A Deep-Learning Model of Virtual Test Drivers

T. Pallacci, N. Mimmo, *Member, IEEE*, P. Sessa, and R. Rabbeni

***Abstract*—Virtual test drivers are becoming a paramount automatic verification tool enabling car makers to test new and advanced vehicle functionalities in a standardized, repeatable, high-quality, and cost-saving way. In this letter, we use modern machine-learning methodologies to build a virtual driver able to test the hill-descent control, one of the driver assistance systems equipping modern cars. The experimental results show that our virtual driver performs as a human driver involved in the same test conditions.**

***Index Terms*—Deep learning.**

I. Introduction

In the context of vehicles' functionality verification and validation phases, car manufacturers commonly use different combinations of human, physical systems, and digital tools with the main goal of minimizing costs, efforts, and time while keeping the highest quality standard [1]. For example, the Hill-Descent Control (HDC) is an advanced driver assistance system that keeps the car's speed constant while descending a hill. When the HDC is active, the driver can regulate the desired descent speed with switch buttons on the steering wheel and with gas and brake pedals. Therefore, testing the HDC also means following test-specific brake pedal trajectories.

Traditionally, one of the most common approaches consists of the so-called *driver-in-the-loop*, in which a professional test driver is directly involved in the control of either the simulated or real vehicle. In this letter, the human test driver is conceived as a system which controls the car to achieve the goal of testing the vehicle's functionalities [2].

On the other hand, performing automatic tests, *i.e.*, tests without human drivers, could represent good practice because they can run 24/7, and are repeatable and standardized, which are favorable features for improving vehicle systems development. For these reasons, car makers are adopting virtual design and testing, which are based on advanced digital technologies.

Goals: In this letter, we investigate the problem of creating a digital twin of human test drivers, here called *Virtual Test Driver* (VTD), for testing the HDC. We aim at modeling a VTD able to dynamically adapt its actions to stimuli coming from road conditions and vehicle dynamics while exploring the HDC domain. Moreover, VTD's responses should be as realistic as possible, *i.e.*, they should be comparable to what human test drivers would do in the same testing conditions [3].

State of the art: Researchers have been trying to describe, reproduce, and simulate car drivers' behavior since the 1950s [4] by focusing mainly on high-level tasks, *i.e.*, speed regulation, car-following, and lane-change [5], [6]. Section III of [7] reports an interesting review of the most recently developed driver models. Roughly, the current literature can be split into *model-based* and *data-driven* methods.

The process of creation of a model, *e.g.*, the VTD, from data is known in the automatic control literature as *system identification*. Roughly, system identification represents the process of finding the model that relates the collected data in the best way, concerning some measurable Key Performance Indicators (KPIs). *Deep learning* represents one of the most recent techniques in the field of machine learning and, in our context, is seen as one of the many possible interpretations of *system identification*. Deep learning has had a long and rich history starting in the 1940s with the development of theories of biological learning [8] via the so-called Artificial Neural Networks (ANNs). Later in the 1980s, *deep ANNs* were born as the principle of learning multiple layers of composition [9]. ANNs can be classified as Feed-Forward (FFNNs) and Recursive (RNNs). FFNNs organize their layers in a cascade while in RNNs the output of some layers is fed back to previous layers. Hochreiter and Schmidhuber [10] introduced the Long Short-Term Memory (LSTM) network, a kind of RNN, to model long sequences, *i.e.*, to model relationships between sequences rather than just fixed inputs, see also [11]. As for machine-learning models of human drivers, in [12], the focus is on lane-change acceptance and the driver is modeled via feed-forward ANNs, which are exploited to represent memoryless systems. In [13], a partly connected multilayered perceptron was exploited to let the VTD imitate human behavior on gas and brake pedals during speed regulation tasks. A driver's intention predictor taking into account emotional factors was designed via a support vector machine in [14]. A deep learning approach in modeling the steering intention of human drivers was proposed in [15].

It is worth noting that the cited models have been developed for classic driving tasks such as speed regulation and car-following. Unfortunately, these models could be useful for testing car functionalities only in regular driving conditions.

Manuscript received 8 March 2024; revised 10 May 2024; accepted 26 May 2024. Date of publication 4 June 2024; date of current version 19 June 2024. Recommended by Senior Editor T. Oomen. *(Corresponding author: N. Mimmo.)*

T. Pallacci, P. Sessa, and R. Rabbeni are with the Department of Electrical, Electronics and Hardware Engineering, Maserati, 41121 Modena, Italy (e-mail: thomas.pallacci@external.maserati.com; pasquale.sessa@maserati.com; roberto.rabbeni@maserati.com).

N. Mimmo is with the Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi", University of Bologna, 40136 Bologna, Italy (e-mail: nicola.mimmo2@unibo.it).

Digital Object Identifier 10.1109/LCSYS.2024.3408229

On the opposite, we are interested in testing car functionalities, especially in off-design conditions.

