IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 29 November 2023, accepted 12 January 2024, date of publication 22 January 2024, date of current version 29 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3356715

RESEARCH ARTICLE

Development and Evaluation of Artificial Neural Networks for Real-World Data-Driven Virtual Sensors in Vehicle Suspension

ELDAR ŠABANOVIČ[1], (Senior Member, IEEE), PAULIUS KOJIS[1], VALENTIN IVANOV[1], (Senior Member, IEEE), MIGUEL DHAENS[2], AND VIKTOR SKRICKIJ[1]

[1]Transport Engineering Faculty, Vilnius Gediminas Technical University, 10223 Vilnius, Lithuania
[2]Tenneco Automotive Europe, 3800 Sint-Truiden, Belgium

Corresponding author: Eldar Šabanovič (eldar.sabanovic@vilniustech.lt)

This work was supported in part by the Research Council of Lithuania under Project S-MIP-23-120; and in part by the European Union Horizon 2020 Framework Program, Marie Skłodowska-Curie Actions, under Grant 872907.

ABSTRACT Vehicle comfort, handling, and stability can be improved by a semi-active suspension with advanced control algorithms using the vertical velocities of the sprung mass (SM) and the unsprung masses (UMs) as inputs. Displacement and acceleration sensors are often used to estimate vertical velocities of UMs. However, these sensors are expensive and are susceptible to degradation. Virtual sensors (VS) have been proposed as a solution, and previous research using simulation data has shown that artificial neural networks (ANNs) can provide usable UM vertical velocity estimates. This study aims at finding ANNs structure and input sample window size to achieve best performance on real-world data. Novel dataset was created and used to test VSs based on eight structures of ANNs combining multilayer perceptron, convolutional neural network, long-term short-term memory (LSTM), and bidirectional LSTM layers. This article presents the results of 104 combinations of ANN structure and sample window size, which required 6240 training sessions. A Bayesian search was used to tune the hyperparameters of ANNs' layers minimizing the root-mean-square error (RMSE) of estimations on the validation data, while a grid search was used to select the sample window size that minimizes RMSE of estimation on test data, ensuring that selected combinations are well generalized. The VS based on ANN with convolutional layers, achieved the lowest RMSE of 0.0210 m/s, and processing time of 0.421 ms for a window size of 23 samples while estimating vertical velocities of vehicle UMs from real-world data.

INDEX TERMS Artificial neural networks, data-driven modeling, virtual sensor, velocity estimation, vehicle suspension.

I. INTRODUCTION

Nowadays, the automotive industry is strongly focusing on automated driving technologies [1], so the ride comfort is becoming increasingly important. In particular, fatigue and motion sickness may be more critical in automated vehicles as the driver is less involved in vehicle control [2]. Occupants may also engage in various activities while driving, such as working on a laptop or reading, leading to higher demands on ride comfort [3]. Some studies show that it would be beneficial to achieve a comfort level in a car similar to that of trains [3], [4]. Nevertheless, various control algorithms are developed for driver assistance systems that can be used in conventional vehicles to improve comfort. These algorithms involve adaptive, semi-active [5], [6], [7], and proactive suspension with road surface preview [8].

The ongoing challenge to improve comfort, stability, and handling without increasing hardware costs requires

The associate editor coordinating the review of this manuscript and approving it for publication was Junho Hong.

continuous enhancement of chassis control performance [9]. For example, semi-active and active actuators must replace passive vehicle suspension components to provide satisfactory comfort and contribute to better vehicle handling [10]. Implementation of active components requires advanced suspension control and state measurement algorithms. Active and semi-active actuators can change the system characteristics by changing the damping force. Commonly, industrial control solutions such as Skyhook or hybrid and other state of the art algorithms use vertical velocities of the unsprung masses (UMs) and the sprung mass (SM) as inputs, and they are estimated instead of direct measurement [11], [12]. No robust sensor is available at a reasonable price for vertical velocity measurements of SM and UM. Vertical velocities of SM are estimated using IMU's measurements and Kalman filters. IMU sensor placed on SM is used by other systems such as dynamic stability control (DSC). Displacement [13] and acceleration sensors [14] are commonly used and velocity of UM is estimated by fusing measurements of these sensors. This requires installation of additional sensors at UMs, leading to higher overall costs, packaging issues, and robustness problems. UM displacement sensors are sensitive to environmental influences and wear with use. Acceleration sensors measurements are prone to integration bias, and installation of such sensors onto each wheel hub is complex. A possible solution to this problem is the use of UM virtual sensors (VSs) [15]. Such sensor could estimate UM vertical velocity using only data from sensors installed on SM.

A VS is a software algorithm that generates signals by combining and processing data received from physical sensors and estimators [16]. The generated data is fed into complex functions or applications [17]. The VS can be model-based and data-driven [18]. In model-based approaches, mathematical models are used to define the relationship between input and output variables. Automotive applications often use variations of the Kalman filter (KF) for this type of VS [19]. For example, the KF in [20] was used to estimate the vehicle sideslip angle, and the tire forces were estimated in [21]. The study [22] used the KF for suspension state estimation. However, due to the nonlinearity of vehicle components, it can be challenging to use mathematical equations to create an accurate VS model [23]. On the other hand, the data-driven VS relies solely on recorded data obtained from observation of system operation [24]. Practical implementations of data-driven VS include multivariate statistical methods and artificial intelligence methods.

Data-driven VS developed for vehicle suspensions has been investigated in previous studies [15], [23], [24], [25]. In [24], the authors presented a data-driven approach based on deep learning (DL) for estimating the road profile height and state variables of vertical displacement and velocity of vehicle UMs using onboard sensors. The proposed VS was compared with the extended KF and the static nonlinear autoregressive exogenous model. The performance results were in favor of the proposed VS model. However, the algorithm was only tested in a simulation environment with a narrow range of driving conditions. Therefore, it is unclear how such sensor would perform with real-world data.

Previous studies [23], [25] provided promising results using data-driven VS based on artificial neural networks (ANNs) and specifically deep neural networks (DNNs) for estimation of UMs' vertical velocities.

The motivation behind this research rises from the observed deficiency in the development and testing of the data-driven VS for UM vertical velocity using real-world datasets. Furthermore, we aim to compare and select ANN structure and determine the most suitable input sample window size to achieve the highest estimation accuracy.

Thus, the objective of this study is to develop and compare data-driven ANN-based VSs (described in Section II) on a real-world dataset. This algorithm is part of vehicle state estimation, which is required as input for feedback control algorithms, that are not part of this article. The research presented in this article includes determining the optimal sampling window length and hyperparameter combinations for each of the selected ANN structures. In order to deal with the lack of datasets, a new dataset was created using raw data collected at the proving ground and under urban driving conditions from a vehicle demonstrator equipped with a semi-active suspension. The ANN training process was repeated 60 times for each combination of structure and sample window size, to select the best hyperparameters using a Bayesian search. Overfitting of the ANN is prevented by selecting the model iteration with the best root mean squared error (RMSE) on validation data from 60 training iterations.

The contributions of this study include:

- Preparation of a real-world dataset for the development of a VS for the vertical velocity of UMs: A major contribution lies in the careful compilation of a real-world dataset tailored for the development and rigorous testing of estimators for the vertical velocity of UM.
- Systematization of tasks and methodologies for the creation of data-driven VSs: This provides a clarification of tasks and outlines a comprehensive methodology for the creation of data-driven VSs for vehicle suspension and sheds light on new approaches in this domain.
- Development and selection of best type and optimal structure of ANN for VS of UMs vertical velocity: Through extensive experimentation, this research identifies the most appropriate type and structures of ANNs that are best suited for the prediction of UM vertical velocity.
- Determination of the optimal input window size for real-time VS performance: The critical dimension of input sampling window size for the tested ANN structures is introduced, which significantly affects the estimation accuracy.
- Analysis of the influence of the input signals on the RMSE of output: In this study, the influence of input

signals on the resulting RMSE of the estimations is rigorously investigated and quantified.
- Testing the impact of input signals' means compensation on output RMSE: The significance of input signal means compensation is evaluated, providing valuable insight into the impact of such compensation on the performance and estimation accuracy of VS.

These contributions collectively advance the state of knowledge in the field of UMs' vertical velocities estimation, offering novel perspectives and methodologies that significantly enhance task understanding and solving capabilities.

The rest of this paper consists of four sections, excluding the introduction. Section II describes experimental setup and experiments for the real-world data collection, collected signals and dataset preparation, and development of VS, including testing the importance of inputs, the eight ANN structures, and hyperparameter search methods. In Section III, the experimental results are presented and analyzed, and the best solution is found. In Section IV, the main results of the research are discussed and the conclusions are drawn.

II. DATA COLLECTION AND NEURAL NETWORK STRUCTURES

A. EXPERIMENTAL SETUP FOR DATA COLLECTION

This research aims to create VS that would replace UM displacement and acceleration sensors (see Figure 1) currently used for estimation of UMs' vertical velocities that are used in suspension control algorithms. There is a lack of a real-world dataset suitable for supervised learning of ANN and performance evaluation. Therefore, the original data needs to be collected using vehicle demonstrator and dataset prepared.

