RESEARCH ARTICLE

Optimizing CNN Architectures for Steering Angle Prediction for Self-Driving Vehicles in Unstructured Roads: A Comparative Study of Activation Functions and Model Complexity

K. C. Ranjit[1], **Bibek Shrestha**[1,*] **and Utsav Ghimire**[1,*]

[1] Computer and Electronics Department, Institute of Engineering, Tribhuvan University, Lalitpur 44700, Nepal

Abstract

This study investigates convolutional neural network (CNN) architectures for predicting steering angles in self-driving vehicles navigating unstructured roads, using road-facing image data. Two complementary experiments are conducted. First, the impact of three activation functions—Exponential Linear Unit (ELU), Rectified Linear Unit (ReLU), and Leaky ReLU—is evaluated on a baseline CNN model. Trained on 14,754 images and validated on 3,585 images, the model with ELU activation achieves the lowest validation mean squared error (MSE) compared to ReLU and Leaky ReLU, demonstrating superior convergence and generalization. Second, the effect of model complexity is examined using ELU activation across simple, moderate, and complex CNN variants. Results indicate that the moderately complex architecture yields the best performance, outperforming both simpler (underfitting) and more complex (overfitting) models in terms of validation MSE. These findings underscore the critical role of appropriate activation functions and balanced network depth in achieving robust, efficient steering prediction for autonomous driving in challenging, unstructured environments.

Keywords: convolutional neural networks, steering angles, activation functions, exponential linear units, rectified linear units, leaky relus.

1 Introduction

Self-driving vehicles represent a transformative technological frontier, combining advances in artificial intelligence, computer vision, and control systems. Steering angle prediction [1] is a critical task in autonomous navigation, requiring real-time, high-accuracy outputs to maintain vehicle stability on complex roadways [2]. Recent advancements in deep learning, particularly Convolutional Neural Networks (CNNs) [3], have shown promise in predicting steering angles from road images. CNNs, owing to their ability to learn hierarchical spatial features, are

Submitted: 17 November 2025
Revised: 17 December 2025
Accepted: 30 December 2025
Published: 10 February 2026

Vol. 2, **No.** 2, 2026.
10.62762/TMI.2025.759110

***Corresponding authors:**
✉ Bibek Shrestha
ranjitkc11111@gmail.com
✉ Utsav Ghimire
bibekshrestha881@gmail.com

Citation
Ranjit, K. C., Shrestha, B., & Ghimire, U. (2026). Optimizing CNN Architectures for Steering Angle Prediction for Self-Driving Vehicles in Unstructured Roads: A Comparative Study of Activation Functions and Model Complexity. *ICCK Transactions on Machine Intelligence*, 2(2), 88–99.

© 2026 ICCK (Institute of Central Computation and Knowledge)

well-suited for this task.

Activation functions [4, 5], which introduce non-linearity [6] into neural networks, are a vital component in CNN architectures. They play a key role in determining the model's ability to learn and generalize. ReLU, ELU, and Leaky ReLU are among the most commonly used activation functions [5] in deep learning models. While ReLU is known for its simplicity and computational efficiency, it suffers from the "dying ReLU" problem, where neurons can become inactive. ELU, on the other hand, has shown the potential to overcome this issue by allowing negative values, thus improving model convergence.

In addition to activation functions, the architecture's complexity—defined by the number of layers and parameters— plays a significant role in a model's accuracy and efficiency. As demonstrated by Bejani and Ghatee [7], deeper networks include more hyper-parameters than shallow ones, which significantly increases the probability of overfitting, while overly simple architectures may underfit and fail to capture the underlying patterns in the data. Striking the right balance is crucial, especially in real-time applications like autonomous driving [8], where computational overhead is a constraint.

In this study, we conduct two sets of experiments. First, we examine the effects of ReLU, ELU, and Leaky ReLU on a baseline CNN model. Second, we evaluate the impact of varying model complexity—using simple, moderate, and complex CNN architectures—on accuracy and generalization performance on steering angle prediction for self-driving vehicles in an unstructured road [2].

2 Literature Review

The application of convolutional neural networks (CNNs) for steering angle prediction [9, 10] in autonomous driving has received substantial attention in recent years. Early approaches largely focused on structured road environments, where lane markings and standardized road layouts simplify the task of learning steering patterns. Bhalla et al. [11] proposed a computer vision model that learns from video data. It involves image processing, image augmentation, behavioral cloning and convolutional neural network model. Ranjith Rochan et al. [12] presented a novel method of computing steering angle for driverless vehicle using computer vision based techniques of relatively lower computing cost.

Notably, NVIDIA's end-to-end learning model demonstrated that CNNs could predict steering angles [13] directly from images, effectively bypassing the need for traditional rule-based systems. Sokipriala [10] designed a model that would be able to clone a driver's behavior using transfer learning from pretrained VGG16, the results showed that the model was able to use fewer training parameters and achieved a low mean squared error (MSE) of less than 2% without overfitting to the training set, and was thus able to drive on roads it had not been trained on.

However, unstructured environments—characterized by a lack of well-defined lanes, variable road textures, and unpaved surfaces [2]—pose unique challenges that limit the efficacy of models optimized for structured settings. Adnan et al. [2] designed a deep learning-based autonomous electric vehicle for operation on unstructured road conditions.

Recent studies have explored a range of techniques, from data augmentation and specialized activation functions to lightweight architectures, aimed at improving model robustness in these unpredictable conditions. Ngoc et al. [14] systematically evaluated the effect of ELU, ReLU, and Leaky ReLU activation functions on a transfer learning-based CNN model for autonomous vehicle steering angle prediction, finding that ELU achieved superior convergence and generalization performance compared to the other activation functions. Varsheny and Singh [6] proposed a novel approach to generalize the ReLU activation function using multiple learnable slope parameters. These learnable slope parameters are optimized for every channel, which leads to the learning of a more generalized activation function (a variant of ReLU) corresponding to each channel. Saleem et al. [15] optimized the hyperparameters of a CNN for steering angle prediction in autonomous vehicles using metaheuristic algorithms, demonstrating that ELU activation combined with the Adagrad optimizer yields the best convergence among the configurations tested.

To overcome the drawbacks of ReLU, such as the dying neuron issue, Clevert et al. [16] proposed the Exponential Linear Unit (ELU), which has demonstrated superior performance in deep convolutional neural networks by improving gradient flow and reducing training time.

