2018 21st International Conference on Intelligent Transportation Systems (ITSC)
Maui, Hawaii, USA, November 4-7, 2018

Connections between classical car following models and artificial neural networks

Fangyu Wu[1], Daniel B. Work[2]

***Abstract*— This article considers the problem of traffic modeling via modeling at the microscopic (i.e., vehicle) scale. It provides a connection between classical ordinary differential equation based models and data driven *artificial neural network* (ANN) based models by showing an example of a car following model which can be exactly expressed as an ANN. In a set of numerical experiments, four ANN models (ranging in structure from a model that is able to exactly capture a classical car following model, to a generic neural network model) are proposed and then trained from data and their resulting accuracy is assessed. It is shown that by adding structure into the neural network (i.e., via the architecture and the activation functions), it is possible to outperform generic ANN models to emergent phenomena such as stop and go waves.**

I. Introduction

Modelling longitudinal car following behaviors has been an important area of research in transportation systems over the last 60+ years [1]–[6]. An accurate car following model enables realistic microscopic simulation and allows for reliable automotive control. These models are particularly important today because the rapid innovation in vehicular autonomy has motivated interests in training self driving cars in simulations based on these microscopic car following models [7].

At the same time, the increased sensing capabilities of vehicles are ushering in a new era of data that may be used to enhance the fidelity of these microscopic car following models. Inspired by the recent successes of deep learning, several works are beginning to investigate the potential to apply *artificial neural networks* (ANN) for car following modelling [8]–[11]. For example, an important ANN is the *multilayer perceptron* (MLP), which is used in [8]–[10].

Recognizing the strong legacy of microscopic car following theory and its successful applications, and at the same time the quickly evolving landscape of data driven machine learning models [12]–[14], in this article we explore some connections between the two approaches. For simplicity, we consider the *full velocity difference model* (FVDM) [15] as a prototypical *ordinary differential equation* (ODE) based microscopic car following model, and on the opposite end of the spectrum we consider *artificial neural networks* (ANN) as an exemplar machine learning based approach.

Using these two models, we illustrate that there exists an artificial neural network which is mathematically equivalent to the FVDM. This is achieved by intelligently choosing the so called *activation function* used in the ANN, and also by imposing a specific structure on the ANN that allows the inputs to be combined in a way that mimics the FVDM. Furthermore, inspired by the connection, we illustrate that it is possible to consider other structured ANNs that are able to capture the general behavior of the FVDM.

With the connection between the data-driven ANN and the ODE-based FVDM established, we can more fully explore ANN based predictors. On the opposite extreme of a carefully constructed ANN outlined above, we also consider a generic ANN based model, in which the architecture and the activation function do not exploit any specific structure or model-specific choices.

Since both the structured ANNs and the generic ANN are neural networks, they can be directly compared using the same calibration procedure to assess their modeling performance. Along these lines, we run a set of numerical experiments to highlight the performance variability of the various models. The main message uncovered from these experiments is the possibility of fine tuning the ANN to connect with the classical car following theory offers potential to produce a new structured ANN models that may outperform pure ODE-based or pure data driven models.

The remainder of the article is as follows. In Section II, we briefly review the full velocity difference model and provide a brief description of artificial neural networks. In Section III, we introduce four artificial neural network models, and show it is possible to design

[1]Fangyu Wu is with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Berkeley, CA 94709

[2]Daniel B. Work is with the Department of Civil and Environmental Engineering and the Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN 37212

This material is based upon work supported by the National Science Foundation under Grant No. CNS-1446702.

978-1-7281-0323-5/18/$31.00 ©2018 IEEE

ANNs that are inspired by the structure of the full velocity difference model. Then in Section IV we detail a set of numerical experiments that compare the different models and highlight the potential benefits of ANNs that are structured to consider car-following features. Finally, Section V provides some potential new directions based on these preliminary experiments.

II. Background

In this section we briefly review the full velocity difference model and a generic artificial neural network, the multilayer perceptron.

A. Full velocity difference model

The task of modelling car following behaviors at the microscopic level (i.e., at the level of individual vehicles) via ordinary differential equations requires the modeler to specify an acceleration function $a(t)$ in order to define the following system:

$$\frac{dx(t)}{dt} = v(t),$$
$$\frac{dv(t)}{dt} = a(t),$$

where $x(t)$ is the vehicle position and $v(t)$ is the vehicle velocity.

