World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:16, No:6, 2022

Automated Driving Deep Neural Network Model Accuracy and Performance Assessment in a Simulated Environment

David Tena-Gago, Jose M. Alcaraz Calero, Qi Wang

Abstract—The evolution and integration of automated vehicles have become more and more tangible in recent years. State-of-the-art technological advances in the field of camera-based Artificial Intelligence (AI) and computer vision greatly favor the performance and reliability of Advanced Driver Assistance System (ADAS), leading to a greater knowledge of vehicular operation and resembling the human behaviour. However, the exclusive use of this technology still seems insufficient to control the vehicular operation at 100%. To reveal the degree of accuracy of the current camera-based automated driving AI modules, this paper studies the structure and behavior of one of the main solutions in a controlled testing environment. The results obtained clearly outline the lack of reliability when using exclusively the AI model in the perception stage, thereby entailing using additional complementary sensors to improve its safety and performance.

Keywords—Accuracy assessment, AI-Driven Mobility, Artificial Intelligence, automated vehicles.

I. Introduction

THE automotive industry has evolved significantly with several main car manufacturers having achieved automated driving capabilities. It is expected that, in 2030, up to 50% of the passenger vehicles sold will have a high degree of autonomy (between level 3 and level 4 [1]), and that up to 15% will be fully automated (level 5), reaching an economic market value of $7 trillion by 2050 [2]. Meanwhile, the reliability, security, ethics and infrastructure etc. are still barriers for the adoption of automated vehicles on a massive scale globally. AI-based methodologies have gained momentum in developing the most avant-garde automated driving solutions. However, the difficulty of data-driven approaches is to collect a large ground truth data-set from a real environment for training and validation. The scarcity of ground truth in real environments prompts developers to focus their experiments on virtual environments such as driving simulators. In these simulators, obtaining data related to the ground makes it possible to know the reliability and accuracy of AI implementations.

Automated driving utilises RADARs, IMU and GPS sensors combined with DNN (Deep Neural Networks) models to reach a wide perception of the *Ego*[1] environment. This methodology tries to straighten the AI model performance by fusing its output with other sensory sources. The reason usually comes from the lack of enough accuracy to perform the context perception uniquely by the DNN model. Although major progress is being made on the ability of these AI camera-based perception modules, the adoption of this technique as the only source of information for the *Ego* is still challenging for the current AI technologies.

The lack of experiments for evaluating a camera-based DNN as the only source of perception for an automated driving architecture has motivated this research. Therefore, the main aim is to conduct an analysis of a representative AI-based automated driving solution to evaluate its accuracy and performance in a simulated environment. This study will explore the capabilities of the DNN model to assess its potential to be the only source of information provided to feed the ADAS trajectory planner module. This research focuses on the design and generation of tailored common driving scenarios and compare the ground truth data obtained from the simulation with the predicted values from the AI model.

The rest of the paper is structured as follows: Section II reviews the current open-source automated driving alternatives and compares the associated research work. Section III describes the AI model's structure, input and output. Section IV assesses the AI model's performance and accuracy in a simulated environment. Section V concludes the paper.

II. Related Work

In the market of automated driving, only a few companies have developed fully functional models in real products, and even more rare are those companies that open-source their implementations. Among the current open-source solutions, the companies Apollo from Baidu, and OpenPilot from Comma.ai stand out. Apollo differs from its competitors by trying to achieve a Level 4[2] of automation, based on a distributed environment in which different modules of LiDARs, RADARs and cameras merge their computations to achieve a virtual representation of the driving environment. In contrast, OpenPilot addresses RADAR and camera-based Level 2 automation, in which the driver still has to actively involve himself to performing tasks in collaboration with the ADAS. These two companies offer the necessary code and resources to run their ADAS through their public repository.

David Tena-Gago is with the School of Computing, Engineering & Physical Sciences, University of the West of Scotland, Paisley, PA1 2BE, United Kingdom (corresponding author, e-mail: david.tena@uws.ac.uk).

