Steering Angle Prediction of Autonomous Vehicle Using Machine Learning

Muhammad Shahid Khan[1], Abdul Rehman Omer[2] and Abdullah Yousafzai[3]

Abstract— **Intelligent Transportation Systems (ITS) encompass a variety of subsystems and technologies, such as basic car navigation management systems, traffic signal control systems, road actor classifications, lidar systems, and autonomous control systems. One crucial aspect of autonomous control systems is steering angle prediction, which is essential for keeping the vehicle within road lines a challenging task. Steering angle prediction has garnered significant attention from researchers, manufacturers, and insurance companies due to its importance in controlling autonomous vehicles (AVs). Various Deep Learning (DL) architectures have been employed to predict AV steering angles in different contexts. However, these methods often involve a large number of parameters and complex algorithms, requiring substantial computational resources and incurring communication overhead and privacy concerns due to the sharing of raw training data. This makes them unsuitable for widespread deployment within ITS. This paper proposes a federated learning-based approach for training a convolutional neural network (CNN) model with significantly fewer parameters. Experimental results demonstrate that our approach outperforms state-of-the-art methods, achieving an average training mean square error of 0.0274 using Sully Chen's dataset.**

I. INTRODUCTION

Automation enables humans to reduce their workload by delegating tasks to autonomous systems powered by various algorithms[1]. This automation is achieved through a range of Artificial Intelligence (AI) techniques, enhancing the reliability of autonomous systems[2]. Among the most popular autonomous systems today are self-driving cars. These vehicles are considered autonomous when they can drive themselves and perform effectively despite unforeseen changes in their environment[3]. Moreover, autonomous vehicles can operate without issues even if a system fails. Autonomous vehicles, also known as driverless cars, self-driving cars, or robotic cars, can travel without human input. They are expected to sense their surroundings, monitor critical systems, and manage control functions, including navigation. Perception involves abstracting the vehicle and its surroundings by receiving and interpreting visual and auditory inputs from both inside and outside the vehicle. The control system then determines the vehicle's movements, considering the path, road conditions, traffic signals, and obstacles. Autonomous vehicles could impact various sectors, including the job market, traffic management, insurance, urban planning, health, welfare, and the automotive industry. Proper regulation is necessary for their deployment. With rapid technological advancements, self-driving vehicles and other automation tools are quickly becoming integral to many aspects of human life, transforming autonomous vehicles into a new piece of infrastructure.

Over the past decade, autonomous vehicles (AVs) have garnered substantial research interest from both industry and academia, aiming to enhance user experiences within intelligent transportation systems (ITS). Machine learning (ML) technologies are leveraged to execute self-driving functions such as steering angle prediction, road actor classification, and path planning. Steering angle prediction refers to determining the angle of the wheels, typically the front wheels, which turn to enable the vehicle to navigate a turn. This angle is generally measured between the vehicle's centerline and the centerline of the turned wheels. The precision in predicting steering angles based on real-time road conditions is crucial for the safety and performance of self-driving cars [4]. Autonomous vehicles must adapt to diverse driving scenarios, including highway driving, city streets, and parking lots. The steering angle is pivotal in adjusting the vehicle's behavior to meet the specific demands of each situation [5]. It directly influences the vehicle's direction and is essential for trajectory and path planning. Correct steering angles are vital for the safe, efficient, and reliable operation of autonomous vehicles. They are critical for path following, obstacle avoidance, adaptability to dynamic environments, and overall vehicle safety and performance. Thus, developing robust models for steering angle prediction remains a significant challenge for autonomous driving functions.

Autonomous vehicles and driver assistance systems rely on accurate steering angle prediction for safe navigation and control. Existing methods for predicting steering angle often involve many parameters and complex algorithms that require significant computational resources. This makes it unsuitable for deployment on resource-constrained platforms such as embedded systems or low-power devices.[6]. To solve this problem, we use a lightweight convolutional neural network (CNN) to predict the steering angle of an autonomous vehicle. In this paper, we aim to develop a lightweight steering angle prediction algorithm with fewer parameters and fewer computational requirements while maintaining acceptable performance. Current approaches focus on lightweight models to address model size and speed issues, which are particularly useful for applications with

*This work was not supported by any organization

[1]Shahid Khan is a gradute student at University of Central Punjab, Pakistan. He is the corresponding author: mshahidkhan430@gmail.com

