SAE INTERNATIONAL

ARTICLE INFO

Article ID: 12-07-04-0025

doi:10.4271/12-07-04-0025

Deep Learning Algorithms for Longitudinal Driving Behavior Prediction: A Comparative Analysis of Convolutional Neural Network and Long–Short-Term Memory Models

Giovanni Lucente,[1,2] Mikkel Skov Maarssoe,[1] Iris Kahl,[1] and Julian Schindler[1]

[1]Institute of Transportation Systems, German Aerospace Center (DLR), Germany
[2]TU Berlin, Fakultät Verkehrs- und Maschinensysteme, Germany

Abstract

In the realm of transportation science, the advent of deep learning has propelled advancements in predicting longitudinal driving behavior. This study explores the application of deep neural network architectures, specifically long–short-term memory (LSTM) and convolutional neural networks (CNNs), recognized for their effectiveness in handling sequential data. Using a 3-s temporal window that includes past vehicle progress, speed, and acceleration, the proposed model, a hybrid LSTM–CNN architecture, predicts the vehicle's speed and progress for the next 6 s. The approach achieves state-of-the-art performance, particularly within a 4 s horizon, but remains competitive even for longer-term predictions. This is achieved despite the simplicity of its input space, which does not include information about vehicles other than the target vehicle. As a result, while its performance may decrease slightly for longer-term predictions due to the lack of environmental information, it still offers reliable predictions and can be applied effectively in scenarios with partial observability. The comparative analysis of multilayer perceptron (MLP), LSTM, and one-dimensional CNN architectures highlights the challenges faced by MLP in capturing the complex nonlinearity of driving behavior. LSTM and CNN demonstrate superior performance, with model complexity influencing outcomes. No statistically significant difference is observed in the performance between LSTM and CNN models.

History

Received: 06 Jan 2024
Revised: 25 Mar 2024
Accepted: 22 May 2024
e-Available: 13 Jun 2024

Keywords

Driver behavior prediction, Deep learning, LSTM, Convolutional neural network

Citation

Lucente, G., Maarssoe, M., Kahl, I., and Schindler, J., "Deep Learning Algorithms for Longitudinal Driving Behavior Prediction: A Comparative Analysis of Convolutional Neural Network and Long–Short-Term Memory Models," *SAE Int. J. of CAV* 7(4):389–404, 2024, doi:10.4271/12-07-04-0025.

ISSN: 2574-0741
e-ISSN: 2574-075X

1. Introduction

In the domain of intelligent transportation systems, prediction of drivers' behaviors has become significantly important. It's essential to portray the traffic scenario realistically, meaning understanding traffic participants as rational agents. These individuals function within a multi-agent environment, aiming to effectively achieve their goals through their driving style. It is an outdated viewpoint to merely consider them as entities following trajectories dictated by their dynamic model. A vehicle can exhibit more aggressive behavior by driving closely to other objects, abruptly accelerating and braking, displaying a high inclination for risk, all with the primary intent of achieving its own goals. Conversely, another vehicle may adopt a more cautious approach, utilizing smoother acceleration profiles, maintaining a significant time gap, and considering other vehicles' intentions alongside its own tasks. Advanced driver assistance systems, along with smart infrastructures, need to consider these aspects to provide a reliable prediction of the upcoming traffic situation and an optimal decision accordingly.

According to the theory of planned behavior [1], most human behaviors are goal-directed. Actions therefore are driven by intentions, so the problem of behavior prediction can be, within the framework of this theory, first faced as an intention understanding problem. The corresponding behavior can be then recorded and represented by a multidimensional time-series data, composed, for example, by acceleration, steering angle, brake pressure, and the like.

The aim of this article is to provide an algorithm to predict the future progress and speed of a vehicle, therefore the driven path profile and speed within a fixed horizon, considering a time window in the past. The algorithm is required to receive as input information about a single vehicle, which can be measured through sensors external to the vehicle (no onboard). The Waymo Open Dataset [2] is used to train, validate, and test the network.

A secondary task of this article is to analyze and compare the architectures that in literature are considered the most suitable for sequential patterns comprehension and prediction, in particular the long–short-term memory (LSTM) models and the one-dimensional (1D) convolutional neural networks (CNN).

The contributions of this study are therefore the following:

- Definition of an LSTM–CNN hybrid architecture model for longitudinal driving behavior prediction, which receives as input the position and speed history of a vehicle and outputs the future speed and progress profile in a time horizon of 6 s. The model achieves state-of-the-art performance, particularly within a 4-s horizon, but remains competitive even for longer-term predictions. This is achieved despite the fact that its input space does not include information about vehicles other than the target vehicle, therefore not including environmental features such as the history of the surrounding vehicles. Consequently, the model can be applied effectively in scenarios with partial observability, which are common in cases of absence of V2I or V2V communication. The effectiveness of prediction algorithms in scenarios with limited observability is a gap in the literature, as most state-of-the-art algorithms depend on full information about the traffic scenario.
- Statistical analysis and comparison of architectures that are considered to be particularly suitable in the research community for comprehension and prediction of sequential patterns, in this case time series. In particular, this article deals with the discussion about which architecture is more suitable for this task between LSTM and CNN, adding a rigorous statistical study in the related literature.

This article is structured as follows: in Section 2 discusses about related work on driving behavior analysis, classification, and prediction, Section 3 describes the methodology, in Section 4 results are shown, analyzed, and compared with the state-of-the-art, while Section 5 draws the conclusions.

2. Related Work

The problem of driver behavior understanding and prediction is faced differently by the researchers. In general, the focus can be set on the aggressiveness recognition and classification, or on the prediction of time-series data of specific measurable variables. The last topic aims to predict features that can describe the behavior of a driver, so that countermeasures can be taken. The most ambitious outcomes when researchers face such problems are trajectory predictors that provide complete information. The problem of predicting specific features that explain the driver's behavior takes the form of regression analysis. All the studies that provide algorithms to recognize driver aggressiveness instead use statistical classification. A brief literature review is presented, starting from driver behavior classification approaches and then focusing on behavior prediction algorithms.

In literature, various classification approaches based on different features can be found [3, 4, 5, 6, 7, 8, 9, 10, 11]. These approaches are defined by considering which characteristics directly result from driving behavior. Some studies seek to determine whether researchers can identify the personality traits, attitudes, and beliefs of individuals who engage in aggressive driving behavior by analyzing opinions and practices of drivers [3, 4]. Other studies use internal measurable variables, such as throttle opening, to classify driving style [5]. Algorithms using throttle opening as a feature can be advantageous for the vehicle's internal control systems, which have access to data generally measured exclusively by onboard vehicle sensors. In [6] the authors explore driving behavior within work zones by employing unsupervised machine learning and vehicle kinematic data, including factors such as standard deviation of speed, standard deviation of longitudinal acceleration, their coefficients of variation, and other measures of volatility. In [7] the authors present an algorithm to classify driving maneuvers with respect to

driving style, with features based on longitudinal acceleration and yaw rate. In [8] an application for pattern recognition of headway data is presented, the feature used is the time history of the gap between the leading and the host vehicle. In [9] a maneuver-based driving behavior classification algorithm based on the smartphone sensor data (signals captured by accelerometers, gyroscopes, and GPS) is proposed. In [10] authors employ messages from connected vehicles to categorize driving behavior and compute a driving style score, utilizing precisely defined temporal volatility measures. The driving score depends on the driving environment, distinguishing between various settings such as highways, commercial streets, and residential streets.

Another differentiation among classification approaches relies on the chosen algorithm. Generally, two main methodologies can be distinguished: supervised and unsupervised learning. Supervised learning is applicable when labeled data is available, signifying a priori knowledge of the class in which the driving record should be categorized. When labeled data is unavailable, unsupervised learning becomes the sole option. Classes are defined by recognizing patterns in the data, without the prior information on what the algorithm should output. Both supervised and unsupervised learning approaches present advantages and disadvantages. Supervised learning relies on the labeled information, ensuring data classification into the desired categories. However, labeling can be subjective and, particularly with large datasets, it can demand excessive effort [11]. Unsupervised learning operates solely on objectively observable patterns. Nevertheless, the outcomes might be less interpretable, or the classification could be based on undesired characteristics. For example, unsupervised learning might lead to classification based on the traffic situation rather than on driver aggressiveness. Hence, it can be stated that with unsupervised learning, the challenge shifts from labeling the data to defining the features. In unsupervised learning, defining features is a critical task because these features must encapsulate the variance in the dataset concerning the desired classification characteristics. The primary algorithms commonly referenced in literature for classifying driver behavior include k-means clustering (kMC) [5, 6, 10], support vector machine (SVM) [5, 11], artificial neural network (ANN) [7, 8], and random forest [9]. Some methodologies are derived from a combination of different algorithms, aiming to harness the strengths of each. For instance, in [5], the authors utilized a kMC-based SVM (kMC-SVM), employing kMC to minimize the number of support vectors and thereby reducing the recognition time. In [11], the authors utilize a semi-supervised SVM, leveraging a small number of labeled data points to improve the effectiveness of unsupervised learning on the extensive unlabeled dataset.