Contributions: In the context of the creation of a human-inspired VTD, in this letter, we adopt a deep learning-based system identification strategy which relies on real-world data collected during test campaigns performed by professional test drivers. We took the inspiration from the driver's architecture depicted in [16]. In particular, we modeled the VTD as a system of two modules called *decision-maker* and *executor*. Both consist of LSTM-NN whose datasets have been properly specialized. The decision-maker elaborates on environmental data acquired by exteroceptive sensors, such as camera and lidar, and, based on the vehicle state, decides the manoeuvres the executor should undertake. Then, the executor, which embeds manoeuvre primitives, performs the action commanded by the decision-maker while gathering information from both exteroceptive and proprioceptive sensors, *e.g.*, IMU. One of the pros of this architecture is its modularity. Indeed, the classes of decisions and the base manoeuvres can be extended independently from each other. This makes the applicability of our VTD easily extendable to driving contexts more generic than HDC testing considered in this preliminary work.

In this letter, the driver's model architecture is not divided into car-following and lane-change sub-modules. Instead, we consider simpler basic manoeuvres, such as accelerating and steering, as primitives. Moreover, we improve the seminal decision-making module by [16] by changing its modeling approach. Differently from [16] we do not rely on statistical rules to define when to start and finish each manoeuvre, which seems quite complicated and makes the managing of each condition non-trivial. Moreover, the framework proposed in [16] had an additional sub-module placed between the decision and the execution ones. This further module, called *planning*, links the environment to the expected effect of executed manoeuvres and it was used to define the driver's intention. The planning module provides constant values, which makes hard the use of recurrent neural networks, such as LSTM, for modeling the executor. Despite commonalities with our VTD, the driver model of [16] was proposed to simulate local traffic scenarios. Contrarily, we took inspiration from [16] to develop a more sophisticated driver model specifically testing vehicle functionalities. Moreover, our VTD's decision-making part extends the applicability of that defined in [16].

We tested our VTD in many challenging hill-decent scenarios, characterized by time-varying slopes and curvatures, in which the comparison with human test drivers has demonstrated the high accuracy of our solution. Our VTD shows a human-like behavior as detailed in Section III-E.

II. Basic Notions, Problem Statement, and Proposed Solution

A. Basic Notions

Learning algorithms learn from experience concerning some class of tasks, $\mathcal{T}$, and are evaluated via measurable performance indices J. More in detail, we define the experience on a specific task $\mathcal{T}$ as the collection of *examples*, *i.e.*, a *dataset* $\mathcal{D}$. In turn, an example is a *collection of features* that has been measured during experiments. As an example,

$$\begin{tikzcd}
z_1(k) \arrow[r] & \odot \arrow[rr] & & + \arrow[rrr, dash] & & & {} \arrow[r] \arrow[d] & z_1(k+1) \\
& & {} \arrow[r] & \odot \arrow[u] & & & \tanh \arrow[dd] & \\
& \sigma \arrow[uu] & \sigma \arrow[u, dash] & \tanh \arrow[u] & & & & \\
z_2(k),\,\mu(k) \arrow[r, dash] & {} \arrow[u] \arrow[r, dash] & {} \arrow[u] \arrow[r, dash] & {} \arrow[u] \arrow[rr] & & \sigma \arrow[r] & \odot \arrow[r] & z_2(k+1),\,\upsilon(k)
\end{tikzcd}$$

Fig. 1. LSTMS-ANN: at each time step k, the network is fed by input data $\mu(k)$, elaborates the output $\upsilon(k)$, and computes the future states $z_1(k+1)$ and $z_2(k+1)$.

in the context of HDC, the task is that of driving a car while activating/deactivating and regulating the functionality via buttons and pedals. We symbolise the ith example as the composition of input and output matrices $Y_i \in \mathbb{R}^{p\times n_i}$ and $U_i \in \mathbb{R}^{q\times n_i}$, with $p, q, n_i \in \mathbb{N}$, where each row represents a feature and each column is the sampling of that feature at a given time instant. In the case of HDC, pedal positions, button status, and IMU data are examples of features. One of the most common machine-learning tasks includes the so-called *regression*. More in detail, assuming that m examples are available, we denote the sets to which belong the columns of Y_i and U_i, for $i = 1, \dots, m$, with $\mathcal{Y} \subseteq \mathbb{R}^p$ and $\mathcal{U} \subseteq \mathbb{R}^q$. In regression, we aim at learning a parameterized function $\Sigma_\Theta : \mathcal{Y} \times \mathbb{R} \to \mathcal{U}$, in which Θ represents the parameter vector, that correlates the input and output data over time. In detail, we aim at tuning Θ to reduce the so-called *training error*, *i.e.*, an error computed on the training dataset, which represents a subpart of $\mathcal{D}$. For the specific case of the HDC, we want the learning to predict the test driver braking actions. More specifically, we aim at minimizing the difference between actual and estimated pedal positions. Besides, to evaluate the abilities of a machine learning algorithm, we are interested in how well the algorithm performs on data that it has not seen before, *i.e.*, the so-called *validation* set. The ability to perform well on the validation set is called *generalization*. Naturally, we want to reduce also the generalization error and, to do so, we usually tune the so-called *hyperparameters*. As an example, a hyperparameter are the dimensions of Θ.

