sensors

MDPI

Article

Simultaneous Estimation of Vehicle Roll and Sideslip Angles through a Deep Learning Approach

Lisardo Prieto González [1,*], Susana Sanz Sánchez [2], Javier Garcia-Guzman [1], María Jesús L. Boada [2] and Beatriz L. Boada [2]

[1] Computer Science Department, Institute for Automotive Vehicle Safety (ISVA), Universidad Carlos III de Madrid, Avda. de la Universidad 30, 28911 Leganés, Madrid, Spain; jgarciag@inf.uc3m.es

[2] Mechanical Engineering Department, Institute for Automotive Vehicle Safety (ISVA), Universidad Carlos III de Madrid, Avda. de la Universidad 30, 28911 Leganés, Madrid, Spain; ssanz@ing.uc3m.es (S.S.S.); mjboada@ing.uc3m.es (M.J.L.B.); bboada@ing.uc3m.es (B.L.B.)

* Correspondence: lpgonzal@inf.uc3m.es; Tel.: +34-91-624-5962

Received: 1 June 2020; Accepted: 29 June 2020; Published: 30 June 2020

Abstract: Presently, autonomous vehicles are on the rise and are expected to be on the roads in the coming years. In this sense, it becomes necessary to have adequate knowledge about its states to design controllers capable of providing adequate performance in all driving scenarios. Sideslip and roll angles are critical parameters in vehicular lateral stability. The later has a high impact on vehicles with an elevated center of gravity, such as trucks, buses, and industrial vehicles, among others, as they are prone to rollover. Due to the high cost of the current sensors used to measure these angles directly, much of the research is focused on estimating them. One of the drawbacks is that vehicles are strong non-linear systems that require specific methods able to tackle this feature. The evolution in Artificial Intelligence models, such as the complex Artificial Neural Network architectures that compose the Deep Learning paradigm, has shown to provide excellent performance for complex and non-linear control problems. In this paper, the authors propose an inexpensive but powerful model based on Deep Learning to estimate the roll and sideslip angles simultaneously in mass production vehicles. The model uses input signals which can be obtained directly from onboard vehicle sensors such as the longitudinal and lateral accelerations, steering angle and roll and yaw rates. The model was trained using hundreds of thousands of data provided by Trucksim® and validated using data captured from real driving maneuvers using a calibrated ground truth device such as VBOX3i dual-antenna GPS from Racelogic®. The use of both Trucksim® software and the VBOX measuring equipment is recognized and widely used in the automotive sector, providing robust data for the research shown in this article.

Keywords: sensor fusion; deep Learning based estimator; vehicle dynamics; roll angle; sideslip angle

1. Introduction

Road vehicles are the most prevalent transportation system. Current estimations set the mortality due to traffic accidents and collisions in around 1.35 million people per year [1]. Over the last several decades, vehicles have been equipped with active systems such as ABS (Anti-Blocking System), ESC (Electronic Stability Controllers), and active suspensions, which improve their comfort, efficiency, and safety. These systems gain greater interest in autonomous vehicles. The primary cause of accidents is related to loss of lateral stability control [2]. In this regard, sideslip and roll angles are critical parameters in vehicular lateral stability. It has become necessary to know about them to design controllers capable of providing adequate performance in all driving scenarios. Specifically,

the roll angle has a high impact on vehicles with an elevated center of gravity, such as trucks, buses, and industrial vehicles, among others, as they are prone to rollover. Presently, there exist devices that allow the direct measurement of both angles, such as a dual antenna GPS (Global Positioning System) or a Kistler S-Motion device [3]. This type of advanced GPS equipment cannot be included in mass production vehicles due to high-cost issues [4]. Because of that, many of the researchers focus on estimating these angles based on available measurements from on-board embedded sensors but independently [5,6]. On the other hand, in works such as [7], both roll and sideslip angles are estimated by using additional external sensors that need to be integrated into the vehicle instead of using the own on-board systems. In general, these approaches are based on vehicle model-based estimation and data-driven-based estimation [8]. The former group includes approaches such as Kalman-filter-based method [9–11], nonlinear-observer-based method [11–13], and robust observers [14–16]. The latter includes Neural Networks, and ANFIS, among others [4,17,18], whose main advantage is that they do not depend on the reference vehicle models. Additionally, vehicles are strong non-linear systems that require specific methods able to tackle this feature. Presently, Machine Learning technologies are gaining attention in the field of improved vehicle driving, providing a diverse application set, ranging from computer vision for the identification of static or dynamic obstacles or the detection of fatigue situations in drivers to vehicle trajectory prediction [19–22]. Deep learning can be defined as a set of machine learning algorithms that implement large neural networks, including many hidden layers (also referred to as Deep Neural Networks—DNNs) for feature generation, learning, classification, and prediction [23]. There are several model architectures and transfer learning techniques that can be applied to solve different problems in the automotive domain. Nevertheless, more research is needed:

(a) to design distributed deep learning systems to improve training times for more complex networks and massive data sets;
(b) to determine how to apply deep learning in other areas of automobile control such as lateral stability [24]; and
(c) it is necessary to assess that the fusion of data coming from low-cost devices and estimations provided by deep machine learning algorithms can fulfill the reliability and appropriateness requirements for using these technologies to improve overall vehicular safety.

The novelty of this work consists of the design and implementation of an efficient and precise Deep Neural Network to simultaneously estimate roll and sideslip angles by using only information provisioned by on-board sensors such as the IMU (Inertial Measurement Unit) and the steering angle sensor. The designed DNN can tackle strong non-linear vehicle behavior. Besides, the proposed DNN does not use previous information from the sensors. Just the one provided at that time, so there are no stability problems associated with an accumulated error.

This article is organized as follows. Section 2 presents the steps taken to define the Deep Learning model for this research work, and the designed Deep Neural Network that solves the problem. Section 3 shows the results as far as predicted precision is concerned. Finally, a discussion related to the results, a set of conclusions, and the next steps to be taken by our research team are introduced in Section 4.

2. Methodology

This section describes the experimental approach adopted to achieve the goals stated for this research work. Showing up next is an overview of the different steps and required aspects before defining a proper Deep Neural Network. Then, different subsections detail each of the main aspects considered in the model creation.

The main components that define our solutions are presented in Figure 1. They can be summarized as:

- Data set with repeatable simulated maneuvers with a complex vehicular model (van).
- Data set with information logged from real driving scenarios.
- Deep Learning predictor model, tested against the first data set.

- Validation of the Deep Learning Network using the second data set.

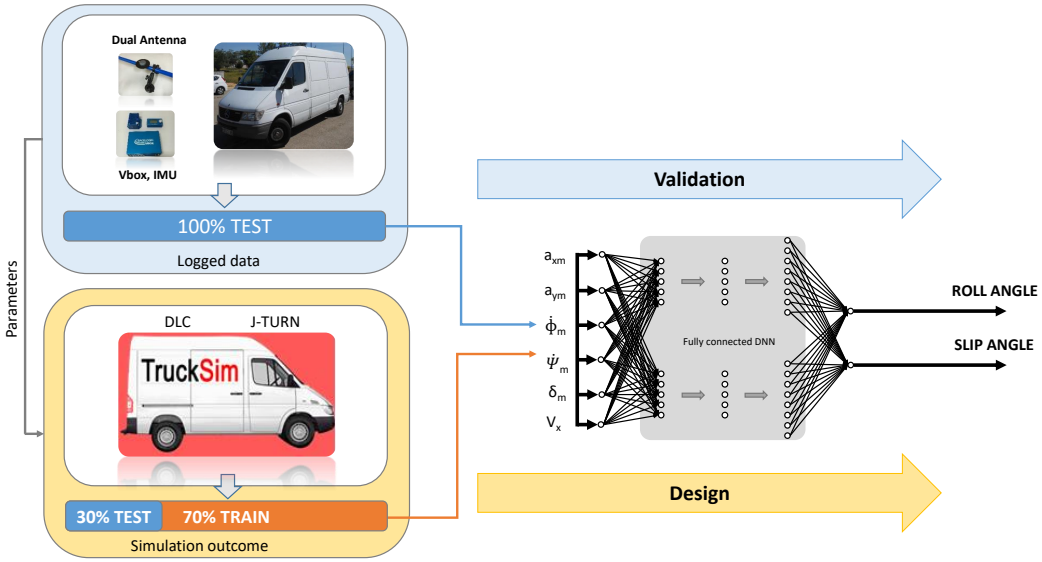


Figure 1. Training and validation scheme.

