

# Inverting the Fundamental Diagram and Forecasting Boundary Conditions: How Machine Learning Can Improve Macroscopic Models for Traffic Flow

Maya Briani<sup>a</sup>, Emiliano Cristiani<sup>a</sup>, and Elia Onofri<sup>a,b</sup>

<sup>a</sup>Istituto per le Applicazioni del Calcolo, Consiglio Nazionale delle Ricerche, Rome, Italy

<sup>b</sup>Dipartimento di Matematica e Fisica, Università degli Studi Roma Tre, Rome, Italy

March 23, 2023

## Abstract

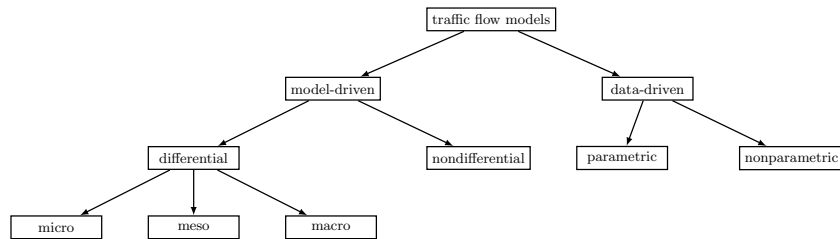
In this paper, we aim at developing new methods to join machine learning techniques and macroscopic differential models for vehicular traffic estimation and forecast. It is well known that data-driven and model-driven approaches have (sometimes complementary) advantages and drawbacks. We consider here a dataset with flux and velocity data of vehicles moving on a highway, collected by fixed sensors and classified by lane and by class of vehicle. By means of a machine learning model based on an LSTM recursive neural network, we extrapolate two important pieces of information: 1) if congestion is appearing under the sensor, and 2) the total amount of vehicles which is going to pass under the sensor in the next future (30 min). These pieces of information are then used to improve the accuracy of an LWR-based first-order multi-class model describing the dynamics of traffic flow between sensors. The first piece of information is used to invert the (concave) fundamental diagram, thus recovering the density of vehicles from the flux data, and then inject directly the density datum in the model. This allows one to better approximate the dynamics between sensors, especially if an accident happens in a not monitored stretch of the road. The second piece of information is used instead as boundary conditions for the equations underlying the traffic model, to better reconstruct the total amount of vehicles on the road at any future time. Some examples motivated by real scenarios will be discussed. Real data are provided by the Italian motorway company Autovie Venete S.p.A.

**Keywords.** traffic; vehicles; fundamental diagram; LWR model; machine learning; LSTM.

**MSC-2020.** 76A30; 68T07.

# 1 Introduction

Traffic state estimation (TSE) and traffic forecast have a long and solid tradition which dates back to the 1950s. A very broad division of the research lines on vehicular traffic flow modeling is summarized in the following diagram:



Model-driven and data-driven approaches have their own advantages and drawbacks, which were well described in, e.g., [43, 52]: Model-driven approaches allow to inject in the simulator the human knowledge of the system, at least if it can be reasonably translated into equations. Differences between agents can be (stochastically) taken into account as well, including drivers' psychological aspects. Differential macroscopic models, in particular, can unveil the power of methodologies based on partial differential equations (PDEs), for example giving the right tools to compute the *Wardrop equilibrium* of a traffic system on a road network [50, 8, 13]. On the other hand, this approach tends to be an over-simplification of traffic physics since the model is never able to catch *all* the features of cars and drivers. Most models are difficult to work with noisy and fluctuated data collected by traffic sensors and the calibration of the numerous parameters is quite challenging. In addition, the numerical scheme used for the discretization of the equations (Godunov, Lax-Friedrichs, etc.), actually needed to solve them, introducing a further, often not negligible, approximation error. Finally, models require additional inputs which are not available in real scenarios, such as, e.g., boundary conditions at any future time.

Data-driven approaches, instead, are more suitable to deal with the nonlinearities which characterize traffic flow and, for this reason, can be more accurate than model-driven approaches, but they are agnostic to the physics of traffic flow and could lead to infeasible estimation results. These methods are also not often interpretable and lack robustness. More importantly, the generalizability of the models is often weak and they have a high dependence on the training data samples. If the quality of training data is poor (missing/overestimated/underestimated data), their predictive accuracy will be severely weakened. Recently, many machine learning (ML) approaches were proposed: they often rely on relatively simple structures (compared to PDEs intricate systems) making them more lightweight from a computational point of view, hence being more suitable for real-time applications. However, dependency on large sets of historical data means that training can be computationally very expensive and can easily lead to data overfitting.

Recent research is growing in interest toward *hybrid approaches* which try

to get the best from each of the two approaches; however, they are rarer in the literature and each paper uses only single aspects from the two methodologies to obtain very different results. This paper tries to advance in this research field by proposing a computational method where ML techniques extrapolate from data the information needed by differential macroscopic models for traffic flow.

## Relevant literature

**Fundamental diagram** First of all, we have to mention the *fundamental diagram*, which is one of the basic ingredients of all model-driven approaches, especially at the macroscopic scale. It defines the relationship between the flux and the density of vehicles [26, 19], see Figure 1. It is plain that the

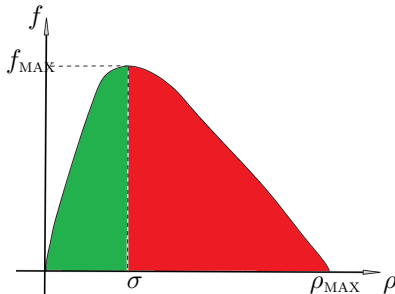


Figure 1: Fundamental diagram  $f = f(\rho)$ . The green part corresponds to the free phase while the red part corresponds to the congested phase. Density  $\sigma$  corresponds to the maximal flux  $f_{\text{MAX}}$  (road capacity). Flux is null when the road is empty ( $\rho = 0$ ) and when the road is fully congested ( $\rho = \rho_{\text{MAX}}$ ) and vehicles are stopped.

flux of vehicles is null in either the case of an empty road (null density) or in the case of a fully congested road (maximal density, stopped bumper-to-bumper vehicles). For intermediate density levels, real data show more complex dynamics, especially in correspondence to the maximal capacity of the road. Indeed, drivers act differently in response to the same traffic conditions and, in addition, accelerations and decelerations are far from being instantaneous. As a consequence, traffic shows some instabilities [6, 35, 46]. In first-order traffic models, the fundamental diagram can be defined by means of a single function while second-order traffic models allow the fundamental diagram to be multivalued, in the sense that a single value of the density can be associated with many values of the flux, exactly as it happens in reality. Recently, an interesting ML-based method to estimate the fundamental diagram was proposed in [41].