Drivers' behavior classification and aggressiveness recognition are one aspect of the general analysis of the attitude of drivers on the roads. A wide section of literature goes more in detail, with the attempt to predict time-series data that constitute explicitly the driving behavior [12, 13, 14, 15, 16, 17]. In [12], the authors propose a generative adversarial network (GAN) architecture for long-term trajectory prediction, particularly suitable for multimodal scenarios, and compare it with LSTM models, which are considered the state-of-the-art in literature for this task. The topic is the long-term trajectory prediction in multimodal scenarios. The trajectories analyzed are therefore from 12 till 20 min long so that there is the possibility of different routes within an urban network. The results show that, even if the LSTM models perform better in unimodal scenarios, in multimodal scenarios they show a greater weakness compared to the proposed GAN. In [13], the authors propose a hidden Markov model (HMM) to extract the driver's intention and attention-based LSTM networks to predict the multivariate temporal aggressive driving behavior. The proposed method combines driver's intention, variable contribution sorting, and time-series processing. The approach is based on the theory of planned behavior [1], which considers every behavior as goal-directed, and therefore driven by intentions. The temporal features that are used to learn and predict the aggressive driving behavior include speed, accelerator, steering angle, and other variables that relate to the other traffic participants such as relative velocity and relative distance. In [14], the authors present a network that combines a multi-target sigmoid regression (MSR) layer with a deep belief network (DBN) to predict the front wheel angle and speed of the ego vehicle. The authors use as features the ego, the surrounding vehicles, and the driver control states. In [15], the authors present a logistic regression model that estimate the probability that the driver will accelerate or decelerate, receiving as input a combination of the current vehicle velocity, the speed limit 8 s ahead, and the accelerator pedal deflection. In [16] the authors propose an algorithm to predict and anticipate the drivers' behavior, so that it can be integrated in the dynamic model of the vehicle, with the aim to improve the performance of the vehicle controller. The approach used is a Gaussian mixture model (GMM) to classify the driving data into different states with different weights, coupled with a HMM to represent the dynamic driving process and determine the transition probability within different states. In [17] the authors present the driver's risk field (DRF), a two-dimensional (2D) field that represents the driver's belief about the probability of an event occurring. The driver's perceived risk is then obtained by multiplying the DRF with the consequence of the event. The authors show that human-like driving behavior emerges when the perceived risk is maintained below a threshold level.

For the behavior prediction task, most of the algorithms that can be found in the literature are data-driven, parametrized as neural networks [12, 13, 14, 16, 18]. The most common architecture that is used for time-series prediction is composed of LSTM layers [12, 13, 19, 20], some works show limitations of LSTM models [12, 18] proposing alternatives such as GANs [12] or CNNs [18]. In [18], in particular, the authors propose a network composed of convolutional layers to predict pedestrian trajectories. The advantage of CNN with respect to LSTM is on the inference time, since all the computations in a CNN are feed-forward in nature. Moreover, this characteristic allows an easy parallelization of the convolutions.

The current state-of-the-art in the field of longitudinal driving behavior prediction, and vehicle trajectory prediction in general, involves the exploration of various learning-based approaches [21, 22, 23, 24, 25]. These approaches aim to minimize prediction errors, extend the prediction horizon, and consider interactions among traffic participants. The authors of [21, 22] investigate the use of LSTM models for trajectory prediction. Specifically, in [21], the authors emphasize model explainability by introducing an LSTM model with spatial–temporal attention mechanisms to enhance interpretability in vehicle trajectory prediction. Meanwhile, in [22], the authors propose a hierarchical LSTM-based method where the trajectory prediction task is broken down into three sequential steps: predicting driving intention, estimating lane change time, and forecasting the trajectory. In [23], the authors present a novel approach that leverages lane information to predict a stochastic future relationship among agents. The proposed model generates diverse yet socially plausible trajectory samples by probabilistically capturing interactions, offering explanatory elements such as waypoint occupancy or inter-agent proximity. In [24], the authors couple the prediction and planning by conditioning the prediction on multiple candidate trajectories of the ego vehicle. This approach proves highly beneficial for autonomous driving in interactive scenarios. In [25], the presented model takes a set of predicted trajectories for each agent as input and outputs a consistently reordered recombination. This recombination module realigns initially independent modalities to avoid collisions and ensure coherence. The prediction results in the studies [21, 22, 24] will be used as a benchmark for comparison.

This article presents models based on MLP, LSTM, and CNN architectures to predict the longitudinal behavior of vehicles in the next 6 s, having as input the previous observed 3 s. Different architectures are compared and the results are presented. The prediction for each vehicle is decoupled with respect to the other traffic participants. The disadvantage of a consequent less precise prediction, particularly if a maneuver starts after the 3 observed seconds, is balanced with the great benefit that the approach is suitable also in case of partial observability, since there is no need to input the state of all the traffic participants in the scenario.

3. Methodology

In this section the approach is presented, and the dataset and the architecture selection are explained in Sections 3.1 and 3.2. The discussion on the time window and prediction horizon is provided in Section 3.3. The choice of the network architecture requires some theoretical considerations, particularly to recall the various types of layers that can be employed, their differences, their utilities and drawbacks.

3.1. Dataset

The data used to tune the parameters of the model come from the Waymo Open Dataset [2], an open source dataset, collected in different cities in the US, composed of two typologies of data: the perception dataset with high-resolution sensor data and labels for 2030 segments, and the motion dataset with object trajectories and corresponding 3D maps for 103,354 segments. In this research, the dataset is composed of samples from the motion dataset.

$$\boldsymbol{X} = \begin{bmatrix} s_0 & v_0 & a_0 \\ s_1 & v_1 & a_1 \\ \dots & \dots & \dots \\ s_{H-1} & v_{H-1} & a_{H-1} \\ s_H & v_H & a_H \end{bmatrix} \quad \boldsymbol{Y} = \begin{bmatrix} s_0 & v_0 \\ s_1 & v_1 \\ \dots & \dots \\ s_{H-1} & v_{H-1} \\ s_H & v_H \\ s_{H+1} & v_{H+1} \\ \dots & \dots \\ s_{N-1} & v_{N-1} \\ s_N & v_N \end{bmatrix} \quad \text{Eq. (1)}$$

Equation 1 shows the input and the output tensors $\boldsymbol{X}$, $\boldsymbol{Y}$ that feed the model in the learning process. The input represents the longitudinal behavior of a vehicle within a time window of 3 s, the output describes the behavior of the same vehicle considering the 3 s observed and the future 6 s, for a total time window of 9 s. The features that describe this behavior are the progress [m], speed [m/s], and acceleration [m/s^2] for the input, while the output is composed of the progress and the speed. The features are sampled every 0.1 s for a horizon of $H = 30$ time steps for the input and $N = 90$ time steps for the output. The scenarios analyzed are 1000, in each scenario there could be different vehicles, therefore different tracks recorded. The samples collected are 15,369 tracks. The dataset is divided into train dataset (70%, 10,758), validation dataset (20%, 3075), and test dataset (10%, 1536). The validation dataset is used to select the best-performing model during training and to keep under control overfitting. The test dataset is then the final test for the model.

3.2. Architecture

In this section, the architecture choice process is presented. Different architectures are compared and the most performing one is chosen. Before presenting the architectures that are tested, a brief theoretical introduction is given, describing the typology of layers that have been used in the tested architectures. In particular, there will be a focus on the LSTM layer and on the one-dimensional convolutional layer, which in literature are considered the most suitable for time-series prediction.

LSTM Layer The LSTM model is a subtype of recurrent neural network (RNN), which was introduced in 1997 by [26] and is used for learning data in sequential patterns, as for instance time-series and natural language texts. LSTM has been introduced to face the problem of short-term memory of RNN models. In Figure 1, the structure of a LSTM cell is

presented. The configuration of the cell allows to retain selected information in long-term memory, avoiding it to vanish as the information proceeds through the neurons.

