Self-Driving Car: Predicting Steering Wheel Angle

Cristian Daniel Rodríguez Vázquez, A01209306, Tecnológico de Monterrey Campus Querétaro

***Abstract* – A convolutional neural network (CNN) approach is used to implement a self-driving car by predicting the steering wheel angle from input images taken by three front cameras located in the center, left and right of the car. The architecture of the model used was developed by NVIDIA for its autonomous navigation system called DAVE-2. The CNN is tested on the Udacity simulator.**

I. INTRODUCTION

Self-Driving cars are capable of sensing its surroundings and moving on its own through traffic and other obstacles with minimum or no human input (Manikandan, 2020). The usage of autonomous vehicles (AVs) has become a leading industry in almost every area of the world. Over the years, faster and more useful vehicles have been produced, but in our accelerated world with more and more cars, unfortunately the number of accidents have increased. In most of the cases, accidents are the fault of the human driver, therefore it could be theoretically replaceable with the help of self-propelled cars (Szikora, 2017). Many relevant companies like Waymo, Zoox, NVIDIA, Nissan, UBER are involved in this product development. With this type of car, the whole automotive transportation's safety, security, and efficiency is increased, and the human errors can be mitigated whilst the drive is made to its best (Manikandan., 2020).

AVs rely on various types of sensor technologies to perceive the environment and to make logical decisions based on the gathered information similar to humans. The most common types of AV sensors are RADAR, LiDAR, ultrasonic, camera, and global navigation system (Vargas et al., 2021).

The Advanced Driver Assistance System (ADAS) is a sis-tiered system that categorizes the different levels of autonomy, it ranges from vehicle being solely human-driven to those that are completely autonomous, as shown in figure 1.

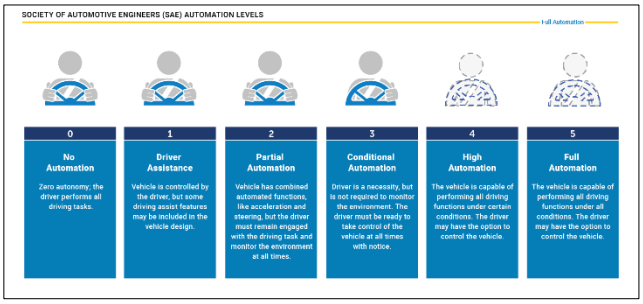

Figure 1. *Levels of autonomy. [1]*

II. STATE OF THE ART

The first work on modern CNNs occurred in the 1990s, inspired by the neocognitron. Yann LeCun et al., in their paper "Gradient Based

Learning Applied to Document Recognition" demonstrated that CNN model which aggregates simpler features into progressively more complicated features can be successfully used for handwritten character recognition (Draelos, 2019).

A CNN is a neural network that has one or more convolutional layers and are used mainly for image processing, classification, segmentation and, also, for other auto correlated data. Prior to the adoption of CNNs, most pattern recognition tasks were performed using initial stage of feature extraction by hand, followed by a classifier. Now, with the advancement of CNNs, it has become possible for features to be learned automatically from training examples surpassing human performance on standard datasets (Bojarski, et al., 2016).

The CNN approach is especially powerful in image recognition tasks because the convolution operation captures the 2D of images. Also, by using the convolution kernels to scan an entire image, relatively few parameters need to be learned compared to the total number of operations (Bojarski, et al., 2016).

III. DATASET

To obtain the data, the training mode was used in the Udacity simulator [2], driving the vehicle manually on the first track for four laps in one direction and four more laps in the opposite direction. The data log is saved in a csv file and contains the path to the images, saved in a folder, as well as steering wheel angle, throttle, reverse and speed. The steering angles came pre-scaled by a factor of 1/25 so that they are between -1 and 1. The data provided consisted of 6563 center, left and right jpg images for a total data size of 19689 examples. The images were of 320 width by 160 height, an example is shown below of the left/center/right images taken at one timestep from the car.

Figure 2. *Left, center and right camera images.*

Data Augmentation

Augmentation helps to extract as much information from data as possible. Four different augmentation techniques were used to increase the number of images that the model would see during training, with this the tendency of overfitting is reduced. The image augmentation techniques used are described as follows:

- Brightness reduction: changing brightness to simulate day and night conditions.
- Left and right camera images: using left and right camera images to simulate the effect of car wandering off to the side and recovering.
- Horizontal and vertical shifts: shifting the camera images horizontally to simulate

the effect of car being at different positions on the road and shifting vertically to simulate the effect of driving up or down the slope.

- Flipping: since the left and right turns in the training data are not even, image flipping was important for model generalization.

The following images, shows examples of the data augmentation applied.

