vehicles

MDPI

Article

Synthesizing Vehicle Speed-Related Features with Neural Networks

Michal Krepelka * and Jiri Vrany

Faculty of Mechatronics, Informatics and Interdisciplinary Studies, Technical University of Liberec, Studentska 2, 46117 Liberec, Czech Republic

* Correspondence: michal.krepelka@tul.cz

Abstract: In today's automotive industry, digital technology trends such as Big Data, Digital Twin, and Hardware-in-the-loop simulations using synthetic data offer opportunities that have the potential to transform the entire industry towards being more software-oriented and thus more effective and environmentally friendly. In this paper, we propose generative models to synthesize car features related to vehicle speed: brake pressure, percentage of the pressed throttle pedal, engaged gear, and engine RPM. Synthetic data are essential to digitize Hardware-in-the-loop integration testing of the vehicle's dashboard, navigation, or infotainment and for Digital Twin simulations. We trained models based on Multilayer Perceptron and bidirectional Long-Short Term Memory neural network for each feature. These models were evaluated on a real-world dataset and demonstrated sufficient accuracy in predicting the desired features. Combining our current research with previous work on generating a speed profile for an arbitrary trip, where Open Street Map data and elevation data are available, allows us to digitally drive this trip. At the time of writing, we are unaware of any similar data-driven approach for generating desired speed-related features.

Keywords: Hardware-in-the-loop; generative models; synthetic data; MLP; LSTM

Citation: Krepelka, M.; Vrany, J. Synthesizing Vehicle Speed-Related Features with Neural Networks. *Vehicles* **2023**, *5*, 732–743. https://doi.org/10.3390/vehicles5030040

Academic Editor: Mohammed Chadli

Received: 10 June 2023
Revised: 15 June 2023
Accepted: 20 June 2023
Published: 26 June 2023

1. Introduction

One of the current trends in the automotive industry is the so-called Digital Twin (DT). The key principle is to create a sensor-induced digital copy of a physical object, which receives constant updates from its real-world counterpart. With DT, it is possible to simulate different scenarios and apply acquired findings to the physical object. [1]

DT use cases in the automotive industry vary from predictive maintenance of brake pads [2] to predicting electric vehicle energy consumption [3].

According to [4], another widely used approach in the automotive industry is Hardware-in-the-loop (HIL) simulation. The fundamental principle of this technique is to physically connect the tested device to a testbed that simulates the assembled product. An overview in [4] compares DT and HIL for electric vehicle (EV) propulsion drive systems. Combining both can yield the best result. The overview suggests initially testing different models with HIL simulation and using the acquired results to build the propulsion system. After performing enough measurements, the paper suggests creating a DT of EV propulsion drive systems.

Yufang et al. [5] showed an approach to predict long-term vehicle speed. Work published in [6] extended this idea, allowing us to generate a vehicle speed profile for an arbitrary trip where an Open Street Map (OSM) and elevation data are available.

This work aims to extend further the work of [6], who suggests the possibility of predicting a vehicle's Controller Area Network (CAN) bus data, such as brake pressure and the percentage of the pressed throttle pedal, along an a priori known trip with available OSM data, elevation data, and synthetic speed.

This paper focuses on predicting driver-induced features: brake pressure, percentage of the pressed throttle pedal, engaged engine gear, and corresponding engine RPM, to answer whether it is possible to predict these features using actual measured speed. The

findings could then be applied on an arbitrary unknown road where the OSM and elevation data are available, allowing us to generate the already-mentioned synthetic speed. For this purpose, we trained machine learning models using a data-driven approach. The prediction of more features for the given trip allows us to digitally drive our test car on an arbitrary trip specified by Global Positioning System (GPS) coordinates.

With a data-driven approach, we do not need an expert to describe complex vehicle dynamics. Instead, only data are necessary to develop models for an arbitrary (petrol-fueled) vehicle. Regular retraining of the models can reflect changes in the vehicles' dynamics if the power declines over time, which is a crucial part of DT.

At the time of writing, we are unaware of any similar data-driven approach for generating desired speed-related features for an arbitrary trip where the OSM and elevation data are available.

The proposed models aid our effort in digitizing the integration testing of the vehicle's dashboard, navigation, or infotainment. The findings will be used in our research partner company's HIL testbed, as digitally driving our test vehicle allows us to perform HIL simulations in a real-world scenario. However, there is a wide variety of other potential applications. For example, predicting vehicle acceleration and deceleration for an arbitrary trip, such as in the case of navigation to a specific place, could allow us to select the most fuel-efficient route. Furthermore, in the case of EVs, we might be able to predict its range more precisely, as according to [7], regenerative braking improves energy efficiency (and also emissions in case of internal combustion engine vehicles capable of regenerative braking).

The main contributions of our paper include:

- Proposing models for predicting speed-related features: brake pressure, percentage of the pressed throttle pedal, engaged engine gear, and corresponding engine RPM.
- Assessing whether vehicle speed contains enough context to utilize models without memory to predict speed-related features.

