Short-term traffic prediction using physics-aware neural networks

Transportation Research Part C 142 (2022) 103772

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Short-term traffic prediction using physics-aware neural networks[☆]

Mike Pereira [a,b,*], Annika Lang [a], Balázs Kulcsár [b]

[a] *Department of Mathematical Sciences, Chalmers University of Technology & University of Gothenburg, S–412 96 Göteborg, Sweden*
[b] *Department of Electrical Engineering, Chalmers University of Technology, S–412 96 Göteborg, Sweden*

ARTICLE INFO

Keywords:
Traffic reaction model
Macroscopic traffic flow model
Traffic prediction
Recurrent neural networks
LSTM

ABSTRACT

In this work, we propose an algorithm performing short-term predictions of the flow and speed of vehicles on a stretch of road, using past measurements of these quantities. This algorithm is based on a physics-aware recurrent neural network. A discretization of a macroscopic traffic flow model (using the so-called Traffic Reaction Model) is embedded in the architecture of the network and yields traffic state estimations and predictions for the flow and speed of vehicles, which are physically-constrained by the macroscopic traffic flow model and based on estimated and predicted space–time dependent traffic parameters. These parameters are themselves obtained using a succession of LSTM recurrent neural networks. The algorithm is tested on raw flow measurements obtained from loop detectors.

1. Introduction

Using neural networks to produce short-term traffic predictions is a subject of growing interest, especially since advances in transportation systems increased substantially the availability of large amount of traffic data from various sources (Zhu et al., 2018; Zhang et al., 2011). We refer the interested reader to Lee et al. (2021) and Do et al. (2019) for surveys of neural networks-based models that have been proposed for short-term traffic predictions. These methods have proven to be particularly fit to tackle the challenges associated with traffic prediction such as the modeling of spatio-temporal dependencies in traffic data or the influence of external factors (e.g. weather and road conditions, unpredictable traffic events) (Vlahogianni et al., 2014).

Particular architectures have been proposed to model the temporal dependencies present in traffic data (see Yin et al., 2020 for a comprehensive survey), which are due for instance to the fact that these data often consist of time series of measurements of traffic-related quantities (such as the occupancy, speed and flow) made by sensors on a road. Consequently, algorithms taking sequential data as input have been proposed to reflect for traffic predictions: the input then consists of a sequence of (possibly multivariate) variables corresponding to measurements at some times $t_{-(N_p-1)} \leq \cdots \leq t_{-1} \leq t_0$ and generates predictions of some quantity of interest at times $t_1 \leq \cdots \leq t_{N_f}$ (where $t_1 \geq t_0$). Examples of such algorithms include time-delayed neural networks (e.g. Zhong et al., 2005; Abhishek and Misra, 2016; Lingras and Mountford, 2001), (graph) convolutional neural networks (e.g. Pan et al., 2019; Yu et al., 2017) and recurrent neural networks (RNNs) (e.g. Xu et al., 2017; Ma et al., 2015; Wondimagegnehu and Alemu, 2017), which will be of particular interest in this work.

☆ ACKNOWLEDGMENT. The authors thank G. Szederkenyi and Gy. Liptak for the helpful discussions about the TRM, as well as P. Boyraz Baykas for her inspiring inputs. The authors thank for their financial support the Chalmers AI Research Centre (CHAIR) (under the projects STONE, RITE, and SCNN) and the Transport Area of Advance of the Chalmers University of Technology , Sweden (under the projects STONE and IRIS). This work has been partially supported and funded by OPNET (Swedish Energy Agency, 46365-1), the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, and the Swedish Research Council, Sweden (project nr. 2020-04170). The traffic data are provided by, and used with the permission of Rijkswaterstaat – Centre for Transport and Navigation (Netherlands)

* Corresponding author.
E-mail addresses: mike.pereira@chalmers.se (M. Pereira), annika.lang@chalmers.se (A. Lang), kulcsar@chalmers.se (B. Kulcsár).

https://doi.org/10.1016/j.trc.2022.103772
Received 24 September 2021; Received in revised form 30 May 2022; Accepted 21 June 2022
Available online 11 July 2022

However, regardless of their great prediction performances, a caveat of these approaches is their black-box nature: indeed, due to their complex nature and high number of parameters, these models lack interpretability as it is hard to understand how (and if) these algorithms manage to produce reliable traffic predictions (Vlahogianni et al., 2014; Do et al., 2019). Also, these algorithms (and their performance) depend on a number of hyperparameters (such as the number of hidden layers, the number of nodes, …) that must be tuned by the practitioner in order to achieve the best performances for a given dataset.

Bearing this in mind, we propose in this work a "grey-box" RNN architecture which aims at providing short-term flow and speed predictions based on past flow and speed measurements along a stretch of road. The originality of this architecture is that it combines a conventional RNN with a discretization of a macroscopic traffic flow model in an attempt to yield interpretable predictions. Indeed, the space–time varying parameters of the traffic model are seen as a set of unobserved time series which are learned from the measurements. In a first step, a conventional RNN is used to extract these time series of parameters from the measurements and predict their evolution over a short period of time. The output of this first algorithm is then passed to the traffic model to yield both a traffic state estimation and prediction along the whole road. In short, the algorithm is able to both complete and predict the speed and flow variations along a road.

Note that the traffic predictions obtained with the proposed approach come directly from the (discretized) macroscopic traffic model, and therefore are bound to follow the associated physical model. The idea of constraining neural networks with physical models is receiving much attention in the context of solving forward and inverse problems for partial differential equations (PDEs). Three main approaches have been proposed in this context:

- the *physics-informed* approach, which consists in using usual (deep) neural architectures to perform the prediction task, while augmenting the cost function used to train the neural network with terms that describe the PDE and its possible boundary or initial conditions (e.g. Raissi et al., 2019; Xu and Darve, 2019),
- the *physics-regularized* approach, which models the data by a Gaussian process (GP) and uses a generative component to constrain the GP to the PDE model (Wang et al., 2022; Yuan et al., 2021b),
- the *physics-aware* approach, which consists in explicitly embedding the PDE within the neural architectures used to perform the prediction task, by dedicating some parts of the network to the approximation of differential operators (e.g. Long et al., 2018, 2019) or to the numerical approximation of the PDE itself (e.g. Berg and Nyström, 2017; Dal Santo et al., 2020; Pakravan et al., 2021).

For traffic-related problems, most of the focus has been put on the *physics-informed* approach (Rempe et al., 2021; Huang and Agarwal, 2020; Liu et al., 2021; Shi et al., 2021). For instance, Huang and Agarwal (2020) proposed a physics-informed method for density estimation, which was then extended by Liu et al. (2021) to support short-term prediction (assuming that the traffic characteristics are maintained during the prediction horizon) and noise reduction. But since the knowledge of the PDE governing the traffic flow is required, they either test their method on data simulated from a known PDE, or perform a preliminary model identification step. Shi et al. (2021) then proposed to include the parameters of the traffic PDE model into the learning process, but limited their study to density estimation.

The physics-regularized approach, on the other hand, extends the capabilities of the physics-informed approach by allowing to exploit incomplete physics knowledge, i.e. allowing to work with partially unspecified PDE models (Wang et al., 2022). The posterior distribution of the GP used to model the data in this setting is regularized through a prior distribution derived from the physical PDE model. This Bayesian setting makes the physics-regularized approach robust to noise in the data and a great tool for uncertainty quantification, as put in evidence by Wang et al. (2022), and more specifically for traffic data by Yuan et al. (2021a,b). These studies however only focused on traffic state estimation, and not on short-term predictions. Besides, the traffic models they used were assuming that the models parameters were constant, and not subject to possible changes in time (or space).

Contrary to these works, our method stems from the physics-aware approach: indeed, much like the method Pakravan et al. (2021) proposed to solve inverse PDE problems, our method consists of using a neural network to learn the space–time varying parameters of the traffic PDE, and then inputting these parameters into a numerical scheme designed to approximate the PDE. But since we also learn how to predict the evolution of these parameters, our method can be seen as an extension of their approach for prediction purposes. To the best of our knowledge, our work presents the first use of physics-aware neural networks for traffic state estimation and prediction and in the presence of real (traffic) data.

The advantages of using a physics-aware approach are twofold. First, the traffic state estimates and predictions both directly arise from the traffic model: we learn and predict the evolution of the model parameters, and then simply run the model under these parameters to get the estimates and predictions. This ensures in particular that key properties of the traffic models (such as non-negativeness, conservation, and boundedness) are always respected. In contrast, the estimates and predictions from physics-informed methods are made by neural architectures that have no connection to the physics of the problem: the connection to the physics is only made in the training phase, when the parameters of the neural network are tuned with a particular cost function. Second, since we explicitly task our neural architecture with learning how to predict the space–time evolution of the parameters from past observations, the resulting traffic state predictions can be interpreted as arising from a model with varying traffic characteristics.

Another original aspect of this research is the use of a particular discretization scheme for the macroscopic traffic model, the so-called Traffic Reaction Model (Lipták et al., 2021), in the context of traffic state predictions. This scheme models flow dynamics along a discretized road as a chain of chemical reactions happening between adjacent cells (now seen as compartments containing "fictional" chemical reactants). This scheme was shown to be able to reproduce complex traffic pattern observed when performing state and parameter estimation from real-world traffic data (Pereira et al., 2021). It also allows to assimilate the space–time varying parameters of the macroscopic traffic model to dimensionless and bounded parameters that are interpreted as the reaction rates of the chemical reactions mentioned above.

In addition to the interpretation that can be given to the parameters of our architecture and the origin of the predictions it produces, most of its hyperparameters are set by the configuration of the detectors along the road of interest, hence removing the burden of tuning them. Moreover, our approach is particularly suited to high resolution data (i.e. data composed of measurements made with high frequency, for instance every minute or less). Such datasets are known to be problematic in prediction applications, due to the high level of noise they might have (Li and Rose, 2011). In such cases, noise reduction techniques, such as smoothing or wavelet algorithms (e.g. Jiang and Adeli, 2004), or aggregation of the measurements are usually applied to the data before proceeding to the predictions. But there is no consensus on which approach to select (and with which parameters), and besides, these techniques have be shown to impact and distort the properties of time series (Vlahogianni and Karlaftis, 2011). Our approach on the other hand naturally incorporates a smoothing of the measurements inherited from the discretized macroscopic traffic model, and is therefore physically justified.

In summary, we propose a physics-aware algorithm for traffic estimation and short-term traffic predictions. Our algorithm uses a (discretization of) macroscopic traffic model to produce traffic state estimations and predictions, even on points of the road where no measurements are taken. As the result, the architecture, the parameters and the outputs of our algorithm all find physical motivations and interpretations.

The outline of this paper is as follows. In Section 2, we present the macroscopic traffic model and its discretization used in out neural network architecture. In Section 3, we provide some reminders on RNNs and present the architecture. Finally, in Section 4, we present numerical experiments based on a dataset of flow and speed measurements taken on a Dutch motorway.

We use the following notations in the remainder of the text. We denote by $\boldsymbol{a} \odot \boldsymbol{b}$ the entry-wise product between two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ of same size, and by $\boldsymbol{a} \oslash \boldsymbol{b}$ their entry-wise ratio. We start the indexation of vectors at 0, and given a vector $\boldsymbol{a}$ of size N, and a set of p indices $I = \{i_1, \dots, i_p\} \subset \{0, \dots N-1\}$, we denote by $\boldsymbol{a}_{[I]} \in \mathbb{R}^p$ the vector formed by the entries of $\boldsymbol{a}$ at the indices I. In particular, $\boldsymbol{a}_{[0:N-2]}$ (resp. $\boldsymbol{a}_{[1:N-1]}$) is the vector containing the first (resp. last) $N-1$ entries of $\boldsymbol{a}$, and for $i \in \{0, \dots, N-1\}$, $\boldsymbol{a}_{[i]}$ is simply the ith entry of $\boldsymbol{a}$. Finally, we denote by $\|\boldsymbol{a}\|$ the usual Euclidean norm of a vector $\boldsymbol{a}$.

2. Traffic models and data

2.1. First order macroscopic traffic flow model

First order macroscopic traffic flow models link the evolution of two quantities describing the flow and state of the traffic at any point in time and space: the flow of vehicles f and the density of vehicles ρ. Both quantities can in particular be coupled through a conservation law expressing a balance between the numbers of vehicles entering and exiting any section of the road and the number of vehicles present in this same section. Such balance laws are common when studying for instance the flow of incompressible fluids and take the following form:

$$\partial_t \rho(t,x) + \partial_x f(t,x) = q(t,x), \quad t \geq 0, \quad x \in \mathbb{R}, \tag{1}$$

where q is a function acting as a source term and describing for instance on- and off-ramps along the road.