Many important analytic models have been proposed since 1950s, such as *Gipps' model* [4] and the *intelligent driver model* (IDM) [6]. One important family of car following models is the *optimal velocity model* (OVM), which was first proposed by Bando et al. [5]. In the OVM, the acceleration function, denoted a_{ovm}, is a function of the spacing $\Delta x(t)$ and the vehicle's own velocity $v(t)$:

$$a_{ovm}(\Delta x(t), v(t)) = k[V(\Delta x) - v(t)]. \quad (1)$$

In (1), k is a parameter and $V(\cdot)$ is called the *optimal velocity function* that denotes the desired speed of the vehicle when the spacing is Δx. It usually takes the form of

$$V(\Delta x(t)) = p_1 + p_2 \tanh(p_3 \Delta x(t) + p_4), \quad (2)$$

where p_1, p_2, p_3 and p_4 are parameters.

Note that the OVM in (1) does not account for the effects of the relative velocity Δv between the vehicle and its leader. This makes it difficult to explain the acceleration behaviors of a single vehicle [16]. To address this problem, Jiang et al. [15] proposed an improved OVM named the full velocity difference model. It extends OVM by adding another term to the acceleration function that considers the relative velocity Δv between the vehicles:

$$a_{fvdm}(\Delta x, v, \Delta v) = k[V(\Delta x) - v] + \lambda \Delta v, \quad (3)$$

where λ is a model parameter.

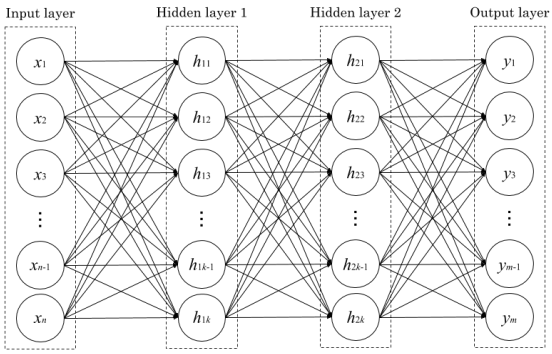


Fig. 1: A typical MLP with n inputs, two hidden layers of i and j nodes, and m outputs.

B. Artificial neural networks

Recently, there has been an interest to construct car following models from ANN models. For example, in [11], a *recurrent neural network* (RNN) is used to describe trajectories in the *Next Generation Simulation* (NGSIM) program [17] and is found to be comparable to the IDM. In this work we consider a generic form of an ANN, which is the multilayer perceptron. The MLP is composed of one or more hidden layers, and each layer is composed of a set of one or more nodes (activation functions). In each node, inputs to the layer are transformed into a scalar output from the node. Considering the outputs from other nodes in the same layer, the collective outputs are then used as an input to nodes in the next layer.

A typical MLP is graphically illustrated in Figure 1. As shown in the figure, the MLP has n inputs and m outputs; in the two hidden layers of i and j nodes, every node is fully connected to all other nodes in its neighboring layers. In general, n, i, j, and m can be any positive integers, and the model can take arbitrary number of hidden layers.

Three typical activation functions used in multilayer perceptrons include the *i*) linear activation, the *ii*) sigmoid activation, and the *iii*) hyperbolic tangent activation. The linear activation function, denoted as l, combines a vector of inputs $\boldsymbol{x}_{in}$ linearly with weights $\boldsymbol{w}$ and a scalar bias b, producing a scalar output x_{out} as

follows:

$$x_{out} = l(\boldsymbol{x}_{in}) = \boldsymbol{w}^\top \boldsymbol{x}_{in} + b. \tag{4}$$

The sigmoid activation function, denoted by σ, also maps the input vector $\boldsymbol{x}_{in}$ to a scalar output x_{out} via the parameters $\boldsymbol{w}$ and b, via the following equation:

$$x_{out} = \sigma(\boldsymbol{x}_{in}) = \frac{1}{1 + e^{-(\boldsymbol{w}^\top \boldsymbol{x}_{in} + b)}}. \tag{5}$$

Finally, the hyperbolic tangent activation, denoted τ, transforms the input vector to a scalar output similarly to (4), (5), but using the following function:

$$x_{out} = \tau(\boldsymbol{x}_{in}) = \tanh(\boldsymbol{w}^\top \boldsymbol{x}_{in} + b). \tag{6}$$