2.1. General Scheme

2.1.1. Deep Learning Model to Accurately Predict Roll and Sideslip Angles

Presently, there exist a wide variety of deep neural networks (DNNs) and related architectures. For instance, recurrent neural networks (RNNs) [25] can be used in tasks as natural language processing (NLP), connected handwriting, speech recognition, or the generation of new sentences and document summaries [26]. Alternatively, long short-term memory (LSTM), initially designed to model temporal sequences considering their respective long-range dependencies with higher accuracy than conventional RNNs. LSTMs allow solving problems such as the creation of outstanding acoustic models for complex languages, and the tagging of parts of the speech with high precision [27–30], among others. Given the nature of the problem, a multilayer perceptron (MLP) has been used in this work. A beneficial feature in the MLP model used, compared to other ANN architectures such as the RNN and LSTM, is that there is no accumulated error derived from the estimation because MLP does not use the outcome predictions to feed the ANN. For a long time and until now, MLPs have been used effectively in several works to perform accurate predictions [31–33]. A multilayer perceptron contains three or more layers that use a nonlinear activation function (usually hyperbolic tangent or logistic function), which allow classifying data that is not linearly separable. Each node in a layer is connected to every node in the following layer, making this kind of ANNs to be fully connected. According to [23], the practical design process should be structured as follows:

- Determine the goals, including what error metric to use, and the expected value for this metric. The problem should drive these goals and error metrics that the application has to solve.
- Establish an end-to-end working pipeline as soon as possible, including the estimation of the appropriate performance metrics.
- Instrument the system adequately to track bottlenecks or issues in the model, such as overfitting, underfitting, or defective data.
- Perform repeatedly incremental changes such as increasing the data set entries, adjusting hyperparameters, or changing algorithms, based on specific findings from the previous instrumentation.

Since the purpose of the work is to create an estimator for roll and sideslip angles in mass production vehicles, the priority is to provide the best forecasts as possible in the least amount of time. As the system is embedded in real vehicles through specific computing units endowed with limited computing capabilities, there must be a balance between the ANN complexity and the prediction accuracy. As error metrics, there has been considered the RMSE (Root-Mean-Square Error) since they allow considering the magnitude of the error to assign a higher loss to larger errors than other metrics (as Mean Absolute Error) do. A large prediction error could result in a dangerous situation, so they need to be minimized.

The inputs handled to predict the roll (ϕ_m) and sideslip (Ψ_m) angles (model outcome) are: longitudinal acceleration (a_{xm}), lateral acceleration (a_{ym}), roll rate ($\dot{\phi}_m$), yaw rate ($\dot{\Psi}_m$), steering angle (δ_v), and longitudinal speed (V_x). These are the typical variables that affect the lateral vehicle dynamics. Notation used in this paper in relation to the DNN model can be seen in Figure 2.

	Symbol	Meaning
Sizes	L	Number of hidden layers in the Deep Neural Network
	$n^{[l]}$	Number of units in layer "l"
	m	Number of examples in the dataset
	n_x	Input size
	n_y	Output size
Objects	X	Input matrix, $\in \mathbb{R}^{n_x \times m}$
	$x^{(i)}$	i^{th} input example represented as a column vector, $\in \mathbb{R}^{n_x}$
	Y	Expected outputs matrix, $\in \mathbb{R}^{n_y \times m}$
	$y^{(i)}$	Expected output for the i^{th} example, $\in \mathbb{R}^{n_y}$
	$W^{[l]}$	Weight matrix for layer "l"
	$b^{[l]}$	Bias vector for layer "l"
	$\hat{y} = A^{[L]}$	Predicted output vector. Also denoted as activation vector for the output layer, $\in \mathbb{R}^{n_y}$
	$A^{[l]}$	Activation vector for layer "l"
	$\mathcal{J}$	Cross-entropy cost

Figure 2. Notation.

Different factors affect the goodness of a DNN model [23]. The number of data for training and testing the model, the hyperparameters (i.e., learning rate or the number of training iterations) configuration, the number of hidden layers, and the number of units per layer. The original DNN was coded from scratch in Python 3.7 [34] due to the robust set of existing libraries oriented towards matrix processing (NumPy [35]), data processing (pandas [36]) and resulted plotting (matplotlib [37]), plus the ability to run the code in different platforms (from development desktop computers to production embedded devices). The "from scratch" approach was taken to optimize and instrument the code as much as needed. However, to train and test the designed DNN architecture efficiently, the experimental version that led to a proper set of hyperparameters and neuron weights was implemented using Keras [38]. This framework was developed with a focus on enabling fast experimentation (transform the idea into a result with the least possible delay).

Several set-ups related to both parameters and hyperparameters were defined and tested. The best configuration found is represented in Figure 3. This DNN is composed of five hidden layers, six inputs, and two outputs. The network presents thirty units in the first layer, sixty in the second layer, ninety in the third layer, one hundred and eighty in the fourth layer, and finally ninety in the fifth layer. This network configuration was the result of using the conventional method of trial and error.

The network was trained using different hyperparameters. Some techniques, as "early stopping" by using a relatively small number of iterations to prevent overfitting, were tested until getting a proper set of hyperparameters. First layer and subsequent ones calculate a vectorized linear function (see Equation (1)) followed by a vectorized activation. The activation function, $\pi(\cdot)$, for these layers is a RELU (Rectified Linear Unit—see Equation (2)), and the output layer outcome (see Equation (3)) is the result of the linear function, providing a linear regression, suitable for predicting the expected values.

$$A^{[l]} = \pi(Z^{[l]}) = \pi(W^{[l]}A^{[l-1]} + b^{[l]}) \tag{1}$$

$$RELU(Z^{[l]}) = max(0, Z^{[l]}) \tag{2}$$

$$\hat{y} = A^{[L]} = Z^{[L]} = W^{[L]}A^{[L-1]} + b^{[L]} \tag{3}$$

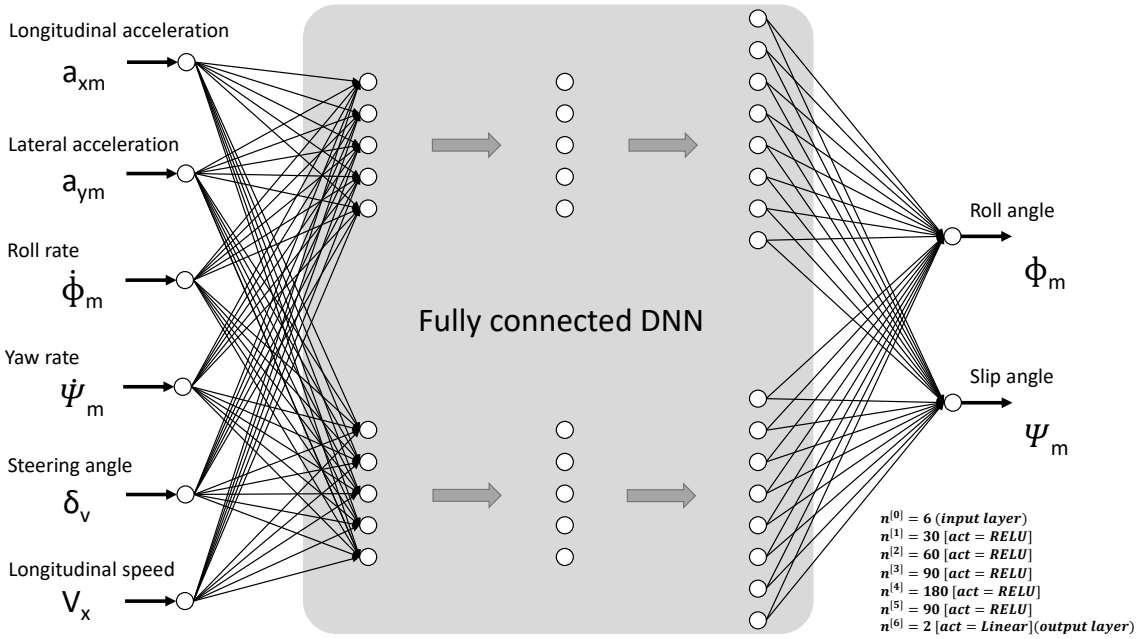


Figure 3. Deep Neural Network configuration used.

