Proceedings of the 6th International Conference of Control, Dynamic Systems, and Robotics (CDSR'19)
Ottawa, Canada – June 6-7, 2019
Paper No. CDSR 132
DOI: 10.11159/cdsr19.132

Short Term Predictions of Preceding Vehicle Speeds for Connected and Automated Vehicles

Mehran Zamani Abnili[1], Nasser L. Azad[2]
[1]University of Waterloo, Mechanical and Mechatronics Engineering Department
200 University Ave. West, Waterloo, Canada
mzamania@uwaterloo.ca
[2]University of Waterloo, Systems Design Engineering Department
200 University Ave. West, Waterloo, Canada
nlashgar@uwaterloo.ca

***Abstract* –** The pursuit of fully automated driving has been a significant point of interest among researchers and industrial communities alike in recent years. As instantly replacing every vehicle with its automated counterpart is implausible, autonomous vehicles need to operate alongside human driven ones. Human drivers tend to show a lot of variation in their driving behaviour, and making non-optimal decisions is a frequent practice. This chaotic environment makes it difficult for controllers on-board automated vehicles to make optimal decisions. Given advanced control techniques, such as model predictive controllers, that can make use of a valid prediction of other traffic participants' behaviours for a significant performance boost, a method to successfully make such predictions will become appealing. In this paper, considering a host connected and automated vehicle or CAV, the performance of a recurrent neural network is investigated for this task using some standard driving cycles data to predict the velocity of the preceding vehicle for multiple horizons in urban driving scenarios.

***Keywords*:** Prediction, Preceding vehicle, Recurrent neural network, Connected and automated vehicles

1. Introduction

Since the dawn of experiments on automated driving systems in the 1920s, the world of technology has seen a lot of progress, especially in the hardware domain. A vast variety of sensors have become available, computers have become orders of magnitude smaller, more affordable and more powerful, a global-inclusive interconnection of devices is imminent with emergence of IoT and the 5th generation of wireless mobile communications, [1] on the software side also, countless control strategies have been studied and experimented with. However, achieving full autonomy in transportation systems is still a very complex course. The highest level of autonomy achieved in mass production cars, as of Q1, 2019 is at level 3 in the SAE scale of autonomy and is very restricted in terms of operational conditions. [2,3] One major drag, slowing the advancement of fully autonomous driving is due to the fact that all vehicles cannot be instantly replaced by their fully autonomous, self-driving counterparts. The environment of an automated vehicle is chaotic due to inappropriate infrastructure and uncertainties associated with human drivers. As a consequence, a lot of recourses need to be invested in an attempt to imitate human driver behaviour rather than a strategy that includes every vehicle as its agent. As it seems impossible to skip this transition period, as chaotic as it may be, it is absolutely crucial to understand human driving behaviours and develop a method to predict surrounding vehicles intents for a short horizon. A good approach for such complex problems is utilizing machine learning and soft-computing techniques, and such is the intent of this study.

Given the interest for having accurate predictions for surrounding traffic participants' behaviour, particularly vehicle speeds in this case of autonomous driving, several studies have been conducted in this area. Although applications may vary from case to case, the goals are alike. Zhang e*t al.* [4] studied "chaining neural networks (CNN)" to process VISSIM data, built on Wiedemann's car following model, by assuming availability of V2X communications. A broader study was conducted by Sun *et al.* [5] to compare three different velocity prediction strategies: exponentially varying, stochastic Markov chain and neural network based. Focusing on the energy management of hybrid electric vehicles (HEVs), and also the application of the mentioned predictors for model predictive controllers (MPCs). Gong *et al.* [6] *to*ok another approach and used dynamic programming method to predict the velocity by utilizing knowledge of the environment *a priori* using GPS and GIS data to minimize fuel consumption. He [7] took a linear regression approach to predict the U.S. Environment Protection Agency (EPA) drive cycles to minimize plug-in HEVs (PHEVs) power consumption. Murphey *et al.* [8] utilized neural networks and dynamic programming to classify the driving environment using various standard drive cycles from EPA. Bender *et al.* [9] took a rule based strategy to predict drive cycles for hybrid hydraulic vehicles and also aimed to optimize the vehicle's energy management with the results. Finally, Thorsell [10] took a neural network approach to predict speed profiles, specifically recurrent neural networks (RNNs), long short-term memory networks (LSTMs) and gated recurrent unit (GRUs). In this study a recurrent neural network is utilized to make predictions for various horizons on the speed of a preceding vehicle.