Audi A6 (2019 model year) with a semi-active suspension prototype developed by Tenneco was used in this research. This demonstrator vehicle was equipped with a dSPACE real-time target machine (RTTM) that runs a suspension controller that uses UM vertical velocity as input. RTTM is also capable of recording sensor, estimated, and algorithms' output data. Therefore, it is used for experimental data acquisition. The data is fed to the connected computer that records it into files. The data is logged at a fixed sample rate of 100 Hz into the recording file.

In order to enable the collecting of required data, additional sensors were installed on the vehicle; they included an IMU placed in the center of gravity of the vehicle, optical flow sensor (OFSs) mounted on the rear left side door for sideslip angle measurement, and UM vertical displacement sensors were mounted between lower suspension links and vehicle body in parallel to dampers. Furthermore, access to the vehicle's controller area network (CAN) channels was granted, this provided access to all standard in-vehicle sensors, which were in their standard locations in the vehicle.

An overview of the system is given in Figure 1. There the sensors are connected to the RTTM which implements suspension controller, signal recording, and proposed UMs' vertical velocities VS. It shows how front-left (FL), front-right (FR), rear-left (RL), rear-right (RR) UM acceleration and displacement sensors can be replaced or duplicated by VS for FL, RF, RL, RR.

Proposed UM vertical velocity VS is meant to use in RTTM instead of the currently used algorithm for estimation from measured UM vertical displacement. This would allow to remove displacement sensors if estimation using VS on real-world data proves to be good enough compared to current estimation.

B. EXPERIMENTS FOR DATA COLLECTION

The dataset was created based on test data collected on the Lommel proving ground using vehicle demonstrator. The procedures below describe the tests performed to collect the data. These tests are commonly used for vehicle

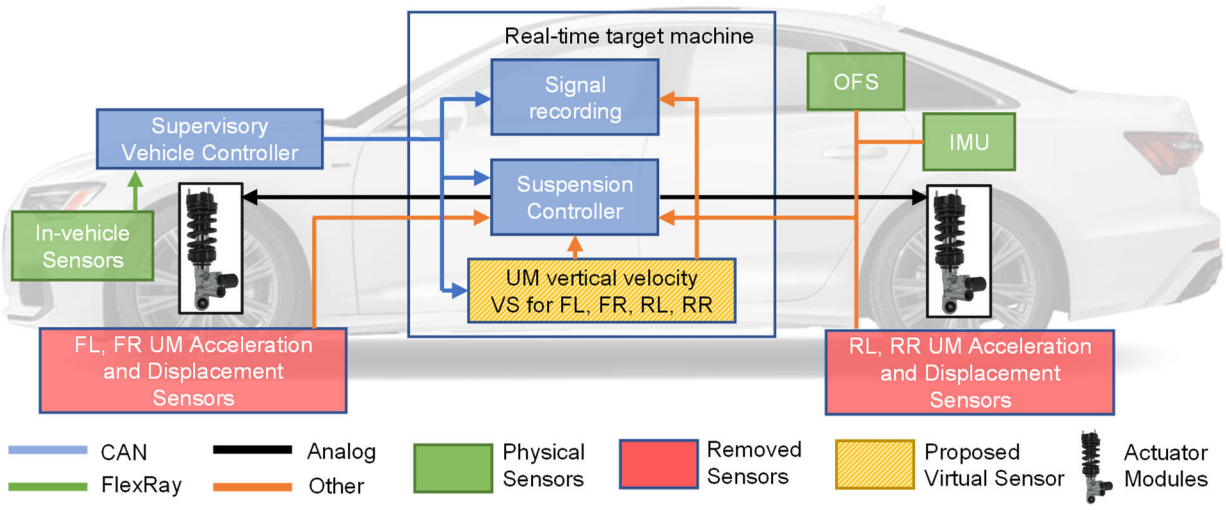


FIGURE 1. System overview.

handling, comfort, and stability studies, therefore recorded in-vehicle sensor signal data should provide enough information for training, validation and testing datasets. The data was recorded for different types of tests: acceleration and braking; skid pad, step steer, double-step steer, obstacle avoidance, sine with dwell, sinusoidal steer, and comfort tests. Also, data was collected under urban driving conditions.

In the acceleration and braking test, maximum acceleration is applied from a standstill in a straight line, and at 100 km/h, the vehicle is brought to a standstill with maximum braking force.

In the skid test (ISO 7975:2019), the driver controls the vehicle at a constant radius of turn. The tests started from velocity of 10 km/h. The vehicle velocity gradually increased. This test was performed up to the velocity level at which the driver could no longer keep the vehicle on the target trajectory.

In the step steering test (ISO 7401:2011), the driver accelerated the vehicle in a straight line until a velocity of 100 km/h was reached. After that, the accelerator pedal was held constant, and the driver performed a stepped steering wheel input (counterclockwise or clockwise) of 100° with a rate of 400°/s. The steering wheel angle was held at 100° for at least 2 s after the first actuation.

In the double-step steering test (ISO 17288-1:2011), the driver accelerated the vehicle in a straight line until a velocity of 100 km/h was reached. Afterwards, the accelerator pedal was held constant, and the driver performed a stepped steering wheel input (counterclockwise) of 100° at a rate of 400°/s. The steering wheel angle was held at 100° for 2 s after the initial actuation. After 2 s, the steering wheel was rotated to -100° and maintained for 2 s. After that, the driver set the steering wheel back to 0°.

The obstacle avoidance test (ISO 3888-2:2011) is a dynamic maneuver in which a vehicle moves rapidly from its original lane to adjacent road lane and returns to the original lane without exceeding lane limits. The goal was to ensure that the vehicle achieves a specific sequence of alternating high lateral acceleration values. During the test, the driver holds the accelerator pedal constantly. The initial longitudinal velocity of the vehicle was measured to ensure the repeatability of the test procedure.

In the sine-with-dwell test (ISO 19365:2016), the vehicle was accelerated to a velocity of just over 80 km/h. Afterwards, a constant accelerator pedal position was held. The steering wheel input has a waveform of a sine wave with a frequency of 0.7 Hz that pauses for 0.5 seconds after reaching the second peak.

The sinusoidal steering test procedure includes driving conditions where the vehicle reaches a lateral acceleration of about 6 m/s^2, which is achieved with a steering wheel amplitude of about 50° at a vehicle velocity of about 80 km/h. Stronger inputs can also be made up to and beyond the handling limit, e.g., steering wheel angle of 100° for a more aggressive maneuver, but not at the handling limit, steering wheel angle of 150° for a maneuver at handling limit. The driver accelerates the vehicle in a straight line until a constant velocity of 80 km/h is reached. The accelerator pedal is held at a constant value. After that, the driver gives a sinusoidal, wave-like steering wheel input with a predefined magnitude. The frequency of the sine wave is approximately 1 Hz. The steering input lasts for two cycles.

The comfort tests included driving on a road with Belgian pavement, driving on a road with bumps at different velocities, and driving on a road with high-class (D-F) pavement irregularities. Driving is performed at a constant velocity in range of 25-70 km/h. These tests represent good and bad driving conditions.

The collected data serves as a comprehensive basis for building DL dataset due to the wide range of real-world driving scenarios. The diversity of these tests, provide valuable data, that is suitable for development of data-driven VS not only for UM vertical velocity but also for other signals that are recorded.

C. SIGNALS AND DATASET PREPARATION

In this paper, estimation of UM vertical velocity using VS is determined as a time series regression task with multiple inputs and multiple outputs (for each wheel simultaneously) when a current and a certain number of previous samples are available.

The main characteristics of each input signal must be described to understand the relationship between input and output and their usability. The values of the signals have different characteristics in terms of continuity, scale, and mean value. Discontinuous signals can be useful as switching signals for changing the behavior of the ANN; however, most ANNs that produce continuous output signals will benefit more from continuous input signals that can be used directly to form the output signal. The mean values of the signals are used to offset that signal towards 0, because it is the point of nonlinear part of selected activation functions. The standard deviation shows the extent to which the signal varies and is a good metric of the signal scale. The scales of the signals must be similar to speed up the DL process; therefore, the signals are normalized by dividing them by the standard deviation. Next, the signals are described and main characteristics are provided in Table 1.

Driver torque demand is a signal available from the drive-by-wire gas pedal. The value of this signal is important to engine and DSC systems. Although not directly, it does affect the longitudinal acceleration that is measured by IMU. The acceleration induces change of pitch angle, thus unloads and loads the suspension and affects the vertical displacement and velocity of the UM.

The DSC regulation signal indicates the system's engagement in braking or engine torque reduction. Active DSC may change the load on opposite sides and ends of suspension, thus influencing the UM vertical velocity and displacement.

The master cylinder pressure signal indicates the applied braking pressure that directly correlates with deceleration.

This deceleration modulates the vertical load distribution, thereby affecting the UM's vertical displacement and velocity.

TABLE 1. Input signals, sources, and statistical characteristics.

