DeepSteer: Autonomous Driving through Convolutional and Recurrent Learning

Lawrence Technological University

Mark Kocherovsky, MS Computer Science Candidate; Giuseppe DeRose, PhD, MS Computer Science Candidate; Nicholas Paul, MSCS, Adjunct Faculty; CJ Chung, PhD, Professor

Department of Math & Computer Science, College of Arts and Sciences

Lawrence Tech Arts and Sciences

Research Day, April 22, 2022

INTRODUCTION

Research into autonomous vehicles (AV) – vehicles that are capable of operation without human input – has seen large strides in the past decade. With traditional methods, road lanes and signs are determined using analysis of a camera's view. Though neural networks have been used in experimental AVs since the late 1980's [1], neural networks have become a much more viable technique in the last several years due to advances in algorithm design and computational power [2]. DeepSteer aims to train a neural network to steer an AV based on input from a front-facing camera on the vehicle. Given a picture of the road in front of the AV – our Autonomous Campus Transport 1 (ACTor 1) platform (seen in Figure 1) – DeepSteer attempts to predict the correct angle for the steering wheel. DeepSteer is also significant because our project is for steering on **roads without lane markings**.

DeepSteer was originally made using convolutional neural networks (CNNs) in TensorFlow 1 [3,4] and compared a CNN+pure DNN combination with CNN+DNN+RNN structures. The recurrent neural network (RNN) and its variations introduce temporal history to the neural network learning process, which is useful in sequential data analysis.

We present a new version of DeepSteer, upgraded to TensorFlow 2, using an expanded dataset and more analysis of the various structures (CNN+DNN, CNN+DNN+RNN, CNN+DNN+LSTM, CNN+DNN+GRU) to show that on the whole, the RNN structure combined with manual pruning techniques produces the most accurate model when tested on unseen data.