Another significant focus in steering angle prediction [1, 10] research has been on optimizing model complexity to balance accuracy with

computational efficiency. Gidado et al. [17] conducted a comprehensive survey on deep learning architectures for steering angle prediction in autonomous vehicles, concluding that CNN-based approaches consistently outperform other architectures and that simpler, lightweight CNN models can achieve competitive accuracy while significantly reducing computational cost. Bejani and Ghatee [7] conducted a systematic review on overfitting control in shallow and deep neural networks, demonstrating that deeper networks inherently carry more hyper-parameters and thus face a higher risk of overfitting, while the adaptation of model complexity to data complexity is a key factor in achieving robust generalization—a principle directly applicable to CNN design for steering angle prediction tasks.

3 Experiment

3.1 Dataset and Preprocessing

The dataset used in this research comprises 29700 images captured from a car operating on Indian roads, each labeled with a corresponding steering angle [10].

Data balancing was done to reduce the bias. After data balancing 11766 images were removed from the dataset. The dataset distribution is shown in Figure 1. The balanced dataset distribution is presented in Figure 2. The dataset was split into 14,754 images for training and 3,585 images for validation. The training and validation split is visualized in Figure 3. To enhance generalization, image augmentation techniques such as flipping, random brightness, and shadowing were applied, as demonstrated in Figure 4. Then the images were resized to 200x66 pixels for consistency and normalized to improve model convergence. The resizing process is illustrated in Figure 5.

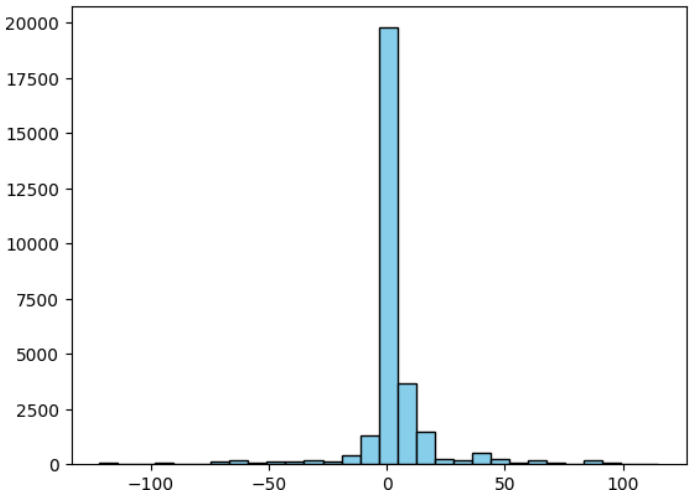

Figure 1. Dataset Distribution.

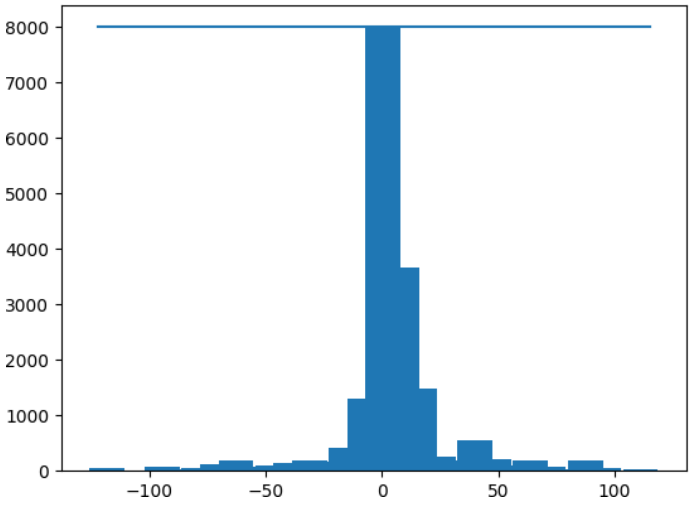

Figure 2. After Data Distribution.

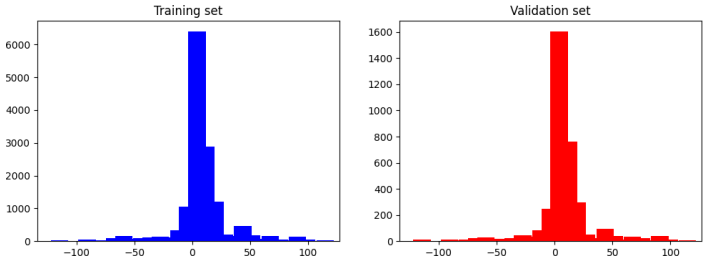

Figure 3. Training and Validation.

3.2 CNN Architecture

A Convolutional Neural Network (CNN) is a specialized type of deep learning model primarily used for processing data that has a grid-like topology, such as images [3]. The general architecture of a CNN is shown in Figure 6.

The detailed description of the CNN architecture used in this research is given below:

3.2.1 Input Layer

The input to the network consists of road images with a resolution of 200x66 pixels and 3 color channels (RGB). This input size is chosen to capture the essential features of the road while reducing computational complexity.

3.2.2 Convolutional Layer

The architecture includes four convolutional layers, progressively extracting high-level features from the input images. The details of the convolutional layers are as follows:

- **Layer 1:** A convolutional layer with 24 filters of size 5×5, applied with a stride of 2×2. The activation function is Exponential Linear Unit

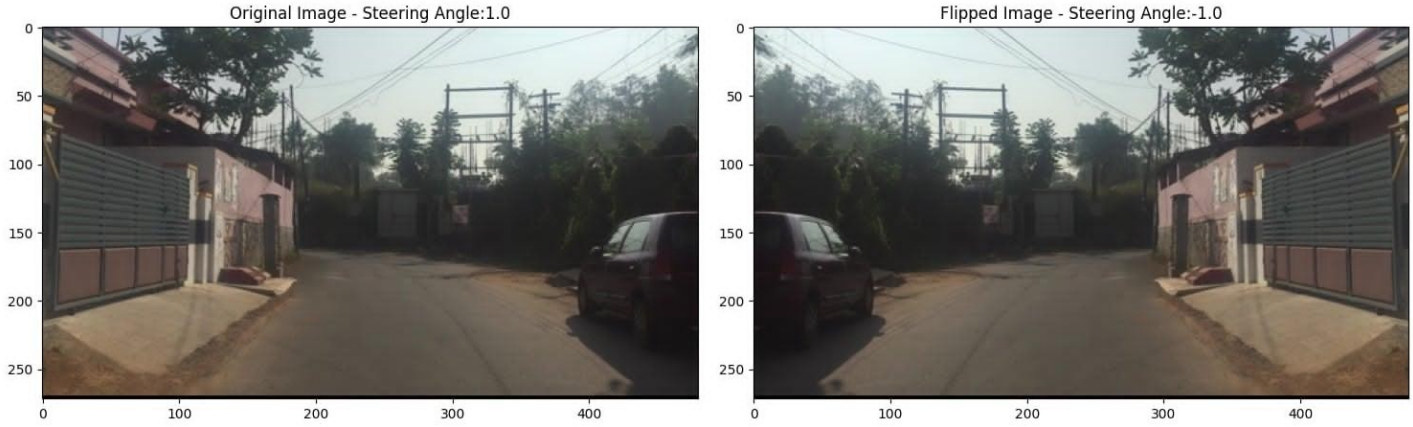

Figure 4. Original image vs augmented image.



