Steering Wheel Angle Prediction using Image Recognition

Sunish Vadakkeveetil
svadakkeveetil@oakland.edu

Dhavan Raveendranath
dhavanraveendra@oakland.edu

Abstract - **Accurate steering wheel angle prediction is a critical element for safe navigation in autonomous vehicles, directly impacting stability and control on the road. Incorrect steering angle estimation can lead to dangerous consequences, as even minor errors or delays in steering response could result in accidents. As autonomous vehicle technology continues to evolve, enhancing the reliability and accuracy of steering angle predictions is increasingly essential, especially given the dynamic and unpredictable nature of real-world driving environments. This project seeks to address this problem by using image recognition and neural networks models to improve steering angle prediction accuracy through data-driven analysis. Specifically, we will leverage the Udacity Steering Wheel Angle Challenge dataset, which includes real-world images and actual steering angles, providing a strong foundation for modeling steering behavior. Building on prior research that utilizes Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks for angle prediction, we aim to replicate and extend these existing approaches to establish a benchmark. Our primary objective is to implement transfer learning techniques to further refine prediction accuracy, evaluating whether pre-trained models can better generalize to unseen driving conditions. By doing so, we hope to gain insights into how transfer learning, could enhance prediction outcomes relative to the results achieved with CNN and LSTM models alone. The evaluation phase will involve dividing our dataset into training and validation sets, enabling a thorough quantitative assessment of model performance on unseen data. We will use Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) as our primary metrics, allowing us to quantify the difference between predicted and actual steering angles. These metrics will help determine the extent to which our approach improves upon previous research, as lower MSE and RMSE values will indicate higher accuracy. Furthermore, we expect to produce visual plots showing the alignment of predicted and actual angles, providing qualitative evidence of model performance. Ultimately, this project aims to contribute to the broader field of autonomous driving by proposing a robust, data-driven method for steering angle prediction that could enhance both safety and performance in real-world applications.**

Keywords— Steering Wheel Angle Prediction, Convolutional Neural Networks, Long Short-Term Memory, Autonomouse Vehicle, Image Recognition.

I. INTRODUCTION

In recent times, autonomous vehicles are a growing area of research. Software is entering the automotive market and is growing on a daily basis. Subscription services and over the air updates which add new features are common. Autonomous vehicles rely on sensors like camera images, RADAR, LiDAR, etc to perceive the environment and generate inputs (steering wheel angle, brake and acceleration throttle) for the vehicle using different modules to drive by itself. To understand where we are in the world of autonomy or the level of autonomy, SAE – Society of Automotive Engineers, an professional organization that generates standards for automotive operations has created six levels of autonomy from L0 to L5 that goes from no autonomy to full autonomy as shown in figure below. As the level of autonomy increases, the human operations in the vehicle reduces and the supporting features required for driving increases. These feature developments are grouped into what is called "ADAS – Automated driver assist features". These ranges from a simple cruise control which controls speed of the vehicle to a set speed of the driver, adaptive cruise control which includes front camera to control speed as well as maintaining safe distance from the vehicle in the front to the recent state of the features like Blue Cruise from Ford and Super Cruise from GM which now controls the trajectory of the vehicle too.

The trajectory of the vehicle is defined by the steering mechanism, where you rotate the steering wheel which turns the wheels to keep you in the desired trajectory. The steering wheel is adjusted based on the number of factors, the curvature of the road, route map, obstacle avoidance maneuvers because of an obstacle in front of the vehicle or lane centering. Precise estimation of the steering wheel angle based on the environment is important for safe operation of the autonomous vehicle.

There are numerous ways to estimate the steering wheel angle based on sensor data from the vehicle ranging from using lane markings from lane detection to processing LiDAR and RADAR data to detect obstacles or access the surroundings of the vehicle. A common way that is widely used is to estimate the steering wheel angle using image data from the camera sensor of the vehicle to train a deep learning algorithm. To assist this development, Udacitiy's Self Driving Nano degree community has released a self driving challenge that