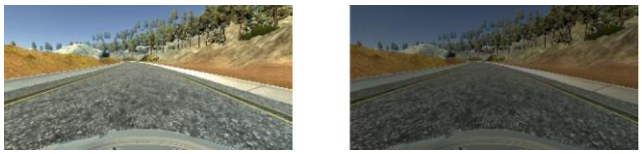

Figure 3. *Brightness reduction.*

Figure 4. *Horizontal and vertical shifts.*

Figure 5. *Image flipping.*

Data Preprocessing

After augmenting the images, data preprocessing was applied to format the images before using them by model training. The aim of preprocessing is to improve the quality of the image so that it can be analyzed in a better way. The image preprocessing techniques used are described as follows:

- Cropping: it was cropped the bottom 25 and top 40 pixels from each image to remove the front of the car and most of the sky above the horizon from the images.
- RGB to YUV: the images were converted from RGB to YUV type as it has more advantages in illumination changing and shape detection.
- Resizing: to be consistent with the NVIDIA model, all the images were resized to 66 x 200.
- Normalization: dividing by 255 the image pixel values to have just pixels values between 0 and 1.

The following figure shows an example of the image preprocessing applied.

Figure 6. *Image after preprocessing.*

Data Generator

As it is needed to generate thousands of new training instances from each original image, it is not possible to generate and store all these data on the disk. Therefore, keras generator were used to read original data from the log file, augment on the fly and use it to train the model producing new images at each batch.

IV. MODEL PROPOSAL

As previously mentioned, a CNN model was used. The model architecture used is inspired by the NVIDIA model used in its DAVE-2 system. The model architecture is shown in figure7.

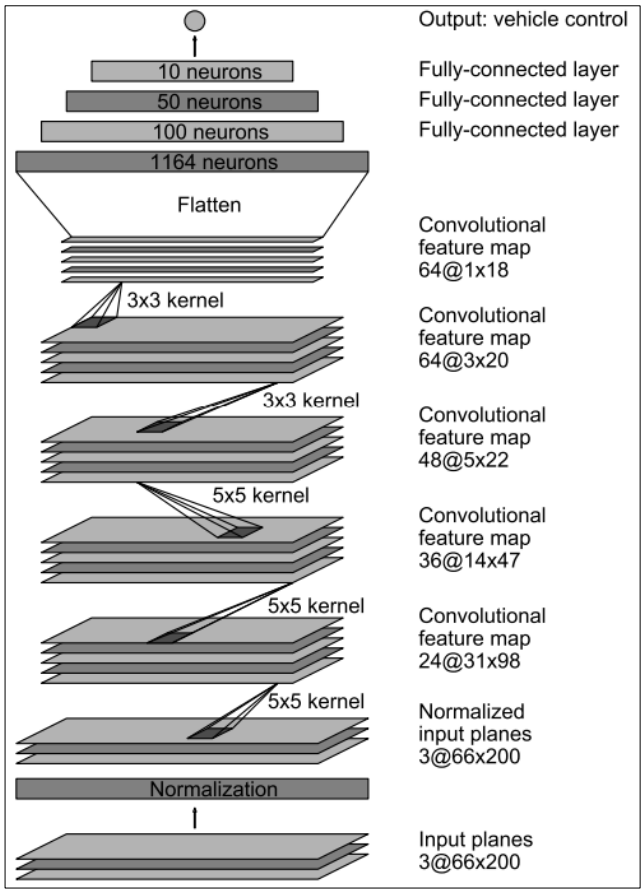

Figure 7. *CNN model architecture. [3]*

This model has the following characteristics:

- Input image is 66 x 200.
- It has a series of three of 5x5 convolutional layers, and the respective depth of each layer is 24, 36 and 48 as going deeper into the network.
- Then, there are two consecutive 3x3 convolutional layers, with a depth of 64.
- The result is flattened to enter to the fully connected phase.
- Finally, there is a series of fully connected layers, of gradually decreasing sizes: 1164, 200, 50 and 10.
- The output layer is of size 1, since the prediction is just one variable, the steering angle.
- It has 252219 trainable parameters.