Professor Jose M. Alcaraz Calero and Professor Qi Wang are with the School of Computing, Engineering & Physical Sciences, University of the West of Scotland, Paisley, PA1 2BE, United Kingdom.

[1]In automated driving systems, the automated vehicle is commonly referred to as the *Ego*

[2]The Society of Automotive Engineers (SAE International) describes 6 levels of automation for automated vehicles. See [1].

TABLE I
COMPARISON TABLE ON THE PREVIOUS WORK DONE ON OPEN-SOURCE FUNCTIONAL AUTOMATED DRIVING AI MODULES

Reference	Target	Version	Assessment	AI module input source
[3]	Apollo	3.5	Camera & RADAR performance and robustness evaluation with noise injection	Kitti dataset
[4]	OpenPilot	0.4.2	Camera & RADAR-based AAC hazard identification & resilience evaluation to camera fault injection	Caltech dataset
[5]	OpenPilot	n/a	Camera & RADAR based FCW risk perception with adversarial weather conditions	CARLA simulator
[6]	OpenPilot	0.6.6	Camera-based ALC evaluation under road dirtiness injection	LGSVL simulator
[7]	OpenPilot	0.8.5	Camera & RADAR fusion-based FCW assessment on collision context	CARLA simulator
Our contribution	OpenPilot	0.8.2	Camera-based *Ego* & leading vehicle dynamics perception accuracy and performance evaluation	CARLA simulator

There are several publications dedicated to investigating how effective their perception modules based on AI and multi-sensor data fusion are. Table I summarizes the research target of the most recent studies on these two implementations.

As shown in Table I, the vast majority of research were oriented towards behavior analysis and risk assessment in the OpenPilot perception module, while only [3] used Apollo. As the table also indicates, the difficulty of obtaining a ground truth data-set to validate the results of the ADAS, forces the researchers to use driving simulators. This causes greater use of OpenPilot over Apollo, since the former option offers a ready-to-use development suite with tools to bridge the CARLA simulator with the automated driving system. Similarly, the table reflects that the main object of study in the evaluation of OpenPilot is the lane lines detection capacity, and in some cases, the leading vehicle detection. It is noted that [6], comparing different lane detection AI techniques, is used over OpenPilot 0.7.0 trajectory planner. This research work has attempted to test the accuracy of the AI model that OpenPilot uses in its version 0.8.2 in relation to the perception it has about the *Ego* dynamics, and the dynamics of potential vehicles in front of it.

III. MODEL ANALYSIS

This section explains the components of the AI model, and its input and output format, as shown in Fig. 1.

A. Model Structure

- Optical feature extraction: its objective is to extract the relevant features from the camera frames by means of a Convolutional Neural Network (CNN). It outputs a 1024-long flattened array that will be used as the head for the following stages of the processing. Its internal structure comprises 70 convolutional layers and 46 ELU activations. It does not have any normalization layers. Its structure is, in turn, based on the EfficientNet-B2 backbone [8].
- Recurrent memory feature extraction: features a GRU-based (Gated Recurrent Unit) neural network. It outputs a 512-long array containing information related to the previous executions of the model. This neural network is used as an efficient approach to either forget or remember parameters extracted from preceding executions of the model, which allows the perception module to have a wider perspective about the previous context throughout the execution process. The output array will be reused as an input for the GRU next iteration. At the end of this stage, the GRU output and the CNN output are concatenated into a 1024x1536 array.
- Predicting output layers: perform the final prediction of the model. It is, in turn divided into 2 distinguished parts:
  - Frame policy: predicts information only based on the image feature extraction carried out by the convolutional layers.
  - Temporal policy: considers the parameters that have changed along the time from previous iterations. Its prediction is based on the input image feature extraction and the GRU output.

  This stage is composed of 6 blocks of prediction, each of them in charge of predicting a different output.