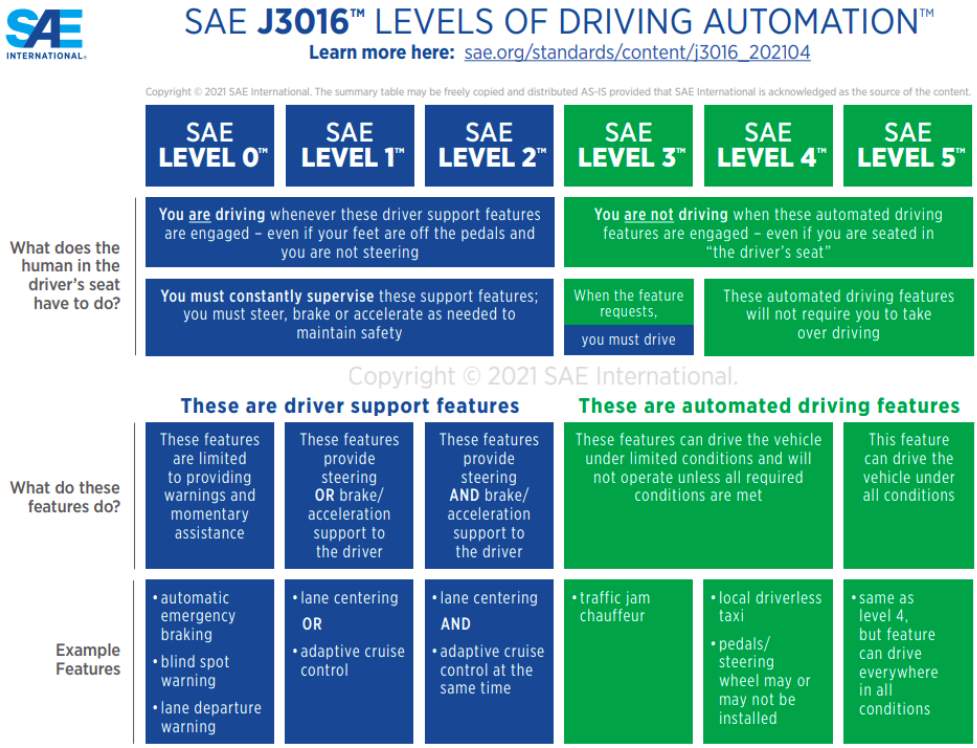

Fig. 1: Levels of Autonomy defined by SAE [1]

encourages developers to come up with unique deep learning based algorithms to predict the steering wheel angle from camera images. This projects, aims at investigating the approach of this challenge and the deep learning models that are developed as part of the challenge, by using the data from the community to train different deep learning models and assessing the performance of the models in a ROS environment to play back the sensor data and estimate the steering wheel angle and play it back to the ROS environment.

II. BACKGROUND

Steering input is one of the critical inputs to safely maneuver an autonomous vehicle. Hence, accurate estimation of the steering angle in real time is important for the safety of autonomous driving. There are 2 main approaches for predicting the steering input, 1 - based on computer vision where the euclidean distance [6, 7] is estimated based on image data from camera sensors to obtain the desired path of the vehicle. This relies mostly on lane marking and road edges to estimate the curvature of the road based on computer vision approaches which is often hard to read especially at lower light or wet conditions when the camera images are distorted, darker or blurred [6]. The 2nd approach is using regression based or NN based models and using camera images to train the models. ALVINN [3] (Automated land vehicle in NN) used a simple CNN based model to estimate the vehicle navigation demonstrating the power of NN for vehicle maneuvering [7]. NVIDIA [2] later developed models which included a combination of CNN and multiple dense layers and used data augmentation to include the effect of image distortion, low light, etc during training. Udacity's self-driving Nano degree team organized a challenge to encourage development of NN based models for steering wheel estimation. Various teams participated and the NN models ranged from simplified models to more complex architectures which included multiple dense layers and LSTM sandwiching the CNN and dense layers [4]. The best predictions had used a combination of LSTM and RNN to predict not only the steering wheel angle but also the vehicle speed and steering torque [4, 8]. Even after the challenge ended many researchers have used the dataset to come up with innovative models using ResNet, data augmentation [5, 9] and using the power of V2V (Vehicle to Vehicle communication) to improve the prediction accuracy by 5 times.