**Model-driven** Mathematical models for traffic flow first appeared in 1955-6 with the seminal papers [29, 37], which introduced the well-known LWR model. It is a first-order (velocity-based) model constituted by a hyperbolic PDE where

the observed quantity is the vehicle density  $\rho$ . The model stems from the reasonable analogy between vehicular dynamics and fluid dynamics. Following the same line, the model was successively extended to the second-order to include inertia (acceleration) effects in [36, 51] and [1, 57], giving birth to the PW and ARZ models, respectively. Independently, in the engineering literature, the CTM model [14] and the METANET model [32] were introduced. These two last models are equivalent to the discretized versions of the LWR and PW model, respectively [49]. The literature about mathematical models is huge, we refer the reader to the surveys [49, 45, 22, 3, 39] for differential models and their calibration, and [30] for nondifferential models (cellular automata).

Some effort was also devoted to generalizing mathematical models to road networks. This is not a trivial task due to the junction conditions which must be added to assure the uniqueness of the solution of the resulting system of PDEs. We refer the reader to [5, 4], and the book [20] for basic concepts.

Another generalization of our interest is that of multi-class models. In this case more than one class of vehicles (e.g., cars and trucks) share the same road. Each class has a specific dynamic and the classes interact with each other in a nontrivial manner. We refer to [17, 44, 38], and to the recent books [19, 26] for an overview of the most used multi-class models.

**Data-driven** Traffic data can be collected by means of several methods and technologies. Commonly one can have Eulerian data, provided by fixed sensors placed along the road (which count passing vehicles), and Lagrangian data, provided by probe vehicles equipped, e.g., with a GPS system. We refer to [9, 10, 53] for an overview on traffic data. Typical objects under analysis are, hence, flux  $f$  and velocity  $v$ , as opposed to the more abstract concept of density  $\rho$  that characterizes the model-driven approaches.

Early works in data-driven mainly rely on statistical approaches based on historical data. More recently, the broad research carried out in ML furnished new lymph to data-driven approaches, with a large corpus of research exploiting from the simplest to the more exotic technique to study TSE problems.

In this paper, we are mostly interested in Artificial Neural Network (ANN) methods for traffic understanding, estimation, and prediction, see [43, 16, 34, 54, 58, 60]. ANNs mainly divide into two families: feed-forward models, like the well-known Single- or Multi-Layer Perceptron (S/MLP) and feed-back models, like Recursive Neural Networks (RNNs). We refer the reader to the recent surveys [27, 48, 18, 31, 42] on this topic and, more specifically, we focus on the use of Long Short-Term Memory RNN (LSTM-RNN, or LSTM in short) for traffic data forecast. LSTM is a powerful tool that proved to be capable of capturing long-range temporal feature dependencies and reducing gradient explosion/vanishing. Among the recent literature, LSTM were often used (both vanilla or as a building block of more complex structures) to perform analysis and prediction on  $f$  and  $v$ : for what concerns velocity, it is the case, e.g., of [21] that mounts a fusion deep learning approach to predict lane-level traffic speed at two minutes, [24] that considers the correlation between car speed and car

type for a prediction model (LSTM + 4 layer MLP) of the highway speed at 5 minutes, and [52] that combines LSTM with a careful data preprocessing aided with wavelets analysis to perform speed prediction at 15 minutes. For what concerns flux, it is the case, e.g., of [59] where a temporal-spatial correlation is integrated into a 2D LSTM network to predict traffic flow at 15 minutes, [47] which integrates weather data with an attention model to perform short-term prediction of the traffic volume, and [18] which describes a novel methodology of an LSTM-based attention model to predict the upcoming flux based on 120 minutes of data (aggregated 10 minutes by 10 minutes).

**Hybrid methods** To overcome the shortcomings of both model- and data-driven approaches while exploiting their potentialities, recent studies introduced coupled methods where physics and data play together. The way the coupling is performed in the literature is very different since a common line is yet to be established: the physical model can be (i) injected in the training process of the ANN, obtaining the so-called Physics-Informed Neural Networks (PINNs), (ii) used in parallel with the ANN, as it is done, e.g., in [41], where the TSE, the model parameter identification, and estimation of the fundamental diagram are performed simultaneously, or (iii) used *after* the ANN, like in the present work and in [23], where data are used to provide consistent boundary conditions at junctions for macroscopic traffic flow models.

The majority of the recent works fall back into the PINNs category, where physics is usually plugged into the model by building a custom cost function, in particular, trying to exploit the powerfulness of deep learning models (PIDL); it is the case, e.g., of [40] where authors focus on highway TSE with observed data from loop detectors and probe vehicles, by building a coupled model with a Physic-Uninformed Neural Network (PUNN) and a PINN with custom loss function based on physical discrepancy measures. Paper [25], analogously, builds a custom loss function relying on CTM and LWR with different fundamental diagrams (Greenshields’, Daganzo’s, and inverse-lambda) to tackle the problem of data sparsity and sensor noise. Another example of custom cost function based on multiple physical aspects is given in [2], where authors perform TSE from probe vehicles data in an urban environment by building a 6-component loss function that is used both to reconstruct the road state and a smoothed version of the probe trajectories. Also worth mentioning is the paper [33], where authors introduce PIDL car-following model architectures encoded with different popular physics-based models to predict the evolution of each vehicle’s velocity. Finally, it is also interesting the approach based on physics regularized Gaussian process (PRGP), like the one proposed in [56] (later extended in [55]) where a stochastic PRGP is developed and a Bayesian inference algorithm is used to estimate the mean and kernel of the PRGP itself.

## Paper contribution

In this paper, we deal with traffic data coming from a series of fixed sensors placed along a highway. Sensors are about 1-to-20 km away from each other

and are able to count vehicles passing under them, estimate vehicles’ velocity, and classify vehicles in terms of their length (dividing them, e.g., between light and heavy vehicles). Sensor data are aggregated every minute, sent to a central processing unit, and stored in a database. The velocity datum corresponds to the mean of the velocities of vehicles of each type observed in the time interval of 1 minute.

The main goal is to estimate the traffic conditions *all along the road* at *current* and *future* time – in terms of macroscopic quantities like flux  $f$ , density  $\rho$ , and velocity  $v$  – by means of a first-order multi-class LWR-like macroscopic differential model which describes the joint dynamics of light and heavy traffic, already introduced in [6]. More in detail, we consider 2 steps:

**Nowcast.** This is the traffic estimation at current time  $t_0$  (now), at every point of the road. To do that, we split the road into several consecutive segments, each of which starts and ends with a sensor. Then, we run the model setting the initial time  $t = t_0 - \Delta t_{\text{PAST}}$  and the final time  $t = t_0$ , where  $\Delta t_{\text{PAST}}$  is a parameter. The model runs in each segment independently and gives the evolution of the density there. At time  $t_0 - \Delta t_{\text{PAST}}$  we assume that the road is empty, then the road starts filling thanks to the sensor data which act as inflow and outflow boundary conditions. If  $\Delta t_{\text{PAST}}$  is sufficiently large, the road fills completely and a reliable density estimate is computed along each segment.