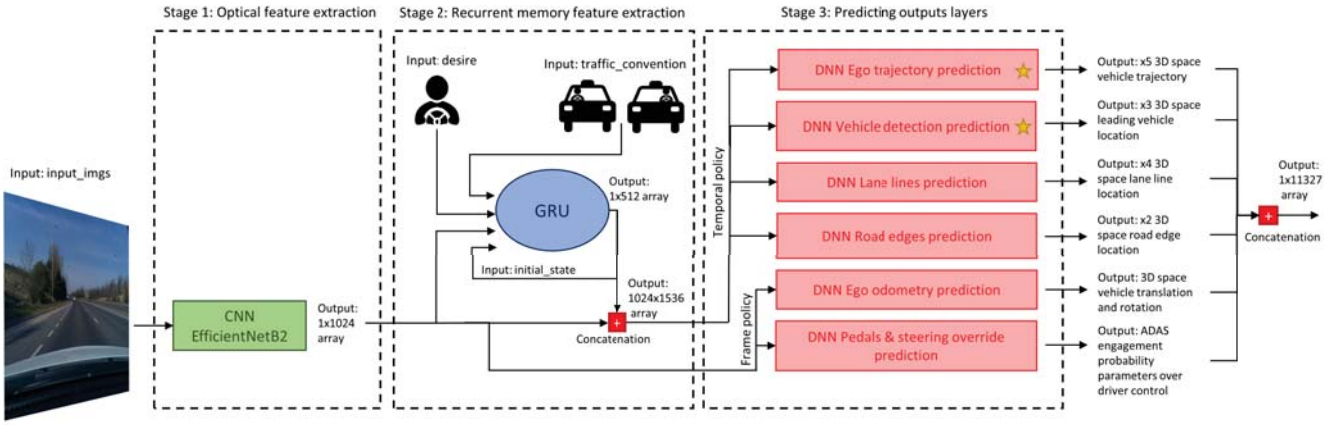

Fig. 1 AI model general structure

B. Model Input

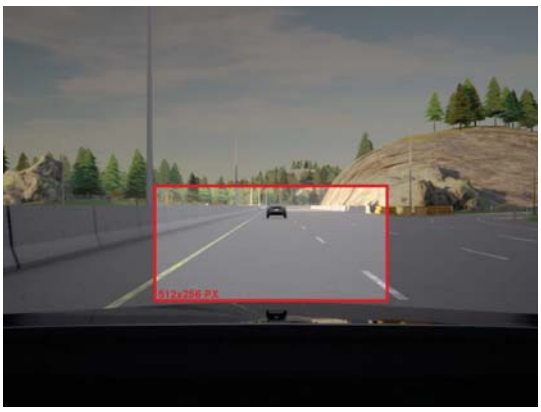

Fig. 2 Example of a RGB frame taken from the simulator. The red rectangle represents the actual image that the neural network uses after being cropped to give it the required size. Note how the input frame is located right over the vehicle's bonnet

- input_imgs: It is defined as a float32[1, 12, 128, 256] array containing the information of 2 consecutive 512x256 YUV420 video frames. In this array, each YUV420 image is represented by a float32[1, 6, 128, 256], where the 4 first sub arrays are the Y component, and the fifth and sixth represent the U and V component, respectively. The model benefits from having 2 consecutive images so that it can have a perception of the motion and displacement made by the objects in its environment between the elapsed time taken between the frames. Fig. 2 shows an example of a 512x256 image taken from CARLA used as input for the model.
- desire: It is defined as a float32[1, 8] array that represents the current intention of the model to perform a lane change. This array is modified whenever the driver voluntarily triggers the blinkers to change the lane. All the values remain equal to 0 except for the action to take, which varies from staying in the lane, turning left or right, changing either to the left or right lane, or keeping to the left or right. Since the array is modified with a rising edge pulse, the default value is "Stay on the lane" when no manual input from the driver is detected.
- traffic_convention: It is defined as a float32[1, 2] array containing the two possible traffic conventions: left-handed driving or right-handed driving.
- initial_state: It is defined as a float32[1, 512] array containing information related to the previous iteration of the model. Its value is calculated by the GRU, which decides what information is useful to keep for further iterations and what to forget.

