikes in speed occur in some test runs, resulting in less smooth control when compared with the other models. Although this model demonstrates strong potential for learning temporal dependencies, further improvements in stability and responsiveness are required to enhance its suitability for real-world deployment.

The velocity fluctuations in Fig. 12(b) are attributable in part to the five-frame input buffer required by the Conv3D architecture: when the Raspberry Pi 4 CPU reaches 85–95% utilization (as shown in Fig. 13), frame buffering latency increases, causing the model to act on a stale temporal context. This latency-driven control mismatch explains why the CNN 3D model's strong offline accuracy does not translate to stable closed-loop behavior on the embedded platform.



(a) Top-view trajectory of the simulated vehicle (b) Velocity variation during track navigation

Fig. 12 Performance visualization of the CNN 3D model with top-view trajectory tracking and velocity analysis

Based on the analysis of driving behaviors across all three models using trajectory visualizations and speed profiles, the researchers conduct a quantitative comparison to more clearly distinguish test performance. The simulation test results are summarized using key metrics, including the number of driving errors on straight segments and during cornering, total test duration, and average driving speed, as reported in Table 2.

Table 2 Summary of field test results

No.	Field test metric	Model			Unit
		Linear	Categorical	CNN 3D	
1	Number of straight-line driving errors	0	0	0	occurrences
2	Number of cornering errors	0	11	8	occurrences
3	Test driving duration	1.50	4.02	3.46	minutes
4	Average driving speed	8.81	10.00	7.84	m/min

The simulation test results presented in Table 2 reveal clear differences in driving performance among the three model architectures. The linear model demonstrates the highest driving stability, with no observed errors on either straight segments or during cornering, and required the shortest test duration. In contrast, both the categorical model and the CNN 3D model exhibit a relatively high number of cornering errors, leading to longer test durations and lower average driving speeds, despite their strong learning capability during the training phase.

These findings indicate that performance in simulated field testing does not necessarily align with training outcomes, particularly for models with more complex architectures. This discrepancy highlights the gap between numerical performance metrics obtained during training and actual control behavior when the models are deployed in an autonomous driving system.

It is noted that the driving performance results reported in Table 2 are based on five laps per model and should be interpreted with this limitation in mind. Due to the restricted number of trials, the possibility that observed performance differences are partly influenced by trial-specific variability—such as minor simulation rendering jitter or CPU scheduler

fluctuations on the Raspberry Pi 4—cannot be fully excluded. Future work should conduct statistically powered evaluations with a larger number of laps and report mean and standard deviation across trials to enable more robust comparisons.

Based on the simulation results in Table 2, which show inconsistencies between driving performance and training outcomes, the researchers further analyze the computational load of the Raspberry Pi 4 board with 4 GB of memory, which served as the primary processing unit on the autonomous vehicle platform. This analysis aims to determine whether hardware resource utilization contributed to control instability. During autonomous driving tests, CPU utilization, RAM usage, and CPU temperature are recorded and compared, as illustrated in Figs. 13-15.