The problem arises how to employ sensor data to enforce boundary conditions: mathematical models typically require *density data* as Dirichlet boundary conditions, but in our case sensors do not provide this information. Alternatively, one can inject the *flux data* directly into the numerical scheme chosen for the discretization of the modeling equations. Unfortunately, this solution is not always feasible because sensor data are not guaranteed to be compatible with the solution of the numerical model; moreover, the solution is not always the “correct” one, especially in the case of congestion events appearing between sensors, see Section 6 for details. This is the reason why we explore a third approach: we train an ANN based on an LSTM to detect congestion formation at sensors in real time. This is an interesting and complex problem *per se*, which gives, as a by-product, a tool for *inverting the concave fundamental diagram without ambiguity*, being able to distinguish the free phase from the congested phase, see Figure 1. The tool is then used to transform the flux datum into a density value, and inject it into the model, thus solving the incompatibility issues mentioned above and also adding new physical information in the model, leading, in most of the cases, to a more accurate solution.

**Forecast.** This is the traffic estimation at any future time  $t_0 < t < t_0 + \Delta t_{\text{FUT}}$ , where  $\Delta t_{\text{FUT}}$  is the duration of the simulation (30 min, in our case). In this case, sensors data are not yet available, hence we forget the sensors and we consider the whole highway as a unique long segment. We employ the same mathematical model considered before, using the nowcast traffic estimation as initial conditions for the density.

Since the model needs the boundary conditions for any time  $t_0 < t < t_0 + \Delta t_{\text{FUT}}$ , i.e. it needs to have an estimation of the number of vehicles which will

enter and leave the road until time  $t_0 + \Delta t_{\text{FUT}}$ , the problem arises how to enforce these boundary conditions, which sensors clearly cannot provide at time  $t$ . To do that, we set up a different LSTM-based ANN to predict sensor data in the time interval  $[t_0, t_0 + \Delta t_{\text{FUT}}]$ . More precisely, the output of the ANN will not be the minute-by-minute flux data, since they are too fluctuating to guarantee a reliable prediction; instead, we opt to predict the *total* number of vehicles in the time interval  $[t_0, t_0 + \Delta t_{\text{FUT}}]$ : a simpler yet useful piece of information since it can be interpreted as a constant boundary condition that, in most of the cases, offer a good accuracy regarding the total mass found along the whole road at  $t = t_0 + \Delta t_{\text{FUT}}$ .

## Paper organization

The rest of the paper is organized as follows. Section 2 introduces the traffic flow data we considered and provides an overview of the benchmark dataset provided by Autovie Venete S.p.A.

Section 3 introduces in general terms the structure of the ANN which will enrich the dataset.

Section 4 discusses the training and the validation of the network which provides information about the real-time detection of congestion events. The same section also discusses short-term (few minutes) forecast of congestion events.

Section 5 discusses the training and the validation of the network which provides information about a mid-term (30 minutes) forecast of the expected traffic volume at sensors.

Section 6 is devoted to the link between the enriched dataset obtained by using the ANNs previously described and the mathematical model, in the cases of both nowcast and forecast.

Finally, Section 7 concludes the work with some final remarks.

## 2 Discussing the data

At a macroscopic level, traffic flow is characterized by three state variables: the *flux*  $f$ , corresponding to the number of vehicles passing through a given point per unit of time, the *density*  $\rho$ , corresponding to the number of vehicles per unit of length, and the *velocity*  $v$ . In the mathematical world, these quantities are functions defined on a continuous space-time domain and the following relation holds true

$$f(x, t) = \rho(x, t) v(x, t) \tag{1}$$

for any point  $x$  of the road and time  $t$ . In the real world, however, these pieces of information can only be measured as discretized samples, therefore they cannot be calculated exactly at the same time. It is also well known that computing  $\rho$  by inverting (1) often leads to bad results, especially for large values of  $\rho$ , cf. [23].

Our benchmark dataset is provided by the Italian motorway company Autovie Venete S.p.A. and it was collected between September 2020 and March



Figure 2: The Italian motorway A4 Trieste–Venice and its branches to/from Udine, Pordenone, and Gorizia, managed by Autovie Venete S.p.A.

2022. It contains traffic data from fixed sensors located along three highways in the North-East of Italy, namely A23, A28, and (part of the) A4, see Figure 2. All of the highways have two lanes per direction except for the A4 which has three lanes in some parts. As usual, vehicles have different speed limits on the basis of their length and weight, moreover, heavy vehicles cannot use the fastest lane.

Data are collected by 45 groups of sensors which provide, in total 301,200 records per day on average. Each group is characterized by a position  $x$  along the road and a direction of travel and consists of *1 sensor per lane* (therefore we have 2 or 3 sensors per group). Each sensor counts every vehicle that passes in front of it, along with its speed, and classifies it according to the German TLS 5+1 class standard [7]. The claimed error on counting is  $\pm 3\%$  while the error on velocity is  $\pm \max\{3 \text{ km/h}, 3\%\}$ . In the following, we aggregate classes 1 and 2 as *light vehicles* (motorcycles, cars, vans, and car trailers) and classes 3, 4, and 5 as *heavy vehicles* (lorries, lorry trailers, tractor vehicles and buses). In Sections 4 and 5 we will further aggregate data by class and by group, respectively.

The spatial granularity is highly variable since the distance between sensors ranges from 1 to 20 km. The temporal granularity is instead more regular since data are transmitted by each sensor every 1 minute, as *aggregate measurements*: this means that the database stores the total number of vehicles passed in that time interval and the average velocity per each class. The measurements are kept for 2 hours for real-time analysis then they are moved to a separate database. Consequently, historicized data are not available in real time.

Some remarks are in order:

- As mentioned above, large density values cannot be recovered by flux and velocity data simply inverting the relation (1). The reason comes from a combination of the measurement aggregation and temporal granularity (low fluxes require long time intervals to be detected because vehicles move slowly).



- Although flux data show a regularity on a daily basis, they are very fluctuating from minute to minute, see Figure 3.
- Most importantly, our data cannot distinguish between an empty road and a fully congested road. In both cases the measured flux is 0 and the velocity is undefined.

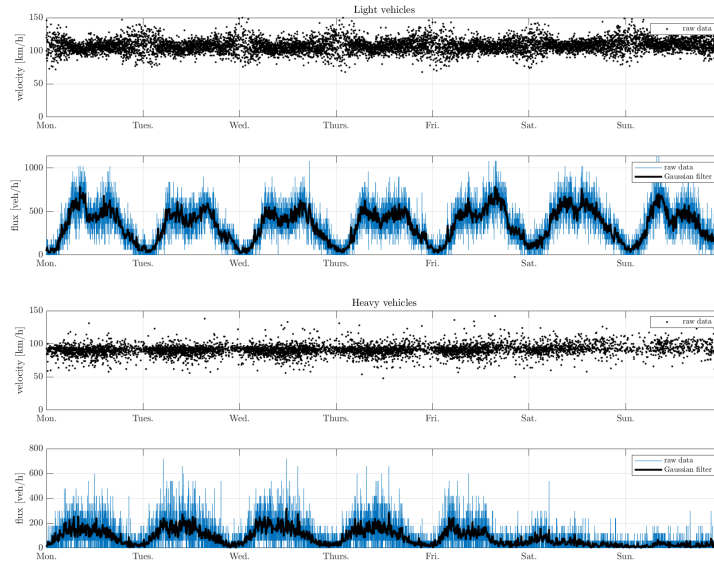


Figure 3: Velocity and flux for light (top) and heavy (bottom) vehicles, entire week from Monday to Sunday. We observe that the raw flux data are very fluctuating from minute to minute, but, applying a Gaussian filter (black line), one can recognize a certain pattern repeated on a daily basis. At night, the flux data of all vehicles drop, while the velocity data of light vehicles become more scattered. As expected, during the weekend, the flux of heavy vehicles is quite low.