III. DATA

The data being used is from the Udacity Self-Driving Car steering wheel prediction challenge dataset. Which is generated by NVIDIAs DAVE-2 System [2]. Specifically, three cameras are mounted behind the windshield of the data-acquisition car. Time-stamped video from the cameras is captured simultaneously with the steering angle applied by the human driver.

Fig. 2: Udacity self-driving car [4,8]

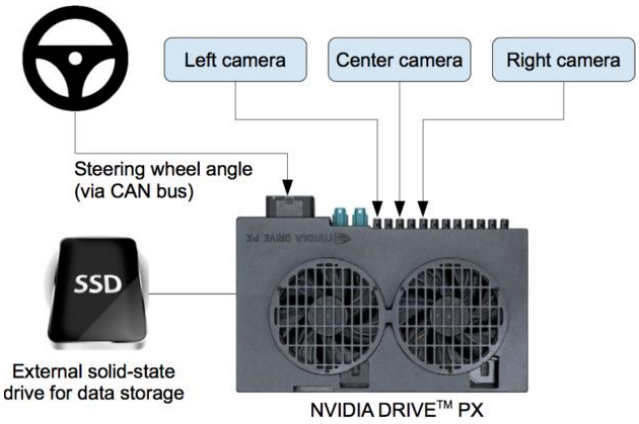

Fig. 3: NVIDIAs DAVE system to collect data [2]

A. *Training Data*

The training data is in the form of 5 ROSBAG image frames and their corresponding vehicle data obtained by tapping into the vehicle's Controller Area Network (CAN) bus.

1. HMB_1: 221 seconds, direct sunlight, many lighting changes. Good turns in beginning, discontinuous shoulder lines, ends in lane merge, divided highway
2. HMB_2: 791 seconds, two lane road, shadows are prevalent, traffic signal (green), very tight turns where center camera can't see much of the road, direct sunlight, fast elevation changes leading to steep gains/losses over summit. Turns into divided highway around 350s, quickly returns to 2 lanes
3. HMB_4: 99 seconds, divided highway segment of return trip over the summit
4. HMB_5: 212 seconds, guardrail and two lane road, shadows in beginning may make training difficult, mostly normalizes towards the end
5. HMB_6: 371 seconds, divided multi-lane highway with a fair amount of traffic

Fig. 4: Windshield mount for the camera

B. *Data Extraction*

The data was provided in the ROSBAG file format. It is a format used by ROS - robot operating system for storing and replaying data for autonomous driving software development. It is not an operating system, but a set of software libraries used for building robot applications. The data that is used for this challenge are provided in a ROSBAG format. The data is played back using the ROS python libraries and the data required for training are extracted. The BAG files provided for training has camera sensor data and CAN information of the vehicle which includes steering, brake and throttle data that are used to drive the vehicle. We had used python with ROS libraries to subscribe to the camera and steering report sensor to extract the images in ".jpg" format and steering information from the CAN signals.

The camera sensor and the steering sensors or CAN bus record the signals at different frequencies, ~50 Hz for camera images and ~100 Hz for the CAN images, while extracting we need to make sure the steering data and the image data is synchronized. The camera images are stored in ".jpg" format and the steering data that goes with the images are stored in an excel format. The label used in this case is the timestamp value.

C. *Preprocessing Training Data*

The image preprocessing pipeline is designed to prepare training data effectively for the steering angle prediction model by transforming raw image data into standardized formats. The process begins by reading image file paths and their corresponding steering angles from input files. Images are resized to a uniform resolution to ensure consistency and are converted into different formats depending on the intended feature extraction. For instance, some images are converted into the HSV (Hue, Saturation, Value) color space, which separates color information (hue) from brightness and intensity (value), making it easier to emphasize changes in lighting and color contrasts. Other images are retained in their original RGB (Red, Green, Blue) format to preserve the full spectrum of color details for comprehensive analysis. Temporal differences between consecutive grayscale images are calculated to capture motion and dynamic changes in the scene, with pixel intensity values normalized to maintain consistency across the dataset.