Figure 5. Original image vs Resized image.

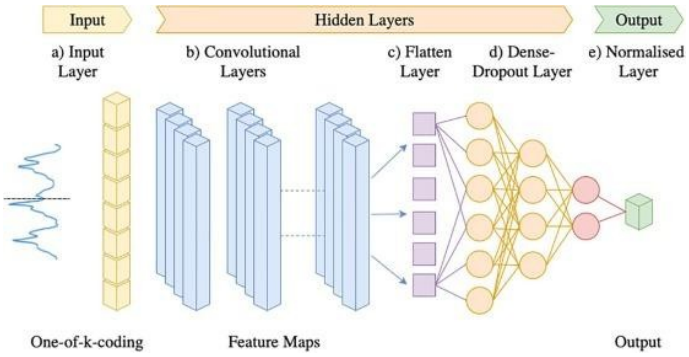

Figure 6. General architecture of a CNN.

(ELU), which is known to improve learning speed and reduce the vanishing gradient problem. L2 regularization with a factor of 0.001 is applied to prevent overfitting. Batch normalization is introduced to standardize the activations and improve convergence during training.

- **Layer 2:** A convolutional layer with 36 filters of size 5×5, with a stride of 2×2. ELU activation is applied, along with L2 regularization. Batch normalization is again employed to stabilize training.
- **Layer 3:** This layer uses 48 filters of size 5×5, with a stride of 2×2. ELU activation and L2 regularization are applied, followed by batch normalization to ensure smooth training.
- **Layer 4:** The final convolutional layer contains 64 filters of size 5×5, with ELU activation and L2 regularization. Batch normalization is applied after this layer to maintain stable activations before flattening the output.

3.2.3 Flatten Layer

After the convolutional layers, the output is flattened into a one-dimensional vector, preparing the data for the fully connected layers. This step transforms the learned features into a suitable format for further processing in the dense layers.

3.2.4 Fully Connected (Dense) Layers:

- **Layer 1:**
  Dense: 100 units with ELU activation, L2 regularization to control overfitting, and **Dropout** of 50% to further reduce overfitting. **Batch Normalization:** Added after the dense layer to

standardize outputs.

- **Layer 2:**
  Dense: 50 units with ELU activation, L2 regularization, and 50% dropout. **Batch Normalization:** Applied for normalized outputs.
- **Layer 3:**
  Dense: 10 units with ELU activation, L2 regularization, and 50% dropout. **Batch Normalization:** Normalizes outputs before the final prediction layer.

3.2.5 *Output Layer:*

Dense: A single unit with no activation, which predicts the steering angle as a continuous value.

3.3 Model Compilation:

The model is compiled using the **Adam optimizer** with an initial learning rate of 1×10^{-3}, and the loss function is a **mean squared error (MSE)**, as it's appropriate for regression tasks like steering angle prediction. The accuracy metric is also tracked during training.

Mean squared error:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

where N denotes the total number of samples, y_i represents the true steering angle for the i-th sample, and $\hat{y}_i$ corresponds to the predicted steering angle for the i-th sample.

3.3.1 *Experiment 1: Comparison of Activation Function Performance*

Activation functions [4] play a crucial role in neural networks by introducing non-linearity, which allows the model to learn complex patterns in the data. The choice of activation function can significantly influence the model's performance, convergence speed, and ability to generalize to new data. This experiment aims to compare various activation functions in a Convolutional Neural Network (CNN) architecture for the task of steering angle prediction, to determine which activation function yields the best convergence and generalization performance.

In this experiment, we evaluated three activation functions, training them over multiple epochs until their performance reached a plateau [15].

i) ELU:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{otherwise} \end{cases}$$

where $\alpha = 1$ by default. The ELU activation function is visualized in Figure 7.

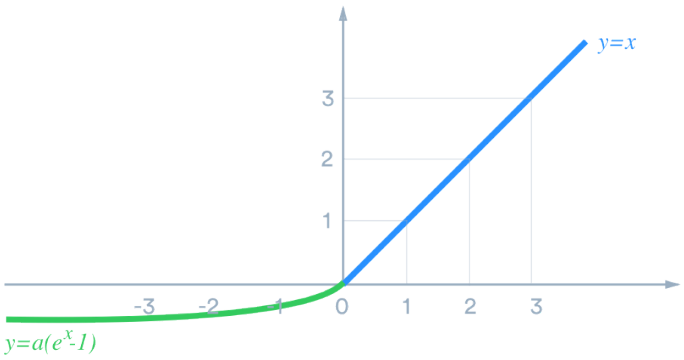

Figure 7. ELU activation function.

ii) ReLU:

$$f(x) = \max(0,\, x)$$

The ReLU activation function is shown in Figure 8.

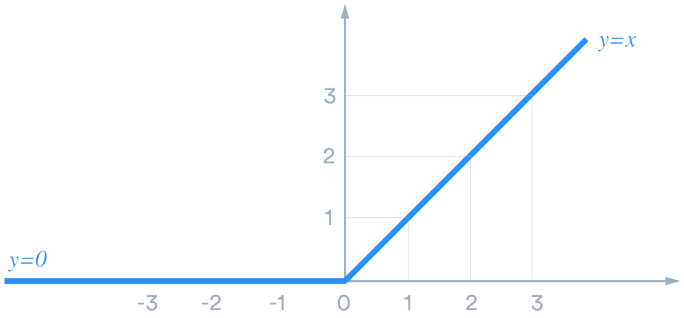

Figure 8. ReLU activation function.

iii) Leaky ReLU:

$$f(x) = \max(0.01x,\, x)$$

The Leaky ReLU activation function is illustrated in Figure 9.