One of the most popular macroscopic traffic flow model, the so-called Lighthill–Whitham–Richards (LWR) model (Richards, 1956; Lighthill and Whitham, 1955), takes the additional assumption that the flow of vehicles can be expressed as a function (called flux function) of the density of vehicles. A popular choice of flux function is the so-called Greenshields flux (Greenshields et al., 1935), for which f takes the form

$$f(t,x) = 4 f_m(t,x) \times \frac{\rho(t,x)}{\rho_m}\left(1 - \frac{\rho(t,x)}{\rho_m}\right), \quad t \geq 0, \quad x \in \mathbb{R}, \quad \rho \in [0, \rho_m], \tag{2}$$

where $\rho_m > 0$ denotes the maximal value the density can take (i.e. the density value corresponding to a bumper-to-bumper traffic) and $f_m \geq 0$ denotes the (possibly space–time-dependent) maximal flow of vehicles. This choice follows naturally when enforcing two straightforward and physically meaningful conditions that a flux function should satisfy, namely that the flow should be 0 when the road is empty (i.e. $\rho = 0$) or at full capacity (i.e. $\rho = \rho_m$).

Note that considering a space–time-dependent maximal flow f_m allows to model for instance situations where the traffic flow is impacted by road conditions or policies that may vary in time or space. For any given bounded and integrable initial condition $\rho_0 \equiv \rho(0,\cdot)$, and any $T > 0$, the existence and uniqueness of (entropy) solutions of the resulting PDE in $C([0,T), L^1(\mathbb{R}) \cap L^\infty(\mathbb{R}))$ has been proven in the general case, where the maximal speed is taken to be a function $(t,x) \mapsto f_m(t,x)$ that is bounded and Lipschitz-continuous (Karlsen and Towers, 2004; Chen and Karlsen, 2005).

2.2. Solving numerically the PDE: The traffic reaction model

In general, no closed-form solution of PDE (1) is available, and therefore, numerical methods must be used to approximate it. In particular, finite volume schemes have been widely used to compute solutions to (hyperbolic) PDEs of the form (1) since these solutions can develop discontinuities in finite time but remain locally integrable (LeVeque, 2002). For reasons which will be clarified in subsequent sections of this paper, we focus on a particular scheme proposed by Lipták et al. (2021) and called Traffic Reaction

Model (TRM). This scheme can be seen as a particular instance of the finite volume schemes investigated by Chainais-Hillairet and Champier (2001).

Let us once again consider PDE (1) on $(\mathbb{R}_+ \times \mathbb{R})$ and the following space–time discretization: in time we consider equidistant time steps $t_n = n\Delta t$ (for some step size $\Delta t > 0$ and $n \in \mathbb{N}$) and in space we consider equispaced cells of size $\Delta x > 0$ with centroids $x_j = j\Delta x$ (for $j \in \mathbb{Z}$). We denote by $x_{j\pm 1/2} = x_j \pm \Delta x/2$ the boundary points of the jth cell. For each cell j and time step t_n, a quantity ρ_j^n is defined as follows. At $t = t_0$,

$$\rho_j^0 = \frac{1}{\Delta x}\int_{x_{j-1/2}}^{x_{j+1/2}} \rho_0(x)\,\mathrm{d}x, \quad j \in \mathbb{Z},$$

where ρ_0 denotes the initial condition associated with PDE (1), and is assumed to be known. Then, the TRM approximates the solution of PDE (1) at (t_n, x_j) by the quantity ρ_j^n obtained through the following recurrence relation

$$\rho_j^{n+1} = \rho_j^n + \rho_m \left(C_{j-1}^n F(\rho_{j-1}^n, \rho_j^n) - C_j^n F(\rho_j^n, \rho_{j+1}^n) \right) + q_j^n, \quad j \in \mathbb{Z}, \quad n \in \mathbb{N}, \tag{3}$$

where the quantities C_j^n and q_j^n are given by

$$C_j^n = \frac{4}{\rho_m \Delta x}\int_{t_n}^{t_{n+1}} f_m(t, x_{j+1/2})\,\mathrm{d}t, \quad j \in \mathbb{Z}, \quad n \in \mathbb{N},$$
$$q_j^n = \frac{1}{\Delta x}\int_{t_n}^{t_{n+1}}\int_{x_{j-1/2}}^{x_{j+1/2}} q(t,x)\,\mathrm{d}x\,\mathrm{d}t, \quad j \in \mathbb{Z}, \quad n \in \mathbb{N},$$

and F denotes the numerical flux defined by

$$F(r, r') = \frac{r}{\rho_m}\left(1 - \frac{r'}{\rho_m}\right), \quad r, r' \in [0, \rho_m].$$

Note in particular that, in order to apply this recurrence relation, the maximal flow f_m and the source term q (or at least their discretized counterparts C_j^n, q_j^n) must be known at any time and space location.

Remark 2.1. Note that the recurrence relation (3) actually stems from a stable forward Euler discretization of the system of ordinary differential equations (ODEs) satisfied by a set of functions $\{\rho_j\}_{j\in\mathbb{Z}}$ and given by

$$\dot{\rho}_j(t) = \frac{1}{\Delta x}\left(f_m(t, x_{j-1/2}) F(\rho_{j-1}, \rho_j) - f_m(t, x_{j+1/2}) F(\rho_j, \rho_{j+1}) \right) + Q_j(t), \quad t > t_0,$$

where for $j \in \mathbb{Z}$, Q_j is given by

$$Q_j(t) = \frac{1}{\Delta x}\int_{x_{j-1/2}}^{x_{j+1/2}} q(t,x)\,\mathrm{d}x, \quad t \geq 0.$$

As defined, the quantity ρ_j^n acts as an approximation of the cell average at time t_n and on the jth cell of the solution ρ of PDE (1) which is the ratio of the integral over the jth cell of the function $x \mapsto \rho(t_n, x)$ to the cell size. Assuming that f_m is bounded and Lipschitz-continuous, Chainais-Hillairet and Champier (2001) proved the convergence, to the entropy solution of the PDE, of a class of finite volume schemes that includes the TRM as the step size $\Delta t, \Delta x \to 0$, provided that these quantities satisfy a so-called Courant–Friedrichs–Lewy (CFL) condition given by

$$\frac{\Delta t}{\Delta x} < \frac{\rho_m}{8\|f_m\|_\infty}, \tag{4}$$

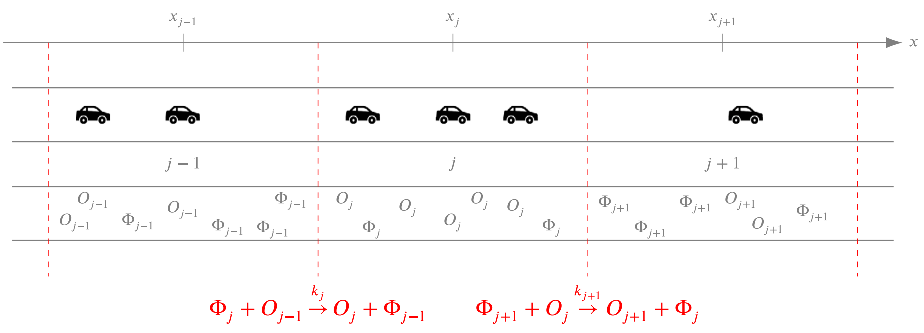
where $\|f_m\|_\infty = \sup_{t\geq 0, x\in\mathbb{R}} |f_m(t,x)|$. Note that since f_m is nonnegative, this condition imposes that the coefficients C_j^n, $j \in \mathbb{Z}$, $n \in \mathbb{N}$, satisfy

$$C_j^n < \frac{1}{2}, \quad j \in \mathbb{Z}, \quad n \in \mathbb{N}. \tag{5}$$

This condition is instrumental into proving the stability, monotonicity and ultimately convergence of the TRM scheme.

Remark 2.2. The TRM can be derived by assimilating the traffic flow dynamic (on the discretized road) to a chemical reaction network (Lipták et al., 2021; Pereira et al., 2021). More precisely, each cell is modeled as a compartment containing two (homogeneously-distributed) fictional chemical reactants: molecules of occupied space O_j and molecules of free space F_j. The density of vehicles ρ_j within a cell j is then assimilated to the concentration of occupied space O_j, and the variations in time of this density is interpreted as resulting from chemical reactions happening at the boundaries of the cell. Namely, the chemical reaction happening at the boundary between the cells j and $j+1$ "transforms" a molecule of occupied space O_j in j and a molecule of free space Φ_{j+1} in $j+1$ into a molecule of free space Φ_j in j and a molecule of occupied space O_{j+1} in $j+1$ (cf. Fig. 2.1).

The TRM then falls from writing the kinetics of these reactions using the law of mass action (Feinberg, 2019) (assuming that the reactions described above happen with some rate k_j that can be time-dependent). As for the coefficients C_j and q_j, they are seen as respectively reaction rates between compartments and external intakes or outtakes of reactants (cf. Pereira et al., 2021 for details).



x_{j-1}
x_j
x_{j+1}
x
$j-1$
j
$j+1$
O_{j-1}
Φ_{j-1}
O_j
Φ_j
Φ_{j+1}
O_{j+1}
$\Phi_j + O_{j-1} \xrightarrow{k_j} O_j + \Phi_{j-1}$
$\Phi_{j+1} + O_j \xrightarrow{k_{j+1}} O_{j+1} + \Phi_j$

Fig. 2.1. Representation of the Traffic Reaction Model interpretation of traffic flow.

In practice, the modeled road is not infinite, and only a finite number of cells is used to discretize it. Hence, boundary conditions must be supplied to define the evolution of the outer cells of the road. Indeed, assuming that the road is discretized using N_x cells (numbered from 1 to N_x), the recurrence relation (3) takes the form

$$\rho_j^{n+1} = \rho_j^n + \rho_m \left(\tilde{F}_{j-1}^n - \tilde{F}_j^n \right) + q_j^n, \quad j \in \{1, \dots, N_x\}, \tag{6}$$

where for the interior cells (cells 2 to $N_x - 1$), the quantities $\tilde{F}_k^n$ are defined as the TRM numerical fluxes:

$$\tilde{F}_k^n = C_k^n F(\rho_k^n, \rho_{k+1}^n), \quad k \in \{1, \dots, N_x - 1\}. \tag{7}$$

For the boundary cells (cells 1 and N_x), the TRM numerical flux would require the quantities $\rho_0^n, \rho_{N_x+1}^n \in [0, \rho_m]$, which are no longer defined. A natural approach would consist in considering these quantities as additional parameters of the model. Then, for instance, the numerical flux $\tilde{F}_0^n$ would be given by

$$\tilde{F}_0^n = C_0^n F(\rho_0^n, \rho_1^n) = \frac{C_0^n \rho_0^n}{\rho_m} \left(1 - \frac{\rho_1^n}{\rho_m} \right)$$

and is therefore parameterized only by the product of the independent parameters C_0^n and ρ_0^n (divided by the maximal density ρ_m). Therefore, we propose to directly consider the factor $C_0^n \rho_0^n / \rho_m \in (0, 1/2)$ as a parameter of the model, and denote it, with a slight abuse of notation, by C_0^n. Using the same approach for the numerical flux $\tilde{F}_{N_x}^n$, we end up with the following expressions:

$$\begin{aligned} \tilde{F}_0^n &= C_0^n \left(1 - \frac{\rho_1^n}{\rho_m} \right) = C_0^n F(\rho_m, \rho_1^n), \\ \tilde{F}_{N_x}^n &= C_{N_x}^n \frac{\rho_{N_x}}{\rho_m} = C_{N_x}^n F(\rho_{N_x}, 0), \end{aligned} \tag{8}$$

where $n \in \mathbb{N}_0$. Note that this setting is equivalent to considering that the road is full upstream (i.e. $\rho_0^n = \rho_m$) and empty downstream (i.e. $\rho_{N_x+1}^n = 0$). The reaction rates C_0^n and $C_{N_x}^n$ are then used to control the amount of vehicles allowed to enter or exit the road between t_n and t_{n+1}.

Remark 2.3. Multiplying the recurrence relation (6) by Δx yields an equation describing the variation of the number of vehicles inside the jth cell between t_n and t_{n+1}:

$$\rho_j^{n+1} \Delta x - \rho_j^n \Delta x = (\rho_m \Delta x) \tilde{F}_{j-1}^n - (\rho_m \Delta x) \tilde{F}_j^n + q_j^n \Delta x.$$

In this last equation, the quantity $q_j^n \Delta x$ corresponds to the net volume of vehicles added or subtracted to the cell by the source term, and the quantity $(\rho_m \Delta x) \tilde{F}_j^n$ corresponds to the volume of vehicles exiting the cell j and entering the cell $(j+1)$. Hence, dividing this last quantity by the time step Δt then gives an approximation of the flow of vehicles $f(t_n, x_{j+1/2})$ at the interface between the cells j and $(j+1)$, and at time t_n:

$$f(t_n, x_{j+1/2}) \approx \frac{\rho_m \Delta x}{\Delta t} \tilde{F}_j^n, \quad j \in \{0, \dots, N_x\}, \quad n \in \mathbb{N}.$$

Hence, the numerical flux $\tilde{F}_j^n$ can be assimilated to an estimation of the flow of vehicles passing the point $x_{j+1/2}$ at time t_n by scaling this flow by a factor $\rho_m \Delta x / \Delta t$.