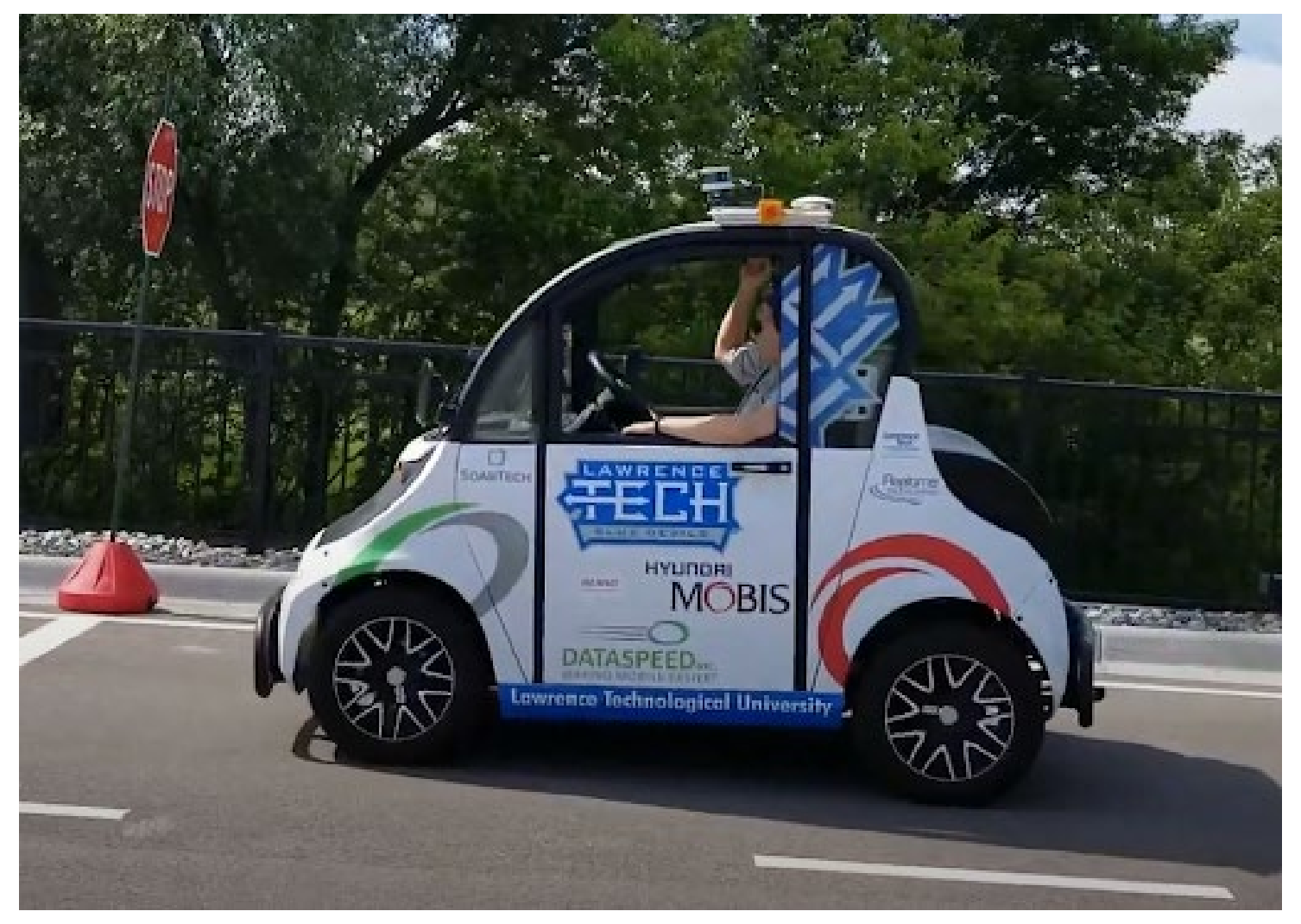


Figure 1: The ACTor1 Platform - N. Paul

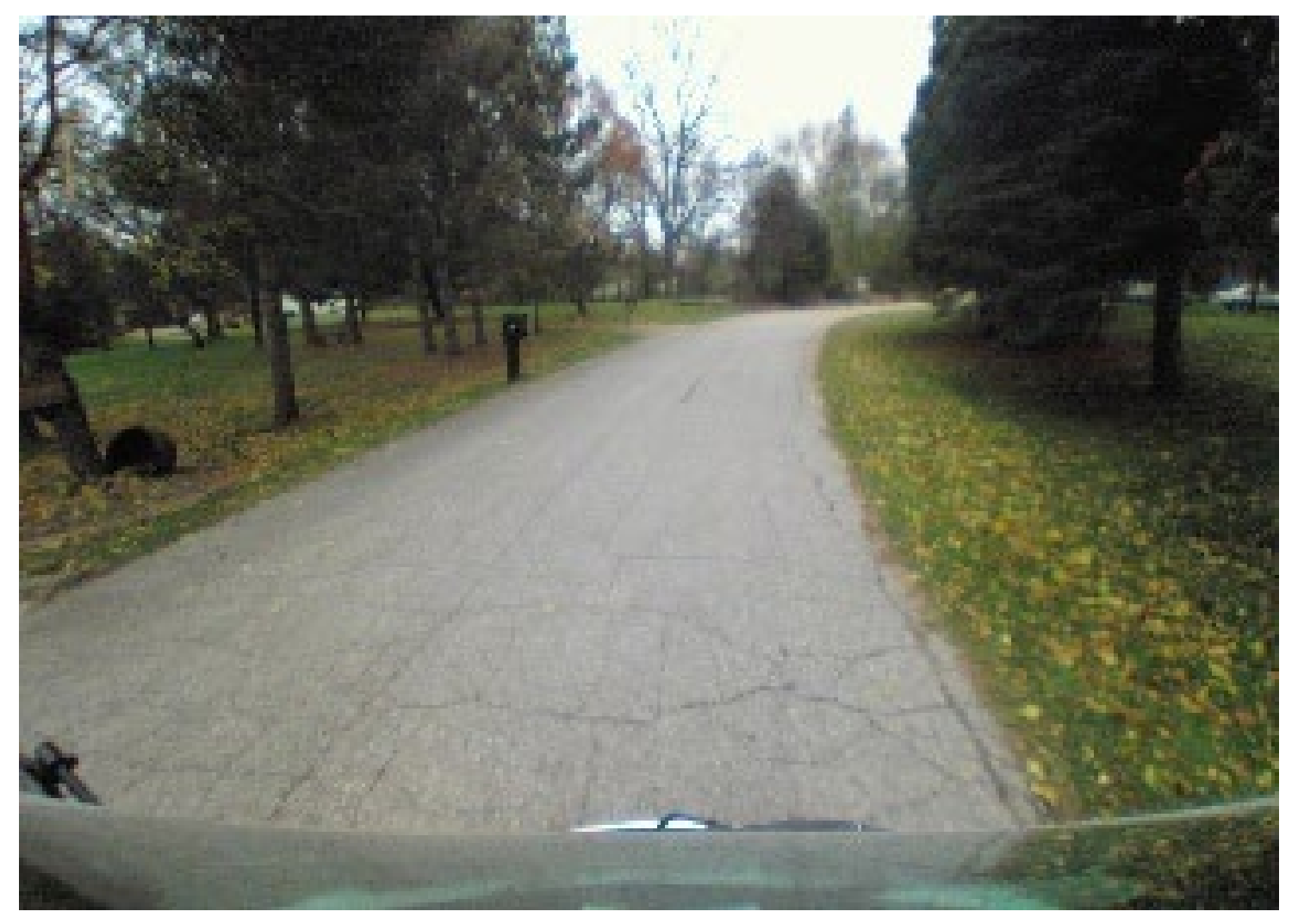


Figure 3: an example of a training data sample. The file's name is 151_-1.1275, indicating that it is the 151st file in that run and the vehicle is turning right at an angle of 1.1275 radians

REFERENCES

[1] D. Pomerleau, ALVINN: An Autonomous Land Vehicle in a Neural Network, in Advances in Neural Information Processing Systems (1988).
[2]. F. Chollet. *Deep Learning with Python*, 1st ed. Apress: Berkeley, CA, 2017; Manning
[3]. N. Paul. Using end-to-end deep recurrent neural network architectures to steer a vehicle (2017).
[4]. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah,M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Tal-war, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Víegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems (2015), software available from tensorflow.org.

Other acknowledgements:

- Joseph Schulte and Justin Dombecki for their work on ACTor vehicles in general.

DESIGN

Neural networks are algorithmic structures designed to perform machine learning, i.e. training a model on datasets in order to make predictions on unseen data. A basic dense neural network (DNN) has a set of processing neurons organized in layers logically connected to the immediate previous and next layers. These connections are represented by values called "weights". The neurons take in input values from the previous layer and perform aggregation operations using the associated weight values. The network "trains" by repeatedly tweaking weights based on error in predictions using a given **training dataset**. Once a model is sufficiently trained, it can be used to **predict** values based on **unseen data.** Convolutional neural networks (CNN), a variation on DNNs, are often used in image recognition tasks. They use feature maps to learn abstract features common to the training set of images. Recurrent neural networks (RNN), and their variations of long short-term memory (LSTM) and gated recurrent units (GRU) allow neural networks to adjust weights not just on the data content, but the temporal sequence of data entry [3,4].

DeepSteer's associated datasets were collected by driving ACTor1 down a set of paths with a program that would save an image of the path in front of it together with the steering wheel angle at that time. During training, the model learns to predict the steering angle from the picture. The structure of each network is as shown in Figure 2. The Time-Series Layer (RNN, LSTM, or GRU) can be reconfigured and even removed using different options in the training program.