Once all the layers perform their computations, it is necessary to compute the cost in order to check if the model is learning correctly. The cross-entropy cost $\mathcal{J}$ (see Equation (4)) was used for this purpose in order to compare the predicted vs. the expected values:

$$\mathcal{J} = -\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}\log\left(a^{[L](i)}\right) + (1-y^{(i)})\log\left(1-a^{[L](i)}\right)) \tag{4}$$

After computing the cost, a linear backward (backpropagation) has to be carried in order to update network parameters. To do so, it is required to calculate the gradients: $dW^{[l]}$ (Equation (5)), $db^{[l]}$ (Equation (6)) and $dA^{[l-1]}$ (Equation (7)), respectively:

$$dW^{[l]} = \frac{\partial \mathcal{J}}{\partial W^{[l]}} = \frac{1}{m}dZ^{[l]}A^{[l-1]T} \tag{5}$$

$$db^{[l]} = \frac{\partial \mathcal{J}}{\partial b^{[l]}} = \frac{1}{m}\sum_{i=1}^{m}dZ^{[l](i)} \tag{6}$$

$$dA^{[l-1]} = \frac{\partial \mathcal{J}}{\partial A^{[l-1]}} = W^{[l]T} dZ^{[l]} \tag{7}$$

The linear-activation backward function will be then computed by using the next Equation (8), being $\pi(\cdot)$ the activation function for RELU and $\pi'(\cdot)$ its derivative:

$$dZ^{[l]} = dA^{[l]} * \pi'(Z^{[l]}) \tag{8}$$

For the last layer, the MSE derivative between predicted and expected normalized values is computed and backpropagated as the first gradient in such a process.

In the last step, when reaching the input layer of the model, it is necessary to update the parameters using gradient descent on every $W^{[l]}$ and $b^{[l]}$ for $l = 1, 2, \ldots, L$. The equations to do so are:

$$W^{[l]} = W^{[l]} - \alpha \, dW^{[l]} \tag{9}$$

$$b^{[l]} = b^{[l]} - \alpha \, db^{[l]} \tag{10}$$

where α is the learning rate hyperparameter. The whole process is repeated for the specific number of iterations defined as a hyperparameter. A graphical description of this process can be seen in Figure 4.

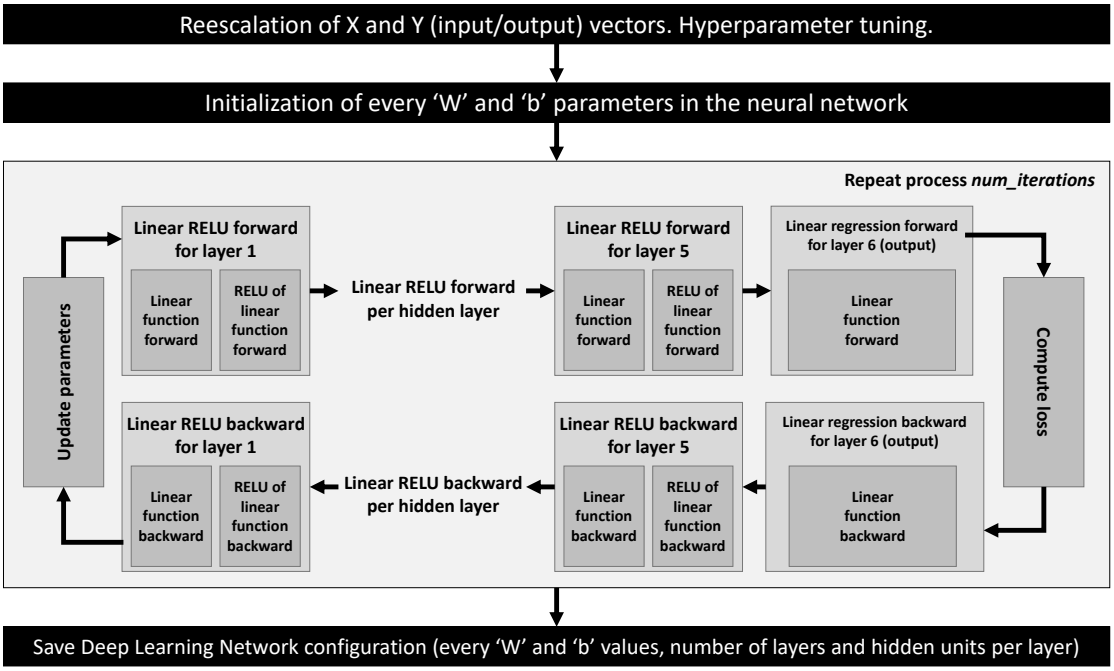


Figure 4. DNN configuration and training procedure followed.

A general overview of the approach followed to train and validate the network is presented in Figure 5. The training process is fulfilled with a significant subset of entries from Trucksim® outcome data. Then the testing process is checked against the remaining subset of entries. All the input data are normalized per component. After testing the trained model, it is required to check the predicted error rates, and in case they are not appropriate, new training has to be done after adjusting the hyperparameters (i.e., to prevent overfitting). Once the predicted error rates are adequate, the validation with data acquired from real driving maneuvers can be performed.

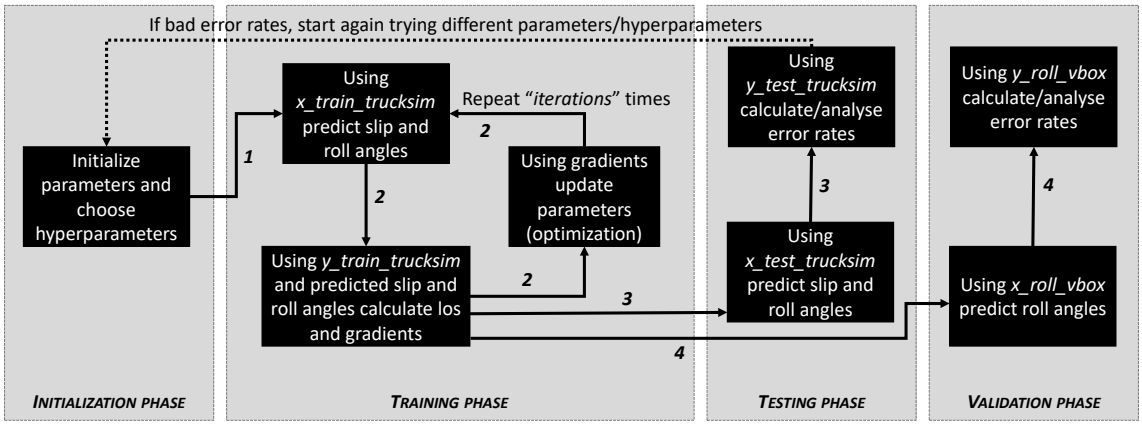


Figure 5. Deep learning approach followed.

2.1.2. Model Dissemination and Extension

The proposed model has been designed as a portable predictor, so the DNN has been implemented in a modular way and using a Python virtual environment, which allows executing the utility function ("predict") from any computer sharing the same architecture. By trying to minimize the number of external dependencies, and by creating the model from scratch, it is possible to run it from any device supporting a Python interpreter. Besides, the creation of a Docker [39] image is in progress. This image can be deployed in any computing device supporting Docker, and it can also be used not just to provide predictions, but also to continue the extension of the model in the same environment as the one used for its creation. Besides, the simplicity of the communication interface provides easy integration with other systems that may benefit from sideslip and roll angle predictions, as the perception-based knowledge model proposed in [40]. The synergy with those kinds of systems may lead to a complex representation and prediction model for vehicular dynamics at different scales, being suitable to enhance vehicular autonomy, by providing more complex and accurate predictions by fusing multiple sensors and information sources.