Parameter	Source	Units	Mean	Std. dev.
Driver torque requirement	CAN	-	86.1	32.3
DSC regulation	CAN	-	$6.64 \cdot 10^{-4}$	0.0258
Master cylinder pressure	CAN	-	0.0138	0.459
Steering angle	CAN	-	-0.0401	0.761
Steering angle direction	CAN	-	0.719	0.450
Steering angle optimized	CAN	-	-0.00630	3.15
Vehicle velocity	CAN	m/s	21.2	2.47
FL wheel velocity	CAN	km/h	76.3	8.84
FR wheel velocity	CAN	km/h	76.2	9.01
RL wheel velocity	CAN	km/h	76.1	8.87
RR wheel velocity	CAN	km/h	76.0	9.19
Acceleration X axis	IMU	*g*	-0.0167	0.0368
Acceleration Y axis	IMU	*g*	-0.0244	0.393
Acceleration Z axis	IMU	*g*	0.995	0.0448
Roll rate	IMU	deg/s	-0.0563	5.37
Yaw rate	IMU	deg/s	-0.8326	13.0
Longitudinal velocity	OFS	cm/s	2140	248
Transversal velocity	OFS	cm/s	6.85	119
Body sideslip angle	OFS	-	204	2180

The steering angle and direction provide information about the steering input of the driver. This information is also used for the lateral acceleration and the distribution of the load on the left and right sides of the vehicle suspension.

The optimized steering angle is the target steering angle calculated by the vehicle taking into account user input and the requirements of other onboard systems.

Vehicle velocity information is important for ride control when the vehicle overcomes surface irregularities, bends, potholes, and turns. It provides information on the extent to which lateral and vertical acceleration can be expected and how quickly the road impact on the front suspension reaches the rear suspension.

The wheels' velocities are provided by the anti-lock braking system over CAN. These signals partly duplicate vehicle velocity, but also adds information about friction.

The X, Y, and Z axes accelerations, roll, and yaw rates obtained from IMU provide information about the state of the vehicle: orientation, acceleration, and rotation due to driver input and external factors.

The body sideslip angle provides information about the difference between the vehicle's heading and its actual direction of travel, usually is used in DSC.

These parameters, along with the vertical velocities of SM and UM, are used in advanced control algorithms for semi-active suspensions. However, not all inputs are equally important in estimating the desired output values. Therefore, it is necessary to evaluate their significance when processed by ANN-based VS. This was done using value replacement methods and evaluation of the RMSE change on the test dataset (see Section III (b)).

The output signals are the UM's vertical velocities for each wheel. The ground truth UM's vertical velocities were estimated from the vertical wheel displacement in reference to full suspension extension point on the experimental vehicle. At first, the signals from the vertical wheel displacement sensors were filtered at 50 Hz using a MATLAB Butterworth low-pass filter. Afterwards, the difference between the current and the last sample was calculated. Finally, the resulting difference between the samples is divided by the sampling time to obtain the vertical velocity of the wheel. This is the same filter used in the real-time processing of vehicle suspension control units. As it does use current and last samples, a delay of about half of sample period is introduced into the filtered signal.

Multiple pasts values of the input signals are needed to estimate the UM velocity at a given time as they provide information about the change in the signal. Therefore, the window length of the input samples is an important parameter as it affects the amount of signal memory available to the ANNs. The presence of past samples allows the learning of signal features at lower frequencies. At a fixed sampling rate of 100 Hz, the tested sample window sizes of 3 to 51 samples provide a signal buffer of 30-510 ms. This allows real-time signal processing using first-in-first-out buffers, with the newest data point added and the oldest removed at each sampling period. The results in the Section III show that the processing time is considerably shorter than sampling period. Compared to low-pass filters, the proposed ANNs avoid problems related to delays that directly depend on the sample window size, as ANNs predict current state from last and current input signals based on learned model.

The total number of samples included in the dataset is about 393 thousand. The samples are randomly divided into 272 thousand (70%) in the training set, 61 thousand (15%) in the validation set, and 60 thousand (15%) in the test set at the experimental record level. Therefore, all samples belonging to one recording were assigned to one of the parts and not seen in the other part, which prevented very similar samples occurring in all parts in due to slow signal changes compared to the sampling frequency. Also, this is only possible way to split data, as most driving action happens in the middle of the recording. The validation samples were used for RMSE minimization during training and Bayesian search, and the testing samples were used to compare ANN structures and sample window size combinations.

A grid search was applied to select the sample window size for each ANN structure to systematically explore the different combinations and determine the best configuration. Grid search is a widely used method for implementing structure and parameter tuning in machine learning. The grid values for the sample window size were selected based on performance variation to ensure that a larger window length was near the minimum RMSE point. Window sizes of 3, 5, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 35 and 51 samples were selected because 3 samples are the least number of samples that can be used to predict the change of UM movement direction and 51 samples provide information of the lowest frequency that is important for suspension control.

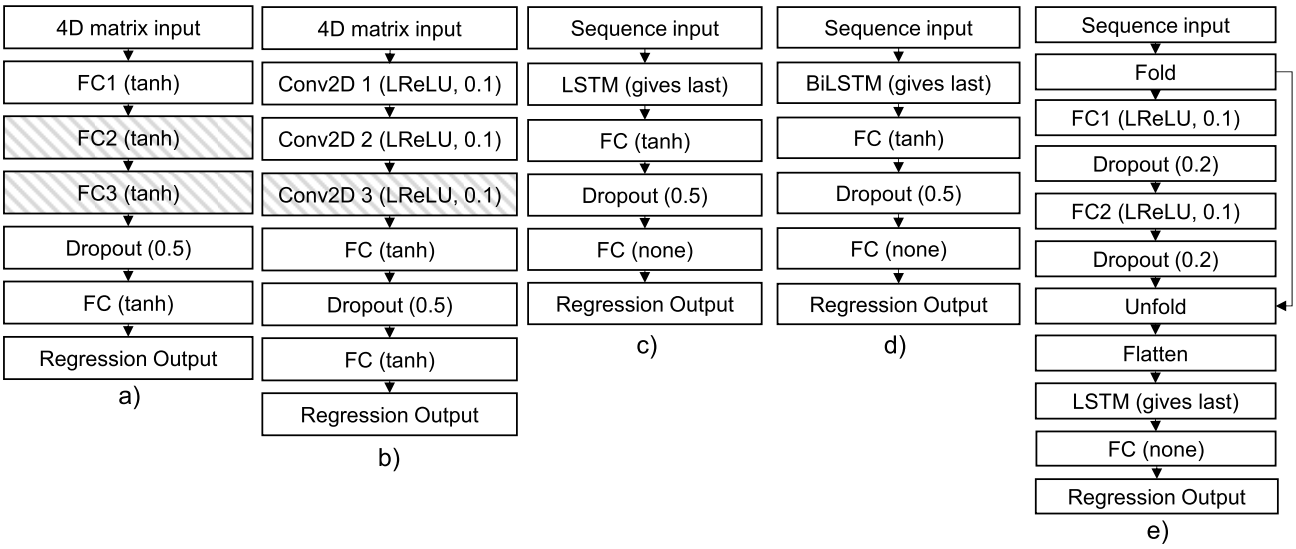


FIGURE 2. Structures of used ANNs: a) 1-3 hidden layers MLP; b) 2-3 layers CNN; c) LSTM; d) BiLSTM; e) MLP-LSTM.

More than 51 sample window size was not tested as it would require even computational time while providing a diminishing increase in accuracy.

D. STRUCTURES OF ARTIFICIAL NEURAL NETWORKS

The selection of an appropriate network type, application domain, and hardware imposes several requirements. First, an algorithm of ANN-based VS should be implemented using MATLAB and Simulink. Second, it should be compiled for a specific RTTM. Third, the compiled algorithm should run one cycle in less than 10 ms, because input signals are sampled at 100 Hz. The requirements limit the types of neural layers and scale of the ANN (including number of layers and number of neurons in them). Therefore, small task-specific ANNs were developed instead of using well-known backbone networks such as ResNet [26]. For the development of VS, ANNs with multilayer perceptron (MLP), convolutional neural network (CNN), long short-term memory (LSTM), bidirectional LSTM (BiLSTM) layers, and MLP-LSTM combination were implemented and compared.

Structures of the networks are shown in Figure 2. There hatched blocks are part of a variant of the same type of network with different number of layers. In Figure 2, only fixed properties of the layers are shown, other important parameters (neurons/units, convolutional kernel's size and stride) are provided in Tables 2-9 and 11. The window size and network hyperparameters were selected using Bayesian and grid searches. This process is described further in Section II (e). Next, each ANN type is explained.