These remarks are important to understand how it can be difficult to detect the formation of a congestion event in real time, which is, in turn, essential for a good estimation and forecast of the traffic flow, even far from the sensor which first observes the congestion. In order to better understand this point, we show in Figure 4 four congestion events which develop with different characteristics. In Figure 5, instead, we show two very similar traffic conditions characterized by a flux drop, which evolve in a totally different manner: one into the free regime and the other into the congested regime. This makes it clear that, despite it is relatively easy to detect congestion events (distinguishing them from empty road conditions) by observing data *a posteriori* (e.g., a whole day), it is very difficult to do the same *at the moment* of the congestion formation.

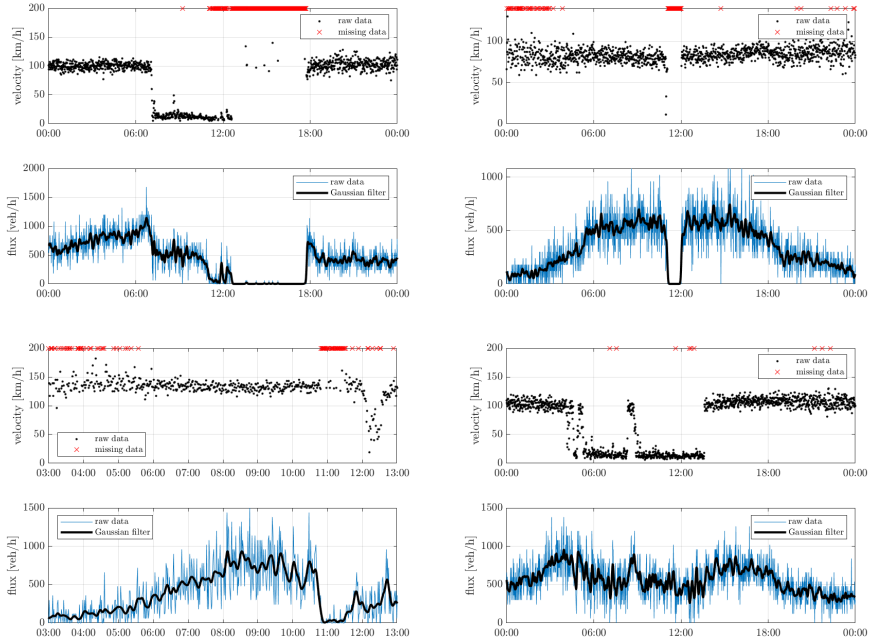


Figure 4: Four congestion events with different features: (top-left) we observe a rapid velocity drop and flux drop, then flux vanishes while velocity is undefined (with some exceptions for some fast vehicles still passing); (top-right) flux and velocity drop abruptly; (bottom-left) flux drops first, then velocity drops; (bottom-right) velocity drops while flux is only partially lowered.

For our purposes, it is crucial to note that 17 groups of sensors out of the 45 deployed are also equipped with advanced technologies<sup>1</sup> that, combining Doppler radar, ultrasound emitters, and passive infrared radiation detectors, are able to determine also stopped vehicles and intense congestion conditions, reporting them as Boolean flags in the corresponding minute. In the following, we will refer to these sensors as  $3T$  sensors and we will use them as one of the main sources of data for supervised learning purposes.

<sup>1</sup>Provided by Asim Technologies Ltd, series TT295.

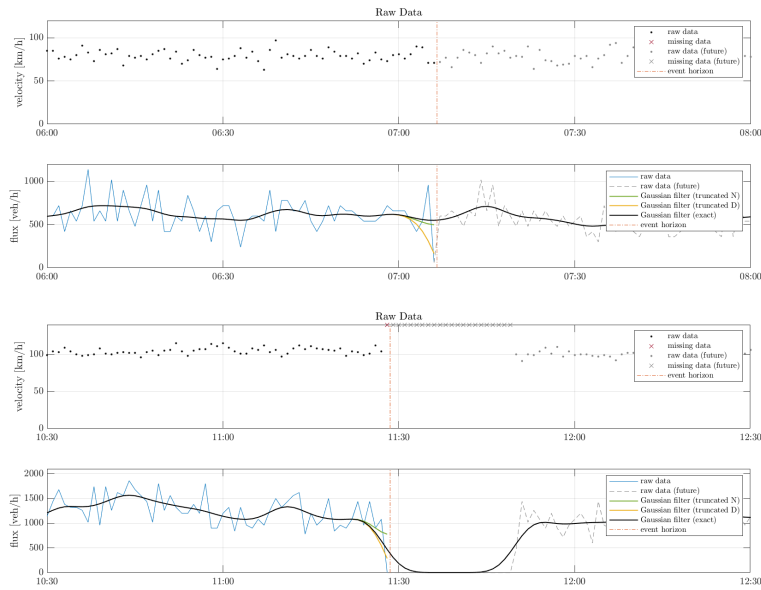


Figure 5: (top) Normal traffic conditions characterized by the usual high fluctuation of the flux. At 7:05 the flux drops abruptly then the traffic restarts normally; (bottom) At 11:29 a very similar situation appears but, this time, it evolves into a queue. Beside the Gaussian filter already shown in Figure 3, here we also show two other Gaussian filters obtained without using future data (beyond the event horizon). Truncation is obtained assuming either Dirichlet-like boundary conditions or Neumann-like boundary conditions. We see that neither Gaussian filters nor raw data are enough to distinguish between the two scenarios at the event horizon.

### 3 Supervised machine learning approach for the dataset

In this section, we set up a unique ML-based approach to solve both the two problems introduced in Section 1, namely the real-time detection of congestion events, and the forecasting of the expected average flux at sensors.

As we have already recalled, among the supervised learning techniques, RNN proved to be very effective in time series analysis. They are, in fact, a specific class of ANN devoted to analyzing temporal sequences  $\mathbf{X} = (\mathbf{x}_t)_{t=1,2,\dots}$  with a fixed number of features  $N_{\text{IN}}$  (namely  $|\mathbf{x}_t| = N_{\text{IN}}$ ), where the same computing unit is iteratively applied once per each step  $t$  of the sequence taking as input the  $N_{\text{IN}}$  features of the current step along with the output of the previous step. In our case, the features which can be extrapolated from the dataset are flux  $f$  and velocity  $v$  organized by class of vehicles and/or by lane.