Among the several kinds of parametric functions proposed by the ANN literature, the so-called *Long Short-Term Memory* (LSTM)-ANNs demonstrated to perform well on time-series data. In particular, Σ_Θ is conceived as the composition of the sigmoid and hyperbolic tangent functions σ and tanh organized in n *cells*, with $n \in \mathbb{N}$. LSTM-ANNs are discrete-time nonlinear systems with input $\mu \in \mathbb{R}^{n_i}$, states $z_1 \in \mathbb{R}^n$ and $z_2 \in \mathbb{R}^n$, and output $\upsilon \in \mathbb{R}^n$, with $n_i, n \in \mathbb{N}$, see Figure 1. In detail, the state and output of an LSTM-ANN evolve accordingly with

$$\begin{aligned} z_1(k+1) &= c(k) z_1(0) = 0 \\ z_2(k+1) &= d(k) z_2(0) = 0 \\ \upsilon(k) &= d(k) \end{aligned} \tag{1a}$$

in which

$$\begin{aligned} c(k) &:= \sigma\left(a^{\mathrm{f}}(k)\right) \odot z_1(k) + \sigma\left(a^{\mathrm{i}}(k)\right) \odot \tanh\left(a^{\mathrm{c}}(k)\right) \\ d(k) &:= \sigma\left(a^{\mathrm{o}}(k)\right) \odot \tanh(c(k)), \end{aligned} \tag{1b}$$

for all $k \in \mathbb{N}_0$, and where $\odot$ represents the Hadamard product. Then, the so-called *activation functions* a^{f}, a^{o}, a^{i}, and a^{c} are determined as

$$\begin{aligned} a^{\mathrm{f}}(k) &:= b^{\mathrm{f}} + W^{\mathrm{f}} z_2(k) + R^{\mathrm{f}}\mu(k) \\ a_i^{\mathrm{i}}(k) &:= b^{\mathrm{i}} + W^{\mathrm{i}} z_2(k) + R^{\mathrm{i}}\mu(k) \\ a^{\mathrm{o}}(k) &:= b^{\mathrm{o}} + W^{\mathrm{o}} z_2(k) + R^{\mathrm{o}}\mu(k) \\ a_i^{\mathrm{c}}(k) &:= b^{\mathrm{c}} + W^{\mathrm{c}} z_2(k) + R^{\mathrm{c}}\mu(k) \end{aligned} \tag{1c}$$

with b^{f}, b^{i}, b^{o}, $b^{\mathrm{c}} \in \mathbb{R}^n$, W^{f}, W^{i}, w^{o}, $w^{\mathrm{c}} \in \mathbb{R}^{n\times n}$, and $R^{\mathrm{f}}, R^{\mathrm{i}}, R^{\mathrm{o}}, R^{\mathrm{c}} \in \mathbb{R}^{n\times n_i}$. Roughly, z_1 accumulates new information, contained in $\tanh(a^{\mathrm{c}})$ and weighted by $\sigma(a^{\mathrm{i}})$, with a time-varying forgetting rate given by $\sigma(a^{\mathrm{f}})$. Moreover, z_2 represents a memory cell which keeps the last output value υ, which is computed as the normalized version of the next-step z_1, *i.e.*, $\tanh(c)$, weighted by $\sigma(a^{\mathrm{o}})$.

Usually, LSTM-ANNs are connected to the input data y and the output u through FF-ANNs. The input FF-ANN elaborates y for using the LSTM-ANN while the output FF-ANN translates the LSTM's output into u-compatible output. Let "i" and "o" be the subscripts for input and output. Define $\ell_{\mathrm{i}}, \ell_{\mathrm{o}} \in \mathbb{N}$ as the number of input and output layers. Then, the input and output networks are defined as

$$\begin{aligned} h_1^{\#}(k) &= \sigma\left(b_1^{\#} + W_1^{\#} y(k)\right) \\ h_{j+1}^{\#}(k) &= \sigma\left(b_j^{\#} + W_j^{\#} h_j^{\#}(k)\right) j = 1, \ldots, \ell_{\#} - 1 \\ \mu(k) &= \sigma\left(b_{\ell_{\#}}^{\#} + W_{\ell_{\#}}^{\#} h_{\ell_{\#}}^{\#}(k)\right) \end{aligned} \tag{1d}$$

in which $\# \in \{\mathrm{i}, \mathrm{o}\}$. The vectors h_j^{i} and h_j^{o} represent the so-called hidden-layer variables. Differently from (1c), the (vectors and matrices of) parameters b_j^{i}, W_j^{i}, b_j^{o}, and W_j^{o} in (1d) may have different dimensions for each layer.

The set of vectors and matrices b, W, R (from which we have hidden all the super- and sub-scripts to keep the notation light), appearing in (1c)–(1d), represents the network parameters Θ which is determined through the training process. Conversely, the number of cells n, the number of input and output layers ℓ_{i} and ℓ_{o}, and the dimensions of b_j^{i}, W_j^{i}, b_j^{o}, and W_j^{o} are considered hyperparameters.

