vehicles

Article

Autonomously Steering Vehicles Along Unmarked Roads Using Low-Cost Sensing and Computational Systems

Giuseppe DeRose, Jr. *,†, Austin Ramsey †, Justin Dombecki, Nicholas Paul and Chan-Jin Chung

Department of Math and Computer Science, Lawrence Technological University, Southfield, MI 48075, USA; aramsey@ltu.edu (A.R.); jdombecki@ltu.edu (J.D.); npaul@ltu.edu (N.P.); cchung@ltu.edu (C.-J.C.)
* Correspondence: gderose@ltu.edu
† These authors contributed equally to this work.

Abstract: The vast majority of autonomous driving systems are limited to applications on roads with clear lane markings and are implemented using commercial-grade sensing systems coupled with specialized graphic accelerator hardware. This research reviews an alternative approach for autonomously steering vehicles that eliminates the dependency on road markings and specialized hardware. A combination of machine vision, machine learning, and artificial intelligence based on popular pre-trained Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) was used to drive a vehicle along roads lacking lane markings (unmarked roads). The team developed and tested this approach on the Autonomous Campus Transport (ACTor) vehicle—an autonomous vehicle development and research platform coupled with a low-cost webcam-based sensing system and minimal computational resources. The proposed solution was evaluated on real-world roads and varying environmental conditions. It was found that this solution may be used to successfully navigate unmarked roads autonomously with acceptable road-following behavior.

Keywords: autonomous vehicles; Self-Drive vehicles; deep learning; convolutional neural networks; recurrent neural networks; image histogram matching

Citation: DeRose, G., Jr.; Ramsey, A.; Dombecki, J.; Paul, N.; Chung, C.-J. Autonomously Steering Vehicles Along Unmarked Roads Using Low-Cost Sensing and Computational Systems. *Vehicles* **2023**, *5*, 1400–1422. https://doi.org/10.3390/vehicles5040077

Academic Editors: Xianke Lin and Chao Shen

Received: 1 August 2023
Revised: 21 August 2023
Accepted: 22 August 2023
Published: 16 October 2023

1. Introduction

Advancements in technology and artificial intelligence has led to a growth in autonomous vehicle research and applications. These vehicles, also known as self-driving cars, have the potential to revolutionize transportation and greatly improve safety on the roads. However, achieving full autonomy is still a challenging task, and researchers and engineers continue to work on improving the performance of autonomous vehicle systems.

One area of focus in the development of autonomous vehicles is the use of machine learning algorithms, specifically Artificial Neural Networks (ANNs), to enable the vehicle to perceive its surroundings and make decisions based on that information. ANNs have been successful in a variety of applications, including image and video recognition, and have shown promising results in autonomous vehicle systems as well. The Autonomous Land Vehicle In a Neural Network (ALVINN) demonstrated in 1988 that it was possible to use artificial neural networks to steer a vehicle automatically by using a video feed of the road in front of it [1]. Since then, deep learning advances have allowed for the use of Convolutional Neural Networks (CNNs) in experiments, which are specifically designed for image-based tasks [2,3]. However, there is still much room for improvement in the performance of these models, particularly in terms of their ability to generalize to new situations and environments.

One approach for improving the performance of CNNs in autonomous vehicle systems is to use models which have been pre-trained on large data sets, allowing the models to learn general features and patterns that can be applied to a variety of tasks. This process, known as transfer learning, has been successful in a number of applications and has the

potential to greatly improve the performance of autonomous vehicles. In this research paper, we explored the use of pre-trained CNNs in autonomous vehicle systems and discuss strategies for improving their performance on unmarked roads. Additionally, we will demonstrate that the introduction of Recurrent Neural Networks (RNNs) will further improve the driving performance. The original work completed by Timmis, Paul, and Chung (denoted Deep Steer) had several limitations that are improved on by this research including undesired vehicle responses and high in-vehicle computational resources [4].

The goal of this work is to address some of the challenges and shortcomings of previous work and to provide a more-robust and reduced-cost autonomous solution. The key limitations of the previous research addressed in this research are: undesired vehicle response at driveways and intersections, restrictive neural network training environment, and high computational resources used during in-vehicle use. Please note that the research presented here is also referred to as "Deep Steer".

2. Materials and Methods

2.1. Solution Methodology

The solution methodology used in this research is shown in Figure 1. The first step, Data Collection, consisted of driving the test vehicle along numerous training routes and collecting forward-facing images along with the associated steering wheel angle to successfully follow the road curvature. The details of the Data Collection are reviewed in Section 3. Data Collection is followed by Model Training. In Model Training, the collected data are used to train various neural networks to understand and evaluate their ability to successfully predict the required steering wheel angle to navigate a given road. Model Training is outlined in Section 4. The last step is Self-Drive and Inferencing. In this step, the researchers tested the trained neural network on unseen road surfaces—inferring the necessary steering wheel angle to successfully navigate the road curvature in an autonomous manner. An overview of Self-Drive and Inferencing is provided in Section 5.2.

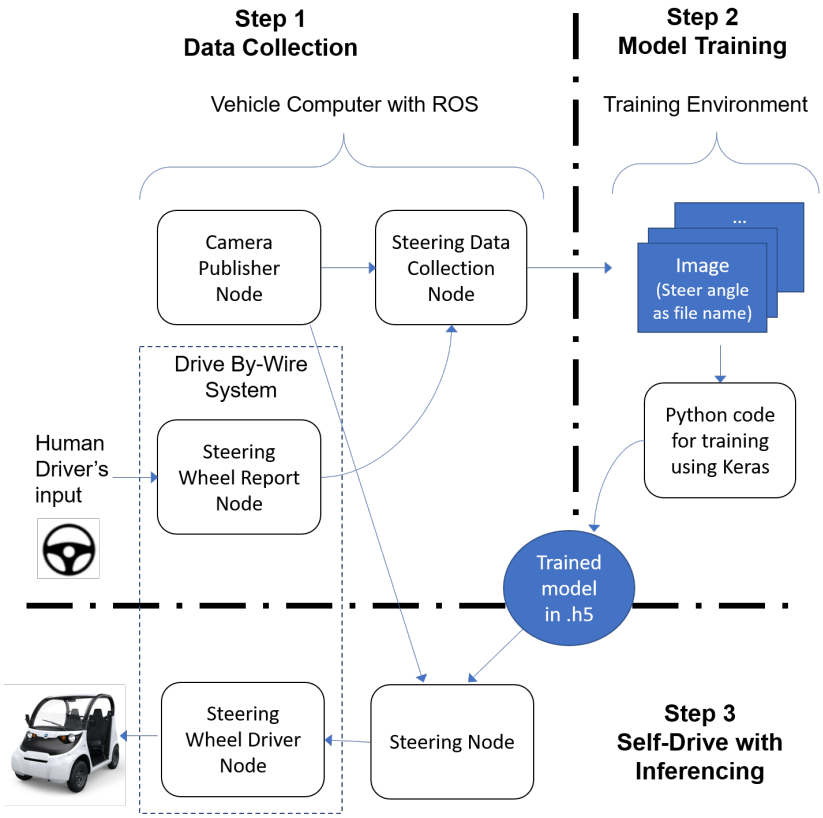

Figure 1. Vehicle integration—Deep Steer solution steps.

2.2. Hardware

A key element to this research is the ability to develop and evaluate various vehicle control strategies on an Autonomous Campus Transport (ACTor) vehicle shown in Figure 2. ACTor is a Polaris GEM 2 battery electric vehicle modified with a DataSpeed drive-by-wire system. A forward facing Genius webcam was mounted to the front of the vehicle, and a Logitech game pad was used for triggering the data acquisition system. The Genius webcam has a 120°-wide angle lens that supports 1080p Full HD video recording up to 30 fps. All necessary vehicle drivers, image processing code, and neural network evaluations were executed on an Ubuntu laptop. This laptop may be considered relatively old in terms of CPU technology having an Intel(R) Core(TM) i7-4500U CPU at 1.8 GHz, 4 cores, and a total physical memory of 8.0 GB.