2. Related Work

The authors of [8] proposed a method to predict short-term, e.g., 10 or 20 s, vehicle velocity using a Long-Short Term Memory (LSTM) neural network with OSM and elevation data. The authors of [6] adopted a similar approach for long-term speed prediction, i.e., vehicle speed prediction can be made for the entire a priori known trip.

Liu et al. [9] predicted brake maneuvers from CAN bus data using the Gated Recurrent Unit (GRU), a version of a Recurrent Neural Network (RNN), and underlined the braking complexity in current cars with the brake assistance system.

Wang et al. [10] proposed an approach to build an LSTM model to identify driver braking intention based on the analysis of brake pedal operation.

In their research, Min et al. [11] presented an LSTM model to predict vehicle deceleration in braking situations where stopping before a traffic light is needed.

Zou et al. [12] suggested utilizing a Mixture of Hidden Markov Models to analyse the personalized driving behaviour of drivers in a car-following scenario. They then categorized test drivers into different groups based on their behaviour and trained a GRU and LSTM for each group to predict the drivers' acceleration in a car-following scenario.

Morton et al. [13] demonstrated the ability to generate predictions for vehicle acceleration distributions in a car-following scenario using an LSTM model.

According to [14], neural networks (specifically LSTM and backward-propagation neural networks) can predict the throttle of a wheel loader performing a loading maneuver.

3. Materials and Methods

3.1. MLP

Multilayer Perceptron (MLP) is a feedforward neural network consisting of multiple computational layers: an input, one or more hidden layers, and an output layer [15]. Even though MLP is limited in time series prediction as it has no memory, it has been used in COVID-19 prediction [16] and long-term speed prediction [5].

The basic building block is the linear layer that can be described by a simple equation:

$$y = xW^T + b \tag{1}$$

where x is the input vector, W is the weight matrix transforming the input shape into the output shape and b is the bias. To introduce non-linearity into the MLP, the Rectified Linear Unit (ReLU) is commonly chosen as an activation function [15]:

$$\phi(x) = max(0, x) \tag{2}$$

3.2. LSTM

The nature of our problem lies in time series forecasting, where it is often necessary to elevate the use of temporal information. An RNN uses a feedback loop to use the output from the previous prediction, hence building some context into the model. However, Vanilla RNNs suffer from both exploding and vanishing gradients when dealing with long-term dependencies. For this purpose, it is better to use a special type of RNN called an LSTM neural network [6,17].

Introducing LSTM cells into the hidden layer is a crucial improvement of the LSTM over Vanilla RNN. For every time step t, the LSTM cell processes input x_t. The information flow is controlled by forget f_t, cell g_t, and output o_t gates [18,19].

The following equations taken from [18] summarise the information flow inside the LSTM cell:

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \tag{3}$$

$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \tag{4}$$

$$g_t = tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \tag{5}$$

$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \tag{6}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{7}$$

$$h_t = o_t \odot tanh(c_t) \tag{8}$$

where σ is the sigmoid function and $\odot$ is an element-wise product. $W_{..}$ are matrices mapping the input x_t or hidden state from the previous time step $t-1$ to respective gates with bias $b_{...}$ The LSTM cell produces a cell state c_t and the hidden state h_t for each time step t. The hidden state h_t is the LSTM cell output. Both states c_t and h_t serve as the input into the next time step $t+1$.

Bidirectional LSTM (BiLSTM), an extension of LSTM, processes the data in standard and reversed order using two unidirectional LSTMs. The visualization in Figure 1 shows the forward LSTM pass producing hidden state $\overrightarrow{h_t}$ and the backward LSTM pass producing hidden state $\overleftarrow{h_t}$ for time step t. The BiLSTM output for every time step t is a product from both LSTM directions. In the case of Pytorch [18], LSTM outputs $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ are concatenated.

LSTM neural networks' success in time series forecasting includes fields such as travel time [19] and vehicle speed [6,17] prediction, making them a reasonable choice for solving our problem.

According to [20], BiLSTM allows for leveraging some additional features from the stock market data, thus improving the prediction precision compared to standard unidirectional LSTM.

For our problem, we decided to try BiLSTM as it could leverage additional features from the data, similar to the human driver that sees the road ahead and already knows the passed road.

