Article

Not peer-reviewed version

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

Giuseppe DeRose [*] , Austin Ramsey , Justin Dombecki , Nicholas Paul , Chan-Jin Chung [*]

Posted Date: 3 August 2023

doi: 10.20944/preprints202308.0258.v1

Keywords: Autonomous vehicles; self-drive vehicles; deep learning; convolutional neural networks; recurrent neural networks; image histogram matching

Article

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

Giuseppe DeRose J. [1,†,*], Austin Ramsey [1,†], Justin Dombecki [1], Nicholas Paul [1] and Chan-Jin Chung [1]

1 Department of Math and Computer Science, Lawrence Technological University, Southfield, MI 48075, USA
* 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 (CNN) and Recurrent Neural Networks (RNN) was used to drive a vehicle along roads lacking lane markings (unmarked roads). The team developed and tested this approach on the ACTor (Autonomous Camus TranspORt) 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

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 show 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 data collection are reviewed in Section 3. Data Collection is followed by Model Training. In Model Training, the collected data is 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.

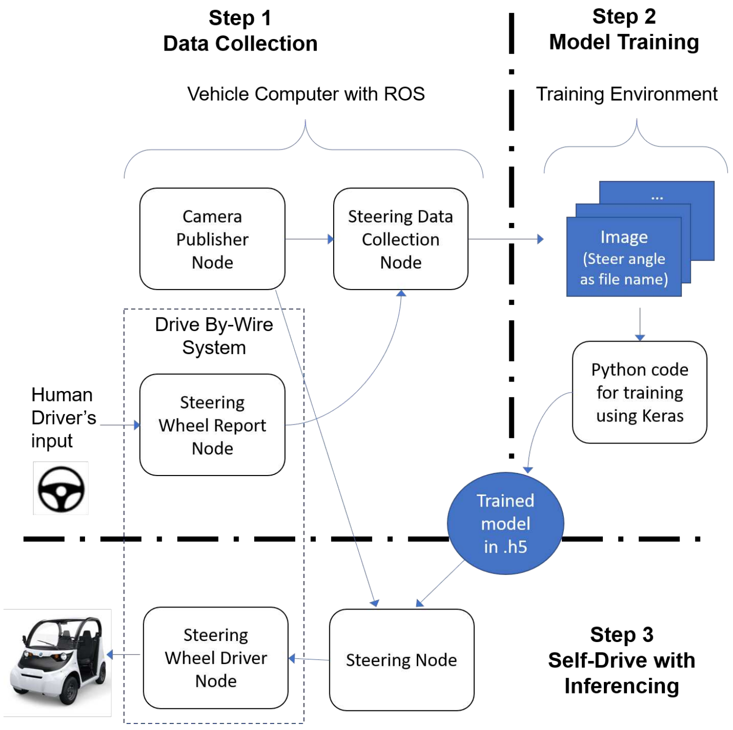

Step 1
Data Collection
Step 2
Model Training
Vehicle Computer with ROS
Training Environment
Camera Publisher Node
Steering Data Collection Node
...
Image (Steer angle as file name)
Drive By-Wire System
Human Driver's input
Steering Wheel Report Node
Python code for training using Keras
Trained model in .h5
Steering Wheel Driver Node
Steering Node
Step 3
Self-Drive with Inferencing

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 30fps. 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 a Intel(R) Core(TM) i7-4500U CPU at 1.8GHz, 4 cores, and total physical memory of 8.0 GB.

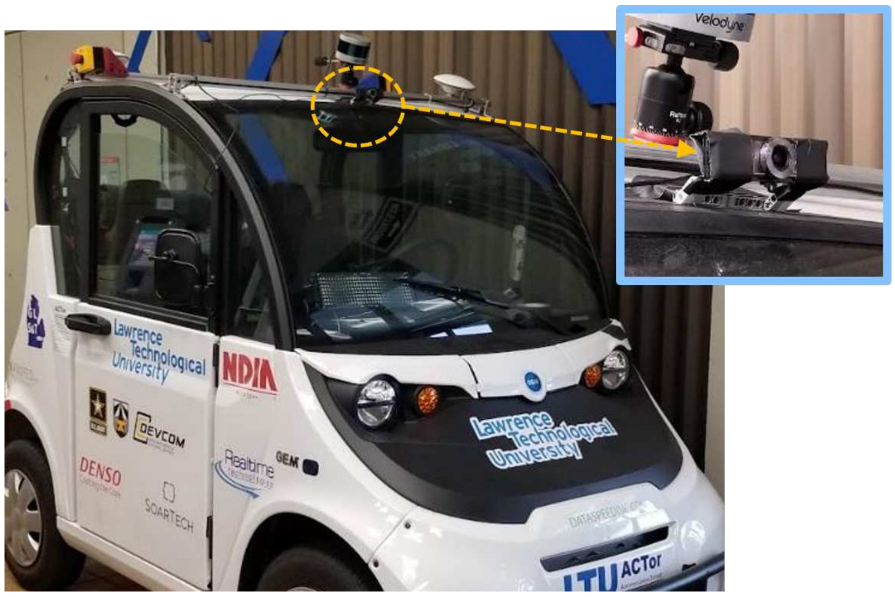


Figure 2. The ACTor 2 research vehicle

2.3. Software

In-vehicle and offline (out-of-vehicle) software environments were required to develop Deep Steer solutions. A Robotic Operating System (ROS) [5] based 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 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, 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. A 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 file.

The data flow of this ROS network is shown pictorially in Figure 3. Please note that the 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 data set that was well suited for the vehicle speeds used in this application (a target vehicle speed of 5 mi/hr). In addition, the vehicle was driven down the center of the road so that the solution may more easily extending to one-way and single vehicle asphalt paths.

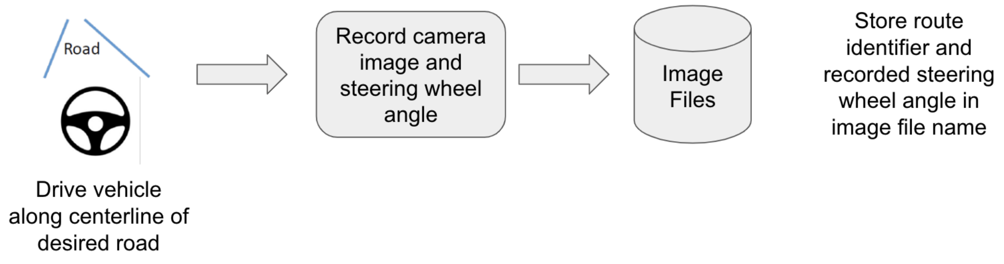

Road
Drive vehicle along centerline of desired road
Record camera image and steering wheel angle
Image Files
Store route identifier and recorded steering wheel angle in image file name

Figure 3. The neural network data acquisition model.

For this research, data was collected on private and public roads (on and off the Lawrence Technological University campus). The routes shown in Figure 4 and Figure 5 were specifically chosen for (1) range of corner curvature and (2) 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.