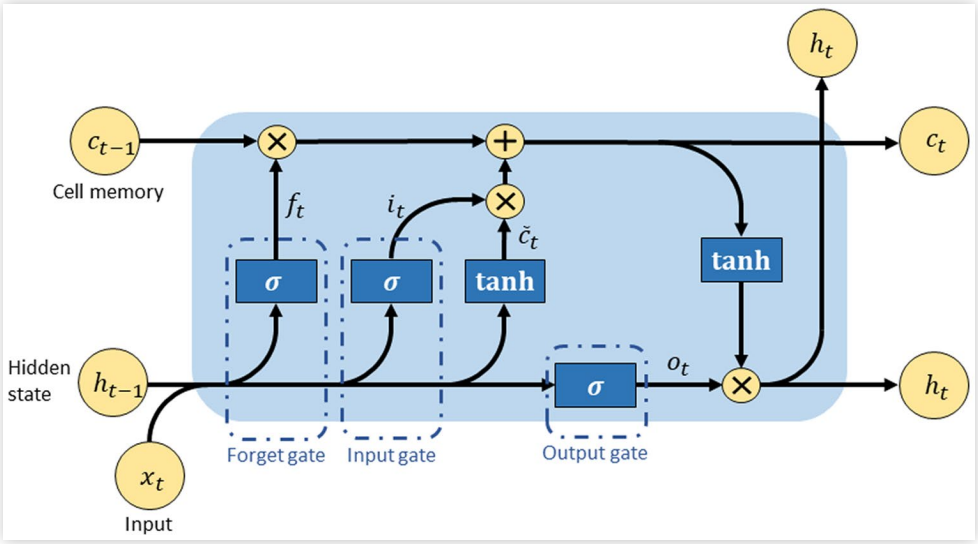
The long-term memory is stored in the cell state c_{t-1}, it passes through the cell and is separated from the hidden state h_{t-1}, which constitutes the results of the previous calculation step and therefore the short-term memory. Each cell receives and processes the corresponding input time step x_t, the previous state of long-term memory c_{t-1}, and the hidden state h_{t-1}. The information contained in these three values is processed in the following gates:

- The Forget Gate: The hidden state and the current input are passed into a sigmoid function. The output f_t, included between 0 and 1 is multiplied by the long-term memory, and it establishes which information from the previous time steps is forgotten and which is preserved.
- The Input/Update Gate: It weights the current input and the hidden state to establish which information is useful from the current input. The output, called input/update gate's activation vector i_t, is then multiplied by the cell input activation vector $\tilde{c}_t$ and then added to the cell state, that is updated as c_t, which preserves the useful past information included in c_{t-1} and includes the new information coming from x_t.
- The Output Gate: It decides which information should be included in the output of the LSTM cell h_t. A sigmoid function receives the hidden state and the current input, it computes o_t, called the output gate's activation vector, which is then multiplied by the cell state c_t, activated by the tanh(·) function. The result constitutes the output of the LSTM unit h_t, or hidden state, which can be given as input to the next LSTM cell.

These three gates regulate the flow of information through the cell and the network. As a result of this structure, useful long-term information is preserved, allowing for efficient prediction in the current and future time steps. The LSTM layer receives as input a tensor $\boldsymbol{X} = \{x_t\}$, a multidimensional array of dimension $n \times k$, where n is the number of steps of the sequential pattern, in the case under analysis the number of time steps of the time series; k is the number of channels (or features) that constitute the input, in this case acceleration, speed, and progress. The number of units u, parameter of the LSTM layer, gives the space dimension of the output $\boldsymbol{H} = \{h_t\}$, that is $n \times u$. Therefore, the output dimension conserves the time length of the input, while the channels' number is given by the number of units.

Convolutional Layer The CNN architecture was introduced to address the issue of the curse of dimensionality that classical multilayer perceptron models face when processing high-dimensional data, such as images. In full connected layers indeed, each neuron is connected to all the neurons in the previous layer, which leads to an explosion in the number of weights. Convolutional layers use local receptive fields, which means that each neuron is connected to a small region of the previous layer, defined by the kernel size. This has the double advantage to reduce the number of parameters and to increase the response to a spatially local input pattern. In convolutional networks indeed, since each neuron is connected to a subset of close neurons in the previous layer, spatially local correlation is apprehended because sparse local connectivity is enforced by the layer. In fully connected layers, inputs that are far apart are treated in the same way as inputs that are close to each other, losing the ability to recognize spatial structure in the data. Another important property of convolutional layer is a consequence of the capability to recognize spatially local correlation is the translation invariance. Patterns can be indeed recognized regardless of their location in the input space, making the CNN particularly well-suited for tasks such as image recognition.

FIGURE 1 The long–short-term memory (LSTM) cell.



In Figure 2 the structure of two typologies of convolutional layers are shown: the two-dimensional model, common for image recognition, and the one-dimensional model, suitable for time-series data type. The figure also represents the convolution process and the role of the kernel. A kernel, convolution matrix or mask, is a matrix of weights that slides over the 2D (or 1D) input data, performing an elementwise multiplication and then summing up the result that is given as activation input to a single output neuron. The most used activation function is the rectified linear activation function (ReLU). The number of filters determine the number of kernels that are used in the convolution, and therefore the dimension of the output space. The paddling is a procedure to add zeros on the borders of the input data structure to guarantee the same dimensionality in output (same image size or same time-series length). Often the convolutional layer is followed by a pooling operator, where max pooling is the most used. This operation partitions the input space into sub-regions and for each of them outputs the maximum value contained, reducing the dimensionality.

Architecture Selection The network architectures that are compared and tested are shown in Figure 3. The set can be divided into four groups: architectures composed of only dense layers, therefore multilayer perceptron models (MLP), architectures composed of 1D convolutional layers, architectures composed of LSTM layers, and one architecture model composed of LSTM and 1D convolutional layer. Notice that:

- The input data are composed of three channels (or features), each one with 30 time steps length, this means a shape of (30.3). The inputs are indeed the 3 s time histories of progress, speed, and acceleration. While 1D convolutional layer and LSTM layer adapt to a multi-channel input, the dense layer must receive exclusively a one-channel input, this explains the necessity to flat the input before the dense layer.
- The output data are composed of two channels (speed and progress) with 91 time steps length, with an output shape of (91.2). The final reshape layer is necessary to modify the one-channel output of a dense layer into the desired two-channel shape.

FIGURE 2 The two-dimensional (a) and the one-dimensional convolutional layer (b).