3. Datasets

The training dataset has to be adequately chosen to take into account all driving situations and conditions. The training dataset has been obtained from a previous experimentally validated Trucksim® van model employing the real Mercedes Sprinter depicted in Figure 6 [41]. The model parameters were adjusted by trial and error according to the differences between the experimental and simulation data. Trucksim® is a software widely recognized and validated for application in vehicle dynamics. The effectiveness of this software allows us to simulate risky maneuvers that would not be possible, for security reasons, to recreate them in reality. Besides, the convenience of the software allows us to modify the road conditions quickly, allowing a much more comprehensive range of tests when obtaining valid information. Finally, the use of simulation software guarantees the reproducibility of the tests. Trucksim® requires to model the vehicle dynamics before simulating it.

In this work, authors considered employing a van, as this type of vehicle has, on average, a higher center of gravity, which makes them more prone to loose lateral stability. The maneuvers simulated in the software are common ones in the experimentation of vehicle dynamics. Double lane change (DLC), Figure 7, JTurn (JT), both dextro-rotatory and levorotatory, Figure 8, and sine steering maneuvers have been used for the training of the proposed learning-based DNN. Each of the maneuvers has been performed in a range from 20 km/h to 120 km/h. Those tests in which the van virtually did not maintain stability have been discarded since the data is no longer valid for network training. Table 1 shows a summary of the data set used for the training and test of the DNN. More than 400,000 entries were used considering a training-testing ratio of 70:30.

Figure 6. Real Mercedes Sprinter van and instruments setup.

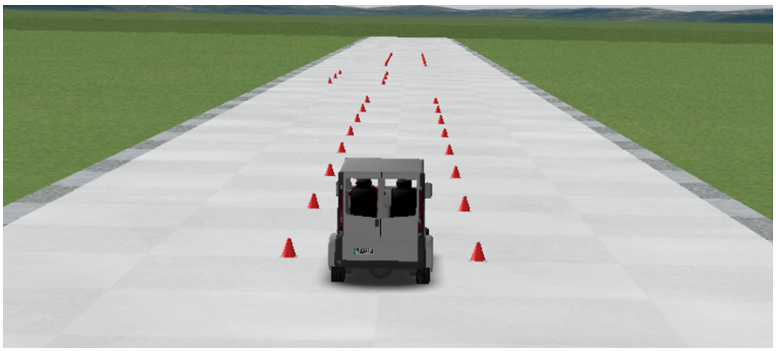


Figure 7. Simulated scenario for a Double Lane Change (DLC) maneuver in Trucksim®.

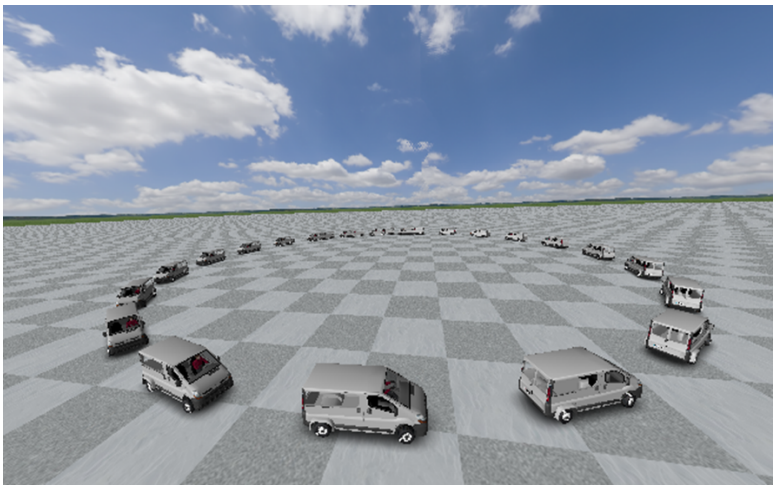


Figure 8. Simulated scenario por a Left J-turn maneuver in Trucksim®.

Table 1. Training DNN data sets (Trucksim®).

Road Friction Coefficient	Maneuver	Speed (km/h)	Steering Angle (deg)
0.3	Left DLC	20, 30, 40, 50, 60, 70	-
0.5	Left DLC	20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120	-
1	Left DLC	20, 30, 50, 60, 70, 80, 90, 100, 110, 120	-
0.3	Right DLC	20, 30, 40, 50, 60, 70	
0.5	Right DLC	20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120	-
1	Right DLC	20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120	-
0.3	Left J-Turn and Right J-turn	20, 30, 40, 50, 60, 70, 80, 90, 100, 110	40, 60, 90, 100, 120
0.5			
1			
0.5, 0.85	Sine steering	30 to 60 km/h in 30 s	±60 (0.2 Hz, 0.5 Hz)
0.5, 0.85	Sine steering	30 to 60 km/h in 30 s	±90 (0.2 Hz, 0.5 Hz)

The data from Trucksim® van physic model simulated driving maneuvers are stored in a CSV file (Comma Separated Values) format. Working directly with a CSV file in Python is not as efficient as using other formats such as Pandas DataFrames [36] or NumPy matrices [35], which also support vectorized operations and broadcasting.

A remarkable characteristic of Deep Neural Networks is that given enough data corresponding to training examples—the structured set of inputs and expected outputs; they are accurate at figuring out the functions that accurately map from inputs to expected results. In the specific case of this work, the training examples contained more than 400,000 entries, which led to an efficient and accurate predictor.

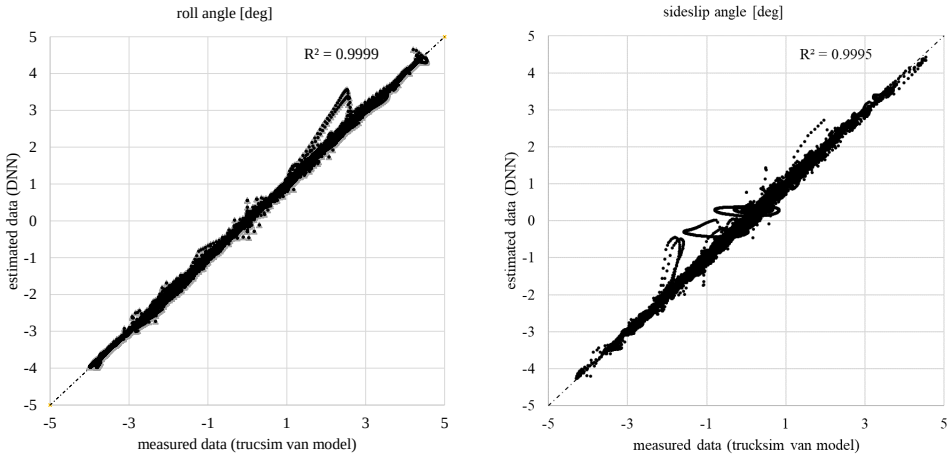
The first step taken was to convert the heterogeneous data sets from CSV into a combined and structured data set. HDF5 [42] format was used, as it is high-performance data management and storage suite that can be applied to manage, process, and store heterogeneous data. HDF5 also allows a fast I/O processing that provides fast access times and storage space optimizations, and portable storage within a self-describing file format. Besides, all data and metadata can be passed along in one file (a useful feature to preserve the data set headers, in the same way as in CSV) and without a limit on the number or size of data objects in the collection, giving great flexibility for big data. Even more, it comes as a multi-platform software library that implements a high-level API with interfaces in multiple programming languages such as Python. Before converting CSV files into HDF5 collections, they had to be parsed by a Python script explicitly made for this task. This script is not just cleaning the files discarding not used variables, but also is in charge of shuffling and splitting the Trucksim® data set into training and testing subsets.