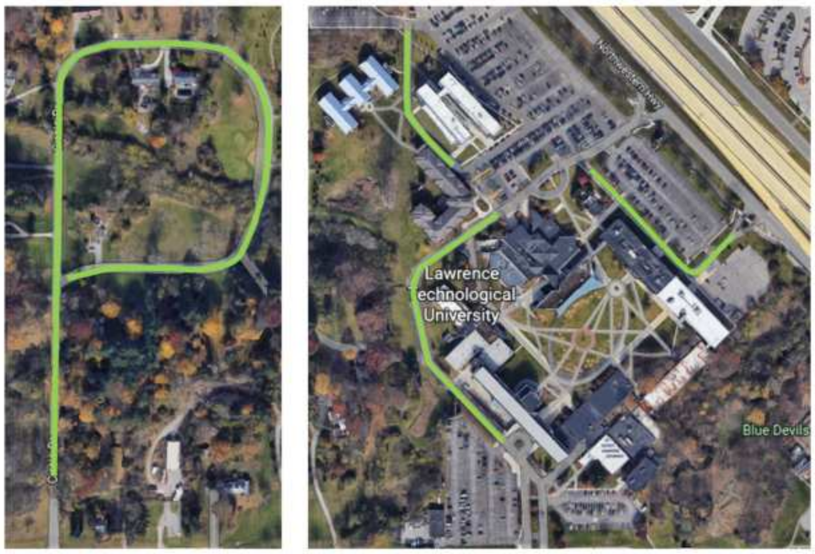


Figure 4. The paths used for data collection and testing. To increase the data set size, the vehicle was driven in both directions for each path. Routes in blue comprised the training data set, while the application route in green made up the testing data set.

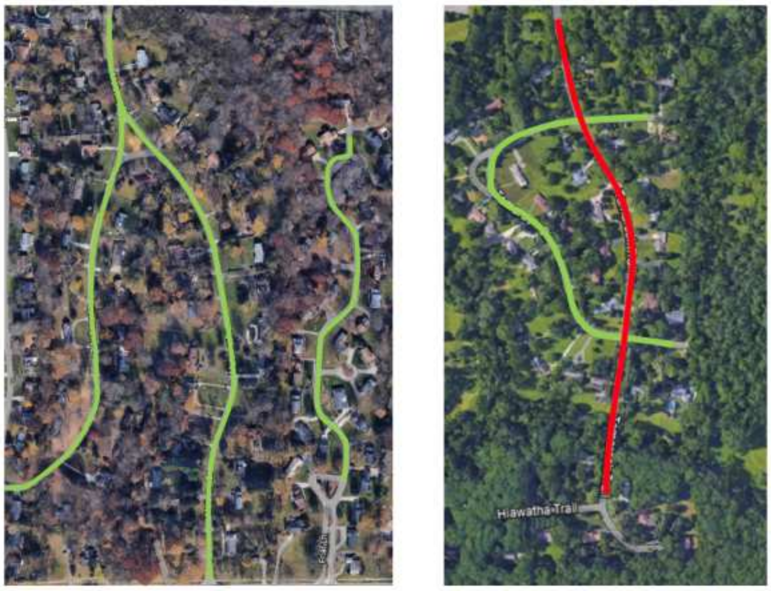


Figure 5. The paths used for data collection and application. Routes in green comprised the training data set, while the application route in red made up the application data set. This data set was not used during neural network training and validation process

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 these distributions are 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 is a major obstacle in development of self-steer systems. Namely, training neural networks with non-uniform training sets can lead to prediction bias. Similar distributions may be noted in the validation and testing data sets displayed in Figures 7 and 8, respectively.

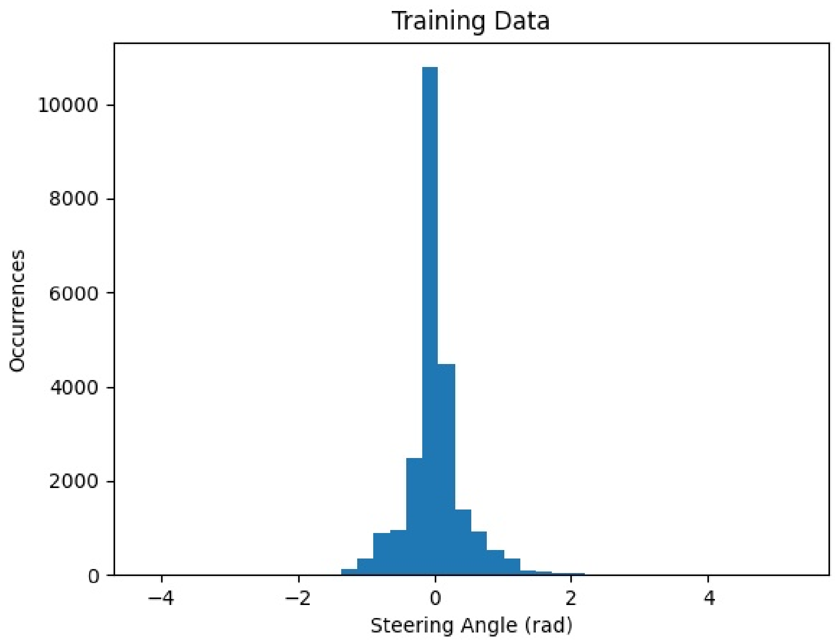


Figure 6. Training data steering wheel angle data distribution.

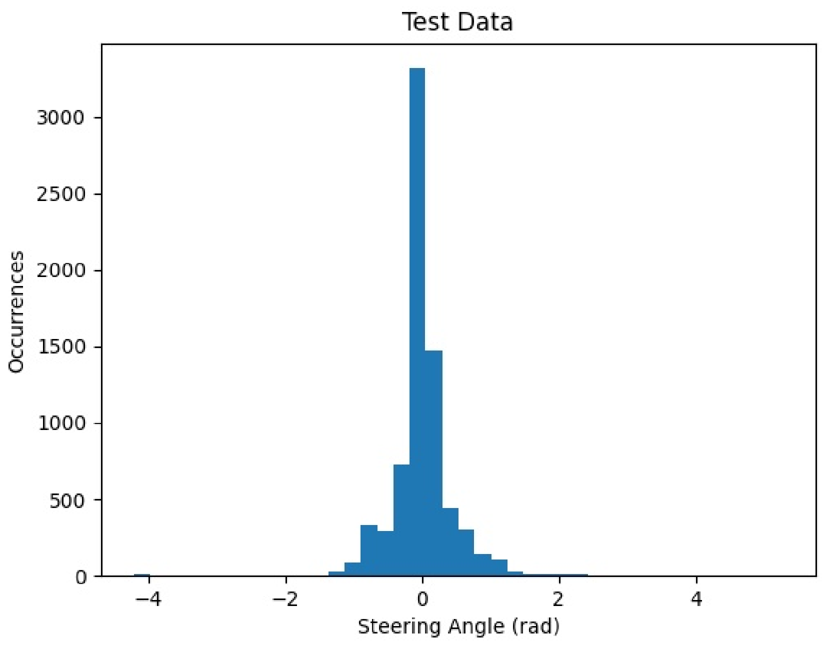


Figure 7. Test data steering wheel angle data distribution.