3.3.2 *Experiment 2: Model Complexity Comparison*

Model complexity is a critical factor that influences a neural network's generalization performance, as deeper architectures with more hyper-parameters are inherently more prone to overfitting while shallower ones risk underfitting [7]. A model that is too simple may underfit the data, failing to capture the underlying patterns [17], while an overly complex model may overfit, learning noise instead of meaningful features. This experiment aims to compare different CNN architectures with varying levels of complexity to

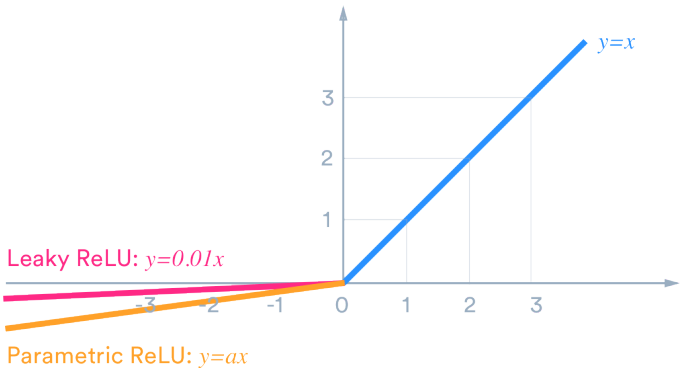

Figure 9. Leaky ReLU.

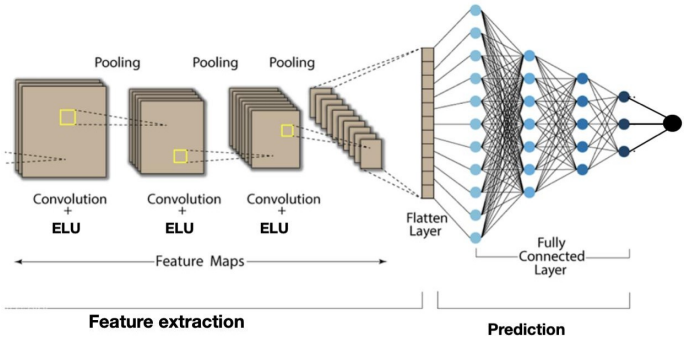

Figure 11. A complex CNN architecture.

assess their impact on steering angle prediction performance, with a focus on generalization ability and training efficiency. In the second experiment, we evaluated the impact of model complexity by testing three variations of the CNN architecture:

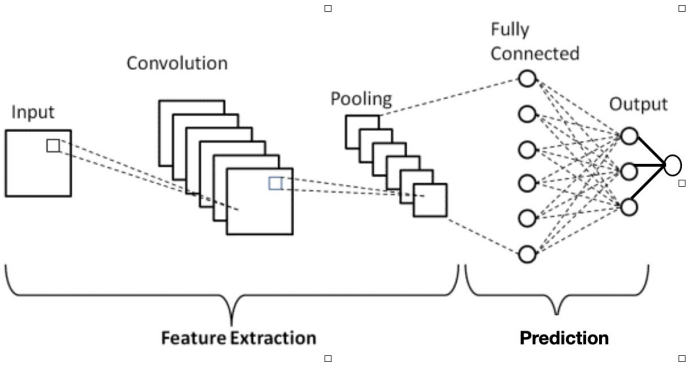

Figure 10. A simple CNN architecture.

- **Simple Model:** 2 convolutional layers, 1 flattened layer, 2 dense layers with 50 and 100 units, and 1 output layer.
- **Moderate Model:** The base model used for experiment 1 contained 4 convolutional layers, 1 flattened layer, 3 dense layers with 100, 50, and 10 units, and 1 output layer.
- **Complex Model:** 5 convolutional layers, 1 flattened layer, 3 dense layers with 200,100 and 50 units, and 1 output layer.

The architectures of the simple and complex models are visualized in Figures 10 and 11, respectively.

4 Result and Discussion

The experimental results were analyzed in two stages to assess the impact of different activation functions [4] and model complexities [7] on predicting steering angles for a self-driving vehicle [1]. These stages focused on key aspects that influence the performance of Convolutional Neural Networks (CNNs), which are crucial for making accurate predictions in real-time [8].

4.1 Activation Function Comparison

The first stage of the experimentation aimed to determine the optimal activation function for predicting steering angles by testing three widely-used activation functions: ELU (Exponential Linear Unit), ReLU (Rectified Linear Unit), and Leaky ReLU [4].

Experimenting with ELU yielded the following results
As shown in Figures 12 and 13, ELU converged after 38 epochs giving the validation loss of 22.2839 at its lowest. The weights were restored, hence the model trained using ELU had a validation loss of 22.2839. The total training time taken for ELU to reach the plateau was 61 minutes and 38 seconds. The average time taken per epoch was 97.316 seconds.

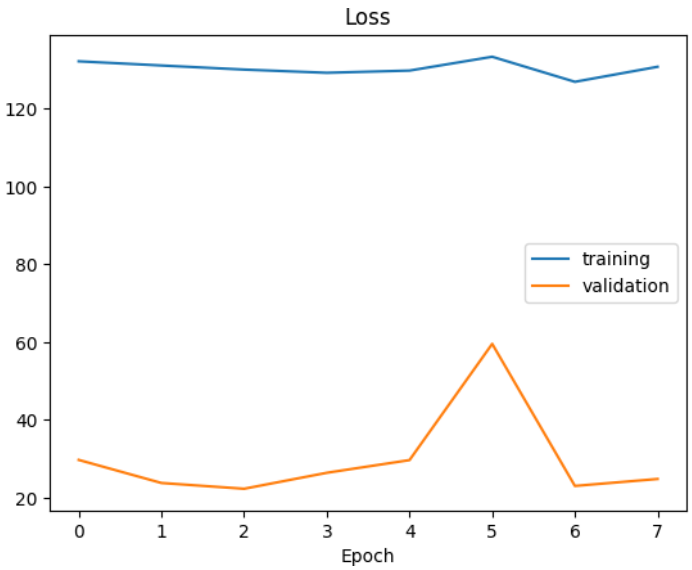

Figure 12. Validation and training loss of ELU from 1 to 30 epochs.

Experimenting with ReLU yielded the following results
As shown in Figures 14 and 15, ReLU converged after 40 epochs giving the validation loss of 57.2455 at its

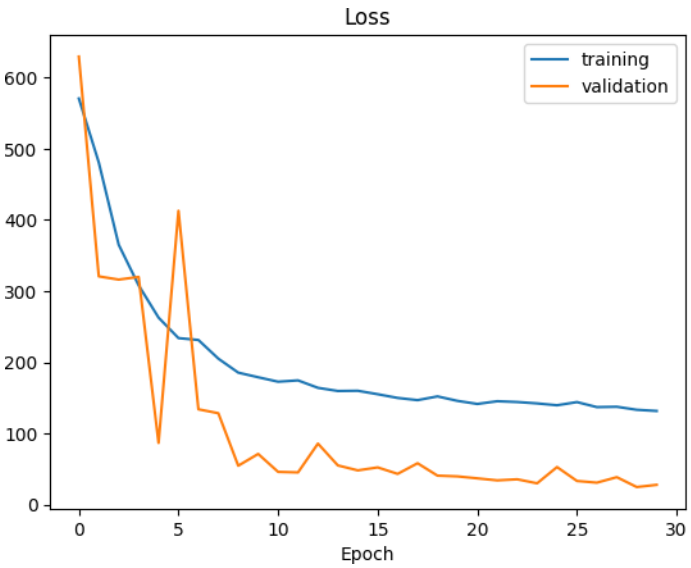

Figure 13. Validation and training loss of ELU from 31 to 38 epochs.