2. Setup

The focus of this study is mostly to find a strategy to predict the movement of surrounding vehicles and does not concern hardware choices. The assumptions made for the data stream from sensors, is that the ego-vehicle is a CAV adequately equipped, to be able to measure the velocity of its peers, and is also given an estimate of its distance from traffic signals that may force a full stop upon it. The choices will be discussed in more detail under Section 3 (Background and network topology); however, in essence, at least one companion variable with correlation to the variable of interest can increase the performance of this method significantly. Also discussed further in Section 4 (Future work) is that this method only one part of the prediction methodology that can also perform stand-alone with tweaked inputs. By replacing the *'distance-to-stop-sign'* with the outputs of a learning statistical graphical model, the prediction method will be more sophisticated and can consider many more case scenarios. This marriage is currently being studied in our team.

The datasets used for training of the method are acquired from the United States Environment Protection Agency (EPA). Illustrated in Fig. 1, the datasets are titled: "EPA Urban Dynamometer Driving Schedule" (often referred to as 'UDDS', 'LA-4' or 'the city test'), "New York City Cycle Driving Schedule", and finally, "LA92 'Unified' Dynamometer Driving Schedule". These datasets are especially well-suited for this application, as they maintain the stop and go situation throughout, while introducing variation in the overall behaviour and tone of driving. These variations come in both length of action (e.g. time spent stationary), and intensity of action (e.g. magnitude of velocity) flavours. Some pre-processing was done on the datasets to prepare them to be used in the neural network training process, which will be discussed in the next section.

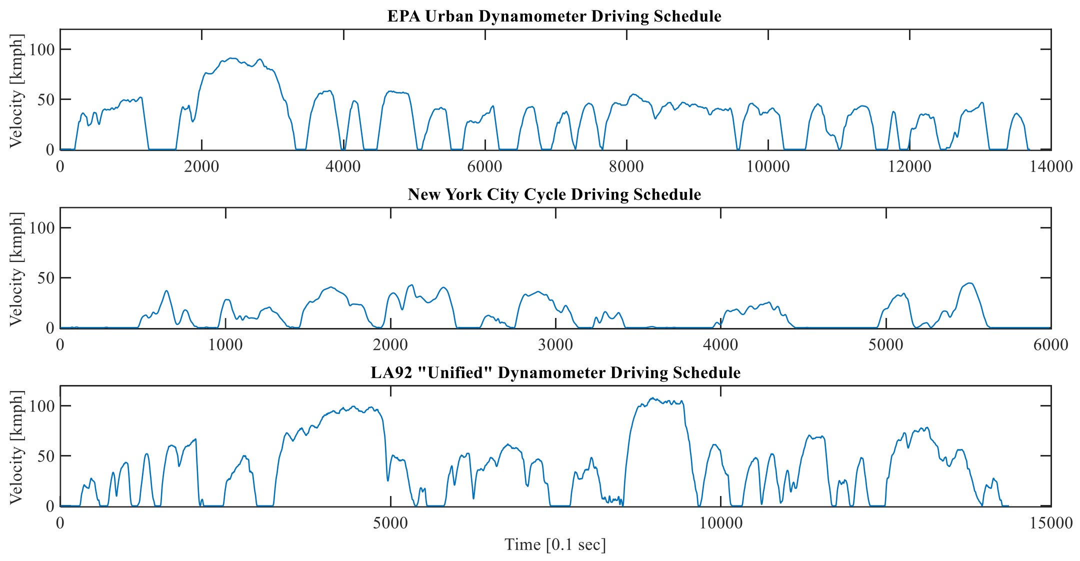

Fig. 1: Datasets used to train the neural network

3. Background and network topology

The method of choice to make the predictions is by utilizing a recurrent neural network. Recurrent neural networks (RNNs) are able to incorporate “memory” to otherwise feed-forward neural networks by cascading the data into subsets of of equal length, reintroducing the history of one or more variables and updating that variable’s time-series by the neural network’s output. Fig. 2 illustrates the network topology for a recurrent neural network with two sets of variables (*x* and *y*) *y*) each with *d* elements of history, Φ activation function on every neuron in the hidden layers, and z^{-1} serving as unit delay.