There are as many rows as variables to learn from, and as many columns as training examples. Analogously, as many rows as expected values (two, sideslip and roll angles), and as many columns as input training examples. The testing and validation phases use the corresponding data sets stored as collections in the aforementioned HDF5 file. Finally, the trained DNN weights are also stored in a specific HDF5 file produced by Keras, which later can be loaded to perform further predictions.

4. Results and Discussion

In this section, simulation results from the learned DNN estimator using the training dataset depicted in Table 1, are firstly presented. Secondly, the proposed estimator is experimentally validated using a real van. Figure 9 shows the comparison between the roll and sideslip angles obtained by Trucksim® with those estimated by the proposed DNN for all maneuvers used during the training.

The regression plots' examination reveals that the proposed DNN learns reasonably good for all data sets, with R values close to 0.999.



(**a**) Roll angle (**b**) Sideslip angle

Figure 9. Comparison between measured data from the TruckSim® van model and the estimated data from DNN.

In addition to the graphical evidence of the effectiveness of the proposed DNN, a quantitative study bears in mind the RMS (Root Mean Square), maximum, and norm errors that have been calculated (Table 2). The norm error is obtained from the following equation [41]:

$$E_t = \frac{\varepsilon_t}{\sigma_t} \tag{11}$$

where

$$\varepsilon_t^2 = \int_0^T (\lambda_{\text{measured}} - \lambda_{\text{estimated}})^2 dt$$

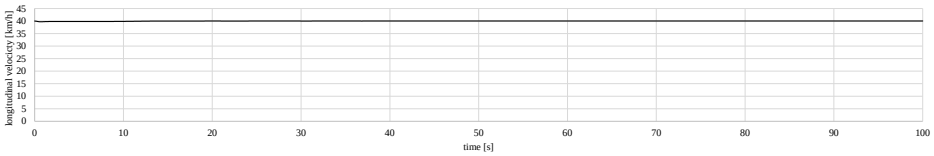
$$\sigma_t^2 = \int_0^T (\lambda_{\text{measured}} - \mu_{\text{measured}})^2 dt$$

$\lambda_{\text{measured}}$ and $\lambda_{\text{estimated}}$ are the measured and estimated roll and sideslip angles, respectively, and μ_{measured} is the mean value of the roll and sideslip angles obtained from trucksim® during the period T.

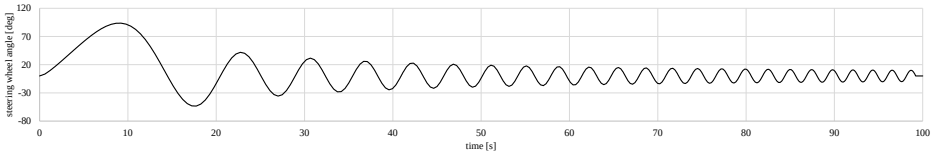
Table 2. Error measurements for the training and test dataset.

	Trucksim® van Model	
	Roll Angle	**Slip Angle**
RMSE [°]	0.018	0.033
$\mathbf{E}_{max}$ [°]	1.05	1.39
$\mathbf{E}_t$ **[-]**	8.11×10^{-5}	3.83×10^{-4}

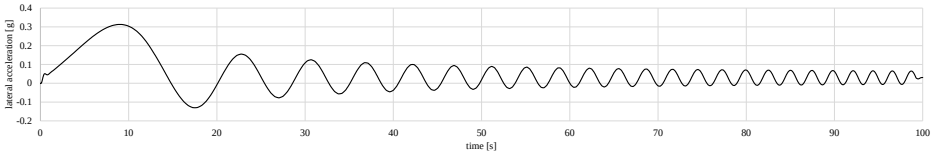
To corroborate the proposed estimator's good performance, a new dataset has been selected for validation. This dataset consists of a handling sine sweep maneuver with a steering wheel angle ranging from $\pm 90°$ to $\pm 10°$ and a frequency from 0.5 to 0.2 Hz with a friction coefficient of 0.85. The vehicle moves at a velocity of 40 km/h. The simulation results are depicted in Figure 10. Table 3 shows the RMS, maximum and norm errors. Results show that the proposed DNN model performs quite well.



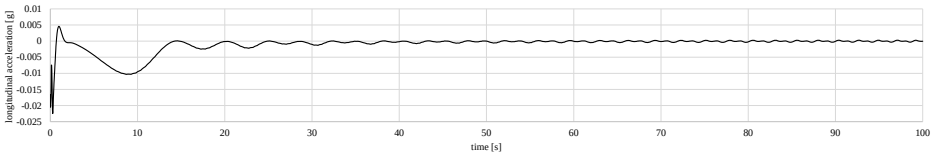
(**a**) Longitudinal velocity



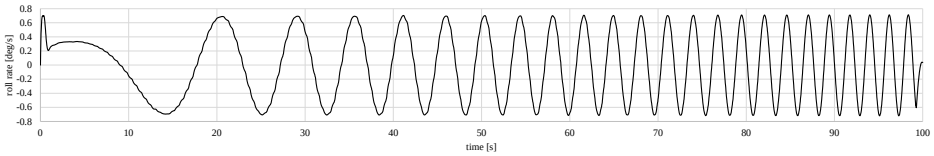
(**b**) Steering wheel angle



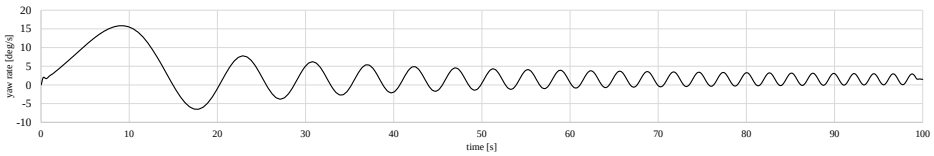
(**c**) Lateral acceleration



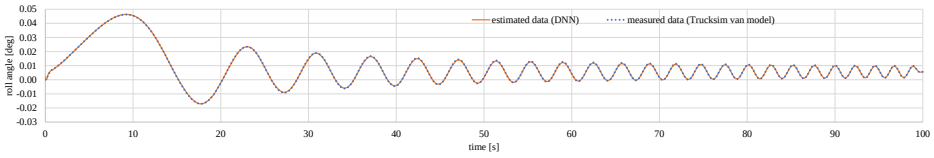
(**d**) Longitudinal acceleration



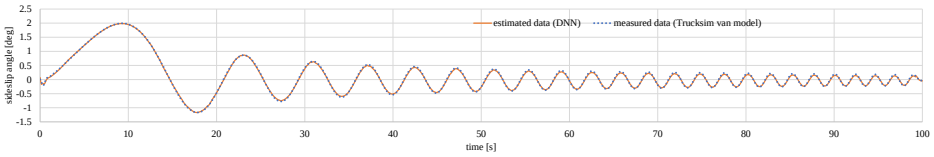
(**e**) Roll rate



(**f**) Yaw rate



(**g**) Roll angle



(**h**) Sideslip angle

Figure 10. Simulation results for the validation test.

Table 3. Error measurements for validation datasets.

	Trucksim® van Model		Real van	
	Roll Angle	**Slip Angle**	**Roll Angle**	**Slip Angle**
RMSE [°]	0.018	0.024	1.19	1.40
$\mathbf{E}_{max}$ [°]	0.11	0.094	3.5	6.36
$\mathbf{E}_t$ **[-]**	0.017	0.040	1.14	0.48

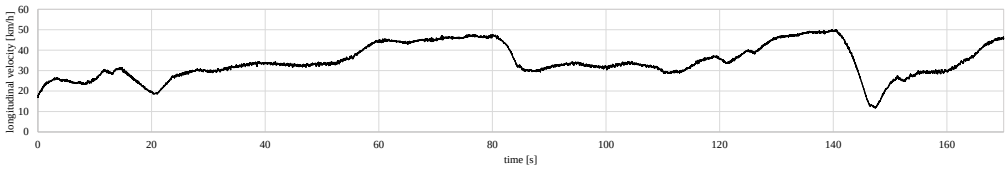
Once the proposed estimator has been trained and validated from Trucksim® van model, the real Mercedes Sprinter van (see Figure 6) has been used for its experimental validation. The studied test-bed van is composed of different experimental kits:

1. Vbox 3i dual antenna data logger from Racelogic.
2. An IMU (Inertial Measurement Unit) sensor from Racelogic mounted near the center of gravity (COG) of the vehicle to provide measurements of the roll and yaw rates and longitudinal and lateral accelerations.
3. Two GPS antennas from Racelogic to provide measurements of the roll and sideslip angles (Ground Truth). The dual antennas must be positioned transverse to the direction of movement to precisely determine the roll angle.
4. Steering angle sensor MSW 250 Nm from Kistler.