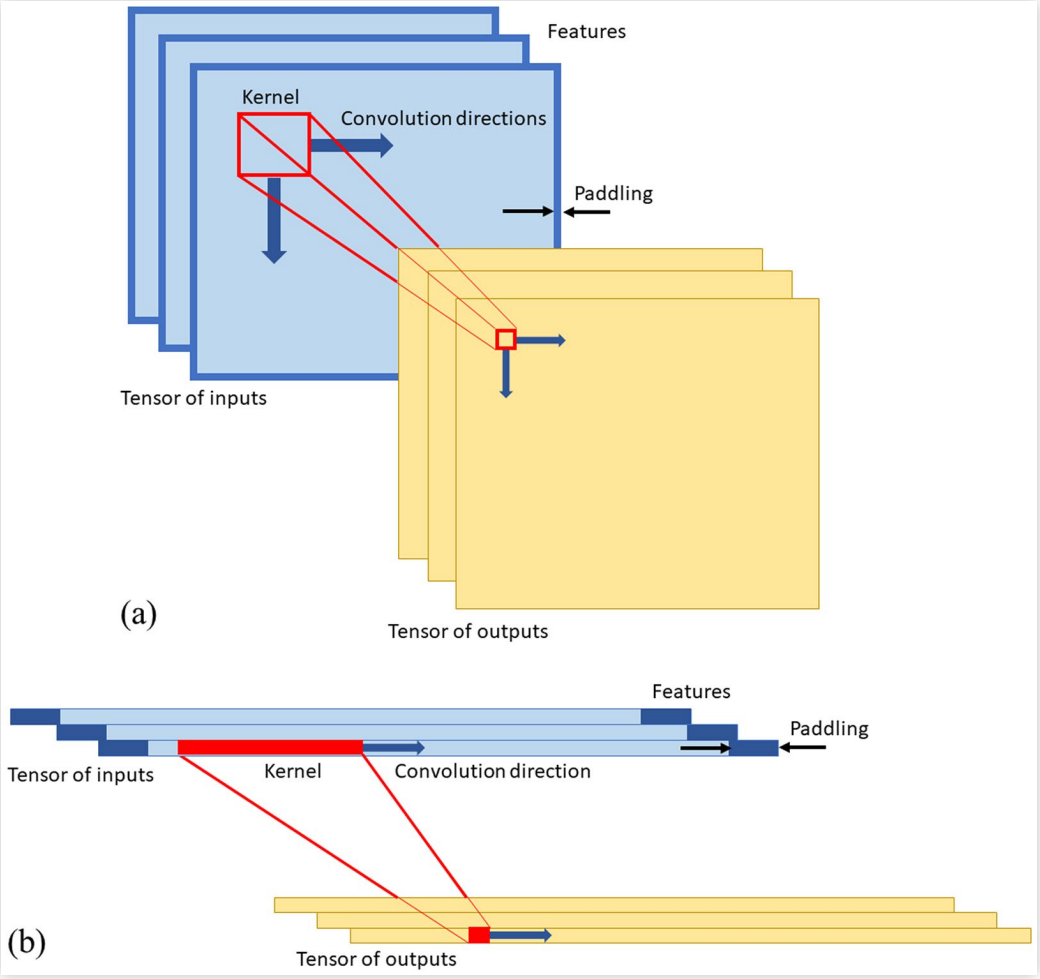
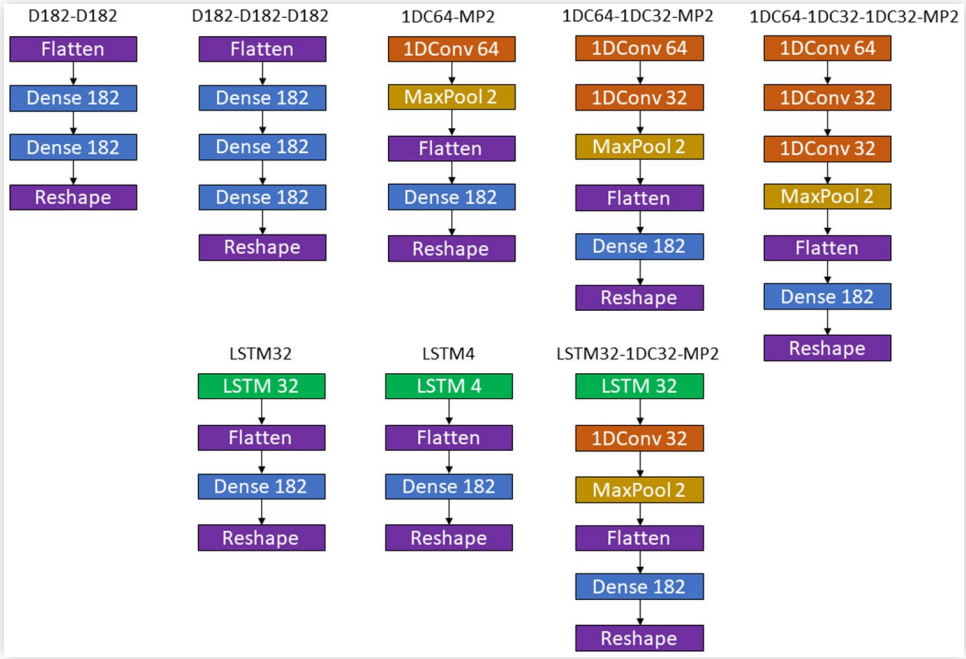


FIGURE 3 Tested architectures.



- For each family of architectures, different complexity levels are tested to estimate the optimal number of parameters and filters.

In Figure 3 the layers are represented with some abbreviations:

- Dense 182 (D182): A dense layer with 182 neurons, the activation function used is ReLU.
- 1DConv 64/32 (1DC64/32): A 1D convolutional layer with 64 or 32 filters, the activation function is ReLU, the kernel size is 3, and paddling is applied.
- MaxPool 2 (MP2): Max pooling layer with a pool size of 2.
- LSTM 4/32: LSTM layer with 4 or 32 units.

In Table 1 the layers used in the architectures are presented, showing particularly their specific parameters, the input and the output shapes. Table 2 shows the complexity of each model, expressed by the total number of parameters.

TABLE 1 Summary of the effects of the layers' parameters on the output shape.

Layer typology	Specific parameter		Input shape	Output shape
Dense	Number of neurons:	n	(k)	(n)
1D convolutional	Number of filters:	m	(t, k)	(t, m)
LSTM	Number of units:	u	(t, k)	(t, u)
MaxPool	Size of the pool:	p	(t, k)	$(t/p, k)$

3.3. Temporal Windows for Input and Output

In this section, two models, namely the 1DC64-1DC32-MP2 and the LSTM32-1DC32-MP2, are trained using different historical time windows and considering various prediction horizons. The goal is to identify the optimal combination to achieve a prediction that does not necessitate an excessively long historical time window while ensuring a reliable long-term horizon.

Given that the dataset consists of segments lasting 9 s each, it is clear that the sum of the prediction horizon and the input time window must be less than 9 s.

The measure on which the horizon selection is based consists in the mean squared error of the 1D final displacement error (FDE), computed on the longitudinal progress. Further details on this measure are provided in Section 4.

Observing Tables 3 and 4 some considerations can be done:

TABLE 2 Number of parameters for each model.

Model	Number of parameters
D182-D182	49,868
D182-D182-D182	83,174
1DC64-MP2	175,542
1DC64-1DC32-MP2	94,358
1DC64-1DC32-1DC32-MP2	97,462
LSTM32	179,510
LSTM4	22,150
LSTM32-1DC32-MP2	95,254

TABLE 3 RMSE (m) of the 1DC64-1DC32-MP2 model for different time windows and prediction horizons. In bold is the best time window for each prediction horizon.

		1DC64-1DC32-MP2 model RMSE [m]							
		Time window [s]							
		1	2	3	4	5	6	7	8
Prediction horizon [s]	1	0.25	**0.19**	0.21	0.23	0.25	0.27	0.25	0.35
	2	0.71	**0.70**	**0.70**	0.76	0.71	0.75	0.88	
	3	1.705	**1.51**	1.63	1.60	1.67	1.72		
	4	3.17	2.89	3.10	2.94	**2.81**			
	5	4.68	4.78	4.44	**4.05**				
	6	6.25	6.43	**6.06**					
	7	8.74	**8.63**						
	8	**10.82**							

TABLE 4 RMSE (m) of the 1DC64-1DC32-MP2 model for different time windows and prediction horizons. In bold is the best time window for each prediction horizon.

		LSTM32-1DC32-MP2 model RMSE [m]							
		Time window [s]							
		1	2	3	4	5	6	7	8
Prediction horizon [s]	1	0.23	0.24	**0.22**	0.24	0.25	0.27	0.31	0.41
	2	0.71	0.77	0.74	0.74	**0.70**	0.77	0.86	
	3	1.52	1.61	1.65	1.64	**1.55**	1.71		
	4	**2.68**	2.79	2.93	2.77	2.85			
	5	4.51	4.28	**4.18**	4.31				
	6	6.44	6.37	**6.29**					
	7	**7.97**	8.08						
	8	**10.03**							

- There is no significant difference in the results between the two models. This observation will also be statistically evident in Section 4.
- Predicting 6 s ahead provides a sufficient planning horizon and is reasonably reliable. Predictions for longer horizons are less accurate, primarily due to the increased importance of interactions between traffic participants in such contexts.
- There is not a significant difference between input time windows of 1, 2, or 3 s. This implies that most of the necessary information for prediction is already accessible within a 1-s time window. However, it is noteworthy that, especially for longer prediction horizons, the 3-s input time window outperforms the others, therefore, it is selected as the reference.

4. Results and Discussion

In this section, we present the performances of various models and conduct a comparative analysis to determine the optimal architecture for the prediction task (Section 4.1). The results are subsequently compared with the state-of-the-art in the field of vehicle behavior prediction and trajectory prediction, providing room for discussion (Section 4.2).

4.1. Model Comparative Analysis

The loss function used for training and validation is the mean squared error (MSE). The choice of the MSE over the mean absolute error (MAE) is due to the fact that MSE offers faster convergence than MAE, due to the bigger error in backpropagation, assuming that the dataset does not contain too many outliers, valid hypothesis for the Waymo dataset.

After the training, the best weights for each model are selected based on the minimum MSE on the validation dataset. The test dataset is used to select the best architecture. The measure on which the model selection is based consists of the MSE of the 1D FDE, computed on the longitudinal progress:

$$MSE_{FDE} = \frac{1}{M_{test}} \sum_{k=1}^{M_{test}} \left(s_N^k - \hat{s}_N^k \right)^2 \qquad \text{Eq. (2)}$$

where M_{test} is the number of samples in the test dataset, s_N^k is the real final progress (at the Nth timestep), and $\hat{s}_N^k$ is the predicted final progress, both of the kth sample. Notice that MSE_{FDE} is different from the MSE used during the training. The decision to use the error on the last time step and not the

MSE on the whole trajectory, like in training, has the aim to highlight the performance differences between the models. The squared error along the trajectory starts from zero and increases as time goes by. The mean of this value is influenced by the first values that, independently from the model, are always close to zero, providing not enough performance information.