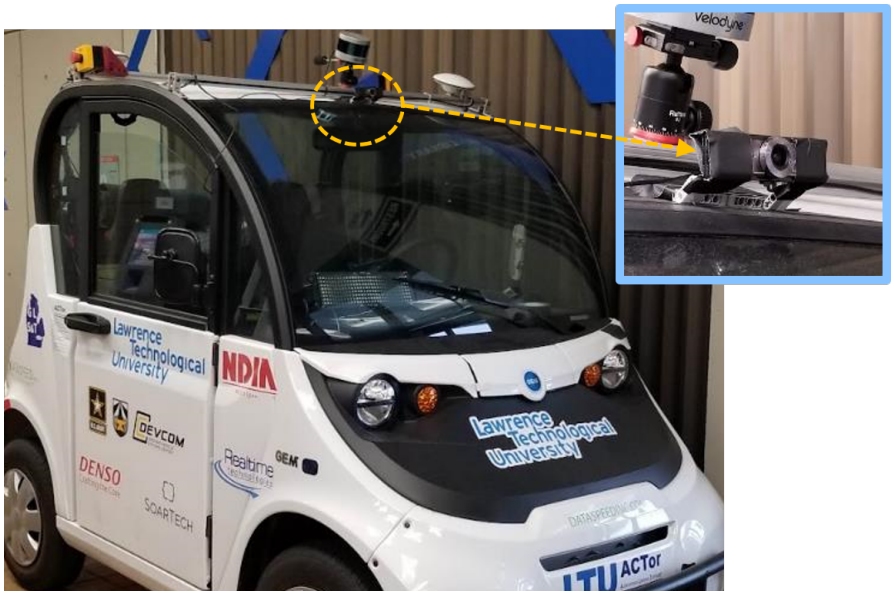

Figure 2. The ACTor 2 research vehicle.

2.3. Software

In-vehicle and offline (out-of-vehicle) software environments were required to develop Deep Steer solutions. A Robotic Operating System (ROS)-based [5] software platform was used in-vehicle to acquire neural network training data and execute the Deep Steer algorithms to steer the vehicle on private and public roads. The ROS platform supports the integration of the DataSpeed drive-by-system, webcam, image processing, and neural network evaluations [6]. The offline software platform was used to develop and train the neural networks. JupyterLab using Python was the primary operating environment and included the following packages: Keras and TensorFlow libraries for neural network construction and training, Pandas DataFrames for data manipulation, SciKit-Learn for dataset splitting, and OpenCV for general image processing and histogram matching.

3. Data Collection

As indicated on Figure 1, the first step in developing a Deep Steer solution is to collect vehicle test data on representative road surfaces. An ROS network with the following elements was developed to collect the data necessary to train the Deep Steer neural network:

- Camera Publishing Node: Receives forward-facing images sent by the Genius webcam and publishes the images to the ROS network.
- Steering Wheel Report Node: Interfaces with the DataSpeed drive-by-wire system and publishes the current steering wheel angle.

- Steering Data Collection Node: Receives webcam images and steering wheel angle data, down-samples the webcam image, and saves the image and steering wheel angle to a file.

The data flow of this ROS network is shown pictorially in Figure 3. Data recording was controlled by a Logitech-game-pad-based trigger. Once triggered, the ROS network would record consequent webcam images and associated steering wheel angle at a specified frequency. Although the system was capable of recording at frequencies approaching 30 Hz, the researchers found a sample rate of 5 Hz provided a dataset that was well suited for the vehicle speeds used in this application (a target vehicle speed of 5 mph). In addition, the vehicle was driven down the center of the road so that the solution could be easily extended to one-way and single-vehicle asphalt paths.

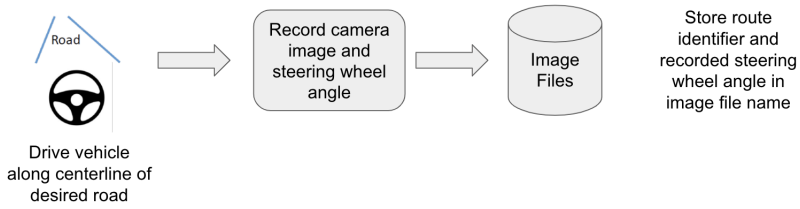

Figure 3. The neural network data acquisition model.

Please note that a vehicle path has an inherit dependency on the input steering wheel angle and vehicle speed. For vehicles that possess understeer-handling characteristics, the steering wheel angle required increases with increasing vehicle speed for a constant radius path [7]. Many traditional lane-keeping systems overcome this dependency by using the sensed line lanes to estimate road curvature and steer the vehicle based on a target vehicle yaw rate at a given speed; these systems require the vehicle speed to be input into the lane-keeping control system. The Deep Steer approach does not account for vehicle speed dependencies. It is assumed that all vehicle Data Collection and usage is performed at a constant vehicle speed. This is one of the primary limitations of the Deep Steer approach.

For this research, the data were collected on private and public roads (on and off the Lawrence Technological University campus). The routes shown in Figures 4 and 5 were specifically chosen for (1) the range of corner curvature and (2) the lack of lane markings with the goal of representing possible real-world scenarios, where a vehicle may need to navigate without the lane markings, providing assistance.

Figure 4. Example paths used for Data Collection and testing. To increase the dataset size, the vehicle was driven in both directions for each path.

Figure 5. Additional paths used for Data Collection and application routes. Routes in green comprised the training, validation, and testing dataset, while the route in red made up the application dataset. The application dataset was not used during the neural network training and validation process.

The data acquisition campaign concluded with over 35,000 images to be used for neural network training and testing. Figure 6 displays a sample distribution of the steering wheel angle for the training routes. Although this distribution is not the uniform shape often desirable for neural network training, they do capture an important element of the driving scenarios used in this research. A majority of the paths tested require very little steering wheel input angle to maintain course. This is inherently true for most driving situations and can be a major obstacle in the development of self-steer systems. Namely, training neural networks with non-uniform training sets can lead to prediction bias. Please note that similar distributions were observed in the testing and validation datasets.

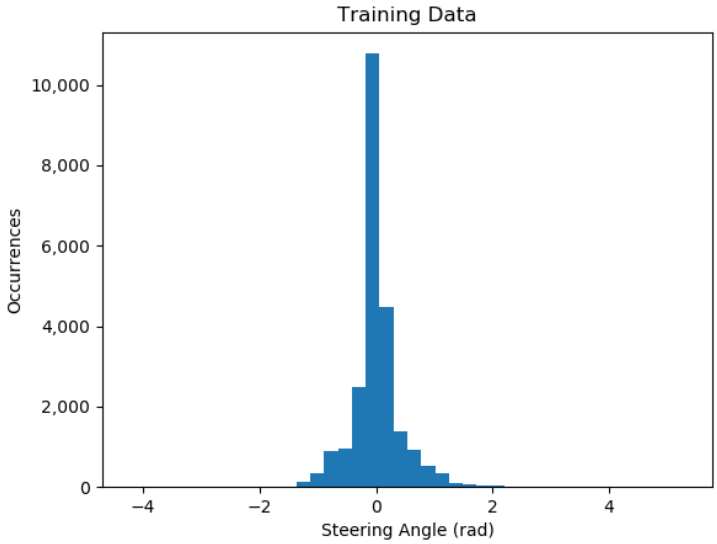

Figure 6. Training data steering wheel angle distribution.

4. Model Structure and Training Methodology

4.1. Model Structure

Several pre-trained neural networks were studied in this paper such as InceptionV3 [8], VGG16 [9], and VGG19 [9]. In order to compare and contrast the work completed here with the Deep Steer work of the previous authors, the same top structure was used with

all three pre-trained neural networks. The top layer consisted of a global averaging pool, a fully connected 1024 Rectified Linear Unit (ReLU) activation node layer, and a single fully connected output node to support regression analysis. An example of this neural network structure and model summary can be seen in Figure 7. The details of the network size and number of parameters are shown in Figure 8. Please note that the InceptionV3, VGG16, and VGG19 networks accept images of size (320 × 240), and the output of the full neural network is the steering wheel angle prediction in radians.

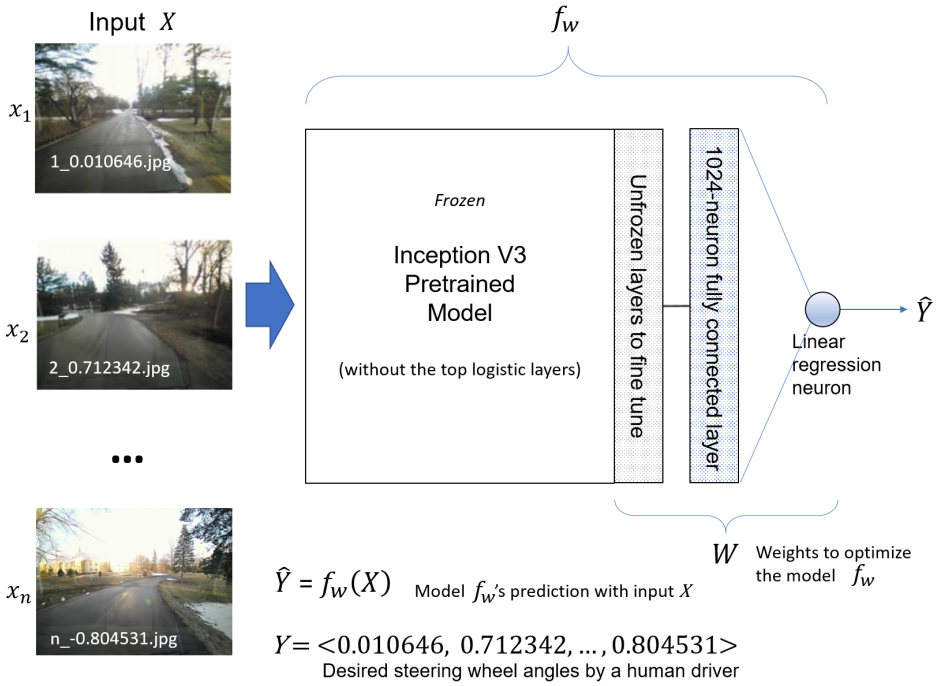

Figure 7. Model architecture.