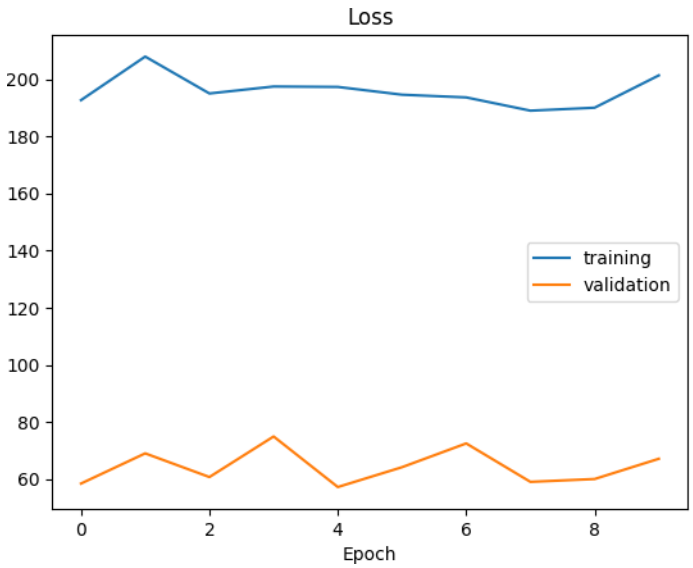

Figure 15. Validation and training loss of ReLU from 31 to 40 epoch.

lowest. The weights were restored, hence the model trained using ReLU had a validation loss of 57.2455. The total training time taken for ReLU to reach the plateau was 85 minutes and 1 second. The average time taken per epoch was 127.525 seconds.

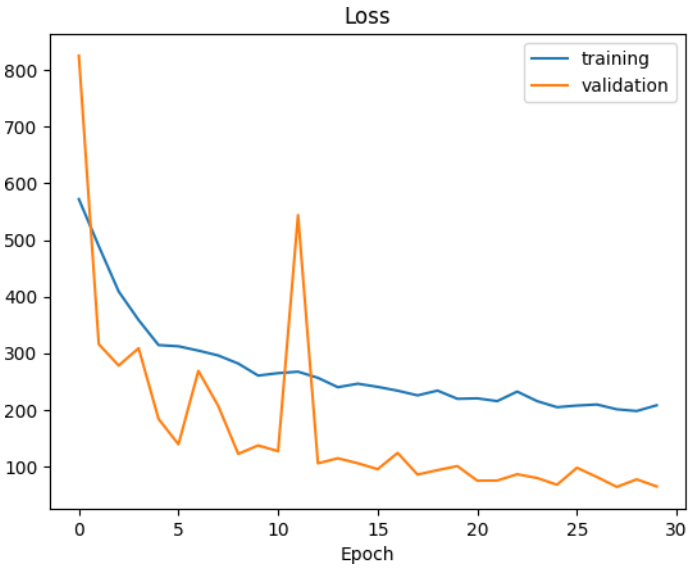

Figure 14. Validation and training loss of ReLU from 1 to 30 epochs.

Experimenting with Leaky ReLU yielded the following results

As shown in Figures 16 and 17, Leaky ReLU converged after 43 epochs giving the validation loss of 46.6719 at its lowest. The weights were restored, hence the model trained using Leaky ReLU had a validation loss of 46.6719. The total training time taken for Leaky ReLU to reach the plateau was 87 minutes and 23 seconds. The average time taken per epoch was 121.93 seconds, and Table 1 contains the summarized result of experiment 1.

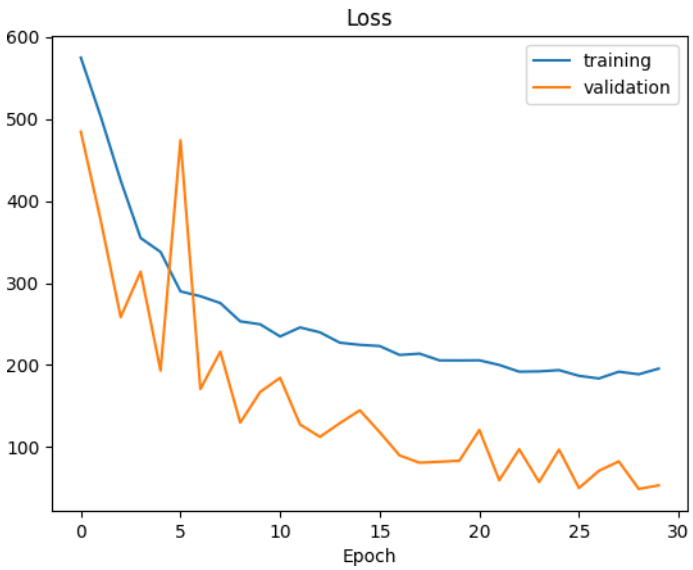

Figure 16. Validation and training loss of Leaky ReLU from 1 to 30 epochs.

4.2 Model Complexity Comparison

While the choice of activation function plays a critical role in determining a CNN's performance, the complexity of the model [7]—the number of layers and the number of parameters—also significantly influences how well the network performs.

In the second experiment, three models were designed with varying levels of complexity to understand the trade-off between performance and computational cost. Each model used the best-performing ELU activation function from the first experiment, allowing for a

Table 1. Results of experiment 1.

Activation function	Lowest Validation loss obtained	Number of epochs take to reach plateau	Average time taken per epoch in seconds
ELU	22.2839	38	97.316
ReLU	57.2455	40	127.525
Leaky ReLU	46.6719	43	121.93

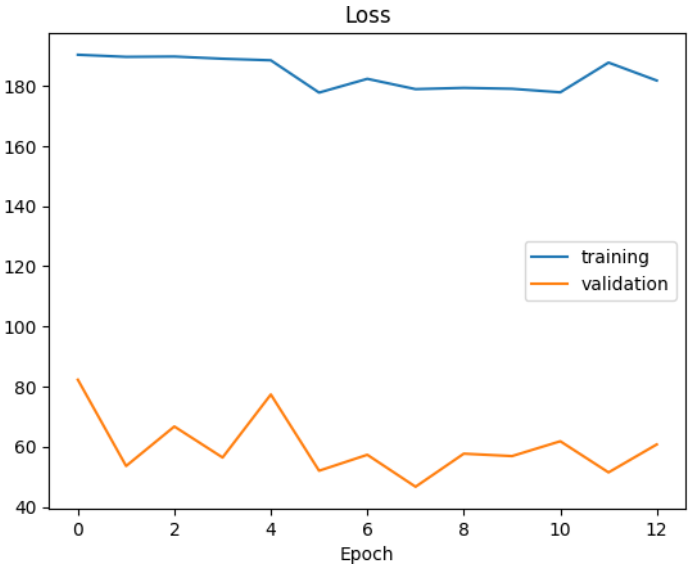

Figure 17. Validation and training loss of Leaky ReLU from 31 to 40 epoch.