Another performance parameter is the mean absolute FDE, defined in the following equation:

$$MAE_{FDE} = \frac{1}{M_{test}} \sum_{k=1}^{M_{test}} \left| s_N^k - \hat{s}_N^k \right| \qquad \text{Eq. (3)}$$

Where the only difference from Equation 1 is the absolute value of the error that replaces the squared error.

Figure 4 shows the box plot of the MSE and MAE of each architecture that has been tested. These box plots have been computed starting from the results of 25 training processes, with the following parameters:

- Optimizer: ADAM;
- Learning rate = 0.001, clip value = 1.0;
- Epochs = 300, batch size = 256.

Tables 5 and 6 show some statistics related to the models' performance distributions on the test data, in terms of MSE and MAE, defined by Equations 2 and 3. In the normality Jarque–Bera test, there is not enough statistical evidence to reject H_0, that is the null hypothesis that the samples are drawn from a normal distribution. The *p*-values, computed from the chi-squared distribution with two degrees of freedom, having as argument the Jarque–Bera statistics, are indeed always higher than 0.05. The confirmation of normality

FIGURE 4 Mean squared error Equation 2 and mean absolute error Equation 3 of each model.

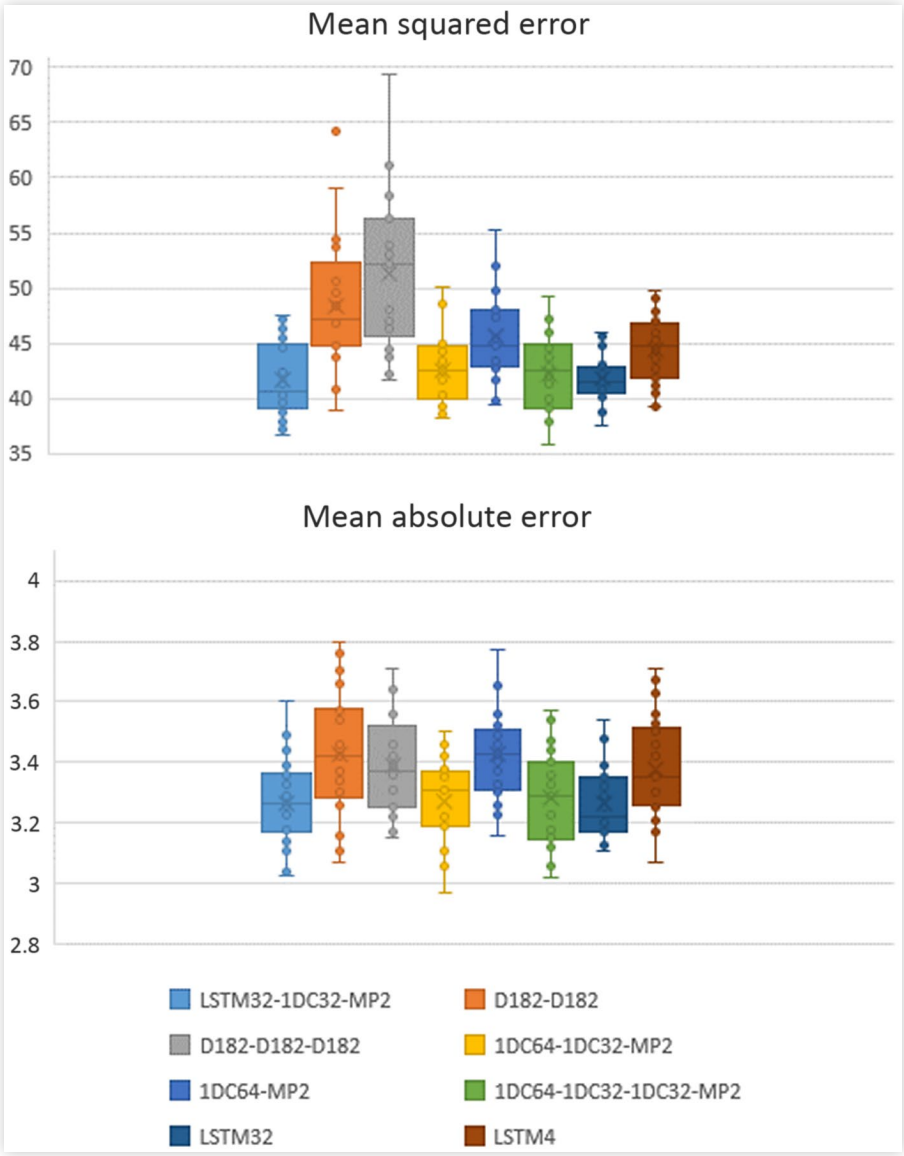


TABLE 5 Statistical information about the MSE of each model. The Jarque–Bera test is accomplished to verify that the data are normally distributed, the *p*-value is referred to the chi-squared distribution with two degrees of freedom. If *p*-value >0.05 there is no sufficient statistical evidence to reject H_0, the null hypothesis of normality of the data.

MSE						
Model	**Mean**	**Std. dev.**	**Skewness**	**Kurtosis (excess)**	**Jarque–Bera**	***p*-value (χ2 test)**
D182-D182	48.40	5.88	0.83	0.86	3.68	0.15
D182-D182-D182	51.23	6.62	0.73	0.70	2.77	0.25
1DC64-MP2	45.56	4.01	0.71	0.05	2.11	0.34
1DC64-1DC32-MP2	42.57	3.05	0.64	0.20	1.77	0.41
1DC64-1DC32-1DC32-MP2	42.22	3.47	0.08	−0.90	0.87	0.64
LSTM32	41.84	2.09	0.21	−0.07	0.19	0.90
LSTM4	44.32	2.99	−0.02	−0.91	0.87	0.64
LSTM32-1DC32-MP2	41.59	3.33	0.44	−1.13	2.25	0.32

TABLE 6 Statistical information about the MAE of each model. The Jarque–Bera test confirms that there is not enough statistical evidence to reject H_0, the null hypothesis of normality of the data.

MAE						
Model	**Mean**	**Std. dev.**	**Skewness**	**Kurtosis (excess)**	**Jarque–Bera**	***p*-value (χ2 test)**
D182-D182	3.42	0.19	0.08	−0.61	0.41	0.81
D182-D182-D182	3.38	0.5	0.34	−0.71	1.02	0.60
1DC64-MP2	3.42	0.14	0.41	−0.18	0.74	0.68
1DC64-1DC32-MP2	3.27	0.13	−0.37	−0.45	0.80	0.67
1DC64-1DC32-1DC32-MP2	3.28	0.15	0.12	−0.94	0.98	0.61
LSTM32	3.26	0.12	0.85	−0.25	3.08	0.21
LSTM4	3.37	0.16	0.37	−0.54	0.88	0.64
LSTM32-1DC32-MP2	3.26	0.13	0.41	0.13	0.74	0.68

of the distributions allows the application of the Student's *t*-test of unequal variances, also called Welch's *t*-test, to check if the model LSTM32-1DC32-MP2, the one with the lowest MSE sample mean, is actually the best-performing one.

Table 7 shows the results of the Welch's *t*-test between the MSE sample of the LSTM32-1DC32-MP2 model and all the others. Some considerations can be deduced:

- The multilayer perceptron (MLP) models, composed by dense layers, are unable to fully capture the complex nonlinearity of driving behavior. The models D182-D182 and D182-D182-D182 perform worse than all the others, independently from the model complexity.
- The LSTM and CNN architectures perform better than the MLP, nonetheless, the model complexity is relevant for the performance. LSTM4 and 1DC64-MP2 have a lower MSE with respect to MLP but their complexity is still too low for an optimal performance.
- There is no statistical difference in the performance, at least for the learning task faced in this article, between LSTM and CNN models. The LSTM32-1DC32-MP2, the 1DC64-1DC32-MP2, the 1DC64-1DC32-1DC32-MP2, and the LSTM32 models reach statistically the same MSE.

As stated, there is no statistical difference in the performance considering the Waymo dataset between the most complex models that include LSTM, CNN, or both layers. Nonetheless, when tested on the NGSIM dataset [27], that is way larger with respect to the Waymo dataset and focused on longitudinal motion, LSTM models outperform the others, showing faster and more stable convergence. Therefore, the LSTM32-1DC32-MP2 is taken as a reference.