Layer (type)	Output Shape	Param #
inception_v3 (Functional)	(None, 6, 8, 2048)	21,802,784
global_average_pooling2d	(None, 2048)	0
dense (Dense)	(None, 1024)	2,098,176
dense_1 (Dense)	(None, 1)	1,025

Total params: 23,901,985
Trainable params: 2,099,201
Non-trainable params: 21,802,784

Figure 8. Example neural network structure.

4.2. Model Training Methodology

To complete the second step for developing a Deep Steer solution, we must train the neural network with a representative dataset (webcam images and associated steering wheel data). In this research, a two-stage training approach including Feature Extraction and Fine-Tuning was used on all network structures evaluated [10,11]. The first stage of training, Feature Extraction, used the default weights in the pre-trained models and focused on training the top 1024-neuron fully connected layer and the output regression layer. The goal of this step is to use the pre-trained models to identify features of the input

images, for example road and driveway edges. The identified features were used as the inputs to train the added layers (fully connected layer and output regression layer). Please note that Feature Extraction does not modify the weights of the pre-trained models. Once the feature-extraction stage of training was complete, Fine-Tuning was implemented to further improve the network performance. During Fine-Tuning, the weights of the pre-trained network were optimized to further improve the full neural network's prediction performance.

A Mean-Squared Error (MSE) loss function comparing the recorded steering wheel angle and the predicted steering wheel angle (see Figure 7) was used in the neural network training exercises. The Keras RMSprop solver, used to train the network, utilizes a plain momentum optimization technique that reduces high variance in Stochastic Gradient Descent (SGD) and softens the convergence to help prevent overfitting [12]. Several different techniques were used during model training to facilitate accurate prediction while reducing the risk of over-training. For example, training optimization callbacks with Early Stopping [10] and Model Checkpointing [13] were used. Early Stopping was paired with validation loss callbacks to allow the model to stop training before the set number of epochs based on the validation data loss performance. Model Checkpointing was utilized to save the models with the lowest validation loss to minimize overfitting. Please note that the RMSprop solver used MSE loss functions; however, the Mean Absolute Error (MAE) loss and validation metrics were utilized for optimization of the Checkpointing and Early Stopping criteria measures. Figures 9–12 demonstrate the example Feature Extraction loss function, Fine-Tuning loss function, Feature Extraction validation, and Fine-Tuning validation metrics during RMS optimization, respectively. In all cases, the introduction of Fine-Tuning yielded minimal improvements to the training loss; however, more importantly, the validation loss function showed greater improvements. For this example, the validation loss function at the conclusion of Feature Extraction was approximately 0.4 radians (MAE); however, the validation loss function at the conclusion of Fine-Tuning reduced to approximately 0.01 radians (MAE)—a significant prediction improvement. In addition, the validation MAE demonstrated little evidence of overfitting. The overall validation performance and lack of overfitting instilled confidence in the trained neural network's ability to accurately predict the steering wheel angle on similar, but unseen roadways.

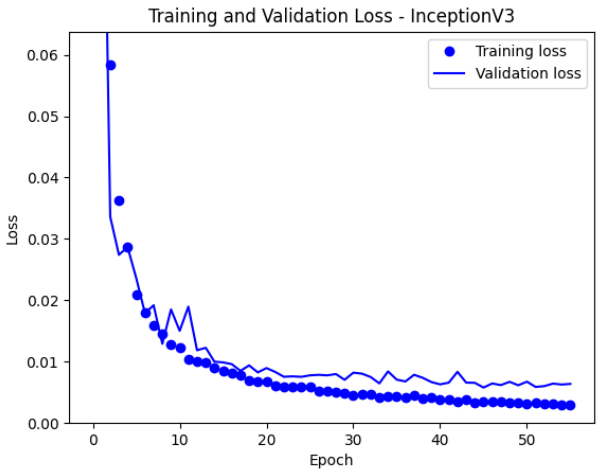

Figure 9. Example InceptionV3 Feature Extraction loss function history.

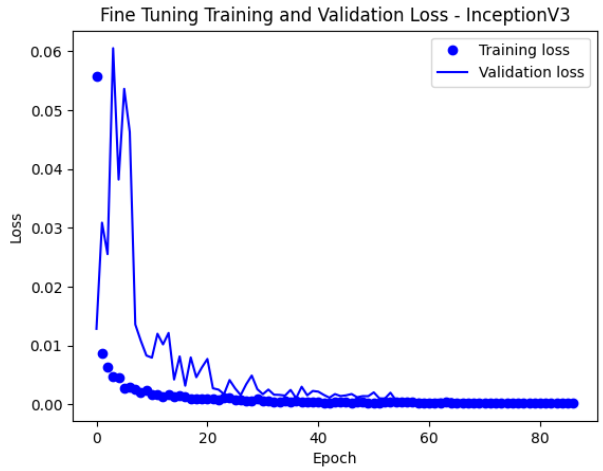

Figure 10. Example InceptionV3 Fine-Tuning loss function history.

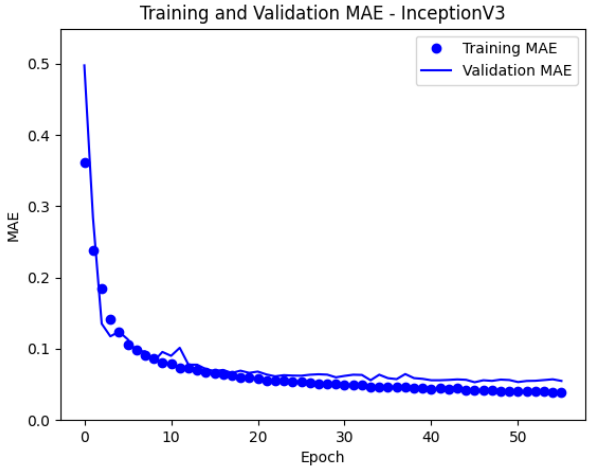

Figure 11. Example InceptionV3 Feature Extraction validation history.

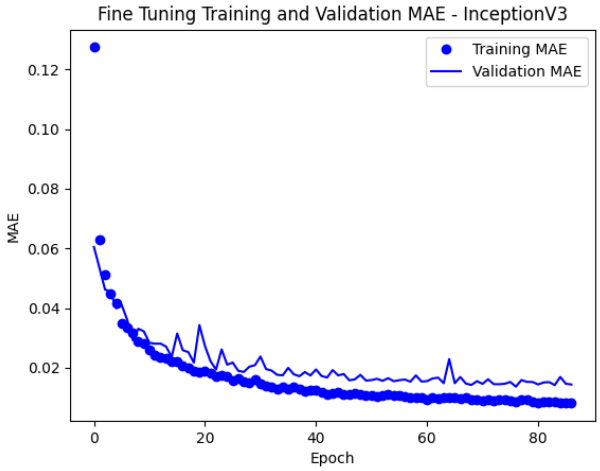

Figure 12. Example InceptionV3 Fine-Tuning loss validation history.

Despite the successful neural network training behavior exhibited above, the researchers encountered several issues in acquiring consistent datasets to generate neural networks that provide Deep Steer road-following behavior with the ACTor vehicle. One challenge was the dependency on the weather and lighting conditions. An attempt was made to ensure that all Data Collection was performed in ideal weather conditions to reduce glare and direct

sunlight towards the webcam. The researchers found that slightly overcast conditions were optimal, yielding minimum glare, reduced shadows, and the most-consistent image brightness. Although efforts were made to collect consistent images for model training, the researchers noticed reduced that steering wheel prediction accuracy occurred due to varying image brightness. To overcome this issue, image histogram matching was added to the neural network pre-processing to minimize the variation of image brightness. The image histogram matching was accomplished through the following steps outlined below [14,15]. An example of this process can be found in Figures 13–16.

Figure 13. Example of the histogram pre-processing technique used to balance images obtained from the front vehicle camera.

