Revista Brasileira de Computação Aplicada, November, 2021

DOI: 10.5335/rbca.v13i3.12135
Vol. 13, Nº 3, pp. 32–41
Homepage: seer.upf.br/index.php/rbca/index

ISSN 2176-6649

ORIGINAL PAPER

An end-to-end approach to autonomous vehicle control using deep learning

Gustavo Antonio Magera Novello[1], Henrique Yda Yamamoto[1], Eduardo Lobo Lustosa Cabral[1,2]

[1]Dept. of Mechatronic Engineering, University of São Paulo, São Paulo, Brazil, [2]Instituto de Pesquisas Energéticas e Nucleares (IPEN), São Paulo, Brazil

*gustavo.novello7@gmail.com; henrique.ydaa@gmail.com; elobocabral@gmail.com

Received: 2020-12-14. **Revised:** 2021-10-12. **Accepted:** 2021-10-30.

Abstract

The objective of this work is to develop an autonomous vehicle controller inside Grand Theft Auto V game, used as a simulation environment. It is used an end-to-end approach, in which the model maps directly the inputs from the image of a car hood camera and a sequence of speed values to three driving commands: steering wheel angle, accelerator pedal pressure and brake pedal pressure. The developed model is composed of a convolutional neural network and a recurring neural network. The convolutional network processes the images and the recurrent network processes the speed data. The model learns from data generated by a human driver´s commands. Two interfaces are developed: one for collecting in-game training data and another to verify the performance of the model for the autonomous vehicle control. The results show that the model after training is capable to drive the vehicle as well as a human driver. This proves that a combination of a convolutional network with a recurrent network, using an end-to-end approach, is capable of obtaining a good driving performance even using only images and speed velocity as sensory data.

Keywords: Autonomous vehicle; Artificial intelligence; Convolutional neural network; Deep learning; Recurrent neural network.

Resumo

O objetivo deste trabalho é desenvolver o controle de um veículo autônomo dentro do jogo *Grand Theft Auto V*, utilizado como ambiente de simulação. É aplicada uma abordagem *end-to-end*, na qual o sistema mapeia diretamente as entradas provenientes da imagem de uma câmera colocada no capô do carro e de uma sequência de valores de velocidade para três comandos de direção: ângulo do volante, pressão do pedal do acelerador e pressão do pedal do freio. O controlador desenvolvido é composto por uma rede neural convolucional e uma rede neural recorrente. A rede convolucional processa as imagens e a rede recorrente processa os dados de velocidade. São desenvolvidas duas interfaces: uma para coleta de dados de treinamento e outra para controlar o veículo dentro do ambiente de simulação. Os resultados mostram que o sistema após o treinamento é capaz de dirigir o veículo tão bem quanto um motorista humano. Isso prova que a combinação de uma rede convolucional com uma rede recorrente, utilizando uma abordagem end-to-end, é capaz de obter um bom desempenho de direção mesmo utilizando apenas imagens e valores de velocidade como dados de sensores.

Palavras-Chave: Aprendizado profundo; Inteligência artificial; Redes neurais convolucionais; Redes neurais recorrentes; Veículo autônomo.

1 Introduction

The development and improvement of GPUs (Graphics Processing Unit) promoted great advances in the field of artificial intelligence. In this context, the use of artificial intelligence to control autonomous vehicles helps to improve the quality of drivers' life making possible the useful use of time spent driving and reducing the amount of accidents caused by human error.

Deep learning neural networks need a lot of data to be trained properly, but the work of capturing images and data for training in a real vehicle demands a lot of resources. Furthermore, it is very difficult to obtain data in all possible conditions of weather, hour of the day, traffic, road types etc. Thus, a form to accelerate the development of autonomous vehicle is to use games that simulate real environments to recreate different scenarios. Using large amounts of synthetic images and simulation data, make it possible to obtain better results when compared to deep learning algorithms that are trained using only real images acquired with a real vehicle (Johnson-Roberson et al., 2016). With a virtual environment it is simpler and faster to obtain a large volume of images and data in different conditions, since there is no need for a driver and a real equipped car with sensors. In addition, tests are performed more safely since the car is in a virtual environment.

The objective of this work is to develop, through simulation, the control of an autonomous vehicle within a virtual environment using images from a camera positioned on the vehicle's hood, vehicle speed data, and driver's commands for steering wheel angle, brake and accelerator pedal pressures. The images and the vehicle speed are the inputs of the model and the driver's commands are used as desired outputs to train the model.

This work is divided in 8 sections. Section 2 presents a brief review of the literature describing two similar works. Section 3 describes the simulation environment. Section 4 describes the interfaces for the game. Section 5 describes the data used for training the model. Section 6 presents the architecture of the model developed. In Section 7 it is presented how the model is trained and finally in Sections 8 and 9 the results and conclusions are presented respectively.

2 Related works

Bojarski et al. (2016) developed a system called DAVE-2 capable of driving in real roads without the need to recognize street elements, such as lanes and boundaries, using only one convolutional neural network to learn the entire vehicle steering process directly from camera images. The system has three cameras positioned on the left, center and right of the car that collect the images simultaneously with the steering wheel angle of the driver. After training, the model is capable of receiving images in real time and calculating the vehicle steering wheel angle. The architecture of the DAVE-2 neural network consists of one normalization layer, five convolution layers followed by three densely connected layers. The first three convolution layers use stride equal to 2 and a 5x5 kernel, and the remaining two use a 3x3 kernel and stride equals to 1. The output of the last densely connected layer is the vehicle steering angle. Tests were carried out with a real car. The results showed that the model was able to generate steering angles autonomously 98% of the time, demonstrating that it is possible to obtain good results for controlling the direction of a vehicle only with the use of cameras.