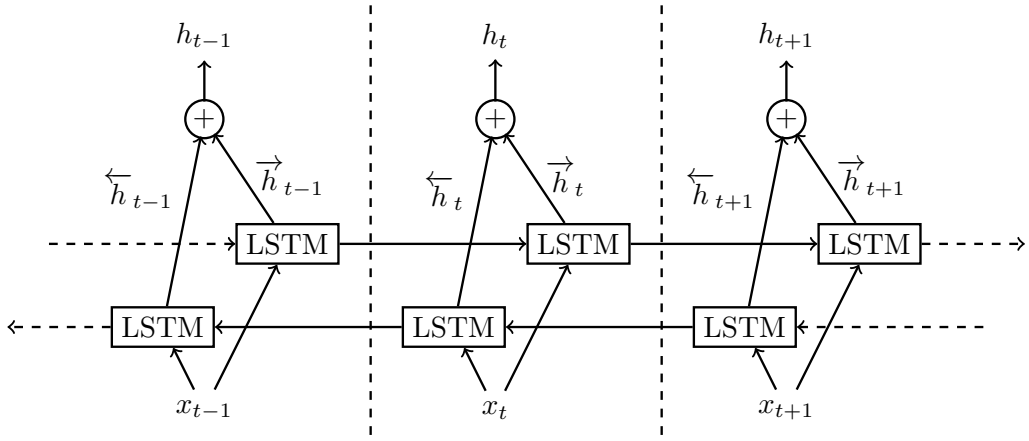

Figure 1. Visualization of the BiLSTM. For every time step t and input x_t, the output is a product of the hidden state $\overrightarrow{h_t}$ from the forward LSTM pass and hidden state $\overleftarrow{h_t}$ from the backward LSTM pass.

3.3. Comparison

To compare and describe the fidelity of the results, we selected two criteria—Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). The following equations define these criteria:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{9}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{10}$$

where n is the number of samples, y_i is the actual value, and $\hat{y}_i$ is the predicted value. We use MAE to determine the overall fit of our prediction and RMSE to detect large errors (such as outliers).

4. Experiments

Our objective was to create models for brake pressure (Brake), percentage of pressed throttle pedal (Throttle), engaged engine gear (Gear), and the corresponding engine RPM (RPM). For this purpose, we decided to try two machine learning architectures: MLP and BiLSTM.

One question to answer is whether the necessary context is already encoded in speed. In other words, would a simple MLP, a neural network without any memory, be precise, or is the context-aware BiLSTM needed as in the case for speed?

4.1. Dataset

In our experiment, we used a Skoda Karoq as a data collection vehicle that a single driver drove. The car has the following specifications:

- fuel: petrol
- transmission: six-gear manual
- power: 110 kW
- weight: 1632 kg
- wheel circumference: 2065 mm

Every ride was recorded and saved in 2 min intervals (or shorter at the end of the ride) called a mini ride. We filtered the mini rides with logged error values or those where the car speed variation was less than 5 $(\text{km/h})^2$, accounting for cases when the car was mainly standing or the speed was constant for the entire mini ride. The obtained dataset contains records from 13,337.85 km recorded during 229.24 h.

The data sampling frequency was 0.1 s, i.e., one mini ride consisted of at most 1200 samples. However, we needed a fixed ride length to train our models, so we split the mini rides into chunks with a length of 1024 samples. To train and test our models, we split the mini ride dataset into 80% train, 10% validation, and 10% test set.

We introduced a new feature representing wheel rotations per minute (wheelsRPM) into our dataset. The data collection vehicle's specification contains vehicle wheel circumference in mm, and feature *speed_esp* is logged in km/h during the ride. To compute wheelsRPM, we used the following formula:

$$wheels\_rpm = \frac{speed\_esp}{wheel\_circumference} \times \frac{1000000}{60} \tag{11}$$

The specific gear ratios of the test car's gearbox were not available. Therefore, we could not directly compute RPM and decided to learn the RPM from the data.

Another introduced feature was the first difference feature. An overview of the dataset features used for training our models is presented in Table 1.

Table 1. Overview of the dataset features.

Feature	Description
engine_rpm	engine RPM (rev/min)
throttle_pedal	pressed throttle pedal (%)
brake_pressure	raw brake pressure (bar)
engine_gear	engaged engine gear
speed_esp	vehicle speed estimated from ESP (km/h)
wheels_rpm	wheels RPM (rev/min)
prev_speed_esp_delta	first speed_esp difference with prev sample
next_speed_esp_delta	first speed_esp difference with next sample
prev_nav_altitude_delta	first altitude difference with prev sample
next_nav_altitude_delta	first altitude difference with next sample
prev_nav_heading_delta	first compass angle difference with prev sample
next_nav_heading_delta	first compass angle difference with next sample
prev_engine_gear_delta	first engine_gear difference with prev sample
next_engine_gear_delta	first engine_gear difference with next sample
prev_wheels_rpm_delta	first wheels_rpm difference with prev sample
next_wheels_rpm_delta	first wheels_rpm difference with next sample
prev_throttle_pedal_delta	first throttle_pedal difference with prev sample
next_throttle_pedal_delta	first throttle_pedal difference with next sample

4.2. MLP Models

We selected the MLP and implemented it in Python using the Pytorch library as a basic model. To introduce non-linearity to our MLP, we used ReLU. In Table 2, we summarise the MLP architecture and parameters used for training. The architecture in Table 2 consists of layers used in the MLP where the top row is the input layer and the bottom row is the output layer. The Linear(num_in, num_out) layer means that we used a linear, i.e., fully connected, layer with num_in input features and num_out output features.