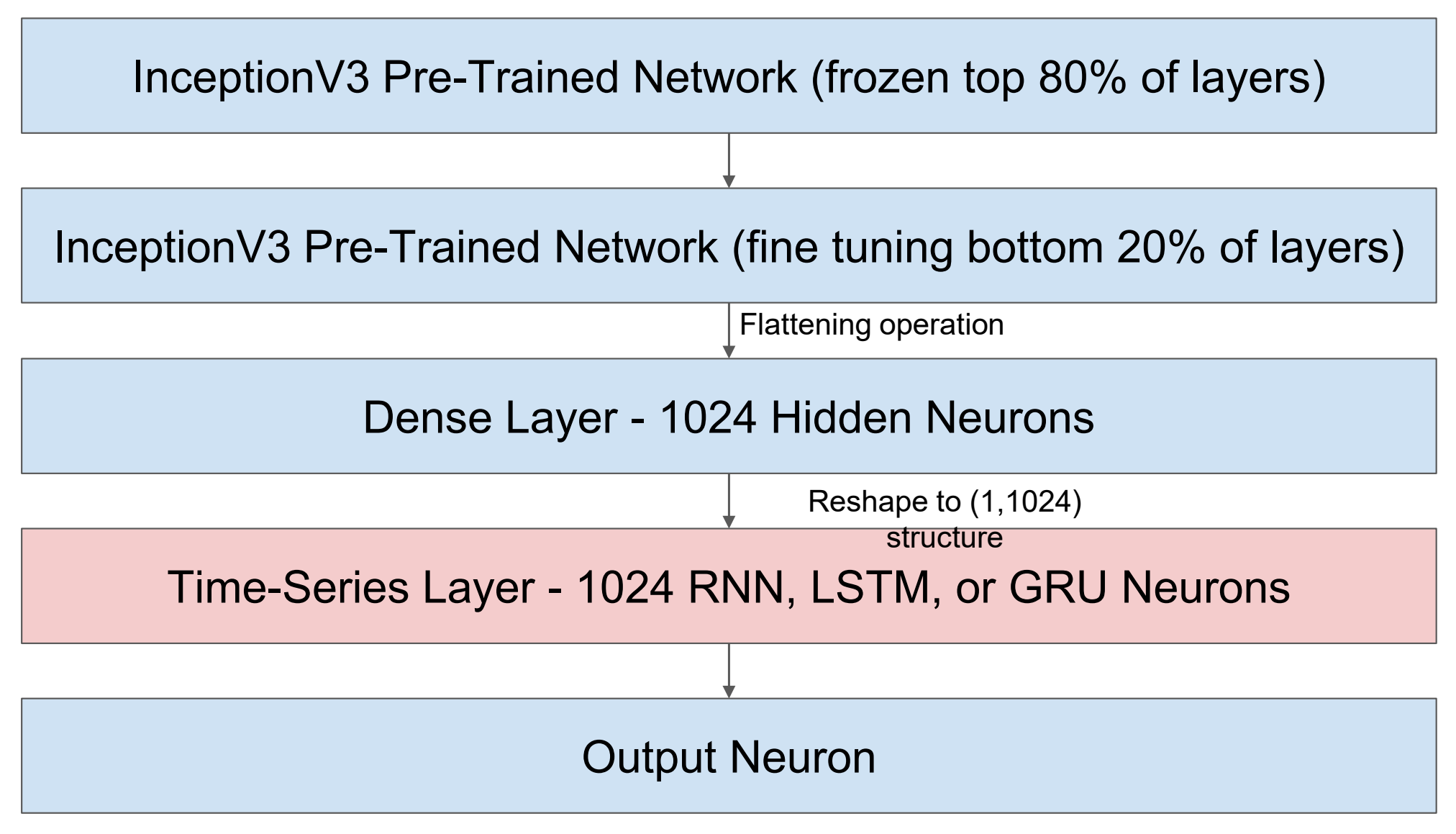

InceptionV3 Pre-Trained Network (frozen top 80% of layers)
InceptionV3 Pre-Trained Network (fine tuning bottom 20% of layers)
Flattening operation
Dense Layer - 1024 Hidden Neurons
Reshape to (1,1024) structure
Time-Series Layer - 1024 RNN, LSTM, or GRU Neurons
Output Neuron

Figure 2: The model uses the InceptionV3 pre-trained model. This allows us to leverage a powerful premade component without having to spend considerable man-hours training a model to recognize basic visual features [4].

Each network structure was measured twice: once with the **full training set**, and once with a **pruned training set**. **Pruning** is a data engineering practice where elements in the set are removed because they might mislead the network during training. We first manually pruned elements because they indicated a low angle when a curve was approaching. Then a number of images were removed at random because the number of low-angle inputs greatly outnumbered the high-angle inputs, introducing **bias** into the program. Figure 3 shows an example of high-angle training data, and figure 4 shows the frequency of low and high-angle turns. However, some data was taken with an **offset,** which may have inadvertently interfered with training. [3,4].