B. Problem Statement

We define x, u, w, and y to be the state, the input, the uncertainties, and the output describing a car, whose dynamics are modeled via the following sampled-time system

$$\begin{aligned} x(k+1) &= f(x(k), u(k), w(k)) x(0) = x_0 \\ y(k) &= h(x(k), w(k)). \end{aligned} \tag{2}$$

in which x_0 denotes the initial condition. The output $y \in \mathbb{R}^p$ is generic and contains classic sensor outputs, such as IMU, GNSS, radar, and more recently available data such as the RGB and depth values of cameras, point clouds, and so forth. Classic vehicular inputs, $u \in \mathcal{U} \subset \mathbb{R}^q$, are the steering wheel angle, the brake and acceleration pedals, the gear selector, the buttons used for (de)activating and regulating automatic functionalities, and so forth. In this letter, we assume that the input vector u is measurable. Moreover, we assume all the inputs are constrained within a bounded domain $\mathcal{U}$. In our formulation, w collects both exogenous disturbances, such as the wind and the road grade, and the sensors' noise. Finally, the state x represents a design choice and may contain

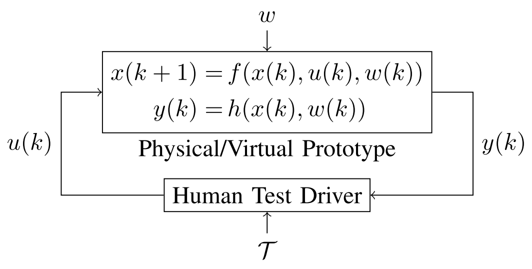

Fig. 2. The data-provider plant consists of a human test driver, performing the task $\mathcal{T}$, *i.e.*, testing the vehicle functionalities, in closed-loop with a vehicle which can be a physical or virtual prototype. Both inputs u and output y are sampled and collected into a dataset $\mathcal{D}$.

variables such as the car's position, speed, attitude, and so on. Moreover, we assume (2) being controlled by a human test driver as depicted in Figure 2.

We model the VTD as a dynamic system, whose state is z, whose input is y, and whose output is $\hat{u}$. In more detail, we conceive two functions g and ι such that

$$\begin{aligned} z(k+1) &= g(z(k), y(k)) z(0) = z_0 \\ \hat{u}(k) &= \iota(z(k), y(k)). \end{aligned} \tag{3}$$

Now, we assume that $m \in \mathbb{N}$ drive runs have been performed during which $n_i \in \mathbb{N}$, with $i = 1, \ldots, m$, samples of inputs and outputs have been collected into the matrices

$$\begin{aligned} Y_i &:= \left[y(n_i),\ y(n_i - 1),\ \ldots,\ y(1)\right] \\ U_i &:= [u(n_i),\ u(n_i - 1),\ \ldots,\ u(1)]. \end{aligned} \tag{4}$$

Regarding Section II-A, each drive run represents an *example* while the input/output matrices (4) are the *collection of features*. We divide $\mathcal{M} := \{1, \ldots, m\}$ into two disjoint subsets, namely $\mathcal{M}_t$ and $\mathcal{M}_v$. We define $\mathcal{D}_t := \bigcup_{i\in\mathcal{M}_t}(Y_i, U_i)$ and $\mathcal{D}_v := \bigcup_{i\in\mathcal{M}_v}(Y_i, U_i)$ and we call them *training* and *validation* set. In this context, the identification process consists of exploiting $\mathcal{D}$ to design g, ι, and z_0 solving the following multi-objective optimization problem

$$\begin{aligned} J_t &:= \min_{g,\iota,z_0} \frac{1}{n_t} \sum_{i\in\mathcal{M}_t} e\left(\hat{u}_i, u_i\right) \\ J_v &:= \min_{g,\iota,z_0} \frac{1}{n_v} \sum_{i\in\mathcal{M}_v} e\left(\hat{u}_i, u_i\right), \end{aligned} \tag{5a}$$

subject to (2) and (3), where n_t and n_v denote the dimensions of $\mathcal{M}_t$ and $\mathcal{M}_v$, and

$$e\left(\hat{u}_i, u_i\right) := \frac{1}{n_i} \sum_{k=1}^{n_i} \|\hat{u}_i(k) - u_i(k)\|^2 \tag{5b}$$

is the Mean-Squared-Error (MSE) associated with the couple $(\hat{u}_i, u_i)$. Roughly, we want to find a VTD model exploiting the training set $\mathcal{D}$ to foresee, as well as possible, the control inputs u of a test driver in the case when only the actual measurement y is available. This problem is usually reported in the literature as a *simulation* problem [[17], §16.4] and it differs from the so-called prediction problem in which the future output is forecast based on both input and output past data. We remark that LSTM-ANNs fit well and better than the so-called Temporal Convolutional Networks into the simulation framework. Moreover, we want to minimis both the training and the validation errors J_t and J_v.

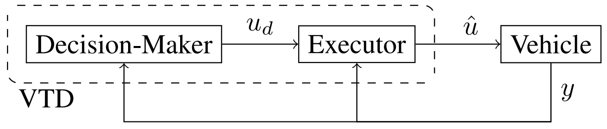

Fig. 3. The proposed VTD is composed of two parts, a decision-maker and a manoeuvre executor. The VTD takes data from the environment and the vehicle state and feeds the vehicle with driving inputs.

C. Proposed Solution

This section describes the algorithm we propose for solving problem (5). We modeled the VTD as a composition of two main parts, *i.e.*, a *decision-maker* and an *executor*, see Figure 3.