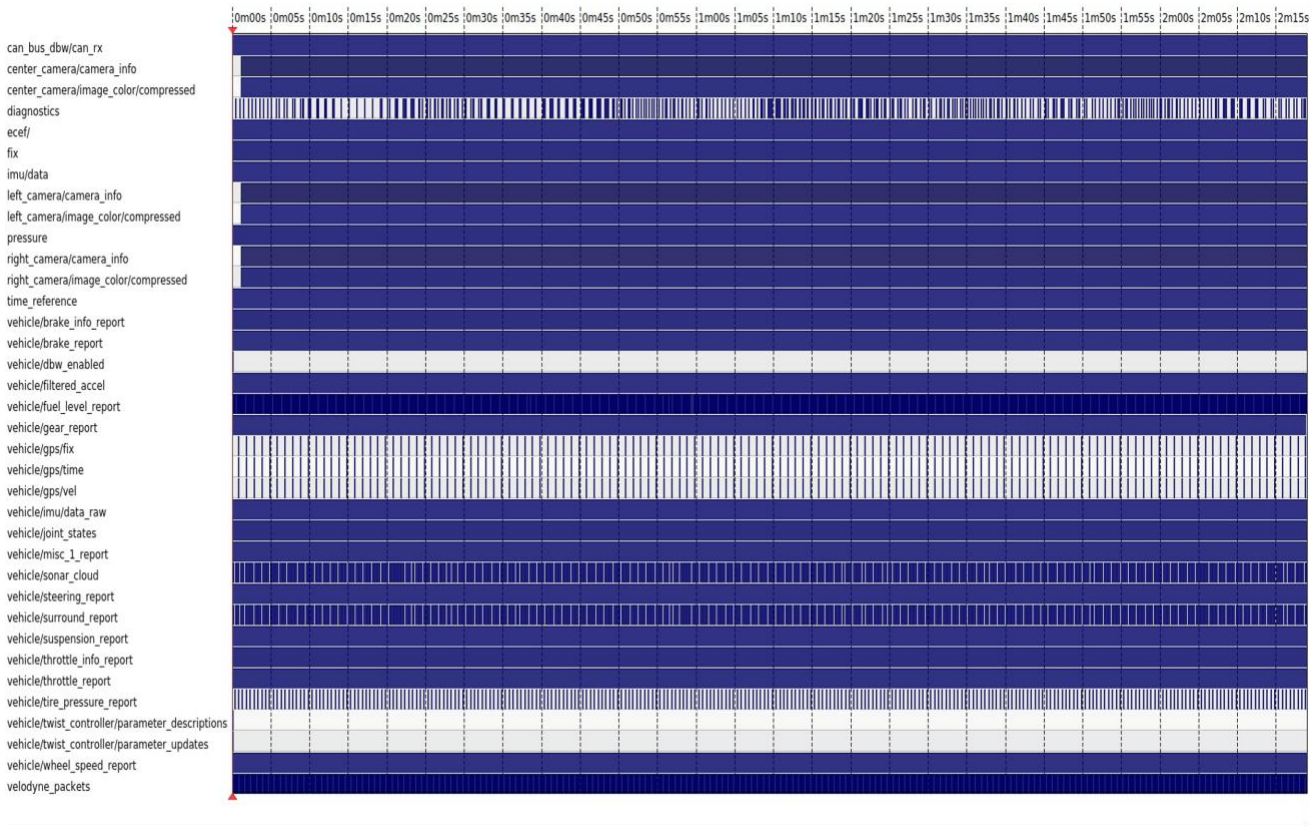

Fig. 5: ROSBAG information of the collected training data

Another variation isolates brightness changes from color-transformed images to highlight shifts in lighting or shadows. The preprocessed data is saved in an efficient .npy format, enabling smooth integration into the training process. This approach ensures the input data is standardized, noise is minimized, and crucial spatial and temporal features are retained, enhancing the model's ability to learn effectively.

D. Test Data

The test data is a set of 5,614 images from the center camera of the vehicle from a driving duration of 280 seconds.

E. Preprocessing Test Data

The image preprocessing pipeline for the test data is designed to enhance the model's ability to predict steering angles by extracting spatiotemporal features from raw image data. First, images are collected from specific directories and resized to a standard resolution of 192x256 pixels. They are then converted to grayscale or HSV formats to capture intensity and color variations. To capture temporal changes between consecutive frames, the pipeline calculates the differences between each pair of consecutive images. These differences are normalized to maintain consistency across the data. For both grayscale and HSV images, the difference is calculated based on the brightness levels, with the HSV format focusing on the brightness channel. The processed data is then organized into multi-channel tensors, which allow the model to analyze the relationships between spatial and temporal features. Additionally, the pipeline includes checks to ensure there are enough images to process, preventing errors due to missing data. Finally, the processed data is saved as structured arrays, making it ready for evaluation by the model. This preprocessing approach effectively prepares the raw image data for tasks like autonomous steering prediction by emphasizing critical changes over time.