Yang et al. (2018) developed a model that uses a convolution neural network together with a recurrent neural network. This model is able to predict the steering angle and the desired speed of the vehicle, receiving as input the road image and a sequence of previous speeds. This work improves the work of Bojarski et al. (2016) in the sense that the model can control both the steering angle and the speed of the vehicle simultaneously, greatly improving the vehicle's driving autonomy in comparison with other models proposed in the work. The convolutional network of model, that the authors named as Multi-modal Multi-task Network, has 5 convolutional layers and 4 densely connected layers, and it is used to determine the steering angle. The output of the recurrent network is concatenated with the output of the second densely connected layer of the convolutional network to obtain the speed command. According to the authors the separation of the calculation into two neural networks reduces considerably the amount of computational processing required and allows processing with high frames per second rates ensuring better performance in real time.

3 Simulation environment

Using a virtual environment facilitates development and reduces costs to create, replicate and iterate situations as compared to a real environment (Martinez et al., 2017). The computer game Grand Theft Auto V (GTA-V) is chosen to create the simulation environment. The GTA-V is an open-world computer game, very rich in elements and details that represents a replica of the real world. Within the GTA-V "world" map a circuit was chosen and set up to simulate the environment to obtain data for training and for testing the model developed for controlling autonomous vehicles. The circuit chosen is

Figure 1: GTA-V circuit map used to collect data for training and testing the developed model

approximately 850 m long and it has several streets where the lane is not delimited. The landscape observed in the route contains vegetation, trees, hills and other elements seen in common real roads. In this first approach, the simulation environment does not include the presence of other vehicles and pedestrians. In addition, it is possible to control variables such as hour of the day and weather conditions facilitating testing and enabling a variety of training conditions. The representation of the circuit map and an its aerial view are represented in Fig. 1. Some examples of images collect by the car hood camera are shown in Fig. 2.

4 Interface with GTA-V game

Two interfaces for the GTA-V game were developed and used to acquire the training data and to control the vehicle inside the game. Theses interfaces are available and described with more details in Novello and Yamamoto (2020).

The data acquisition interface captures real-time images displayed on screen at 10 fps rate using the ImageGrab module from Python library Pillow. These images are resized to (240, 150, 3) pixels and stored in RGB format in Numpy arrays with pixel values ranging from 0 to 255. The driving data is acquired with the PyGame library that reads the joystick and triggers commands from a controller handled by a human driver. The joystick controls the steering wheel and the accelerator and brake pedals. Speed data is collected with in-game modifications named ScriptHookV and Native Trainer (Blade, 2019). Native Trainer is used with few modifications.

The control interface sends driving commands during simulations inside the game using as inputs real-time acquired images and normalized values of speed. These data are acquired with the data acquisition interface and are normalized to be used in model. The model output is processed using a virtual joystick simulated with vJoy driver and the Pvjoy library. Finally, the software x360ce sends the commands to the game to control the vehicle. A diagram that summarizes the operation of these two interfaces is shown in Fig. 3.

5 Training Dataset

The dataset used for training the model consists of about 130 thousand data samples captured at a rate of 10 fps, completing more than 3½ hours of video. The videos were collected at different hours of the day with different weather conditions to improve the generalization of the model. Associated with each image of the videos there are 4 parameters: steering wheel angle, accelerator and brake pedal pressures, and vehicle speed. For collecting this data, a human driver drove the vehicle inside the GTA-V game environment.

The steering wheel angle, the accelerator and the brake pedal pressures represent relative values varying between zero and one. For the accelerator and brake pedal pressures, zero corresponds to the natural unpressed position. For the steering wheel angle, 0.5 corresponds to the neutral position of the steering wheel, values above 0.5 correspond to clockwise angles and under 0.5

Figure 2: Examples of images collected from the car hood camera used for training the model.

to counterclockwise angles. An example of the steering wheel angle, accelerator and brake pedal pressures, and vehicle speed for one lap in the circuit is shown in Fig. 4. A summary of these values, that shows the distributions of the accelerator and brake pedal pressures, vehicle speed and steering wheel angle in the dataset are presented in Fig. 5. Note that in the brake pedal pressure histogram the amount of values is in logarithmic scale because it remained in neutral position most of time. Also note that as the circuit has more curves to the right there is a higher concentration of positive steering wheel values.

The driver's steering wheel angle, and accelerator and brake pedal pressures are the desired outputs for the model while the speed history and the images are the model inputs. This data is used to train the model with supervised learning. The solution to the problem follows an end-to-end approach where a direct association between the images and the driver's driving commands is learnt by the model, so that it is not necessary to consider the detailed kinematics and dynamics of the vehicle.

Each input sample consists of one image and 50 vehicle speed values, i.e., the current speed and 49 previous values. This 50 samples of speed correspond to a temporal sequence of 5 seconds and are normalized to have zero mean and values between -1 to 1, so the values are subtracted by the mean and divided by a scale factor. For each input it is associated the desired output with three values that corresponds to the steering wheel angle, accelerator pedal pressure and brake pedal pressure. The total number of examples in the dataset is 129,515. This data is divided into a training dataset with 60% of the examples, a validation dataset with 20% of the examples and a test set with the remaining 20% of the examples. Before separation, the data is randomly shuffled so that to ensure the diversity of the data in the both training and validation datasets. This dataset is available in Cabral et al. (2020).