The vehicle is equipped with an image processor collecting environmental data through a camera and elaborating high-level information such as the presence of surrounding automobiles and their position and speed relative to the ego vehicle, time series of position and speed relative to the ego vehicle of surrounding automobiles, road conditions, road slope and curvature, and the current speed of the ego vehicle. The decision-maker takes the inputs from the image processor and elaborates a *decision*, *i.e.*, a signal that exclusively activates one of the manoeuvres among a prescribed set. For example, in this preliminary work, our VTD can choose between *brake* and *accelerate*. Then, the decision to brake or accelerate is passed to the executor who modulates the pedal deflections based on the current vehicle state and environmental data.

We divided z into two parts, namely z_d and z_e, the former for modeling the decision-maker and the latter for the executor. Then, accordingly to the architecture of Figure 3, we modeled the decision-maker as

$$\begin{aligned} z_d(k+1) &= g_d(z_d(k), y(k)) z_d(0) = z_{d,0} \\ u_d(k) &= \iota_d(z_d(k), y(k)). \end{aligned} \tag{6a}$$

The input y represents the data provided by the image processor and the proprioceptive vehicular sensors while the output u_d represents the *decision* signal. Similarly, the executor takes the form

$$\begin{aligned} z_e(k+1) &= g_e(z_e(k), u_d(k), y(k)) z_e(0) = z_{e,0} \\ \hat{u}(k) &= \iota_e(z_d(k), u_d(k), y(k)). \end{aligned} \tag{6b}$$

which receives as input u_d and y and elaborates the control action $\hat{u} \in \mathcal{U}$. In particular, we model (6a) and (6b) as a network having the following structure (see Section II-A): an input network with $\ell_\mathrm{i} = 1$, an n-cell LSTM network, and an output network with $\ell_\mathrm{o} = 2$. That is, (6a) and (6b) are systems in the form (1) with parameters Θ_d and Θ_e respectively. The dimensions of Θ_d and Θ_e represent hyperparameters whose tuning is detailed in Section III-C.

III. Experimental Results

This section describes the details of the proposed VTD specialized in testing the HDC. The data used in this letter are protected by an NDA and cannot be made public. However, they are common to vehicles equipped with stereo cameras or 3D lidars, range sensors, a GPS receiver, IMUs, and potentiometers.

A. Dataset Preparation

We collected data from a fleet of test vehicles equipped with HDC, and driven with HDC active and inactive, on highways, urban, and peri-urban roads in Europe and North America to let the VTD perform well in broadly different driving scenarios. We used a Controller Area Network (CAN) bus and a data logger to collect and synchronize proprioceptive and exteroceptive vehicular data. Finally, we improved the data quality through a Butterworth denoising filter.

Some of the trips on downhill roads were removed from $\mathcal{D}$ and used for the performance test described in Section III-E. In detail, let us denote with $\mathcal{D}_p \subset \mathcal{D}$ the subset of data used for the performance assessment. Then, we divided the dataset $\mathcal{D} \setminus \mathcal{D}_p$ into two parts, namely $\mathcal{D}_d$ and $\mathcal{D}_e$, with the former dedicated to the training and validation of the decision-maker and the latter to the executor. Indeed, since this latter is trained only on braking manoeuvres, we eliminated data relative to phases in which the gas pedal was in action. The dataset $\mathcal{D}_d$ consists of more than 51,900 trips, amounting to about 6,000 drive hours. In this preliminary work, the dataset $\mathcal{D}_e$ includes only braking manoeuvres for more than 368,800 examples, amounting to about 4,800 drive hours. Finally, we split $\mathcal{D}_d$ in two disjoint subsets $\mathcal{D}_{d,t}$ and $\mathcal{D}_{d,v}$, and $\mathcal{D}_e$ into $\mathcal{D}_{e,t}$ and $\mathcal{D}_{e,v}$, with $\mathcal{D}_{d,t}$ and $\mathcal{D}_{e,t}$ representing the decision-maker and executor training sets, and $\mathcal{D}_{d,v}$ and $\mathcal{D}_{e,v}$ denoting the decision-maker and executor validation sets. Approximately the 85% of $\mathcal{D}_d$ and $\mathcal{D}_e$ were used to train nets, and the 15% for validating them.

The VTD inputs are the ego vehicle dynamics, the inter-vehicle distances, and the road curvature. It is important to mention that the decision-maker considers the three closest vehicles, whereas the executor takes into account only the closest one. Indeed, we found a tight correlation between the test driver's braking response and the relative position of the nearest vehicle. Moreover, we tested that providing the ANN with more information does not improve the VTD's performance. Finally, it is worth mentioning that we feed the VTD with a look-ahead road curvature instead of the curvature at the current car's location.

B. Implementation Technicalities

The VTD was trained in MATLAB with input sequences' length of tens of samples, describing short-time manoeuvres, for the executor and tens of thousands samples, collected along whole trips, for the decision-maker. Both networks were trained using Adaptive Moment Estimation (ADAM). For the desktop-computer executions, we used a workstation with an Intel Gold 6242R CPU and 128 GB RAM, running MATLAB 2021a. It takes about 1.3 seconds to simulate 10 seconds of the synthetic environment in which the executor and decision-making LSTM-ANNs were called 107 and 113 times respectively, requiring approximately 0.4s each.