In Table 2 MSELoss stands for the Mean Squared Error Loss and CrossEntropyLoss for the Cross-Entropy Loss. We used the former for regression and the latter for classification. We trained our models by using AdamW, the Pytorch implementation of the adaptive gradient method Adam with decoupled weights proposed in [21].

Table 2. Summary of the MLP model architecture and parameters. Each column represents one of our models.

	Brake	**Throttle**	**Gear**	**RPM**
# inputs	10	10	17	20
# outputs	1	1	7	1
Architecture	Linear(10, 256) ReLU(256, 256) Linear(256, 256) ReLU(256, 256) Linear(256, 256) ReLU(256, 256) Linear(256, 1)	Linear(10, 256) ReLU(256, 256) Linear(256, 256) ReLU(256, 256) Linear(256, 256) ReLU(256, 256) Linear(256, 1)	Linear(17, 256) ReLU(256, 256) Linear(256, 256) ReLU(256, 256) Linear(256, 256) ReLU(256, 256) Linear(256, 7)	Linear(20, 256) ReLU(256, 256) Linear(256, 256) ReLU(256, 256) Linear(256, 256) ReLU(256, 256) Linear(256, 1)
Loss function	MSELoss	MSELoss	CrossEntropyLoss	MSELoss
Batch size	800	800	800	800
Epochs	100	120	100	100
Optimizer	AdamW	AdamW	AdamW	AdamW
lr	1×10^{-4}	5×10^{-5}	1×10^{-4}	1×10^{-4}
weight decay	0.05	0.05	0.05	0.05

We use regression to predict a single output value for Brake, Throttle, and RPM. The first difference features (delta features) are the only means of providing context to the MLP. Thus the MLP context is short. As a result, predictions vibrate/oscillate, making them hard to use or useless for simulation purposes. Therefore, to prevent this, we applied a rolling mean with a window size of 5 for the Brake and Throttle during post-processing. The window for the RPM was only the size of 3. For all predictions, $max(0, x)$ was applied to prevent negative values, as negative feature values do not make sense in this scenario.

The nature of the Gear feature is ambiguous as it has the properties of both categorical and ordinal variables. On the one hand, gears $1, 2, ..., 6$ can be ordered. On the other hand, neutral gear is labeled 0 in our dataset but does not fit in this order. The reason for this is that engaging neutral is highly dependent on the driving style. Some drivers might only use neutral when the car is not moving, while others might use it for coasting (disengaging gear to reduce slowing down). Coasting might happen from any engaged gear. Hence, the Gear feature cannot be considered an ordinal variable. For this reason, we decided to treat it as categorical during training.

The Gear model predicts seven values: six gear shifts and a neutral gear, i.e., gear is not engaged. The best result was selected by the *argmax* function from the seven output features.

The Gear also suffers from a short context. As a result, the model often switches gears up and down on consecutive samples, which is undesirable. We used a simple filter to stabilize the gear switching: if the gear changes and returns within five consecutive samples, i.e., within 0.5 s, we used the original value. We first applied this filter from left to right and then in reverse.

Table 3 summarises the features used as the input for our models. The letter "T" marks the target feature for the model in the column. The letter "x" marks the input feature of the model. It is worth noting that using the deltas significantly improved training. Even though models for Brake and Gear use *throttle_pedal* deltas as the input features and Brake and RPM models use *engine_gear* deltas, we trained them on the dataset with actual measurements instead of the predicted one. The reason for this was to preserve original feature dependencies.

Table 3. Summary of the features used as the input for our models. Letter "T" denotes the target feature, and the letter "x" denotes the input feature of the model.

	Brake	**Throttle**	**Gear**	**RPM**
engine_rpm				T
throttle_pedal	x	T		
brake_pressure	T			
engine_gear	x		T	x
speed_esp	x	x	x	x
wheels_rpm			x	
prev_speed_esp_delta	x	x	x	x
next_speed_esp_delta	x	x	x	x
prev_nav_altitude_delta	x	x	x	x
next_nav_altitude_delta	x	x	x	x
prev_nav_heading_delta			x	
next_nav_heading_delta			x	
prev_engine_gear_delta	x			x
next_engine_gear_delta	x			x
prev_wheels_rpm_delta			x	
next_wheels_rpm_delta			x	
prev_throttle_pedal_delta	x		x	
next_throttle_pedal_delta	x		x	

4.3. LSTM Models

We know the entire ride itinerary and the vehicle's speed along this ride during the time of prediction (and training). Considering the nature of our problem, it is reasonable to elevate the power of the BiLSTM.

Table 4 summarises the parameters and architecture used for the BiLSTM models. The main difference from the MLP models is the introduction of BiLSTM as the first two layers of the model, allowing it to build the necessary context.

Table 4. Summary of the LSTM models' architecture and parameters. Each column represents one of our models.