Recalling then that, according to the fundamental relations of traffic flow, the speed of vehicles can be obtained as the ratio between the flow and the density of vehicles, we propose to approximate the speed of vehicles $v(t_n, x_{j+1/2})$ at the point $x_{j+1/2}$ at time t_n as

$$v(t_n, x_{j+1/2}) = \frac{f(t_n, x_{j+1/2})}{\rho(t_n, x_{j+1/2})} \approx \frac{f(t_n, x_{j+1/2})}{(\rho_j^n + \rho_{j+1}^n)/2} \approx \frac{\Delta x}{\Delta t} \frac{\tilde{F}_j^n}{(\rho_j^n / \rho_m + \rho_{j+1}^n / \rho_m)/2}, \quad j \in \{0, \dots, N_x\}, \quad n \in \mathbb{N}.$$

In particular, we take $\rho_0^n = \rho_1^n$ and $\rho_{N_x+1} = \rho_{N_x}$ to treat the boundary cases. If we now denote by $\tilde{V}_j^n$ the so-called numerical speed defined by

$$\tilde{V}_j^n = \frac{\tilde{F}_j^n}{(\rho_j^n/\rho_m + \rho_{j+1}^n/\rho_m)/2}, \quad j \in \{0,\dots,N_x\}, \quad n \in \mathbb{N},$$

we obtain that $\tilde{V}_j^n$ can be assimilated to an estimation of the speed of vehicles passing the point $x_{j+1/2}$ at time t_n by scaling it by a factor $\Delta x/\Delta t$.

3. Traffic predictions using neural networks

3.1. Recurrent neural networks

Recurrent neural networks (RNN) are a particular type of neural networks designed to take sequential or time-ordered data as inputs, and widely used in applications such as time series prediction, natural language processing, or speech recognition (Karpathy et al., 2015). We now provide a short introduction on these algorithms, but refer the interested reader to Karpathy et al. (2015), Sutskever (2013), Goodfellow et al. (2016) and Sherstinsky (2020) for more details on the matter.

Given an input sequence $\boldsymbol{X} = (\boldsymbol{x}^{(1)},\dots,\boldsymbol{x}^{(N_{\text{seq}})})$ (composed of vectors of size $N_{\text{in}} > 0$), a RNN computes a sequence of so-called cell states $\boldsymbol{S} = (\boldsymbol{s}^{(1)},\dots,\boldsymbol{s}^{(N_{\text{seq}})})$ (composed of vectors of size $N_{\text{state}} > 0$) which are turned into a sequence of outputs $\boldsymbol{Y} = (\boldsymbol{y}^{(1)},\dots,\boldsymbol{y}^{(N_{\text{seq}})})$ (composed of vectors of size $N_{\text{out}} > 0$). Assuming that an initial state $\boldsymbol{s}^{(0)}$ has been set, the operations performed by the RNN can be summed up by the following recurrence relations:

$$\begin{cases} \boldsymbol{s}^{(n)} = \mathcal{F}_1(\boldsymbol{s}^{(n-1)},\ \boldsymbol{x}^{(n)}), \\ \boldsymbol{y}^{(n)} = \mathcal{F}_2(\boldsymbol{s}^{(n)},\ \boldsymbol{x}^{(n)}), \end{cases} \tag{9}$$

where $n \in \{1,\dots,N_{\text{seq}}\}$ and $\mathcal{F}_1, \mathcal{F}_2$ are two nonlinear (vector-valued) functions depending on parameters determined during the training of the RNN. For instance, in the so-called *Simple RNN* (SRNN), the functions $\mathcal{F}_1$ and $\mathcal{F}_2$ are *MultiLayer Perceptrons* (MLPs) (cf. Appendix A).

A shortcoming of RNNs is the problem of vanishing and exploding gradients, which occur when backpropagation is used to compute the gradient of the cost function used when training the algorithm. This can result in a training phase that fails to produce adequate parameters for the model to yield low-error predictions. The *Long Short-Term Memory* (LSTM) architecture was proposed to avoid this pitfall. In this setting, the state vectors $\boldsymbol{s}^{(n)}$ ($n \in \{1,\dots,N_{\text{seq}}\}$) can be seen as composed by two subvectors $\boldsymbol{c}^{(n)}$ and $\boldsymbol{h}^{(n)}$ of the same size N_{out}, i.e. $\boldsymbol{s}^{(n)} = (\boldsymbol{c}^{(n)}, \boldsymbol{h}^{(n)})$ and $N_{\text{state}} = 2N_{\text{out}}$. The nonlinear functions $\mathcal{F}_1$ and $\mathcal{F}_2$ defining the RNN in (9) then take the form

$$\begin{cases} \boldsymbol{s}^{(n)} = \begin{pmatrix} \boldsymbol{c}^{(n)} \\ \boldsymbol{h}^{(n)} \end{pmatrix} = \begin{pmatrix} \mathcal{P}_1(\boldsymbol{h}^{(n-1)},\ \boldsymbol{x}^{(n)}) \odot \boldsymbol{c}^{(n-1)} + \mathcal{P}_2(\boldsymbol{h}^{(n-1)},\ \boldsymbol{x}^{(n)}) \odot \mathcal{P}_3(\boldsymbol{h}^{(n-1)},\ \boldsymbol{x}^{(n)}) \\ \mathcal{P}_4(\boldsymbol{h}^{(n-1)},\ \boldsymbol{x}^{(n)}) \odot \sigma(\boldsymbol{c}^{(n)}), \end{pmatrix} \\ \boldsymbol{y}^{(n)} = \boldsymbol{h}^{(n)}, \end{cases}$$

where $n \in \{1,\dots,N_{\text{seq}}\}$ and $\mathcal{P}_i$, $i \in \{1,\dots,4\}$ denote single layer perceptrons (cf. Appendix A) and σ denotes the application of a nonlinear activation function (usually $\tanh$) to all entries of a given vector and $\odot$ denotes the component-wise product between two vectors (cf. Fig. 3.1 for a visual representation of an LSTM). Note in particular that $\mathcal{P}_i$, $i \in \{1,\dots,4\}$, takes as input a vector of size $N_{\text{in}} + N_{\text{out}}$ and gives as an output a vector of size N_{out}. Hence, in total, a LSTM RNN has $4N_{\text{out}}(N_{\text{in}} + N_{\text{out}} + 1)$ parameters.)

3.2. The TRM as a RNN

Since the TRM can be seen as a forward Euler discretization of a system of ODEs (cf. Remark 2.1), it can also be interpreted as an instance of RNN for which the successive states of the RNN represent successive states of the discretized system of ODEs (Ruthotto and Haber, 2019; Sherstinsky, 2020). Hence, taking the notations introduced in the previous subsection, the state vector $\boldsymbol{s}^{(n)}$ of the RNN associated with the TRM is no other than the vector containing the normalized numerical solution of PDE (1) at the kth time step, i.e. $\boldsymbol{s}^{(n)} = (\rho_1^n/\rho_m,\dots,\rho_{N_x}^n/\rho_m) \in \mathbb{R}^{N_x}$, where N_x is the number of road cells after discretization. As for the input $\boldsymbol{x}^{(n)}$, it is given by the vector containing the reaction rates at the nth time step and each road interface, i.e. $\boldsymbol{x}^{(n)} = (C_0^{n-1},\dots,C_{N_x}^{n-1}) \in \mathbb{R}^{N_x+1}$.

Since our goal is to perform flow and speed predictions, we consider the TRM as a RNN with two outputs: the numerical fluxes $\boldsymbol{y}_f^{(n)} = (\tilde{F}_0^{n-1},\dots,\tilde{F}_{N_x}^{n-1}) \in \mathbb{R}^{N_x+1}$ and the average speeds $\boldsymbol{y}_v^{(n)} = (\tilde{V}_0^{n-1},\dots,\tilde{V}_{N_x}^{n-1}) \in \mathbb{R}^{N_x+1}$, as defined in Section 2.2. By definition of the TRM recurrence (3), when omitting the source terms, the corresponding RNN Eqs. (9) are given, for any $n \in \{1,\dots,N_{\text{seq}}\}$, by

$$\begin{cases} \boldsymbol{s}^{(n)} = \boldsymbol{s}^{(n-1)} + \boldsymbol{f}^{(n-1)}_{[0:N_x-1]} - \boldsymbol{f}^{(n-1)}_{[1:N_x]}, \\ \boldsymbol{y}_f^{(n)} = \boldsymbol{f}^{(n-1)}, \\ \boldsymbol{y}_v^{(n)} = \boldsymbol{v}^{(n-1)}, \end{cases}$$

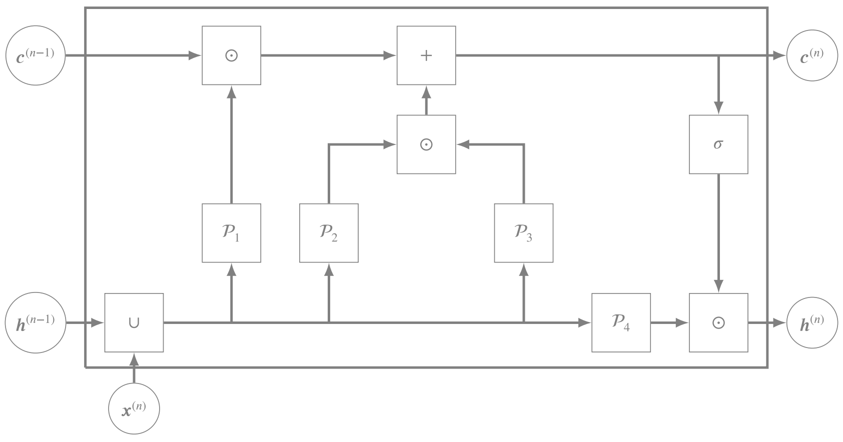


Fig. 3.1. LSTM architecture: $\mathcal{P}_i$, $i \in \{1, \dots, 4\}$ denote single layer perceptrons, $\odot$ denotes the component-wise product between two vectors, $+$ denotes the sum of two vectors and σ denotes the application of an activation function to the entries of a vector.

Table 3.1
Notations used to define the physics-aware neural network.

Notation	Definition
Δx	Space discretization step of the road (also used by the TRM)
ΔT	Time step between consecutive measurements
P_t	Time subdivision factor of the TRM used to comply with the CFL condition (cf. Remark 3.1)
Δt	Time step used by the TRM ($\Delta t = \Delta T / P_t$)
N_x	Number of road cells after discretization (N_x = (Length of the road)/Δx)
N_i	Total number of road interfaces in the road ($N_i = N_x + 1$)
N_o	Number of interfaces for which measurements are available ($N_o \le N_i$)
N_p	Number of past time steps used for the predictions
N_f	Total number of predicted time steps

where the vectors $\boldsymbol{f}^{(n-1)} \in \mathbb{R}^{N_x+1}$ and $\boldsymbol{v}^{(n-1)} \in \mathbb{R}^{N_x+1}$ are defined by

$$\boldsymbol{f}^{(n-1)} = \boldsymbol{x}^{(n-1)} \odot \begin{pmatrix} 1 \\ \hline \boldsymbol{s}^{(n-1)} \end{pmatrix} \odot \begin{pmatrix} 1 - \boldsymbol{s}^{(n-1)} \\ \hline 1 \end{pmatrix}, \quad \boldsymbol{v}^{(n-1)} = \boldsymbol{f}^{(n-1)} \oslash \begin{pmatrix} \boldsymbol{s}^{(n-1)}_{[0]} \\ \hline (\boldsymbol{s}^{(n-1)}_{[0:N_x-2]} + \boldsymbol{s}^{(n-1)}_{[1:N_x-1]})/2 \\ \hline \boldsymbol{s}^{(n-1)}_{[N_x-1]} \end{pmatrix}.$$

Hence, note that the densities approximated by the TRM define the internal state of the corresponding RNN, and not its output. We therefore work with an approach similar to state-space models, where the state of the system is the density of vehicles, the dynamics of the system are defined by the TRM (and therefore by extension, the macroscopic traffic flow model (1)) and the output of the model are the numerical fluxes and average speeds of the TRM.