IV. MODELS

The project aims to implement a combination of CNN with activation layer and then fully connected layer at the end. It also tried to implement a Long short term memory layer in between the CNN and dense layers to compare the improvement in the performance.

- CNN layer with single fully connected layer/ dense layer: Here we are using multiple convolutional layers with each layer separated by an activation layer. At the end we use a single fully connected layer. For the activation layer we are testing ReLu, Parametric ReLu and Leaky ReLu and Batch Normalization.

Table 1: Different CNN models and its description

Model	CNN Layers	Activation Layer	Fully Connected Layer/ Dense Layer	Issues
Comma Model Prelu	3	PReLu	1 (512 units)	No
Comma model lrelu	3	LReLu	1 (512 units)	No
Comma Model Batch Normalization	3	BN & ReLu	1 (512 units)	No
Comma Model ReLu	2	ReLu	1 (256 units)	No
Comma Model Prelu LSTM	3	PReLu & 1 LSTM	1 (512 units)	No
Comma Model Large	2	ReLu	1 (1024 units)	Yes (Same predictions for test set)
Comma Model Large Dropouts	2	ReLu	1 (1024 units) with dropouts 0.5	No
NVIDIA	5	ReLu	3 (100, 50, 10 units)	Yes (Same predictions for test set)

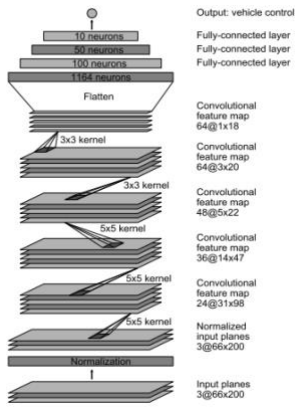

Fig. 6: CNN architecture for the NVIDIA model [2]

- CNN layer with multiple FC/ Dense layers. In this case, we were trying a model which has multiple FC layers separated by ReLu activation layer to see any improvement in the prediction performance.

Due to the limitation of the machine used to train the model, we limited the batch size to be 8 and the number of epochs to be 10. In all the cases, the test set was the same, "HMB_3.bag" but we had varied the training and validation sets. 3 different configurations were tried, 2 of the cases had ⅘ bag files as training set and 1 file to be the validation set. One configuration used 80% of each of the bag file as training set and remaining 20% was considered to be validation set.

Table 2: Configuration of the training and validation set

Configuration	Training Sets	Validation Set	Test set
1	HMB - 1, 2, 3, 5	HMB - 4	HMB_3
2	HMB - 1, 2, 4, 5	HMB - 3	HMB_3
3	80% of HMB 1-5	20% of HMB 1-5	HMB_3

V. RESULTS

In the results section, the results for the best configuration was included for each model that was tested.

Table 3: Results for the different NN models evaluated

Model	MSE	RMSE
Comma Model PRelu	0.00864	0.09295160031
Comma model LReLu	0.03348	0.1829754082
Comma Model Batch Normalization	0.05733	0.2394368393
Comma Model 6	0.00662	0.08136338243
Comma Model Prelu LSTM	0.01642	0.1281405478
Comma Model Prelu Large with Drop out	0.01307	0.1143241007
Rambo	0.00324	0.05692099788