[2]Abdul Rehman is also a graduate student at University of Central Punjab, Pakistan arocofcr84@gmail.com

[3]Research Fellow at Artificial Intelligence and Cyber Future Institute, Charles Sturt University, Australia. ayousafzai@csu.edu.au

limited computational resources, such as mobile or edge devices. A variety of lightweight CNN architectures can be tailored to the limited onboard computing capabilities of autonomous vehicles. The choice of architecture depends on the specific requirements of the problem, available computing resources, and the desired tradeoff between model size and accuracy. It is often necessary to experiment with and customize multiple architectures based on application requirements. In the traditional machine learning model, a large amount of data storage in one place, which may lead to privacy breaches. It requires large amounts of time, resources, and speed while computing a large dataset. The federated average algorithm (FedAvg), introduced by [7], has demonstrated robustness in CNN training. We trained CNN models using the FedAvg algorithm, a promising method for jointly training machine learning models across multiple edge devices in a privacy-preserving manner. The effectiveness of federated learning (FL) depends on the performance of the participating models and their ability to handle the inherent challenges of distributed learning. Models developed for client devices in FL must balance model size, computational efficiency, and adaptability to the variety of observed non-IID data distributions. This is because client devices typically have limited computational resources and connection bandwidth.

II. Related Work

Deep neural networks have been effectively employed to accurately predict steering angles in autonomous vehicles. These models utilize lane line coordinates and a path equation to determine the appropriate steering angle for each video frame. Performance evaluations using the Comma.ai and Udacity datasets have demonstrated mean square errors of 0.87 and 2.33, respectively[8].This study developed a model for self-driving cars to accurately predict steering angles by utilizing transfer learning from a pre-trained VGG16. The model achieved a low mean squared error (MSE) and demonstrated the ability to drive on new roads without overfitting[9]. The Udacity dataset was used to evaluate its performance.A CNN model was proposed to predict the steering angle for self-driving cars, achieving a validation loss of 11.89% and a training loss of 10.74% [10]. The model successfully operated the car on both tracks, showcasing its accuracy and efficiency. Falling between Level 1 and Level 2 automation, the use of CNN enhances its performance. However, the challenge lies in training a deep neural network for this task due to the large volume of training data required.The model, inspired by NVIDIA's architecture, employs VGG16 with 15,129,149 parameters, which significantly increases training time and computational costs. In contrast, NVIDIA's model consists of 1,229,193 parameters[11].The paper introduces a ResNet-152 Neural Network, Navida, and DenseNet-210 model designed for autonomous self-driving cars. These models showcase experimental accuracy and effectively preserve lane integrity.The architecture of these models comprises 1164, 100, 50, 10, and 1 parameter. Despite achieving mean square errors of 0.512, 0.791, 0.294, and 0.278513, respectively, these models have a significant number of parameters that necessitate time-consuming feature extraction.The CNN contains 204,661 parameters, the LSTM contains 48,658,241 parameters, and the combined CNN+LSTM model has 1,125,153 parameters. These models yield mean squared errors (MSE) of 2.1666, 0.0866, and 0.7422, respectively[12]. The ResNet model utilizes 20.6 million parameters, while the Inception Net model employs 21.7 million parameters[13].

III. Material

A. Dataset

The dataset utilized in this paper was obtained from Sully Chen's GitHub repository [14]. Sully Chen, a researcher at The National Institutes of Health, embarked on a college project named Tensor Flow Auto Pilot, driven by his enthusiasm for machine learning. He reverse-engineered his car's CANBUS interface to collect labeled driving data, which includes a series of images from the front dash camera along with corresponding steering angles. Each image was captured at a rate of 30 frames per second, resulting in approximately 25 minutes of video data segmented into individual frame images. The steering angles, initially recorded in degrees in the data.txt file, begin with 0.0 degrees for the first 22 frames, indicating straight steering. To standardize the steering angle values, they were converted from degrees to radians. Sully Chen provides two datasets for self-driving purposes, with Dataset 1 comprising around 63,000 images totaling 3.1GB. This data was collected in the vicinity of Rancho Palos Verdes and San Pedro, California, and follows the format: filename.jpg angle, year-mm-dd hr:min:sec