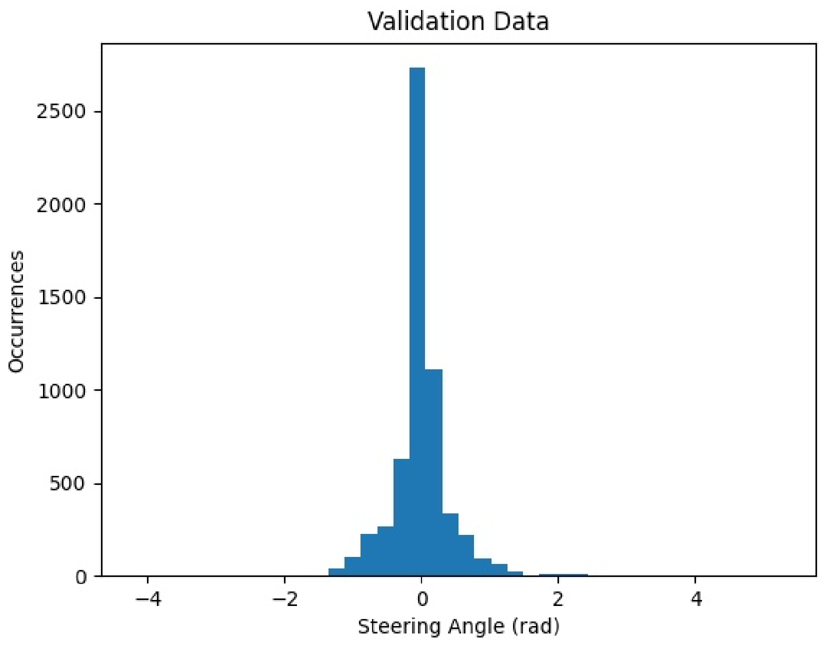


Figure 8. Validation data steering wheel angle data distribution.

4. Model Structure and Training Methodology

4.1. Model Structure

Several pre-trained neural networks were studied in this paper such as InceptionV3 [7], VGG16 [8], and VGG19 [8]. 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 9. The details of the network size and number of parameters are shown in Figure 10. Please note that InceptionV3, VGG16, and VGG19 networks accept images of size (320 x 240) and the output of the full neural network is the steering wheel angle prediction in radians.

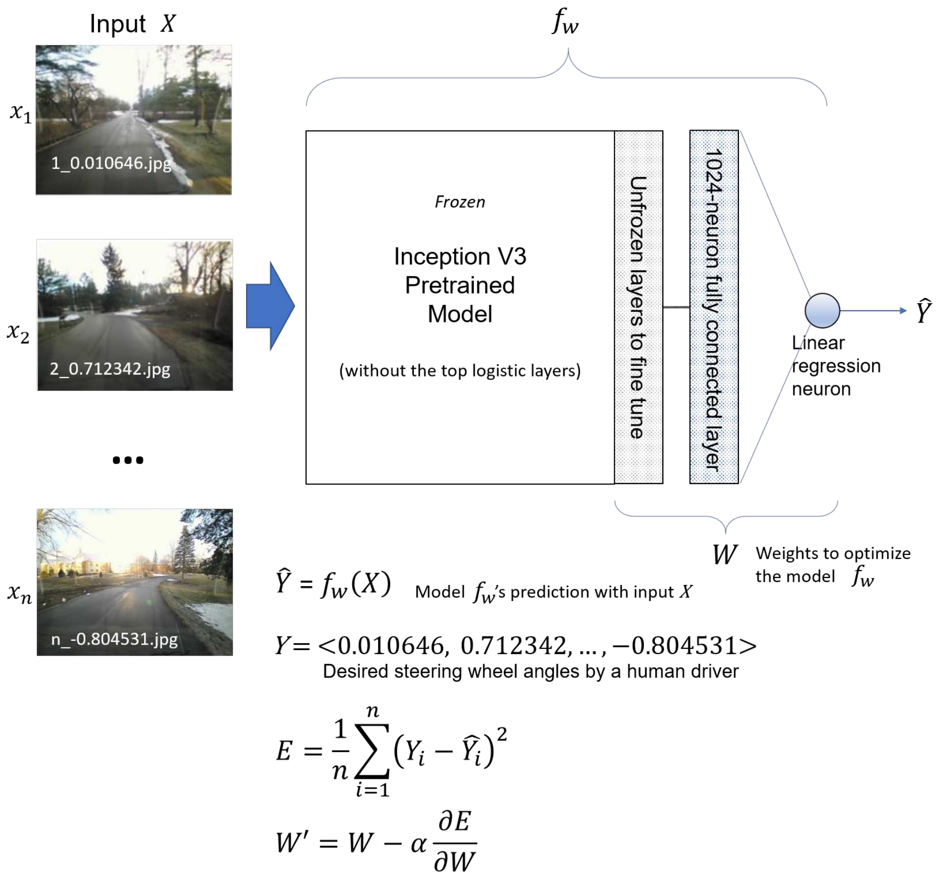
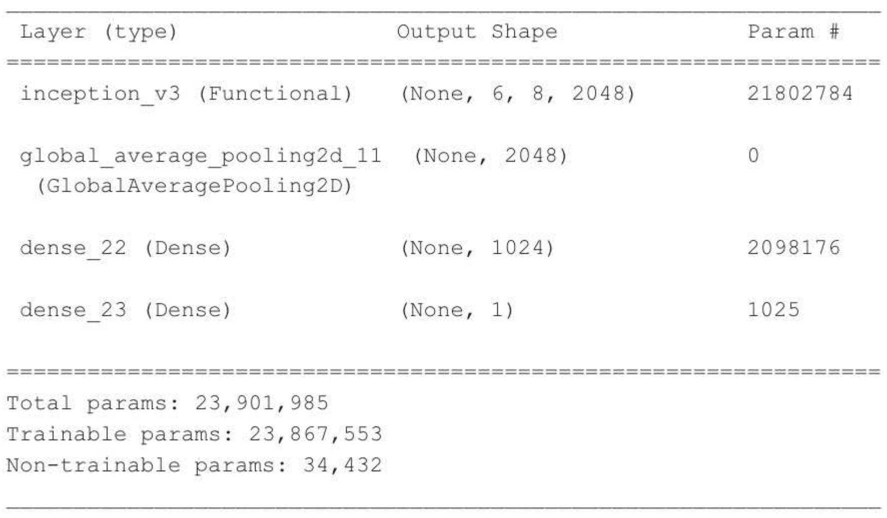