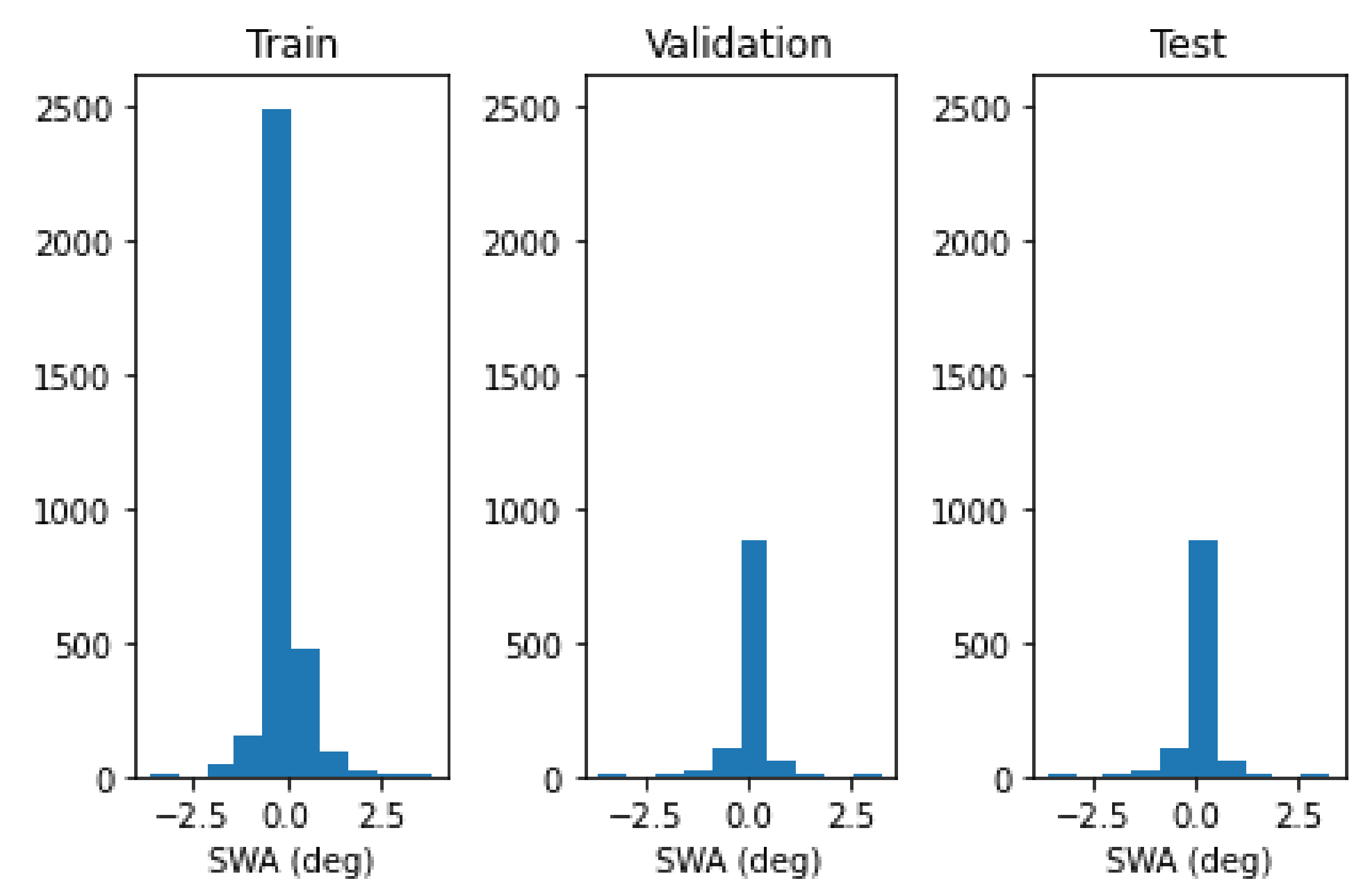


Figure 4: Amount of inputs before pruning; high angle inputs are greatly outnumbered by low-angle inputs

RESULTS

Out of the models trained **without pruning** the data, the GRU had the best performance, with a mean average error (MAE) of 0.2028 radians. The difference in MAE between the best and worst models was 0.022 radians, indicating very close agreement. However, analysis of all four models shows that the model is far less accurate on high-angle turns than on low-angle turns, which indicated evidence of a biased dataset. This analysis is shown in Figure 5. Despite the **offset**, the models performed close enough to conjecture that the offset did not significantly interfere with training, though this would require further study.