The output of the decision-making LSTM-NN, u_d, ranges from 0 to 1 and activates the braking module. Let $T \in [0, 1]$ be a threshold to discriminate braking manoeuvres. Then, the output of the decision-maker is pre-processed by the executor through an ideal Schmitt trigger, *i.e.*, a comparator with hysteresis, whose domain has been centred at the threshold T. The width of the hysteresis and T are some of the hyperparameters we tuned during the validation step.

Remark 1: Instead of having a continuous output $u_d \in [0, 1]$, later quantized with the procedure described above, one could have designed (6b) with a single binary output. We observed during preliminary tests that this optional solution works worse than ours.

The output of the braking module, $\hat{u}$, represents the pedal deflection and is defined in the domain [0, 1].

C. Training

We trained the decision-maker and the executor on the datasets $\mathcal{D}_{d,t}$ and $\mathcal{D}_{e,t}$ respectively. Both the nets were trained with no dropout layers. Therefore, to prevent overfitting, we applied L2 regularization [18] and we stopped the training when the loss function J_t was not improved for n_e consecutive epochs, where $n_e \in \mathbb{N}$ is a design parameter. The aim of introducing an n_e-based stopping criterion was to enhance the training while reducing the training time. The network parameters were estimated through Adaptive Moment Estimation, which performs well in the class of problems considered in this letter, as documented in [19]. More in detail, we found the learning rate 5e-3 representing a good compromise between estimation accuracy and convergence time. Finally, we normalized all the data to prevent scaling effects. The decision-maker network has 5 LSTM layers with 200 hidden units/layer while the executor has 7 LSTM layers with 100 hidden units/layer.

D. Validation

The generalization ability of the trained decision-maker and executor was assessed by calculating J_v, see (5a), restricted on $\mathcal{D}_{d,v}$ and $\mathcal{D}_{e,v}$. In Figure 4 we documented the distribution of the (square root of) $e(\hat{u}_i, u_i)$, *i.e.*, the MSE defined in (5b) and computed on $\mathcal{D}_{d,v}$ and $\mathcal{D}_{e,v}$ for the decision-maker and the executor respectively. The mean of $e(\hat{u}_i, u_i)$ for the decision-maker and the executor are approximately 0.16 and 0.06. Roughly, the larger variance of the decision-maker error is due to the large variety of conditions inducing the test drivers to decide to anticipate or postpone the braking. On the other hand, once the braking action has started, the executor replicates well the braking pedal trajectory.

E. Performance Test Campaign

After training and evaluating the decision-maker and the executor individually, we focused on the performance assessment of the VTD as a whole. Later, we synthetically characterized the VTD macroscopic behavior through some quantitative KPIs.

For the performance test described in this section, the VTD was deployed in a Hardware-In-the-Loop setup where the VTD's controls are sent to a virtual car via CAN. In detail, the test bench includes actual vehicle wiring and the main ECUs. Moreover, the vehicle and engine dynamics and the environment are simulated through a dedicated real-time emulator. As for the environment, we created digital twins of downhill roads, with possibly varying curvature, by using

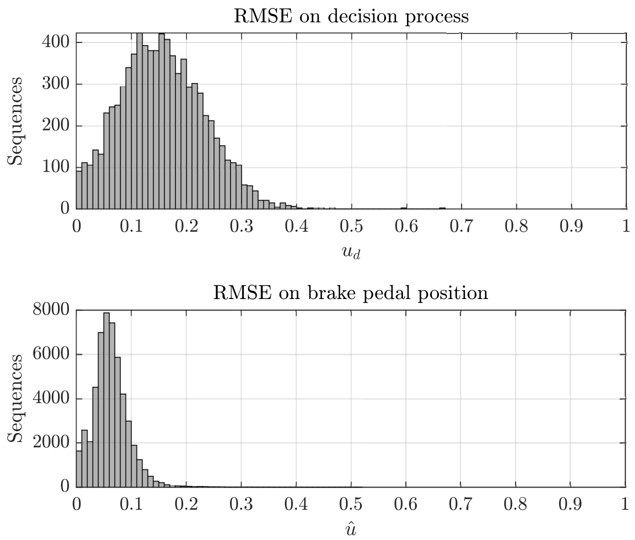

Fig. 4. Distribution of Root-MSE (RMSE) of Decision-Making Process (top) and Braking Execution (bottom) LSTM-NNs on their validation sets.

GNSS data. Vehicles and objects surrounding the ego vehicle were perceived via a 30°-field-of-view simulated camera.