Input X
f_w
x_1
1_0.010646.jpg
x_2
2_0.712342.jpg
...
x_n
n_-0.804531.jpg
Frozen
Inception V3 Pretrained Model
(without the top logistic layers)
Unfrozen layers to fine tune
1024-neuron fully connected layer
Linear regression neuron
$\hat{Y}$
W Weights to optimize the model f_w
$\hat{Y} = f_w(X)$ Model f_w's prediction with input X
$Y = <0.010646, 0.712342, \ldots, -0.804531>$
Desired steering wheel angles by a human driver
$E = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$
$W' = W - \alpha \frac{\partial E}{\partial W}$

Figure 9. Model architecture.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_v3 (Functional)   (None, 6, 8, 2048)        21802784

 global_average_pooling2d_11  (None, 2048)             0
  (GlobalAveragePooling2D)

 dense_22 (Dense)            (None, 1024)              2098176

 dense_23 (Dense)            (None, 1)                 1025

=================================================================
Total params: 23,901,985
Trainable params: 23,867,553
Non-trainable params: 34,432
_________________________________________________________________
```

Figure 10. 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 data set (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. 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. Once feature-extraction stage of training was complete, fine-tuning was implemented to further improve the network performance.

A Mean Square Error (MSE) loss function 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 [9]. 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 [11] 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 Mean Absolute Error (MAE) loss and validation metrics were utilized for optimization Checkpointing and early stopping criteria measures. Figures 11, 12, 13 and 14 demonstrate 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. Additionally, the MAE of both the training and validation data was greatly improved with fine-tuning with little evidence of overfitting.

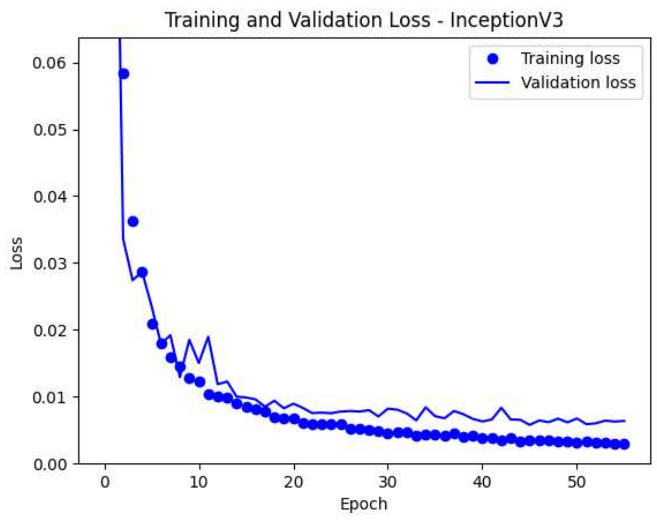


Figure 11. Example InceptionV3 Feature Extraction loss function history.

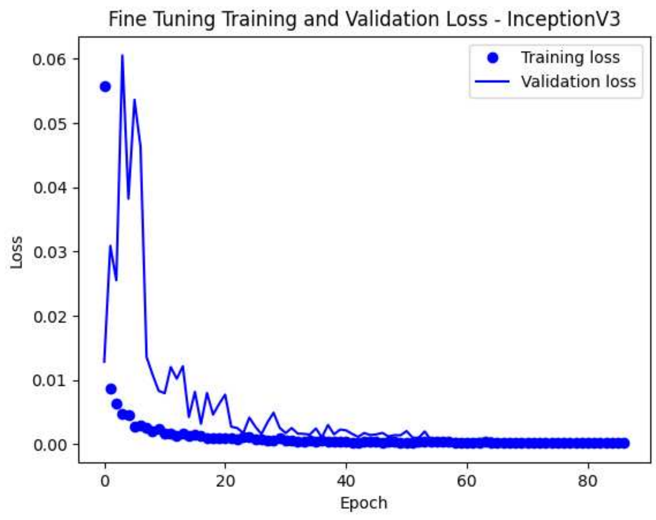


Figure 12. Example InceptionV3 Fine Tuning loss function history.

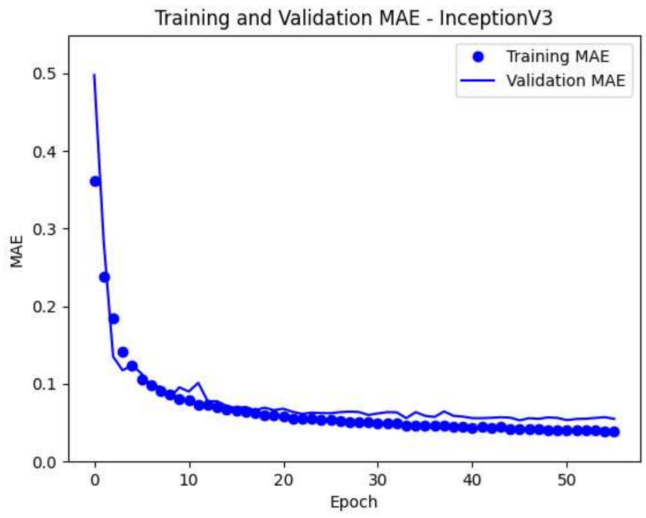


Figure 13. Example InceptionV3 Feature Extraction validation history.

Despite the successful neural network training behavior exhibited above, the researchers encountered several issues in acquiring consistent data sets 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 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 [12,13]. An example of this process can be found in Figure 15.

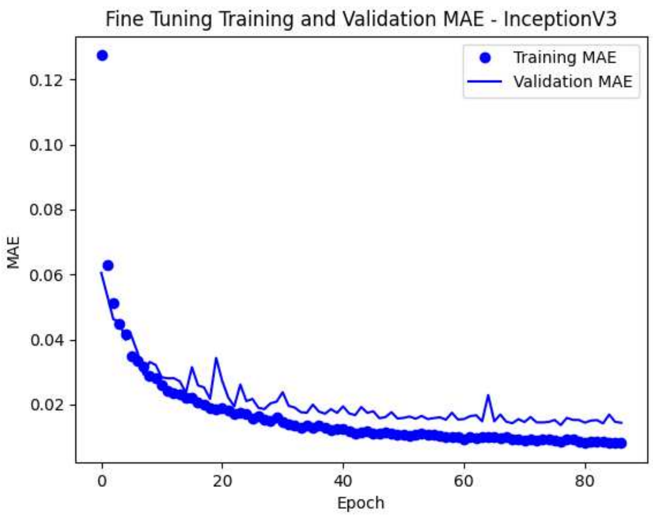


Figure 14. Example InceptionV3 Fine Tuning loss validation history.

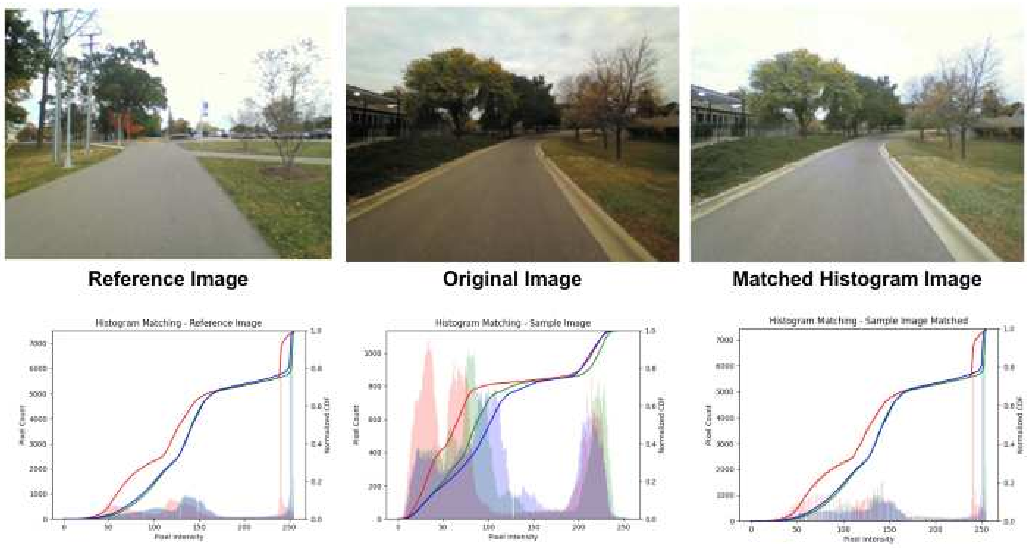


Figure 15. Histogram pre-processing technique used to balance images obtained from the front vehicle camera.

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

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

2. Calculate the cumulative distribution function (CDF) for both the input image and the reference image.

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