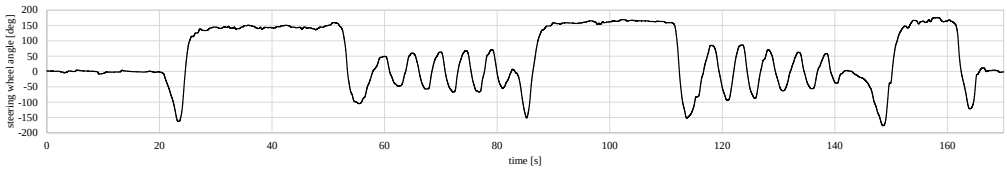
The experimental scenario and the real maneuvers performed by the van are depicted in Figure 11. The van's sequence of maneuvers is a straight line during the first 20 s, J-Turn maneuvers in the roundabouts and slalom maneuvers on the straight sections of the road. The van speed profile is shown in Figure 12.

Figure 11. Experimental scenario for the validation test.

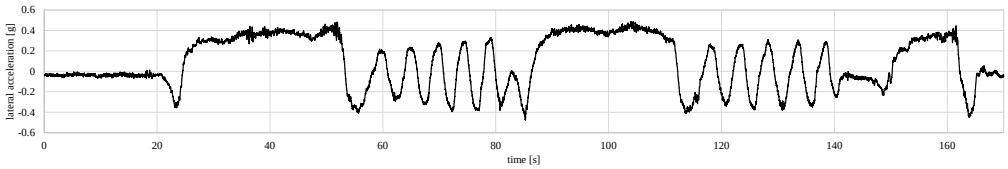
The results are depicted in Figure 12. Table 3 shows RMS, maximum, and norm errors for both roll and sideslip angle predictors using the validation data set from real maneuvers logged with a Racelogic VBOX IMU. As can be seen, the trained DNN behaves slightly better, estimating the roll angles than the sideslip angles. For a better estimation, different filters such as Kalman filter [17,41] and H_∞ [18] can be used as it has been done in previous works, to filter noise and minimize the errors' estimation. One of the primary sources of error in the proposed DNN is that road irregularities and road banks have not been considered. Although these disturbances have not been taken into account, the obtained errors in the estimation of the roll and sideslip angles are deemed acceptable. These disruptions could be estimated by designing more complex observers [43,44]. Nevertheless, this involves installing more sensors on the vehicle, increasing its cost, and the overall computing time.



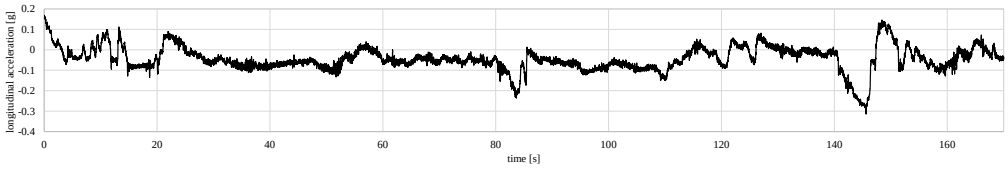
(**a**) Longitudinal velocity



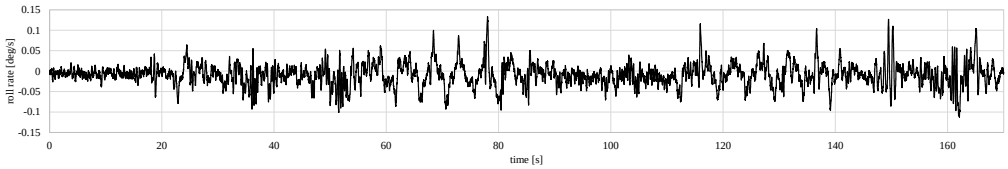
(**b**) Steering wheel angle



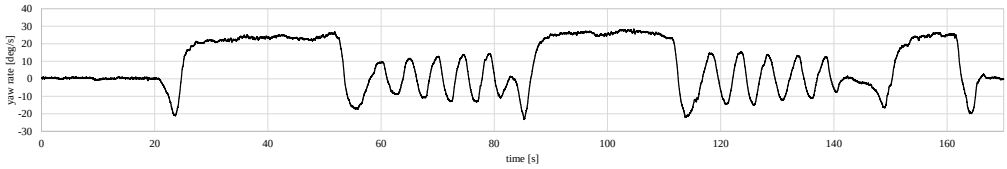
(**c**) Lateral acceleration



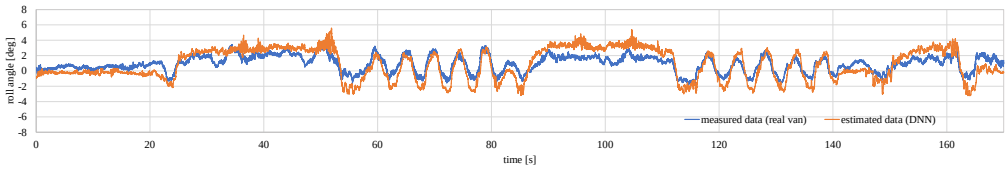
(**d**) Longitudinal acceleration



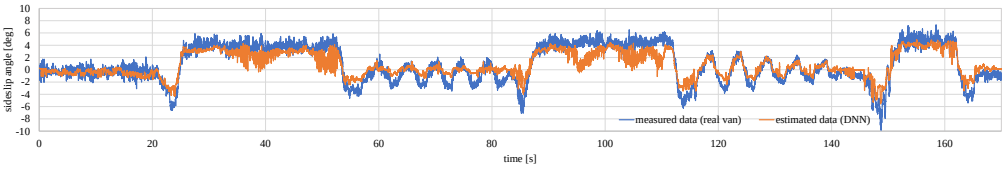
(**e**) Roll rate



(**f**) Yaw rate



(**g**) Roll angle



(**h**) Sideslip angle

Figure 12. Experimental results for the validation test.

Finally, a comparison with other methods previously proposed for the estimation of roll and sideslip angles has been made (see Table 4). These methods do not estimate both angles jointly in contrast to the proposed method. Considering the roll angle estimation, the van used was the same as in Boada et al. [18]. As is depicted in Table 4, the proposed method achieves better results both in maximum and norm errors. On the other hand, if the sideslip angle is taking into account, the method proposed shows better simulation results than the method proposed by Kim et al. [27] as is reported in Table 3. Regarding experimental results, although the RMS and maximum errors obtained in [27] are smaller, it should be noted that the type of vehicle used to carry out the essays is different. The van used in this work is characterized by a high CoG (Center of Gravity), and it is equipped with a soft suspension which involves a more significant chassis movement, thereby affecting the longitudinal and lateral load transfer and the vehicle's dynamic.

Table 4. Comparison with other methods for roll and sideslip angles estimation.