6 Background

As explained in Chollet (2017), Convolutional Neural Networks are widely used in image processing and pattern recognition. The main step of this technique is convolution, which consists on sliding a filter, a matrix with specified dimensions, through the desired image. The filter slides through the image with a determined spacing, called stride, performing multiple matrix multiplications with pixels values. A bias is added to the result and an activation function (nonlinear function) is applied. A convolutional layer has multiple filters that extract specific patterns from the image.

The main advantage of a convolutional layer is the capability to learn and extract local patterns in images, such as vertical, horizontal edges, textures or colors. After the filters are learnt, they can be used to extract these patterns in any image. This generalization makes CNNs very efficient for image processing. The first layers extract

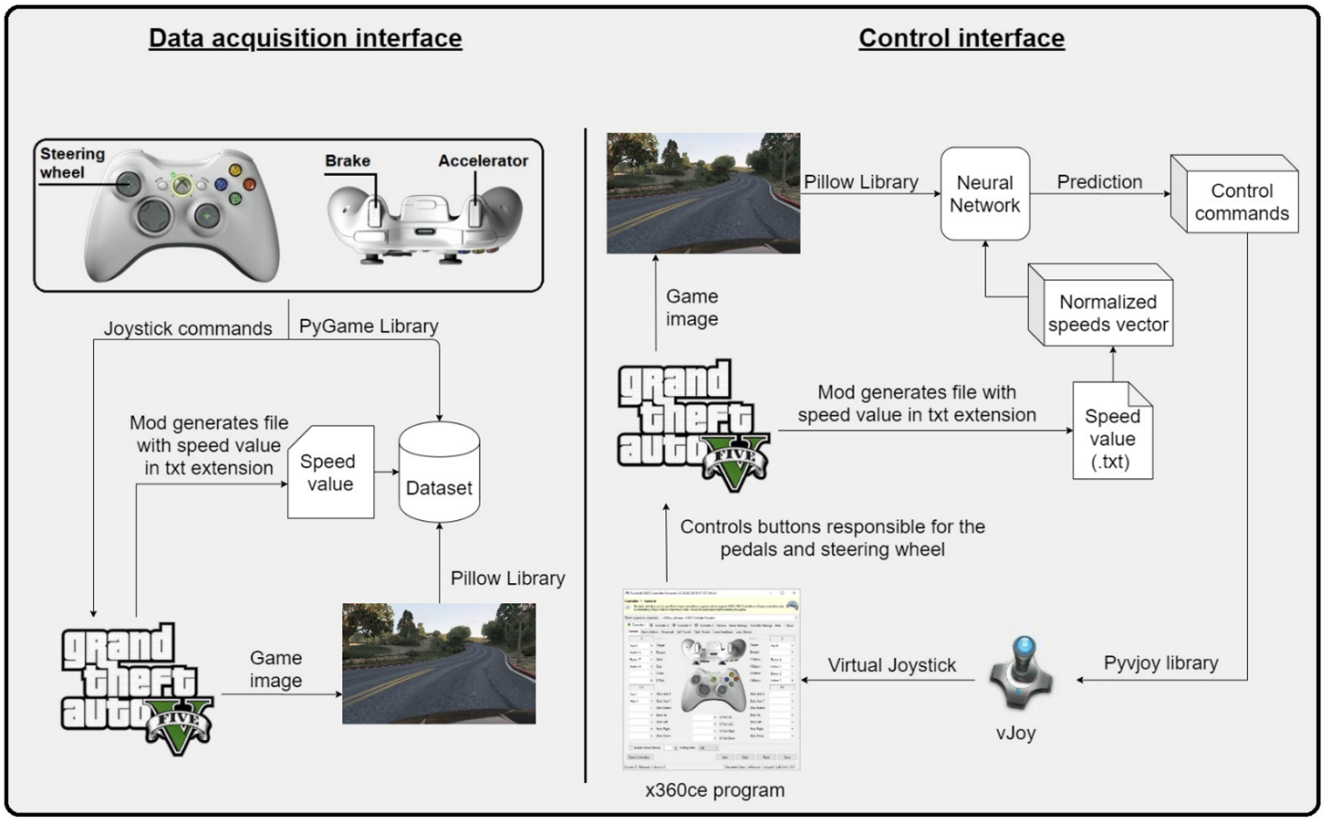

Figure 3: Diagrams of the interfaces developed for the GTA-V game used to acquire and send data.