	Brake	**Throttle**	**Gear**	**RPM**
# inputs	10	10	17	20
# outputs	1	1	7	1
Architecture	BiLSTM(10, 256) BiLSTM(256, 256) ReLU(256, 256) Linear(256, 256) ReLU(256, 256) Linear(256, 1)	BiLSTM(10, 256) BiLSTM(256, 256) ReLU(256, 256) Linear(256, 256) ReLU(256, 256) Linear(256, 1)	BiLSTM(17, 160) BiLSTM(160, 160) ReLU(160, 160) Linear(160, 7)	BiLSTM(20, 256) BiLSTM(256, 256) ReLU(256, 256) Linear(256, 256) ReLU(256, 256) Linear(256, 1)
Loss function	MSELoss	MSELoss	CrossEntropyLoss	MSELoss
Batch size	200	200	210	200
Epochs	300	150	200	300
Optimizer	AdamW	AdamW	AdamW	AdamW
lr	4×10^{-6}	1×10^{-5}	1×10^{-5}	4×10^{-6}
weight decay	0.05	0.05	0.05	0.05

To train our BiLSTM models we used identical features to the MLP models mentioned in Table 3. The deltas significantly improved training as well as in the case of the MLP. The models were trained on the dataset with actual measurements to preserve original feature dependencies instead of using predicted values as the input features.

In contrast to the MLP models, the BiLSTM models only needed one type of post-processing: applying $max(0, x)$ to prevent negative values in Brake, Throttle, and RPM.

As in the case of the MLP, we treated the Gear as a categorical variable, and its best result was selected by the *argmax* function from the seven output features. Contrasting with the MLP model, gear stabilization was not necessary.

5. Results

To objectively compare the trained models with actual measurement data, we used RMSE and MAE. Even though these metrics provide the necessary information, we also decided to demonstrate the model capabilities in graphs. We selected one ride with 2385 samples recorded in a city and rural roads. This ride was not part of the training or validation dataset. The data collection vehicle was a Skoda Karoq.

To better visualize the prediction comparison, only the first 100 s (1000 samples) of the selected ride are in each graph. Figure 2 shows the captured speed profile during the ride.

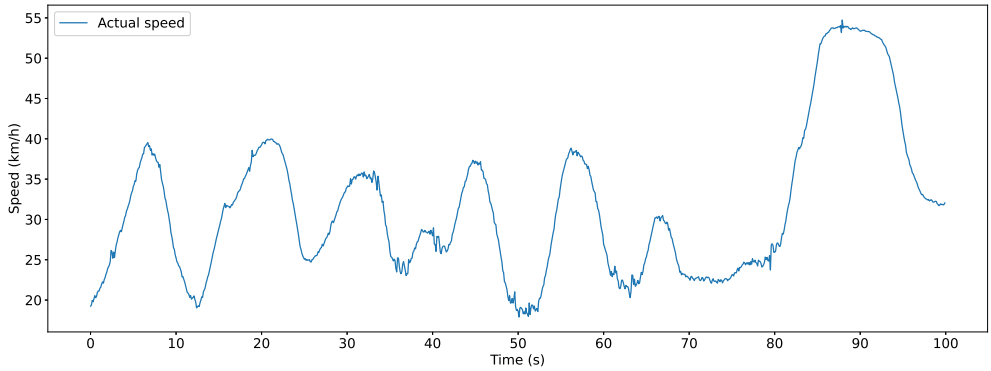

Figure 2. Speed profile for the selected ride.

Figure 3 shows the Brake time series captured during the ride. Both models effectively predict the trend. However, the MLP model's output is subjectively worse as it vibrates. Applying a longer rolling mean window can prevent vibration. However, this would increase the RMSE and MAE.

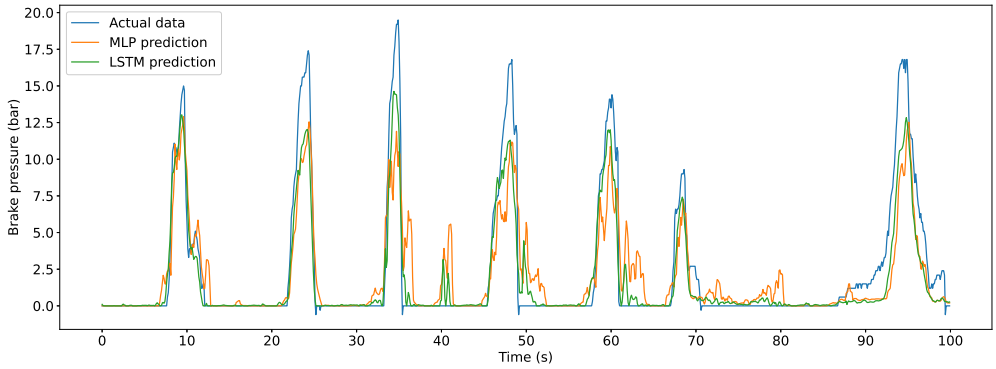

Figure 3. Brake pressure for the selected ride.