		Roll Angle	**Sideslip Angle**	
		Boada et al. [18]	**Kim et al. [27]**	
Type of Test		**Experimental**	**Simulation**	**Experimental**
Errors	**RMSE [°]**	-	0.14–0.96	0.12–0.19
	E_{max} [°]	5.5	0.92–3.84	0.75–1.92
	E_t [-]	1.95	-	-
Type of vehicle		van	sedan	
Type of NN		MLP	LSTM + deep ensemble	

Considering the size of training data (more than 400 K elements), and the DNN width and depth, each training block took initially around 40 min. After including Keras in the training code, this time was reduced to one fourth. The implemented DNN uses vectorization and efficient matrix handling libraries, such as NumPy, which made this process faster. With other configurations, including more profound and broader networks, the times varied, requiring initial and sometimes more than seven hours per training process. Regarding the predictions, in the same type of computer, processing 70 K elements take around 3 s. This model has to be tested in small embedded systems, such as the Raspberry Pi 4 Model B+ prototyping boards, to assess that predictions are provided at least at a sustained rate of 50 Hz (20 ms). This real-time constraint must be satisfied with safety systems. The performance results from different studies, where the authors used the previous Raspberry Pi 3 Model B+ to run deep learning models [45–47] in Keras and TensorFlow, and all of them heavier than the one presented in this work, show that the average time for evaluating an input is ranged from 0.001 s to 0.12 s. Given that the projected hardware to be used is the Raspberry Pi 4 Model B+, which has been proved to perform more than four times better than the former model [48], it seems feasible that the system will satisfy the real-time constraint mentioned before.

5. Conclusions

The main objective behind this work was to define a proper Deep Learning approach to simultaneously estimate sideslip and roll angles in commercial vehicles by capturing the required input values via a set of sensors such as IMU and steering angle sensor. Specifically: longitudinal acceleration, lateral acceleration, roll rate, yaw rate, and the steering angle. The DNN configuration was designed trying to minimize the impact in processing the outcomes; this is, looking for a compromise between width, depth, and prediction quality after considering the hardware restrictions and limited performance of vehicular embedded systems. Besides, the proposed DNN does not use previous information from the sensors but only the one that is provided at that step of time, so that there are no stability problems associated with an accumulated error. Several configurations were trained and

tested, altering both the network shape and the hyperparameters set. The validation was conducted in two phases, the first one, using a subset of the outcome data from Trucksim®, a well known vehicular simulator, that was used to model the physics of a real vehicle (van), and to use that van in a series of simulations consisting of performing different driving maneuvers. In the second validation phase, and after passing the quality check by analyzing the predictor errors against the test data subset, the model was assessed against the data captured by a ground truth device, specifically a Racelogic VBOX IMU, used in real driving maneuvers with a similar van such as the one modeled in Trucksim®. Finally, errors against this real data were also analyzed to check if the predictor is suitable for being used in real vehicles, which seems to be the case.

Future Work

Considering the outstanding predictions returned by the implemented DNN model, it is planned to integrate it into an enhanced version of the IoT architecture proposed in previous works. It is composed of a Raspberry Pi 4 Model B+ (4 GB), a BNO55 IMU Shield, the Racelogic VBOX used in this study, a laptop, and a 4G mobile hotspot device to provide connectivity among the prototyping board (RPi4) device and the laptop. At the same time, this IoT ecosystem will be integrated into two different testing vehicles (the Mercedes Sprinter van used to capture the validation data in this work, and a buggy). The objective of this new experiment is to validate both the roll and sideslip angles predicted in real-time, comparing the results with the ground truth device (Racelogic VBOX). On the other hand, there are also research efforts put on improving the values of the hyperparameters, and the DNN structure/combination of activation functions in the inner hidden layers, trying to increase even more the outcome precision and reducing the computational cost to make it perfectly suitable for its integration into ultra-low-cost embedded systems.

Author Contributions: Conceptualization, L.P.G., S.S.S., J.G.-G., M.J.L.B. and B.L.B.; Data curation, S.S.S.; Funding acquisition, M.J.L.B. and B.L.B.; Investigation, L.P.G., S.S.S. and M.J.L.B.; Methodology, L.P.G., S.S.S., J.G.-G., M.J.L.B. and B.L.B.; Software, L.P.G. and M.J.L.B.; Supervision, M.J.L.B.; Validation, L.P.G., J.G.-G., M.J.L.B. and B.L.B.; Writing—original draft, L.P.G., S.S.S. and J.G.-G.; Writing—review and editing, L.P.G., S.S.S., M.J.L.B. and B.L.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Agencia Estatal de Investigacion (EAI) of the Ministry of Science and Innovation of the Government of Spain through the project RTI2018-095143-B-C21.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ABS	Anti-Blocking System
ANN	Artificial Neural Network
CoG	Center of Gravity
DNN	Deep Neural Network
ESC	Electronic Stability Controllers
GPS	Global Positioning System
IMU	Inertial Measurement Unit
LSTM	Long Short-Term Memory
MLP	Multilayer Perceptron
MSE	Mean Squared Error
NLP	Natural Language Processing
RMSE	Root-Mean-Square Error
RNN	Recurrent Neural Networks

References

1. World Health Organization. *Global Status Report on Road Safety 2018: Summary*. Technical Documents; WHO: Geneva, Switzerland, 2018. Available online: https://apps.who.int/iris/handle/10665/277370 (accessed on 29 June 2020).
2. Saleh, L.; Chevrel, P.; Claveau, F.; Lafay, J.; Mars, F. Shared Steering Control Between a Driver and an Automation: Stability in the Presence of Driver Behavior Uncertainty. *IEEE Trans. Intell. Transp. Syst.* **2013** *14*, 974–983. [CrossRef]
3. Xiong, L.; Xia, X.; Lu, Y.; Liu, W.; Gao, L.; Song, S.; Han, Y.; Yu, Z. IMU-Based Automated Vehicle Slip Angle and Attitude Estimation Aided by Vehicle Dynamics. *Sensors* **2019**, *19*, 1930. [CrossRef] [PubMed]
4. Guzman, J.G.; Gonzalez, L.P.; Redondo, J.P.; Sanchez, S.S.; Boada, B.L. Design of Low-Cost Vehicle Roll Angle Estimator Based on Kalman Filters and an IoT Architecture. *Sensors* **2018**, *18*, 1800. [CrossRef] [PubMed]
5. Chindamo, D.; Lenzo, B.; Gadola, M. On the vehicle sideslip angle estimation: A literature review of methods, models, and innovations. *Appl. Sci.* **2018**, *8*, 355. [CrossRef]
6. Guo, H.; Cao, D.; Chen, H.; Lv, C.; Wang, H.; Yang, S. Vehicle dynamic state estimation: State of the art schemes and perspectives. *IEEE/CAA J. Autom. Sin.* **2018**, *5*, 418–431. [CrossRef]
7. Nam, K.; Oh, S.; Fujimoto, H.; Hori, Y. Estimation of sideslip and roll angles of electric vehicles using lateral tire force sensors through RLS and Kalman filter approaches. *IEEE Trans. Ind. Electron.* **2012**, *60*, 988–1000. [CrossRef]
8. Jin, X.; Yin, G.; Chen, N. Advanced Estimation Techniques for Vehicle System Dynamic State: A Survey. *Sensors* **2019**, *19*, 4289. [CrossRef]
9. Chen, B.C.; Hsieh, F.C. Sideslip angle estimation using extended Kalman filter. *Veh. Syst. Dyn.* **2008**, *46*, 353–364. [CrossRef]
10. Li, L.; Jia, G.; Ran, X.; Song, J.; Wu, K. A variable structure extended Kalman filter for vehicle sideslip angle estimation on a low friction road. *Veh. Syst. Dyn.* **2014**, *52*, 280–308. [CrossRef]
11. Liu, Y.H.; Li, T.; Yang, Y.Y.; Ji, X.W.; Wu, J. Estimation of tire-road friction coefficient based on combined APF-IEKF and iteration algorithm. *Mech. Syst. Sig. Process.* **2017**, *88*, 25–35. [CrossRef]
12. Rath, J.J.; Veluvolu, K.C.; Defoort, M.; Soh, Y.C. Higher-order sliding mode observer for estimation of tyre friction in ground vehicles. *IET Control Theory Appl.* **2014**, *8*, 399–408. [CrossRef]
13. Cheli, F.; Braghin, F.; Brusarosco, M.; Mancosu, F.; Sabbioni, E. Design and testing of an innovative measurement device for tyre–road contact forces. *Mech. Syst. Sig. Process.* **2011**, *25*, 1956–1972. [CrossRef]
14. Zhang, C.; Chen, Q.; Qiu, J. Robust $\mathcal{H}_\infty$ filtering for vehicle sideslip angle estimation with sampled-data measurements. *Trans. Inst. Meas. Control* **2019**, *39*, 1059–1070. [CrossRef]
15. Zhang, H.; Huang, X.; Wang, J.; Karimi, H.R. Robust energy-to-peak sideslip angle estimation with applications to ground vehicles. *Mechatronics* **2015**, *30*, 338–347. [CrossRef]
16. Zhao, L.; Liu, Z. Vehicle Velocity and Roll Angle Estimation with Road and Friction Adaptation for Four-Wheel Independent Drive Electric Vehicle. *Math. Prob. Eng.* **2014**. [CrossRef]
17. Boada, B.; Boada, M.; Diaz, V. Vehicle sideslip angle measurement based on sensor data fusion using an integrated ANFIS and an Unscented Kalman Filter algorithm. *Mech. Syst. Sig. Process.* **2016**, *72–73*, 832–845. [CrossRef]
18. Boada, B.; Boada, M.; Vargas-Melendez, L.; Diaz, V. A robust observer based on $\mathcal{H}_\infty$ filtering with parameter uncertainties combined with Neural Networks for estimation of vehicle roll angle. *Mech. Syst. Sig. Process.* **2018**, *99*, 611–623. [CrossRef]
19. Kuutti, S.; Bowden, R.; Jin, Y.; Barber, P.; Fallah, S. A Survey of Deep Learning Applications to Autonomous Vehicle Control. *IEEE Trans. Intell. Transp. Syst.* **2020**, 1–22. [CrossRef]
20. Park, S.H.; Kim, B.; Kang, C.M.; Chung, C.C.; Choi, J.W. Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26–30 June 2018; pp. 1672–1678.
21. Moujahid, A.; Tantaoui, M.E.; Hina, M.D.; Soukane, A.; Ortalda, A.; ElKhadimi, A.; Ramdane-Cherif, A. Machine Learning Techniques in ADAS: A Review. In Proceedings of the 2018 International Conference on Advances in Computing and Communication Engineering (ICACCE), Paris, France, 22–23 June 2018.