3. Calculate the transformation T to map the old intensity values to 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 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 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 data sets was 80% train and validation and 20% test. Additional data was collected on the application routes. This data was not used as part of the training and validation process; however, it was used as truly "unseen" data to evaluate the effectiveness of steering prediction. A summary of the size the training, valdation, test and application data sets is shown in Table 1.

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

Train	Validation	Test	Application
23,623	5,906	7,383	7,692

The training, validation and testing performance of the InceptionV3 based network are displayed in Figure 16, Figure 17 and Figure 18. Similar diagrams are provided for VGG16 and VGG19 based models in Appendix A and B.

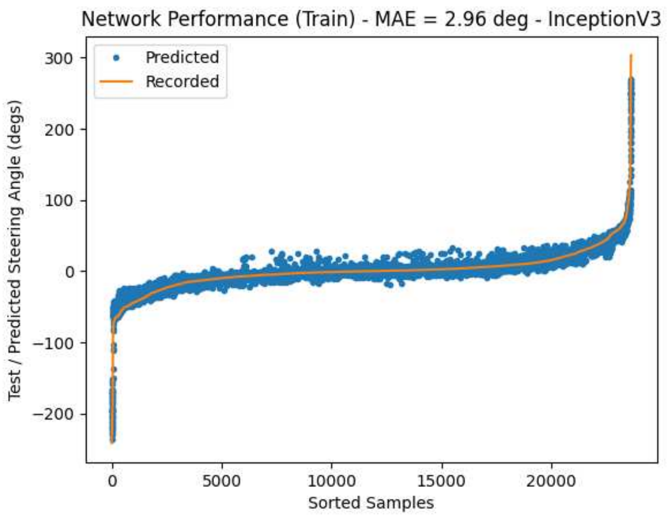


Figure 16. InceptionV3 based neural network train performance.

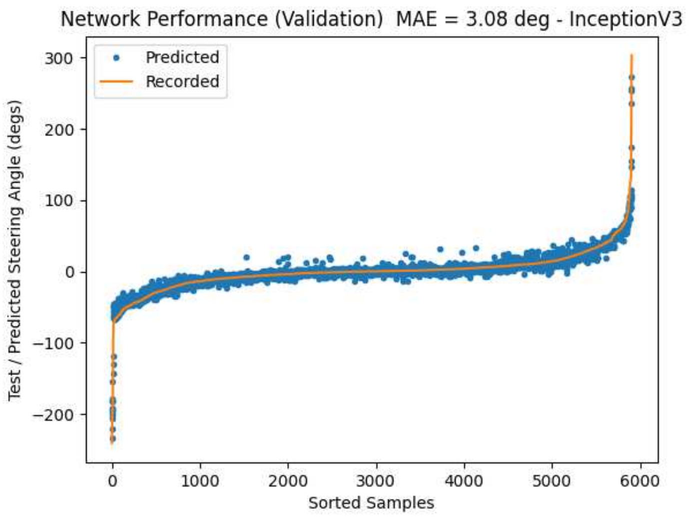


Figure 17. InceptionV3 based neural network validation performance.

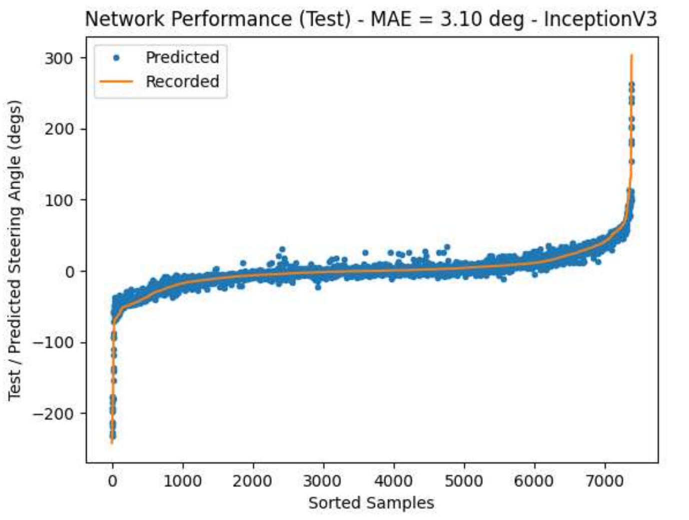


Figure 18. InceptionV3 based neural network test performance.

The training results show that the InceptionV3 model did well in predicting vehicle steer angle for the training, validation and testing data sets. The VGG16 and VGG19 models provided similar performance as the InceptionV3 models, but some important trends that 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 demonstrates slightly higher error for mid-range steering wheel angle (SWA) responses $|SWA| \leq 40°$, but lower error in predicting the steering responses at the larger steering inputs $|SWA| > 40°$. The behavior results in a lower overall test MAE of 3.10° for the InceptionV3 based model. A summary of the model training, validation and test MAE results for all three networks is displayed in Table 2.

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

	Inceptionv3	VGG16	VGG19
Train	2.96	3.82	4.33
Validation	3.10	3.81	4.29
Test	3.10	3.88	4.41

5.2. Model Application Results - Self-Drive with Inference

The final step in developing and testing a Deep Steer solution is to test self-driving behavior with inferencing. To accomplish this, a 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. (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 19 displays the data flow for the Deep Steer testing on the ACTor vehicle. The webcam was integrated with a ROS camera publishing node that generated forward road images at approximately 30 Hz. In a effort to 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 Hz - 10 Hz. Each camera image was then processed with a histogram matching algorithm with the reference image in Figure 15. 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.

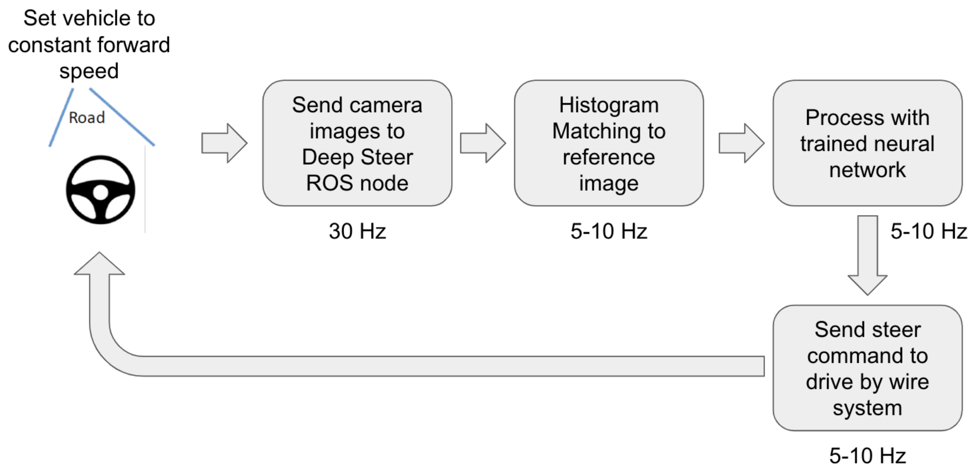

Set vehicle to constant forward speed
Road
Send camera images to Deep Steer ROS node
30 Hz
Histogram Matching to reference image
5-10 Hz
Process with trained neural network
5-10 Hz
Send steer command to drive by wire system
5-10 Hz

Figure 19. Neural network steering angle prediction.