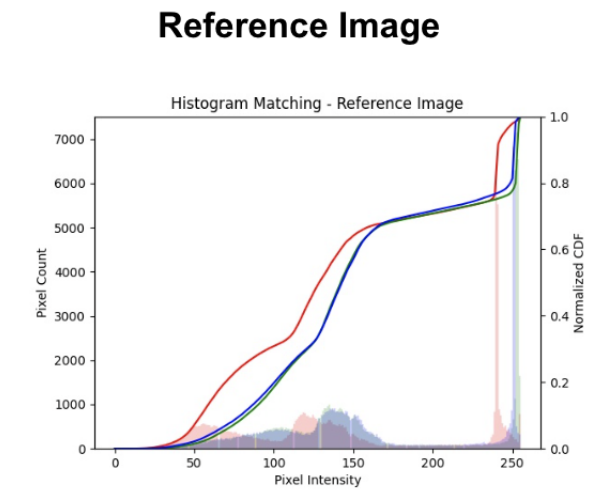

Figure 14. Histogram of the reference image used to balance images.

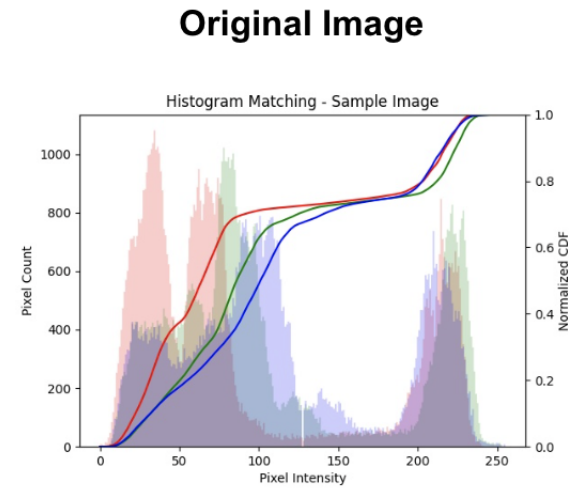

Figure 15. Histogram of an example original image.

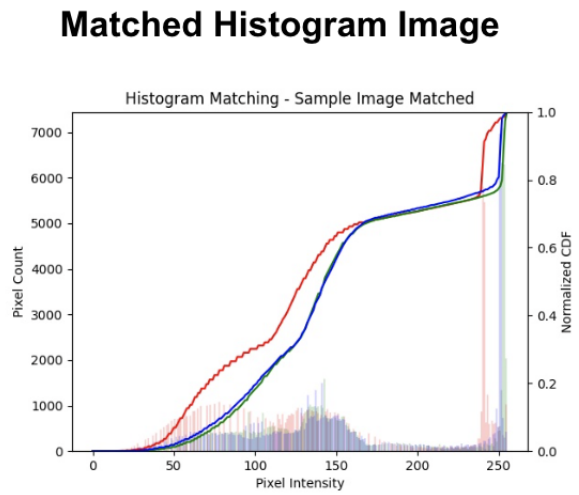

Figure 16. Histogram of the image after applying the balancing technique.

1. Obtain the histogram for both the input image and a reference image:

$$\begin{aligned} h(i) &= p_i \\ &= (\#\text{ of pixels of intensity } i/\text{total }\#\text{ of pixels}) \\ &= \text{Probability Density Function (PDF)} \end{aligned} \tag{1}$$

2. Calculate the Cumulative Distribution Function (CDF) for both the input image and the reference image.

$$\begin{aligned} CDF = H(j) &= \sum_{i=0}^{j} h(i) \\ \text{where } j &= 0, 1, ..., 245, 255 \end{aligned} \tag{2}$$

3. Calculate the transformation T to map the old intensity values to the new intensity values for both the input image and reference image:

$$T(j) = floor((K-1) * CDF_j) \tag{3}$$

4. Use the transformed intensity values for both the input image and reference image to map the intensity values of the input image to the new values.

The authors observed several benefits with the histogram-matching implementation. Histogram matching successfully reduced the variation in neural network steering wheel angle prediction in non-ideal environmental conditions. This reduction in steering prediction variations led to a smoother vehicle response and improved road-following behavior.

5. Results

5.1. Model Training Results

A model training investigation was performed for the InceptionV3-, VGG16-, and VGG19-based networks using the model structure and training methodology reviewed in the previous section. Data from all the training routes were used to train, validate, and test the neural networks. The final proportion of the datasets was 80% training and validation and 20% testing. Additional data were collected on the application routes. These data were not used as part of the training and validation process; however, they were used as truly "unseen" data to evaluate the effectiveness of steering prediction. A summary of the size of the training, validation, testing, and application datasets is shown in Table 1.

Table 1. Number of training, validation, testing, and application images used for neural network development and testing.

Training	Validation	Testing	Application
23,623	5906	7383	7692

The training, validation, and testing performances of the InceptionV3-based network are displayed in Figures 17–19. Similar diagrams are provided for the VGG16- and VGG19-based models in Appendices A and B.

The training results showed that the InceptionV3 model performed well in predicting the vehicle steering angle for the training, validation, and testing datasets. The VGG16 and VGG19 models provided similar performance as the InceptionV3 model, but some important trends were noticed. For all models, the training data steering predictions possessed less error on average than the test data steering predictions. This was anticipated prior to the model training activity; however, one may notice a telling difference in the magnitude of the larger steering angle responses. Namely, the InceptionV3-based model demonstrated slightly lower error for mid-range Steering Wheel Angle (SWA) responses $|SWA| \leq 40°$, but much lower error in predicting the steering responses at the larger steering inputs $|SWA| > 40°$ when compared to the VGG16 and VGG19 models. The behavior resulted in a lower overall test MAE of 3.10° for the InceptionV3-based model. A summary of the model training, validation, and testing MAE results for all three networks is displayed in Table 2.

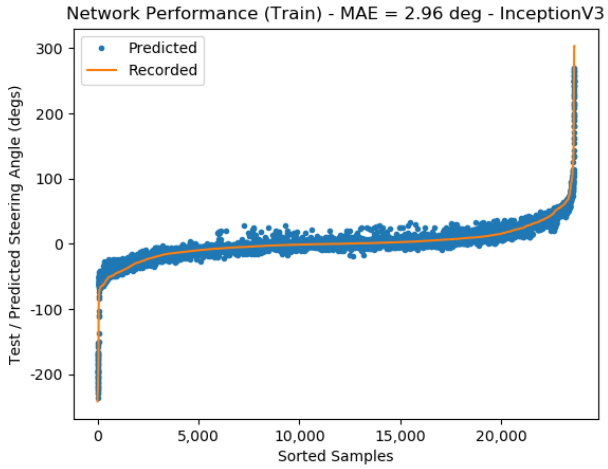

Figure 17. InceptionV3-based neural network training performance.

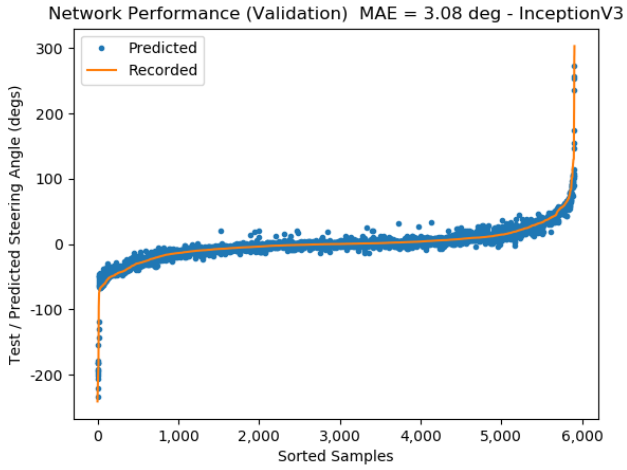

Figure 18. InceptionV3-based neural network validation performance.

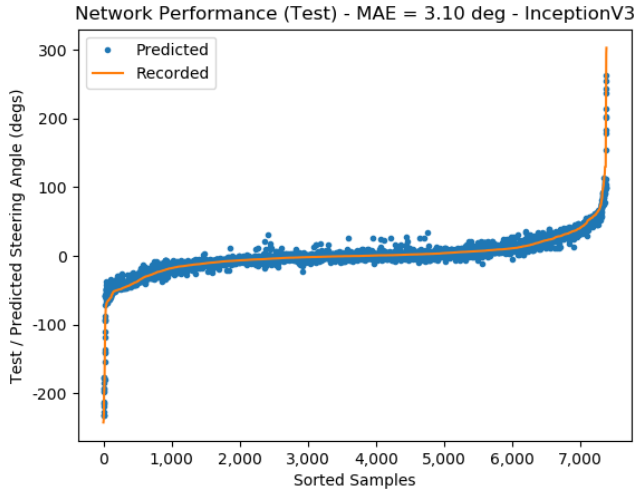

Figure 19. InceptionV3-based neural network testing performance.

Table 2. Training, validation, and testing MAE (deg).

	InceptionV3	VGG16	VGG19
Training	2.96	3.82	4.33
Validation	3.08	3.81	4.29
Testing	3.10	3.88	4.41

An additional neural network performance analysis was performed using traditional R^2 metrics. The results for this analysis are shown in Table 3. This data also demonstrates the performance of the InceptionV3 based model versus the VGG16 and VGG19 models. Lastly, a graphical representation of the R^2 behavior may be viewed in Appendix C.