C. Model Output

The model outputs a concatenated float32[1, 11327] array, containing a 10815-long main prediction block array extracted from the combination of each individual prediction block, as well as a 512-long array obtained from the GRU. The main prediction blocks whose accuracy and performance have been assessed in this research (marked in Fig. 1 with a star) are described below.

- Output 1: *Trajectory*. Represents the most complex output, following a Multiple Hypothesis Prediction [9] (MHP) technique that provides 5 different possible solutions describing the trajectory to carry out by the *Ego*. Each of the 5 solutions comprises 1981 values, where the last one represents the categorical cross-entropy loss of each solution, indicating how much they match the driving policy that the model has been trained with, and therefore, how likely it is to be the best among the predictions. This categorical cross-entropy loss value is the criterion by which the prediction is chosen. The rest of the values are divided into two groups with the same number of values (990), indicating the value and its standard deviation of the speed, acceleration, orientation rate and location of the *Ego* for X metres ahead. The value of X will depend on how confident the model is about the context layout (i.e., road, lane lines, *Ego* speed, etc.).
- Output 2: *Vehicle detection*. Represents the information about the leading vehicle in front of the *Ego*. This output benefits from a MHP technique, where again, only 1 of the 5 solutions will be chosen to determine the location and dynamics of one potential leading vehicle. The output predicts the leading vehicle's relative speed, acceleration and relative position to the *Ego*, as well as the standard deviation of each of those values. This data field also determines the probability that there is a leading vehicle in the exact same moment that the input frame shows, and also 2 and 4 seconds after.

D. Model Training

The model was trained with real driving video footage that represent the ideal human behaviour (ground truth) in combination with simulated data from data-augmented images. A small part of the training data set is publicly available at [10].

IV. Our Evaluation

A. Testbed Description

All the tests in this research were conducted in a computer with the following specifications: Intel(R) Xeon(R) CPU, 32 GiB DDR4 RAM, and a Nvidia Quadro M4000 GPU. The simulator used for creating the testing environments, including collecting the ground truth, is CARLA version 9.11 using Python 3.8.10. The AI model used belongs to version 8.2. Nvidia drivers used are version 460, along with CuDNN version 8.0.4 and CUDA version 11.0.

In both testing scenarios, first a 1164x874 video at 20Hz was generated from the vehicle's perspective in CARLA. To maximize the degree of realism, the position and orientation of the dashboard camera were manually modified so that it resembled the perspective that the model usually has when installed in a vehicle. For this purpose, the dimensions of a Toyota Prius were used. The generated video was later cropped

Fig. 3 Testing scenario 1: The red line expresses the trajectory followed by the vehicle in the circuit

and converted into a set of 512x256 YUV420 images to be used as the input *input_imgs* to recreate the model prediction. The input *desire* is configured so the model considers that no desire to perform a lane change is present, and therefore, to stay constantly on its lane. The input *traffic_convention* is configured so the model considers that the traffic is left-handed, i.e., the European convention. Lastly, the input *initial_state* is always represented by the *initial_state* output obtained from the model's last iteration. This means that the prediction series's length is always N-1, being N the number of steps carried out by the simulation. It is important to mention that the average elapsed time taken by the model to perform a single iteration of the prediction loop in both scenarios was 9.0682 ms.

B. Testing Scenario 1: Ego Dynamics Perception

In this scenario, the objective is to assess the accuracy of the *Ego trajectory* output described in Section III-C - *Output 1* when the model is exposed to a common real scenario. Fig. 3 depicts the trajectory. The vehicle travelled alone in a road with some curves, increasing progressively its speed until it reaches a predefined limit of 22 m/s (marked with the label A in Fig. 3). After this threshold is reached, the vehicle starts progressively decelerating. The different curves present in the circuit were chosen on purpose to check the capabilities of the model when it comes to measure the current travel speed and acceleration of the *Ego*, and its orientation rate when it performs such turns. Fig. 4 shows an analysis of the prediction results against the real values tested in scenario 1.