The trained neural networks were next exercised on the application route data set to ensure that Deep Steer could predict steering request with reasonable error characteristics to support fully autonomous physically testing. For this analysis, the vehicle was driven manually along the application route while recording the front webcam image data and the associate driver steering input. This data was then used to evaluate the accuracy of the trained neural networks and the results of the steering angle prediction model are displayed in Figure 20. The InceptionV3 model demonstrated a 4.25° MAE while the VGG16 and VGG19 models provided MAE's of 5.05° and 5.17°, respectively. It can be noted that is a great improvement from the original work completed by Timmis, Paul and Chung, where the predicted steering angle yielded 15.2° of error on average for similar testing [4]. Again, it is important to note that images collected from this route were not used in model training and validation and complete "unseen" to the neural network.

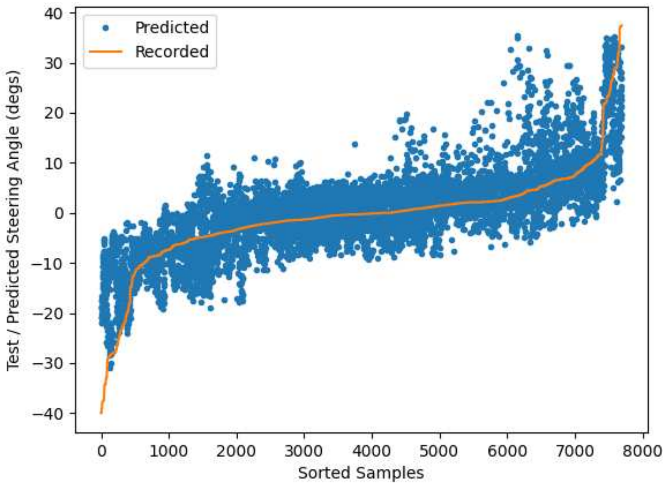


Figure 20. InceptionV3 based neural network performance on unseen, application route data.

Given the overall steering prediction demonstrated in the Figure 20, the ACTor vehicle was tested in a fully autonomous mode, using the Deep Steer algorithm to steer the vehicle. The DataSpeed longitudinal controller was leverage to ensure constant vehicle speed of approximately 5 mi/hr. 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 performance on the application route [14]. 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 exhibit good road following behavior, but were definitely influence by the presence of the interface and some driveways. Overall the InceptionV3 based model provides the most robust road following behavior with minimal influence from intersections and driveways.

One of the goals of this research was to delivery 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 $50 USD. In order to minimum 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 jerking response of the vehicle, while tests above 10 Hz led to lagging in the steering predictions due to image queue buffering the ROS network. From this investigation, it can be concluded that driving the vehicle at 5 Hz provided 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, a application of Recurrent Neural Networks were studied. To promote consistent comparisons, the same top structure was used; however additional layers were added to support RNN's. 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 21. Please note that leveraged

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 August 2023 doi:10.20944/preprints202308.0258.v1

recurrent neural network accepts images of size (320 x 240) and the output of the full neural network is the steering wheel angle prediction in radians.

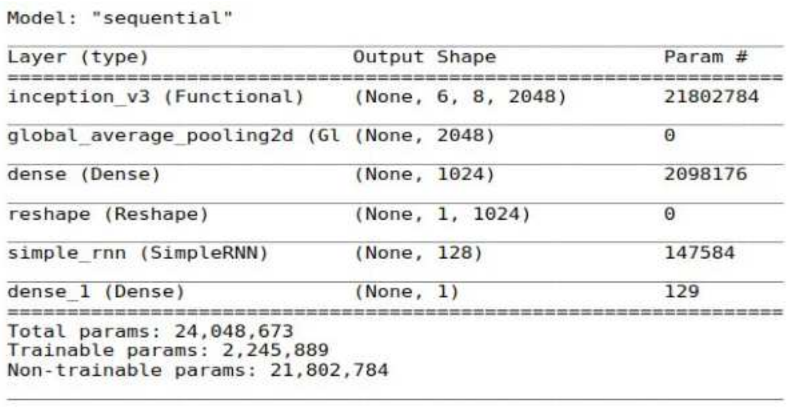

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_v3 (Functional)    (None, 6, 8, 2048)        21802784
_________________________________________________________________
global_average_pooling2d (Gl (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 1024)              2098176
_________________________________________________________________
reshape (Reshape)            (None, 1, 1024)           0
_________________________________________________________________
simple_rnn (SimpleRNN)       (None, 128)               147584
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
=================================================================
Total params: 24,048,673
Trainable params: 2,245,889
Non-trainable params: 21,802,784
_________________________________________________________________
```