We focused only on the braking manoeuvres to test the HDC functionalities. In detail, we defined $\mathcal{D}_p$ by selecting from $\mathcal{D}$ a family of $\mathcal{N}_p \subset \mathcal{M}$ examples associated with these testing conditions. Besides, we tested the VTD in a family of $\mathcal{N}_s \subset \mathbb{N}$ braking manoeuvres synthetically generated. Let $p_i \in \mathbb{R}$ be a performance parameter computed for the ith manoeuvre and define $\mathcal{P}_p := \{p_i\}_{i\in\mathcal{N}_p}$ and $\mathcal{P}_s := \{p_i\}_{i\in\mathcal{N}_s}$ as the sets of validation and synthetic parameters. Then, to compare the human and virtual test drivers, we defined the following absolute and relative KPI functions

$$\kappa^a(\mathcal{P}_p, \mathcal{P}_s) = \frac{1}{n_s}\sum_{i\in\mathcal{N}_s} p_i - \frac{1}{n_p}\sum_{j\in\mathcal{N}_p} p_j$$

$$\kappa^r(\mathcal{P}_p, \mathcal{P}_s) = n_p \frac{\kappa_i^a(\mathcal{P}_p, \mathcal{P}_s)}{\sum_{j\in\mathcal{N}_p} p_j}, \tag{7}$$

where n_v and n_s are the dimensions of $\mathcal{N}_v$ and $\mathcal{N}_s$. As for the choice of performance parameters, let $n_i \in \mathbb{N}$ and $\Delta T_i > 0$ be the number of samples and the sample time of the ith braking manoeuvre. Moreover, let $u_i(k)$, $v_i(k)$, $a_i(k)$ be the kth sample of the brake pedal position, the vehicle speed, and the vehicle acceleration of the ith manoeuvre, with $k = 1, \dots, n_i$. Then, we defined the mean and minimum acceleration (maximum decelerations), $a_i^{\text{mean}} := n_i^{-1}\sum_{k=1}^{n_i} a_i(k)$ and $a_i^{\text{min}} := \inf_{k=1,\dots,n_i} a_i(k)$, braking duration, $T_i := n_i \Delta T_i$, maximum brake pedal position, $u_i^{\text{max}} := \sup_{k=1,\dots,n_i} u_i(k)$, and initial and ending braking speed, $v_i^0 := v_i(1)$ and $v_i^T := v_i(n_i)$, for each $i = 1, \dots, n_v$. To exploit (7), we defined the sets $\mathcal{A}_\#^{\text{mean}} := \cup_{i\in\mathcal{N}_\#} a_i^{\text{mean}}$, $\mathcal{A}_\#^{\text{min}} := \cup_{i\in\mathcal{N}_\#} a_i^{\text{min}}$, $\mathcal{T}_\# := \cup_{i\in\mathcal{N}_\#} T_i$, $\mathcal{U}_\#^{\text{max}} := \cup_{i\in\mathcal{N}_\#} u_i^{\text{max}}$, $\mathcal{V}_\#^0 := \cup_{i\in\mathcal{N}_\#} v_i^0$, and $\mathcal{V}_\#^T := \cup_{i\in\mathcal{N}_\#} v_i^T$, for each $\# \in \{p, s\}$. The values of the KPIs determined on these sets are reported in Table I. Errors in the mean and maximum deceleration during braking manoeuvres, $\kappa^a(\mathcal{A}_v^{\text{mean}}, \mathcal{A}_s^{\text{mean}})$ and $\kappa^a(\mathcal{A}_v^{\text{min}}, \mathcal{A}_s^{\text{min}})$, and mean duration, $\kappa^a(\mathcal{T}_v, \mathcal{T}_s)$, show that the intensity of braking actions was comparable to the average behavior observable from the real-world data. Moreover, VTD was not too aggressive or too cautious, it did not slam on the

TABLE I
VTD PERFORMANCE IN BRAKING MANOEUVRES

KPI	Unit	Value
$\kappa^a(\mathcal{A}_v^{\text{mean}}, \mathcal{A}_s^{\text{mean}})$	[m/s^2]	0.02
$\kappa^a(\mathcal{A}_v^{\text{min}}, \mathcal{A}_s^{\text{min}})$	[m/s^2]	0.32
$\kappa^a(\mathcal{T}_v, \mathcal{T}_s)$	[s]	0.07
$\kappa^r(\mathcal{U}_v^{\max}, \mathcal{U}_s^{\max})$	[%]	3.88
$\kappa^r(\mathcal{V}_v^0, \mathcal{V}_s^0)$	[%]	9.32
$\kappa^r(\mathcal{V}_v^T, \mathcal{V}_s^T)$	[%]	2.58

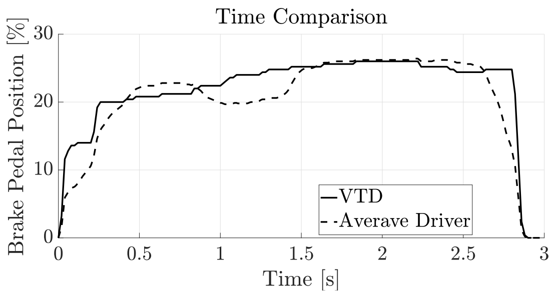

Fig. 5. Comparison between the average human test driver behavior and the VTD one, involved in the same performance test, regarding the braking response.

brakes or push the pedal too light or too heavy. This conclusion is corroborated by the error on the maximum position of the braking pedal, $\kappa^r(\mathcal{U}_v^{\max}, \mathcal{U}_s^{\max})$, which is lower than 4%. On the other hand, the small speed errors at the beginning of braking manoeuvres, $\kappa^r(\mathcal{V}_v^0, \mathcal{V}_s^0)$, suggest that VTD is as sensitive to speed as human drivers and that it starts braking in similar conditions. Similarly, small speed errors at the end of braking manoeuvres, $\kappa^r(\mathcal{V}_v^T, \mathcal{V}_s^T)$, suggest that VTD brakes as human drivers. In general, the relative errors in Table I are smaller than 10% and the absolute ones are not significant from a macroscopic point of view. The ability to replicate the human test driver brake pedal profile is appreciable in Figure 5 in which we superposed the VTD brake pedal to the average of all the human test driver brake pedals in the same VTD test conditions. Therefore, VTD can potentially be used in automated tests to emulate human braking actions.