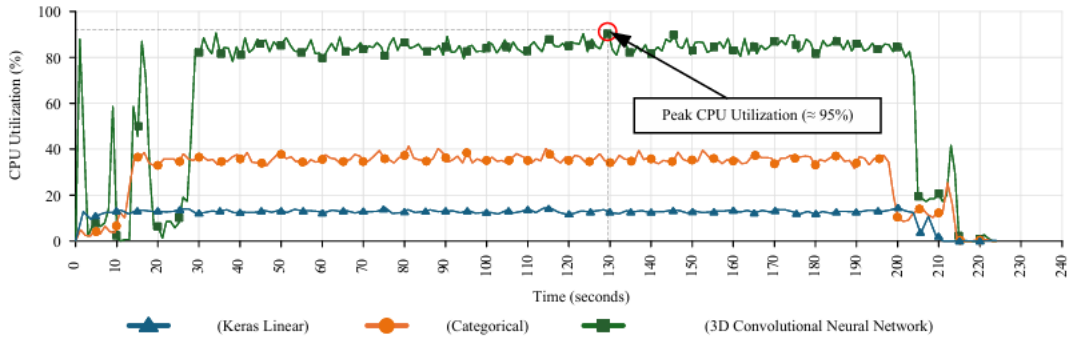


Fig. 13 CPU utilization comparison of autonomous driving models

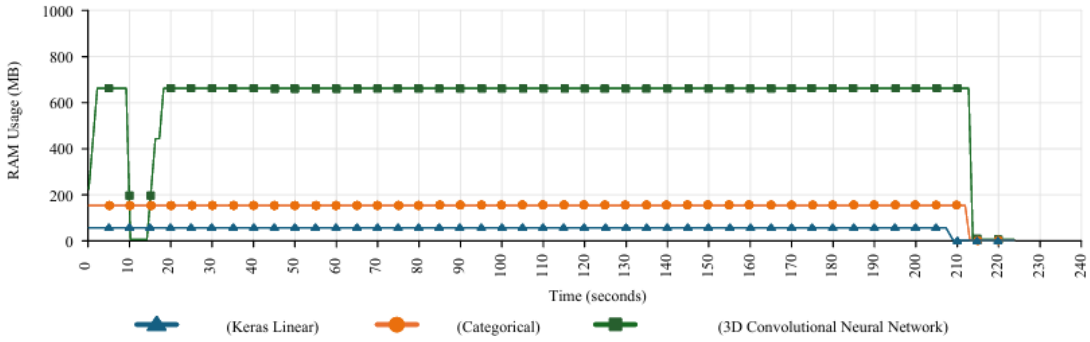


Fig. 14 RAM usage comparison of autonomous driving models

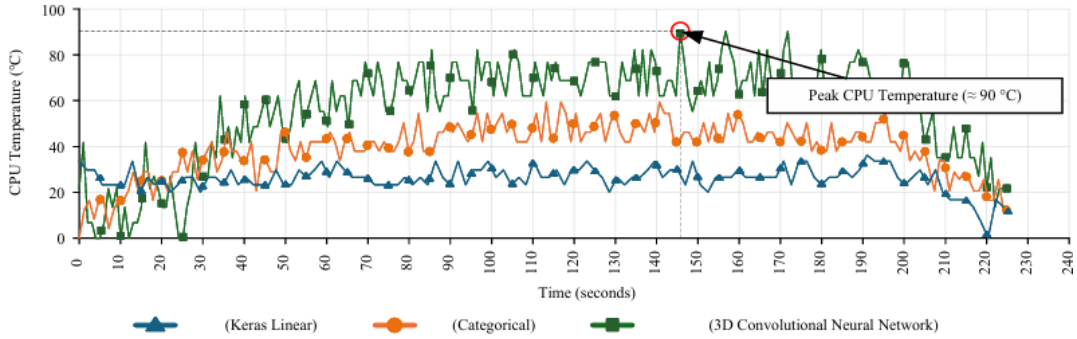


Fig. 15 Operating temperature comparison of autonomous driving models

Fig. 13 shows that the linear model exhibits the lowest CPU utilization (approximately 15%), followed by the categorical model (approximately 45–50%). In contrast, the CNN 3D model consistently consumes the highest CPU resources throughout the testing period, with utilization levels of approximately 85–95%.

Fig. 14 illustrates that the linear model exhibits the lowest RAM usage (approximately 10–50 MB), followed by the categorical model (approximately 50–150 MB). In contrast, the CNN 3D model consistently consumes the highest amount of memory, ranging from approximately 300 to 1,200 MB throughout the testing period.

Fig. 15 shows that the linear model operates at the lowest temperature (approximately 40–50 ℃), followed by the categorical model (approximately 45–60 ℃), while the CNN 3D model consistently exhibits the highest operating temperature throughout the testing period (approximately 65–85 ℃).