Table 3. Training, validation, and testing R^2 performance.

	InceptionV3	VGG16	VGG19
Train	0.97	0.95	0.91
Validation	0.97	0.94	0.90
Testing	0.97	0.94	0.90

5.2. Model Application Results—Self-Drive with Inference

The final step in developing and testing a Deep Steer solution is to test the self-driving behavior with Inferencing. To accomplish this, an ROS network with the following elements was developed to steer the vehicle using neural network outputs based on processed webcam images (down-sampled and histogram balanced):

- Camera Publishing Node: Receives forward-facing images sent by the Genius webcam and publishes the images to the ROS network (the same node used the Data Collection implementation).
- Steering Node: Receives images from the webcam, down-samples the images, applies histogram matching, evaluates the neural network, and publishes the desired steering wheel angle.
- Steering Wheel Driver Node: Publishes desired steering wheel angle to the DataSpeed drive-by-wire system to actuate the ACTor steering system.

Figure 20 displays the data flow for the Deep Steer testing on the ACTor vehicle. The webcam was integrated with an ROS camera publishing node, which generated forward road images at approximately 30 Hz. To prepare road images for evaluation by the neural network, all images were reduced to 320 × 240 px. This image size was used to maintain consistency with the previous Deep Steer research [4] and reduce the

computational resources required in the vehicle. Please note that a further reduction in image size may be investigated in future research. In an effort to further reduce the computational needs of the ROS network, camera images were skipped to provide actual processing of the camera images in a range of 5–10 Hz. Each camera image was then processed with a histogram-matching algorithm with the reference image in Figures 13–16. Next, the processed image was evaluated by the trained neural network to produce a desired steering wheel angle. Lastly, the desired steering wheel angle was processed by the drive-by-wire system to steer the vehicle.

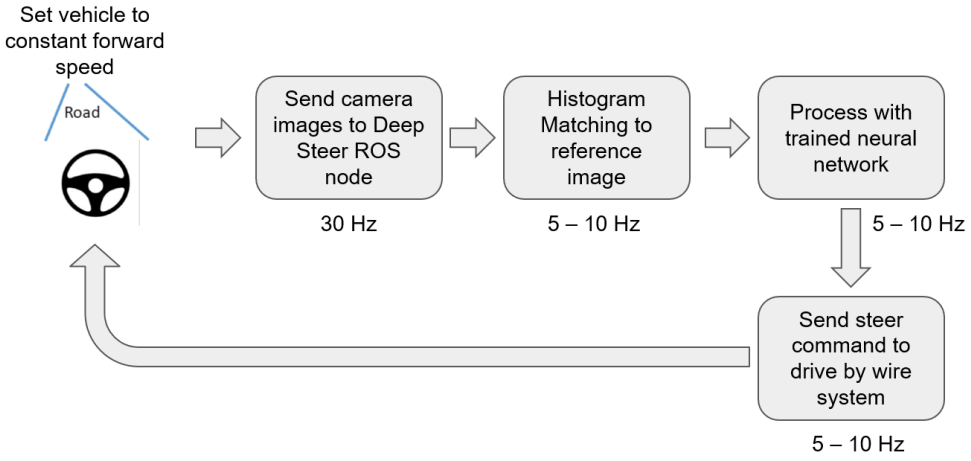

Figure 20. Neural network steering angle prediction.

The trained neural networks were next exercised on the application route dataset to ensure that Deep Steer could predict the steering request with reasonable error characteristics to support fully autonomous physical testing. For this analysis, the vehicle was driven manually along the application route while recording the front webcam image data and the associated driver steering input. These data were then used to evaluate the accuracy of the trained neural networks, and the results of the steering angle prediction model are displayed in Figure 21. The InceptionV3 model demonstrated a 4.25° MAE, while the VGG16 and VGG19 models provided MAEs of 5.17° and 5.05°, respectively. It can be noted that this is a great improvement from the original work completed by Timmis, Paul, and Chung, where the predicted steering angle yielded a 15.2° error on average for similar testing [4]. Again, it is important to note that images collected from this route were not used in the model training and validation and were completely "unseen" by the neural network.

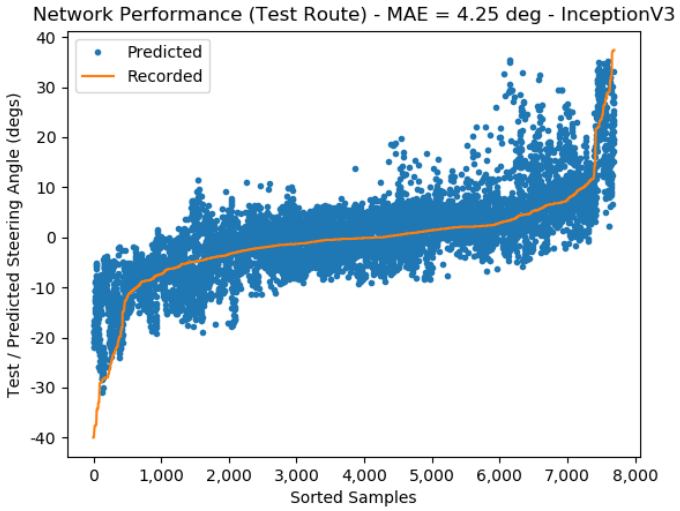

Figure 21. InceptionV3-based neural network performance on unseen, application route data.

Given the overall steering prediction demonstrated in Figure 21, the ACTor vehicle was tested in a fully autonomous mode, using the Deep Steer algorithm to steer the vehicle. The DataSpeed longitudinal controller was leveraged to ensure a constant vehicle speed of approximately 5 mph. When using the InceptionV3-based model, the vehicle followed the application route roadway. In addition, the vehicle successfully traveled through an intersection and was not steered off path due to the existence of driveways. The authors provided video evidence of the vehicle's performance on the application route [16]. The observed behavior demonstrated a clear improvement in undesired vehicle responses due to intersections and driveways when compared to the original work completed by Timmis, Paul, and Chung [4]. Unfortunately, the performance of the VGG16- and VGG19-based models did exhibit the same road-following performance. These models exhibited good road-following behavior, but were definitely influenced by the presence of intersections and some driveways. Overall, the InceptionV3-based model provided the most-robust road-following behavior with minimal influence from intersections and driveways.

Additional subjective tests were performed on the application route at vehicle speeds approaching 20 mph [16]. The resulting tests showed successful road-following behavior with lower stability. Driver interventions were necessary to maintain the vehicle on the roadway, especially in high road curvature areas. This performance demonstrated one of the shortcomings of Deep Steer—an inherit dependence on vehicle speed.

One of the goals of this research was to deliver a robust, fully autonomous steering behavior using minimal sensing and computational resources. The use of a simple wide-angle webcam definitely meets the low-cost sensing requirements. Similar webcams can be purchased for under USD 50. In order to minimize the overall in-vehicle computational resources, the authors investigated varying the sampling rate from the webcam and the associated rate of steering commands. Tests below 5 Hz led to a jerking response of the vehicle, while tests above 10 Hz led to lagging in the steering predictions due to image queue buffering in the ROS network. From this investigation, it can be concluded that driving the vehicle at 5 Hz provided a smooth vehicle response and minimal computational resources.

5.3. Leveraging Recurrent Neural Networks

Along with the other pre-trained neural networks studied in this paper, an application of recurrent neural networks was studied. To promote consistent comparisons, the same top structure was used; however, additional layers were added to support the RNNs. For this work, the top layer consisted of a global averaging pool, a fully connected 1024 Rectified Linear Unit (ReLU) activation node layer, a reshape layer, a SimpleRNN layer, and a single fully connected output node. An example of this neural network structure and model summary is displayed in Figure 22. For consistency with previous work, the recurrent neural network accepts images of size (320 $\times$ 240), and the output of the full neural network is the steering wheel angle prediction in radians.

For this analysis, the same application route dataset was used as the other models studied in this paper to evaluate the accuracy of the trained neural networks, and the results of the steering angle prediction model are displayed in Figure 23. The RNN model demonstrated a 3.82° MAE. It can be noted that was an improvement from the other neural networks studies in this paper. A summary of the model training, validation, and testing MAE results for the network is displayed in Table 4. Controlling the data feed of the new RNN-based model was slightly altered by adding a skip and append function, which takes a data frame and start index as the input. It returns a data frame starting at the specified index, including only every n images. The batch size is the number of images fed into the neural network at a time, during training, which was optimized. This is significant when working with an RNN layer as the model optimizes with the memory of the previous image. It is important to mention that the RNN model provides a robust alternative to the InceptionV3-based model with respect to road-following behavior with minimal

influence from intersections and driveways. An important aspect of the RNN-based model is that it allowed for smoother re-centering of the steering wheel when exiting sharper turns. The authors provided video evidence of the vehicle performance on the application route [17].