III. Proposed ANN based car following models

In this section, we first show that it is possible to write the FVDM car following model as a neural network in such a way that the ANN is mathematically equivalent to the FVDM. Then we consider 4 ANNs. The first ANN inspired by the FVDM and is able to reproduce exactly the FVDM under the correct weights in the activation functions. The second ANN has the same structure as the first ANN but the activation functions are changed so that it cannot exactly reproduce the FVDM. The third ANN is a generic MLP with only a single wide hidden layer. The final ANN is a generic MLP with has three narrow hidden layers. The models are presented in more detail below.

A. Full velocity difference model as an ANN

First we make the direct connection that it is possible to view the FVDM acceleration function as an ANN with a carefully chosen structure and set of activation functions.

The connection is straightforward. First we rewrite (3) in the following form:

$$a_{fvdm}(\Delta x, v, \Delta v) = kV(\Delta x) - kv + \lambda \Delta v, \tag{7}$$

which is easy to recognize as a mapping of three inputs, $V(\Delta x), v, \Delta v$ through a linear activation function (4). Next, we note that the input $V(\Delta x)$ used in (7) is itself an output of a scalar input Δx mapped through a tangent activation function (6). As a consequence, it is possible to construct an ANN architecture as shown in Figure 2, which exactly reproduces the FVDM acceleration function.

We will use this observation to guide the design of additional ANNs as delineated in the following section.

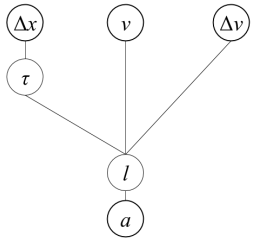


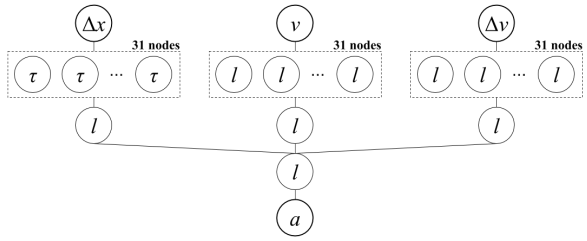
Fig. 2: The FVDM acceleration function can be viewed as an artificial neural network with three network inputs and a combination of linear and hyperbolic tangent activation functions.

B. Proposed ANN models for car following

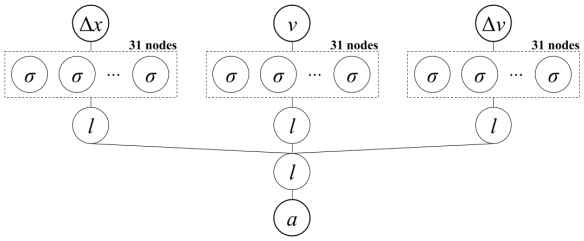
In this section we introduce four ANN models by choosing a combination of architectures and activation functions to understand the importance of the structure of the model on the performance of the ANN. For each ANN based model, we assume the input space to be a three dimensional vector composed of the spacing Δx, the vehicle velocity v, and the relative velocity Δv between the vehicle in front and the current vehicle. The output of each ANN model is the current acceleration of the vehicle, a. As a result, each ANN should be viewed as a data driven acceleration model that can be used to simulate traffic when combined with a standard numerical ODE solver.

1) ANN Model 1. Branched tanh-linear network: The main idea of the first model is to propose an architecture for the ANN in which is more general than the FVDM but is able to exactly recover the FVDM as a special case. Precisely, we consider an ANN with an architecture depicted in Figure 3a.

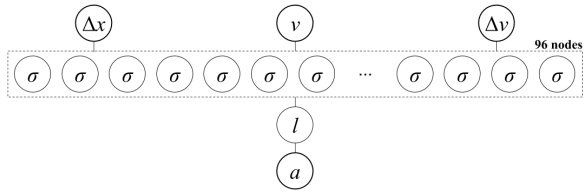
We briefly comment on the structure of the ANN architecture shown in Figure 3a. First, note the input to the ANN is a three dimensional vector of spacing, velocity, and relative velocity. On the top layer of the network, the spacing is the only input to a bank of 31 tangent activation functions, each one described by (6). Similarly, the velocity is the only input into a bank of 31 linear activation functions, and the relative velocity is the only input into its own bank of 31 linear activation functions. In the next layer, the output of the bank of 31 tangent activation functions operating on the spacing is combined via a linear activation function. Similarly, the outputs of the 31 linear activation functions operating on the velocity and the 31 linear activation functions operating on the relative velocity are linearly combined via separate linear activation functions operating on layer 2. The output of the three linear activation functions