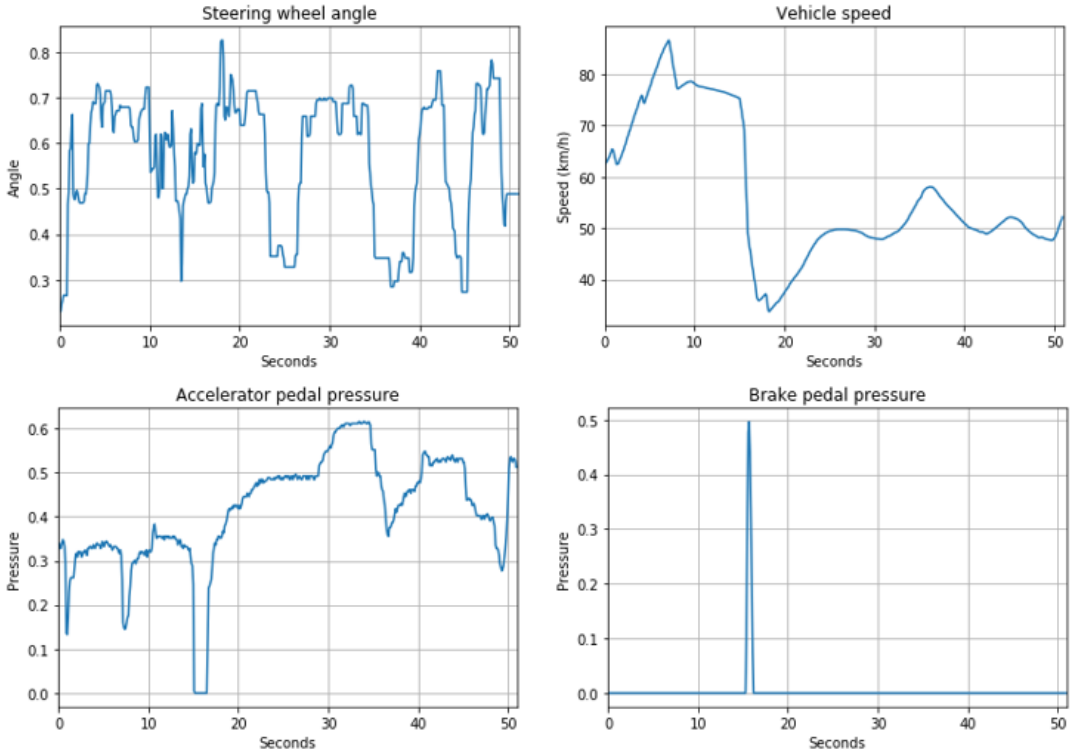

Figure 4: Example of the steering wheel angle, accelerator and brake pedal pressures and vehicle speed for one lap in the circuit.

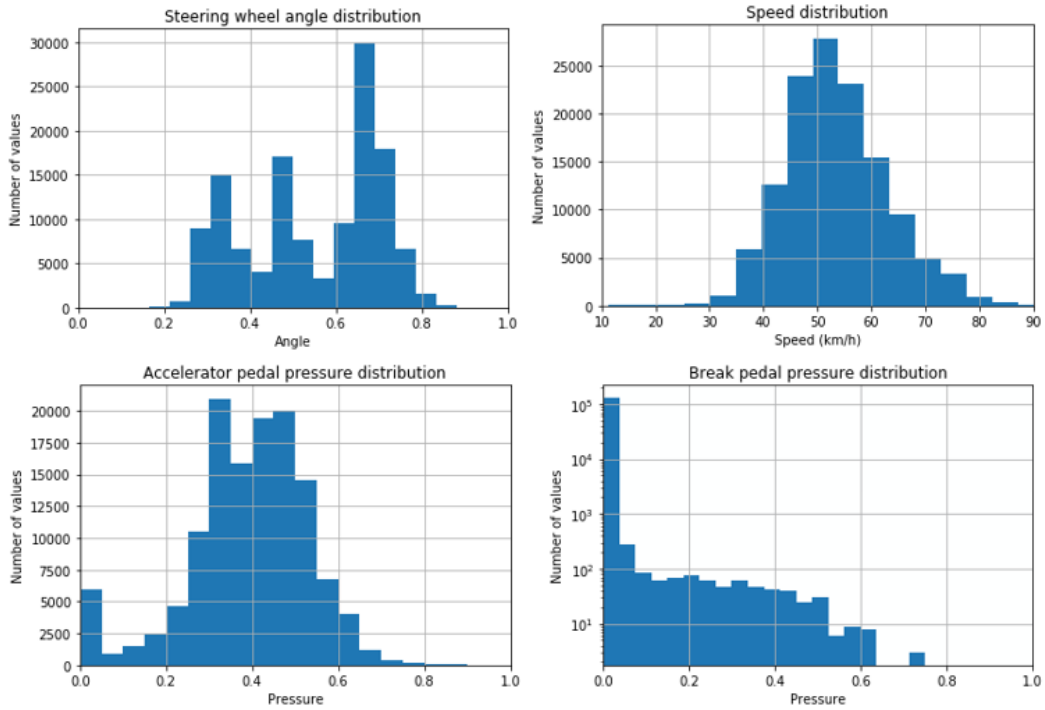

Figure 5: Distribution of the data for steering wheel angle, accelerator and brake pedal pressures, and vehicle speed. The amount of values of brake pedal pressure distribution is in logarithmic scale.

simpler patterns from the images and the deeper ones can recognize complex patterns. Fig. 6 illustrates this aspect: the first layer recognizes the contours of the cat's face, eyes and ears and the second layer, more complex patterns. With this, it is possible to identify the image as a cat in the output layer, for example.

Recurrent Neural Networks are used to process sequences and time series, which, unlike dense and convolutional networks, require memory. This enables the previous output to influence on next predictions, thus, the network output does not depend only on the current input vector.

7 Model architecture

The model developed is similar to the model of Yang et al. (2018), i.e., it has a convolutional network and a recurrent network. The images are the input of the convolutional network and the sequential speed history is the input of the recurrent network.

