Steering Angle prediction in Autonomous Vehicles

Amulya Gunturu[1]

[1]Central Michigan University, Department of Computer Science, Mt. Pleasant, MI, USA
[1]guntu1a@cmich.edu

ABSTRACT

Deep Learning has been a trending topic, and its evolution has had a significant impact on the automotive industry. Driving a car through traffic is a challenging task, and automating that involves very complex models built, as the decision of steering and human life's are dependent on it. Different Deep Learning architectures are applied to predict the steering angle in the Autonomous Vehicle (AVs). A survey was conducted on the researches that have been implemented in the steering angle prediction. It is found that most researches depends on Convolution Neural Network (CNN) over other Deep Learning architectures in predicting the steering angle of autonomous driving vehicles[1]. This project uses Convolution Long Short-term memory Neural Network (ConvLSTM)to train the machine learning model to predict the accurate steering angle and increase the performance by considering the temporal relation between the input images.
The project involves securing the data from the front cameras (Left, Centre, and Right) of the car as input and processing the steering angle prediction. The traditional approach of Convolution Neural Network involves matrix multiplication. This approach only considers the Visual camera frames as input and thus ignoring the temporal relationship between the images[2]. This relationship between the images will improve the efficiency of predicting the steering angle. The ConvLSTM Neural Network combines a CNN (Convolutional Layers) and an LSTM, replacing matrix multiplication with convolution operations at each gate in the LSTM cell[3], which will give more accurate results over the traditional CNN prediction of steering angle.
The regression problem solved with Convolutional Long Short Term Memory approach is an improved version of the traditional Convolutional Neural Networks approach and improves the steering angle prediction accuracy when the data contains noise and less useful information. As the Long Short Term Memory architecture stores the past information and uses that information along with current to predict the steering angle it has improved control accuracy of the steering.

Introduction

There have been many situations where humans are drunk or careless and have caused traffic accidents. However, with the introduction of AV, it is possible to activate the automated mode and drive safely and without error. In addition, AVs will provide greater mobility for the elderly and disabled. The introduction of AV will play a vital role in reducing many road accidents and saving many lives. Machine learning plays a critical role with the ability to shift the task of formulating rules explicitly into designing a system capable of learning the rules. Alternatively, these machine learning models can also be introduced in the aviation industry, where crewless aircraft can be operated.

Controlling a vehicle to drive in a particular lane is a crucial task, and this is done by humans solely based on visual cues. In non-autonomous vehicles, humans act as an end-to-end system. These decisions were made by the brains of humans and were never explicitly formulated. The Neural Networks, an Imitation of human brains for decisions, were developed and evolved into Deep Learning techniques. Deep Learning (DL) has positively impacted AV control, especially steering angle prediction, due to its ability to process unlabeled raw data.[1]

Convolution Neural Network (CNN), a powerful technique, allows analyzing the image input and outputs the steering angle. In the recent times, it is proven that Convolutional Neural Networks (CNN) are beneficial for image processing, and these kinds of neural networks have applications in various fields. For example, they are computer vision, face recognition, scene labeling, action recognition and etc.[4]. The problem with CNN is that it considers input images as frames and cannot consider the temporal relation between the frames, which is a feature to recognize motion. Therefore, this project proposes the usage of ConvLSTM Neural Network to predict the steering angle.

Recurrent Neural Network is one of the types in Artificial Neural Network that considers the temporal relationship between the input images using memory cells. This will aid in learning the changes and dynamics from image to image and constantly attempting to predict the steering angle based on the relationship. When we combine Long-short Term memory, the ability to memorize selective modeling has been possible.

This project uses Convolutional Long-Short Term Memory Recurrent Neural Network for improved performance and accuracy in predicting steering angle using the input images from front cameras (right, center, left). This technique will improve the performance as there is temporal relation between input images, and all the matrix multiplication at the input layer of LSTM

will be replaced with convolution operation.

Methods

Data

The data set used for this project was acquired from Sully Chen's Github repository[5]. It was curated by Sully Chen, who is a Researcher at The National Institutes of Health. With a passion for learning and implementing machine learning, he took up a college project named Tensor Flow Auto Pilot. He did reverse engineering on the signal system for his own car's CANBUS interface to collect the labeled driving data and implemented the acquired data to published paper[6]. Driving data includes a sequence of images from his car's front dash camera and the steering angle at each image while driving.

A typical camera captures 30 frames per second. Since we have 45,406 images from the front dash camera images, the data is captured approximately 25 minutes of video, and it is broken down into each frame image. The data set images, and the respective steering angle in degrees is maintained in the data.txt file. If we see the file for the first 22 temporal frames, the angle is 0.0 degrees, which means that the steering is maintained straight. To reduce the big numbers representing the steering angle, we convert them from degrees to radians.