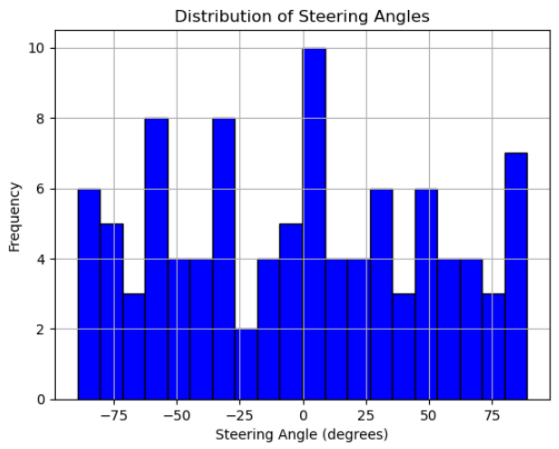

Fig. 1. Steering angle distribution before resize/normalization

The original images, initially sized at 256x455, are depicted in Fig 1, showcasing their histogram distribution. These images were resized to 66x200 using Scipy and visualized using Matplotlib, as illustrated in Fig 2, which displays the histogram distribution of the resized image. Normalization aids in reducing variations in illumination, contrast, and color balance, thereby improving the performance of a steering angle prediction model.

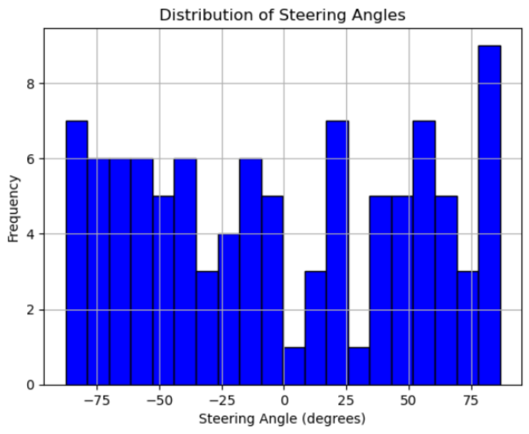

Fig. 2. Steering angle distribution after resize/normalization

Layer (Type)	Filter	Padding	Activation	Output Shape	Param#
Conv2d-1	3	Same	Relu	[-1, 16, 66, 200]	448
MaxPool2d-2	3	Same	-	[-1, 16, 33, 100]	0
Conv2d-3	3	Same	Relu	[-1, 32, 33, 100]	4,640
MaxPool2d-4	3	Same	-	[-1, 32, 16, 50]	0
Conv2d-5	3	Same	Relu	[-1, 64, 16, 50]	18,496
MaxPool2d-6	3	Same	-	[-1, 64, 8, 25]	0
Conv2d-7	3	Same	Relu	[-1, 32, 8, 25]	18,464
Conv2d-8	3	Same	Relu	[-1, 16, 8, 25]	4,624
MaxPool2d-9	-	-	Relu	[-1, 16, 4, 12]	0
Flatten-10	-	-	Relu	[-1, 768]	0
Linear-11			Relu	[-1, 32]	24,608
Linear-12			Relu	[-1, 10]	330
Linear-13	-	-	sigmoid	[-1, 1]	11

TABLE I
TABLE 1 SUMMARY OF THE LIGHTWEIGHT STEERING PREDICTION ALGORITHM

Dataset 1 comprises 63,000 images, with 20,000 allocated for training. Dataset 2 includes 45,000 images, with 5,000 designated for evaluation. In total, 250,000 images are utilized for both training and testing purposes. The initial 80% of the data is utilized for training, while the remaining 20% is set aside for testing, ensuring that the test data remains untouched during the training process.

B. Development Environment

The development environment utilized for this paper is jupyter notebook this open-source web application enables the creation and sharing of documents containing live code equations visualizations and narrative Text Jupyter Notebook supports multiple programming languages including python r Julia among others it allows the execution of python code and provides access to computing resources such as CPU and GPUs at no cost additionally Jupyter Notebook is compatible with TensorFlow pytorch various image processing libraries and deep learning libraries.

IV. METHOD

A. Architecture of the proposed lightweight models

Table 1 presents a summary demonstrating the architecture of the developed models for steering angle prediction as was already mentioned, each of the suggested models has the same feature extraction section, which is made up of thirteen hierarchical blocks. The first convolutional layer (conv1) applies 16 filters of size 3x3 with a stride of 1 and padding of 1. This layer extracts 16 different features from the input image. following the first convolutional layers called the input layer of our model. The second convolutional layer applies 32 3x3 filters with the same stride and padding. This layer extracts more complex features from the output of the first convolutional layer. The third convolutional layer applies 64 3x3 filters. This layer further increases the depth of feature extraction. The fourth convolutional layer applies 32 3x3 filters, and the fifth convolutional layer applies 16 3x3 filters. These layers continue to extract more and more abstract features.