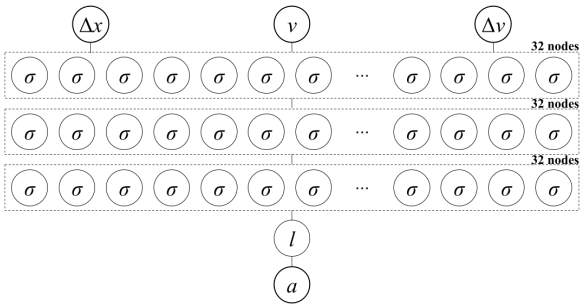
(a) Architecture of ANN Model 1. Branched tanh-linear network.



(b) Architecture of ANN Model 2. Branched sigmoid network.



(c) Architecture of ANN model 3. Flat wide network.



(d) Architecture of ANN model 4. Deep stacked network.

Fig. 3: Network architectures of the 4 ANNs. The letters inside the nodes denote the following activation functions: l is the linear activation; σ is the sigmoid activation; and τ is the hyperbolic tangent activation. As a control measure, the total number of hidden nodes in the four models is 96.

in layer 2 are combined by another linear activation function in layer 3 to output a final acceleration value.

The choice of the architecture was inspired by the FVDM model in the sense that the FVDM architecture in Figure 2 can be recovered by the network in Figure 3a by proper choice of the weights in the various activation functions, and in particular by setting many of the parameters in the linear activation function to zero.

2) ANN model 2. Branched sigmoid network: In practice we usually only have a partial understanding the system, so we propose a second, more generic, ANN model with an architecture shown in Figure 3b. The model is the same as Model 1 except that all activation functions in the first layer are changed to sigmoid activations. This choice is due to the fact that the use of sigmoid activation functions are commonly used to construct MLP models when no additional information about the activation functions are known.

3) ANN model 3. Flat wide network: For comparison, we also design two generic MLP models. As seen in [8]–[11], commonly used MLP models can be classified into two broad categories: *i*) a shallow network with a single wide hidden layer, and *ii*) a deep network with multiple narrow hidden layers.

Under the first category, we design an MLP model, denoted Model 3, with an architecture illustrated in Figure 3c. The model takes Δx, v, and Δv as inputs, feeds the vector of inputs to every node in the single 96-node sigmoid activated hidden layer, and finally linearly combines the outputs of those 96 hidden nodes to produce the acceleration output a.

4) ANN model 4. Deep stacked network: Finally under the second category of generic MLP models, we propose an additional benchmark, denoted Model 4, with an architecture as shown in Figure 3d. The model takes Δx, v, and Δv as inputs, feeds them to every node in the first 32-node sigmoid activated hidden layer, then passes the outputs of the first hidden layer through the second and the third 32-node hidden layers, and finally linearly combines the outputs in the last hidden layer to produce an produce the acceleration output a.

5) Motivation for network architectures: The underlying motivation for the proposal of the four models above is to test, given the same 96 hidden nodes and the same training procedure, which architecture demonstrates the most predictive power. The number 96 is chosen to be small enough to allow fast computation, but large enough to retain the general expressivity of a neural network model.

IV. Numerical Experiments

In this section, we present a series of numerical experiments to test the performance of the various ANN models, and in particular to illustrate the importance of the overall ANN architecture in the quality of the models that result. We describe the experiment design, the training and testing performance of the models, and discuss the main results.

A. Experiment design

To evaluate the effectiveness of the ANN models, we train the four ANN models described in Section III-B with a trajectory dataset obtained assuming the FVDM as the underlying base model. We note that by using the FVDM model to synthetically generate the data, we expect Model 1 to perform the best, precisely because it is theoretically possible to exactly replicate the FVDM in the model if the correct parameters are learned from the data. In contrast, the remaining models are not able to exactly replicate the FVDM, but as will be shown in this section, their performance varies significantly depending on the architecture. In a more realistic real world setting, this highlights the benefits of integrating any known structure into the architecture of the ANN to improve performance of the model.