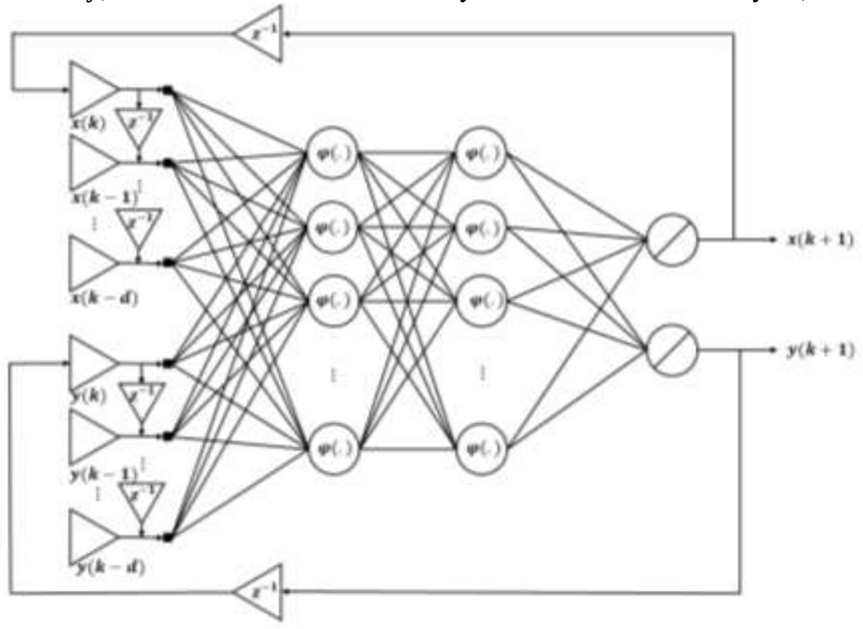

Fig. 2: Network topology of a RNN with 2 sets of data.

The network investigated in this paper is a fully connected RNN consisting of 5 hidden layers with sigmoid activation functions. Number of neurons in each layer were chosen with the following equation, where *‘N’* denotes a fixed number representing the size of its subscript, and *‘n’* denotes the layer identifier, being 1 at the first hidden layer and *i* at the *i*th hidden layer:

$$N_{neurons\ in\ layer} = (N_{inputs} + N_{outputs}) * (N_{hidden\ layers} - n_{current\ layer} + 1) \tag{1}$$

For the case of sigmoid activation function, we can write:

$$\Phi(v) = \frac{1}{(1 + e^{-v})} \tag{2}$$

$\Phi(v)$ being an elementwise function of vector *v*, the input for next hidden layer can be computed by multiplying the output of the current hidden layer by the weights of the connecting edges ($v_n = \Phi(v_{n-1}) \times W_{n-1|n}$). What is referred to as ‘training the network’ is, in essence, updating the weight matrices between each layer *n-1* and *n*, ($W_{n-1|n}$), such that the error between the expected output and network output for the same input is minimized. For that purpose, the gradient descent algorithm is used, which requires the derivative of the activation function, $\Phi(v)$, found below:

$$\Phi'(\nu) = \Phi(\nu) \circ (1 - \Phi(\nu)) \tag{3}$$

In order to apply this method to the aforementioned datasets, some pre-processing had to be done on the data. A design choice to make the employment of the network for a real-life scenario plausible, had to be choosing the history, or memory window small enough, so that the network would not take an unreasonably long time to come online, but long enough, so that it would have enough data to work with, and to be able to produce accurate predictions. This window was chosen to be 1.5 seconds. For the back propagation algorithm, each dataset was split in half, the first half was used for training and the second half for testing and validation purposes. To have a large enough dataset for training, and to be able to incorporate 1.5 second memory of movement history, the datasets were up-sampled using low-pass interpolation by a factor of 10, so that the final dataset would have a sampling rate of 10Hz. This way, 1.5 seconds of movement history would be 15 samples. Finally, to extract the companion variable, namely, *'distance-to-stop-sign*', the data was integrated to result in the global position, and then the location at which the first instance of each full-stop occurred would be flagged as a stop-sign location. From there, the smallest positive distance of the *'distance-to-stop-sign'* set was considered.

4. Results

The network predictions have been illustrated in Fig. 3 to 5 for 1, 2, 5, and 10 second horizons. The network trained with the UDDS data seems to capture the changes in the speeds relatively well (Fig. 3a), especially for smaller horizons. From 700 deciseconds to 2500 deciseconds, the network prediction is overestimating the speed of the vehicle for the most part. That is due to the fact that after integrating the velocity to find the stop sign locations, the area underneath was very large, therefore for the network the next stop sign is too far for the vehicle to decrease its speed. This can be alleviated by replacing the accompanying variable with a different value, or capping its value to a threshold. Although capping the *'distance-to-stop-sign'* may increase the accuracy of velocity predictions, finding the optimal maximum value would be a challenge, also this could result in loss of useful information that would have otherwise been provided to the network. The general trend is that as the prediction horizon increases, the accuracy decreases, which is expected. The metric used to assess the accuracy of the network was the correlation between the actual data and the prediction, as well as the mean absolute error. The error bounds are illustrated in Fig. 3 through Fig. 5.