The MLP structure is the simplest and most flexible (see Figure 2. a). It includes one or more hidden layers of neurons. Each neuron has a weighted connection to all neurons in the previous layer. The connection weights were changed during the learning process. The disadvantage of this flexibility is the high computational complexity since each connection requires an additional multiplication. Many multiplications and additions are performed in the planned task; therefore, using a matrix form of the equations is more efficient, which can be accelerated by a multiple data instruction operation in modern processors, especially in graphical processing units (GPUs). In this context, an MLP layer with an arbitrary number of neurons can be expressed by the following equation:

$$\boldsymbol{y} = \sigma(\boldsymbol{W} \times \boldsymbol{x} + \boldsymbol{b}), \tag{1}$$

where y is the output vector; $\boldsymbol{W}$ is the learnable input connection weight matrix of shape (k, n), where k is the number of neurons, n is the length of the input vector; $\boldsymbol{x}$ is the input vector of shape (n, 1), where n is the length of the input vector; $\boldsymbol{b}$ is the bias vector; and σ is the nonlinear activation function. The hyperbolic tangent (tanh) function was used as an activation function for all the MLP layers, excluding the last function that did not use activation. This equation can be simplified by concatenating the bias vector with the weight matrix and adding one to the input vector $\boldsymbol{x}$.This simplifies Equation (1) into a single matrix multiplication as follows:

$$\boldsymbol{y} = \sigma(\boldsymbol{W} \times \boldsymbol{x}), \tag{2}$$

where, $\boldsymbol{W}$ is the learnable input connections and bias weights' matrix of shape (k, n), where k is the number of neurons, n is the length of the input vector + 1; $\boldsymbol{x}$ is the input vector of shape $(n + 1, 1)$ because of the added value of 1 instead of a bias variable, where n is the length of the input vector. This equation can be processed faster because only dot product and activation operations are required. Most machine learning frameworks abstract related mathematical operations into high-level functions or ANN building blocks and use optimized code for processing. The described combination of weighting matrix and bias vector is done within functions,

and an input vector is a single input of such functions. An MLP layer in a complex ANN is often referred to as a fully connected layer (FC) because neurons in the layer have weighted connections with neurons in the previous layer. MLPs with up to 3 FC layers and different numbers of neuron units were tested, and the best combinations were selected for comparison with other architectures.

CNNs have emerged as a powerful class of DL models, particularly due to their exceptional performance in image analysis and recognition [28]. In recent years, their applicability has been extended to various domains, including signal processing and VS, due to their ability to extract relevant features from input data [29]. They are particularly effective in capturing local patterns and hierarchical representations from such data [30]. Therefore, it was decided to test a CNN in VS.

In a CNN, each convolutional layer consists of a set of adaptive filters that are convolved with the input data to extract relevant features. These filters capture local patterns in the data by performing element-wise multiplication and aggregating the results [31]. Pooling layers can be then used to shrink the feature maps and reduce their spatial extent while preserving the most relevant information [32]. The extracted features are then passed to one or more fully linked layers, which perform classification or regression tasks based on the learned representations [33].

A CNN layer with any number of units can be described by the following equation [34]:

$$X_{n,i,j} = \sum_{n=1}^{N} \left(\sum_{h=1}^{H} \sum_{w=1}^{W} \sum_{c=1}^{C} \times W_{n,w,h} \cdot A_{si+w,sj+h,c} \right) + B_n, \quad (3)$$

where $X_{n,i,j}$ is the value of the output feature map matrix of the convolutional layer (before activation function) at index n, i, j (n – index of the convolutional kernel; i – index of the row in feature map matrix; j – index of the column in feature map matrix); $W_{n,h,w}$ – value of kernel filter at index n, h, w (n – index of the convolutional kernel; h – index of the row in the convolutional kernel; w – index of the column in the convolutional kernel; $A_{si+w,sj+h,c}$ value of input array with index $si + w, sj + h, c$ (s stride of kernel; c channel of input array); B_n – bias of convolutional neuron with index n; N – number of convolutional neurons in the layer; H number of kernel rows; W number of kernel columns; C – number of channels in the filter that is the same as channels count of the input array. Each signal has own channels.

In this study, a CNN-based DNN architecture tailored for this application is proposed to meet the real-time data processing requirements of a VS. The DNN consists of 2 or 3 convolutional layers, each followed by a nonlinear activation function of a leaky rectified linear unit (LReLU) to extract the essential features of signals in the time domain and the relationships between signals as images in 2D space (see Figure 2. b)). The LReLU can be described by the following equation:

$$\boldsymbol{f}_t = \max(a\boldsymbol{X}_t, \boldsymbol{X}_t), \quad (4)$$

where a is the settable coefficient of steepness of the activation function in the negative region of the input values (a was set to 0.1).

LReLU avoids the problem of ANNs not learning when using rectified linear unit [35]. Zero input values cause the zero-gradient problem for negative $\boldsymbol{X}_t$ values. The output of the CNN layers is converted to a vector and given to FC layers, allowing the network to learn and make predictions at a higher level, similar to an MLP.

In the case of VS, the inputs to MLP and CNN are provided as W×M arrays, where W is the sample window size and M is the number of sensor inputs. These inputs are passed to the ANN in each sampling step and provide the output corresponding to the last sample. In this way, the output delay is minimized to the ANN processing time.

Unlike MLP or CNN, the LSTM stores its last state and output and uses them as inputs for decision making when processing the next sample in the sequence (see Figure 2. c)). The LSTM inputs were supplied as a W-length sequence of vectors, the length of which corresponds to the number of M-sensor inputs. The LSTM consists of a forget, input, output, and cell input gate (see Figure 3). The following equations can describe an LSTM layer with any number of units

$$\boldsymbol{f}_t = \sigma_g(\boldsymbol{W}_f \times \boldsymbol{x}_t + \boldsymbol{U}_f \times \boldsymbol{h}_{t-1} + \boldsymbol{b}_f), \quad (5)$$
$$\boldsymbol{i}_t = \sigma_g(\boldsymbol{W}_i \times \boldsymbol{x}_t + \boldsymbol{U}_i \times \boldsymbol{h}_{t-1} + \boldsymbol{b}_i), \quad (6)$$
$$\boldsymbol{o}_t = \sigma_g(\boldsymbol{W}_o \times \boldsymbol{x}_t + \boldsymbol{U}_o \times \boldsymbol{h}_{t-1} + \boldsymbol{b}_o), \quad (7)$$
$$\hat{c}_t = \sigma_c(\boldsymbol{W}_c \times \boldsymbol{x}_t + \boldsymbol{U}_c \times \boldsymbol{h}_{t-1} + \boldsymbol{b}_c), \quad (8)$$
$$\boldsymbol{c}_t = \boldsymbol{f}_t \circ \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \circ \hat{c}_t, \quad (9)$$
$$\boldsymbol{h}_t = \boldsymbol{o}_t \circ \sigma_c(\boldsymbol{c}_t), \quad (10)$$

where $\boldsymbol{f}_t$ – the forget gate's result; $\boldsymbol{i}_t$ – the input/update gate's result; $\boldsymbol{o}_t$ – the output gate's result; $\hat{\boldsymbol{c}}_t$ – the cell input result; $\boldsymbol{c}_t$ – the memory cell state; $\boldsymbol{W}_f$, $\boldsymbol{W}_i$, $\boldsymbol{W}_o$, $\boldsymbol{W}_c$ – the learnable input connection weights' matrices for forget, input, output, and memory cell gates; $\boldsymbol{U}_f$, $\boldsymbol{U}_i$, $\boldsymbol{U}_o$, $\boldsymbol{U}_c$ – the learnable recurrent connection weight matrices for each gate; $\boldsymbol{x}_t$ – the input data matrix; $\boldsymbol{h}_{t-1}$ – the recurrent data matrix; t – the sample's number in input sequence; ∘ - the element-wise multiplication of matrixes (Hadamard product);σ_g – sigmoid activation function; σ_c – tanh activation function.

The LSTM can be unrolled in time to observe the interfaces between successive input samples. Each time, the same LSTM unit processes the next data item in a sequence of length W and an input vector of length N. In such cases, t varies from 0 to W-1.

BiLSTM is similar to LSTM in that it uses the same structure (see Figure 2. d)). Unlike the LSTM unit, a BiLSTM unit uses two connected LSTM structures to obtain the last sample and outputs the current sample, while the other unit obtains the next sample and outputs the current sample. The outputs of the two LSTM structures for the same sample are linked. The output at each step depends not only on the past samples in the sequence, but also on the next data sample.

Such processing is possible only if all samples of the input sequence are available.

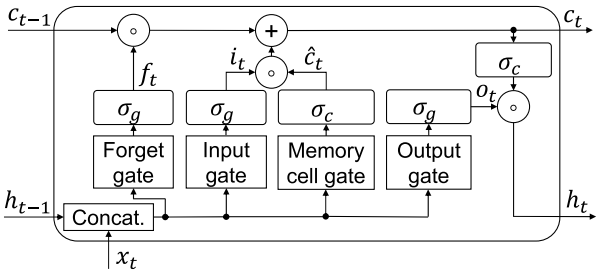


FIGURE 3. The internal structure of one LSTM unit.

The last structure type tested was MLP-LSTM, a combination of MLP and LSTM (see Figure 2. e)). In this model, the sequence data is first convolved and preprocessed with MLP before being deconvolved and processed with LSTM. This structure was tested to determine if an additional feature extraction layer could be better combined with the memory capability of LSTM.

E. ANN TYPE AND STRUCTURE SELECTION

Eight described above ANN structures with various types and numbers of layers were selected for investigation.

After determining all possible combinations of the selected sample window sizes and structures, a Bayesian search is applied to fine-tune the hyperparameters of the convolutional and fully connected layers. Bayesian search is a probabilistic optimization method that iteratively updates a probability distribution over hyperparameters based on the observed performance of the model. By leveraging prior knowledge and iteratively refining the distribution, a Bayesian search aims to efficiently explore the hyperparameter space and identify the most promising hyperparameter values.