Figure 21. RNN model architecture.

For this analysis, the same application route data set 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 22. The RNN model demonstrated a 3.82° MAE. It can be noted that is an improvement from the other neural networks studies in this paper. A summary of the model training, validation and test MAE results for the network is displayed in Table 3. Controlling the data feed of new RNN based model was slightly altered by adding a skip and append function that takes a dataframe and start index as input. It returns a dataframe starting at the specified index, including only every n images. The batch size is number of images fed into the neural network at a time, during training, which was optimized. This is significant when working with a RNN layer as the model optimizes with the memory of the previous image. It is important to mention that the RNN model provides the 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 [15].

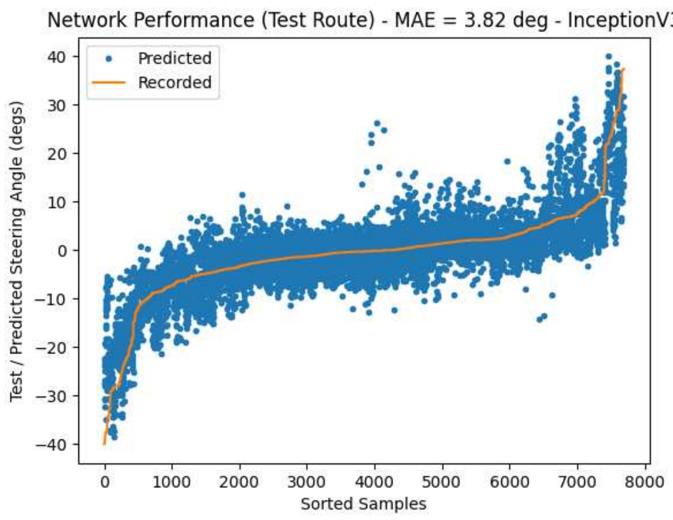


Figure 22. RNN performance on unseen, application route data.

Table 3. Training, validation, test and test route MAE (deg).

	Recurrent Neural Network
Train	2.51
Validation	2.64
Test	2.67
Test Route	3.82

6. Conclusion

The vehicle was tested in real-time on the application route shown in red in Figure 5 and evaluated subjectively. The InceptionV3 based models 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. 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 show improved prediction error in all phases of testing and training as well as demonstrating improved road following behavior during subjective testing.

Author Contributions: Conceptualization, G.D., C.-J.C., N.P.; methodology, G.D., C.-J.C., N.P.; software, G.D., A.R.; validation, G.D., J.D.; formal analysis, G.D.; investigation, G.D.; resources, G.D., A.R.; data curation, G.D., A.D.; writing—original draft preparation, G.D., A.R.; writing—review and editing, G.D., J.D., C.-J.C.; visualization, G.D., A.R., 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 ACTor electric vehicles, sensors, on-board computers, by-wire systems, other parts, and vehicle maintenance cost: US 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 train, validation, test and application route error for the VGG16 based Deep Steer model.

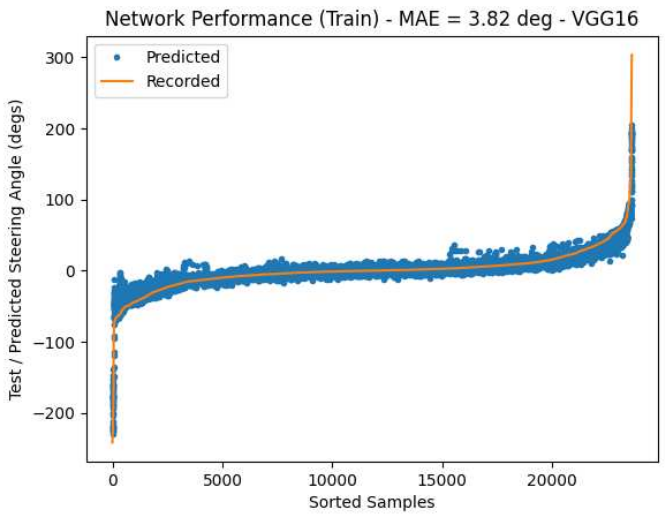


Figure A1. VGG16 based neural network train performance.

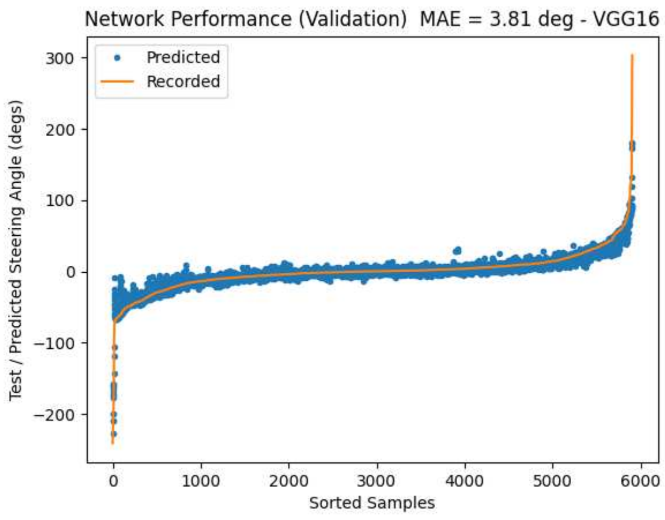


Figure A2. VGG16 based neural network validation performance.

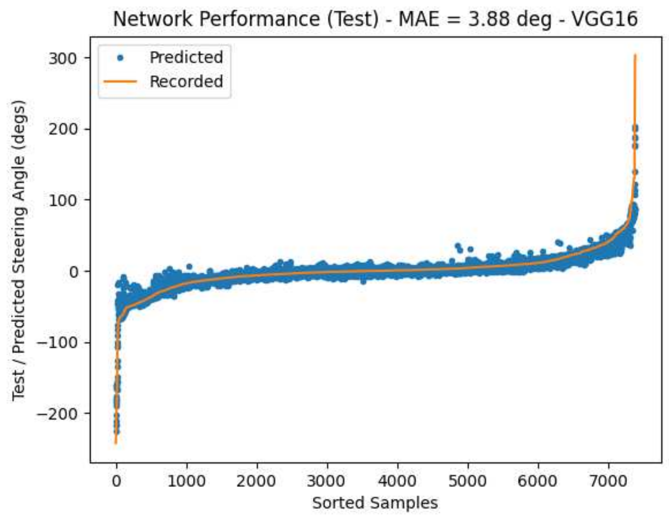


Figure A3. VGG16 based neural network test performance.

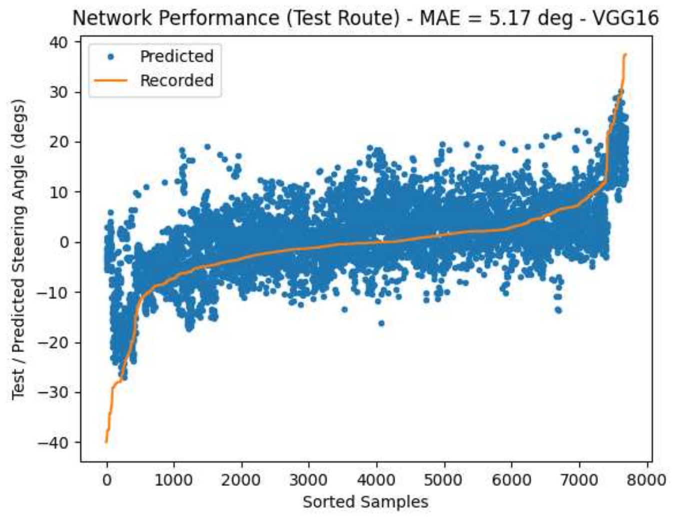


Figure A4. VGG16 based neural network performance on unseen, application route data.