We briefly describe the loss function used to train the models. Let the true acceleration function used to generate the synthetic data be denoted as a^*, and let the observed trajectory data from the vehicle be denoted as $\{x^*(t)|t \in [t_0, t_1]\}$. Based on Treiber and Kesting's [18] local maximum-likelihood calibration method, we set the training objective to be minimizing the following mean square loss function:

$$\min_{\boldsymbol{p}} \quad L(\boldsymbol{p}) = \frac{1}{N_t} \sum_{t=t_0}^{t_1} [a^*(t) - \hat{a}(t|\boldsymbol{p})]^2, \qquad (8)$$

where N_t is the total number of sample points in the training dataset; for a ODE solver with a constant time step Δt, $N_t = \frac{t_1 - t_0}{\Delta t}$. The term $\hat{a}(t|\boldsymbol{p})$ is the proxy model with parameters $\boldsymbol{p}$. In this experiment, we have $a^* \equiv a_{fvdm}$ and $\hat{a}$ is the acceleration produced by one of the four ANN models.

Inspired by [19], [20], we simulate 10 identical 5 m long vehicles in a single-lane 250-meter-long circular track using a FVDM to obtain the training data. Specifically, with slight modification from [16], we assume each vehicle drives according to the following FVDM:

$$\begin{aligned} a_{fvdm} =& 3.2431 \tanh(0.13\Delta x - 2.22) \\ & - 0.41v + 0.2\Delta v + 2.7675. \end{aligned} \qquad (9)$$

Loss	Model 1	Model 2	Model 3	Model 4
Train (m/s^2)	4.80×10^{-3}	5.94×10^{-3}	1.27×10^{-2}	1.15×10^{-2}
Test (m/s^2)	2.39×10^{-2}	2.84×10^{-1}	1.11×10^{1}	5.12×10^{0}

TABLE I: Training and testing loss Models 1 through 4 after 100 epochs.

Under this set of parameters, the FVDM is able to generate stop and go traffic jams, which is an important emergent phenomenon which we would ideally like to recover from the ANN based models. Each vehicle is initially at zero velocity and approximately uniformly spaced. Using an explicit Runge-Kutta method [21], [22], we simulate the system for 500 seconds with a time step of $\Delta t = 0.1$ second. Taking one measurement per second per vehicle, we obtain a training dataset of 50,000 sample points, each containing the velocity, relative velocity, spacing, and acceleration of a vehicle.

To train the models, we initialize the ANN parameters $\boldsymbol{p}$ with Xazvier's method [23] and solve the optimization problem (8) using the Adam method [24] with a learning rate of 0.0001 for 100 iterations.

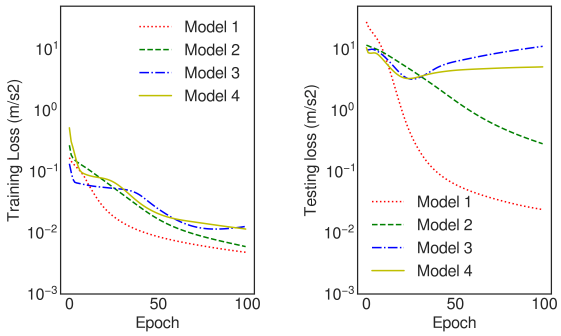
To understand the potential of the ANN models to generalize well beyond the data contained in the training dataset, we generate a test dataset by uniformly sampling 2,000 points in the following subdomain of FVDM:

$$\begin{aligned} 1m \leq & \Delta x \leq 50m, \\ 0.25m/s \leq & v \leq 20m/s, \\ -24m/s \leq & \Delta v \leq 25m/s. \end{aligned}$$

The predicted acceleration from each of the ANNs is then compared to the acceleration of the FVDM.

B. Training and testing losses

The training errors and the testing errors of the four models are displayed epoch by epoch in Figure 4. An epoch is a complete pass through the training dataset. The final losses of each model in training and testing are presented in Table I. Across the epochs, we observe that the four models achieve comparable results in training. However, Model 1 and Model 2 are significantly better than Model 3 and Model 4 on the test dataset. After approximately epoch 25, both of the generic MLP models (Models 3 and 4) start to overfit, while the structured ANN models (Models 1 and 2) continue to improve. Even though Model 2 does not use the same activation function as the FVDM, the fact that a portion of the first hidden layer is used to learn the relationship between each input individually helps it outperform the models that are more general in their structure.