A pretrained convolutional network is used in the convolutional part of the model developed. Three convolutional networks, pre-trained with the Imagenet dataset (Krizhevsky et al., 2012), were tested. The VGG16 network (SIMONYAN et al., 2015) was the one that performed best so it was chosen. Based on the work of Yang et al. (2018) to process the sequential speed history a recurrent network is used in parallel to the convolutional network. The complete model is shown in Fig. 7.

The convolutional part of the model shown in Fig. 7 is the left branch. This network is composed of the convolutional part of the VGG16 network (SIMONYAN et al., 2015) and two densely connect layers separated by a dropout layer. The first dense layer has 256 units and the second dense layer has 50 units. The RELU activation function is used in these two layers. Image normalization is performed inside the network using a lambda layer which divides the pixels of the images by 255.

The recurrent part of the model is the right branch shown in Fig. 7. This network has a LSTM layer followed by a densely connected layer. The LSTM layer has 128 units and the dense layer 50 units with RELU activation function.

The outputs of the convolutional and recurrent networks are concatenated and then processed by two dense layers to calculated the output of the model, i.e., the vehicle's driving controls (steering wheel angle, accelerator pedal pressure and brake pedal pressure). The first of these two layer has 50 units with RELU activation function and the output layer has 3 units with sigmoid activation layer.

The total number of parameters of the model is 18,501,111, but only 3,786,423 are trainable. The other 14,714,688 parameters are VGG16 pre-trained frozen parameters.

The great advantage of this configuration used in the model is that it allows the visual information contained in the images to be used together with the vehicle's temporal speed information to determine all the vehicle's driving commands. This model attempts to reproduce the way humans drive a vehicle, both in terms of information received and of driving commands generated.

Comparing with the model developed by Yang et al. (2018), many modifications are made. In the model of Yang et al. the outputs are just the steering wheel angle and the desired speed for the vehicle. In addition, in the model of Yang et al. the convolutional and recurrent parts are joined in such a way that only the recurrent part receives information from the convolutional part. Thus, the convolutional part of the model of Yang et al. only calculates the vehicle's steering angle and does not receive any temporal information of the vehicle's speed.

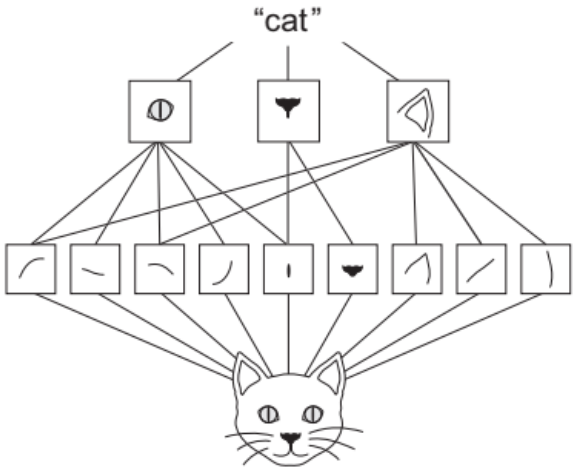

Figure 6: Hierarchical learning of convolutional neural networks (Chollet, 2017).

8 Training

The model is trained with the input data (camera images and speed history) and desired outputs (vehicle steering commands generated by a human driver). Table 1 shows the set of hyperparameters used for training.

Table 1: Hyperparameters used for training the model.

Hyperparameters	
Batch Size	128
Epochs	100
Dropout rate	0.4
Optimizer	Adam
Learning Rate	$1x10^{-4}$

The cost function and the metric used are respectively the mean square error (MSE) and the mean absolute error (MAE). The values obtained for these functions for the validation data are: MSE equal to 0.0015, or 0.15%, and MAE equal to 0.0216, or 2.16%. The dropout layer helped to reduce the overfit during training. These values are very small, showing that the model is capable of performing the desired transformation of the input data into the output data. Fig. 8 shows the values of the loss function (MSE) and the metric (MAE) in each epoch during training.