22. Huval, B.; Wang, T.; Tandon, S.; Kiske, J.; Song, W.; Pazhayampallil, J.; Andriluka, M.; Rajpurkar, P.; Migimatsu, T.; Cheng-Yue, R.; et al. An Empirical Evaluation of Deep Learning on Highway Driving. *arXiv* **2015**, arXiv:1504.01716.
23. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: London, UK, 2017; Available online: http://www.deeplearningbook.org (accessed on 29 June 2020).
24. Marina, L.A.; Trasnea, B.; Grigorescu, S.M. A Multi-Platform Framework for Artificial Intelligence Engines in Automotive Systems. In Proceedings of the 2018 22nd International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, 10–12 October 2018.
25. Che, Z.; Purushotham, S.; Cho, K.; Sontag, D.; Liu, Y. Recurrent neural networks for multivariate time series with missing values. *Sci. Rep.* **2018**, *8*, 6085. [CrossRef]
26. Singh, J. Real world applications of neural networks in natural language processing. *Int. J. Recent Trends Eng. Res.* **2018**, *4*, 61–63.
27. Kim, D.; Min, K.; Kim, H.; Huh, K. Vehicle sideslip angle estimation using deep ensemble-based adaptive Kalman filter. *Mech. Syst. Sig. Process.* **2020**, *144*, 106862. [CrossRef]
28. Zia, T.; Zahid, U. Long short-term memory recurrent neural network architectures for Urdu acoustic modeling. *Int. J. Speech Technol.* **2019**, *22*, 21–30. [CrossRef]
29. Wang, P.; Qian, Y.; Soong, F.K.; He, L.; Zhao, H. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *arXiv* **2015**, arXiv:1510.06168.
30. Sak, H.; Senior, A.; Beaufays, F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In Proceedings of the Fifteenth Annual Conference of the International Speech Communication Association, Singapore, 14–18 September 2014.
31. Hendry and Chen, R.C.; Using Deep Learning to Predict User Rating on Imbalance Classification Data. *IAENG Int. J. Comput. Sci.* **2019**, *46*, 109–117.
32. Fooshee, D.; Mood, A.; Gutman, E.; Tavakoli, M.; Urban, G.; Liu, F.; Huynh, N.; Van Vranken, D.; Baldi, P. Deep learning for chemical reaction prediction. *Mol. Syst. Des. Eng.* **2018**, *3*, 442–452. [CrossRef]
33. Frank, L.R.; Ferreira, Y.M.; Julio, E.P.; Ferreira, F.H.C.; Dembogurski, B.J.; Silva, E.F. Multilayer Perceptron and Particle Swarm Optimization Applied to Traffic Flow Prediction on Smart Cities. In Proceedings of the International Conference on Computational Science and Its Applications, Saint Petersburg, Russia, 1–4 July 2019; pp. 35–47.
34. Python Software Foundation. Python Language Reference. Version 3.7.4. Available online: https://docs.python.org/3/reference/ (accessed on 29 June 2020).
35. Oliphant, T.E. *A Guide to NumPy*; Trelgol Publishing: North Charleston, SC, USA, 2006, Volume 1.
36. McKinney, W. Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference, Austin, TX, USA, 28 June–3 July 2010; Volume 445, pp. 51–56.
37. Hunter, J.D. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.* **2007**, *9*, 90. [CrossRef]
38. Keras. Available online: https://keras.io (accessed on 21 June 2020.)
39. Merkel, D. Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux J.* **2014**. Available online: https://www.linuxjournal.com/content/docker-lightweight-linux-containers-consistent-development-and-deployment (accessed on 29 June 2020.)
40. González, L.P. Hacia una Representación del Conocimiento Basada en la Percepción. Ph.D. Thesis, Computer Science and Technology Department, Universidad Carlos III de Madrid, Madrid, Spain, 2017. Available online: https://e-archivo.uc3m.es/handle/10016/24810 (accessed on 29 June 2020.)
41. Vargas-Meléndez, L.; Boada, B.L.; Boada, M.J.L.; Gauchía, A.; Díaz, V. A Sensor Fusion Method Based on an Integrated Neural Network and Kalman Filter for Vehicle Roll Angle Estimation. *Sensors* **2016**, *16*, 1400. [CrossRef]
42. The HDF Group. Hierarchical Data Format Version 5. Available online: http://www.hdfgroup.org/HDF5 (accessed on 21 June 2020).
43. Boada, B.L.; Garcia-Pozuelo, D.; Boada, M.J.L.; Diaz, V. A Constrained Dual Kalman Filter Based on pdf Truncation for Estimation of Vehicle Parameters and Road Bank Angle: Analysis and Experimental Validation. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 1006–1016. [CrossRef]
44. Boada, B.L.; Boada, M.J.L.; Zhang, H. Sensor Fusion Based on a Dual Kalman Filter for Estimation of Road Irregularities and Vehicle Mass Under Static and Dynamic Conditions. *IEEE/ASME Trans. Mechatron.* **2019**, *24*, 1075–1086. [CrossRef]

45. Shiddieqy, H.A.; Hariadi, F.I.; Adiono, T. Implementation of deep-learning based image classification on single board computer. In Proceedings of the 2017 International Symposium on Electronics and Smart Devices (ISESD), Yogyakarta, Indonesia, 17–19 October 2017; pp. 133–137.
46. Morehead, A.; Ogden, L.; Magee, G.; Hosler, R.; White, B.; Mohler, G. Low Cost Gunshot Detection using Deep Learning on the Raspberry Pi. In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 12–19 December 2019; pp. 3038–3044.
47. Curtin, B.H.; Matthews, S.J. Deep Learning for Inexpensive Image Classification of Wildlife on the Raspberry Pi. In Proceedings of the 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), Vancouver, BC, Canada, 17–19 October 2019; pp. 82–87.
48. Larabel, M. Initial Raspberry Pi 4 Performance Benchmarks—Phoronix, 2019. Available online: https://www.phoronix.com/scan.php?page=article&item=raspberry-pi4-benchmarks&num=5 (accessed on 29 June 2020).