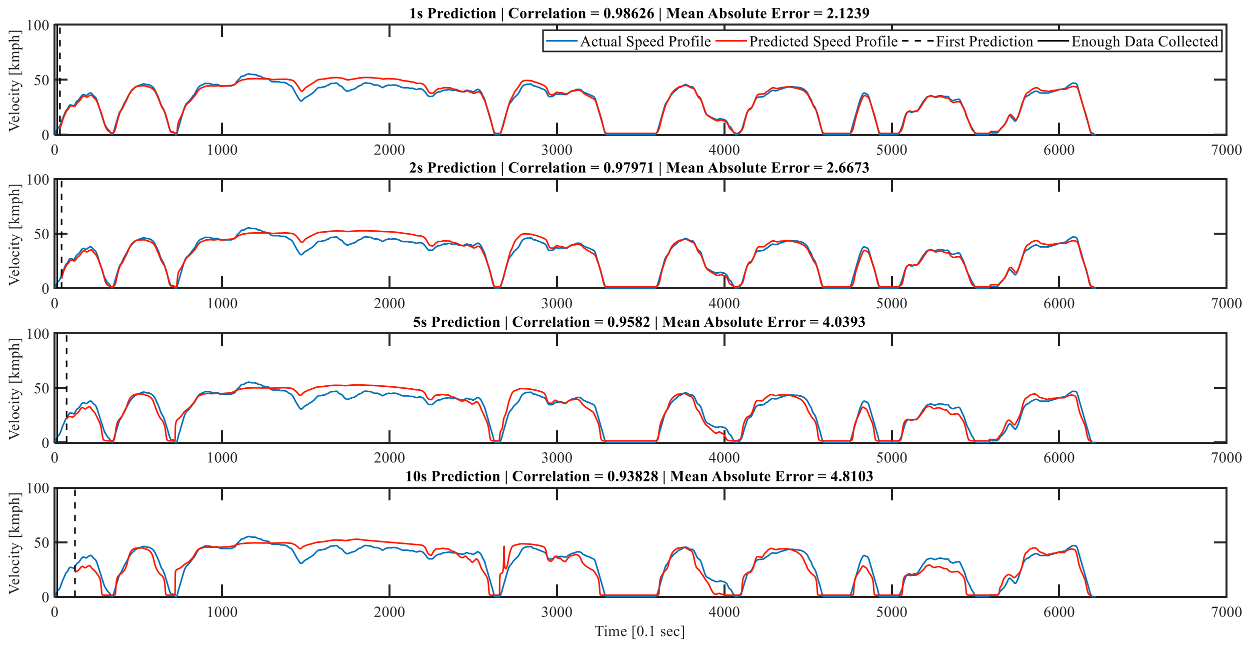

Fig. 3a: Prediction results for UDDS data.