(a) Training loss. (b) Testing loss.

Fig. 4: Training and testing errors of Models 1 through 4 over 100 epochs.

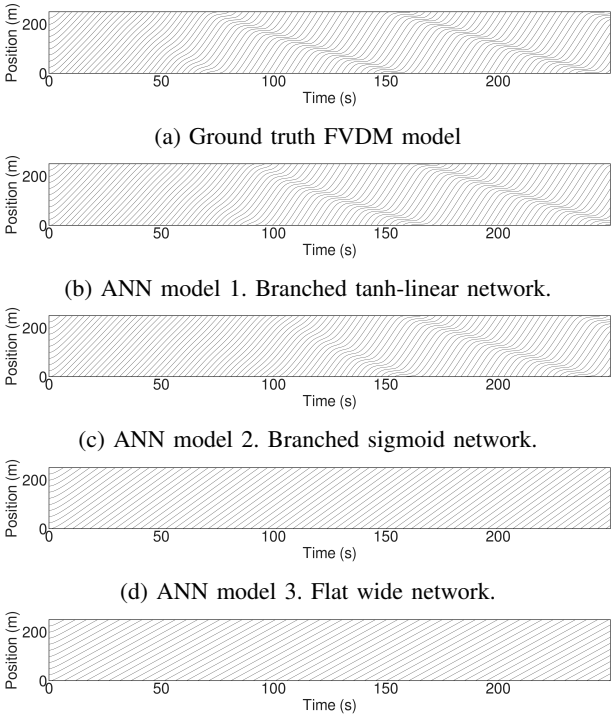
C. Checking the emergent properties of the trained ANN models

Under the same initial condition as was used to generate the training data, we simulate each of the trained ANNs and compare the resulting trajectories in Figure 5. In the plot, we observe the highest performing model (i.e., the one that captures the stop and go wave pattern most accuately), is Model 1, which was known a-priori to be able to reproduce the FVDM based on its structure and choice of activation functions. More importantly, the result indicates that Model 2 is also sufficiently accurate to be able to model the emergence of stop-and-go waves which are present in trajectories simulated using the FVDM, while the more generic Models 3 and 4 are not able to reproduce stop and go waves.

Finally, the acceleration response of the four models as well as the ground truth FVDM are shown in Figure 6. In the plot, the corresponding acceleration a is visualized for Δx, v, and Δv within the sampling range of the test data. The magnitude of acceleration is represented in a color space defined in the color bar on the right with green indicating high acceleration, and red denoting high deceleration. From the figure, we observe that acceleration responses of Model 1 and 2 are much closer to the true acceleration response of FVDM than those of the generic MLP Models 3 and 4. This also indicates that the structure of the ANN plays a role in helping the model generalize beyond the data used to train the model.

D. Discussion

The main difference between the first two models and the last two models is that model 3 and 4 are significantly less generalizable than model 1 and 2.



(a) Ground truth FVDM model

(b) ANN model 1. Branched tanh-linear network.

(c) ANN model 2. Branched sigmoid network.

(d) ANN model 3. Flat wide network.

(e) ANN model 4. Deep stacked network.

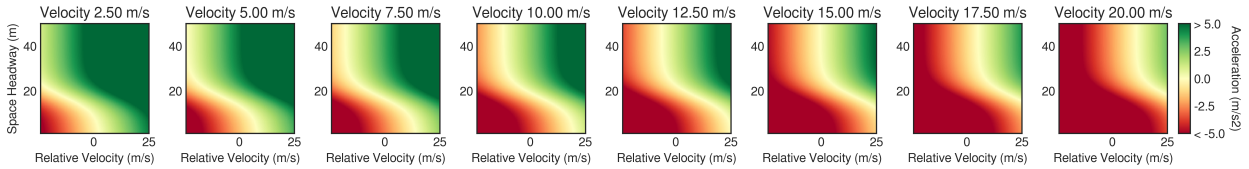
Fig. 5: Simulated trajectories of of the FVDM used to generate training data, and trajectories from the resulting trained models Model 1 through MOdel 4 using the same initial conditions. For Models 3 and 4, we choose the trained models with the lowest testing errors, which are at epoch 27 for Model 3 and epoch 25 for Model 4.