Figure 5 shows some episodes predicted with the model LSTM32-1DC32-MP2. The model succeeds to understand some slight change in the maneuver in the first 3 s and to correctly interpret them for the prediction of the next seconds. In Figure 6 the learning curve of the same model is presented. The fact that the performance on the training dataset is close to the performance on the validation dataset ensures the absence of overfitting. Figure 7 illustrates the learning curve of the model 1DC64-1DC32-MP2. Compared to the LSTM32-1DC32-MP2, there is a noticeable increase in noise during the learning process and slower convergence.

4.2. Comparative Analysis with State-of-the-Art and Discussion

In this subsection the results of the best-performing architectures are compared with prediction models that can be found

TABLE 7 Results of the student's *t*-test of unequal variances between the MSE samples of the LSTM32-1DC32-MP2 model and all the others. The results show that there is no statistical significance in the difference between the MSE performance of the LSTM32-1DC32-MP2, the 1DC64-1DC32-MP2, the 1DC64-1DC32-1DC32-MP2, and the LSTM32 models.

Student's *t*-test between the MSE samples: LSTM32-1DC32-MP2 vs					
	Mean	Std. dev.	*t* statistic	d.o.f.	*p*-value
D182-D182	48.40	5.88	5.06	38	1.16E−05
D182-D182-D182	51.23	6.62	6.52	35	1.55E−07
1DC64-MP2	45.56	4.01	3.83	47	3.76E−04
1DC64-1DC32-MP2	42.57	3.05	1.09	49	0.27
1DC64-1DC32-1DC32-MP2	42.22	3.47	0.66	49	0.51
LSTM32	41.84	2.09	0.32	42	0.74
LSTM4	44.32	2.99	3.08	49	3.39E−03

in literature and considered the state-of-the-art. The works that are taken as reference are:

- L. Lin, W. Li, H. Bi, and L. Qin, "Vehicle Trajectory Prediction Using LSTMs with Spatial–Temporal Attention Mechanisms," (2022) [21].
- Min, H., Xiong, X., Wang, P. et al., "A Hierarchical LSTM-Based Vehicle Trajectory Prediction Method Considering Interaction Information," (2024) [22].
- Song, H., Ding, W., Chen, Y., Shen, S., Wang, M.Y., and Chen, Q., "PiP: Planning-Informed Trajectory Prediction for Autonomous Driving," (2020) [24].

The referenced studies were not trained and evaluated using the Waymo dataset, but rather on the NGSIM dataset [27], which includes two freeway segments and two arterial segments. The topology and traffic conditions depicted in the NGSIM dataset render trajectory prediction and longitudinal motion prediction quite comparable. In freeway scenarios, the lateral motion and its associated prediction error are significantly lower than those of longitudinal motion. Essentially, in the context of trajectory prediction on freeways, the primary source of error (by an order of magnitude) originates from predicting longitudinal motion.

To have a proper comparison, it was necessary to train and test the model proposed in this article on the NGSIM dataset.

For the comparison, the root mean squared final displacement error (RMSE) will be used, defined in Equation 2. In [22], errors in both X and Y coordinates are tracked, but only the error in Y is reported here. In [21, 24] the RMSE considers both the X and Y position error.

Comparison In Table 8, the performance of each algorithm is presented in terms of RMSE. All algorithms are evaluated across prediction horizons of 1, 2, 3, 4, and 5 s. Here are some considerations regarding each field in the table:

- Prediction: The algorithms of the reference studies predict the future trajectory of the vehicle analyzed (X and Y coordinates). In [22], the X and Y RMSE are reported separately, enabling an analysis of the Y coordinate, which, in a freeway scenario, can be considered comparable to the progress. The proposed algorithm predicts progress ($\int V$).
- History: It refers to the historical time window that the algorithm receives as input.
- Input features: The features that each algorithm receives in the historical temporal window. The term "target vehicle" refers to the vehicle for which the future trajectory is predicted. These are:
  - X, Y: Coordinates of the target vehicle.
  - $\int V$: Progress of the target vehicle.
  - V: Speed of the target vehicle.
  - A: Acceleration of the target vehicle.
  - Env: Environmental information. The expression describes contextual information or external factors that influence the behavior of the vehicle in question. This information may include distance from the leading vehicle, surrounding traffic, and other environmental factors that can affect vehicle movement. It is important to note that this type of information is often unavailable in scenarios with partial observability, particularly when attempting to predict the behavior of vehicles surrounding the ego vehicle based solely on the ego vehicle's observability.

Some comments for each referenced study:

- L. Lin, W. Li, H. Bi, and L. Qin, "Vehicle Trajectory Prediction Using LSTMs with Spatial–Temporal Attention Mechanisms," (2022) [21]. The study evaluates the prediction only on the short-term horizon of 1 s, predicting the trajectory and taking as input the histories of all the vehicles within a grid centered on the target vehicle to predict.
- Min, H., Xiong, X., Wang, P. et al., "A Hierarchical LSTM-Based Vehicle Trajectory Prediction Method Considering Interaction Information," (2024) [22]. The study predicts the trajectory for all the horizons. The algorithm takes as input the vector {x, y, v_x, v_y, a_x, a_y, L, LH, CH, RH, dL}, which contains the position, speed, and acceleration in every component, the lane L, the

FIGURE 5 Some examples of longitudinal behavior prediction with the LSTM32-1DC32-MP2 model. The evolution of the progress is plotted with respect to the time steps.

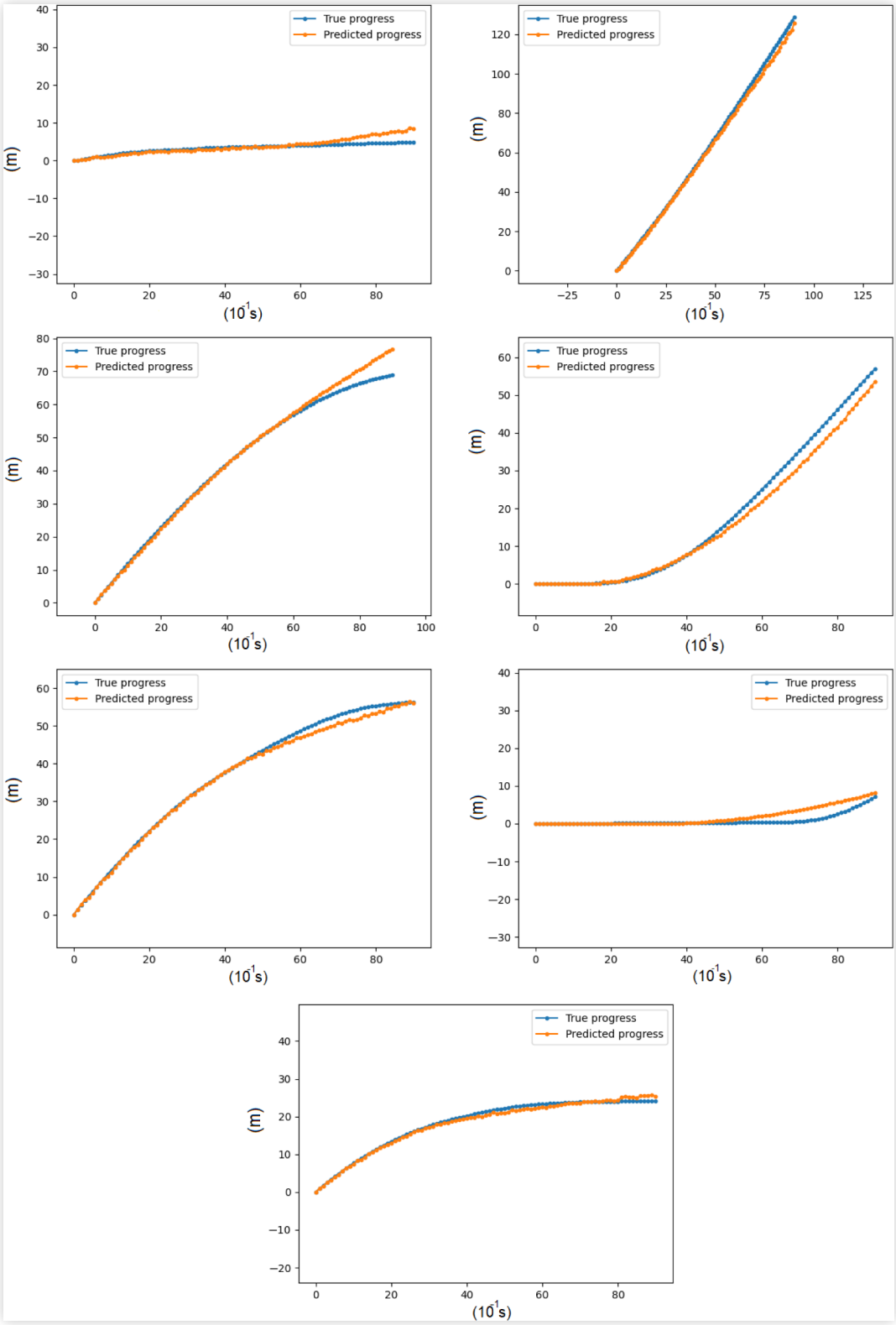
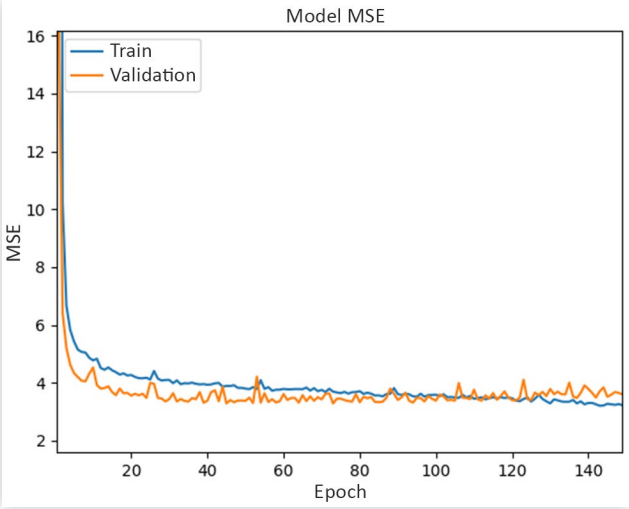