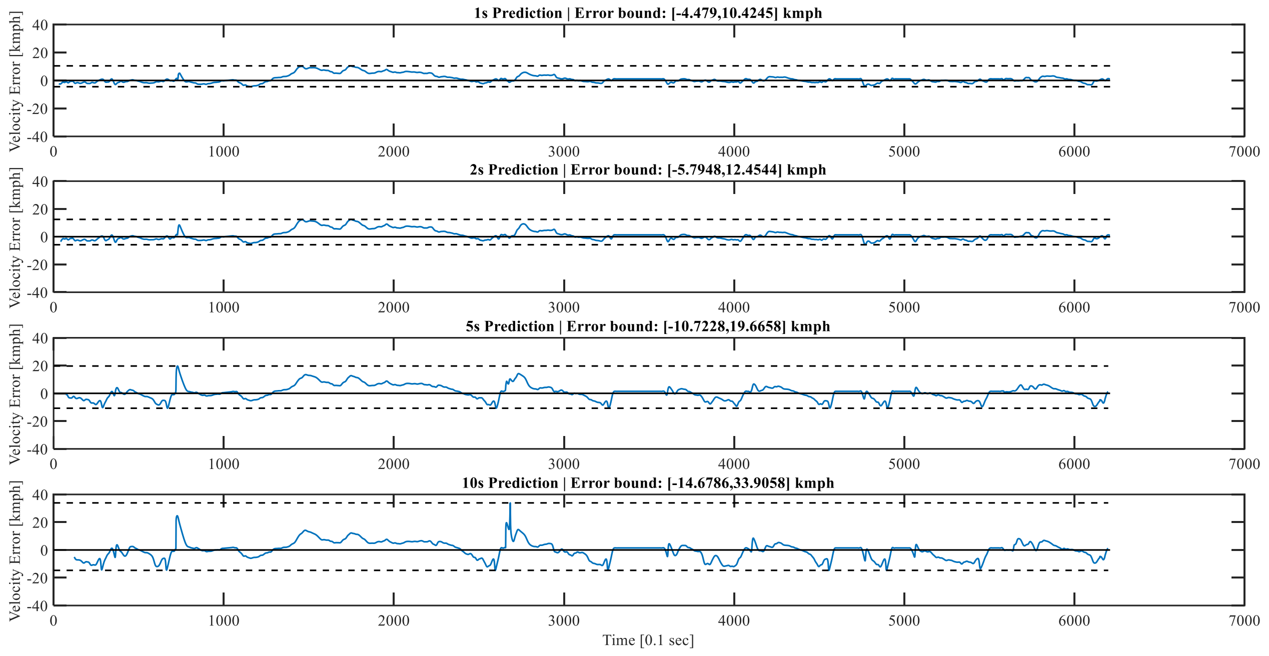

Fig. 3b: Errors for UDDS data.

In the New York City Cycle Driving Schedule (Fig. 4a and 4b), more loss in the accuracy is observed as the prediction horizon increases, which is due to having less correlation between the main variable, velocity, and the accompanying variable, *'distance-to-stop-sign'*. In other words, the accompanying variable does not capture the variations in the velocity due to integration. However, for shorter prediction horizons, the accuracy is still relatively high.

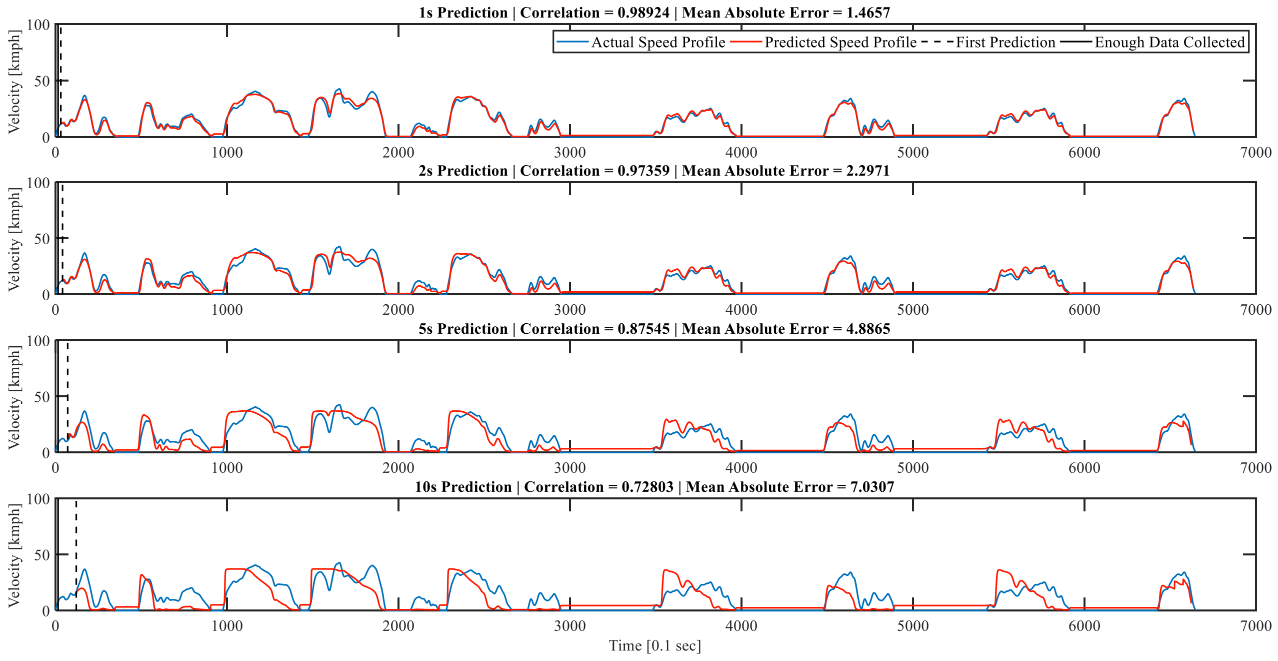

Fig. 4a: Prediction results for NYC data.