Based on the results presented in Figs. 13-15, substantial differences in resource utilization on the Raspberry Pi 4 board (4 GB RAM) are evident across the model types. The CNN 3D model imposes the highest computational load, consuming the most CPU and RAM resources and maintaining the highest operating temperature. In contrast, the linear model demonstrates lower resource usage and greater operational stability, whereas the categorical model exhibits moderate resource consumption. These findings highlight the resource constraints associated with deploying autonomous driving models on embedded processing platforms. They also provide important evidence for the subsequent discussion on the relationship between model complexity, control performance, and computational overhead.

Overall, the results indicate the linear model provides the most balanced solution for small-scale autonomous driving under limited computational resources by achieving stable control performance with low computational overhead. In contrast, the CNN 3D model offers superior learning capability but requires greater computational resources, making it more suitable for embedded platforms with higher processing capability or applications where prediction accuracy is prioritized over computational efficiency.

This study emphasizes that model selection for autonomous driving systems should not rely solely on training loss or numerical accuracy metrics. Instead, evaluation frameworks should incorporate closed-loop driving behavior and hardware resource limitations to ensure stable and effective deployment in real-world embedded autonomous systems.

A limitation of the present study is that the evaluation was conducted exclusively within the DonkeyCar simulation environment. Differences in visual appearance, actuator dynamics, sensor latency, and hardware characteristics may influence model performance after physical deployment. Consequently, the relative performance of the evaluated models may differ under real-world operating conditions.

5. Conclusion

This study conducted a comparative evaluation of three vision-based autonomous driving models—the linear model, the categorical model, and the CNN 3D model—under identical datasets and experimental conditions. The evaluation framework jointly considered training behavior, quantitative prediction accuracy, simulated closed-loop driving performance, and computational resource utilization on an embedded platform. The objective was to assess not only learning capability but also real-time deployability on resource-constrained embedded platforms. The main findings of this study can be summarized as follows:

(1) Training performance: during the training phase, the CNN 3D model achieved the lowest training loss, indicating a strong ability to learn spatiotemporal features from sequential image data. In contrast, the linear model required the fewest training epochs and the shortest training time, reflecting its simpler architecture and computational efficiency.

(2) Closed-Loop driving behavior: simulation results reveal that superior training performance did not consistently translate into better closed-loop driving behavior. Although the CNN 3D model exhibit favorable learning performance, both models exhibited multiple cornering errors and unstable motion during simulation. By comparison, the linear model demonstrated the most stable driving behavior, with no observed errors on straight or curved segments and the shortest overall test duration.

(3) Computational resource utilization: the analysis of resource usage on the Raspberry Pi 4 platform showed that model complexity had a significant impact on system load. The CNN 3D model imposed substantially higher CPU and memory usage and operated at higher temperatures throughout testing, while the linear model maintained lower resource consumption and greater thermal stability. These differences directly influence real-time responsiveness and system reliability, particularly during demanding maneuvers such as cornering.

Future work will focus on extending the evaluation through repeated driving trials with statistical analysis, measuring additional real-time performance metrics such as inference latency and end-to-end system delay, conducting validation on physical autonomous vehicles, and investigating model optimization techniques, including pruning and quantization, to further improve the deployment efficiency of computationally intensive models.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, et al., "Deep Reinforcement Learning for Autonomous Driving: A Survey," IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 6, pp. 3182-3200, 2021.

[2] Z. Zhu, X. and H. Zhao, "A Survey of Deep RL and IL for Autonomous Driving Policy Learning," IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 11, pp. 14043-14065, 2022.

[3] L. Le Mero, D. Yi, M. Dianati, and A. Mouzakitis, "A Survey on Imitation Learning Techniques for End-to-End Autonomous Vehicles," IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 9, pp. 14128-14147, 2022.

[4] D. Coelho and M. Oliveira, "A Review of End-to-End Autonomous Driving in Urban Environments," IEEE Access, vol. 10, pp. 75296-75311, 2022.

[5] P. S. Chib and P. Singh, "Recent Advancements in End-to-End Autonomous Driving Using Deep Learning: A Survey," IEEE Transactions on Intelligent Vehicles, vol. 9, no. 1, pp. 103-118, 2024.