IV. Conclusion and Further Developments

In this letter, we have described a new approach for automated tests without human drivers. As a first application, we focused on HDC testing. The obtained results show that the behavior of the proposed VTD is comparable with humans in terms of decision and execution of braking manoeuvres. As a further development, additional modules could be added to deal with other manoeuvres, aiming at an ultimate VTD able to cover the whole testing domain.

Acknowledgment

A special thanks to Dr. D. Balzer, Eng. G. Di Mare, Eng. I. Lassrich, Eng. G. Bordonaro, Eng. M. Angelini, and Eng. L. Reali for their valuable contribution to this project.

References

[1] C. Birchler, N. Ganz, S. Khatiri, A. Gambi, and S. Panichella, "Cost-effective simulation-based test selection in self-driving cars software," *Sci. Comput. Program.*, vol. 226, Mar. 2023, Art. no. 102926.
[2] C. C. Macadam, "Understanding and modeling the human driver," *Veh. Syst. Dyn.*, vol. 40, nos. 1–3, pp. 101–134, 2003. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1076/vesd.40.1.101.15875
[3] I. I. Delice and S. Ertugrul, "Intelligent modeling of human driver: A survey," in *Proc. IEEE Intell. Veh. Symp.*, 2007, pp. 648–651.
[4] N. M. Negash and J. Yang, "Driver Behavior modeling toward autonomous vehicles: Comprehensive review," *IEEE Access*, vol. 11, pp. 22788–22821, 2023.
[5] M. Plöchl and J. Edelmann, "Driver models in automobile dynamics application," *Veh. Syst. Dyn.*, vol. 45, nos. 7–8, pp. 699–741, 2007. [Online]. Available: https://doi.org/10.1080/00423110701432482
[6] D. Ni, "Limitations of current traffic models and strategies to address them," *Simulat. Model. Pract. Theory*, vol. 104, Nov. 2020, Art. no. 102137. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569190X20300769
[7] W. Wang et al., "Decision-making in driver-automation shared control: A review and perspectives," *IEEE/CAA J. Automatica Sinica*, vol. 7, no. 5, pp. 1289–1307, Sep. 2020.
[8] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, pp. 115–133, Dec. 1943.
[9] D. E. Rumelhart and J. L. McClelland, "Learning and relearning in Boltzmann machines," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, vol. 1. Cambridge, MA, USA: MIT Press, 1986, pp. 282–317.
[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[11] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, "Multi-digit number recognition from street view imagery using deep convolutional neural networks," 2013, *arXiv:1312.6082*.
[12] A. Díaz-Álvarez, M. Clavijo, F. Jiménez, E. Talavera, and F. Serradilla, "Modelling the human lane-change execution behaviour through multilayer perceptrons and convolutional neural networks," *Transp. Res. F, Traffic Psychol. Behav.*, vol. 56, pp. 134–148, Jul. 2018.
[13] L. Xu, J. Hu, H. Jiang, and W. Meng, "Establishing style-oriented driver models by imitating human driving behaviors," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 5, pp. 2522–2530, Oct. 2015.
[14] X. Wang, Y. Guo, C. Bai, Q. Yuan, S. Liu, and J. Han, "Driver's intention identification with the involvement of emotional factors in two-lane roads," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 11, pp. 6866–6874, Nov. 2021.
[15] Y. Xing, C. Lv, Y. Liu, Y. Zhao, D. Cao, and S. Kawahara, "Hybrid-learning-based driver steering intention prediction using neuromuscular dynamics," *IEEE Trans. Ind. Electron.*, vol. 69, no. 2, pp. 1750–1761, Feb. 2022.
[16] T. Djukic et al. "Safe-up project deliverable D.2.4: Definition of the future use cases: Scope and data to build digital twins of use cases." Safe-UP Project. 2022. [Online]. Available: https://www.safe-up.eu/deliverables
[17] L. Ljung, *System Identification: Theory for the User*. Upper Saddle River, NJ, USA: Prentice Hall PRT, 1999.
[18] E. Phaisangittisagul, "An analysis of the regularization between L2 and dropout in single hidden layer neural network," in *Proc. 7th Int. Conf. Intell. Syst., Model. Simulat. (ISMS)*, 2016, pp. 174–179.
[19] A. Barakat and P. Bianchi, "Convergence and dynamical behavior of the ADAM algorithm for nonconvex stochastic optimization," *SIAM J. Optim.*, vol. 31, no. 1, pp. 244–274, 2021. [Online]. Available: https://doi.org/10.1137/19M1263443

Open Access funding provided by 'Alma Mater Studiorum – Universitá di Bologna' within the CRUI CARE Agreement