In particular, LSTM-RNN is nowadays the preferred tool for many problems related to time series, being able to capture temporal features or dependencies also in long-range periods of time. The idea underlying the LSTM is to divide the output  $\mathbf{h}_t$  (or *hidden state*, in the LSTM jargon) of each step from the processed data that generates it, hence keeping a sort of “internal memory”  $\mathbf{c}_t$  (or *cell state*) of the LSTM itself. Both  $\mathbf{h}_t$  and  $\mathbf{c}_t$  shares the same dimension  $N_{\text{HID}}$  (namely  $|\mathbf{h}_t| = |\mathbf{c}_t| = N_{\text{HID}}$ ) which also represents the “length” of the memory (time window) the LSTM is capable to capture. In particular, the output of the LSTM always relies only on the last  $N_{\text{HID}}$  steps of the series under analysis: outputting the result after a number of steps smaller than  $N_{\text{HID}}$  provides a less accurate answers. Conversely to  $N_{\text{IN}}$ , such parameter must be tuned with suitable approaches (hyper-tuning). The cell state can be then updated depending on the (eventually normalized) novel input  $\mathbf{x}_t$  conditioned over the previous output  $\mathbf{h}_{t-1}$ .

In practice, the LSTM update is based on four different components  $\mathbf{g}_t^{\text{F}}$ ,  $\mathbf{g}_t^{\text{I}}$ ,  $\mathbf{g}_t^{\text{C}}$ , and  $\mathbf{g}_t^{\text{O}}$  (or *gates*, with  $|\mathbf{g}^{\cdot}| = N_{\text{HID}}$ ) that evaluate over independently weighted combinations of  $\mathbf{h}_{t-1}$  and  $\mathbf{x}_t$  as:

$$\begin{aligned}
 \mathbf{g}_t^{\text{F}} &= \text{sigmoid}\left(\mathbf{b}^{\text{F}} + \mathbf{W}^{\text{F}} \times \mathbf{x}_t + \mathbf{R}^{\text{F}} \times \mathbf{h}_{t-1}\right), \\
 \mathbf{g}_t^{\text{I}} &= \text{sigmoid}\left(\mathbf{b}^{\text{I}} + \mathbf{W}^{\text{I}} \times \mathbf{x}_t + \mathbf{R}^{\text{I}} \times \mathbf{h}_{t-1}\right), \\
 \mathbf{g}_t^{\text{C}} &= \tanh\left(\mathbf{b}^{\text{C}} + \mathbf{W}^{\text{C}} \times \mathbf{x}_t + \mathbf{R}^{\text{C}} \times \mathbf{h}_{t-1}\right), \\
 \mathbf{g}_t^{\text{O}} &= \text{sigmoid}\left(\mathbf{b}^{\text{O}} + \mathbf{W}^{\text{O}} \times \mathbf{x}_t + \mathbf{R}^{\text{O}} \times \mathbf{h}_{t-1}\right),
 \end{aligned} \tag{2}$$

where  $\times$  denotes the standard matrix-vector product, and  $\mathbf{W}^{\cdot}$ ,  $\mathbf{R}^{\cdot}$ , and  $\mathbf{b}^{\cdot}$  are called respectively *input weights*, *recurrent weights*, and *biases*. Such weights are (typically) randomly assigned in the beginning and are the objective of the training phase through the back-propagation of the error.

Each gate serves a different purpose, where the first three rule the update of the cell state  $\mathbf{c}$  and the fourth determine the next hidden state  $\mathbf{h}$  (regulating

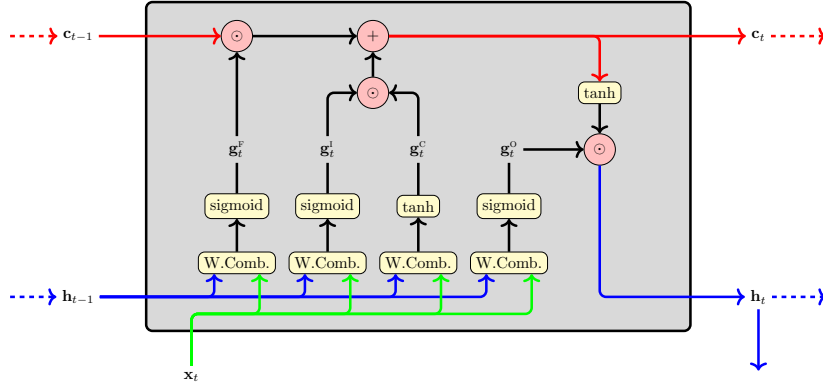


Figure 6: Schematic structure of the LSTM computing unit evaluating input at time  $t$ . W.Comb. represents the weighted combination with bias given as  $\mathbf{b} + \mathbf{W} \times \mathbf{x}_t + \mathbf{R} \times \mathbf{h}_{t-1}$ . All the components are vectors of length  $N_{\text{HID}}$  but  $\mathbf{x}$  which is of length  $N_{\text{IN}}$ . We recall that  $\mathbf{h}_t$  is also the output of the network.

the cell state contribution). More formally:

$\mathbf{g}^f$  or *forget gate*: weakens  $\mathbf{c}_{t-1}$  by applying a transformation within the range  $(0, 1)$ .

$\mathbf{g}^i$  or *input gate*: decides how the candidate influences  $\mathbf{c}_t$ , being a transformation of range  $(0, 1)$ .

$\mathbf{g}^c$  or *candidate gate*: represents the cell input activation that regulates  $c$ , being of range  $(-1, 1)$ .

$\mathbf{g}^o$  or *output gate*: decides how  $\mathbf{c}_t$  will compose the output, applying a transformation of range  $(0, 1)$ .

Hence, (see also Figure 6 for a pictorial representation) the update rules are given by

$$\mathbf{c}_t = \mathbf{g}_t^f \odot \mathbf{c}_{t-1} + \mathbf{g}_t^i \odot \mathbf{g}_t^c, \quad \mathbf{h}_t = \mathbf{g}_t^o \odot \tanh(\mathbf{c}_t), \quad (3)$$

where  $\odot$  denotes the Hadamard (element-wise) product.