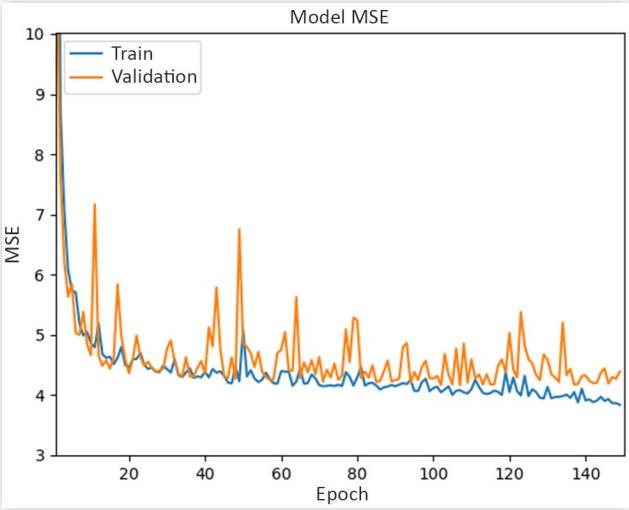
FIGURE 6 Learning curve of the LSTM32-1DC32-MP2 model on the training and validation dataset.



deviation from the center of the lane dL, and the headway of the nearest vehicle in front of the left lane, current lane, and right lane (LH, CH, RH).

- Song, H., Ding, W., Chen, Y., Shen, S., Wang, M.Y., and Chen, Q., "PiP: Planning-Informed Trajectory Prediction for Autonomous Driving," (2020) [24]. The study predicts the trajectory for all the horizon. The algorithm estimates the future states of a set of target vehicles around the ego vehicle, conditioning on the tracking history of all the vehicles surrounding each target vehicle and the planned future of the controllable ego vehicle.
- LSTM32-1DC32-MP2. The model proposed in this article. The algorithm predicts the future progress solely based on the past trajectory of the target vehicle, without considering any environmental information about the surrounding vehicles.
- LSTM32-1DC32-MP2 (*). This variant of the LSTM32-1DC32-MP2 model includes as input the space headway and the distance between the target vehicle and its frontal vehicle as environmental information. This adjustment ensures a fair comparison with other models, all of which also incorporate environmental information.

FIGURE 7 Learning curve of the 1DC64-1DC32-MP2 model on the training and validation dataset.



Discussion

Table 8 shows that the model LSTM32-1DC32-MP2 reaches the state-of-the-art performance in the prediction horizon 1 s, 2 s, 3 s, despite its simplicity and lack of information about vehicles surrounding the target one. However, its performance deteriorates in the 4 s and 5 s horizons (although still comparable with [22] in the 4 s horizon). This decline is attributed to the increased importance of interactions on longer horizons, where relying solely on the target vehicle's history becomes insufficient. To address this limitation and to ensure a fair comparison with other models, the variation LSTM32-1DC32-MP2 (*) has been introduced. This model incorporates the history of the space headway of the target vehicle as input, thus including environmental information. With this addiction on environmental information (space headway), the variation LSTM32-1DC32-MP2 (*) achieves state-of-the-art performance even in the 4 s and 5 s prediction horizon.

The strength of the LSTM32-1DC32-MP2 model lies in its simplicity, which allows for state-of-the-art performance up to a prediction horizon of 3 s, and still achieves good performance for 4 s and 5 s horizons. Its straightforward input structure enables its application in scenarios where only the target vehicle is observable, without access to its surrounding vehicle data. One such scenario is platooning, where the leading vehicle is observable, and despite the unavailability of data for the vehicle in front of it, reliable predictions can be made for the next few seconds. In contrast, other algorithms rely on the history of all vehicles in the traffic scenario, which may not always be available. The state-of-the-art performance of the LSTM32-1DC32-MP2 (*) variation confirms that the weakness in the long-term horizon is attributed to the lack of environmental information rather than an intrinsic flaw in the model (Table 9).

5. Conclusions

This article introduces deep learning models for predicting longitudinal driving behavior. It leverages the characteristics of specific architectures, such as long–short-term memory (LSTM) models and CNNs, architectures considered well-suited for understanding and predicting sequential data, like time series. The models take as input a 3-s time window, consisting of the past progress, speed, and acceleration of a vehicle. They then output the speed and progress profile of

TABLE 8 Comparison of the RMSEs of some state-of-the-art prediction algorithms for different horizons. In the table the variable that is predicted is indicated, the past history and its features received as input. The model LSTM32-1DC32-MP2 is the one proposed in this article. The variant LSTM32-1DC32-MP2 (*) includes as input the space headway, as environmental information. This adjustment ensures a fair comparison with other models, all of which also incorporate environmental information.

	RMSE (m)					Input features						
Paper	1 s	2 s	3 s	4 s	5 s	Pred.	Hist.	X, Y	∫ V	V	A	Env.
Attention LSTMs Lin et al. (2022, [21])	0.56					(X, Y)	3 s	X				X
Hierarchical LSTM Min et al. (2024, [22])	1.39	1.70	3.06	3.52	4.53	Y	5 s	X		X	X	X
PiP Song et al. (2020, [24])	0.55	1.18	1.94	2.88	4.04	(X, Y)	3 s	X				X
LSTM32-1DC32-MP2	0.49	1.33	2.48	3.95	5.69	∫ V	3 s		X	X	X	
LSTM32-1DC32-MP2 (*)	0.50	1.26	2.01	3.14	4.56	∫ V	3 s		X	X	X	X

the vehicle for the next 6 s. Among these models, the hybrid LSTM–CNN architecture is regarded as the most effective in terms of MSE, MAE, convergence speed, and stability. The model achieves state-of-the-art performance, particularly within a 4-s prediction horizon, while maintaining good performance for longer-term predictions. Its simple input structure allows for application in scenarios where only the target vehicle is observable, without access to data from surrounding vehicles. This is common in situations lacking V2I or V2V communication. In contrast, other algorithms rely on the history of all surrounding vehicles, which may not always be available.

Furthermore, the performances of multilayer perceptron (MLP), LSTM, and one-dimensional CNN architectures are compared. This article contributes to the discussion by providing a statistical analysis to determine which of these architectures better suit the task of comprehending and predicting time series. The results of this analysis can be summarized in the following points:

- MLP models, composed of dense fully connected layers, struggle to fully capture the complex nonlinearity of driving behavior.
- The LSTM and CNN architectures outperform the MLP. However, it's important to note that model complexity plays a significant role in achieving optimal performance.
- There is no statistical difference in performance, at least for the learning task addressed in this article, between LSTM and CNN models.

TABLE 9 Comparison between the two versions of the LSTM32-1DC32-MP2 model (without and with the information about the space headway) in terms of FDE.