As a RNN, the TRM has a few particularities. First, and as described above, it has no tuning parameters, and its inputs, output and states are interpretable as components of a traffic flow model. Second, one can note the nonlinearities that appear in the derivation of the TRM are only polynomial, as the states and outputs can be written as polynomial expressions of the inputs and past states.

3.3. Physics-aware neural network for traffic state estimation and prediction

In our setting, we assume that traffic is observed on a stretch of road with no on/off-ramps, for sake of simplicity. We gather in Table 3.1 all the notations used in this section.

The road is discretized into segments of length Δx and the flow and speed of vehicles are recorded every $\Delta T > 0$ seconds at some locations inside the road. In particular, we assume that the data then consists of measurements of the flow and speed of vehicles at some (or all) of the interfaces between the road segments (or "road interfaces" for short). We denote by N_o the number of road interfaces at which measurements are taken and by $N_i \ge N_o$ the total number of road interfaces in the road. In particular, the number N_s of segments used to discretize the road satisfies $N_s = N_i - 1$.

We propose to use RNNs to perform short term predictions of the flow and speed of vehicles based on the TRM. More precisely, we start with measurements at N_p consecutive time steps $t_{-N_p+1}, \dots, t_{-1}, t_0$ (where $t_{i+1} = t_i + \Delta T$, $i \in \{-N_p + 1, \dots, -1\}$) at the N_o observed interfaces of the road and wish to predict the flow and speed of vehicles at time $t_{N_f} = t_0 + N_f \Delta T$ at both the observed and unobserved interfaces of the road. Hence we aim at conjointly inpaint and forecast the flow and speed of vehicles along the road.

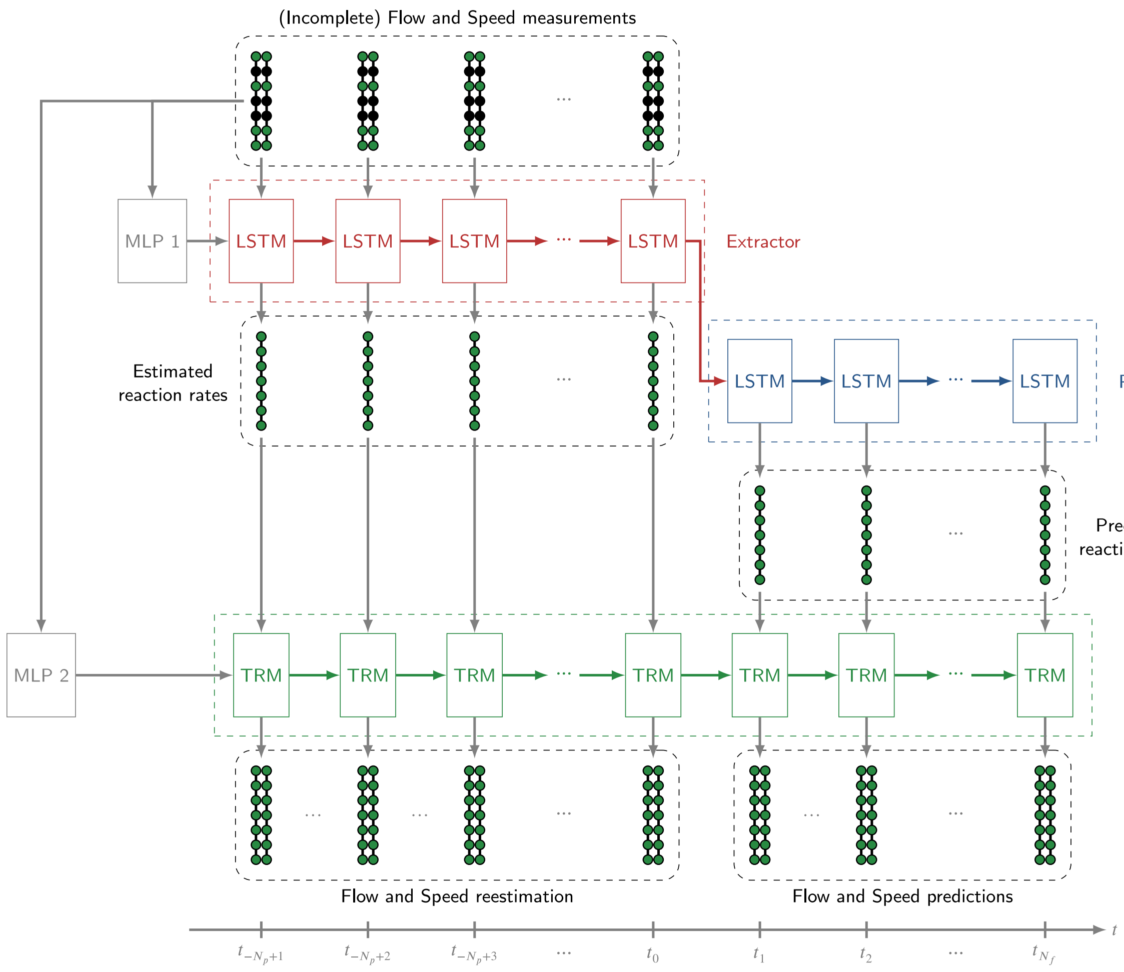


Fig. 3.2. Detail of the architecture used for predictions. The circles stand for road interfaces and green if an observation or estimate is available, and black otherwise. The data, which consists a sequence of flow and speed measurements, is fed to a first LSTM RNN (the Extractor) tasked with estimating the reaction rates that gave rise to the observations. These reaction rates are then extended to future time steps using a second LSTM RNN, the Predictor. All these reaction rates then serve as input of a TRM RNN which produces a smoothed version of the observed traffic states, and extends them to future time steps. The blocks MLP 1 and MLP 2 correspond to the multilayer perceptrons used to initialize respectively the Extractor and the TRM.

We propose to use the TRM RNN in Section 3.2 to model the evolution of the traffic along the road between t_{-N_p+1} and t_{N_f}. We hence consider a TRM defined on the discretized road, and to comply with the CFL condition (4), we follow the multilevel approach proposed in Pereira et al. (2021) and consider a time step Δt for the TRM given by $\Delta t = \Delta T / P_t$ where P_t is an integer such that

$$P_t > \frac{8\|f_m\|_\infty}{\rho_m}\frac{\Delta T}{\Delta x}.$$

Such a choice ensures that Δt and Δx satisfy the CFL condition (4), and therefore the stability of the TRM (Pereira et al., 2021; Lipták et al., 2021).

Remark 3.1. Recall then that ρ_m and $\|f_m\|_\infty$ denote respectively the maximal values the density and flow of vehicles can take. The former can be approximated by the ratio between the number of lanes in the road and the average length of the vehicles. As for the latter, recall that the speed of vehicles can be identified with the ratio between the flow and the density. Assuming that this speed is bounded by some value v_m (for instance 180 km/hour) and using (2), we can upper-bound $\|f_m\|_\infty$ by $v_m\rho_m/4$. Hence, we can take P_t to be the smallest integer such that

$$P_t > 2v_m\frac{\Delta T}{\Delta x}.$$

Then, the corresponding TRM RNN requires us to specify:

- an input sequence, taken as the sequence of $(N_f + N_p - 1)P_t + 1$ vectors, containing the values of the TRM reaction rates (at every road interface) at times $t_{-N_p+1} + k\Delta t$, for $k \in \{0, \dots, (N_f + N_p - 1)P_t\}$;

Table 3.2
Sizes and number of parameters. Note that the Predictor LSTM has no inputs. The notations are defined in Table 3.1. The last column gives the number of parameters for the architecture used in the numerical experiment of Section 4, as an example.

	Input size	State size	Output size	Number of trainable parameters	Numerical experiment
MLP 1	$2N_o$	–	$2N_i$	$4N_i(N_i+N_o+1)$	14,144
Extractor	$N_p \times (2N_o)$	$2N_i$	$N_p \times N_i$	$4N_i(N_i+2N_o+1)$	17,264
MLP 2	N_o	–	N_i-1	$(N_i-1)(N_i+N_o+1)$	3468
Predictor	–	$2N_i$	$N_f \times N_i$	$4N_i(N_i+1)$	11,024
TRM	$\left((N_f+N_p-1)P_t+1\right)\times N_i$	N_i-1	$\left((N_f+N_p-1)P_t+1\right)\times(2N_i)$	0	0
Total	–	–	–	$(13N_i-1)(N_i+N_o+1)$	45,900

- an initial state, taken as the vector containing the density of vehicles inside each road cell at time t_{-N_p+1}.

The TRM RNN then outputs two sequences of $(N_f+N_p-1)P_t+1$ vectors, containing respectively the values of the flow and of the speed of vehicles at every road interface, between t_{-N_p+1} and t_{N_f}. In particular, at t_0, we get a traffic state estimation for the flow and speed of vehicles, available everywhere on the road. The next N_f elements of the output sequences then give the desired flow and speed predictions at times steps $t_1,\dots,t_{N_f-1}$ (also everywhere on the road).

In order to specify the input sequence needed by the TRM RNN, we proceed as follows. Starting from the N_p flow and speed measurements at times $t_{-N_p+1},\dots,t_0$ and at the N_o observed road interfaces, we use a LSTM RNN to estimate the reaction rates at the same time stamps and at all of the road interfaces (including the unobserved ones). Namely, we consider a LSTM network whose input sequence is given by the flow and speed measurements and whose outputs are used as estimates for the reaction rates at times $t_{-N_p+1},\dots,t_0$. We refer to this LSTM network as "Extractor" as it is in fact used to extract reaction rates estimates from flow and speed measurements. The initial state of this RNN is obtained as the output of a 2-layer MLP taking as input the flow and speed measurements at time t_{-N_p+1}.

Then, the reaction rates at times $t_1,\dots,t_{N_f}$ are predicted using a second LSTM RNN with no inputs, and with initial state given by the last state of the Extractor network (which corresponds to the state of Extractor at time t_0). We refer to this second RNN as "Predictor", as it is in fact used to predict reaction rates beyond the time scope of the input measurements (i.e. for times greater than t_0).

Remark 3.2. Note that since the Extractor and Predictor are tasked with estimating and predicting reaction rates, their outputs should take values in $(0,1/2)$, in accordance with the CFL condition (cf. Eq. (5)). This is achieved by choosing the activation functions of these RNNs to be sigmoid functions (hence ensuring that the output is in $(0,1)$) and scaling the output by a factor $1/2$ before using them.

Hence, we end up with reaction rates at times $t_{-N_p+1},\dots,t_0,t_1,\dots,t_{N_f}$ and at every road interface. We then complete these reaction rates by assuming that the reaction rates stay constant between these time steps, thus allowing us to have reaction rates at any time $t_{-N_p+1}+k\Delta t$, for $k\in\{0,\dots,(N_f+N_p-1)P_t\}$ (as needed by the TRM RNN). These reaction rates are then used in the TRM RNN described above and yield estimates for the flow and speed of vehicles at the same time steps. In particular, since the TRM is nothing but the discretization of the continuous traffic model (1), the TRM RNN then recreates, under this traffic model, the traffic dynamics during the time scope of the observations (i.e. between t_{-N_p+1} and t_0) and extends it to future times (i.e. t_1 to t_{N_f}) to yield predictions. As for the initial state of the TRM RNN, it is obtained as the output of a 2-layer MLP taking as input the ratio between the flow and speed measurements at time t_{-N_p+1}.

In the end, our proposed architecture takes as input a sequence of past flow and speed observations, and outputs a re-estimation of these observations under the assumption that they arise from the traffic flow model (1), and then uses this same model to extend the traffic dynamics to future times, hence yielding the flow predictions. In particular, when the observations consist in noisy (raw) measurements, their re-estimation by the TRM RNN can be seen as a smoothing of the measurements. This smoothing is a consequence of the smooth nature of the TRM as a finite volume discretization of a PDE, and was already noted by Pereira et al. (2021). Hence, it has a physical interpretation as resulting from the traffic flow model (1), and is integrated to the learning process. We sum up in Fig. 3.2 the architecture used for predictions.

The number of free parameters to learn, as well as the size of the input, output and (when applicable) state vectors or sequences of the different networks involved in our prediction approach are presented in Table 3.2. These parameters are determined through training, in a supervised learning setting: hence, a cost function evaluating the discrepancy between "ideal" outputs of the network and the actual outputs obtained is minimized to determine a set of optimal parameters.

In practice, we perform training as follows. We assume that we have at our disposal a "long",[1] history of flow and speed measurements. Note in particular that, following Remark 2.3 we scale the flow and speed measurements in order to make them comparable to the outputs of the TRM RNN. From these (scaled) observations, we can form a large set of so-called training examples, which consist of pairs $(\boldsymbol{X},\boldsymbol{Y})$, where

[1] "Long" here can be understood as having a total time scope for the observation much larger than $(N_p+N_f)\Delta t$.