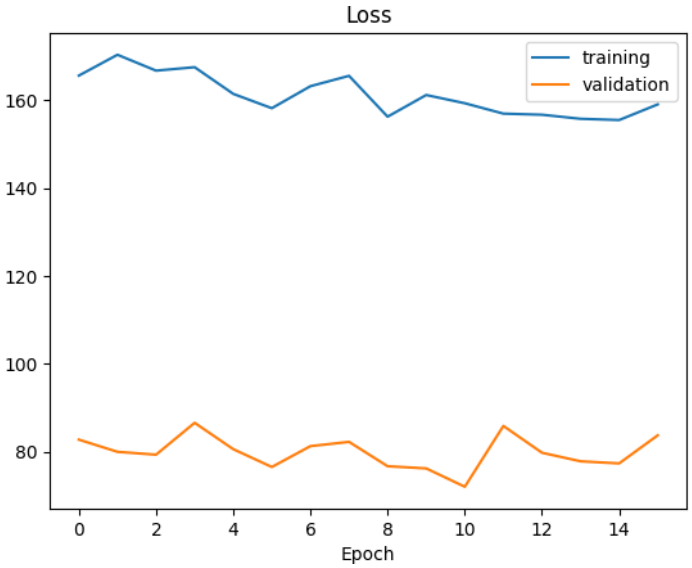

Figure 19. Validation and training loss of simple model from 28 to 43 epochs.

fair comparison. The simple model had fewer layers and was designed to be lightweight, requiring less computational power, making it suitable for real-time deployment on devices with limited resources, such as edge devices in self-driving cars.

Experimenting with the simple model yielded the following results.

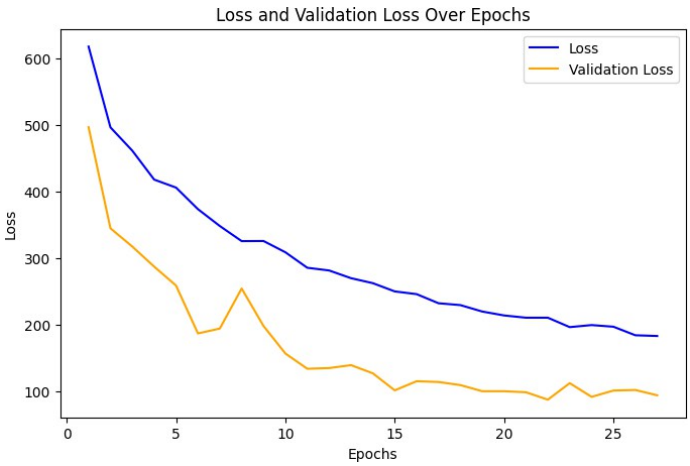

Figure 18. Validation and training loss of simple model from 1 to 27 epochs.

As shown in Figures 18 and 19, the simple model with ELU activation function converged after 43 epochs giving the validation loss of 71.9829 at its lowest. The best weights were restored, hence this model had a validation loss of 71.9829. The total training time taken for the simple model to reach the plateau was 68 minutes and 39 seconds. The average time taken per epoch was 95.8 seconds.

This shows that the model with fewer layers was not able to capture the complexities of the dataset effectively, resulting in higher validation loss compared to more complex models. The choice of the ELU activation function helped with faster convergence, but the overall architecture still limited the model's ability to learn intricate patterns. This indicates that increasing the depth or adjusting other hyper-parameters might be necessary for better performance. Further experiments with different architectures could provide insights into improving model accuracy and reducing validation loss.

The moderately complex model struck a balance between simplicity and depth. It included more layers than the simple model, allowing it to capture more complex patterns in the data, without being overly cumbersome in terms of computation.

Experimenting with the moderate model yielded the following results.

As shown in Figures 20 and 21, the moderate complexity model converged after 38 epochs giving the validation loss of 22.2839 at its lowest. The weights were restored, hence this model trained had a validation loss of 22.2839. The total training time taken for this model to reach the plateau was 61 minutes and 38 seconds. The average time taken per epoch was 97.316 seconds.

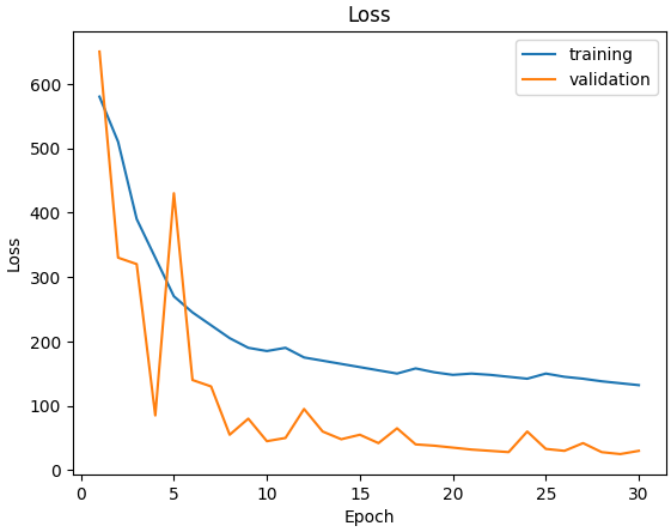

Figure 20. Validation and training loss of moderate model from 1 to 30 epochs.

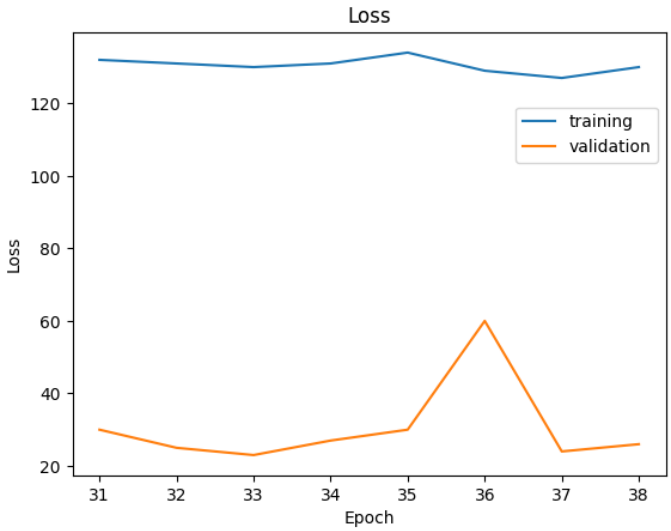

Figure 21. Validation and training loss of moderate model from 31 to 38 epochs.