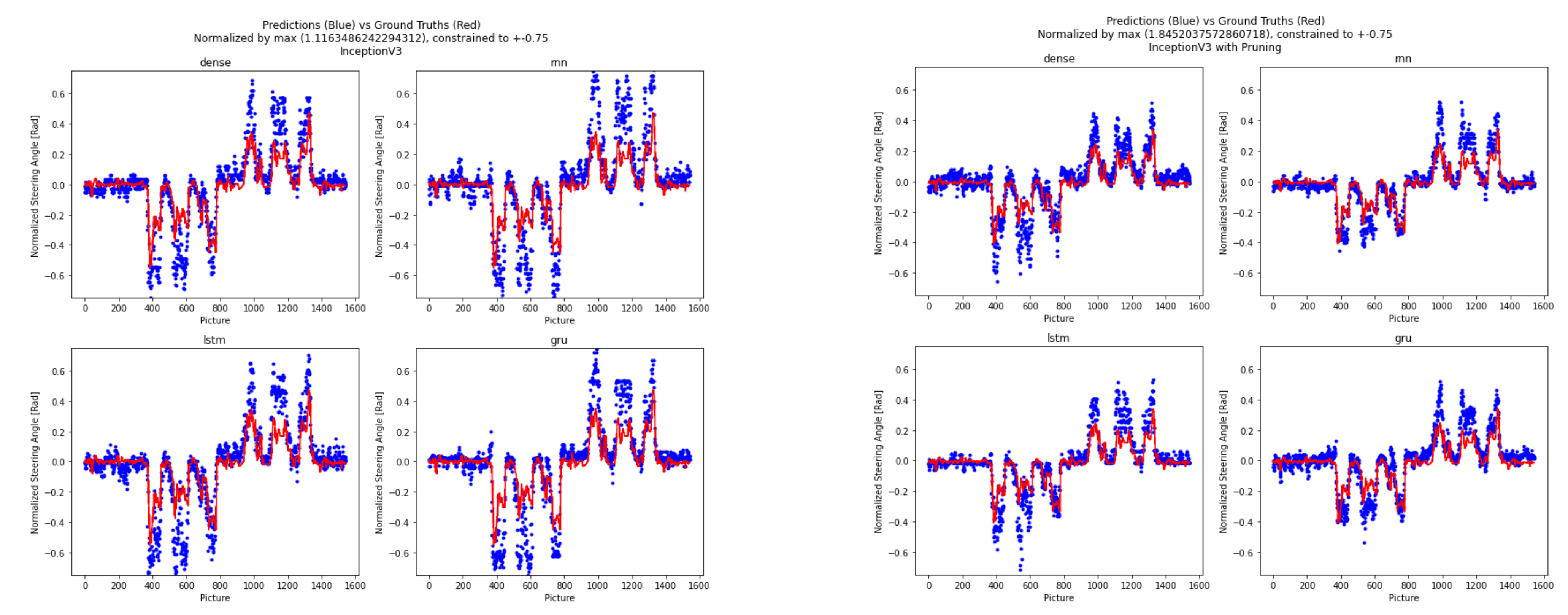
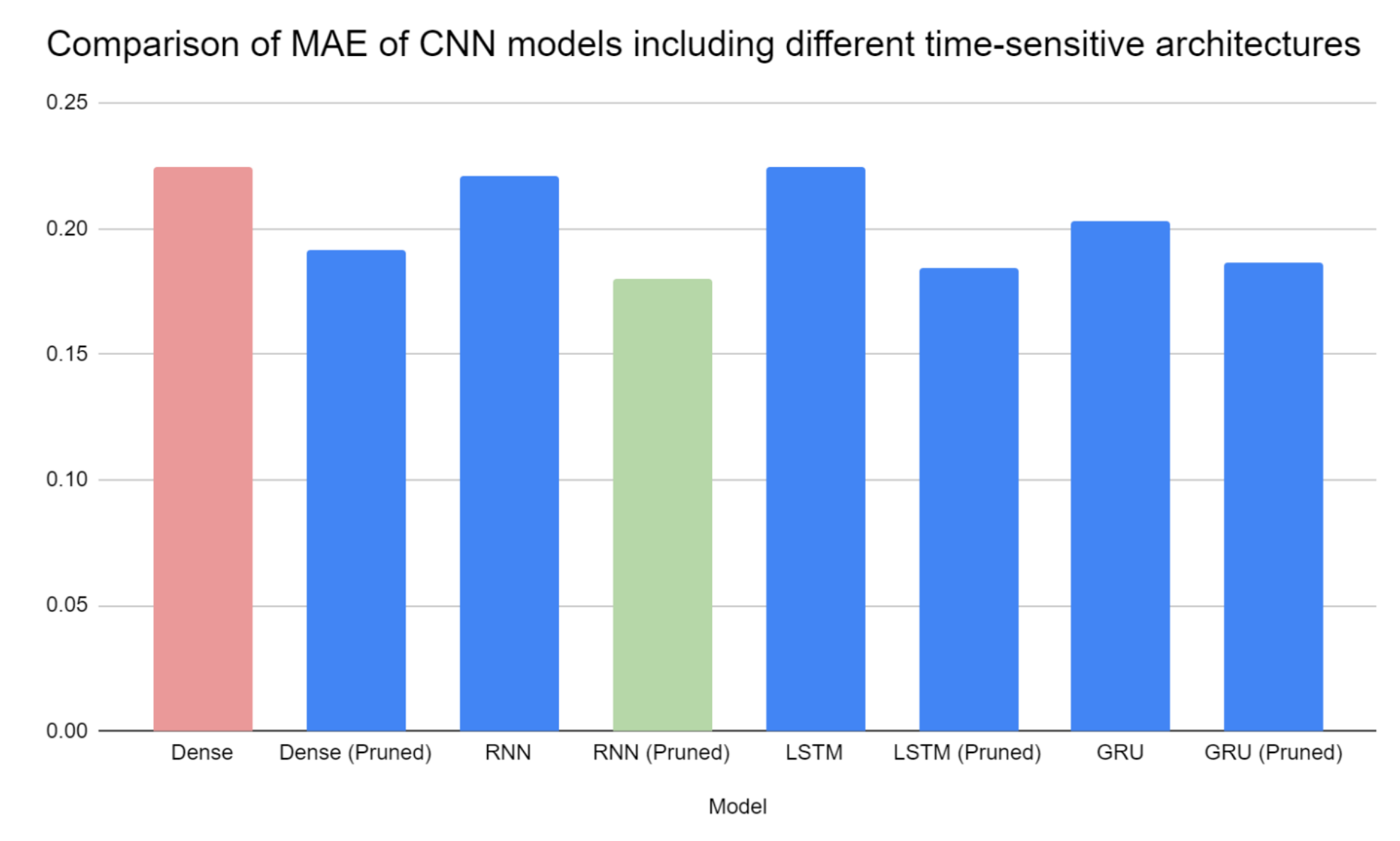