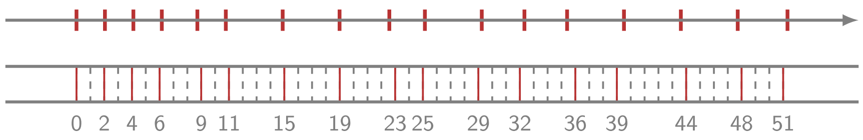


Fig. 4.1. Road and detectors. The red ticks on the horizontal axis mark the actual position of the detectors on the road. Below this axis, we represent the road discretization we consider, obtained by dividing the road into segments of 160 m: we use continuous red lines for the road interfaces for which data are available, and black dashed lines for the others.

- $\boldsymbol{X} = (\boldsymbol{X}_f, \boldsymbol{X}_v)$ is a set containing a sequence of N_p consecutive flow measurements $\boldsymbol{X}_f = (\boldsymbol{f}_{-N_p+1}, \dots, \boldsymbol{f}_0)$ and a sequence of N_p consecutive speed measurements $\boldsymbol{X}_v = (\boldsymbol{v}_{-N_p+1}, \dots, \boldsymbol{v}_0)$, both corresponding to some consecutive observation times $t_{-N_p+1}, \cdots, t_0$;
- $\boldsymbol{Y} = (\boldsymbol{Y}_f, \boldsymbol{Y}_v)$ is a set containing a sequence of $N_p + N_f$ consecutive flow measurements $\boldsymbol{Y}_f = (\boldsymbol{f}_{-N_p+1}, \dots, \boldsymbol{f}_{N_f})$ and a sequence of $N_p + N_f$ consecutive speed measurements $\boldsymbol{Y}_v = (\boldsymbol{v}_{-N_p+1}, \dots, \boldsymbol{v}_{N_f})$, both corresponding to some consecutive observation times $t_{-N_p+1}, \dots, t_0, \dots, t_{N_f}$; note in particular that the first N_p elements of $\boldsymbol{Y}_f$ (resp. $\boldsymbol{Y}_v$) are the same as the elements of $\boldsymbol{X}_f$ (resp. $\boldsymbol{X}_v$).

In an example $(\boldsymbol{X}, \boldsymbol{Y})$, $\boldsymbol{X}$ is used as an input of the flow and speed prediction network $\mathcal{N}$ that we are training, and $\boldsymbol{Y}$ is used as the ideal output we would like to get from $\mathcal{N}$.

Let us denote by $\widehat{\boldsymbol{Y}} = \mathcal{N}(\boldsymbol{X}, \boldsymbol{\theta}) = (\widehat{\boldsymbol{Y}}_f, \widehat{\boldsymbol{Y}}_v)$ the output of $\mathcal{N}$ with a set of parameters $\boldsymbol{\theta}$, and when inputting $\boldsymbol{X}$. In particular, $\widehat{\boldsymbol{Y}}_f = (\widehat{\boldsymbol{f}}_{-N_p+1}, \dots, \widehat{\boldsymbol{f}}_{N_f})$ is the sequence of estimated and predicted flows, and $\widehat{\boldsymbol{Y}}_v = (\widehat{\boldsymbol{v}}_{-N_p+1}, \dots, \widehat{\boldsymbol{v}}_{N_f})$ is the sequence of estimated and predicted speeds. Let us then consider a set of N_{ex} examples $\{(\boldsymbol{X}^{(j)}, \boldsymbol{Y}^{(j)})\}$, $j \in \{1, \dots, N_{\text{ex}}\}$. We determine the optimal set of parameters $\boldsymbol{\theta}$ associated with these examples by minimizing the cost function L given by

$$L(\boldsymbol{\theta}) = \frac{1}{\alpha_f}\ell(\boldsymbol{Y}_f, \widehat{\boldsymbol{Y}}_f) + \frac{1}{\alpha_v}\ell(\boldsymbol{Y}_v, \widehat{\boldsymbol{Y}}_v) + \frac{1}{\alpha_r}R(\boldsymbol{\theta}), \tag{10}$$

where $\alpha_f, \alpha_v, \alpha_r > 0$ are scaling constants, $R(\boldsymbol{\theta})$ is a regularization term, and the term $\ell(\boldsymbol{Y}_f, \widehat{\boldsymbol{Y}}_f)$ (resp. $\ell(\boldsymbol{Y}_v, \widehat{\boldsymbol{Y}}_v)$) measures the discrepancy, over the examples, between the network outputs and the flow (resp. speed) measurements. In particular, we evaluate these discrepancies in the mean-squared sense, i.e. we take

$$\ell(\boldsymbol{Y}_f, \widehat{\boldsymbol{Y}}_f) = \frac{1}{N_{\text{ex}}}\sum_{j=1}^{N_{\text{ex}}}\left(\frac{1}{N_p}\sum_{k=-N_p+1}^{0}\left\|\widehat{\boldsymbol{f}}_k^{(j)} - \boldsymbol{f}_k^{(j)}\right\|^2 + \frac{1}{N_f}\sum_{l=1}^{N_f}\left\|\widehat{\boldsymbol{f}}_l^{(j)} - \boldsymbol{f}_l^{(j)}\right\|^2\right),$$

$$\ell(\boldsymbol{Y}_v, \widehat{\boldsymbol{Y}}_v) = \frac{1}{N_{\text{ex}}}\sum_{j=1}^{N_{\text{ex}}}\left(\frac{1}{N_p}\sum_{k=-N_p+1}^{0}\left\|\widehat{\boldsymbol{v}}_k^{(j)} - \boldsymbol{v}_k^{(j)}\right\|^2 + \frac{1}{N_f}\sum_{l=1}^{N_f}\left\|\widehat{\boldsymbol{v}}_l^{(j)} - \boldsymbol{v}_l^{(j)}\right\|^2\right).$$

Note that the first two inner-sums defining $\ell(\boldsymbol{Y}_f, \widehat{\boldsymbol{Y}}_f)$ and $\ell(\boldsymbol{Y}_v, \widehat{\boldsymbol{Y}}_v)$ have very particular roles: the first one assesses the capacity of the TRM to recreate the dynamics of the observations, and the second one assesses its capacity to extend it to make predictions.

As for the regularization term, its role is to counteract the tendency the TRM might have to overfit the data through physically unsound reaction rates. As proposed by Pereira et al. (2021), a regularization term limiting the spatial and temporal variations of these reaction rates can be used for this purpose: we hence choose $R(\boldsymbol{\theta})$ to be defined as

$$R(\boldsymbol{\theta}) = \frac{1}{2}\left(\frac{1}{N_p + N_f}\sum_{n=-N_p+1}^{N_f}\frac{1}{N_x}\sum_{i=0}^{N_x-1}\left(C_{i+1}^n - C_i^n\right)^2 + \frac{1}{N_x+1}\sum_{i=0}^{N_x}\frac{1}{N_p+N_f-1}\sum_{n=-N_p+1}^{N_f-1}\left(C_i^{n+1} - C_i^n\right)^2\right).$$

Finally, note that the scaling constants α_f, α_v and α_r are chosen to normalize the errors computed in the terms $\ell(\boldsymbol{Y}_f, \widehat{\boldsymbol{Y}}_f)$, $\ell(\boldsymbol{Y}_v, \widehat{\boldsymbol{Y}}_v)$ and $R(\boldsymbol{\theta})$: namely we take α_f as an upper-bound for the square of flow values, α_v as an upper-bound for the square of speed values, and α_r as an upper-bound for the square of reaction rates. In practice, following Remarks 3.1 and 3.2 and the scaling of flow and speed values discussed in Remark 2.3, we set:

$$\sqrt{\alpha_f} = (v_m \rho_m/4)/(\rho_m \Delta x/\Delta t), \quad \sqrt{\alpha_v} = v_m/(\Delta x/\Delta t), \quad \sqrt{\alpha_r} = 1/2. \tag{11}$$

4. Numerical experiments

4.1. Setup

The code used in this section was implemented in Python, using Tensorflow and Keras (2.4.0). We use a dataset which consists of flow and speed measurements made by detectors placed along a stretch of freeway A12 in the Netherlands (courtesy of Rijkswaterstaat – Centre for Transport and Navigation). This data was collected (almost) every day from January 5, 2006 to May 19, 2006. Each day, the flow and speed of vehicles at each station were recorded every minute, for a total duration of 5 h.

Table 4.1
Prediction RMSE obtained when using our approach (TRM) and when repeating the last known measurement. Note that the flow (resp. speed) RMSE values are scaled by a factor $\sqrt{\alpha_f}$ (resp. $\sqrt{\alpha_v}$).

(a) Errors on flow prediction.

	TRM Prediction	Last known measurement
3-step prediction	4.24e−02	5.72e−02
5-step prediction	4.40e−02	6.07e−02
10-step prediction	4.90e−02	6.58e−02

(b) Errors on speed prediction.

	TRM Prediction	Last known measurement
3-step prediction	4.66e−02	6.34e−02
5-step prediction	5.16e−02	7.81e−02
10-step prediction	6.35e−02	9.88e−02

In the numerical experiment, we consider only a subsection of the road of length $\approx 8210\,\mathrm{m}$. This stretch of road and the position of its 17 detectors is represented in Fig. 4.1, where we can notice that the detectors are approximately separated by (multiples of) 160 m. This allows us to discretize the road into segments of length $\Delta x = 160\,\mathrm{m}$ and have the detectors located at the interfaces of these segments (cf. Fig. 4.1). Note in particular that the considered stretch of road contains an off-ramp (which starts at the location of the interface 23) and an on-ramp (which ends at the location of the interface 29). Nevertheless, we do not explicitly include them in the TRM as we have no information on the in- and out-fluxes of vehicles they generate.

The Dutch motorway dataset contains 129 days of measurements. We create a training dataset containing 93 days of measurements, a validation set containing 18 days of measurements (used to choose the hyperparameters of the algorithm) and finally use the remaining 18 days of measurements as a test set (used to check the performances of the algorithm). These measurements are used as is, without any smoothing or preprocessing (besides a spatial linear interpolation when a sensor stopped working). Besides, while training the algorithm, we hide the data from the detectors marked as 2 and 25 in Fig. 4.1 and which we refer to as validation interfaces (or validation points), in order to assess the capacity of the algorithm to complete and predict the flow at unobserved interfaces. In the end, we have $N_o = 15$ observed interfaces (those marked $0, 4, 6, 9, 11, 15, 19, 23, 29, 32, 36, 39, 44, 48, 51$ in Fig. 4.1), for a total of $N_i = 52$ road interfaces.

We consider the problem of predicting the flow and speed at a road interface up to 10 minutes in the future. Taking the notations introduced in Section 3.3, we therefore consider the architecture yielding a prediction $N_f = 10$ time steps in the future. The algorithm will then return both a smooth estimate of the most recent measurements it received as an input, as well as predictions of what the flow and speed might be $1, 2, \dots, 10$ time steps in the future, one time step being equal to $\Delta T = 1$ minute. Finally, note that the subdivision factor P_t of the time step is set according to Remark 3.1, and is equal to $P_t = 38$.

4.2. Analysis of the results

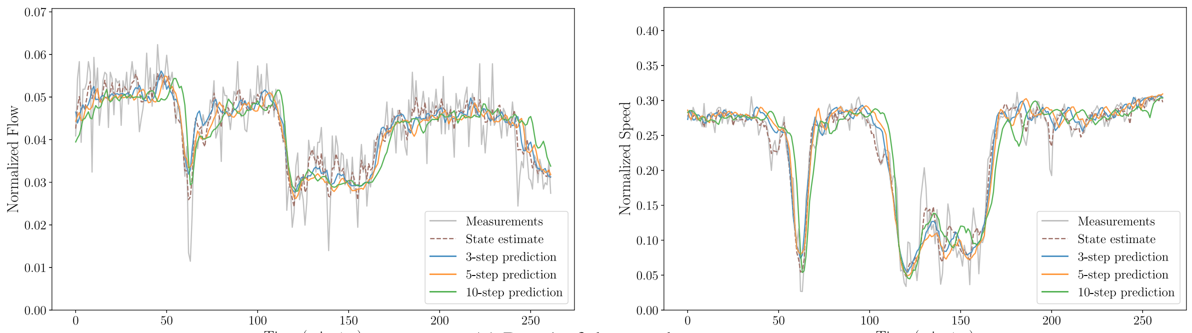
In order to determine the number of past observations N_p, we perform a quick grid search: we train the algorithm with $N_p \in \{5, 7, 10, 12, 15, 18, 20\}$ and choose the value of N_p yielding the smallest root-mean-square error (RMSE) and mean absolute percentage error.[2] (MAPE) over the validation dataset. These errors are reported in Appendix B Based on these results, we take $N_p = 18$ as this value yields small values of both RMSE and MAPE. Once that the number of past observations is chosen, we apply the associated training algorithm to the validation dataset to assess its performance.