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_v3 (Functional)   (None, 6, 8, 2048)        21,802,784

 global_average_pooling2d (  (None, 2048)              0
 GlobalAveragePooling2D)

 dense (Dense)               (None, 1024)              2,098,176

 reshape (Reshape)           (None, 1, 1024)           0

 simple_rnn (SimpleRNN)      (None, 128)               147,584

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 24,048,673 (91.74 MB)
Trainable params: 2,245,889 (8.57 MB)
Non-trainable params: 21,802,784 (83.17 MB)
_________________________________________________________________
```

Figure 22. RNN model architecture.

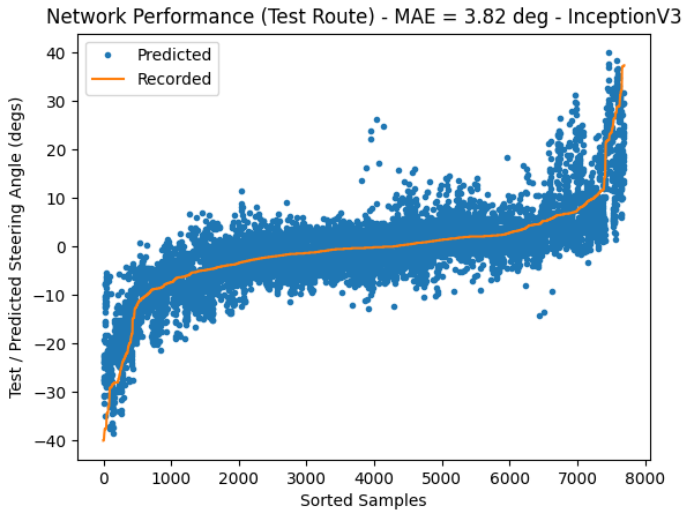

Figure 23. RNN performance on unseen, application route data.

Table 4. Training, validation, testing, and test route MAE (○).

	Recurrent Neural Network
Training	2.51
Validation	2.64
Testing	2.67
Test Route	3.82

6. Future Work

Over the course of this research, the authors noted several items that may be investigated in further detail. One such area is the inherit dependency on the input steering wheel angle and vehicle speed. The authors would like to examine methods for applying Deep Steer to a varying vehicle speed, perhaps leveraging an approximate vehicle understeer gradient to predict steering angles at higher vehicle speeds. In addition, the authors suggest

continuing research on optimizing the proposed solution. This work shows that steering a vehicle autonomously using only forward-facing road images and minimal computational resources is achievable; however, there are several areas where additional savings may be found, for example reducing the amount of training data needed to train the neural network to minimize the Data Collection step. To support this effort, a reduction in the processed image size (currently 320 × 240) may also in investigated. Reducing the processed image size may allow for fewer training images and faster neural network training. Lastly, if smaller processed image sizes yield acceptable results, a further reduction of in-vehicle computational resources may be achieved. The authors look forward to further investigating these topics.

7. Conclusions

The vehicle was tested in real-time on the application route shown in red in Figure 5 and evaluated subjectively. The InceptionV3-based model performed better than the VGG16 and VGG19 models. The InceptionV3 tests demonstrated that the vehicle was able to follow the route very well with limited steering errors; the vehicle followed the curved route as intended through intersections and past driveways. The VGG16- and VGG19-based models did not behave as well. For both models, the neural network occasionally under-predicted the steering required for sharper corners. The steering wheel angle under prediction led to a vehicle trajectory heading off the road, leading to test termination. Based on the observed vehicle behavior, the InceptionV3-based model is the recommended model. To further improve the vehicle road-following behavior, a recurrent neural network was added to the model architecture. The addition of the RNN showed improved prediction error in all phases of testing and training, as well as demonstrated improved road-following behavior during subjective testing.