In particular, it is important to notice that the output  $\mathbf{h}$  of the LSTM is a  $N_{\text{HID}}$  vector and, consequently, it needs to be manipulated to either give a prediction or a classification. Our tool of choice is a vanilla SLP Feed-Forward Network that condenses the  $N_{\text{HID}}$  features in

- *Prediction task*: a  $N_{\text{PRED}}$ -length output  $\mathbf{o}$ , where each entry represents an individual prediction.
- *Classification task*: a  $N_{\text{CLASS}}$ -length features-vector  $\mathbf{z}$ , where each entry represents a class of the problem, then fed into a softmax layer to transform

them into a probability vector  $\hat{\mathbf{o}}$  (of being of a specific class). We recall the softmax function is defined component-wise as follows:

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=0}^{N_{\text{CLASS}}-1} e^{z_j}}, \quad i = 0, \dots, N_{\text{CLASS}} - 1. \quad (4)$$

The final output can be either the probability vector  $\hat{\mathbf{o}}$  or the index of the highest-probability class  $o$ , i.e.  $o = \text{argmax}(\hat{\mathbf{o}})$ . Do note that  $\hat{\mathbf{o}}$  can be used as a *confidence* indicator of the prediction as the closer is  $\max(\hat{\mathbf{o}})$  to 1, the more certain is the prediction according to the ANN.

We recall that SLP is a regular weighted combination with bias, namely the output  $\mathbf{z}$  of length either  $N_{\text{PRED}}$  or  $N_{\text{CLASS}}$  is obtained as  $\mathbf{z}_t = \mathbf{b} + \mathbf{W} \times \mathbf{h}_t$ , where  $\mathbf{W}$  and  $\mathbf{b}$  are a suitable-sized weight-matrix and bias-vector.

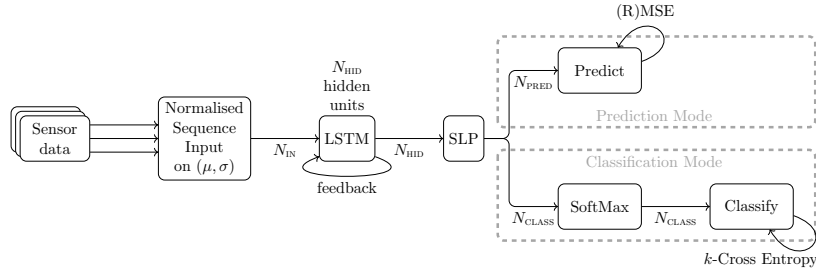


Figure 7: Pipeline of the enriching tool. The sensor data sequence is globally feature-wise normalized before feeding it in the LSTM, one step at a time. The output of the LSTM is then processed by an MLP Network having either a single or no hidden layer (actually resulting in a single convolution step). Finally, depending whether it is a classification task or not, a SoftMax layer is applied before the result is given.

To complete the pipeline of our methodology (a summary can be found in Figure 7), we further need to specify how to evaluate the system performance against our ground-truth data, i.e. we need to choose an error function to evaluate the distance of our guesses from ground truth (a loss function, in the jargon).

- *Prediction task*: being the prediction output a real number, since we are not interested in weighting differently over- and under-errors, we went for the vanilla (Root) Mean Squared Error. Denoting with  $\mathbf{y}_t$  and  $\mathbf{o}_t$  respectively the ground truth and the output of the network corresponding to the input  $\mathbf{x}_t$ , we recall the (root) mean squared error being:

$$\text{MSE} = \frac{1}{|\mathbf{X}|} \frac{1}{N_{\text{PRED}}} \sum_{t=1}^{|\mathbf{X}|} \|\mathbf{o}_t - \mathbf{y}_t\|^2, \quad \text{RMSE} = \sqrt{\text{MSE}}. \quad (5)$$

- *Classification task*: a very common choice for Boolean classification loss is the  $k$ -cross entropy (with  $k = 2$ ). In particular, since our dataset shows a great disequilibrium between positive and negative samples, we decide (instead of enriching it with synthetic data that are often complex to generate) to weight the loss entropy on the positive ratio  $0 < p_r < 1$ . We recall that the softmax applied to a binary classification task returns a 2-value vector  $\hat{\mathbf{o}}_t = (1 - \hat{o}_t, \hat{o}_t)$  where  $\hat{o}_t$  represents the probability that the positive class is chosen. Hence, if  $y_t$  is the ground-truth bit corresponding to input  $\mathbf{x}_t$  – meaning that our target  $\mathbf{y}_t$  is either  $(1, 0)$  or  $(0, 1)$  – then the weighted binary-cross entropy is defined as:

$$\text{CrossEntropy} = -\frac{1}{|\mathbf{X}|} \sum_{t=1}^{|\mathbf{X}|} \left( ((1-p_r) \cdot y_t \cdot \ln(\hat{o}_t)) + (p_r \cdot (1-y_t) \cdot \ln(1-\hat{o}_t)) \right). \quad (6)$$

## 4 Detection and short-term forecast of congestion

In this Section, we build a labeled dataset of congestion events and we use the previously described ANN-based methodology to build two classifiers: a first classifier  $\mathfrak{F}_c$  for performing real-time detection of congestion events, and a second classifier  $\mathfrak{F}_p$  to perform short-term forecasting of the same congestion events. The two classifiers act as a sort of “congestion alarm” and “congestion pre-alarm” launchers, respectively.

### 4.1 Building the dataset

The dataset is created using data gathered by the  $3T$  sensors. Since the congestion event is detected by each sensor (i.e., in each lane), we decided to aggregate the data of all classes of vehicles and working lane by lane. Therefore, the features are the total flux  $f$  and the average velocity  $v$  of all vehicles (all classes). (Actually, recovering *a posteriori* a piece of information per class is often quite easy. In fact, if a congestion event is detected in the slow lanes and not in the fast lane, it is highly probable that the congestion is for heavy vehicles only, since they cannot use the fast lane, see Section 2.)

We split the data per day, so to perform an incremental training phase using only randomized batches of days of data at a given time. Therefore, we get dataset samples constituted by a two-feature 1440-long ( $24 \text{ h} \times 60 \text{ min}$ ) sequence  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_{1440})$ , with  $\mathbf{x}_t = (f_t, v_t)$ ,  $t = 1, \dots, 1440$ . The dataset should be labeled with a suitable 1440-long Boolean array  $\mathbf{y} = (y_1, \dots, y_{1440})$  reporting whether the corresponding minute  $t$  is labeled as a congestion event or not. The array  $\mathbf{y}$  is computed by means of a logical combination of three different computational procedures, each of which leads to a Boolean flag. More precisely, we have.

$$\mathbf{y} = \mathbf{b}^{3T} \vee \mathbf{b}^f \vee \mathbf{b}^v,$$

where  $\vee$  denotes the logical *or* and:

- $\mathbf{b}^{3T}$  is the flag for the congestion event transmitted directly by the  $3T$  sensors. On average, the amount of daily congestion events per sensor is pretty low ( $< 0.1\%$ ), hence, in order to make the dataset more balanced, we selected the samples reporting at least  $\sim 1\%$  of positive labeling (i.e.  $\sum_{t=1}^{1440} (b_t^{3T} \geq 15)$ ). We noticed that the  $3T$  detection procedure is quite “conservative”, meaning that it tends to report a congestion event only if there is a stable queue under the sensor. Moreover, the exact definition of congestion event, as well as the physical and computational procedure used to detect it are not publicly available and they are not known by the authors. For these reasons, we decided to enrich the true-sample with the two following heuristics.
- $\mathbf{b}^f$  considers the flux array  $\mathbf{f} := (f_1, \dots, f_{1440})$  only, along with a 10-fold Gaussian regularization  $\mathbf{f}^*$  obtained via a convolution with a triangular kernel, cf. Figure 3; a congestion event is reported at time  $t$  if the sensor detects the following condition during the daytime (from 5 AM to 8 PM):

$$f_{t-1} < 2, \quad \wedge \quad f_t < 2, \quad \wedge \quad f_t^* < 2, \quad \wedge \quad \left( \frac{1}{60} \sum_{s=t-60}^{t-1} f_s \right) - f_t^* > 2, \quad (7)$$

where  $\wedge$  denotes the logical *and* operator. The idea behind this heuristic is that a sufficient condition for a congestion event is a low flux at the current time combined with a sudden drop of the flux (i.e., a high average flux in the previous time period).