	FDE (m)				
Model	1 s	2 s	3 s	4 s	5 s
LSTM32-1DC32-MP2	0.32	0.94	1.84	2.97	4.33
LSTM32-1DC32-MP2 (*)	0.32	0.82	1.47	2.33	3.42

In conclusion, while this study provides valuable insights into deep learning algorithms for predicting longitudinal driving behavior, there are several avenues for future research and development. Exploring the possible application of generative models in this field, investigating the impact of additional features, and adapting the model for lateral behavior prediction are promising directions, which could enhance the applicability and robustness of the proposed framework. Additionally, integrating the proposed deep learning approach into decision-making algorithms as a module for understanding driving behavior would provide a characterization for each agent, thereby improving the overall decision-making process. These potential future directions open new horizons for understanding and applying deep learning in predicting driving behavior and enhancing decision-making processes.

Acknowledgements

This publication was made using the Waymo Open Dataset, provided by Waymo LLC under license terms available at waymo.com/open.

Contact Information

Giovanni Lucente, corresponding author
giovanni.lucente@dlr.de

References

1. Ajzen, I., "The Theory of Planned Behavior," *Organizational Behavior and Human Decision Processes* 50, no. 2 (1991): 179-211, doi:https://doi.org/10.1016/0749-5978(91)90020-T.
2. Waymo, "Waymo Open Dataset: An Autonomous Driving Dataset," 2019, https://www.waymo.com/open.
3. Miles, D.E. and Johnson, G.L., "Aggressive Driving Behaviors: Are There Psychological and Attitudinal Predictors?" *Transp. Res. Part F Traffic Psychol. Behav.* 6 (2003): 147-161, doi:https://doi.org/10.1016/S1369-8478(03)00022-6.
4. Vanlaar, W., Simpson, H., Mayhew, D., and Robertson, R., "Aggressive Driving: A Survey of Attitudes, Opinions and Behaviors," *J. Saf. Res.* 39 (2008): 375-381, doi:https://doi.org/10.1016/j.jsr.2008.05.005.
5. Wang, W. and Xi, J., "A Rapid Pattern-Recognition Method for Driving Styles Using Clustering-Based Support Vector Machines," in *2016 American Control Conference (ACC)*, Boston, MA, 2016, 5270-5275, https://doi.org/10.1109/ACC.2016.7526495.
6. Khanfar, N.O., Ashqar, H.I., Elhenawy, M., Hussain, Q. et al., "Application of Unsupervised Machine Learning Classification for the Analysis of Driver Behavior in Work Zones in the State of Qatar," *Sustainability* 14, no. 22 (2022): 15184, doi:https://doi.org/10.3390/su142215184.
7. Brombacher, P., Masino, J., Frey, M., and Gauterin, F., "Driving Event Detection and Driving Style Classification Using Artificial Neural Networks," in *2017 IEEE International Conference on Industrial Technology (ICIT)*, Toronto, ON, Canada, 2017, 997-1002, https://doi.org/10.1109/ICIT.2017.7915497.
8. MacAdam, C., Bareket, Z., Fancher, P., and Ervin, R., "Using Neural Networks to Identify Driving Style and Headway Control Behavior of Drivers," *Veh. Syst. Dyn.* 29 (1998): 143-160, doi:https://doi.org/10.1080/00423119808969557.
9. Xie, J. and Zhu, M., "Maneuver-Based Driving Behavior Classification Based on Random Forest," *IEEE Sensors Letters* 3, no. 11 (2019): 1-4, doi:https://doi.org/10.1109/LSENS.2019.2945117.
10. Mohammadnazar, A., Arvin, R., and Khattak, A.J., "Classifying Travelers' Driving Style Using Basic Safety Messages Generated by Connected Vehicles: Application of Unsupervised Machine Learning," *Transportation Research Part C: Emerging Technologies* 122 (2021): 102917, doi:https://doi.org/10.1016/j.trc.2020.102917.
11. Wang, W., Xi, J., Chong, A., and Li, L., "Driving Style Classification Using a Semisupervised Support Vector Machine," *IEEE Transactions on Human-Machine Systems* 47, no. 5 (2017): 650-660, doi:https://doi.org/10.1109/THMS.2017.2736948.
12. Rossi, L., Ajmar, A., Paolanti, M., and Pierdicca, R., "Vehicle Trajectory Prediction and Generation Using LSTM Models and GANs," *PLoS One* 16, no. 7 (2021): e0253868, doi:https://doi.org/10.1371/journal.pone.0253868.
13. Wenxiang, X., Wang, J., Ting, F., Gong, H. et al., "Aggressive Driving Behavior Prediction Considering Driver's Intention Based on Multivariate-Temporal Feature Data," *Accident Analysis & Prevention* 164 (2022): 106477, doi:https://doi.org/10.1016/j.aap.2021.106477.
14. Yang, L., Zhao, C., Lu, C., Wei, L. et al., "Lateral and Longitudinal Driving Behavior Prediction Based on Improved Deep Belief Network," *Sensors* 21 (2021): 8498, doi:https://doi.org/10.3390/s21248498.
15. Eppink, J., Kolff, M., Venrooij, J., Pool, D. et al., "Probabilistic Prediction of Longitudinal Driving Behaviour for Driving Simulator Pre-Positioning," in *22nd Driving Simulation & Virtual Reality Conference 2022 Europe*, Antibes, France, 2023, http://resolver.tudelft.nl/uuid:61588ae9-bbba-4afe-85ab-a6e277f1148f.
16. Xu, H., Zhang, Y., Qin, Y., Gao, M. et al., "Modelling of Longitudinal and Lateral Behavior of Drivers and Automatic Prediction-Evaluation," in *2023 6th International Conference on Electronics Technology (ICET)*, Chengdu, China, 2023, 1199-1206, https://doi.org/10.1109/ICET58434.2023.10211701.
17. Kolekar, S., de Winter, J., and Abbink, D., "Human-Like Driving Behaviour Emerges from a Risk-Based Driver Model," *Nat Commun* 11 (2020): 4850, doi:https://doi.org/10.1038/s41467-020-18353-4.
18. Nikhil, N. and Morris, B.T., "Convolutional Neural Network for Trajectory Prediction," in: Leal-Taixé, L. and Roth, S. (eds.), *Computer Vision – ECCV 2018 Workshops*, Lecture Notes in Computer Science, Vol. 11131 (Cham: Springer, 2019), https://doi.org/10.1007/978-3-030-11015-4_16.
19. Nan, S., Ran, T., Li, T., Sun, J. et al., "From Driving Behavior to Energy Consumption: A Novel Method to Predict the Energy Consumption of Electric Bus," *Energy* 261, no. Part A (2022): 125188, doi:https://doi.org/10.1016/j.energy.2022.125188.
20. Zhang, C., Che, G., and Gao, B., "Vehicle Driving Behavior Predicting and Judging Using LSTM and Statistics Methods," in *2020 4th CAA International Conference on Vehicular Control and Intelligence (CVCI)*, Hangzhou, China, 2020, 199-203, https://doi.org/10.1109/CVCI51460.2020.9338600.
21. Lin, L., Li, W., Bi, H., and Qin, L., "Vehicle Trajectory Prediction Using LSTMs with Spatial–Temporal Attention Mechanisms," *IEEE Intelligent Transportation Systems Magazine* 14, no. 2 (2022): 197-208, doi:https://doi.org/10.1109/MITS.2021.3049404.
22. Min, H., Xiong, X., Wang, P. et al., "A Hierarchical LSTM-Based Vehicle Trajectory Prediction Method Considering Interaction Information," *Automot. Innov.* 7 (2024): 71-81, doi:https://doi.org/10.1007/s42154-023-00261-0.
23. Park, D., Ryu, H., Yang, Y., Cho, J. et al., "Leveraging Future Relationship Reasoning for Vehicle Trajectory Prediction," 2023, https://doi.org/10.48550/arXiv.2305.14715.
24. Song, H., Ding, W., Chen, Y., Shen, S. et al., "PiP: Planning-Informed Trajectory Prediction for Autonomous Driving,"

in: Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.M. (eds.), *Computer Vision – ECCV 2020*, Lecture Notes in Computer Science, Vol. 12366 (Cham: Springer, 2020), https://doi.org/10.1007/978-3-030-58589-1_36.

25. Gilles, T., Sabatini, S., Tsishkou, D.V., Stanciulescu, B. et al., "THOMAS: Trajectory Heatmap Output with Learned Multi-Agent Sampling," 2021, https://doi.org/10.48550/arXiv.2110.06607.
26. Hochreiter, S. and Schmidhuber, J., "Long Short-Term Memory," *Neural Computation* 9 (1997): 1735-1780, doi:https://doi.org/10.1162/neco.1997.9.8.1735.
27. U.S. Department of Transportation Federal Highway Administration, "Next Generation Simulation (NGSIM) Vehicle Trajectories and Supporting Data," Provided by ITS DataHub through Data.transportation.gov, 2016, https://doi.org/10.21949/1504477.