Figure 4 depicts the logged Throttle values. As in the case of the Brake, the actual Throttle values and LSTM prediction look similar. MLP prediction is subjectively worse than LSTM prediction as the output vibrates and lags behind the actual values.

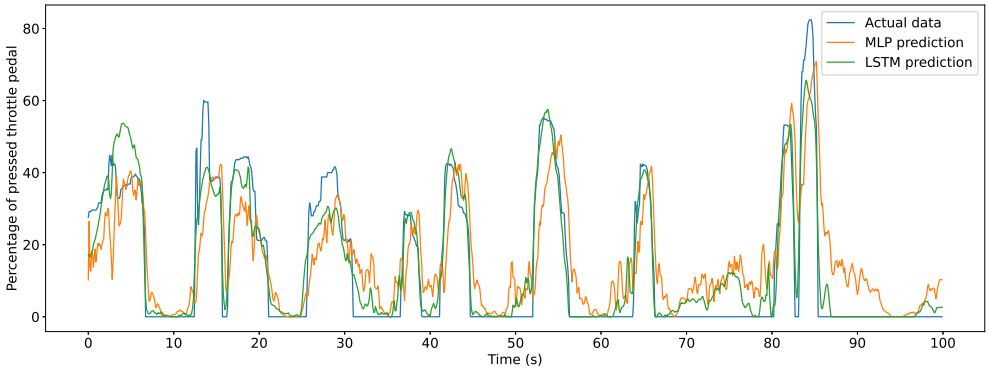

Figure 4. Percentage of pressed throttle pedal for the selected ride.

As can be seen in Figure 5, the MLP model effectively predicts the RPM's trend, apart from the obvious misprediction for samples around the 50th second. The LSTM model's prediction shows no issues.

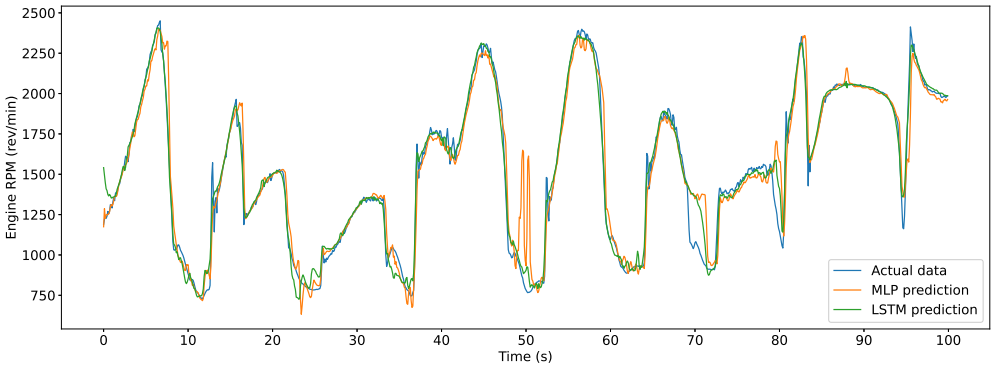

Figure 5. Engine RPM for the selected ride.

Figure 6 shows the Gear for the selected ride. The MLP and LSTM models learned to disengage the gear to neutral (marked as 0 in the figure). None of the models performed it precisely. Since disengaging the gear depends on the driver's behavior, both models performed reasonably well.

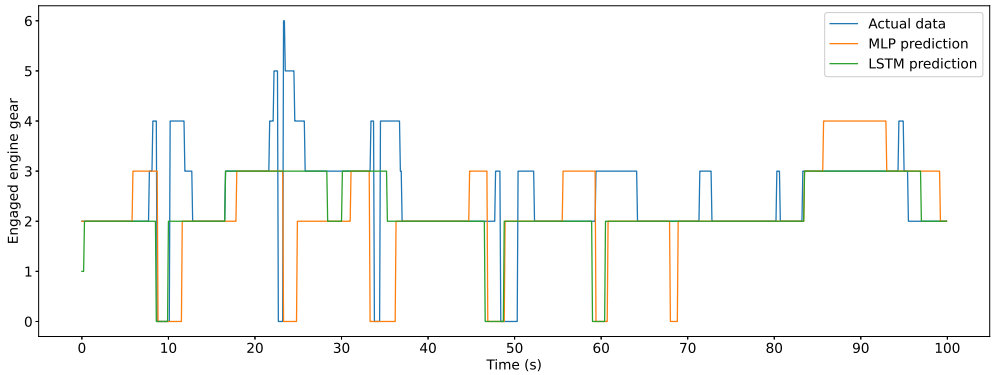

Figure 6. Engaged engine gear for the selected ride. Engaged gears have values $1, 2, ..., 6$ and the disengaged gear (neutral gear) is marked as 0.