- $\mathbf{b}^v$  considers instead the velocity array  $\mathbf{v} := (v_1, \dots, v_{1440})$  only, along with a regularization  $\mathbf{v}^*$  (obtained as before). A congestion event is reported at time  $t$  if the sensor detects the following conditions during the daytime (from 5 AM to 8 PM):

$$v_t < v_{t-1}, \quad \wedge \quad 0 < v_t < 65, \quad \wedge \quad \max\{v_s^* - v_t \mid s = t-1, \dots, t-15\} \geq 40. \quad (8)$$

The idea behind this heuristic is that a sufficient condition for a congestion event is a low velocity at the current time combined with a sudden drop of the velocity (i.e., a high velocity in the previous time period).

**Remark.** The two heuristics were developed under a collaboration with the data owner and have been human-validated through over a month of direct observations.

**Remark.** We point out that the two heuristics both use data that are not available yet at time  $t$  (because of the regularizations), making them useless to perform real-time applications.

Analyzing the average and the standard deviation of the flux data we found that most of the sensors dispatched on A4 behave very differently from the



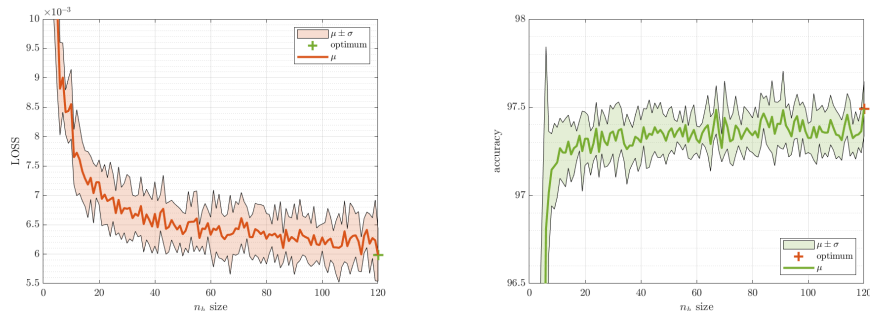


Figure 8:  $k$ -Cross Entropy (loss, on the left) and accuracy (on the right) achieved by the training sessions for  $\mathfrak{F}_c$  at the varying of  $N_{\text{HID}}$ . As it can be seen, the better parameter is  $N_{\text{HID}} = 120$  (i.e. corresponding to 2 hours) suggesting that having more data would refine the results even more.

others ( $\mu \sim 1e+4$  vs.  $2e+3$  vehicles/day and  $\sigma \sim 1e+4$  vs.  $5e+3$ ). This is not surprising since A4 connects more populated areas compared to A23 and A28. For this reason, we split the sensors into two disjoint sets, namely *high flux* (HF) and *low flux* (LF), being LF all the sensors in A23, A28, and in the fastest lane of A4 (where it has 3 lanes). In the rest of this and the following section we consider HF sensors only, the procedure and analysis for LF sensors being similar.

We split the HF 119-day dataset  $\{(\mathbf{X}^{(d)}, \mathbf{y}^{(d)})\}$ ,  $d = 1, \dots, 119$  (171,360 training minutes) in 96 training days and 23 validation days. The average sample positive rate after the enrichment is  $p_r = 4.2\%$ , hence we weighted the  $k$ -cross entropy function opportunely.

## 4.2 Training the model

We started by training the HF classifier  $\mathfrak{F}_c$  first: we performed multiple training sessions to estimate the suitable size  $N_{\text{HID}}$  of the memory of the LSTM. Each training session was carried out by using the ADAM optimizer working on randomly shuffled batches of size 24 samples over 100 epochs arranged with 9 progressive refinements of the learning rate (10 epochs per refinement).

Figure 8 reports the results in terms of loss and accuracy for the various training sessions. We found  $N_{\text{HID}} = 120$  being the most suitable parameters for the model (with  $N_{\text{IN}} = 2$ ,  $N_{\text{CLASS}} = 2$ ), yielding best accuracy of 97.70% (99.75% if weighted w.r.t.  $p_r$ ) with corresponding loss value  $5.2 \times 10^{-3}$ . We recall that  $N_{\text{HID}} = 120$  was the largest amount of data available in real-time applications for our framework, see Section 2.

Secondly, we tackled the problem of training the classifier  $\mathfrak{F}_p$ . We fixed the dimensions of the model – namely  $N_{\text{IN}} = 2$ ,  $N_{\text{HID}} = 120$ , and  $N_{\text{CLASS}} = 2$  – and we focused on how many minutes we could anticipate the classification of  $\mathfrak{F}_c$  without losing too much in accuracy. The trivial way would be producing a labeled

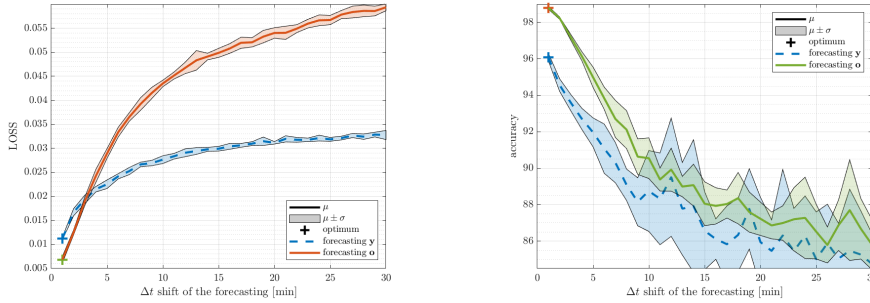


Figure 9:  $k$ -Cross Entropy (loss, on the left) and accuracy (on the right) achieved by the training sessions for  $\mathfrak{F}_p$  at the varying of the time window of the forecasting  $\Delta t$  ( $x$ -axis) both for predicting  $\mathbf{y}^{(d)}$  (in dashed blue) and  $\mathbf{o}^{(d)}$  (in orange/green). As expected, the smaller  $\Delta t$ , the better the prediction.

dataset for the forecasting task by applying a  $\Delta t$  minutes shift to the vectors  $\mathbf{y}^{(d)}$  in order to “bring backward” the congestion events. However, we prefer forecasting the output  $\mathbf{o}^{(d)}$  of  $\mathfrak{F}_c$  itself (i.e. the argmax of the probability vector) rather than the target  $\mathbf{y}^{(d)}$  of  $\mathfrak{F}_c$ . This change in objective is in fact convenient for two main reasons: (i) the output  $\mathbf{o}^{(d)}$  is somehow a “regularized” version of the real target  $\mathbf{y}^{(d)}$ , hence it should be easier to forecast; (ii) philosophically speaking, we are expecting that a  $\mathfrak{F}_p$  alarm to be followed by a  $\mathfrak{F}_c$  one. In order to further correct the dataset, however, we also filtered the novel target vector from all the isolated congestion events, actually filtering the ones that probably were false-positives of  $\mathfrak{F}_c$ .