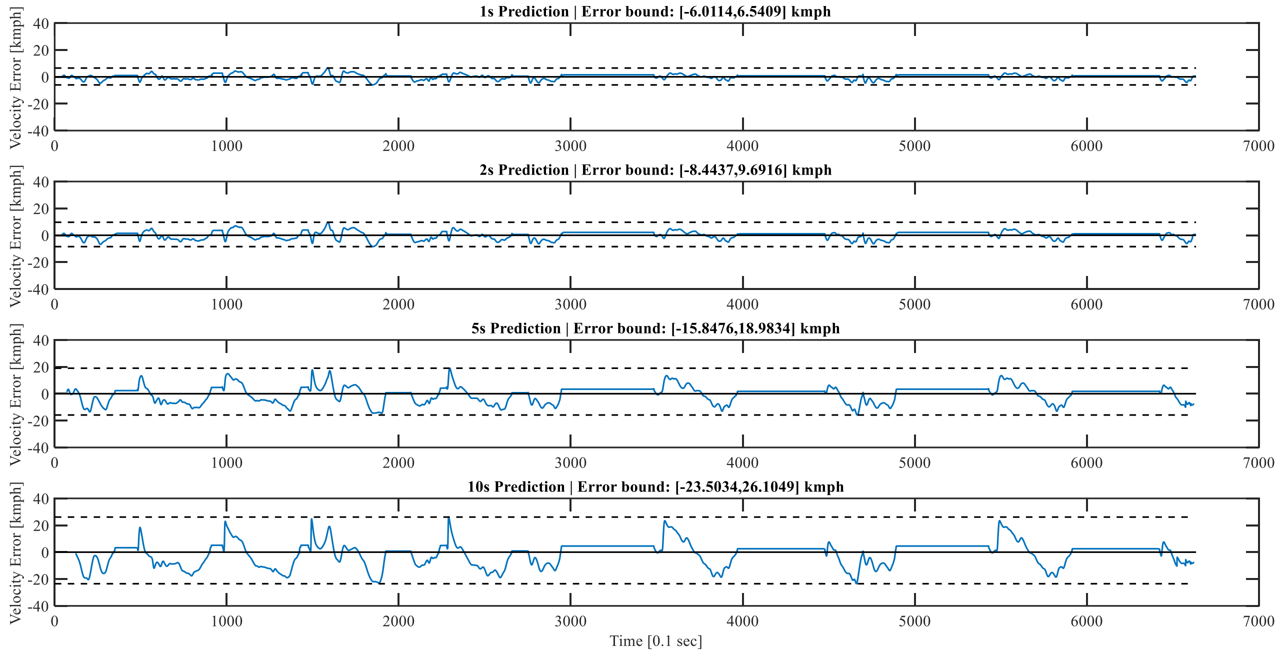

Fig. 4b: Errors for NYC data.

The Results for LA92 (Fig. 5a and 5b) are very similar to that of UDDS but with higher accuracy due to larger data. The over-estimation of velocity in the predictor in sections with large area under the curve is again apparent due to the same reason.

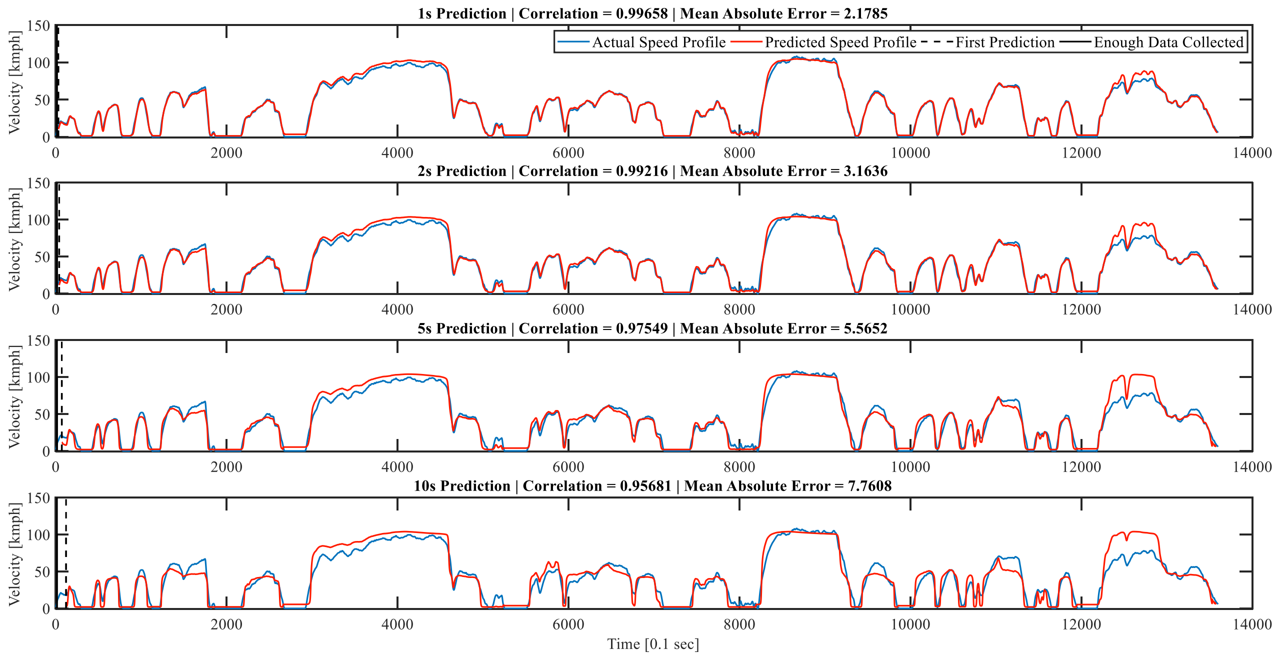

Fig. 5a: Prediction results for LA92 data.