The course of the selected ride between the 20th and 30th second interval demonstrates the complexity and certain unpredictability of modeling the interaction between the driver and the car. In Figure 6 for the gear and Figure 2 for speed, we can see that the actual driver stops accelerating, engages neutral, and then shifts to sixth gear. From this point, the driver gradually downshifts while slowing down. In contrast, our LSTM model kept the gear

engaged without any change, and the MLP model kept the gear disengaged until the driver wanted to accelerate. From our experience, all three approaches are valid and only depend on the driving strategy.

If we treat Gear prediction as a classification, the MLP model achieved an accuracy of 60.29%, and the LSTM model achieved an accuracy of 84.32% on the selected ride. Considering the driving strategy problem, we decided to treat Gear as a continuous variable and compared the predictions with actual values using MAE and RMSE to understand and evaluate its behaviour better.

Table 5 presents the RMSE and MAE for the selected ride to provide an objective measure. On this particular ride, the LSTM models objectively achieved a better performance.

Table 5. MAE and RMSE for the selected ride for the MLP and LSTM models.

	Brake		**Throttle**		**Gear**		**RPM**	
	MLP	LSTM	MLP	LSTM	MLP	LSTM	MLP	LSTM
RMSE	1.869	1.380	14.824	8.618	1.020	0.680	90.520	54.138
MAE	0.845	0.567	11.138	5.731	0.540	0.244	41.074	27.787

We also present the results for the entire test dataset in Table 6. We can partially explain the difference between the entire test set and the selected ride in brake pressure by different predictions when the car stops. The brake pressure depends on the driver's behaviour ("resting foot on the brake"). Similarly, a partial explanation for the RPM differences is in the idling RPM.

For completeness, if we treat Gear as a classification, on the entire test dataset, the MLP model achieved 72.01% accuracy, and the LSTM model achieved an accuracy of 79.49%.

Table 6. MAE and RMSE for rides in the entire test dataset for both the MLP and LSTM models.

	Brake		**Throttle**		**Gear**		**RPM**	
	MLP	LSTM	MLP	LSTM	MLP	LSTM	MLP	LSTM
RMSE	3.369	3.090	11.582	6.412	0.821	0.769	135.171	97.928
MAE	1.173	0.965	7.789	3.923	0.379	0.299	55.607	40.086

6. Conclusions

In this paper, we proposed models to predict the brake pressure, percentage of the pressed throttle pedal, engaged engine gear, and engine RPM. As we are extending previously published work, this prediction is possible on an arbitrary ride where the OSM and elevation data are available, and hence a speed profile can be created.

To test if there is enough context encoded in the speed profile, we trained models based on two architectures: MLP and BiLSTM. The used metrics show that all models based on BiLSTM perform better than our basic MLP models. However, both the MLP and BiLSTM models showed promising results as their prediction followed the trend of the actual measurements. Considering this, the BiLSTM models are better in scenarios where more precise results are needed. The MLP models might better fit cases where we expect frequent model retraining as they are faster to train. All of the models are suitable for our HIL testing scenario.

One significant advantage of our approach is that we do not need an expert to describe the vehicles' dynamics but only data to develop models suitable for our HIL simulation of an arbitrary (petrol-fueled) vehicle. Another advantage is that if the vehicle power declines over time, the MLP or BiLSTM can be regularly retrained and reflect the changes without an expert's complex modeling of the events.

In the future, we aim to use our models for digital driving simulations with DT, such as the already-mentioned brake pad predictive maintenance. If we can acquire the necessary

data, we also aim to test the presented approach on other vehicles, such as diesel, plug-in hybrid EVs, and EVs.

Author Contributions: Conceptualization, M.K. and J.V.; methodology, M.K. and J.V.; software, M.K.; validation, M.K.; formal analysis, M.K.; investigation, M.K. and J.V.; resources, M.K. and J.V.; data curation, M.K. and J.V.; writing—original draft preparation, M.K.; writing—review and editing, M.K. and J.V.; visualization, M.K.; supervision, J.V.; project administration, J.V.; funding acquisition, J.V. All authors have read and agreed to the published version of the manuscript.

Funding: This paper was supported by the Technology Agency of the Czech Republic project CK01000020, "Development of a GNSS route generator and CANBUS signal with machine learning using Software Defined Radio", and project CK02000136, "Virtual Convoy—a comprehensive environment for testing CAR2X communication systems".

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Restrictions apply to the availability of experimental data. Data were obtained from our research partner company and are available only with permission of Entry Engineering s.r.o.

Acknowledgments: We acknowledge the support of the Faculty of Mechatronics, Informatics, and Interdisciplinary Studies of the Technical University of Liberec.