The whole data, 45,406 images, cut to 3000, is divided into training data, validation data, and test data by considering the temporal relation in the frames. The first 60 percent of data is split into training data, the next 20 percent is split into validation data, and the remaining 20 percent is used as test data. The test data will not be used while training the data, and validation data is used as a signal to tune the hyperparameters and monitor the training.

The data set has each image frame of 256x455 length and width to keep the same aspect ratio; all the input images are resized to 66 x 200. These images are resized using the scipy library and visualized using the matplotlib library.

Development Environment

The Development Environment used for this project implementation is Google colaboratory, or colab for short, and a product from Google Research. Colab is a hosted Jupyter notebook service that requires no setup to use. It is a writeable and executable python code that can access computing resources, including GPUs, free of cost. Google colab is equipped with a 12GB NVIDIA TESLA K80 GPU that supports Tensor Flow[7], Keras[8] stack and all the types of deep learning libraries and 13GB RAM capacity.

Training and Evaluation Protocol

After pre-processing, which includes normalization, data cleaning, resizing of the input data, eight different types of feedforward, deep architectures were built, trained, evaluated, predicted and analyzed. The first architecture consists of 4 hidden layers, each with 64, 128, 64, and 32 nodes using rectified linear unit (RELU) as the activation function. This architecture is the basic CNN model inspired by Wang, P.Y Nguyen[9] and Zhang and Huang[10]. The metrics used to measure performance in this model is mean squared error (loss function) and mean absolute error, and the optimizer used is adam. The number of epochs was calculated by monitoring the validation accuracy. When the validation accuracy starts to settle, i.e., overfitting, the count of the number of epochs will be stopped. The training accuracy and loss, validation accuracy, and loss of the built model are shown in the figure below.

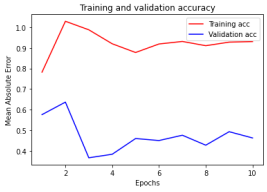

Figure 1. Training and validation Accuracy of 1st model

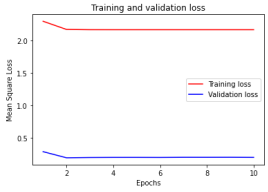

Figure 2. Training and validation loss of 1st model

After observing the patterns of evaluation and prediction, this architecture is over-fitting from the third epoch itself and does not predict accurately as the mean squared error of the predictions by model and labels of the test data is 0.202. The steering angle is NOT predicted as it is expected. The main goal of this project is to build a model that considers the temporal relation between the frames and predict the steering angle accurately, and this can be done using a Convolutional Long Short Term Neural Network (C-LSTM). So, the next second model architecture is C-LSTM neural network. The predictions of the trained neural network with the test labels are as shown in the below figure.

After analyzing the training and predictions that this model has provided after training, we still have the scope for improving the accuracy and reducing the over-fitting. The mean squared error of the second model is 0.3939, which also states that the

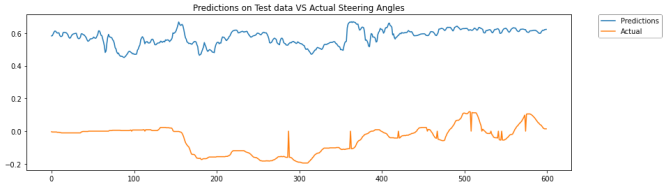

Figure 3. Predictions on Test data VS Test label of model 2

predictions are still not as accurate as it is supposed to be. The data augmentation helps the model improve the performance accuracy of the model and increase the accuracy in predicting the steering angle. The accuracy after augmenting the data on test data is 0.89. The next architecture used to increase the accuracy is a C-LSTM model with a drop out of 20 percent in the dense layer and using the optimizer as Adagrad. This model drops 20 percent of the hidden layers, and the optimizer adagrad helps with the learning rate of 0.001. The training and validation accuracy and training and validation loss of this model is as shown in the Figure 3.

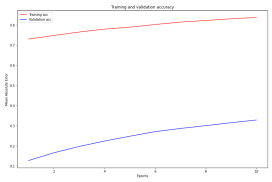

Figure 4. Training and Validation Accuracy of model 4

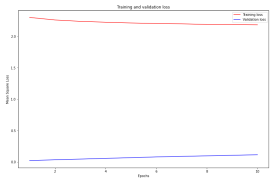

Figure 5. Training and Validation Loss of model 4

The above curves tell us that there is not much improvement in terms of accuracy in prediction as we have replaced the optimizer with Adagrad. The fifth model architecture that I am using to analyze has optimizer Adam, drop out, and L2 regularizer with the C-LSTM model. The accuracy of this model is increased to 82 percent. The visualization of the predictions on test data and test labels are shown in the Figure 4 and 5.