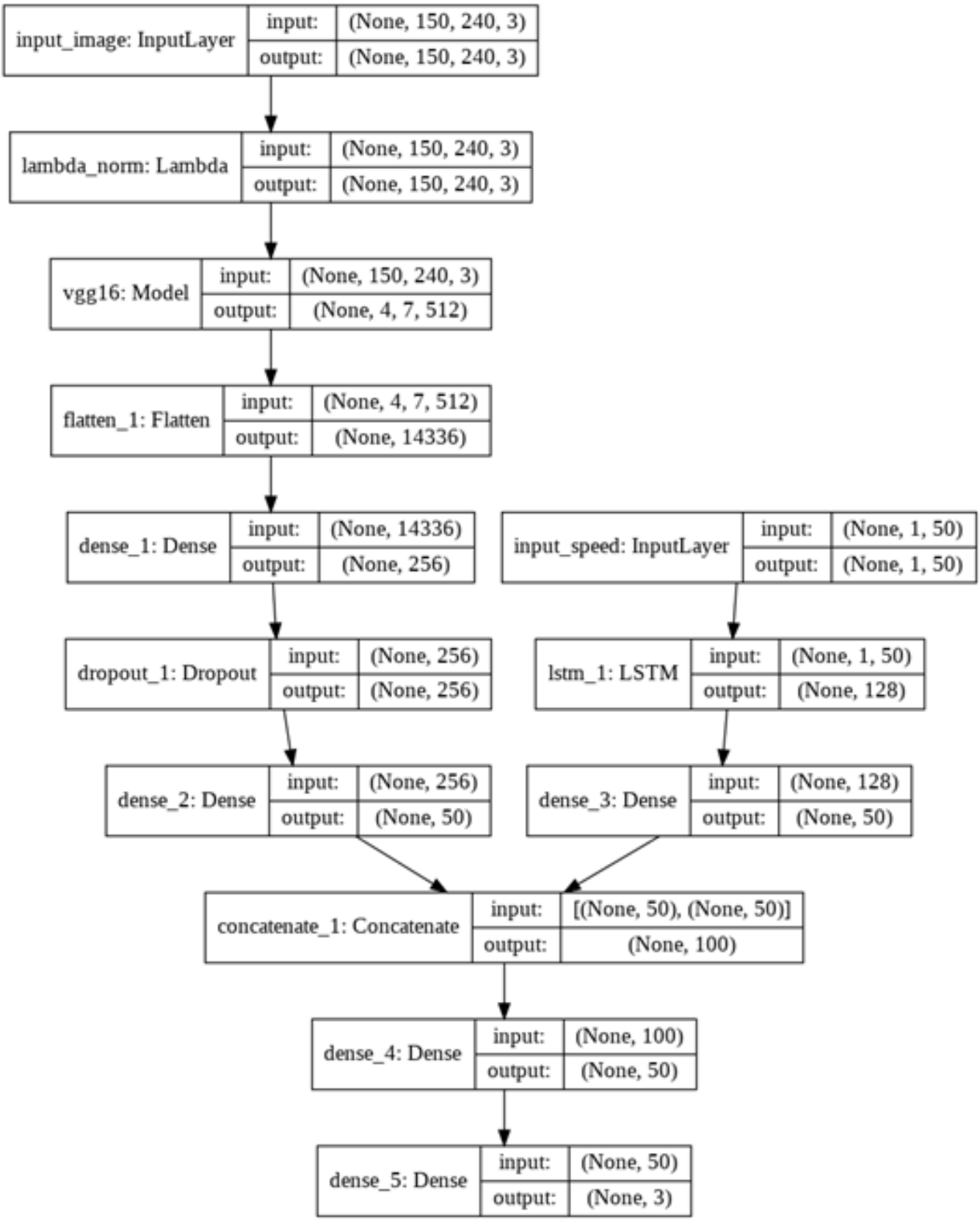

Figure 7: Scheme of the model developed for autonomous driving vehicle.

9 Results

The model developed is tested under different weather conditions and hours of the day to verify its performance in driving the vehicle inside the game environment. The vehicle completes a circuit loop in about 1 minute and 13 seconds without colliding with any obstacle on the road and without deviating from the correct road lane. A video that summarizes the model driving performance in different weather and hours of the day may be seen in Novello and Yamamoto (2019).

To graphically view the vehicle control values generated by the model, an auxiliary dataset is obtained that contains driving through one circuit lap. This data contains the images, the vehicle speed and the human driver's steering commands. Speed values and images are used as model input and the model output is compared to the human driver's actions. These data allow to compare the angle of the steering wheel and the pressures on the pedals of the accelerator and brake used by the human driver with those calculated by the model. The results are shown in Figs. 9 and 10. Note that the brake pedal is not pressed on this lap. From the results shown in Fig. 9, it is observed that the steering wheel angle generated by the model behaves similarly to that performed by the human driver, with a mean absolute error of 2.9%. The results presented in Fig. 10 show that at the beginning of the circuit, the model accelerates the vehicle more than the human driver, but since there is not a reference speed value, it is not possible to evaluate this result negatively, even because the vehicle controlled by the model is able to perform the circuit lap without any collision and keeping the vehicle within the expected road lane.

On some rare occasions, the vehicle driven by the model has failed to remain on the track. Note that for this failure condition there are no examples in the database used in the training showing which would be the right action to take. The location of the circuit where this fault has the highest occurrence is on the left curve indicated by the number 3 in Fig. 1. This failure is probably caused by the data imbalance, as there is a predominance of right curves in the chosen circuit. Note that curve 3 follows closely curves 1 and 2. In some situations, poor vehicle orientation at the exit of curve 1 (see Fig. 2) caused the failure, in some other cases, a high speed at the entrance of curve 2, which is sharper, may have been the cause. It is noteworthy that in the road before curve number 1 there are no tracks demarcating the lanes.

In order to estimate the frequency of failure of the model a vehicle's autonomy measurement is defined. This measurement is defined as the time interval for the vehicle get lost on the track. This does not consider the cases where the vehicle exceeds the lane limits and recovers immediately. The results shown in Table 2 are obtained under five weather and hours of the day conditions. The information on the fault location shows the number of the curve where the vehicle get lost. Note that the last test performed is interrupted without any failure after exceeding 80 minutes of complete vehicle autonomy.

10 Conclusions

In this work a neural network controller is developed for an autonomous vehicle based on an end-to-end approach using a simulation environment. To simulate a street environment, the game GTA-V was used due to the diversity of scenarios and the realism of the details of the images making them similar to reality. Two interfaces for the GTA-V game were developed allowing to collect about data in different conditions of weather and hours of the day and to control the vehicle inside the game.

The model is composed of a convolutional neural network in parallel with a recurrent neural network. The outputs of these two networks are concatenated and further processed to generate the vehicle's driving commands. To train the neural network model, the images from the camera and the vehicle speed history are used as input data, and the steering wheel angle and pressure controls on the accelerator and brake pedals determined by the human driver are used as desired output data.