As shown in graph A, the model could predict well (>95% of accuracy) the speed of the *Ego* throughout the simulation, even when the vehicle started decelerating. However, at iteration 985, when the car performed a very sharp turn (visible in Fig. 3 with the label B), the accuracy was reduced up to a 80%. Then, after 20 measurements, the model re-adapts to the context. The series achieved a Mean Square Error (MSE) of 2.0304. Since both CARLA and the AI model predict the *Ego* speed in a 3D coordinate array, the calculation of the speed has been done following this equation 1, where variables x, y and z represent the 3D coordinates in the simulated environment:

$$speed = \sqrt{x^2 + y^2 + z^2} \tag{1}$$

In graph B, the model closely matched the prediction, achieving 0.0057 MSE. The biggest inaccuracy happened again at iteration 985, where the vehicle achieved an angular speed of -1 rad/s in the z axis (yaw), and the model could only predict -0.45 rad/s. This was likely to be caused by the model not being properly trained to perform sharp turns.

Graph C shows a comparison the vehicle's lateral and longitudinal acceleration. Similarly to graph B, the prediction of the lateral acceleration is done with a high percentage

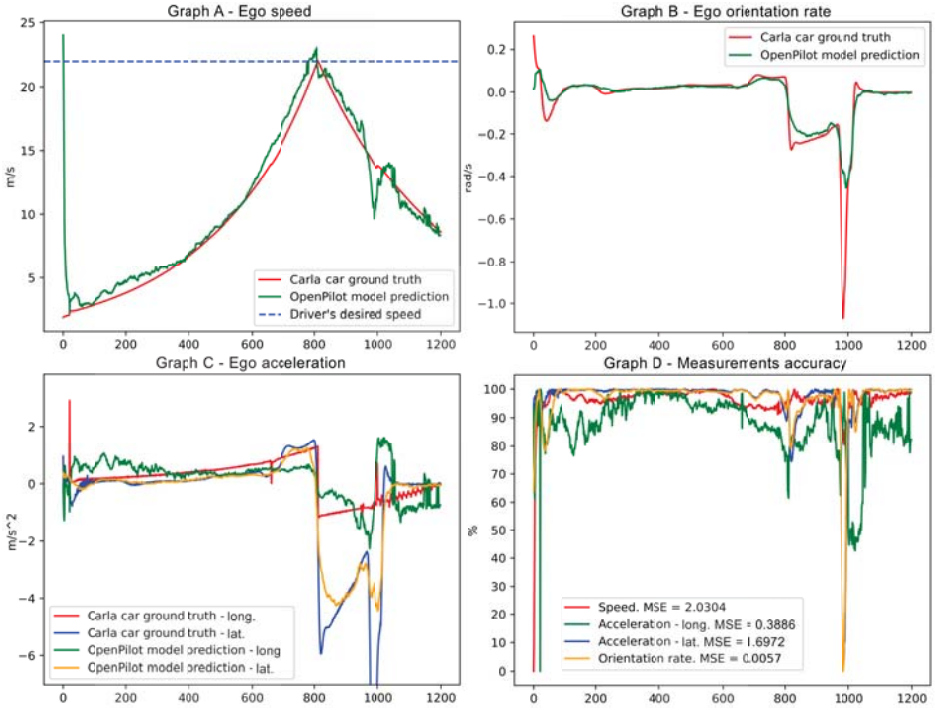

Fig. 4 Testing scenario 1 results. Note that in all the sub-graphs, the x axis represents the iteration number of the simulation and the model execution when the measurement was done. In the simulation, the desired speed was reached at iteration 815

of accuracy throughout the entire measurement, achieving a MSE of 1.6972, although this value is spoiled by the same phenomena mentioned in graph B. Clearly, the model cannot notice a lower acceleration of -4.5 m/s^2 nor a lower orientation speed of -0.45 rad/s. As for the longitudinal acceleration, a higher error rate can be observed with a MSE of 0.3886. In this case, the model could not predict as good as with the lateral acceleration, even at the early stage of the simulation, where the vehicle mainly drove straight. In addition, when the vehicle reached the desired speed at iteration 815, the model also noticed a deceleration, although much less accurately. Please note that the graph's Y axis has been limited to improve its readability, since the ground truth for the lateral acceleration reached a value of -15.3 ms^2.