Author Contributions: Conceptualization, G.D.J., C.-J.C., and N.P.; methodology, G.D.J., C.-J.C., and N.P.; software, G.D.J. and A.R.; validation, G.D.J. and J.D.; formal analysis, G.D.J.; investigation, G.D.J.; resources, G.D.J. and A.R.; data curation, G.D.J. and A.R.; writing—original draft preparation, G.D.J. and A.R.; writing—review and editing, G.D.J., J.D., and C.-J.C.; visualization, G.D.J., A.R., and C.-J.C.; project administration, C.-J.C.; funding acquisition, C.-J.C. All authors have read and agreed to the published version of the manuscript.

Funding: This specific research received no external funding. However, the following companies and organizations sponsored the acquisition of the ACTor electric vehicles, sensors, on-board computers, by-wire systems, other parts, and vehicle maintenance cost: U.S. Army Ground Vehicle Systems Center (GVSC), DENSO, SoarTech, Realtime Technologies, GLS&T, Hyundai MOBIS, DataSpeed, NDIA Michigan, Veoneer, and Lawrence Technological University.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Datasets are not publicly available due to privacy concerns.

Acknowledgments: The authors thank the Department of Mathematics and Computer Science and the College of Arts and Sciences at Lawrence Technological University for administrative and financial support. The authors also thank Ryan Kaddis, LTU CS AI Robotics Lab Assistant.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ACTor	Autonomous Campus Transport
ALVINN	Autonomous Land Vehicle In a Neural Network
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
MAE	Mean Absolute Error
MSE	Mean-Squared Error

RMS	Root Mean Square
RNN	Recurrent Neural Network
ROS	Robotic Operating System
SGD	Stochastic Gradient Descent
SWA	Steering Wheel Angle

Appendix A

This Appendix provides the training, validation, testing, and application route error for the VGG16-based Deep Steer model.

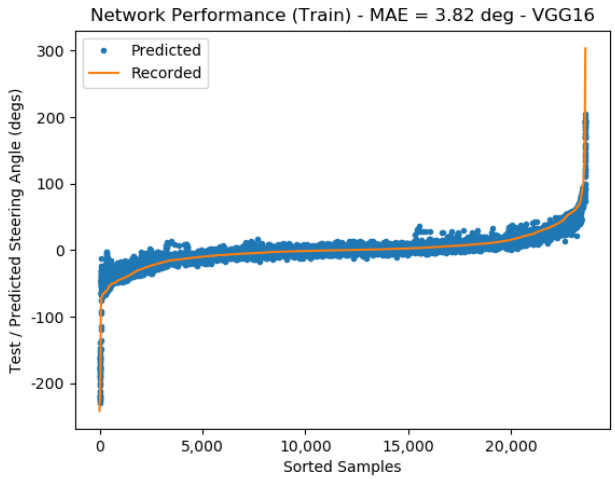

Figure A1. VGG16-based neural network training performance.

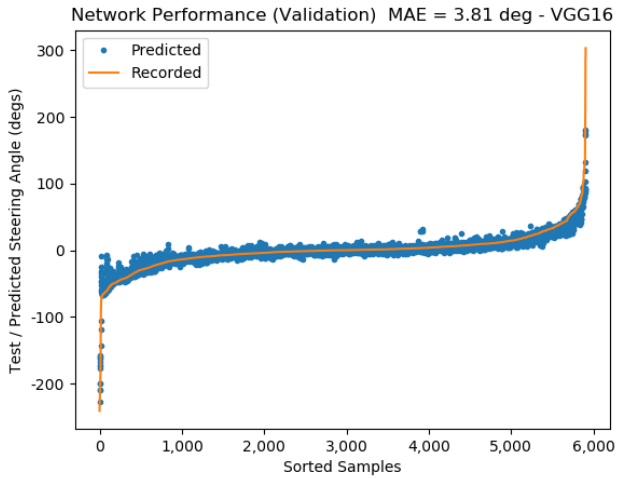

Figure A2. VGG16-based neural network validation performance.

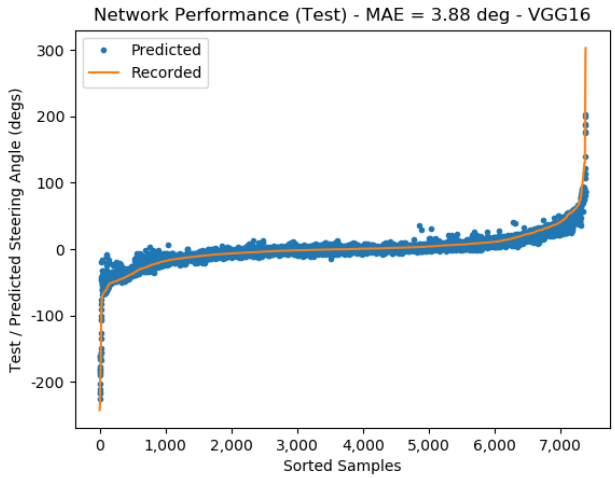

Figure A3. VGG16-based neural network testing performance.

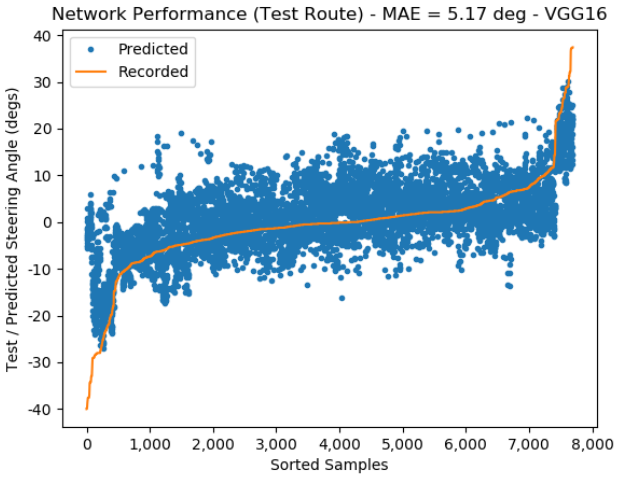

Figure A4. VGG16-based neural network performance on unseen, application route data.

Appendix B

This Appendix provides the training, validation, testing, and application route error for the VGG19-based Deep Steer model.

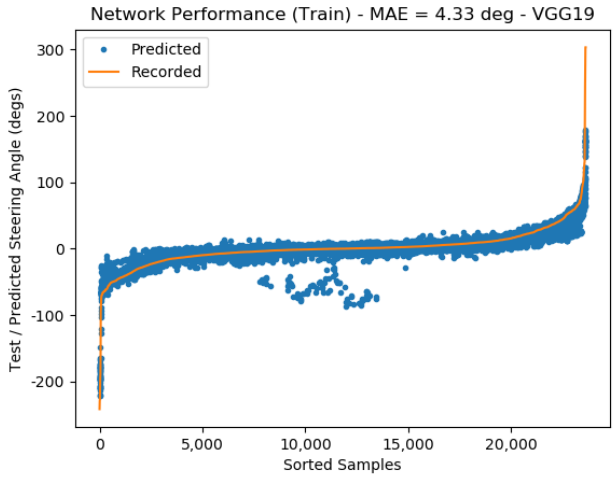

Figure A5. VGG19-based neural network training performance.

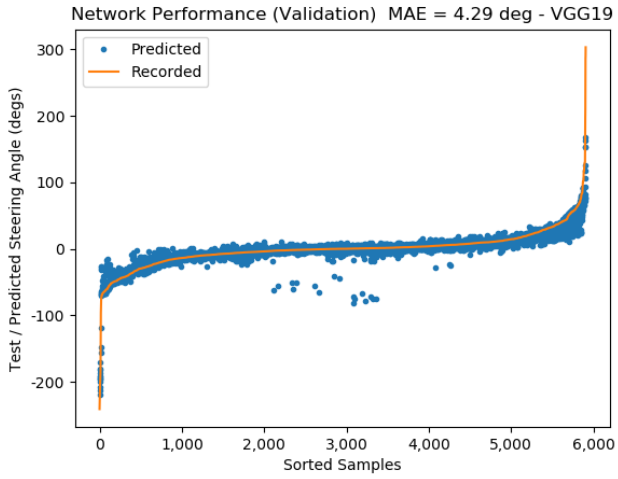

Figure A6. VGG19-based neural network validation performance.

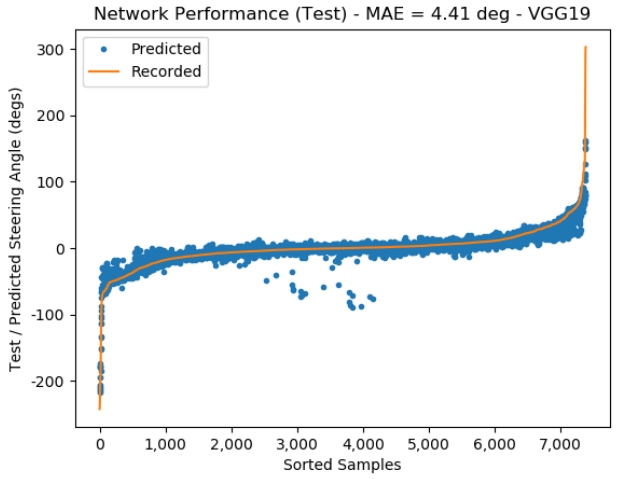

Figure A7. VGG19-based neural network testing performance.

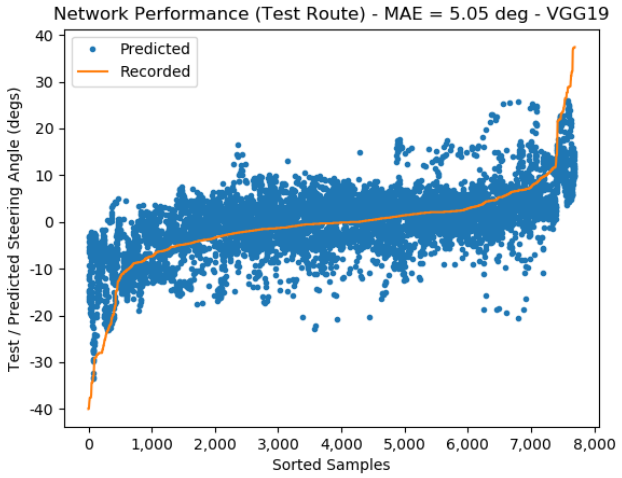

Figure A8. VGG19-based neural network performance on unseen, application route data.

Appendix C

This appendix provides training, validation and test R^2 performance of the three models investigated.

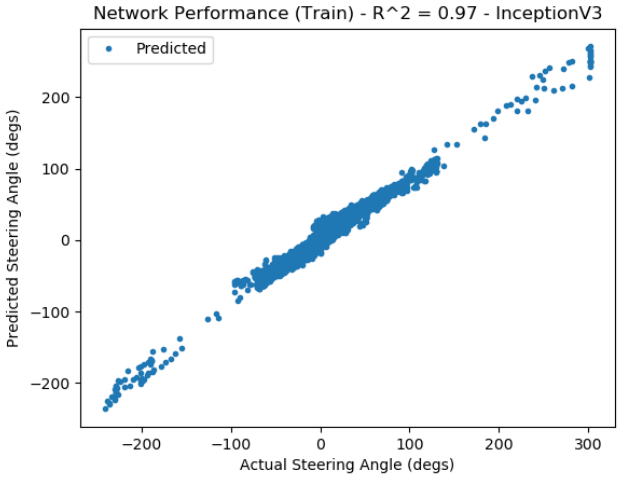

Figure A9. InceptionV3-based neural network training performance.

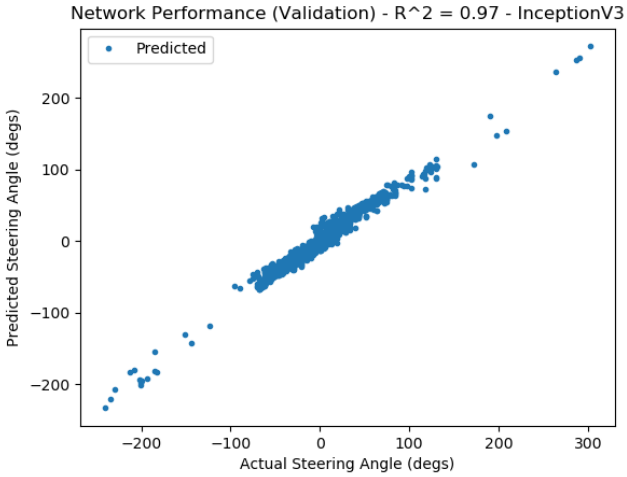

Figure A10. InceptionV3-based neural network validation performance.