Without good generalizability, a small error at time t will tend to lead to larger error at time $t+1$. Over time, the errors from model 3 and 4 accumulate and become too significant for the models to recover to the right track, a phenomenon called *distributional drift*. On the contrary, model 1 and 2 have good generalizability, which allows them to automatically correct small deviations in the predictions and hence never drift too far from the true trajectory. Therefore, the results from model 1 and 2 are more accurate than those from model 3 and 4 in both microscopic and macroscopic levels.

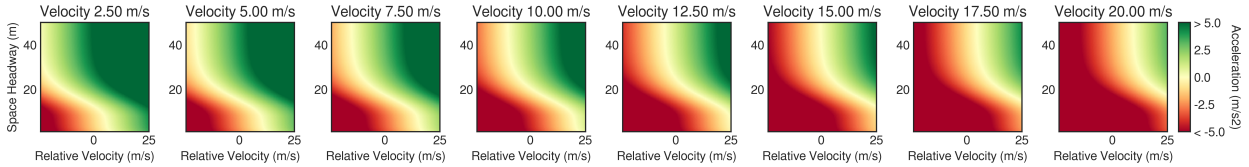
Last but not least, despite the effectiveness of model 1 and 2, exactly why model 1 and 2 are more generalizable than model 3 and 4 is still a theoretical challenge. We consider this question as one of the main research tasks for future works.

V. Conclusion

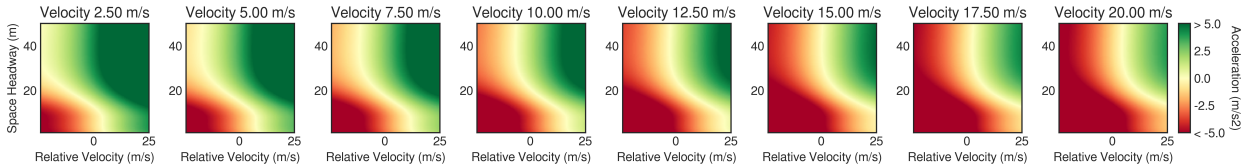
This article provided preliminary connections between classical car following models and data driven artificial



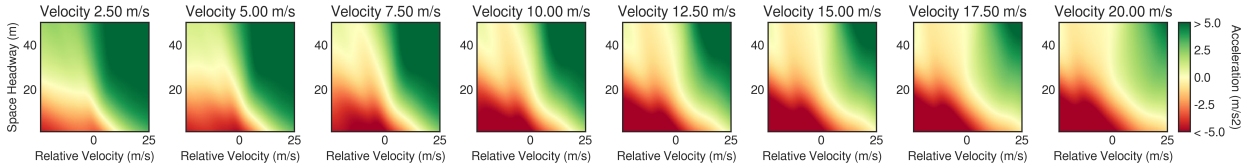
(a) Ground truth model. FVDM a_{fvdm}.



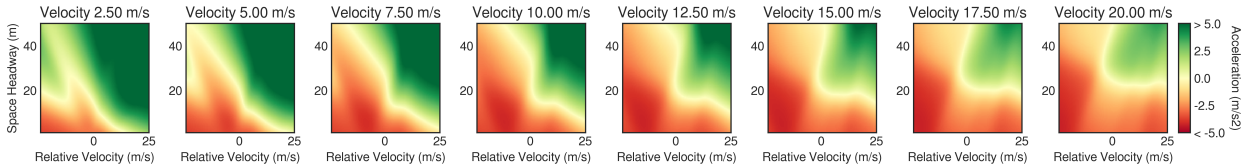
(b) ANN model 1. Branched tanh-linear network.



(c) ANN model 2. Branched sigmoid network.



(d) ANN model 3. Flat wide network.



(e) ANN model 4. Deep stacked network.

Fig. 6: Acceleration responses of the FVDM and Models 1 through 4. Note that for Models 3 and 4 we select the models with the lowest testing errors, which are at epoch 27 for Model 3and epoch 25 for Model 4.

neural networks. Inspired by the fact that it is possible to rewrite a classical car following model as an equivalent ANN, it was shown that structured ANNs have the potential to improve performance when applied to model car following dynamics. As a first step, the performance of a few candidate architectures were assessed in numerical experiments in which the underlying dynamics were known.