4.2.1. Comparison of the prediction to the traffic state measurements

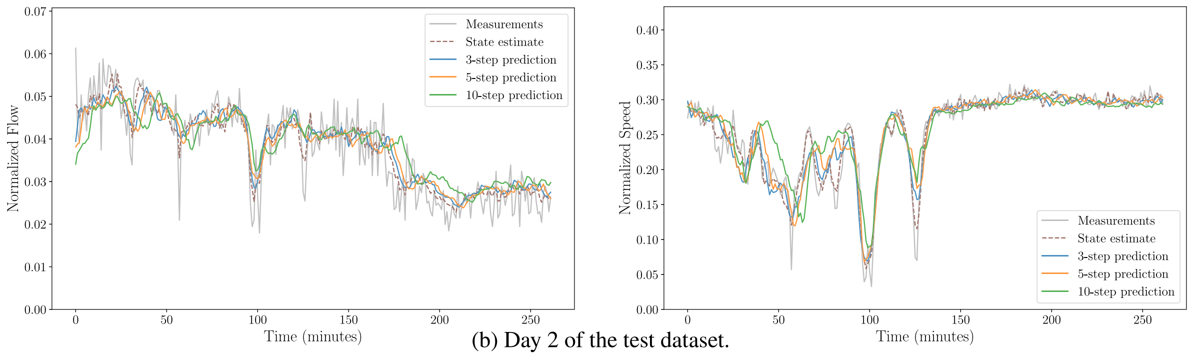
First, we look at how good the predictions 3, 5 and 10 minutes ahead are. Examples of such predictions are presented in Fig. 4.2, where we show for two different days in the test dataset, a comparison between the 3-, 5- and 10-step flow and speed predictions and the corresponding measurements. We also compute, over all the measurements in the test dataset (but considering only the road interfaces used in the training phase), the RMSE between predictions and measurements (also called prediction RMSE). We compare them to the RMSE obtained when we use the last known measurement (i.e. the measurement at time t_0) as a predictor for the flow in the future (i.e. at time t_3, t_5 or t_{10}). These errors are reported in Table 4.1. We note that the TRM predictions show an improvement of 25% to 27% for the flow, and of 26% to 35% for the speed, compared to the case where the last known measurement is used as predictor.

We then take a closer look at how these prediction RMSEs distribute themselves along the road. To do so, we compute, on each road interface, the RMSE between the TRM predictions and the measurements, and display these errors in Fig. 4.3. We note that the errors on the trained interfaces are stable across the interfaces, and that therefore there is no major difference in the quality of the predictions across space. As for the two validation interfaces 2 and 25 (which were hidden during training), we see that the flow prediction RMSE takes values of the same order as the one obtained on the trained interfaces. For the speed prediction RMSE however, we observe significantly higher RMSE on the validation interfaces (around twice as much). This raises the question of the level of smoothing introduced by our approach, which we now investigate.

[2] The mean absolute percentage error between a set of n measurements $y^{(1)}, \dots, y^{(n)}$ and some associated predictions $\hat{y}^{(1)}, \dots, \hat{y}^{(n)}$ is defined as: MAPE $= (1/n) \sum_{k=1}^{n} |\hat{y}^{(k)} - y^{(k)}| / |y^{(k)}|$.

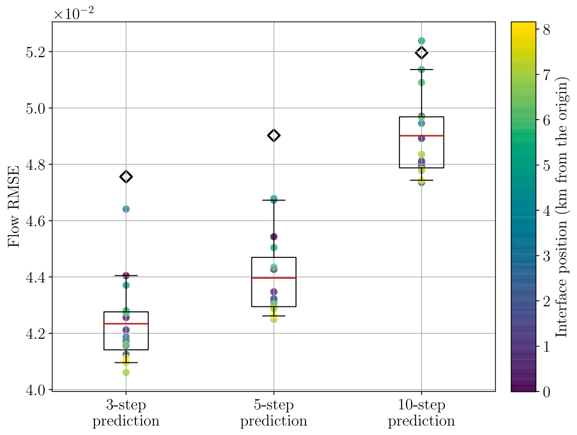


(a) Day 1 of the test dataset.

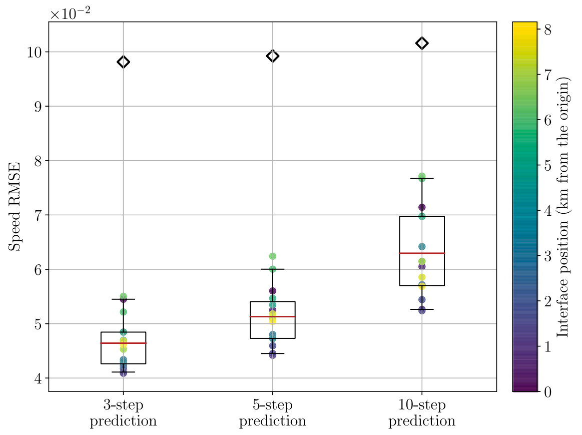


(b) Day 2 of the test dataset.

Fig. 4.2. Measured, smoothed and predicted flows (on the left) and speeds (on the right) on the road interface 11 on two different days of the test set. Note that the flow and speed measurements are normalized according to Remark 2.3.



(a) Flow predictions.

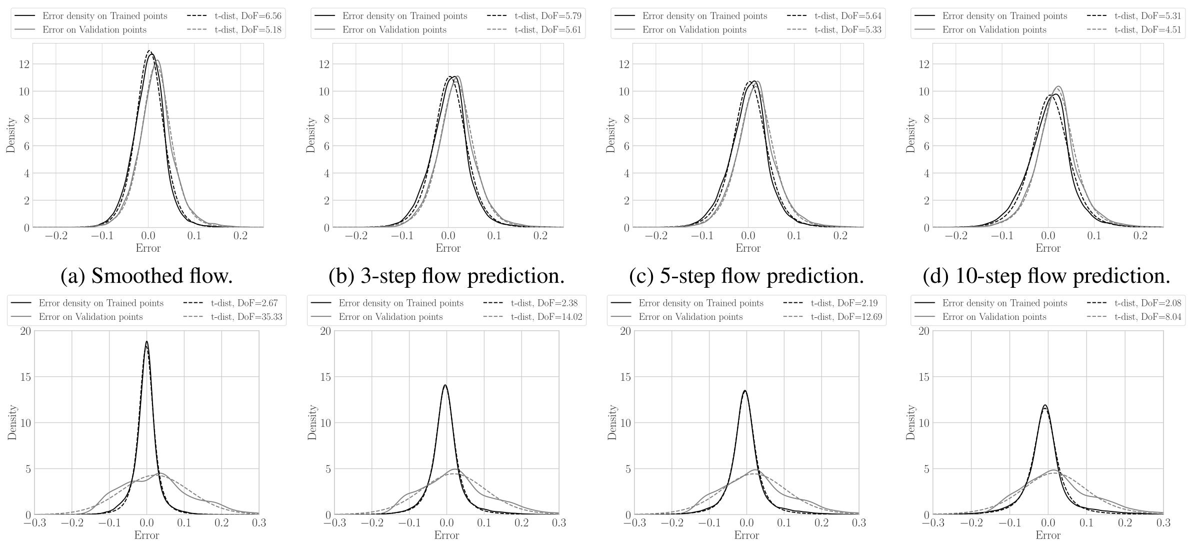


(b) Speed predictions.

Fig. 4.3. Prediction RMSE on each road interface. Each circle marks the RMSE between prediction and measurements at a given interface, and the color of a circle represents the location of the interface on the road. The symbol ⋄ marks the RMSE obtained on the validation points. As for the boxplot, the whiskers mark the 5th and 95th quantiles and the box marks the 1st and 3rd quartiles. Finally, the red line marks the mean RMSE over the trained points. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.2.2. Smoothing of the traffic states

Recall that our algorithm yields predictions by recreating, under a macroscopic traffic model, the traffic dynamics as observed in the past observations (at times $t_{-17}, \dots, t_0$) and extending them to future times $(t_1, \dots, t_{10})$. In particular, at t_0, a smoothed version of the most recent observation is available: we refer to it as smoothed or estimated flow/speed. These flow and speed estimates are for instance represented in some particular cases in Fig. 4.2, and represent the current traffic state as seen by the macroscopic traffic model. We perform in Fig. 4.4 a comparison between the measurements and these smoothed states, as well as between the measurements and their predictions.



(a) Smoothed flow. (b) 3-step flow prediction. (c) 5-step flow prediction. (d) 10-step flow prediction.

(e) Smoothed speed. (f) 3-step speed prediction. (g) 5-step speed prediction. (h) 10-step speed prediction.

Fig. 4.4. Distribution of the difference (also called error) between various quantities computed by the algorithms and the corresponding measurements. The black (resp. gray) solid lines correspond to error densities computed on the trained (resp. validation) interfaces. The dashed lines present the result of fitting Student t-distributions on these densities. Note that the flow (resp. speed) error values are scaled by a factor $\sqrt{\alpha_f}$ (resp. $\sqrt{\alpha_v}$).

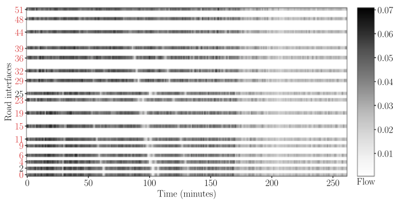
Regarding the smoothed and predicted flows, note that the errors (defined as the plain difference) between the estimated quantities and the measurements, and represented in Figs. 4.4(a)–4.4(d), have approximately symmetric distributions, may it be the errors computed on the trained interfaces or on the validation interfaces. We note that these distributions are approximately centered around 0, with very slight biases around 0 that may be explained by the limited size of the samples used to compute the densities. Hence, we can conclude that the algorithm does not tend to overestimate or underestimate the flows, and behaves similarly whether or not the interface was trained (since in both cases, the distributions are very similar). This therefore hints that the smoothing and predictions of the flow are robust to missing values.

We also fitted Student t-distributions onto these error densities, and obtained that they can be well approximated by Student t-distributions with a number of degrees of freedom between 4.5 and 6.5 (cf. Figs. 4.4(a)–4.4(d)). Hence, we have a non-Gaussian distribution of these errors, with fatter tails than a Gaussian distribution. These fat tails reflect the ability of the algorithm to propose smoothed and predicted flows that are robust to outliers in the measurements. Note however that since the number of degrees of freedom of the fitted distributions are relatively high, they behave quite similarly to a Gaussian distribution, and hence the over-smoothing of extreme measurements is limited.

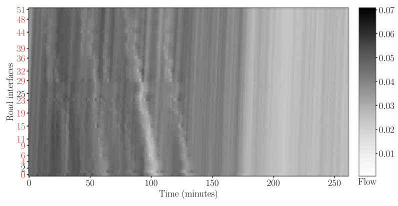
Regarding now the smoothed and predicted speeds obtained on the trained interfaces, we can draw the same conclusions as for the flow. The only difference is that the number of degrees of freedom of the fitted t-distributions now lies between 2 and 3 (cf. Figs. 4.4(e)–4.4(h)). This means that now, the tails of the error distribution are significantly fatter than the tails of a Gaussian distribution. Hence, the algorithm tends to over-smooth extreme speed measurements.

As for the smoothed and predicted speeds obtained on the validated interfaces, their distributions can be approximated by (nearly) centered t-distributions with a high degree of freedom (between 8 and 35), and therefore by Gaussian distributions. Note also that they exhibit higher variances than the errors computed on the trained cells. This higher level of smoothing of the measurements can be explained by the fact that the validation interface 25 is located in the section of the road that includes an on-ramp, but we did not include on-ramps into our TRM (since no measurements were available). This hence seems to hint that our algorithms manage to recreate the perturbation of flow dynamics by on-ramps, but struggle to do it accurately for the speed, proposing instead an over-smoothed estimate. Such caveat could be resolved by actually including on-ramps into the architecture or resorting to PDE models that allow more flexibility on the speed modeling (eg. second-order models).