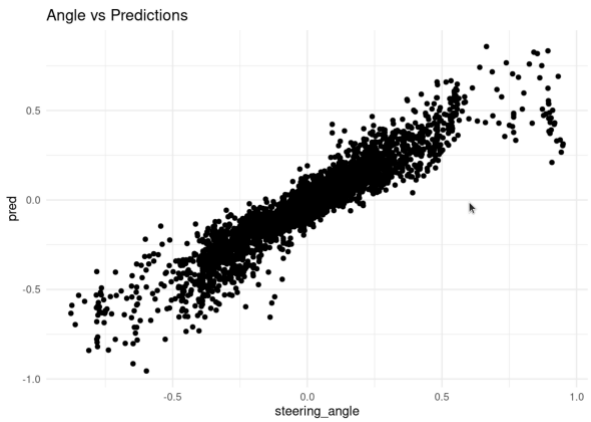

Fig. 7: Comparison of predicted vs actual steering angle

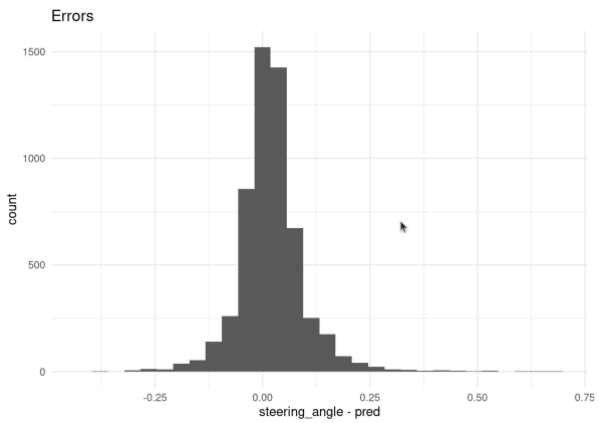

Fig. 8: Steering angle prediction error

Fig. 9: Steering wheel angle prediction (black) vs actual (white)

VI. CONCLUSION

The project focused on using camera data to estimate the steering wheel angle input using CNN architecture with different layers and combining it with fully connected layers and activation functions. The data for training was in a ROSBAG format obtained from a self driving nano degree challenge organized by Udacity. 7 different NN models were looked into by varying single and multi dense layers, ReLu - Normal, Leaky and parametric and batch normalization as activation functions. The hyper parameters and the batch sizes were chosen based on the machine limitation. The data was split into a training and validation set with 3 different configurations. A specific dataset was used as a test set and the different models were evaluated based on MSE and RMSE. Training, validation and test dataset went through image processing and steering angle was extracted from the BAG files at an approximate time step of the image data. Based on the evaluation metrics, the ReLu model with a single dense layer performed the best.

Future steps, we would like to try to out hyper parameter tuning for some the multiple dense layer models (NVIDIA 1-3) to figure out what the issue with the training was because of which we were getting the exact same results make use of the clusters to get higher computation capacity to increase the batch size and also the number of neurons in each layer. In addition, data augmentation can be performed to increase the data size and include different conditions, like minimal light setting, different shadows for the images and also distorting to account for different weather conditions, like snow and rain where the water droplets might distort the camera vision.

REFERENCES

[1] SAE International Recommended Practice, Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles, SAE Standard J3016_202104, Revised April 2021, Issued January 2014.

[2] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba “End to end Learning for Self-Driving Cars”, CoRR, vol. arXiv:1604.07316, 2016.

[3] Pomerleau, Dean A. "Alvinn: An autonomous land vehicle in a neural network." *Advances in neural information processing systems* 1 (1988).Christoph Trattner, David Elsweiler, "Food Recommender Systems Important Contributions, Challenges and Future Research Directions", 2017.

[4] Udacity Open-source Data sets: https://github.com/udacity/self-driving-car

[5] Du, Shuyang, Haoli Guo, and Andrew Simpson. "Self-driving car steering angle prediction based on image recognition." *arXiv preprint arXiv:1912.05440* (2019).

[6] Umamaheswari, V., Amarjyoti, S., Bakshi, T., Singh, A.: Steering angle estimation for autonomous vehicle navigation using hough and euclidean transform. In: IEEE International Conference on Signal Processing Informatics Communication and Energy Systems (SPICES), pp. 1–5 (2015)

[7] Oussama, Aatiq, and Talea Mohamed. "A literature review of steering angle prediction algorithms for self-driving cars." *Advanced Intelligent Systems for Sustainable Development (AI2SD'2019) Volume 4-Advanced Intelligent Systems for Applied Computing Sciences* (2020): 30-38.

[8] https://medium.com/udacity/teaching-a-machine-to-steer-a-car-d73217f2492c

[9] https://github.com/udacity/camera-mount