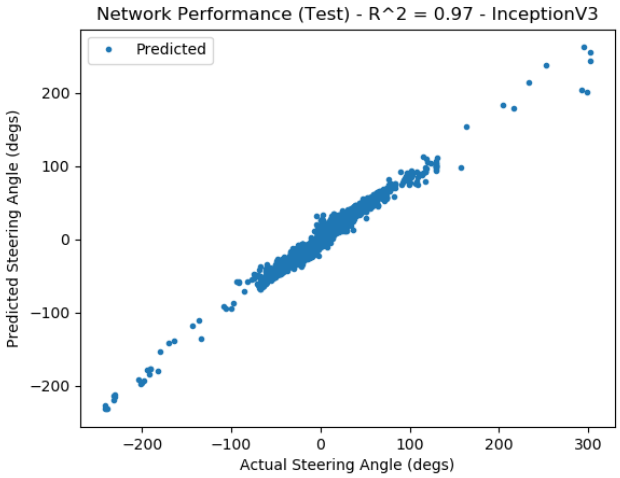

Figure A11. InceptionV3-based neural network testing performance.

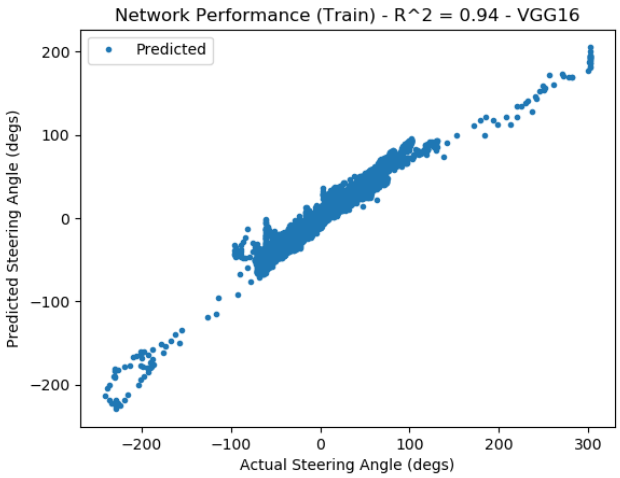

Figure A12. VGG16-based neural network training performance.

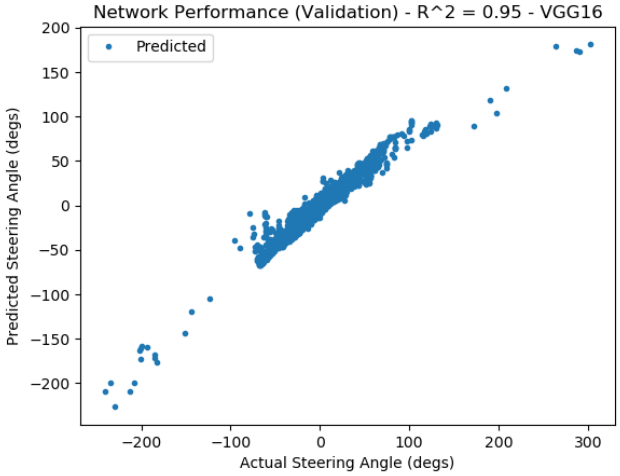

Figure A13. VGG16-based neural network validation performance.

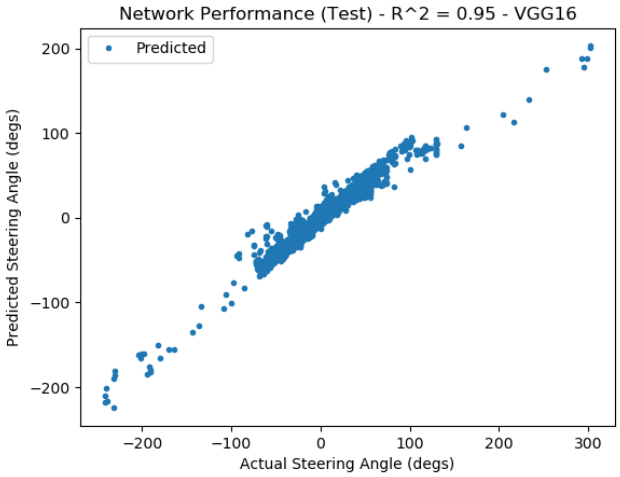

Figure A14. VGG16-based neural network testing performance.

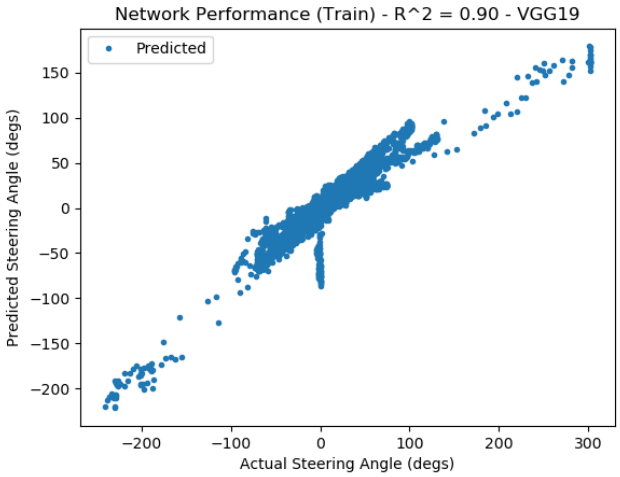

Figure A15. VGG19-based neural network training performance.

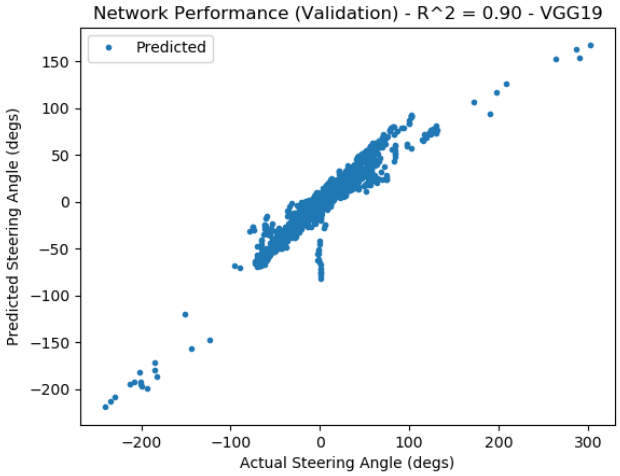

Figure A16. VGG19-based neural network validation performance.

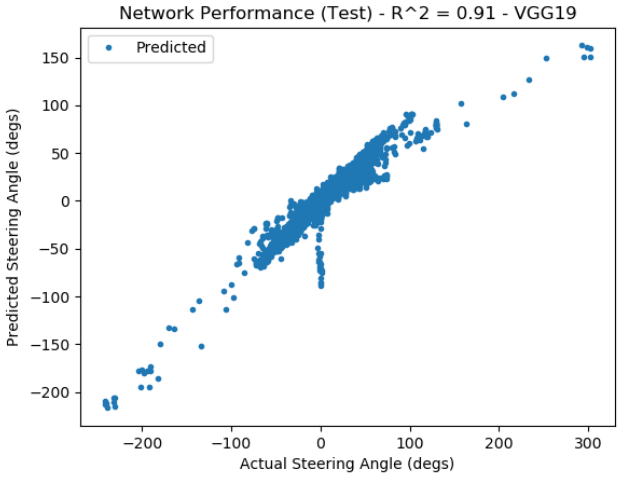

Figure A17. VGG19-based neural network testing performance.

References

1. Pomerleau, D.A. Alvinn: An Autonomous Land Vehicle in a Neural Network. In *Advances in Neural Information Processing Systems*; 1988. Available online: https://proceedings.neurips.cc/paper/1988/file/812b4ba287f5ee0bc9d43bbf5bbe87fb-Paper.pdf (accessed on 31 July 2023).
2. Wang, Y.; Shen, D.; Teoh, E.K. Lane detection using spline model. *Pattern Recognit. Lett.* **2000**, *21*, 677–689. [CrossRef]
3. Wang, Y.; Teoh, E.K.; Shen, D. Lane detection and tracking using b-snake. *Image Vis. Comput.* **2004**, *22*, 269–280. [CrossRef]
4. Ian Timmis, N.P.; Chung, C.J. Teaching vehicles to steer themselves with deep learning. In Proceedings of the 2021 IEEE In-ternational Conference on Electro Information Technology (EIT), Mt. Pleasant, MI, USA, 14–15 May 2021.
5. Quigley, M. ROS: An open-source Robot Operating System. In Proceedings of the ICRA Workshop on Open Source Software, Kobe, Japan, 12–17 May 2009; Volume 3.
6. Robotic Operating System. Available online: https://www.ros.org (accessed on 31 July 2023).
7. Gillespie, T.D. *Fundamentals of Vehicle Dynamics*; Premiere Series Books; SAE International: Warrendale, PA, USA, 1992.
8. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. *CoRR arXiv* **2015**, arXiv:1512.00567.
9. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
10. Chollet, F. *Deep Learning with Python*; Simon and Schuster: New York, NY, USA, 2018.
11. Li, Z.; Hoiem, D. Learning Without Forgetting. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 614–629.
12. Kathuria, A. Intro to Optimization in Deep Learning: Momentum, RMSPROP and Adam. Available online: https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam (accessed on 31 July 2023).
13. Brownlee, J. How to Checkpoint Deep Learning Models in Keras. Available online: https://machinelearningmastery.com/check-point-deep-learning-models-keras. (accessed on 31 July 2023).
14. Addison, A. Difference between Histogram Equalization and Histogram Matching. Available online: https://automaticaddison.com/difference-between-histogram-equalization-and-histogram-matching. (accessed on 31 July 2023).
15. Shapira, D.; Avidan, S.; Hel-Or, Y. Multiple histogram matching. In Proceedings of the 2013 IEEE International Conference on Image Processing, Melbourne, Australia, 15–18 September 2013. [CrossRef]
16. Chung, C.J.; DeRose, G. 2023. Available online: https://youtu.be/aZJaIr5ilos (accessed on 31 July 2023).
17. Chung, C.J.; Ramsey, A. 2023. Available online: https://youtu.be/yGqs5YsmQiU (accessed on 31 July 2023).