In our future work, we are interested in several extensions. While the numerical experiments conducted in this article are easier to analyze, a natural extension is to apply the methods on data collected from real human drivers where the true driving model is unknown to the ANN designer. We are also interested in considering a wider range of architectures for the ANNs, for example by considering more layers, more structures, and other neural networks beyond the multilayer perceptron.

REFERENCES

[1] Louis A Pipes. An operational analysis of traffic dynamics. *Journal of Applied Physics*, 24(3):274–281, 1953.

[2] Gordon Frank Newell. Nonlinear effects in the dynamics of car following. *Operations Research*, 9(2):209–229, 1961.

[3] Nicola Bellomo and Christian Dogbe. On the modeling of traffic and crowds: A survey of models, speculations, and perspectives. *SIAM review*, 53(3):409–463, 2011.

[4] Peter G Gipps. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, 15(2):105–111, 1981.

[5] Masako Bando, Katsuya Hasebe, Akihiro Nakayama, Akihiro Shibata, and Yuki Sugiyama. Dynamical model of traffic congestion and numerical simulation. *Physical review E*, 51(2):1035, 1995.

[6] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000.

[7] Cathy Wu, Aboudy Kreidieh, Kanaad Parvate, Eugene Vinitsky, and Alexandre M Bayen. Flow: Architecture and benchmarking for reinforcement learning in traffic control. *arXiv preprint arXiv:1710.05465*, 2017.

[8] Jia Hongfei, Juan Zhicai, and Ni Anning. Develop a car-following model using data collected by" five-wheel system". In *Proceedings of the IEEE Conference on Intelligent Transportation Systems*, volume 1, pages 346–351. IEEE, 2003.

[9] Linsen Chong, Montasir Abbas, and Alejandra Medina. Simulation of driver behavior with agent-based back-propagation neural network. *Transportation Research Record: Journal of the Transportation Research Board*, (2249):44–51, 2011.

[10] Alireza Khodayari, Ali Ghaffari, Reza Kazemi, and Reinhard Braunstingl. A modified car-following model based on a neural network model of the human driver effects. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 42(6):1440–1449, 2012.

[11] Mofan Zhou, Xiaobo Qu, and Xiaopeng Li. A recurrent neural network based microscopic car following model to predict traffic oscillation. *Transportation Research Part C: Emerging Technologies*, 84:245–264, 2017.

[12] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[13] Kunihiko Fukushima. Neural network model for a mechanism of pattern recognition unaffected by shift in position-neocognitron. *IEICE Technical Report, A*, 62(10):658–665, 1979.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[15] Rui Jiang, Qingsong Wu, and Zuojin Zhu. Full velocity difference model for a car-following theory. *Physical Review E*, 64(1):017101, 2001.

[16] Tian Jun-Fang, Jia Bin, and Li Xing-Gang. A new car following model: comprehensive optimal velocity model. *Communications in Theoretical Physics*, 55(6):1119, 2011.

[17] Vassili Alexiadis, James Colyar, John Halkias, Rob Hranac, and Gene McHale. The next generation simulation program. *Institute of Transportation Engineers. ITE Journal*, 74(8):22, 2004.

[18] Martin Treiber and Arne Kesting. Microscopic calibration and validation of car-following models–a systematic approach. *Procedia-Social and Behavioral Sciences*, 80:922–939, 2013.

[19] Raphael E Stern, Shumo Cui, Maria Laura Delle Monache, Rahul Bhadani, Matt Bunting, Miles Churchill, Nathaniel Hamilton, Hannah Pohlmann, Fangyu Wu, Benedetto Piccoli, et al. Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments. *Transportation Research Part C: Emerging Technologies*, pages 205–221, 2018.

[20] Fangyu Wu, Raphael Stern, Shumo Cui, Maria Laura Delle Monache, Rahul Bhadani, Matt Bunting, Miles Churchill, Nathaniel Hamilton, Benedetto Piccoli, Benjamin Seibold, et al. Tracking vehicle trajectories and fuel rates in oscillatory traffic. 2017.

[21] John R Dormand and Peter J Prince. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980.

[22] Lawrence F Shampine. Some practical runge-kutta formulas. *Mathematics of Computation*, 46(173):135–150, 1986.

[23] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.