Appendix B

This appendix provides train, validation, test and application route error for the VGG19 based Deep Steer model.

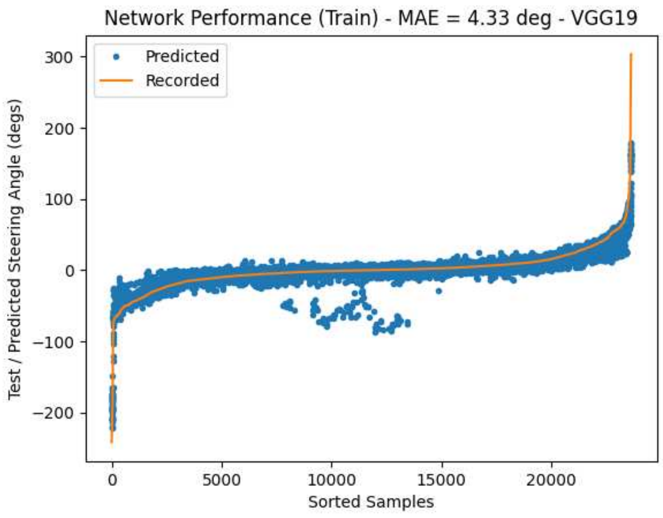


Figure A5. VGG19 based neural network train performance.

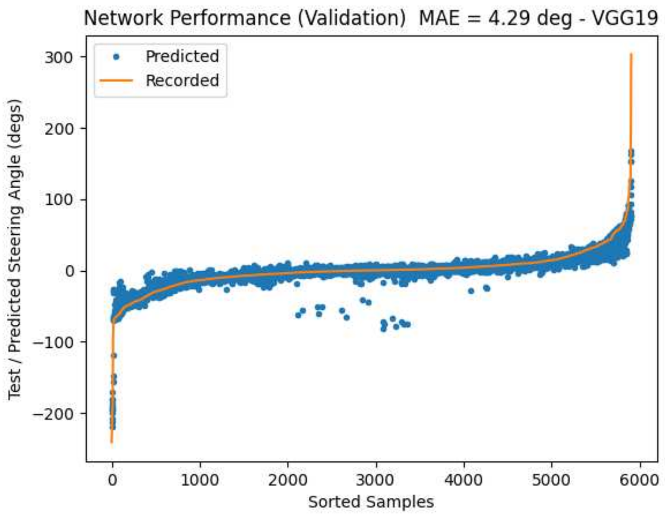


Figure A6. VGG19 based neural network validation performance.

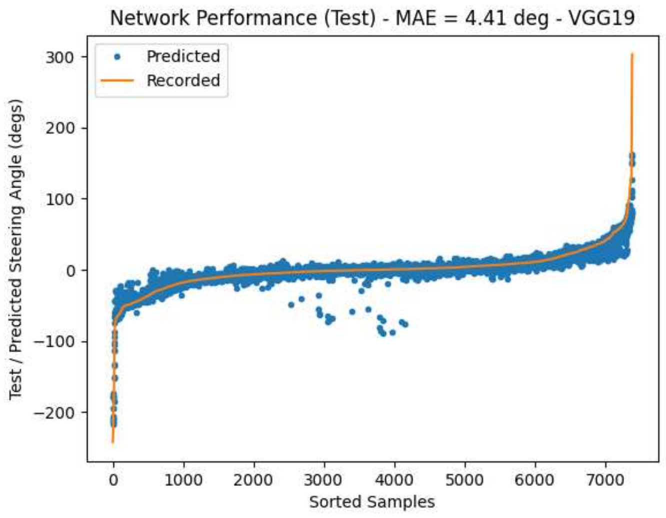


Figure A7. VGG19 based neural network test performance.

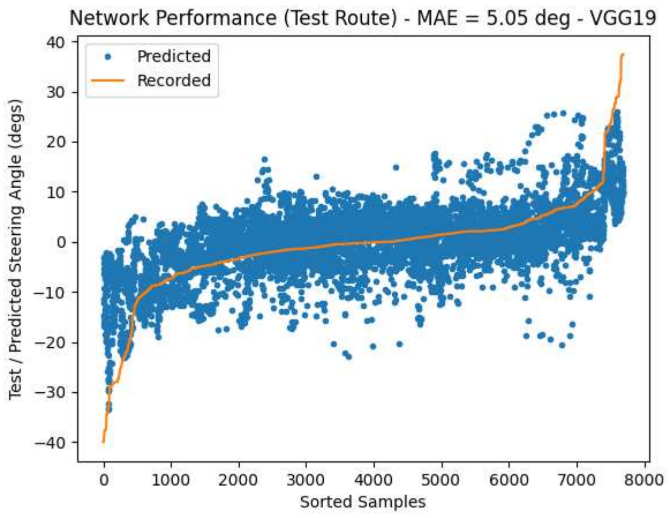


Figure A8. VGG19 based neural network performance on unseen, application route data.

References

1. Pomerleau, D.A. Alvinn: An autonomous land vehicle in a neural network. In Proceedings of the Advances in neural information processing systems, 1988.
2. Yue Wang, D.S.; Teoh, E.K. Lane detection using spline model. In Proceedings of the Pattern Recognition Letters, 2000.
3. Yue Wang, E.K.T.; Shen, D. Lane detection and tracking using b-snake. In Proceedings of the Image and Vision computing, 2004.
4. Ian Timmis, N.P.; Chung, C.J. Teaching vehicles to steer themselves with deep learning. In Proceedings of the In 2021 IEEE In-ternational Conference on Electro Information Technology (EIT), 2021.
5. Quigley, M. ROS: an open-source Robot Operating System. In Proceedings of the ICRA workshop on open source software. Vol. 3, 2009.
6. Laboratory, S.A.I. Robotic operating system. 2023.
7. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. *CoRR* **2015**, *abs/1512.00567*, [1512.00567].
8. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the International Conference on Learning Representations, 2015.

9. Kathuria.; Ayoosh. Intro to Optimization in Deep Learning: Momentum, RMSPROP and Adam. 2020.
10. Chollet, F. Deep learning with Python. In Proceedings of the Simon and Schuster, 2018.
11. Brownlee, J. How to Checkpoint Deep Learning Models in Keras. 2022.
12. Addison, A. Difference between Histogram Equalization and Histogram Matching. 2021.
13. D. Shapira, S.A.; Hel-Or, Y. Multiple histogram matching. In Proceedings of the 2013 IEEE International Conference on Image Processing, 2013. https://doi.org/10.1109/ICIP.2013.673846.
14. Chung, C.J.; DeRose, G. https://youtu.be/aZJaIr5ilos, 2023. [Online; accessed 31-July-2023].
15. Chung, C.J.; Ramsey, A. https://youtu.be/yGqs5YsmQiU, 2023. [Online; accessed 31-July-2023].