This shows that the moderate complexity model was able to effectively capture the complexities of the dataset, achieving a lower validation loss compared to simpler models. Its architecture likely allowed it to learn the underlying patterns and relationships in the data more efficiently. This balance between model complexity and performance suggests that fine-tuning hyper-parameters and layer configurations can significantly impact model accuracy and convergence speed. The complex model featured numerous layers and a large number of parameters. This model was designed to capture even the most intricate patterns in the road images, but its computational demands were significantly higher.

Experimenting with the complex model yielded the following results.

As shown in Figures 22 and 23, the complex model converged after 34 epochs giving the validation loss of 50.7887 at its lowest. The weights were restored, hence this model trained had a validation loss of 50.7887. The total training time taken for this model to reach the plateau was 202 minutes and 2 seconds. The average time taken per epoch was 356.6 seconds, which is the highest time taken per epoch among all the experiments.

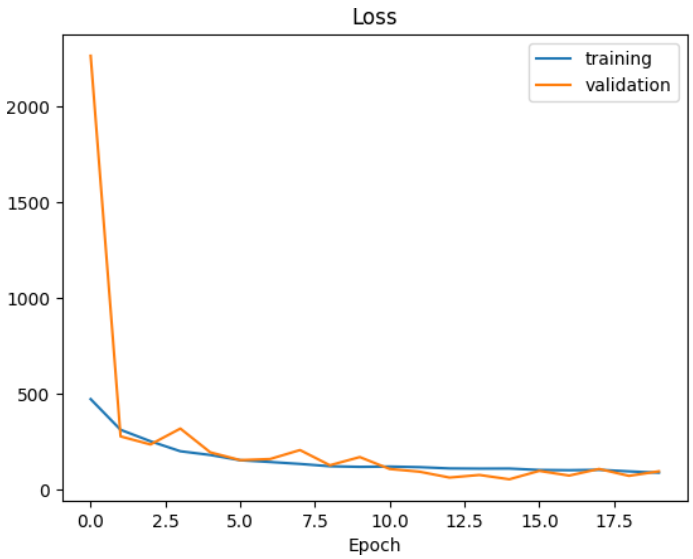

Figure 22. Validation and training loss of complex model from 1 to 20 epochs.

In retrospect, the complex model gave the lowest training loss of 73.7307 among all the experiments that were conducted. However, its validation loss was worse than that of the moderate complexity model. This shows that the complex model had overfitted the training dataset, hence it was unable to perform well on the validation dataset, despite the usage of L2 regularization. This observation highlights that regularization alone does not guarantee improved generalization, especially in models with excessive complexity. Table 2 contains the summarized result of

Table 2. Results of experiment 2.

CNN models	Lowest Validation loss obtained	Number of epochs take to reach plateau	Average time taken per epoch in seconds
Simple	71.9829	43	95.8
Moderate	22.2839	38	97.316
Complex	50.7887	34	356.6

Table 3. Results Produced by moderate complexity model.

Images	Predicted angle	True angle	Error	Time
	-33 degrees	-38 degrees	5 degrees	15 ms
	35 degrees	40 degrees	5 degrees	14 ms
	2 degrees	4 degrees	2 degrees	12 ms
	-9 degrees	-14 degrees	5 degrees	15 ms
	-124 degrees	-122 degrees	2 degrees	14 ms
	124 degrees	115 degrees	9 degrees	14 ms

experiment 2.

Analyzing the above results shows that the higher the model complexity, the larger the training time gets. However, increasing the model complexity does not guarantee lower loss on the validation dataset. Also, as the complexity increases, the plateau is reached faster.

To further demonstrate the performance of the moderate complexity model, we tested it on sample images from unstructured roads. The prediction results are summarized in Table 3.

The moderate complexity model delivered outstanding results, even when tested on unstructured road images. Despite the absence of normalization, the steering angle predictions maintained exceptional accuracy, with errors of just 5°, 5°, 2°, 5°, 2°, and 9°. This showcases the model's robustness, handling complex real-world conditions with ease.

What's more remarkable is the speed. The prediction times ranged between 12 and 15 milliseconds per image, running on the M1 chip, which is not classified as extravagant hardware. This translates to an impressive 66 frames per second (fps), demonstrating the model's real-time capabilities, even though the

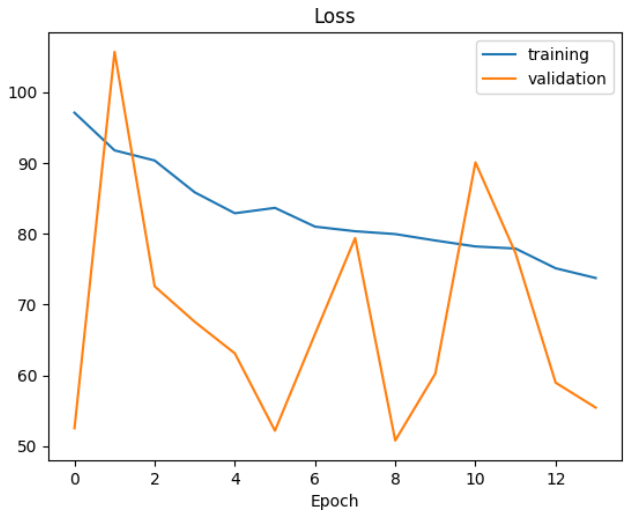

Figure 23. Validation and training loss of complex model from 21 to 34 epochs.

M1 chip is not specialized for self-driving vehicles unlike dedicated autonomous driving hardware platforms [8].

This combination of low error rates and fast prediction times makes this model both practical and highly efficient for autonomous driving, ensuring accurate and timely steering responses without requiring top-tier hardware.

5 Conclusion

In conclusion, this research has illuminated critical insights into the dynamics of activation functions and model complexity in deep learning, with significant implications for applications like self-driving vehicles on unstructured roads. The first experiment highlighted the effectiveness of the Exponential Linear Unit (ELU) activation function, which facilitated convergence within 38 epochs and resulted in a validation loss of 22.2839. This finding underscores the importance of selecting appropriate activation functions to enhance model performance and training efficiency.