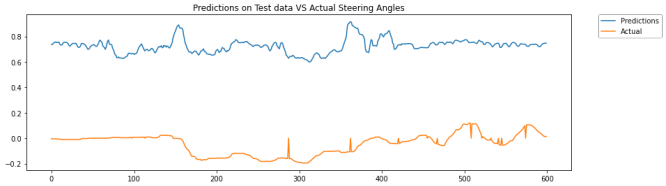

Figure 6. Predictions on Test data VS Test label

As all the above models are only predicting till the accuracy of 0.89. To further increase this accuracy, another architecture used to analyze further is transfer learning using VGG16 and adding LSTM on top. This sixth model uses existing features and weights of the VGG16 and sends the sequential data to the LSTM model. This architecture increases the accuracy of the test data prediction to 95 percent. To further increase the model's accuracy, the next architecture used to analyze is fine-tuning, where the last layer of VGG16 is unfrozen and trained by adding the LSTM layer on the top. However, this model is still not meeting the exact predictions of the test labels. The last model architecture that we used as part of this project implementation is data augmentation on the Feature extraction model, i.e., model 6. This model helped reach the accuracy to almost 100 percent, as shown in the Figure 6.

Table 1 represents the results of all eight architectures on train and test data.

The training accuracy was able to reach 95-98 percent at most of the times in the above architecture. However, the test accuracy was not able to meet this range as the test data that is used for training this model is small, and it is almost around zero radians most of the time. Intuitively, making use of techniques such as data augmentation, drop out, L2 regularization, feature extraction, fine-tuning, and data augmentation with feature extraction to avoid over-fitting of the C-LSTM model helps to increase the training accuracy. The consideration of the temporal relationship between the frames helps in predicting the steering angle as the model starts to learn the dynamic changes between the input frames. The number of predictions made was of the same size as the test data. Different type of techniques to avoid overfitting helps increase accuracy.

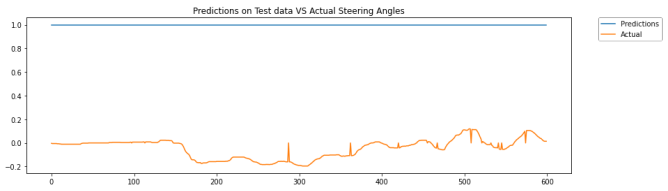

Figure 7. Predictions on Test data VS Test label

Architecture	Accuracy on train data (Mean Absolute Error)	Accuracy on test data (mae)
5-layered CNN with Adam optimizer	0.87	0.46
5-layered CNN+LSTM with Adam optimizer	0.87	0.94
5-layered CNN+LSTM with Adam optimizer (with data augmentation)	1.52	0.94
5-layered CNN+LSTM+ 20 % D.O. with Adagrad optimizer	0.83	0.32
5-Layered CNN + LSTM +20% D.O. + L2 Regularizer with Adam optimizer	0.82	0.77
Freezed VGG16+LSTM with Adam Optimizer	1.3	1.04
VGG16(Fine Tuning) + LSTM with Adam Optimizer	1.3	1.04
Freezed VGG16 + LSTM+20% D.O. with Adam Optimizer	1.3	1.04

Table 1. Performance of eight different Architectures

Discussion

A steering angle prediction using dashcam images as the input is a regression problem. This problem can be addressed by Convolutional Neural Networks (CNN) or Convolutional Long Short Term Memory (C-LSTM), a CNN - RNN (Recurrent Neural Network) architectures. The CNN architecture considers one image at a time as input (can be either sequential or non-sequential) and predicts the steering angle for that image. C-LSTM considers the multiple frames in sequence as input and predicts the steering angle for all the sequences of images in a sequence. This approach of using C-LSTM architecture in predicting the steering angle is different from the standard CNN. It considers both visual and dynamic temporal relations between the frames of input. The end-to-end model starts initially with the passing sequence of input images through CNN and gets a vector representation as an output of the CNN, and this output is fed to LSTM (Long Short Term Memory). LSTM architecture has a recursive structure that behaves as memory, keeps the information about the past, and solves the regression problem by considering the dependencies between current and past images.[11] Figure 8 represents the above-said architecture.