The advantage of the Bayesian search for hyperparameters lies in its ability to handle a limited amount of data and efficiently use computational resources. It adapts to the information gained during the search process and dynamically focuses on promising regions of the hyperparameter space. This adaptive nature allows the Bayesian search to converge to optimal hyperparameters with fewer evaluations compared to other methods, such as grid search.

However, it is important to note that Bayesian search typically requires more computational resources and time than grid search because of its iterative nature and the need to update the probability distributions [27]. Additionally, the performance of the Bayesian search depends heavily on the choice of the prior distribution and acquisition function used to guide the search process.

The training process was repeated at least 60 times for each ANN and window-size combination structure using ADAM optimization algorithm to select connection weights in neural networks and settings of LSTM and BiLSTM memories, while performing the Bayesian search for selection of hyperparameters. Each training session lasted for 30 epochs. One epoch is a training cycle for all the training dataset samples. The selected initial learning rate was 0.001 with one learning rate drop by a factor of 0.2 after 20 epochs. The learning rate reduction enables more accurate minimization in the selected minimum point of the cost function. Also, the gradient rate threshold of 1 was set, to reduce the effect of high error during the start of the learning process. The data samples were shuffled before each epoch to improve the stability of the learning process. This article presents the results of 104 ANN structure and sample window size combinations, which required 6240 training sessions. The testing of RMSE and processing duration were carried out for each pair of ANN type and sample window size after training. The test set is being processed using a central processing unit (CPU) only and a batch size of 1, which includes sensor data that fills the selected sample window size. One Nvidia Geforce RTX 2080 Ti GPU was used for training using mini-batch sizes up to 10922 samples, and was limited by the amount of GPU memory, therefore mini-batch size was calculated dividing 32 768 by the selected sample window size. One computing core of the Intel i7-8700K central processor was used for processing duration testing using single sample mini-baches as in real-time processing. One core of the central processor was used as it reassembles well the processing on RTTM if no multi-threading is supported. The processor of RTTM provides lower core clocks, different CPU architecture and instruction sets, therefore the results are comparable between the test only. Simulink implementation on the same CPU would further reduce the computing duration, while implementation on RTTM would increase the processing duration. The average processing duration is calculated by dividing the test set processing time by the number of samples in it.

Overfitting is possible when training ANNs. Overfitting the training data usually results in the RMSE increase for the validation data, whereas the RMSE of the training data decreases. Several measures were adopted to prevent overfitting. First, the checkpoint with the best RMSE for the validation data is recorded for each Bayesian iteration. Second, dropout layers were introduced before the output FC layer in each ANN, and earlier in the MLP-LSTM structure. Thus, the risk of overfitting is reduced to the maximum possible extent for the current dataset.

The MATLAB code for VS training and testing is available at: https://github.com/eldux/UMvelocityVSresearch.

III. RESULTS

A. ANN TYPE AND SAMPLE WINDOW SIZE TESTING RESULTS

The performance metrics achieved by the simplest MLP with one hidden FC layer after hyperparameter search and learning with window sizes ranging from 3 to 51 samples are shown in Table 2. For these networks The RMSE ranged from 0.0235 to 0.0261 m/s, and the processing time ranged from 0.339 to 0.411 ms/sample. The processing duration is not strongly correlated with the number of neuron units in FC or

the window size. This is due to hardware limitations caused by the latency in data transfer between the memory and the computational units. The best performance for an MLP with one hidden layer was achieved using window size of 51 samples - RMSE of 0.0235 m/s and processing duration of 0.391 ms/sample. This model contained 752 hidden neurons. Next, 2 hidden layer MLP was tested.

TABLE 2. Metrics of 1 hidden layer MLP with various sample window sizes.

Window size	FC units	RMSE, m/s	Duration, ms/sample
3	752	0.0261	0.354
5	534	0.0256	0.359
7	863	0.0253	0.375
9	982	0.0250	0.386
11	945	0.0248	0.392
13	920	0.0246	0.361
15	682	0.0243	0.356
17	568	0.0241	0.339
19	547	0.0240	0.397
21	1024	0.0238	0.378
23	977	0.0236	0.362
27	1015	0.0236	0.411
35	768	0.0235	0.392
51	752	0.0235	0.391

The performance metrics of the MLP with two hidden layers are listed in Table 3. The RMSE ranges from 0.0218 to 0.0256 m/s and processing duration from 0.404 to 0.507 ms/sample. The highest accuracy of 0.0218 m/s RMSE was achieved using a window size of 51 input samples with a processing time of 0.404 ms/sample. The model contained 1118 hidden neurons. Next, 3 hidden FC layer MLP was tested.

TABLE 3. Metrics of 2 hidden layers MLP with various sample window sizes.

Window size	FC1 units	FC2 units	RMSE m/s	Duration ms/sample
3	773	1024	0.0256	0.405
5	919	1024	0.0252	0.417
7	868	1023	0.0247	0.435
9	1011	804	0.0246	0.419
11	1017	927	0.0242	0.424
13	1022	918	0.0240	0.445
15	1022	895	0.0239	0.449
17	1019	1023	0.0236	0.460
19	840	963	0.0235	0.439
21	1020	1021	0.0234	0.465
23	993	1024	0.0233	0.496
27	1014	1008	0.0230	0.490
35	1023	1023	0.0229	0.507
51	334	784	0.0218	0.404

The performance of the MLP with three hidden FC layers is shown in Table 4. The RMSE ranges from 0.0218 to 0.0246 m/s, while the duration from 0.350 to 0.6 ms/sample. The best model achieved an RMSE of 0.0218 m/s and a processing duration of 0.350 ms/sample with a window size of 51 samples. This model has 1222 hidden neurons and is faster than MLP with 2 hidden layers while providing same RMSE. The number of neurons on the second layer (FC2) is 50, which is noticeably lower compared to the first (FC1) and third (FC3) layers. This hourglass-like shape leads to a reduction of connections between layers and faster performance. Next, a DNN with 2 CNN layers was tested.

It should be pointed out that for all tested structures RMSE difference with input window sizes from 3 to 51 samples was in a range 18% from minimum RMSE.

TABLE 4. Metrics of 3 hidden layers MLP with various sample window sizes.

Window size	FC1 units	FC2 units	FC3 units	RMSE m/s	Duration ms/sample
3	418	1018	873	0.0246	0.463
5	999	9	1014	0.0241	0.372
7	999	15	1021	0.0239	0.387
9	1020	10	1004	0.0237	0.372
11	697	1013	897	0.0235	0.502
13	767	1020	894	0.0233	0.505
15	793	1019	1007	0.0230	0.538
17	674	9	928	0.0230	0.366
19	898	876	1022	0.0229	0.553
21	890	1015	838	0.0228	0.561
23	1019	873	958	0.0229	0.600
27	998	10	802	0.0227	0.411
35	1012	13	971	0.0223	0.424
51	162	50	1010	0.0218	0.350

The performance of DNN with 2 CNN layers was tested and is shown in Table 5. The RMSE ranges from 0.0210 to 0.0234 m/s and the processing duration from 0.365 to 0.620 ms/sample. The best RMSE of 0.0210 m/s was achieved for window sizes of 21, 23, and 35 samples. Among them, one with input sample window size of 23 has shortest processing duration among them - 0.421 ms/sample. This model achieved a 3.67% reduction in RMSE but was 20.3% slower compared to MLP with 3 hidden FC layers. This model has 195 convolutional units and 94 FC neuron units. Processing time was within acceptable limits, while performance was improved. Next, the 3-layer CNN was tested.

TABLE 5. Metrics of 2-layer CNN with various sample window sizes.

Window size	Units, kernel size, (stride)		FC units	RMSE m/s	Duration ms/sample
	1st Conv.	2nd Conv.			
3	128, 1x3, (1,1)	105, 1x3, (1,1)	218	0.0234	0.368
5	128, 1x5, (1,2)	128, 1x5, (1,2)	223	0.0225	0.365
7	123, 1x3, (1,1)	122, 1x7, (3,1)	210	0.0221	0.399
9	126, 1x2, (1,1)	123, 1x9, (4,1)	157	0.0219	0.436
11	119, 1x3, (1,1)	128, 1x3, (1,1)	211	0.0217	0.420
13	92, 1x3, (1,1)	127, 1x11, (1,5)	234	0.0214	0.409
15	125, 1x3, (1,1)	118, 1x5, (1,2)	202	0.0213	0.448
17	125, 1x3, (1,1)	128, 1x3, (1,1)	240	0.0211	0.485
19	127, 1x2, (1,1)	114, 1x7, (1,3)	206	0.0211	0.453
21	99, 1x3, (1x1)	127, 1x11, (1,5)	188	0.0210	0.438
23	69, 1x3, (1,1)	126, 1x10, (1,5)	94	0.0210	0.421
27	127, 1x2, (1,1)	121, 1x3, (1,1)	239	0.0212	0.516
35	125, 1x3, (1,1)	125, 1x11, (1,6)	238	0.0210	0.620
51	124, 1x2, (1,1)	124, 1x3, (1,1)	155	0.0214	0.540

The performance of the DNN with 3 CNN layers is shown in Table 6. The RMSE ranges from 0.0215 to 0.0231 m/s, and the duration from 0.429 to 0.604 ms/sample. The best accuracy of 0.0215 m/s was achieved for 19, 21, 27, and 35 samples windows sizes with a best processing duration of 0.525 ms/sample for window size of 19 samples. That model has 324 convolutional neurons and 200 FC neuron units. The overall performance was worse compared to DNN with 2 CNN layers. Therefore, DNNs with more

than 3 CNN layers were not tested. Next, DNN with LSTM was tested.