Lastly, graph D depicts the accuracy rate of the measurements throughout the simulation. The four measurements explained have a lower accuracy rate around iteration 985, which corresponds to the sharp turn seen in Fig. 3 B. The model performs noticeably worse at sharp turns, but has a good accuracy rate when is exposed to more common scenarios, such as straight roads or slightly sharp curves, especially for the speed, lateral acceleration and orientation speed. All the measurements, except the lateral acceleration, have a much lower accuracy rate at the very first iterations, for instance, as clearly seen in graph A, where the initial ground truth value was 1 m/s^2, but the model predicted 24 m/s^2 (86.4 Km/h), which means that the model is initially biased to this speed. This phenomena happened in the rest of the measurements approximately from iteration 1 to 21. Since both the model and the simulation worked at 20 Hz, this means that the model takes 1 s to adapt to the context from when the prediction loop starts. The results outline that there is no correlation between the vehicle's speed and the model accuracy, but rather the sharpness of the turn taken.

C. Testing Scenario 2: Leading Vehicle Dynamics Perception

Scenario 2 assesses the accuracy of the prediction of the *Vehicle detection* output described in Section III-C - *Output 2*, with 2 vehicles travelling on a straight road following the same trajectory. The first vehicle simulates the automated vehicle, whereas the second is to be detected to know the distance between them, its acceleration and its relative speed compared with the first one. Both vehicles start 173.42 metres far apart travelling at 2 m/s (Fig. 5 A), and accelerate progressively. The first vehicle constantly increases its speed at a 0.29% rate in each iteration, and so the second does at a 0.16% rate. This scenario evaluates the ability of the model to measure the dynamics of the leading vehicle. Fig. 6 shows the prediction results. Note that the distance between vehicles has been measured from the front end of the first vehicle to the rear end of the second one.

Graph A shows the correlation between the predicted distance and the ground truth. The graph clearly showcases the bad prediction of the model from iteration no. 0 to 625. From this moment, the model drastically improves the prediction, coinciding with a 75% of confidence about the presence of a

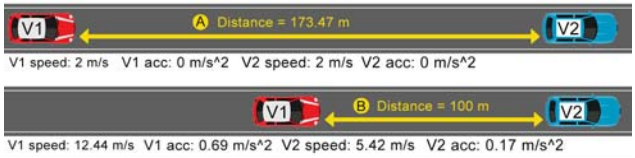

Fig. 5 Testing scenario 2. Label A indicates the initial layout and configuration of both vehicles at the beginning of the simulation; label B indicates the moment when the model starts being confident about the present of a leading vehicle

leading vehicle, visible in Graph D. This can possibly mean that the model starts being aware of the presence of a leading vehicle when it is no further than 100 m ahead. This condition of both vehicles located at 100 meters away is depicted in Fig. 5 B. This lack of accuracy in longer distances led to achieving a MSE of 9945.3215. Although this value is mostly spoiled by the big inaccuracy of the previous iterations, where the vehicle was not certain about the presence of a leading car.

In graph B, even when the model had a confidence higher than 75% (from iteration 625), the prediction was not accurate. The best prediction happened at iteration 664, achieving 94% accuracy. But this peak was not followed by the next iterations, when the accuracy fell to 48% at iteration 741. Although the prediction tends to have a downwards slope similar to the ground truth, the values are not close, with a MSE of 27.7193. Thus, there is a significant room for improvement in the prediction of the relative speed between vehicles.