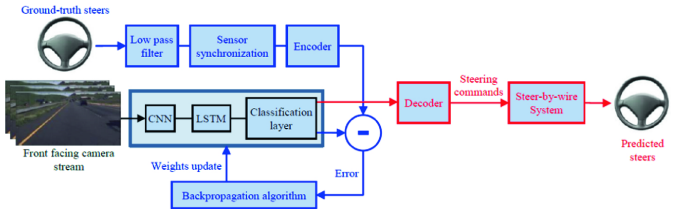

Figure 8. Passing Dash camera images through CNN + LSTM architecture

The C-LSTM architecture usage in predicting the steering angle can benefit when the input image contains noise or less useful information to train the model. For example, we consider a real-time situation where the current image is affected by sunlight or when the vehicle reaches the dead end. In these scenarios, information about the relationship between current and past frames can help predict the steering angle.

In this project, we used a time step of 3 when passing through the LSTM, which means that in 3 seconds, the model creates an LSTM of 90 timesteps cause in one second, it is 30 frames per second. Hence, the LSTM model is many to many as it considers 90 inputs and provides 90 outputs at once. This process involves many loops, which in turn means a longer run time for running the model. The major disadvantage of using the C-LSTM model is that it takes longer and needs more memory. However, once the model's training is completed, we can use the model to predict many other regression problems.

Unlike the traditional approach, i.e., CNN architecture, the C-LSTM approach is more complex and expected to work more efficiently[12]. The baseline model for this problem is finding the mean squared error for the test data and the mean of train data. It turned out as 0.20. So, any model evaluated as part of this project should result in a mean squared error less than 0.20. The CNN architecture has the test root mean square error as 0.19, and the C-LSTM architecture has the test root mean square error as 3.69, which is supposed to be less than 0.20. This approach can be further improved by training the model with extensive data set because the current model is trained on 3000 sequential images. If we consider the typical camera scenario that captures 30 Frames per second, this model is trained with less than 2minutes of sequential input data. Another way to improve the performance of the data is using more giant time steps; even though lesser time steps results in faster computations, the model will improve performance with more big-time steps. However, if we use more giant time steps, one needs more memory and faster GPU s to train the model.

There is a lot of sequential data that is available in the comma.ai[13] one can explore the current build model using the larger dataset and figure out the reason behind the more mean squared error value than expected. The other possible extension would be integrating this model with GPS, which considers the geo-spatial information.

Conclusion

In this project, I presented a new approach in the en-to-end model for steering angle prediction, which is more reliable than the traditional approach of using CNN to improve the control accuracy. The approach involves using C-LSTM architecture, which is expected to yield the lowest mean squared value compared with the existing models. Using more significant memory, faster GPU, and a more extensive dataset can improve the steering angle prediction accuracy of the model. I find that data augmentation and C-LSTM architecture have closer predicted steering angles than the actual steering angles. Using drop out, regularizer, data augmentation techniques, and optimizer ADAM improved the performance of the prediction. However, using Transfer learning methods was not result in improved values than the feature extraction model. I tried implementing the feature extraction, fine-tuning and feature extraction with data augmentation, and all of them resulted in the same mean absolute error value.

References

1. Gidado, U. M. *et al.* A survey on deep learning for steering angle prediction in autonomous vehicles. *IEEE Access* **8**, 163797–163817 (2020).
2. ConvLSTM. Convolutional lon short term memory neural network (2021).
3. Santokhi, J., Daga, P., Sarwar, J., Jordan, A. & Hewage, E. Temporal autoencoder with u-net style skip-connections for frame prediction. *arXiv preprint arXiv:2011.12661* (2020).
4. Smolyakov, M., Frolov, A., Volkov, V. & Stelmashchuk, I. Self-driving car steering angle prediction based on deep neural network an example of carnd udacity simulator. In *2018 IEEE 12th international conference on application of information and communication technologies (AICT)*, 1–5 (IEEE, 2018).
5. Chen, S. Autopilot - tensorflow github repository (2021).
6. Bojarski, M. *et al.* End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* (2016).
7. Tensor. Tensor flow (2021).
8. Keras. Dataset preprocessing (2021).
9. Wang, P. Y. & Nguyen, V. Improving training for end-to-end autonomous driving. .
10. Zhang, J. & Huang, H. Steering angle prediction for autonomous cars based on deep neural network method. In *2020 Australian and New Zealand Control Conference (ANZCC)*, 205–208 (IEEE, 2020).
11. Mastery, M. Gentle introduction to cnn lstm recurrent neural networks (2021).
12. Valiente, R., Zaman, M., Ozer, S. & Fallah, Y. P. Controlling steering angle for cooperative self-driving vehicles utilizing cnn and lstm-based deep networks. In *2019 IEEE intelligent vehicles symposium (IV)*, 2423–2428 (IEEE, 2019).
13. comma.ai. Comma.ai research github repository (2021).