Each convolutional layer is followed by a max-pooling layer (pool1, Pool2, Pool3, Pool5) with a kernel size of 2x2 and a stride of 2. Max pooling reduces the spatial dimension of feature maps while retaining the most important information. Before passing the feature map to a fully connected layer, the feature map is flattened into a one-dimensional vector using nn.Flatten(). This converts spatial information into a linear format. The smoothed feature vectors are passed through three fully connected layers (fc1, fc2, fc3) with output sizes of 32, 10, and 1, respectively.

B. Training Process

The Federated Averaging (FedAvg) algorithm [7] is modified, as demonstrated by Algorithm 1. The number of communication rounds T, the client-side epoch number, the server-side epoch number, and the server-side learning rate are the four key hyper-parameters in our FedAVG framework. The number of samples used in each local training iteration on the participating devices is determined by learning rate and batch size (B). The amount of data sampled for each computation depends on the batch size number.

In the FedAvg algorithm, a randomly selected number of clients is chosen on each round, and the gradient of the loss over all the data held by these clients is computed. In Algorithm 1, C controls the global batch size, where C=1 corresponds to full-batch gradient descent. To enhance results, we set C=1, meaning the server awaits updates from all participating clients during training. The entire training dataset is divided into subsets based on the number of clients, and both a global model and multiple client models are defined using the CNN architecture.

Initially, all client models are initialized with the same parameters as the global model. Optimizers are defined for each client model. The process iterates through 500 rounds. At each round of communication, a subset of clients is randomly selected, with 20 clients chosen per round. Each selected client performs updates by training on its local dataset for ten epochs using its local model and optimizer. The loss from each client's update is accumulated. Once all selected clients have completed their updates, the global model is updated by aggregating the weights from all selected clients using a server aggregate function, as defined in the algorithm. Finally, the global model is evaluated on the test dataset, and the test loss is recorded.

Algorithm 1 Federated Learning Server-Side Aggregation

```
1: procedure AGGREGATING($C, K$)
2:     Initialize Global Model $w^0$
3:     for each round $t = 1, 2, 3, \ldots, T$ do
4:         $m \leftarrow \max(C \times K, 1)$
5:         $S_t \leftarrow$ Random set of $m$ Clients
6:         for each client $K \in S_t$ do
7:             Send $w^{(t-1)}$ to client $K$
8:             $w_k^t \leftarrow \text{ClientUpdate}(K, w^{(t-1)})$
9:         end for
10:        $w^t = \sum_{k=1}^{K} \frac{n_k}{n} w_k^t$
11:    end for
12:    return $w^t$
13: end procedure
14: procedure CLIENTUPDATE($w^t$)
15:    $w \leftarrow w^t$
16:    $B \leftarrow$ Split into $P_k$ batch size $B$
17:    for each local epoch $i$ to $E$ do
18:        for each batch $b$ in $B$ do
19:            compute gradient $g^{(b,i)} \leftarrow \nabla L(w, b)$
20:            Update local model $w \leftarrow w - ng^{(b,i)}$
21:        end for
22:    end for
23: end procedure
```

C. *Loss function*

For model training, the Mean Square Error (MSE) loss function was utilized. This function imposes significant penalties on large errors due to the square term, emphasizing the importance of avoiding such errors. MSE is commonly employed as a loss function for regression tasks. Since steering angle prediction is a regression problem, Mean Squared Error (MSE) was chosen to quantify the error. Mean Squared Error (MSE) is calculated as:

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

where N represents the number of samples, y_i represents the actual value of the i-th sample, and $\hat{y}_i$ represents the predicted value of the i-th sample.

V. Result and Discussion

A. *Model Architecture Comparison*

Whereas the ResNet model consists of three blocks with 512, 100, and 50 parameters, the current steering angle prediction model has 204,661 parameters. There are 204,661 parameters in the CNN Model, 48,658,241 parameters in the LSTM Model, and 1,125,153 parameters in the CNN+LSTM. With 71,621 parameters, the suggested lightweight CNN architecture combines a convolutional neural network (CNN) to provide an effective and lightweight solution. This model offers a good balance between computational efficiency and model performance, making it appropriate for deployment in environments with limited resources or real-time applications. Fig 3 illustrates how the parameters of the onboard model and the suggested model are compared.