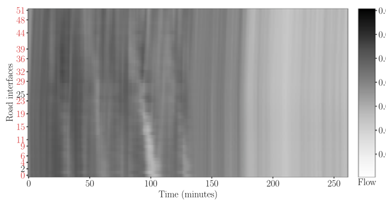
In conclusion, our algorithm yields smoothed flows that are physics-aware (since they arise from the macroscopic traffic model) and well-behaved (since they are robust to outliers and unbiased). Besides, these smoothed flows are available on all of the interfaces dividing the road, and not only at the locations where measurements were taken. We give in Figs. 4.5 and 4.6 an example of the full reconstruction of the flow and speed, as well as predictions of these variables for a given day of the test dataset. Besides the smoothing and inpainting capabilities of our approach, we note that the method is also able to recreate waves propagating through space time, thanks to the fact that the predictions arise from a macroscopic traffic model which incorporates them. Note also that between the interfaces 23 and 29, we observe a discontinuity in the waves, which can once again be seen as a direct consequence of the presence of an on-ramp between these interfaces which perturbed the traffic conditions.



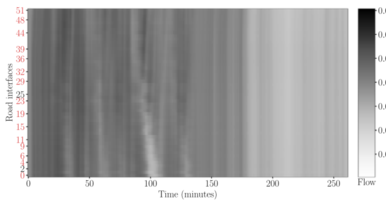
(a) Flow measurements.



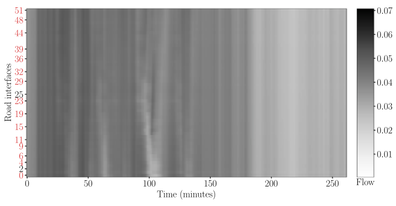
(b) Reconstructed smooth flow.



(c) 3-step prediction.

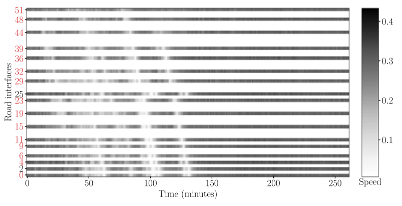


(d) 5-step prediction.

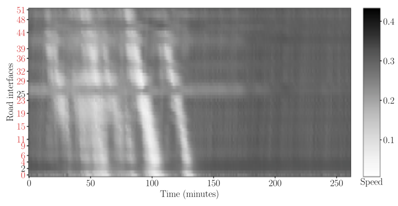


(e) 10-step prediction.

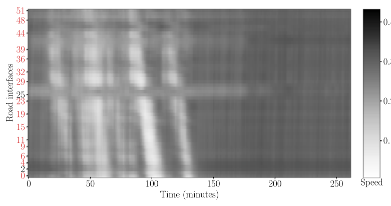
Fig. 4.5. Comparison between the space–time variations of the flow measurements with the reconstructed smooth flow and prediction on Day 2 of the test dataset. The indices marked in red correspond to the observed interfaces, and those in black to the hidden interfaces. The grayscale represents the value of the normalized flow at a given location and time.



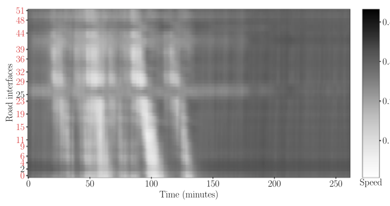
(a) Flow measurements.



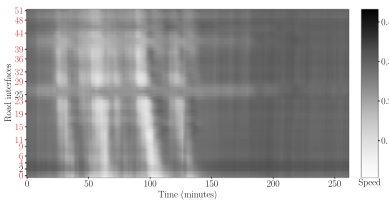
(b) Reconstructed smooth flow.



(c) 3-step prediction.



(d) 5-step prediction.



(e) 10-step prediction.

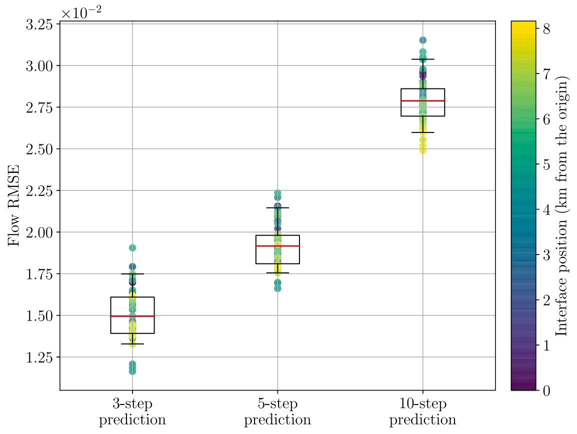
Fig. 4.6. Comparison between the space–time variations of the speed measurements with the reconstructed smooth speed and predictions on Day 2 of the test dataset. The indices marked in red correspond to the observed interfaces, and those in black to the hidden interfaces. The grayscale represents the value of the normalized speed at a given location and time.

4.2.3. *Comparison of the prediction to the traffic state estimates*

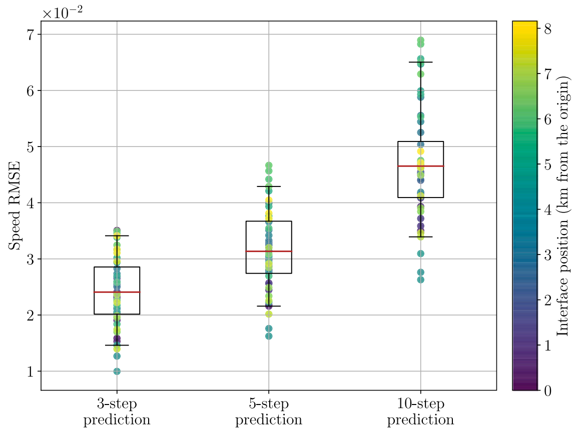
If we now take the smoothed flows, which are available everywhere on the road, as ground truth when computing prediction RMSEs (instead of using the raw flow measurements), we note that we get errors that are drastically reduced and less spread out compared to the case where we compare our predictions to the raw measurements (cf. Fig. 4.7): indeed, reductions ranging from 40% to 70% can be observed on the considered prediction horizons when considering either the trained or validation interfaces (cf. Appendix C). These observations are coherent with the fact that our algorithm was optimized to extend to the future the dynamics that gave rise to the smoothed flow: hence, the predictions obtained by our algorithm can be seen as accurate predictions of a physics-aware smoothing of the raw measurements.

4.3. *Comparison to existing methods*

Comparing the prediction performances obtained in the numerical experiment with other works is no easy task. Indeed, for the comparison to be fair, our algorithm should be compared to approaches that take as input the same type of information, namely only past raw flow measurements taken at a relatively high frequency (every minute in our case). Besides, the question of the metric used to compare different studies is not straightforward. Indeed the prediction errors are compared based on a comparison between the



(a) Flow.



(b) Speed.

Fig. 4.7. RMSE between the predictions and the smoothed flow or speed computed by the algorithm, over the entire road. Each circle marks the RMSE between the prediction and the smoothed flow or speed computed by the algorithm at a given interface, and the color of a circle represents the location of the interface on the road. As for the boxplot, the whiskers mark the 5th and 95th quantiles and the box marks the 1st and 3rd quartiles. Finally, the red line marks the mean. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

prediction and what is assumed to be the ground truth. However, most of the studies apply some preprocessing on the raw data (e.g. aggregation or smoothing) and use the result as ground truth. Since these steps usually differ from one study to the other, and are not present at all in ours, comparing the performances becomes complicated. Finally, the question of the amount of parametrization in a model and the amount of data used to train it play a central role in the performance of an algorithm, and are hard to compare from one approach to the other.

Let us omit the above comments and compare anyway the prediction performances of our algorithm with other deep neural network approaches. To the best of our knowledge, no physics-aware neural network has been proposed for traffic state prediction and the physics-informed neural networks proposed up until now mainly focus on traffic state estimation (and in particular density) rather than short-term prediction. We were only able to find the work of Liu et al. (2021), who proposed an architecture for density (not flow or speed) estimation and prediction, and the work of Barreau et al. (2021) which focused on density reconstruction from probe vehicles measurements. However both methods were not evaluated on real data or compared to other methods. On the other hand, more studies have been conducted on the use of deep learning methods for short-term predictions, which is why we focused on them. As noted in the review article of Do et al. (2019), such approaches were able to yield predictions MAPE ranging from 5% to 10%. Compared to the MAPEs we computed from the smoothed flows (cf. Fig. 4.8) we retrieve similar values.

Hence, the takeway message we would like to convey is that it is possible to propose an architecture that takes care of both traffic state estimation and prediction from raw data, yields physically-constrained estimates and predictions, and does not lose in accuracy compared to the over-parameterized deep learning models.

5. Conclusion

In this work, we propose a physics-aware algorithm for short-term flow and speed predictions based a macroscopic traffic flow model. This algorithm has two main components. The first one is a RNN which aims at extracting the space–time varying parameters of the traffic flow model associated with a set of past observations and extending them to future time steps. These parameters are then passed to the second component, which is a TRM discretization of the traffic flow model: we then obtain both flow and speed estimates, at past and future time steps, that arise directly from the traffic flow model. A numerical experiment, conducted on a dataset of raw flow and speed measurements taken along a stretch of freeway in the Netherlands, highlights the advantages of our approach.

First, the algorithm outputs a traffic state estimation of the most recent time of measurement it got as input, which can be used as a smooth estimate for the traffic state. This is particularly desirable when dealing with raw data, for which choosing and justifying the right aggregation or smoother to reduce the noise can be cumbersome. In our case, the noise reduction is streamlined with the prediction and is physically grounded by the macroscopic traffic flow model. As for the predictions, they arise from the same traffic model and are therefore a simple continuation in time of the dynamics inferred from the past observations. Hence both the smoothing of raw data and the subsequent predictions have a clear physical interpretation. Finally, the algorithm yields smooth flow and speed estimates and predictions everywhere on the road (and not only at the locations where measurements are made). Hence our algorithm also serves as a data augmentation tool.

Future work include the extension of the approach to networks, which can be made possible by extending the TRM part of the architecture to networks. Besides, note that the TRM, as described by Lipták et al. (2021), can be used to model more complex fundamental diagrams than the Greenshields flux, including for instance trapezoidal fundamental diagrams. Adapting our current

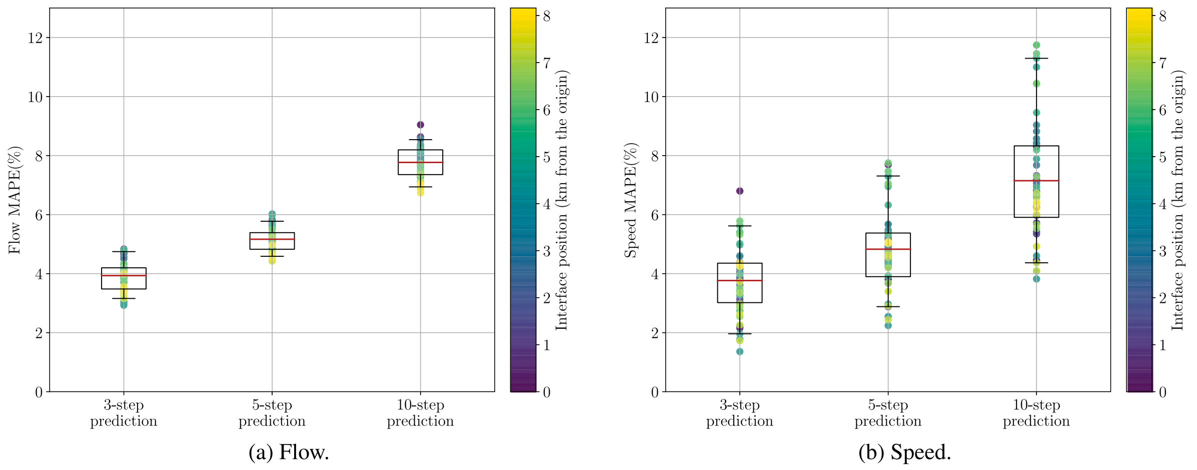


Fig. 4.8. MAPE between the predictions and the smoothed flow or speed computed by the algorithm, over the entire road. Each circle marks the MAPE between the prediction and the smoothed flow or speed computed by the algorithm at a given interface, and the color of a circle represents the location of the interface on the road. As for the boxplot, the whiskers mark the 5th and 95th quantiles and the box marks the 1st and 3rd quartiles. Finally, the red line marks the mean. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

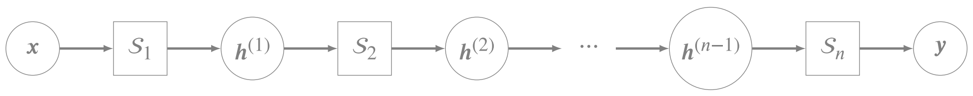


Fig. A.1. Multilayer perceptron: $S_1,\dots,S_n$ denote single layer perceptrons, $h^{(1)},\dots,h^{(n-1)}$ denote hidden layers, and circles are used to denote input and/or output vectors.

architecture to such fundamental diagrams then provides an interesting outlook. Finally, proposing physics-aware architectures based on more complex traffic flow models than the LWR model used in this work, like for instance second-order models, might prove useful to improve some of the prediction performances, in particular on points of the road where no measurements are taken.