Figure 8 shows the complete summary of the implemented model.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 31, 98, 24)        1824
 conv2d_1 (Conv2D)           (None, 14, 47, 36)        21636
 conv2d_2 (Conv2D)           (None, 5, 22, 48)         43248
 conv2d_3 (Conv2D)           (None, 3, 20, 64)         27712
 conv2d_4 (Conv2D)           (None, 1, 18, 64)         36928
 flatten (Flatten)           (None, 1152)              0
 dense (Dense)               (None, 100)               115300
 dense_1 (Dense)             (None, 50)                5050
 dense_2 (Dense)             (None, 10)                510
 dense_3 (Dense)             (None, 1)                 11
=================================================================
Total params: 252,219
Trainable params: 252,219
Non-trainable params: 0
_________________________________________________________________
```

Figure 8. *CNN model summary.*

The model uses the exponential linear unit (ELU) nonlinear as activation function, which in contrast to ReLU, ELU have negative values which allows to push mean unit activations closer to zero. Figure 9 shows the difference between ReLU and ELU.

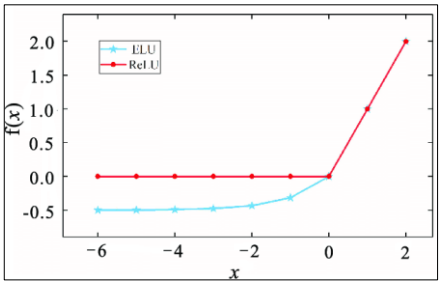

Figure 9. *ELU vs ReLU comparison. [4]*

Also, the Adam optimizer was used, which is an extension to stochastic gradient descent, it uses the squared gradients to scale the learning rate, updating the networks weights iterative based in training data. The learning rate used was an alpha of 0.0001. A summary of the parameters used for the model are shown on table 1.

Epochs	5
Steps per epoch	20000
Batch size	64
Optimizer	Adam (0.0001)
Activation function	ELU

Table 1. *Parameters used for the model.*

The model was trained using a NVIDIA GeForce RTX 3050 GPU and the entire training took about 6 hours.

V. TEST & VALIDATION

The initial dataset was divided into training and validation, being 80% for training and 20% for validation. Figure 10 shows the loss comparison between training and validation.

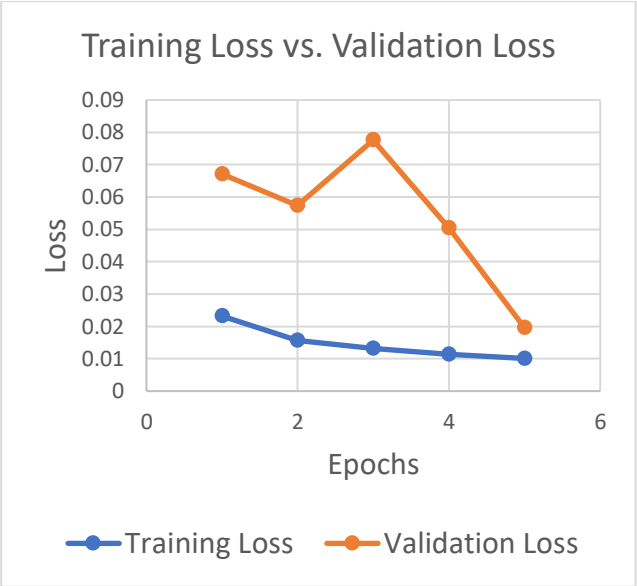

Figure 10. *Training loss vs. Validation loss.*

As we can see in figure 10, as the epoch increases, the loss value decreases. Although at the beginning there is a big gap between training loss and validation loss, at the end, they become almost equal.

The model was tested on both tracks. On the first track, where the training data was collected, the model was able to run smoothly, however it is a bit unstable as it makes many small turns between left and right.

On the other hand, in track 2 we can see that it has a better behavior and manages to run completely with a very good performance, so we can conclude that the model generalizes in a good way.

The video of the performance on both tracks is available at the following link: https://youtu.be/dgYWUmMOcOk

VI. CONCLUSIONS

It has been demonstrated that it is possible to create a model that reliably predicts the steering wheel angles of a vehicle using deep neural networks and a bunch of data augmentation techniques.

After observing the results, I believe that the obtained model is a very good one, since it performed well on both tracks. Even though, I think a better training can be done adjusting the parameters and/or increasing the number of epochs.

Possibly a model that has the same performance as the current one or even better can be achieved with less training time by adjusting the model parameters.

For the future, it would be good to explore the performance using another model architecture, using other data augmentation techniques, experiment with a reinforcement learning model and even test the performance of the model on a real car.

VII. REFERENCES

[3] Bojarski, M., et al. (2016). *End to End Learning for Self-Driving Cars.* Retrieved from https://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf

Draelos, R. (2019). *The History of Convolutional Neural Networks.* Retrieved from https://towardsdatascience.com/a-short-history-of-convolutional-neural-networks-7032e241c483

Gundling, C. (2017). *CNN Model Comparison in Udacity's Driving Simulator.* Retrieved from https://towardsdatascience.com/cnn-model-comparison-in-udacitys-driving-simulator-9261de09b45

Manikandan, T. (2020). *Self-driving car.* Retrieved from https://www.researchgate.net/publication/341132361_Self-driving_car

Singh, H. (2018). *Predicting Steering Angles using Deep Learning: Behavior Cloning.* Retrieved from https://becominghuman.ai/behavioral-cloning-in-deep-learning-using-keras-3786f5914b72

Szikora, P. (2017). *Self-driving cars – The human side.* Retrieved from https://www.researchgate.net/publication/324095773_Self-driving_cars_-_The_human_side

[2] Udacity. (2016). *Udacity's Self-Driving Car Simulator.* Retrieved from https://github.com/udacity/self-driving-car-sim

[1] Vargas, J., Alsweiss, S., Toker, O., Razdan, R. & Santos, J. (2021). *An Overview of*

Autonomous Vehicles Sensors and Their Vulnerability to Weather Conditions. Retrieved from https://www.mdpi.com/1424-8220/21/16/5397/pdf

Yadav, V. (2016). *An augmentation based deep neural network approach to learn human driving behavior.* Retrieved from https://chatbotslife.com/using-augmentation-to-mimic-human-driving-496b569760a9#.yget03t5g

[4] Yang, L., et al. (2020). *Random Noise Attenuation Based on Residual Convolutional Neural Network in Seismic Datasets.* Retrieved from https://www.researchgate.net/publication/339351694_Random_Noise_Attenuation_Based_on_Residual_Convolutional_Neural_Network_in_Seismic_Datasets