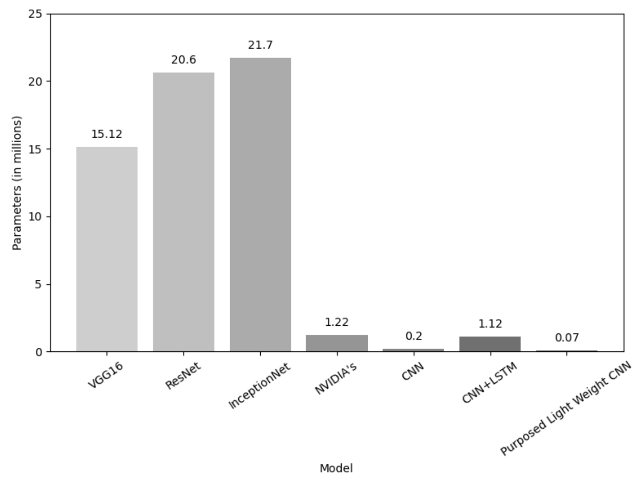

Fig. 3. Parameter comparison between onboard and Proposed Light Weight CNN

B. *Performance Analysis*

The VGG16 model for steering angle prediction demonstrated a training loss of 0.512 and a validation loss of 0.514. In contrast, the standalone CNN model showed higher losses, with a training loss of 2.3557 and a validation loss of 0.195. Integrating CNN with LSTM significantly improved the results, achieving a much lower training loss of 0.0866, but it resulted in a higher validation loss of 3.6485. The ResNet and Inception Net models exhibited training losses of 0.0529 and 0.0580, and validation losses of 0.0542 and 0.0591, respectively.

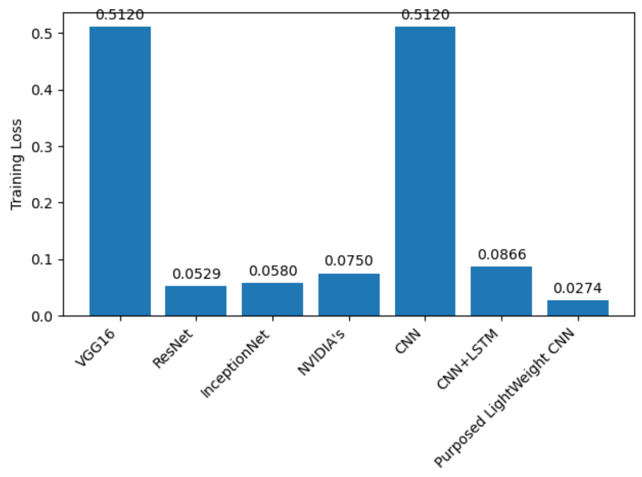

Fig. 4. Training loss/MSE comparison between onboard and purposed Light weight CNN

We propose a lightweight CNN model that achieves 0.0011 validation loss and 0.0274 training loss. The training loss comparison between the current model and our suggested model is depicted in Fig. 4, and the validation loss comparison between the current model and our suggested lightweight CNN model is shown in Fig. 5.

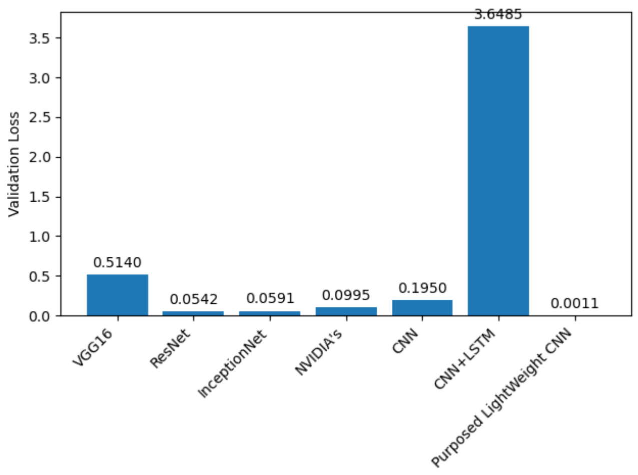

Fig. 5. Validation loss/MSE comparison between onboard and purposed Light weight CNN

Our lightweight steering prediction algorithm surpasses previous models by achieving superior training and validation losses, improved accuracy in steering angle predictions, and better overall loss metrics. Using a larger dataset provides a more extensive range of driving scenarios, thereby enhancing the model's generalization capabilities and prediction accuracy.