Figure 5: Graphs demonstrating the performance of each model on each element in the testing set. Red lines indicate the correct value associated with each element, while blue dots represent the predicted values for that element. Both sets of results are normalized by their max values and use the same axis, with the most extreme outliers outside the y-axis range for clarity. **Left:** before pruning, even though the models have approximately 0.2 radians of error on average, it still shows incredibly poor performance on high-angle inputs, and better performance on low-angle inputs. **Right:** After pruning, even though predictions of high-angle inputs are worse than on low-angle inputs, the problem is somewhat mitigated as the model is more accurate on those elements.

The best model **after pruning** was the RNN model, with an MAE of 0.1799 radians. Even though the MAE difference between the pruned and non-pruned models is very small, **all pruned models outperformed each non-pruned model.** This becomes even more clear when looking at the predictions themselves: the pruned models appear to have much more accurate predictions than their non-pruned counterparts. Figure 6 shows a comparison of each model's experimental results, while Figure 7 shows an example prediction on a testing image.



	Loss	MAE
Dense	0.1198	0.2248
Dense (Pruned)	0.0829	0.1916
RNN	0.1095	0.2211
RNN (Pruned)	0.0721	0.1799
LSTM	0.1166	0.2243
LSTM (Pruned)	0.0875	0.1846
GRU	0.0996	0.2028
GRU (Pruned)	0.0791	0.1865

Figure 6: Comparison of pruned and non-pruned models by architecture. **Left:** graphical comparison of each model's MAE. The non-pruned Dense (CNN+DNN only) bar is highlighted in red to indicate that it is the worst model overall, and the pruned RNN (CNN+DNN+RNN) model is highlighted in green to indicate that it is the best model overall. **Right:** Numerical comparison of each model's loss and MAE values. The pruned RNN model is highlighted in green to indicate that it is the best model overall, and the non-pruned Dense model is highlighted in red to indicate that it is the worst model overall.