TABLE 6. Metrics of 3-layer CNN with various sample window sizes.

Window size	Units, kernel size, (stride)			FC units	RMSE m/s	Duration ms/sample
	1st Conv.	2nd Conv.	3rd Conv.			
3	120, 1x3, (1,1)	128, 1x3, (1,1)	112, 1x3, (1,1)	218	0.0231	0.442
5	126, 1x2, (1,1)	113, 1x4, (1,2)	111, 1x5, (1,2)	248	0.0228	0.429
7	106, 1x3, (1,1)	126, 1x3, (1,1)	122, 1x7, (1,3)	254	0.0224	0.462
9	126, 1x3, (1,1)	73, 1x5, (1,2)	128, 1x9, (1,4)	244	0.0222	0.454
11	91, 1x3, (1,1)	123, 1x3, (1,1)	122, 1x11, (1,5)	244	0.0219	0.461
13	123, 1x3, (1,1)	84, 1x2, (1,1)	111, 1x3, (1,1)	255	0.0219	0.487
15	108, 1x2, (1,1)	97, 1x2, (1,1)	126, 1x3, (1,1)	197	0.0217	0.476
17	76, 1x3, (1,1)	128, 1x2, (1,1)	127, 1x3, (1,1)	90	0.0216	0.459
19	99, 1x3, (1,1)	101, 1x2, (1,1)	124, 1x3, (1,1)	200	0.0215	0.525
21	111, 1x3, (1,1)	94, 1x2, (1,1)	128, 1x3, (1,1)	237	0.0215	0.570
23	111, 1x3, (1,1)	94, 1x2, (1,1)	99, 1x2, (1,1)	216	0.0217	0.442
27	91, 1x3, (1,1)	83, 1x2, (1,1)	115, 1x3, (1,1)	237	0.0215	0.571
35	77, 1x2, (1,1)	110, 1x3, (1,1)	119, 1x3, (1,1)	114	0.0215	0.574
51	119, 1x2, (1,1)	105, 1x2, (1,1)	118, 1x3, (1,1)	176	0.0215	0.604

The performance of the DNN with LSTM is shown in Table 7. The RMSE ranges from 0.0226 to 0.0244 m/s and the processing duration from 0.512 to 1.079 ms/sample. The best accuracy of 0.0226 was achieved for 35 samples window with a processing duration of 1.079 ms/sample. The RMSE is 0.0005 m/s higher compared to the best CNN based DNN. This model has 189 LSTM and 222 FC neuron units. The overall performance is worse than the best DNNs with 2 or 3 convolutional layers, and 2 or 3 hidden FC layers but better than ANN with one hidden FC layer. Next, BiLSTM was tested.

TABLE 7. Metrics of LSTM with various sample window sizes.

Window Size	LSTM units	FC units	RMSE m/s	Duration ms/sample
3	255	105	0.0244	0.512
5	255	91	0.0238	0.554
7	256	73	0.0234	0.605
9	183	86	0.0232	0.592
11	255	79	0.0231	0.704
13	256	122	0.0229	0.776
15	188	151	0.0229	0.733
17	255	146	0.0228	0.908
19	256	127	0.0228	0.909
21	209	165	0.0228	0.901
23	126	190	0.0227	0.650
27	145	117	0.0227	0.830
35	189	222	0.0226	1.079
51	183	230	0.0227	1.069

The performance of the BiLSTM is shown in Table 8. The RMSE ranged from 0.0225 to 0.0244 m/s and the duration from 0.564 to 1.019 ms/sample. The best accuracy of 0.0225 was achieved for sample window size of 35 with a processing duration of 0.986 ms/sample. This model has 98 BiLSTM and 238 FC neuron units. The RMSE is 0.0001 m/s lower than the best LSTM but lower by 0.0015 m/s than the best DNN with 2 convolutional layers. It estimates more accurately than an ANN with one hidden FC layer. It appears that bidirectional signal propagation provides a small RMSE advantage of 0.0001 m/s in the case of VS. BiLSTM was not investigated further.

TABLE 8. Metrics of BiLSTM with various sample window sizes.

Window size	BiLSTM units	FC units	RMSE, m/s	Duration (ms/sample)
3	184	157	0.0244	0.564
5	205	92	0.0238	0.642
7	150	67	0.0235	0.622
9	187	56	0.0232	0.743
11	139	152	0.0229	0.720
13	132	256	0.0229	0.735
15	132	256	0.0228	0.772
17	132	256	0.0227	0.797
19	132	256	0.0226	0.857
21	132	256	0.0226	0.914
23	83	202	0.0226	0.753
27	132	256	0.0226	1.019
35	98	238	0.0225	0.986
51	95	226	0.0227	0.961

The performance achieved by DNN based on MLP-LSTM combination is shown in Table 8. The RMSE ranges from 0.0229 to 0.0240 m/s and the duration from 0.478 to 1.151 ms/sample. The best accuracy of 0.0229 was obtained for 21 and 23 sample windows sizes with the shortest processing duration of 0.568 ms/sample for 21 sample window size. This model has 511 FC neuron units and 68 LSTM units. The processing time of MLP-LSTM seems to be slower than that of the DNN with 2 hidden FC layers and the same RMSE. Bayesian search resulted in lower LSTM unit counts for all investigated window sizes compared to LSTM and BiLSTM, and on average shorter processing times. The accuracy was worse than the best ANNs with LSTM/BiLSTM, CNN, and 2 or 3 hidden FC layers, but better than an ANN with one hidden FC layer.

TABLE 9. Metrics of MLP-LSTM with various sample window sizes.

Window size	FC1 units	FC2 units	LSTM units	RMSE m/s	Duration ms/sample
3	202	255	36	0.0240	0.478
5	255	255	34	0.0237	0.488
7	251	235	74	0.0234	0.524
9	237	253	89	0.0232	0.529
11	254	254	35	0.0231	0.496
13	255	229	44	0.0230	0.530
15	252	237	71	0.0230	0.567
17	249	255	50	0.0230	0.509
19	200	252	76	0.0230	0.548
21	256	255	68	0.0229	0.568
23	254	232	94	0.0229	0.598
27	206	252	79	0.0230	0.607
35	255	253	92	0.0230	0.650
51	256	224	199	0.0229	1.151

Figure 4 compares all ANNs for all window sizes in a graph. There the most promising window sizes are 19 to 35. MLP, LSTM, and BiLSTM show decreasing RMSE up to

35 samples window. The RMSE difference between LSTM and BiLSTM was small, and the dependence on window size was very similar.

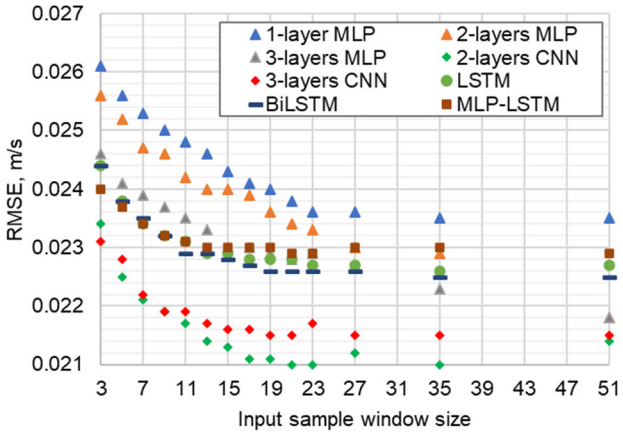


FIGURE 4. UM vertical velocity estimation RMSE using various sample window sizes with all tested ANNs.

Estimated FL UM vertical velocity and the ground truth signals are shown in Figure 5. Estimation was made using VS based on DNN with 2 convolutional layers. The estimate of the lower-frequency high-amplitude UM vertical velocity is shown in Figure 5 (a), and the estimate of the higher-frequency UM vertical velocity is shown in Figure 5 (b). It can be seen that the VS estimation tracks more closely the changes at lower frequencies and higher amplitudes, while rejecting high frequency oscillations.

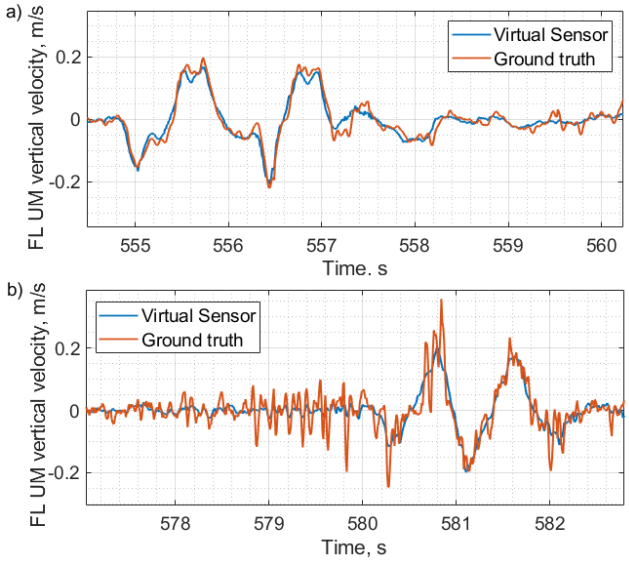


FIGURE 5. Virtual Sensor and Ground truth comparison: a) lower frequency signal example; b) higher frequency signal example.