Graph C depicts the evolution of the leading vehicle's acceleration in the longitudinal axis. The leading vehicle is slower than the *Ego* despite of both being accelerating. Although the prediction shows that the leading vehicle has an increasingly positive acceleration, the values are not very accurate after the 75% confidence barrier is passed. Nevertheless, the acceleration prediction achieved the lowest MSE of 0.0983. Similarly as happened in scenario 1, the model seems to be biased in the first iterations towards a specific result for each prediction value. The prediction did not yield useful information unless the confidence of the presence of a leading vehicle was above 75%. In fact, this can be also seen in OpenPilot *radarState* source code, where a minimum lead confidence of 50% is needed to consider the prediction as valid. Hence, the prediction are much better when the leading vehicle is at least located 100 metres ahead. Again, the results indicate no clear correlation between the *Ego* speed and the model accuracy.

Seemly as in the previous scenario, graph D showcases the accuracy of the predictions carried out by the AI model. It also includes the confidence of the model when the prediction was made, associated to every prediction iteration. As mentioned above, there is an slight improvement in the speed and even more noticeable in the distance prediction accuracy as the model prediction confidence increases.

V. Conclusions

This research has assessed the accuracy and performance of a mainstream automated driving AI model in a simulated environment using CARLA. The tests have been useful to

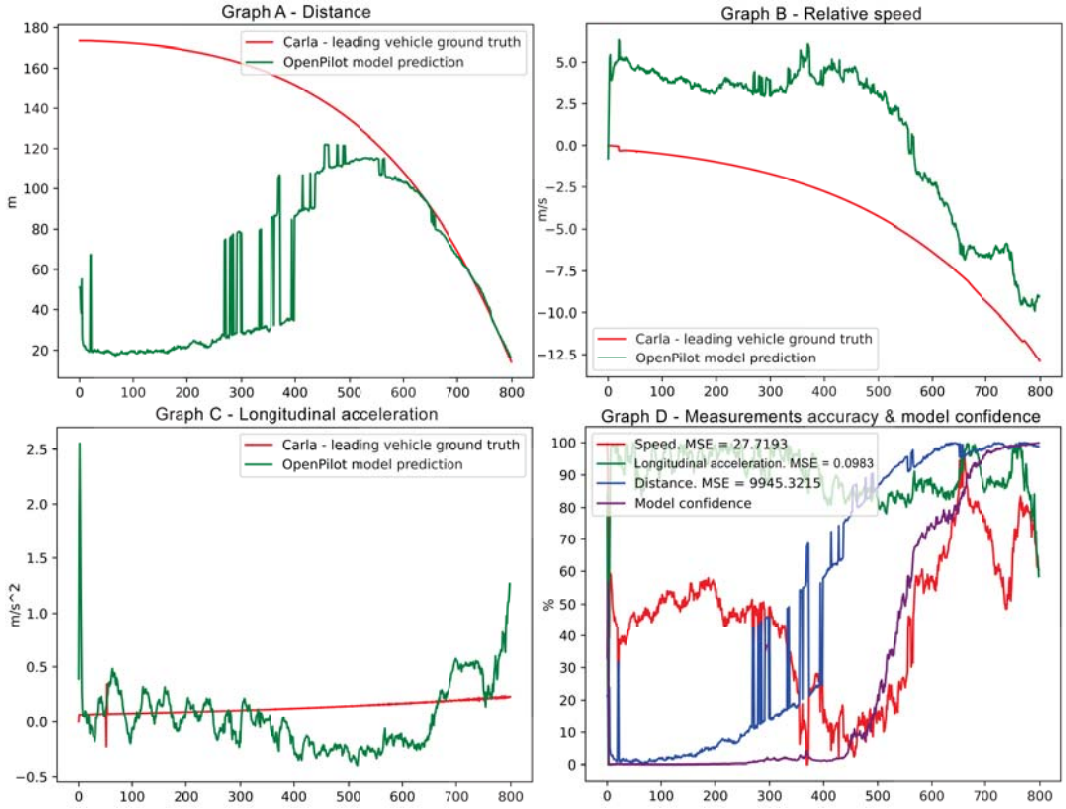

Fig. 6 Testing scenario 2 results. The x-axis of the sub-graphs represent the simulation iteration number and the model execution