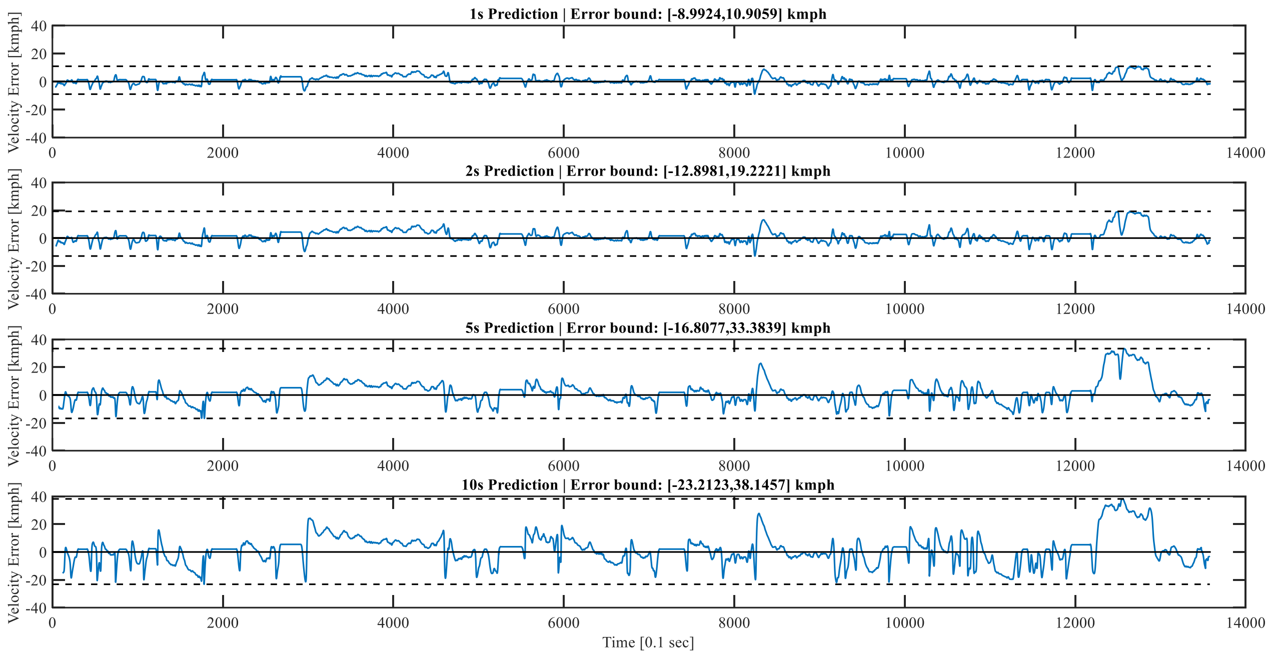

Fig. 5b: Errors for LA92 data.

Considering the error figures and comparing them to the prediction results, it is apparent that the errors peak at instants in which there is a sudden change in the velocity, but then soon after the change is captured by the network, the errors diminish. This behaviour is evident in the results for the New York City drive cycle, due to a lower number of sample points. The low number of sample points, especially takes its toll on higher prediction horizons as seen in Fig. 4a. In general, neural networks tend to rely heavily on the quality of the data that they are trained with. In this application, the best kind of dataset is one that is large, and consists of a fair amount of variations in both intensity and duration of actions. This is supported by superior results yielded by LA92 and UDDS datasets, as well as the large error figures in the NYC data. The significance of this is especially pronounced when compared to the approach taken in [4]. For a one-dimensional car-following application, variations in the magnitudes of speed are either completely removed from the process or are minimally introduced in a variable fixed-maximum velocity manner (i.e. when the vehicle speed reaches its local maxima, it will continue with the same speed until it is forced to stop), which although relatively simplifies the problem, in turn cripples the method in terms of generalization. Considering the absolute values for velocity and general motion, grants the benefit to take advantage of this method in considerably more cases and scenarios, such as merging in highways, intersections, and roundabouts. Another substantial point to consider is the loss of accuracy as prediction horizon increases. In like manner, this can be mitigated by increasing the size of the training dataset. As the training data grows, larger horizons gain more validity. The accuracy loss due to increased prediction horizon is also evident in the results of [5] as the predicted velocity is consistently divergent. Although this may be due to different plotting styles, a pairwise comparison between the results of the method discussed in this study with the results of the neural network based method in [5] show a fundamental difference. In [5], the results exhibit a kind of passiveness as the predictions keep advancing in the same course. Whereas in this study, apparent in the peak of error in UDDS results at 2600 decisecond of 10 second prediction horizon, it seems as if the network is actively correcting its results. Additionally, although training times may be trivial for the aforementioned datasets as the training is done offline, they are a good indication of scalability of the solution and can be found in Table 1.