[6] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li, "End-to-End Autonomous Driving: Challenges and Frontiers," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 46, no. 12, pp. 10164-10183, 2024.

[7] J. Zhao, Y. Wu, R. Deng, S. Xu, J. Gao, and A. F. Burke, "A Survey of Autonomous Driving from a Deep Learning Perspective," ACM Computing Surveys, vol. 57, no. 10, article no. 263, pp. 1-60, 2025.

[8] P. Moraes, M. Rodriguez, K. S. Kappel, H. Sodre, S. Fernandez, I. Nunes, et al., "Mini Autonomous Car Driving Based on 3D Convolutional Neural Networks," Proceedings of the IEEE Brazilian Symposium on Robotics, pp. 206-212, 2025.

[9] S. Hu, L. Chen, P. Wu, H. Li, J. Yan, D. Tao, "ST-P3: End-to-End Vision-Based Autonomous Driving via Spatial-Temporal Feature Learning," Proceedings of the European Conference on Computer Vision, pp. 533-549, 2022.

[10] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, A. Geiger, "TransFuser: Imitation with Transformer-Based Sensor Fusion for Autonomous Driving," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 45, no. 11, pp. 12878-12895, 2023.

[11] X. Jia, P. Wu, L. Chen, J. Xie, C. He, J. Yan, et al., "Think Twice Before Driving: Towards Scalable Decoders for End-to-End Autonomous Driving," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 21983-21994, 2023.

[12] A. M. Karadeniz and N. F. Koçak, "Steering Angle Prediction in Autonomous Vehicles: A Deep Learning Approach Combining VGG16 and LSTM," Journal of Computer & Electrical and Electronics Engineering Sciences, vol. 2, no. 2, pp. 46-51, 2024.

[13] B. Khawaldeh, A. M. Mora, and H. Faris, "Optimizing Steering Angle Prediction in Self-Driving Vehicles Using Evolutionary Convolutional Neural Networks," AI, vol. 5, no. 4, pp. 2147-2169, 2024.

[14] S. Paniego, E. Shinohara, and J. M. Cañas, "Autonomous Driving in Traffic with End-to-End Vision-Based Deep Learning," Neurocomputing, vol. 594, article no. 127874, 2024.

[15] H.-I. Liu, M. Galindo, H. Xie, L. K. Wong, H. H. Shuai, Y. H. Li, et al., "Lightweight Deep Learning for Resource-Constrained Environments: A Survey," ACM Computing Surveys, vol. 56, no. 10, article no. 267, pp. 1-42, 2024.

[16] G. Akkad, A. Mansour, and E. Inaty, "Embedded Deep Learning Accelerators: A Survey on Recent Advances," IEEE Transactions on Artificial Intelligence, vol. 5, no. 5, pp. 1954-1972, 2023.

[17] C. Silvano, D. Ielmini, F. Ferrandi, L. Fiorin, S. Curzel, L. Benini, et al., "A Survey on Deep Learning Hardware Accelerators for Heterogeneous HPC Platforms," ACM Computing Surveys, vol. 57, no. 11, article no. 286, 2025.

[18] X. Wang, Z. Tang, J. Guo, T. Meng, C. Wang, T. Wang et al., "Empowering Edge Intelligence: A Comprehensive Survey on On-Device AI Models," ACM Computing Surveys, vol. 57, no. 9, article no. 228, pp. 1-39, 2025.

[19] R. Cordova-Cardenas, D. Amor, and Á. Gutiérrez, "Edge AI in Practice: A Survey and Deployment Framework for Neural Networks on Embedded Systems," Electronics, vol. 14, no. 24, article no. 4877, 2025.

[20] T. Huang, H. Pan, W. Sun, and H. Gao, "Sine Resistance Network-Based Motion Planning Approach for Autonomous Electric Vehicles in Dynamic Environments," IEEE Transactions on Transportation Electrification, vol. 8, no. 2, pp. 2862-2873, 2022.



Copyright© by the authors. Licensee TAETI, Taiwan. This article is an open-access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY-NC) license (https://creativecommons.org/licenses/by-nc/4.0/).