determine the degree of accuracy of the model, evaluating the its perception of the *Ego* dynamics, and the leading vehicle's dynamics. The first test has showed that the model could predict very accurately the values for the speed, the orientation rate and the acceleration in the lateral axis when exposed to a common driving condition. However, longitudinal acceleration did not seem to be so accurate. In general, the model provided accurate results (summarized in Table II) in common scenarios such as straight and not too sharp curves (label A), and a worse accuracy and adaptation time in very sharp curves (label B). The second scenario proved that the model can potentially have an accurate perception of the dynamics of the leading vehicle when it is closer than 100 metres ahead (label B). However, the acceleration and the relative speed of the leading vehicle were not so accurate. A very revealing conclusion is that the model is initially biased towards the values that the model was trained with, and that it takes 20 iterations (1 s) to adapt to the context. Although the model can perform with very satisfying execution times (<10ms), it is not totally safe, according to a simple interpretation of MSE scores obtained, to be as a single source of information when planning the behavior of an automated vehicle. Therefore, the inclusion of other sensors such as RADARs or IMUs to enhance its reliability should be complementary to have a more accurate perception the of the *Ego* context.

TABLE II
RESULTS SUMMARY

	Output	Average accuracy (%)	
		Label A (iters. 790 to 810)	Label B (iters. 975 to 995)
Testing scenario 1	Speed	94.81	88.04
	Orientation rate	95.11	30.08
	Lateral acceleration	92.02	28.00
	Longitudinal acceleration	76.70	80.31
	Output	Average accuracy (%)	
		Label A (iters. 1 to 20)	Label B (iters. 615 to 635)
Testing scenario 2	Relative speed	52.67	70.96
	Longitudinal acceleration	82.50	84.33
	Distance	7.56	99.62
	Model confidence*	5.52	77.67

The results of the model confidence do not refer to accuracy rate but rather the confidence about the presence of a leading vehicle.
The table is separated by scenarios, assessment parameters, and simulation moments. The labels refer to a specific relevant moment in each testing scenario, and both are defined in Figs. 3 and 5.

ACKNOWLEDGMENT

This work is funded by the European Commission H2020 5G-PPP Program under Grant Agreement Number H2020-ICT-2020-2/101016941 "5G INDUCE: Open Cooperative 5G Experimentation Platforms for The Industrial Sector NetApps".

REFERENCES

[1] S. SAE, "J3016 standard: Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," 2021.
[2] R. Lanctot *et al.*, "Accelerating the future: The economic impact of the emerging passenger economy," *Strategy analytics*, vol. 5, 2017.
[3] A. Toschi *et al.*, "Characterizing perception module performance and robustness in production-scale autonomous driving system," in *IFIP ICNPC*, 2019, pp. 235–247.
[4] A. H. M. Rubaiyat *et al.*, "Experimental resilience assessment of an open-source driving agent," in *2018 IEEE PRDC*, 2018, pp. 54–63.
[5] J. Norden *et al.*, "Efficient black-box assessment of autonomous vehicle safety," *arXiv preprint arXiv:1912.03618*, 2019.
[6] T. Sato *et al.*, "Hold tight and never let go: Security of deep learning based automated lane centering under physical-world attack," *arXiv preprint arXiv:2009.06701*, 2020.
[7] Z. Zhong *et al.*, "Detecting safety problems of multi-sensor fusion in autonomous driving," *arXiv preprint arXiv:2109.06404*, 2021.
[8] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*, 2019, pp. 6105–6114.
[9] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2090–2096.
[10] "Comma.ai - comma2k19," https://github.com/commaai/comma2k19, accessed: 06/08/2021.