CRediT authorship contribution statement

Mike Pereira: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration. **Annika Lang:** Conceptualization, Methodology, Validation, Formal analysis, Writing – original draft, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Balázs Kulcsár:** Conceptualization, Methodology, Validation, Formal analysis, Investigation, Resources, Writing – original draft, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Appendix A. Multilayer perceptrons

We provide in this section a short definition of the notions of single layer and multilayer perceptrons, and refer the interested reader to Bishop (2006) for more details.

A (single layer) perceptron is an artificial neural network (ANN) which takes as input a vector $\boldsymbol{x} \in \mathbb{R}^{N_{\text{in}}}$ ($N_{\text{in}} \in \mathbb{N}$) and returns an output vector $\boldsymbol{y} \in \mathbb{R}^{N_{\text{out}}}$ ($N_{\text{out}} \in \mathbb{N}$) whose entries $y_1, \dots, y_{N_{\text{out}}}$ are given by:

$$y_i = \sigma(\boldsymbol{W}^{(i)}\boldsymbol{x} + b_i), \quad i \in \{1, \dots, N_{\text{out}}\},$$

where $\boldsymbol{W}^{(i)} \in \mathbb{R}^{N_{\text{in}} \times N_{\text{out}}}$ is a weight matrix, $b_i \in \mathbb{R}$ is a bias term and σ is a nonlinear function called activation function. Examples of activation functions commonly used in practice include the Heaviside function, the sigmoid function, the hyperbolic tangent and the ReLu function. Hence each entry of the output of a perceptron is nothing but a linear combination of the entries of its input vector to which we add a bias term and apply an activation function.

A MultiLayer Perceptron (MLP), also called feed-forward neural network, is the ANN obtained by stacking together multiple (single layer) perceptrons: the output of each perceptron is used as input of the next one (cf. Fig. A.1). In such a configuration, the layers between the input vector and the final output vector are called hidden layers, and single layer perceptrons can be seen as a particular case of MLP with no hidden layer.

Table B.1
Grid search results: prediction RMSE and MAPE on the validation dataset when predicting up to $N_f = 10$ time steps ahead using N_p past observations. Note that the flow (resp. speed) RMSE values are scaled by a factor $\sqrt{\alpha_f}$ (resp. $\sqrt{\alpha_v}$).

	$N_p = 5$	$N_p = 7$	$N_p = 10$	$N_p = 12$	$N_p = 15$	$N_p = 18$	$N_p = 20$
RMSE Flow	5.01e−02	4.92e−02	4.94e−02	4.88e−02	4.86e−02	4.85e−02	4.87e−02
RMSE Speed	5.22e−02	5.16e−02	5.11e−02	5.05e−02	5.08e−02	5.11e−02	5.21e−02
MAPE Flow	18.99%	18.43%	18.26%	17.27%	17.26%	16.37%	16.72%
MAPE Speed	9.73%	9.53%	9.6%	9.38%	9.46%	9.57%	10.08%

Table C.1
RMSE between the predictions and either the measurements, or the smoothed flow or speed computed by the algorithm. Note that the flow (resp. speed) RMSE values are scaled by a factor $\sqrt{\alpha_f}$ (resp. $\sqrt{\alpha_v}$).

(a) Errors on flow prediction.

	On training points		On validation points	
	Prediction/Raw measurement	Prediction/Smoothed flow	Prediction/Raw measurement	Prediction/Smoothed flow
3-step prediction	4.24e−02	1.65e−02	4.76e−02	1.39e−02
5-step prediction	4.40e−02	2.04e−02	4.90e−02	1.88e−02
10-step prediction	4.90e−02	2.87e−02	5.20e−02	2.81e−02

(b) Errors on speed prediction.

	On training points		On validation points	
	Prediction/Raw measurement	Prediction/Smoothed Speed	Prediction/Raw measurement	Prediction/Smoothed Speed
3-step prediction	4.66e−02	2.82e−02	9.81e−02	1.44e−02
5-step prediction	5.16e−02	3.57e−02	9.92e−02	1.98e−02
10-step prediction	6.35e−02	5.09e−02	1.02e−01	3.15e−02

Appendix B. Results of the parameter search

We present here the results of the grid search performed to choose the number of past observations used by the prediction algorithm in the numerical experiment in Section 4. (See Table B.1).

Appendix C. Comparison of the prediction errors on the trained and validation interfaces

We present here the prediction errors obtained (on the test dataset) when comparing the predictions to either the raw measurements or to the smooth states estimated by the algorithm. (See Table C.1).

References

Abhishek, K., Misra, B.B., 2016. Hybrid genetic algorithm and time delay neural network model for forecasting traffic flow. In: 2016 IEEE International Conference on Engineering and Technology. ICETECH, pp. 178–183.

Barreau, M., Aguiar, M., Liu, J., Johansson, K.H., 2021. Physics-informed learning for identification and state reconstruction of traffic density. arXiv preprint arXiv:2103.13852.

Berg, J., Nyström, K., 2017. Neural network augmented inverse problems for PDEs. arXiv preprint arXiv:1712.09685.

Bishop, C.M., 2006. Pattern recognition. Mach. Learn. 128 (9).

Chainais-Hillairet, C., Champier, S., 2001. Finite volume schemes for nonhomogeneous scalar conservation laws: Error estimate. Numer. Math. 88 (4), 607–639.

Chen, G.-Q., Karlsen, K.H., 2005. Quasilinear anisotropic degenerate parabolic equations with time-space dependent diffusion coefficients. Commun. Pure Appl. Anal. 4 (2), 241–266.

Dal Santo, N., Deparis, S., Pegolotti, L., 2020. Data driven approximation of parametrized PDEs by reduced basis and neural networks. J. Comput. Phys. 416, 109550.

Do, L.N.N., Taherifar, N., Vu, H.L., 2019. Survey of neural network-based models for short-term traffic state prediction. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 9 (1), e1285.

Feinberg, M., 2019. Foundations of Chemical Reaction Network Theory. Springer.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press, http://www.deeplearningbook.org.

Greenshields, B.D., Bibbins, J.R., Channing, W.S., Miller, H.H., 1935. A study of traffic capacity. In: Highway Research Board Proceedings, vol. 1935, National Research Council (USA), Highway Research Board.

Huang, J., Agarwal, S., 2020. Physics informed deep learning for traffic state estimation. In: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems. ITSC, IEEE, pp. 1–6.

Jiang, X., Adeli, H., 2004. Wavelet packet-autocorrelation function method for traffic flow pattern analysis. Comput.-Aided Civ. Infrastruct. Eng. 19 (5), 324–337.

Karlsen, K.H., Towers, J.D., 2004. Convergence of the Lax–Friedrichs scheme and stability for conservation laws with a discontinuous space-time dependent flux. Chin. Ann. Math. 25 (03), 287–318.

Karpathy, A., Johnson, J., Fei-Fei, L., 2015. Visualizing and understanding recurrent networks. arXiv preprint arXiv:1506.02078.

Lee, K., Eo, M., Jung, E., Yoon, Y., Rhee, W., 2021. Short-term traffic prediction with deep neural networks: A survey. IEEE Access 9, 54739–54756.

LeVeque, R.J., 2002. Finite Volume Methods for Hyperbolic Problems, volume 31. Cambridge University Press.

Li, R., Rose, G., 2011. Incorporating uncertainty into short-term travel time predictions. Transp. Res. C 19 (6), 1006–1018.

Lighthill, M.J., Whitham, G.B., 1955. On kinematic waves II. A theory of traffic flow on long crowded roads. Proc. R. Soc. A 229 (1178), 317–345.
Lingras, P., Mountford, P., 2001. Time delay neural networks designed using genetic algorithms for short term inter-city traffic forecasting. In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. Springer, pp. 290–299.
Lipták, G., Pereira, M., Kulcsár, B., Kovács, M., Szederkényi, G., 2021. Traffic reaction model. arXiv preprint arXiv:2101.10190.
Liu, J., Barreau, M., Čičić, M., Johansson, K.H., 2021. Learning-based traffic state reconstruction using probe vehicles. IFAC-PapersOnLine 54 (2), 87–92.
Long, Z., Lu, Y., Dong, B., 2019. PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network. J. Comput. Phys. 399, 108925.
Long, Z., Lu, Y., Ma, X., Dong, B., 2018. PDE-Net: Learning PDEs from data. In: International Conference on Machine Learning. PMLR, pp. 3208–3216.
Ma, X., Tao, Z., Wang, Y., Yu, H., Wang, Y., 2015. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. Transp. Res. C 54, 187–197.
Pakravan, S., Mistani, P.A., Aragon-Calvo, M.A., Gibou, F., 2021. Solving inverse-PDE problems with physics-aware neural networks. J. Comput. Phys. 440, 110414.
Pan, Z., Liang, Y., Wang, W., Yu, Y., Zheng, Y., Zhang, J., 2019. Urban traffic prediction from spatio-temporal data using deep meta learning. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1720–1730.
Pereira, M., Boyraz Baykas, P., Kulcsár, B., Lang, A., 2021. Parameter and density estimation from real-world traffic data: A kinetic compartmental approach. arXiv preprint arXiv:2101.11485.
Raissi, M., Perdikaris, P., Karniadakis, G.E., 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. 378, 686–707.
Rempe, F., Loder, A., Bogenberger, K., 2021. Estimating motorway traffic states with data fusion and physics-informed deep learning. In: 2021 IEEE International Intelligent Transportation Systems Conference. ITSC, IEEE, pp. 2208–2214.
Richards, P.I., 1956. Shock waves on the highway. Oper. Res. 4 (1), 42–51.
Ruthotto, L., Haber, E., 2019. Deep neural networks motivated by partial differential equations. J. Math. Imaging Vision 1–13.
Sherstinsky, A., 2020. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D 404, 132306.
Shi, R., Mo, Z., Huang, K., Di, X., Du, Q., 2021. A physics-informed deep learning paradigm for traffic state and fundamental diagram estimation. IEEE Trans. Intell. Transp. Syst..
Sutskever, I., 2013. Training Recurrent Neural Networks (Ph.D. thesis). University of Toronto Toronto, Canada.
Vlahogianni, E.I., Karlaftis, M.G., 2011. Temporal aggregation in traffic data: Implications for statistical characteristics and model choice. Transp. Lett. 3 (1), 37–49.
Vlahogianni, E.I., Karlaftis, M.G., Golias, J.C., 2014. Short-term traffic forecasting: Where we are and where we're going. Transp. Res. C 43, 3–19.
Wang, Z., Xing, W., Kirby, R., Zhe, S., 2022. Physics informed deep kernel learning. In: International Conference on Artificial Intelligence and Statistics. PMLR, pp. 1206–1218.
Wondimagegnehu, M.B., Alemu, G., 2017. Short time traffic flow identification and prediction at road junctions using recurrent multilayer perceptron. In: 2017 IEEE AFRICON. IEEE, pp. 15–20.
Xu, K., Darve, E., 2019. The neural network approach to inverse problems in differential equations. arXiv preprint arXiv:1901.07758.
Xu, J., Rahmatizadeh, R., Bölöni, L., Turgut, D., 2017. Real-time prediction of taxi demand using recurrent neural networks. IEEE Trans. Intell. Transp. Syst. 19 (8), 2572–2581.
Yin, X., Wu, G., Wei, J., Shen, Y., Qi, H., Yin, B., 2020. A comprehensive survey on traffic prediction. arXiv preprint arXiv:2004.08555.
Yu, B., Yin, H., Zhu, Z., 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. arXiv preprint arXiv:1709.04875.
Yuan, Y., Wang, Q., Yang, X.T., 2021a. Traffic flow modeling with gradual physics regularized learning. IEEE Trans. Intell. Transp. Syst..
Yuan, Y., Zhang, Z., Yang, X.T., Zhe, S., 2021b. Macroscopic traffic flow modeling with physics regularized Gaussian process: A new insight into machine learning applications in transportation. Transp. Res. B 146, 88–110.
Zhang, J., Wang, F.-Y., Wang, K., Lin, W.-H., Xu, X., Chen, C., 2011. Data-driven intelligent transportation systems: A survey. IEEE Trans. Intell. Transp. Syst. 12 (4), 1624–1639.
Zhong, M., Sharma, S., Lingras, P., 2005. Short-term traffic prediction on different types of roads with genetically designed regression and time delay neural network models. J. Comput. Civ. Eng. 19 (1), 94–103.
Zhu, L., Yu, F.R., Wang, Y., Ning, B., Tang, T., 2018. Big data analytics in intelligent transportation systems: A survey. IEEE Trans. Intell. Transp. Syst. 20 (1), 383–398.