The results of the loss function and the metrics obtained during training indicate little overfit and a good performance on the validation and test datasets. The

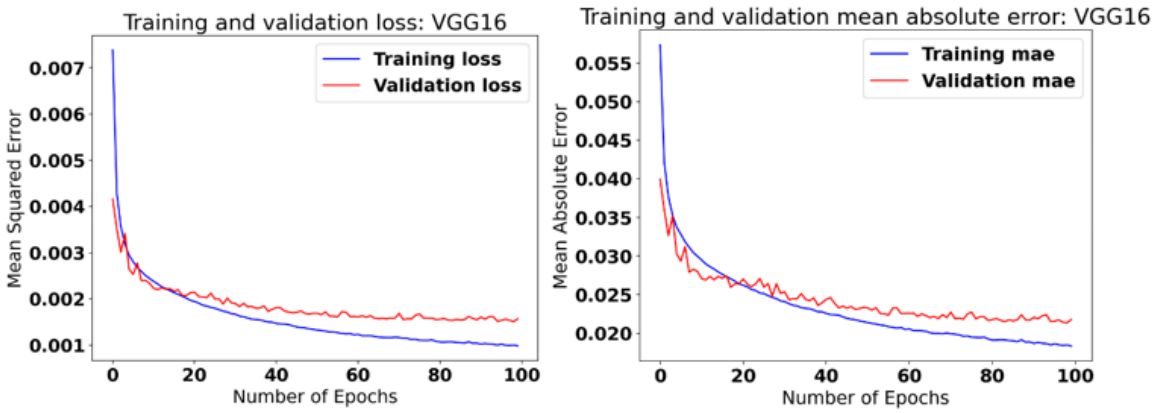

Figure 8: Results of loss function (MSE) and metric (MAE) during training.

model's performance is good, being able to autonomously drive the vehicle through several laps of the circuit. Some point failures of the model are detected mainly in curves to the left. This failure is probably caused by the fact that the data has more examples with curves to the right.

Finally, the results obtained allow to conclude that the end-to-end approach used together with the neural network model architecture are satisfactory, being possible to obtain a good level of autonomy for a vehicle using as input data only images and a temporal sequence of speed.

To improve and enrich the simulation in a virtual environment of an autonomous vehicle control, it is intended in future work to make the following improvements:

- The database collected has a good diversity of weather conditions and hours of the day, but it is necessary to collect data on different types of roads and also to include other vehicles, pedestrians, traffic signs etc.

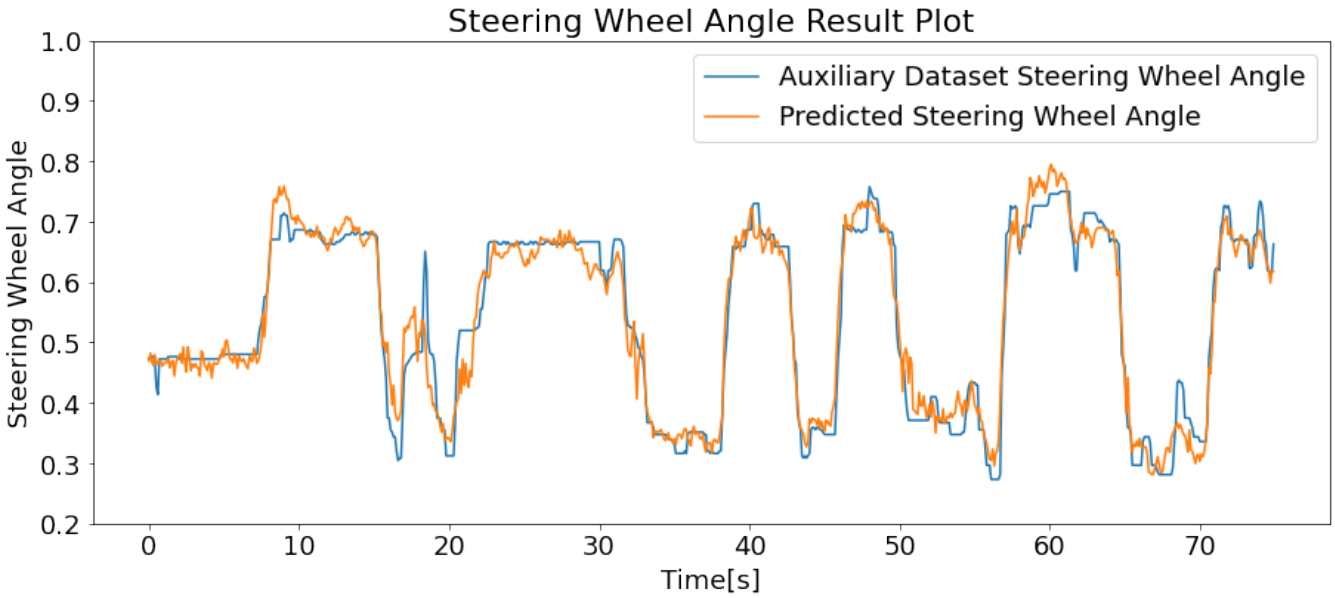

Figure 9: Comparison between steering wheel angle used by the human driver and that calculated by the model in one lap of the circuit.