The second experiment revealed a striking contrast in model complexity: the moderate complexity model achieved a superior validation loss of 22.2839 after 38 epochs, despite having fewer layers. This outcome emphasizes that while deeper architectures may intuitively seem advantageous, they can lead to overfitting, as demonstrated by the more complex model's inability to generalize effectively, even with L2 regularization in place. The training time of 97.316 seconds per epoch for the moderate complexity model in comparison to the 356.6 seconds training time per epoch for the more complex model further reinforces the need for a strategic model design that balances complexity with performance. Notably, the moderate complexity model demonstrated low steering angle prediction errors of just 5°, 5°, 2°, 5°, 2°, and 9° as shown in Table 3, highlighting its robustness. Furthermore, with prediction times averaging between 12 and 15 milliseconds, the model is expected to reach an impressive frame rate of over 66 frames per second (fps), ensuring timely decision-making crucial for safe autonomous navigation.

These insights are particularly relevant to the development of self-driving vehicles navigating unstructured environments, where model robustness and generalization are crucial for safe and effective operation. By carefully selecting activation functions and optimizing model complexity, future work can enhance the reliability of predictive models in real-world driving scenarios.

Moving forward, this research sets the stage for inquiries into more nuanced regularization techniques and hyper-parameter optimization, particularly as they relate to activation functions and architectural depth. By continuing to explore these avenues, we can refine our approach to model design, ultimately leading to more robust and effective solutions for complex applications like autonomous driving for unstructured roads.

Data Availability Statement

Data will be made available on request.

Funding

This work was supported without any funding.

Conflicts of Interest

The authors declare no conflicts of interest.

AI Use Statement

The authors declare that no generative AI was used in the preparation of this manuscript.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Singhal, V., Gugale, S., Agarwal, R., Dhake, P., & Kalshetti, U. (2019, September). Steering angle

prediction in autonomous vehicles using deep learning. In *2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA)* (pp. 1-6). IEEE. [CrossRef]

[2] Adnan, A., Rahman, G. M., Hossain, M. M., Mim, M. S., & Rahman, M. K. (2022, May). A deep learning based autonomous electric vehicle on unstructured road conditions. In *2022 IEEE 12th Symposium on Computer Applications & Industrial Electronics (ISCAIE)* (pp. 105-110). IEEE. [CrossRef]

[3] Chaudhari, R., Dubey, S., Kathale, J., & Rao, R. (2018, April). Autonomous driving car using convolutional neural networks. In *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)* (pp. 936-940). IEEE. [CrossRef]

[4] Hazra, R., Kumar, A., & Baranidharan, B. (2019). Effect of various activation function on steering angle prediction in CNN based autonomous vehicle system. *International Journal of Engineering and Advanced Technology (IJEAT), 9*(2), 3806–3811. [CrossRef]

[5] Dubey, S. R., Singh, S. K., & Chaudhuri, B. B. (2022). Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing, 503*, 92-108. [CrossRef]

[6] Varsheny, M., & Singh, P. (2021). Optimizing nonlinear activation function for convolutional neural networks. *Signal, Image and Video Processing, 15*(6), 1323–1330. [CrossRef]

[7] Bejani, M. M., & Ghatee, M. (2021). A systematic review on overfitting control in shallow and deep neural networks. *Artificial Intelligence Review, 54*(8), 6391-6438. [CrossRef]

[8] Hassan, I. U., Zia, H., Fatima, H. S., Yusuf, S. A., & Khurram, M. (2022). A lightweight convolutional neural network to predict steering angle for autonomous driving using CARLA simulator. *Modelling and Simulation in Engineering, 2022*(1), 5716820. [CrossRef]

[9] George, K. S., Abhiram, A., Jose, A., Madhav, A. S., Najiya, N., & Aswin, S. (2022, March). Steering Angle Estimation for an Autonomous Car. In *2022 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)* (Vol. 1, pp. 168-173). IEEE. [CrossRef]

[10] Sokipriala, J. (2021). Prediction of steering angle for autonomous vehicles using pre-trained neural network. *European Journal of Engineering and Technology Research, 6*(5), 171–176. [CrossRef]

[11] Bhalla, A., Nikhila, M. S., & Singh, P. (2020, December). Simulation of self-driving car using deep learning. In *2020 3rd international conference on intelligent sustainable systems (ICISS)* (pp. 519-525). IEEE. [CrossRef]

[12] Ranjith Rochan, M., Aarthi Alagammai, K., & Sujatha, J. (2018). Computer vision based novel steering angle calculation for autonomous vehicles. In *Proceedings of the 2018 Second IEEE International Conference on Robotic Computing (IRC)*, Laguna Hills, CA, USA (Vol. 31). [CrossRef]

[13] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., ... & Zieba, K. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*. [CrossRef]

[14] Ngoc, H. T., Hong, P. P., Quoc, A. N., & Quach, L. D. (2024). Steering angle prediction for autonomous vehicles using deep transfer learning. *Journal of Advances in Information Technology, 15*(1), 138–146. [CrossRef]

[15] Saleem, H., Riaz, F., Shaikh, A., Rajab, K., Rajab, A., Akram, M., & Al Reshan, M. S. (2021). Optimizing steering angle predictive convolutional neural network for autonomous car. *Computers, Materials and Continua, 71*(2), 2285-2302. [CrossRef]

[16] Clevert, D. A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*. [CrossRef]

[17] Gidado, U. M., Chiroma, H., Aljojo, N., Abubakar, S., Popoola, S. I., & Al-Garadi, M. A. (2020). A survey on deep learning for steering angle prediction in autonomous vehicles. *IEEE Access, 8*, 163797-163817. [CrossRef]



K. C. Ranjit received his B.E. degree in Electronics, Communication, and Information Engineering from Tribhuvan University in 2024. He is currently serving as an Assistant Lecturer at the National College of Engineering, Tribhuvan University. His research interests include machine learning, computer vision, Internet of Things (IoT), autonomous vehicles, and multi-agent learning. (Email: ranjitkc11111@gmail.com)



Bibek Shrestha received his B.E. degree in Electronics, Communication, and Information Engineering from Tribhuvan University in 2024. He is currently an Assistant Lecturer at Cosmos College of Technology and Management, Pokhara University. His research interests include machine learning, computer vision, multi-agent learning, AI agents, Internet of Things (IoT), autonomous vehicles, artificial intelligence, wireless communication, and modern physics. (Email: bibekshrestha881@gmail.com)



Utsav Ghimire received his B.E. degree in Electronics, Communication, and Information Engineering from Tribhuvan University in 2024. He is currently pursuing a Master's degree at the University of the West of Scotland. His research interests include machine learning, computer vision, robotics, Internet of Things (IoT), and autonomous vehicles. (Email: utsavghimire51@gmail.com)