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

References

1. Piromalis, D.; Kantaros, A. Digital Twins in the Automotive Industry: The Road toward Physical-Digital Convergence. *Appl. Syst. Innov.* **2022**, *5*, 65. [CrossRef]
2. Rajesh, P.K.; Manikandan, N.; Ramshankar, C.S.; Vishwanathan, T.; Sathishkumar, C. Digital Twin of an Automotive Brake Pad for Predictive Maintenance. *Procedia Comput. Sci.* **2019**, *165*, 18–24. [CrossRef]
3. Zhang, Z.; Zou, Y.; Zhou, T.; Zhang, X.; Xu, Z. Energy Consumption Prediction of Electric Vehicles Based on Digital Twin Technology. *World Electr. Veh. J.* **2021**, *12*, 160. [CrossRef]
4. Ibrahim, M.; Rassõlkin, A.; Vaimann, T.; Kallaste, A. Overview on Digital Twin for Autonomous Electrical Vehicles Propulsion Drive System. *Sustainability* **2022**, *14*, 601. [CrossRef]
5. Yufang, L.; Mingnuo, C.; Wanzhong, Z. Investigating long–term vehicle speed prediction based on BP–LSTM algorithms. *IET Intell. Transp. Syst.* **2019**, *13*, 1281–1290. [CrossRef]
6. Vrany, J.; Krepelka, M.; Chumlen, M. Generating Synthetic Vehicle Speed Records Using LSTM. In *Artificial Intelligence Applications and Innovations. AIAI 2023. IFIP Advances in Information and Communication Technology*; Maglogiannis, I., Iliadis, L., MacIntyre, J., Dominguez, M., Eds.; Springer: Cham, Switzerland, 2023; Volume 675. [CrossRef]
7. Clarke, P.; Muneer, T.; Cullinane, K. Cutting vehicle emissions with regenerative braking. *Transp. Res. Transp. Environ.* **2010**, *15*, 160–167. [CrossRef]
8. Deufel, F.; Jhaveri, P.; Harter, M.; Gießler, M.; Gauterin, F. Velocity Prediction Based on Map Data for Optimal Control of Electrified Vehicles Using Recurrent Neural Networks (LSTM). *Vehicles* **2022**, *4*, 808–824. [CrossRef]
9. Liu, S.; Koch, K.; Gahr, B.; Wortmann, F. Brake Maneuver Prediction—An Inference Leveraging RNN Focus on Sensor Confidence. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019; pp. 3249–3255. [CrossRef]
10. Wang, S.; Zhao, X.; Yu, Q.; Yuan, T. Identification of Driver Braking Intention Based on Long Short-Term Memory (LSTM) Network. *IEEE Access* **2020**, *8*, 180422–180432. [CrossRef]
11. Min, K.; Yeon, K.; Jo, Y.; Sim, G.; Sunwoo, M.; Han, M. Vehicle Deceleration Prediction Based on Deep Neural Network at Braking Conditions. *Int. J. Automot. Technol.* **2020**, *21*, 91–102. [CrossRef]
12. Zou, Y.; Ding, L.; Zhang, H.; Zhu, T.; Wu, L. Vehicle Acceleration Prediction Based on Machine Learning Models and Driving Behavior Analysis. *Appl. Sci.* **2022**, *12*, 5259. [CrossRef]
13. Morton, J.; Wheeler, T.A.; Kochenderfer, M.J. Analysis of Recurrent Neural Networks for Probabilistic Modeling of Driver Behavior. *IEEE Trans. Intell. Transp. Syst.* 2017, 18, 1289–1298. [CrossRef]
14. Huang, J.; Cheng, X.; Shen, Y.; Kong, D.; Wang, J. Deep Learning-Based Prediction of Throttle Value and State for Wheel Loaders. *Energies* **2021**, *14*, 7202. [CrossRef]
15. Aggarwal, C.C. *Neural Networks and Deep Learning*; Springer: Cham, Switzerland, 2018. [CrossRef]
16. Borghi, P.H.; Zakordonets, O.; Teixeira, J.P. A COVID-19 time series forecasting model based on MLP ANN. *Procedia Comput. Sci.* **2021**, *181*, 940–947. [CrossRef] [PubMed]

17. Du, Y.; Cui, N.; Li, H.; Nie, H.; Shi, Y.; Wang, M.; Li, T. The Vehicle's Velocity Prediction Methods Based on RNN and LSTM Neural Network. In *2020 Chinese Control Furthermore, Decision Conference (CCDC)*; IEEE: Piscataway, NJ, USA, 2020; pp. 99–102. [CrossRef]
18. Pytorch LSTM Documentation. Available online: https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html (accessed on 12 January 2023).
19. Duan, Y.; Wang, F. Travel time prediction with LSTM neural network. In Proceedings of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 1–4 November 2016; pp. 1053–1058. [CrossRef]
20. Siami-Namini, S.; Tavakoli, N.; Namin, A.S. The Performance of LSTM and BiLSTM in Forecasting Time Series. In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 9–12 December 2019; pp. 3285–3292. [CrossRef]
21. Loshchilov, I.; Hutter, F. Decoupled Weight Decay Regularization. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.