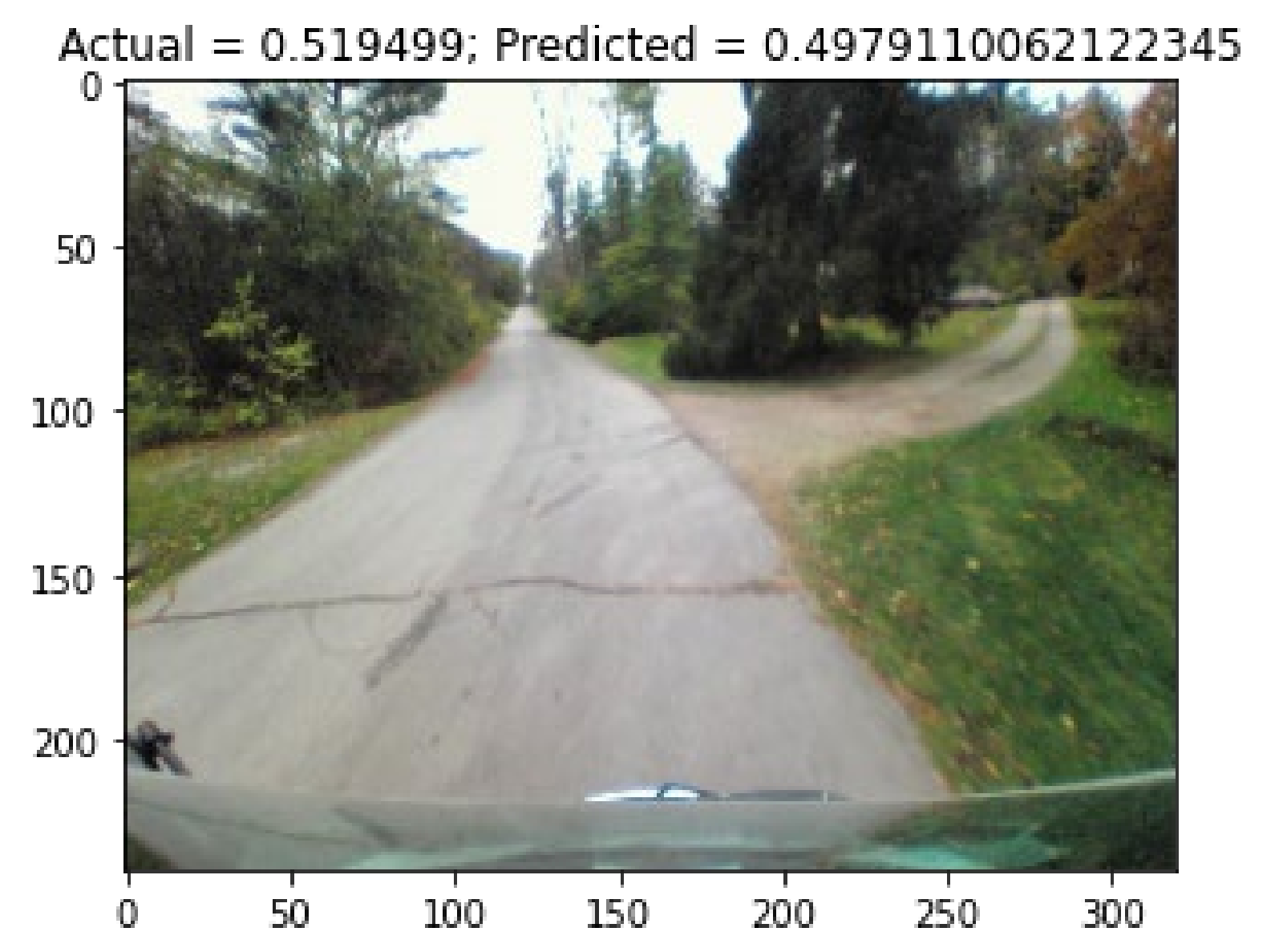


Figure 7: A test dataset sample run through the model. Though the actual turning angle was about 0.52 radians to the left, the model predicts that it was turning at 0.50 radians. This is a very close result, indicating that the model works as designed.

DISCUSSION

DeepSteer proves that Recurrent Neural Networks can be used to direct autonomous vehicles. RNNs can predict the steering wheel angle of a car based on the view of the road in front of it to within 0.2 radians of the correct value on average. However, more work is needed to improve the prediction accuracy. First, more data, especially at higher turning angles, needs to be collected. The new data should also include different environmental conditions, such as precipitation, traffic, and different road markings. The data might be able to be further pruned and organized to improve time-series sequentialism. Finally, the model architecture could be further refined within the RNN framework to better control the training process.