Table 1: Training times for the dataset using Intel Core i7 5820K @3.4GHz.

Dataset	UDDS	NYC	LA92
Time [s]	33972.6	25022.4	42761.3

Moreover, the turnaround time for 10 second prediction horizon was on average ≈0.0246 seconds (40.65 Hz), with very low deviation, which with 10Hz sampling rate makes it real-time applicable with a large headroom. These figures can be further improved by migrating the method to a lower level programming language such as C++ or Java.

4. Future Work and Conclusion

As mentioned in Section 2 (Setup), this method is meant to be a companion for a learning statistical graphical model to handle low level, continuous variables, for instance, vehicle speeds. The statistical predictions of high level discrete variables, such as lane ID, can replace the *'distance-to-stop-sign'*. The authors speculate that by doing so, the accuracy of predictions will increase significantly, and will make the method more versatile. This speculation is also supported by our analysis of the results, especially for the UDDS and LA92 datasets. The overestimation of the velocity for the sections that the area underneath the velocity curve is too large for the network can be alleviated by replacing *'distance-to-stop-sign'* as discussed in the results section. Though, it seems that even without the incorporation of the statistical methods, the network is able to make decent predictions as a stand-alone module, especially for shorter horizons which are more meaningful to most controllers that can make use of the prediction data. Also, multiple workarounds can be applied given the source of error is identified, which is calibrating the method in the post. These calibrations can be done in a wide variety of ways, the most promising of which would be sequential training of the network after the deployment with real data collected in each performance.

5. Acknowledgements

The authors would like to thank Toyota, NSERC, and OCE for sponsoring this research project.

6. References

[1] TELCOMA (2018, July 26). “5g Technology Introduction” [Online]. Available: https://telcomaglobal.com/blog/17780/5g-technology-introduction

[2] M. McAleer (2017, July 11) “Audi’s self-driving A8: drivers can watch YouTube or check emails at 60km/h” [Online]. Available: https://www.irishtimes.com/life-and-style/motors/audi-s-self-driving-a8-drivers-can-watch-youtube-or-check-emails-at-60km-h-1.3150496

[3] *SAE Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*, SAE Standard J3016, 2018.

[4] F. Zhang, J. Xi, and R. Langari, “Real-Time Energy Management Strategy Based on Velocity Forecasts Using V2V and V2I Communications,” IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 2, pp. 416–430, 2017.

[5] C. Sun, X. Hu, S. J. Moura, and F. Sun, “Velocity Predictors for Predictive Energy Management in Hybrid Electric Vehicles,” IEEE Transactions on Control Systems Technology, vol. 23, no. 3, pp. 1197–1204, 2015.

[6] Q. Gong, Y. Li, and Z.-R. Peng, “Trip-Based Optimal Power Management of Plug-in Hybrid Electric Vehicles,” IEEE Transactions on Vehicular Technology, vol. 57, no. 6, pp. 3393–3401, 2008.

[7] Y. He, “Vehicle-Infrastructure Integration Enabled Plug-In Hybrid Electric Vehicles for Energy Management” Ph.D. dissertation, Dept. Civil. Eng., Clemson Univ., Clemson, SC, 2013.

[8] Y. L. Murphey, Z. Chen, L. Kiliaris, J. Park, M. Kuang, A. Masrur, A. Phillips, “Neural learning of Driving Environment Prediction for Vehicle Power Management” in *International Joint Conference on Neural Networks, 2008*, pp. 3775-3761.

[9] F. A. Bender, M. Kaszynski, and O. Sawodny, “Drive Cycle Prediction and Energy Management Optimization for Hybrid Hydraulic Vehicles,” IEEE Transactions on Vehicular Technology, vol. 62, no. 8, pp. 3581–3592, 2013.

[10] E. Thorsell, “Vehicle speed-profile prediction without spatial information” Master’s Thesis, Dept. Computer Science, Chalmers Univ. of Technology, Gothenburg, Sweden, 2013.