Developed VS for UM is using data from SM, the high frequency oscillations are not reconstructed, as they are not transmitted to SM (due tire and suspension damping). Lower frequency signals are more important in suspension control, thereby the performance of developed VS is sufficient.

In summary, the best-performing VS was based on DNN with 2 convolutional layers using input window size of 23 samples, and the second-best was based on DNN with 3 convolutional layers using input window size of 19 samples.

B. RESULTS OF INPUT IMPORTANCE TESTING

The importance of the inputs was tested by setting each input one by one to 0 or 1 and recording the difference of the RMSE on the test set. 0 is used as it would mean no input, and 1 is used as it is the standard deviation after normalization. The results are shown in Table 10.

TABLE 10. Input signal importance by change of test RMSE.

Parameter	ΔRMSE (compared to 0.0210 m/s)	
	To 0	To 1
Driver torque requirement	+0.0002	+0.0003
ESP regulation	0.0000	0.0000
Master cylinder pressure	0.0000	+0.0001
Steering angle	+0.0005	+0.0010
Steering angle direction	+0.0007	+0.0012
Steering angle optimized	+0.0012	+0.0020
Vehicle velocity	+0.0007	+0.0009
FL wheel velocity	+0.0014	+0.0027
FR wheel velocity	+0.0002	+0.0004
RL wheel velocity	+0.0004	+0.0006
RR wheel velocity	+0.0003	+0.0004
Acceleration X axis	+0.0009	+0.0013
Acceleration Y axis	+0.0020	+0.0018
Acceleration Z axis	+0.0034	+0.0063
Roll rate	+0.0301	+0.0475
Yaw rate	+0.0036	+0.0046
Longitudinal velocity	+0.0004	+0.0009
Transversal velocity	+0.0002	+0.0007
Body sideslip angle	+0.0001	+0.0006

These results show that the IMU roll rate is the most important input signal for estimating the UM vertical velocity. Other important signals include the IMU yaw rate, accelerations on X, Y, Z axes, steering angle optimized, and front left wheel velocity. Slip angle, longitudinal and transversal velocity, FR, RL, RR wheels velocities, driver torque requirement, and master cylinder pressure signals are less important. DSC regulation signal could be omitted, as it is not having impact. However, there is no signal that, when set to zero or one, improves performance. Redundant velocity readings from the wheel velocity sensors and transversal velocity readings from the OFS sensor reduce the effects of signal and measurement noise, as removing these signals increases RMSE. Yet if it is not possible to use OFS, it can be discarded as the impact on performance would be manageable, and the model could rely more on the IMU and CAN data instead.

C. RESULTS OF NO INPUT MEAN COMPENSATION

To better understand how mean value compensation of the input signals affects the performance, additional tests were performed where only the standard deviation was normalized each signal. This is important because the mean and standard deviation were calculated using only the training data. The mean was subtracted from the validation and test set data, and the results were divided by the standard deviation to bring all signals to approximately the same level.

Table 11 shows the performance results obtained with various sample window sizes, which are compared with those

in Table 5. The lowest RMSE was obtained with window sizes of 21 and 27. The processing duration for window size of 27 samples was lower – 0.425 ms/sample. The RMSE range of 0.0210-0.0233 m/s was similar to the one achieved by DNN model with 2 convolutional layers trained on data with mean subtraction. The computation time was not affected by not using the mean compensation. These results show that there is no significant difference between using or not using mean compensation.

TABLE 11. Metrics of 2-layer CNN with various sample window sizes and no input mean subtraction.

Window Size	Units, kernel size, (stride)		FC units	RMSE m/s	Duration ms/sample
	1st Conv.	2nd Conv.			
3	126, 1x3, (1,1)	128, 1x3, (1,1)	244	0.0233	0.444
5	128, 1x5, (1,2)	213, 1x5, (1,2)	203	0.0225	0.428
7	117, 1x7, (1,3)	128, 1x7, (1,3)	174	0.0222	0.403
9	117, 1x7, (1,3)	128, 1x7, (1,3)	218	0.0218	0.468
11	123, 1x3, (1,1)	127, 1x11, (1,5)	249	0.0215	0.444
13	111, 1x3, (1,1)	128, 1x3, (1,1)	238	0.0215	0.414
15	127, 1x3, (1,1)	119, 1x10, (1,5)	164	0.0211	0.428
17	126, 1x3, (1,1)	110, 1x11, (1,5)	242	0.0211	0.431
19	123, 1x2, (1,1)	125, 1x3, (1,1)	207	0.0211	0.498
21	91, 1x3, (1,1)	127, 1x3, (1,1)	98	0.0210	0.471
23	81, 1x2, (1,1)	123, 1x3, (1,1)	255	0.0211	0.527
27	126, 1x11, (1,5)	126, 1x11, (1,5)	103	0.0210	0.425
35	128, 1x3, (1,1)	104, 1x3, (1,1)	112	0.0213	0.488
51	127 1x11, (1,5)	106, 1x11, (1,5)	248	0.0211	0.399

IV. DISCUSSION AND CONCLUSION

The real-world dataset developed for this study provides a valuable resource for future research and industrial implementation of ANN-based VSs. It addresses the previous limitation of the lack of such datasets for supervised DL. The study of the optimal sampling window length for input signals and ANN hyperparameters contributes to understanding how ANN models can be tuned to achieve maximum VS efficiency and accuracy in estimation of UM vertical velocity. The results of the study suggest that DNNs with convolutional layers have the greatest potential to achieve this goal and outperform other types of DNNs by providing the lowest RMSE of 0.0210 m/s and a sufficiently short processing time of 0.421 ms/sample with a window size of 23 samples. A closer examination of the ground truth and estimated signals showed that CNN-based VS rejects higher frequency oscillations together with process and measurement noise while preserving average values. This is because SM parameters are used as inputs, which are already damped by the suspensions system. In addition, it was found that VS relies primarily on the roll rate provided by the IMU. However, none of the signals can be removed to achieve better performance, and DSC regulation signal can be removed without loss of accuracy. Other tests showed that the input signal mean subtraction has little impact on performance.

In summary, our results represent a significant advance in data-driven VS for vehicle suspensions control, especially for UM vertical velocity estimation. As one of the most important inputs for suspension control, accurate estimation of vertical velocities of UM and SM can significantly improve vehicle comfort, handling, and stability without the need for costly physical sensors. The findings of this research can speed up research of VS for UM vertical velocity development.

Future research should further optimize the ANN with convolutional layers to achieve even better accuracy and implementation on in-vehicle RTTM, such as the dSPACE MicroAutoBox, while maintaining a processing time of less than 10 ms. Such integration would allow real-time testing of the created algorithms and analysis of the impact on comfort when using VS instead of physical sensors. For this purpose, also a quarter-car test setup or demonstrator vehicle can be used.

ACKNOWLEDGMENT

The experimental data was collected and supplied by Tenneco Automotive Europe.