VI. CONCLUSIONS

The rise of autonomous systems, particularly in self-driving vehicles, has been fueled by machine learning research, reducing human workload by automating tasks. Steering angle prediction is vital for safe vehicle operation, helping with path following and obstacle avoidance. Models like DNN, ResNet, CNN, CNN+LSTM, and NVIDIA are commonly used for this task, but their large number of parameters leads to long training times and can affect model accuracy. Developing reliable, efficient models is crucial for enhancing the safety and performance of autonomous vehicles. This paper presents a lightweight CNN model for steering angle prediction, offering improved control accuracy over traditional CNNs. The model predicts accurate steering angles with minimal error for any track, using image preprocessing to enhance input data. Its low-dimensional architecture reduces training time without compromising accuracy. Additionally, the paper explores Federated Learning as an alternative to traditional methods, achieving better accuracy. Steering prediction is treated as a regression problem, and model performance is evaluated using mean square error (MSE).

VII. LIMITATION

Clients must frequently exchange model updates with the central server, which can be bandwidth-intensive, especially when dealing with large models or numerous devices. Malicious devices can send corrupted model updates, poisoning the global model and reducing its accuracy or introducing vulnerabilities.

VIII. FUTURE WORK

In the future implement the proposed algorithm in a trace-driven simulation environment for real-time testing on embedded systems.

REFERENCES

[1] H. Saleem, F. Riaz, L. Mostarda, M. A. Niazi, A. Rafiq, and S. Saeed, "Steering angle prediction techniques for autonomous ground vehicles: a review," *IEEE Access*, vol. 9, pp. 78567–78585, 2021.

[2] H. U. Ahmed, Y. Huang, P. Lu, and R. Bridgelall, "Technology developments and impacts of connected and autonomous vehicles: An overview," *Smart Cities*, vol. 5, no. 1, pp. 382–404, 2022.

[3] I. Wibawa, C. Ekaputri, *et al.*, "Speed and steering control system for self-driving car prototype," in *Journal of Physics: Conference Series*, vol. 1367, p. 012068, IOP Publishing, 2019.

[4] J. Maanpää, J. Taher, P. Manninen, L. Pakola, I. Melekhov, and J. Hyyppä, "Multimodal end-to-end learning for autonomous steering in adverse road and weather conditions," in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 699–706, IEEE, 2021.

[5] R. Valiente, M. Zaman, S. Ozer, and Y. P. Fallah, "Controlling steering angle for cooperative self-driving vehicles utilizing cnn and lstm-based deep networks," in *2019 IEEE intelligent vehicles symposium (IV)*, pp. 2423–2428, IEEE, 2019.

[6] M. Smolyakov, A. Frolov, V. Volkov, and I. Stelmashchuk, "Self-driving car steering angle prediction based on deep neural network an example of carnd udacity simulator," in *2018 IEEE 12th international conference on application of information and communication technologies (AICT)*, pp. 1–5, IEEE, 2018.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.

[8] F. S. Faizi and A. K. Alsulaifanie, "Steering angle prediction via neural networks," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 31, no. 1, pp. 392–399, 2023.

[9] J. Sokipriala, "Prediction of steering angle for autonomous vehicles using pre-trained neural network," *European Journal of Engineering and Technology Research*, vol. 6, no. 5, pp. 171–176, 2021.

[10] M. G. G, "International research journal of engineering and technology," *IRJET*, vol. 9, pp. 347–354, 2022.

[11] N. Yadav and R. Mody, "Predict steering angles in self-driving cars," 2017.

[12] V. Singhal, S. Gugale, R. Agarwal, P. Dhake, and U. Kalshetti, "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, 2019.

[13] P. Barua, J. C. Hagler, D. J. Lamb, and Q. Tian, "Towards greener and attention-aware solutions for steering angle prediction," *arXiv preprint arXiv:2211.11133*, 2022.

[14] V. Singhal, S. Gugale, R. Agarwal, P. Dhake, and U. Kalshetti, "Sully chen dataset," in *Sully Chen DataSet," Rancho Palos Verdes and San Pedro California. Link, 2018*, pp. 1–6, Rancho Palos Verdes and San Pedro California, 2018.