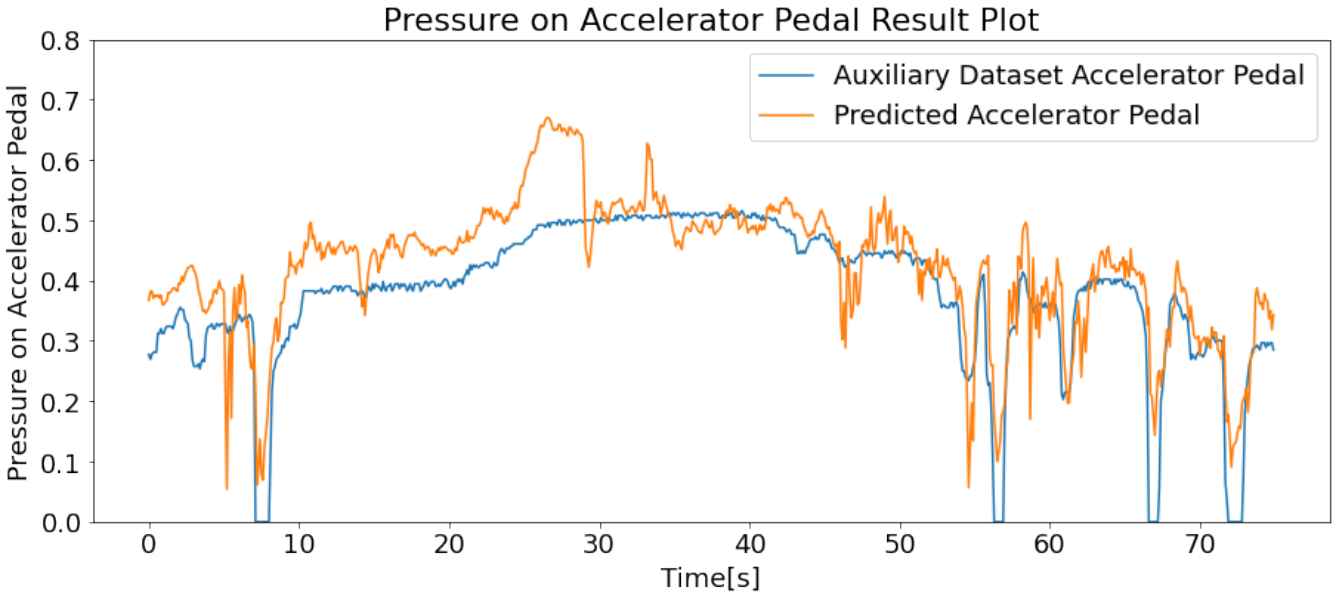

Figure 10: Comparison between the pressure on the accelerator pedal used by the human driver and that calculated by the model in one lap of the circuit.

Table 2: Time between failure of the vehicle under the model driving control.

Weather	Hour of the day	Failure location	Time of autonomy
Clear	Night	3	5 min and 49 sec
Clear	Morning	3	8 min and 9 sec
Raining	Afternoon	2	1 min and 8 sec
Raining	Sunset	3	11 min and 18 sec
Cloudy	Sunset	3	Unlimited

- Design a control architecture capable of planning the vehicle's route in order to increase the autonomy of the vehicle controller.
- Train the model to recover from error states by adding in the training data examples of situations representing the recovery of the vehicle in error states.

References

Blade, A. (2019). Ab software development - script hook v. Available at http://www.dev-c.com/gtav/scripthookv/.

Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J. and Zieba, K. (2016). End to end learning for self-driving cars, *CoRR* **abs/1604.07316**. Available at http://arxiv.org/abs/1604.07316.

Cabral, E. L. L., Novello, G. A. M. and Yamamoto, H. Y. (2020). Autonomous vehicle dataset. Available at https://github.com/elcabral/Autonomous-vehicle-dataset.

Chollet, F. (2017). *Deep Learning with Python*, 1 edn, Manning Publications Co., Greenwich, CT, USA.

Johnson-Roberson, M., Barto, C., Mehta, R., Sridhar, S. N. and Vasudevan, R. (2016). Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?, *CoRR* **abs/1610.01983**. Available at http://arxiv.org/abs/1610.01983.

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Available at http://www.cs.toronto.edu/~hinton/absps/imagenet.pdf.

Martinez, M. A., Sitawarin, C., Finch, K., Meincke, L., Yablonski, A. and Kornhauser, A. L. (2017). Beyond grand theft auto V for training, testing and enhancing deep learning in self driving cars, *CoRR* **abs/1712.01397**. Available at http://arxiv.org/abs/1712.01397.

Novello, G. A. M. and Yamamoto, H. Y. (2019). Controle de veículo autônomo - gta v. Available at https://www.youtube.com/watch?v=8m_QSb9NPIM&list=PLdO8eGidHLo-vDP5z3dpIgSQJq8RCQxfM.

Novello, G. A. M. and Yamamoto, H. Y. (2020). Veiculo-autonomo-gta-v. Available at https://github.com/henriqueyda/Autonomous-Vehicle-GTA-V.

SIMONYAN, ZISSERMAN, 2015. Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. Available at https://arxiv.org/pdf/1409.1556.pdf.

Yang, Z., Zhang, Y., Yu, J., Cai, J. and Luo, J. (2018). End-to-end multi-modal multi-task vehicle control for self-driving cars with visual perception, *CoRR* **abs/1801.06734**. Available at http://arxiv.org/abs/1801.06734.