REFERENCES

[1] *The Future of the EU Automotive Sector*, Policy Dept. Econ., Sci. Quality Life Policies, Eur. Parliament, Ecorys, Luxembourg, U.K., 2021.
[2] K. N. de Winkel, T. Irmak, R. Happee, and B. Shyrokau, "Standards for passenger comfort in automated vehicles: Acceleration and jerk," *Appl. Ergonom.*, vol. 106, Jan. 2023, Art. no. 103881, doi: 10.1016/j.apergo.2022.103881.
[3] J. Theunissen, A. Tota, P. Gruber, M. Dhaens, and A. Sorniotti, "Preview-based techniques for vehicle suspension control: A state-of-the-art review," *Annu. Rev. Control*, vol. 51, pp. 206–235, Jan. 2021, doi: 10.1016/j.arcontrol.2021.03.010.
[4] J. Förstberg, "Ride comfort and motion sickness in tilting trains," Ph.D. dissertation, Institutionen för Farkostteknik, Dept. Vehicle Eng., Roy. Inst. Technol., Stockholm, Sweden, 2000. [Online]. Available: https://kth.diva-portal.org/smash/get/diva2:8728/FULLTEXT01.pdf
[5] X. Wang, W. Zhuang, and G. Yin, "Learning-based vibration control of vehicle active suspension," in *Proc. IEEE 18th Int. Conf. Ind. Informat. (INDIN)*, vol. 1, Warwick, U.K., Jul. 2020, pp. 94–99, doi: 10.1109/INDIN45582.2020.9442091.
[6] L. Ming, L. Yibin, R. Xuewen, Z. Shuaishuai, and Y. Yanfang, "Semi-active suspension control based on deep reinforcement learning," *IEEE Access*, vol. 8, pp. 9978–9986, 2020, doi: 10.1109/ACCESS.2020.2964116.
[7] M. E. Shalabi, H. El-Hussieny, A. A. Abouelsoud, and A. M. R. F. Elbab, "Control of automotive air-spring suspension system using Z-number based fuzzy system," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dali, China, Dec. 2019, pp. 1306–1311, doi: 10.1109/ROBIO49542.2019.8961492.
[8] S. Qi, W. Li, A. Zhu, and X. F. Bai, "Responses of preview-based vehicle suspension system on discrete impact roads," in *Proc. 6th CAA Int. Conf. Veh. Control Intell. (CVCI)*, Nanjing, China, Oct. 2022, pp. 1–6, doi: 10.1109/CVCI56766.2022.9964996.
[9] J. Theunissen, "Explicit model predictive control for active suspension systems with preview," Ph.D. dissertation, Dept. Mech. Eng. Sci., Univ. Surrey, Surrey, U.K., Nov. 24, 2020, doi: 10.15126/thesis.00851885.
[10] Y. Zheng, B. Shyrokau, and T. Keviczky, "3DOP: Comfort-oriented motion planning for automated vehicles with active suspensions," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Aachen, Germany, Jun. 2022, pp. 390–395, doi: 10.1109/IV51971.2022.9827152.
[11] C. Poussot-Vassal, O. Sename, L. Dugard, R. Ramirez-Mendoza, and L. Flores, "Optimal skyhook control for semi-active suspensions," *IFAC Proc. Volumes*, vol. 39, no. 16, pp. 608–613, 2006, doi: 10.3182/20060912-3-DE-2911.00106.
[12] D. Hernandez-Alcantara, L. Amezquita-Brooks, N. Morales-Villarreal, and O. A. Juarez-Tamez, "Velocity estimation algorithms for suspensions," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Ottawa, ON, Canada, Nov. 2019, pp. 1–5, doi: 10.1109/GlobalSIP45357.2019.8969344.
[13] R.-K. Kim and K.-S. Hong, "Skyhook control using a full-vehicle model and four relative displacement sensors," in *Proc. Int. Conf. Control, Autom. Syst.*, Seoul, South Korea, Oct./Dec. 2007, pp. 268–272, doi: 10.1109/ICCAS.2007.4406920.

[14] K. Jeong and S. B. Choi, "Vehicle suspension relative velocity estimation using a single 6-D IMU sensor," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7309–7318, Aug. 2019, doi: 10.1109/TVT.2019.2920876.

[15] M. Viehweger, C. Vaseur, S. van Aalst, M. Acosta, E. Regolin, A. Alatorre, W. Desmet, F. Naets, V. Ivanov, A. Ferrara, and A. Victorino, "Vehicle state and tyre force estimation: Demonstrations and guidelines," *Vehicle Syst. Dyn.*, vol. 59, no. 5, pp. 675–702, May 2021, doi: 10.1080/00423114.2020.1714672.

[16] S. Kabadayi, A. Pridgen, and C. Julien, "Virtual sensors: Abstracting data from physical sensors," in *Proc. Int. Symp. World Wireless, Mobile Multimedia Networks(WoWMoM)*, New York, NY, USA, Jun. 2006, p. 592, doi: 10.1109/WOWMOM.2006.115.

[17] D. Martin, N. Kühl, and G. Satzger, "Virtual sensors," *Bus. Inf. Syst. Eng.*, vol. 63, no. 3, pp. 315–323, Jun. 2021, doi: 10.1007/s12599-021-00689-w.

[18] D. Wang and X. Li, "A novel virtual sensor modeling method based on deep learning and its application in heating, ventilation, and air-conditioning system," *Energies*, vol. 15, no. 15, p. 5743, Aug. 2022, doi: 10.3390/en15155743.

[19] S. Kerst, B. Shyrokau, and E. Holweg, "A model-based approach for the estimation of bearing forces and moments using outer ring deformation," *IEEE Trans. Ind. Electron.*, vol. 67, no. 1, pp. 461–470, Jan. 2020, doi: 10.1109/TIE.2019.2897510.

[20] A. Bertipaglia, B. Shyrokau, M. Alirezaei, and R. Happee, "A two-stage Bayesian optimisation for automatic tuning of an unscented Kalman filter for vehicle sideslip angle estimation," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2022, pp. 670–677, doi: 10.1109/IV51971.2022.9826998.

[21] M. Acosta, S. Kanarachos, and M. E. Fitzpatrick, "A virtual sensor for integral tire force estimation using tire model-less approaches and adaptive unscented Kalman filter," in *Proc. 14th Int. Conf. Informat. Control, Autom. Robot.*, Madrid, Spain, Jul. 2017, pp. 386–397, doi: 10.5220/0006394103860397.

[22] N. Pletschen and P. Badur, "Nonlinear state estimation in suspension control based on Takagi–Sugeno model," *IFAC Proc. Volumes*, vol. 47, no. 3, pp. 11231–11237, 2014, doi: 10.3182/20140824-6-ZA-1003.02500.

[23] P. Kojis, E. Šabanovič, and V. Skrickij, "Deep neural network based data-driven virtual sensor in vehicle semi-active suspension real-time control," *Transport*, vol. 37, no. 1, pp. 37–50, May 2022, doi: 10.3846/transport.2022.16919.

[24] G. Kim, S. Y. Lee, J.-S. Oh, and S. Lee, "Deep learning-based estimation of the unknown road profile and state variables for the vehicle suspension system," *IEEE Access*, vol. 9, pp. 13878–13890, 2021, doi: 10.1109/ACCESS.2021.3051619.

[25] E. Šabanovič, P. Kojis, Š. Šukevičius, B. Shyrokau, V. Ivanov, M. Dhaens, and V. Skrickij, "Feasibility of a neural network-based virtual sensor for vehicle unsprung mass relative velocity estimation," *Sensors*, vol. 21, no. 21, p. 7139, Oct. 2021, doi: 10.3390/s21217139.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[27] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012, doi: 10.5555/2188385.2188395.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[29] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Anchorage, AK, USA, May 2017, pp. 1578–1585, doi: 10.1109/IJCNN.2017.7966039.

[30] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[31] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis.*, Zurich, Switzerland. Cham, Switzerland: Springer, 2014, pp. 818–833, doi: 10.1007/978-3-319-10590-1_53.

[32] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated recognition, localization and detection using convolutional networks," in *Proc. ICLR*, 2013, pp. 1–16.

[33] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2017. [Online]. Available: http://www.deeplearningbook.org

[34] E. Šabanovič, "Sasūkos dirbtiniu neuronu tinklu igyvendinimas spartinančiuosiuose irenginiuose vaizdams analizuoti realiuoju laiku: Daktaro disertacija (=implementation of convolutional neural networks in accelerating units for real-time image analysis)," Ph.D. dissertation, Dept. Electron. Syst., Vilnius Tech., Technika, Vilnius, Lithuania, 2019, doi: 10.20334/2019-051-M.

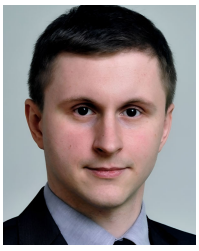
[35] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. Int. Conf. Mach. Learn.*, vol. 28, Atlanta, GA, USA, Jan. 2013, pp. 1–6. [Online]. Available: https://ai.stanford.edu/ amaas/papers/relu_hybrid_icml2013_final.pdf

ELDAR ŠABANOVIČ (Senior Member, IEEE) received the M.Sc. degree in electronics engineering and the Ph.D. degree in field of electrical and electronics engineering science from Vilnius Gediminas Technical University, Vilnius, Lithuania. He is currently a Senior Research Fellow with the Transport and Logistics Competence Centre, Transport Engineering Faculty, Vilnius Gediminas Technical University. His current research interests include deep neural networks, signal processing, image recognition, and autonomous vehicles.

PAULIUS KOJIS received the M.Sc. degree in transportation engineering from Vilnius Gediminas Technical University, Vilnius, Lithuania, where he is currently pursuing the Ph.D. degree in transportation engineering.

VALENTIN IVANOV (Senior Member, IEEE) received the Ph.D. and D.Sc. degrees in automotive engineering from Belarusian National Technical University, Minsk, in 1997 and 2006, respectively. He was an Assistant Professor, an Associate Professor, and a Full Professor with Belarusian National Technical University. In 2007, he was a Research Professor and became an Alexander von Humboldt Fellow with Technische Universität Ilmenau, Germany, where he was a Marie Curie Fellow, in 2008. He is currently the Coordinator of several European industrial–academic projects and Marie Sklodowska-Curie Actions. His research interests include vehicle dynamics, electric vehicles, and automotive control systems.

MIGUEL DHAENS received the M.Sc. degree in electro-mechanical engineering from KIH, Ostend, Belgium. He is currently an Engineering Manager with the Global Research Ride Performance Team, Tenneco, Limburg, Belgium, and responsible for defining the research road map and coordinating the global research activities of Tenneco's Ride Performance business.



VIKTOR SKRICKIJ received the Ph.D. degree in transport engineering from Vilnius Gediminas Technical University, Lithuania, in 2014. He is currently a Research Director with the Transport and Logistics Competence Centre, Vilnius Gediminas Technical University. His current research interests include vehicle dynamics, automated driving, and mobility.