The results of the training with the two proposed datasets are reported in Figure 9, where loss and accuracy are compared at the varying of the  $\Delta t$  shift. Analyzing the accuracy, it can be seen that trying to forecast  $\mathbf{o}^{(d)}$  is in general simpler; the loss, on the contrary, grows slower when the target is set to  $\mathbf{y}^{(d)}$  actually prompting that training on  $\mathbf{o}^{(d)}$  is less conservative in terms of false-positive. We found  $\Delta t = 4$  min being the most suitable parameter (in particular with the second dataset, still having comparable loss) yielding interesting predictions while keeping the average accuracy above 95% (with the best peak of 96.50%, or 98.82% if weighted) with a corresponding average loss value of 0.24, hence comparable to results obtained by  $\mathfrak{F}_c$ .

### 4.3 Performance evaluation of the real-time classifier $\mathfrak{F}_c$

Figure 10 shows a few examples of day-length classification of congestion events for both LF and HF sensors. We report the velocity  $v$ , flux  $f$ , and the confidence of the classification (given as  $\hat{o}$  rescaled in the interval  $[-1, +1]$ , i.e.  $\hat{o} \cdot 2 - 1$ ). We can notice that when the drop is significant only in  $v$  but not in  $f$ , the model shows a little uncertainty to classify the event (Figure 10(top-left)); however, if the drops involve also  $f$ , then the model is able to classify the congestion even if

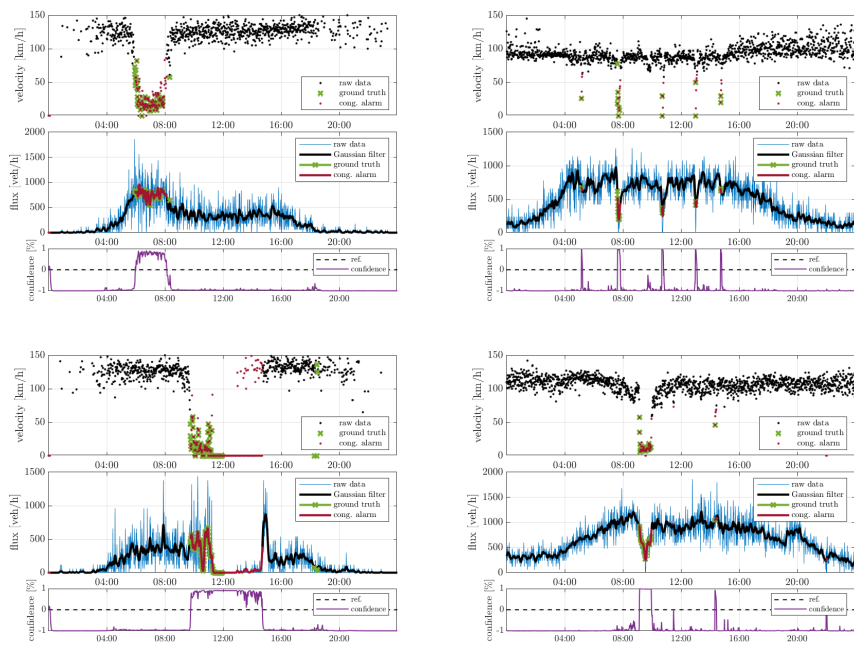


Figure 10: Day-length classification of congestion events. Each plot reports the velocity (top), the flux (middle) and the confidence of the model for the classification (bottom). Left: LF sensors; Right: HF sensors.

they are short in time (Figure 10(top-right)). In particular circumstances, the model is able to classify critical events also when the ground truth is uncertain, like when the flux is close to zero and velocity is high (Figure 10(bottom-left)). Note that this is a typical situation at night, with no congestion; finally, the model also keeps the congestion alarm on when the critical situation is vanishing: this comes from the fact that the model does not know the future, conversely to the regularization applied by the heuristics (Figure 10(bottom-right)).

#### 4.4 Performance evaluation of the short-term prediction classifier $\mathfrak{F}_p$

In order to give an idea about the actual possibility to predict congestion events, in Figure 11 we show four cases we have commonly observed. Figure 11(top-left) shows a very favorable case in which we are able to predict the congestion 4 minutes before the actual formation. This is possible because the velocity starts lowering a bit before dropping down, together with the flux. Figure 11(top-right) shows a case where a long-standing congestion is preceded by a confusing scenario in which the congestion alarm is on and off. In this case, the congestion pre-alarm is constantly on. Figure 11(bottom-left) shows a case where prediction is quite hard, even for a trained human: congestion begins abruptly with a velocity drop, and the ANN is able to predict it only 1 minute in advance. Finally, Figure 11(bottom-right) shows an uncertain situation which can likely evolve into a queue (but it does not, at least for some minutes). The pre-alarm is on, and even if this is formally incorrect (the congestion pre-alarm is not followed by a congestion alarm) the uncertainty fully justifies the ANN behavior.

## 5 Computation of expected traffic volume

In this section, we tackle the problem of performing mid-term prediction (30 min) of the expected traffic volume by building an ANN-based predictor  $\mathfrak{P}$ . Let us recall that, unlike in the previous section, here we aggregate sensors belonging to the same group (following the definition given in Section 2), while working on the two distinct, but coupled, classes of vehicles (light and heavy). This is important because the dynamics of the two classes are very different one from the other and, at the same time, they are strongly interconnected.

### 5.1 Building the datasets

Ground-truth data  $\mathbf{y}$  can be easily produced for each (group of) sensor, only needing the real flux of light vehicles  $\mathbf{f}^L$  and that of heavy vehicles  $\mathbf{f}^H$  for the  $T$  upcoming minutes. These compose the feature vector  $\mathbf{x}$  as  $(\mathbf{f}^L, \mathbf{f}^H)$  and the corresponding ground truth  $\mathbf{y}$  as:

$$y_t = \left( \frac{1}{T} \sum_{s=t+1}^{t+T} f_s^L, \frac{1}{T} \sum_{s=t+1}^{t+T} f_s^H \right). \quad (9)$$

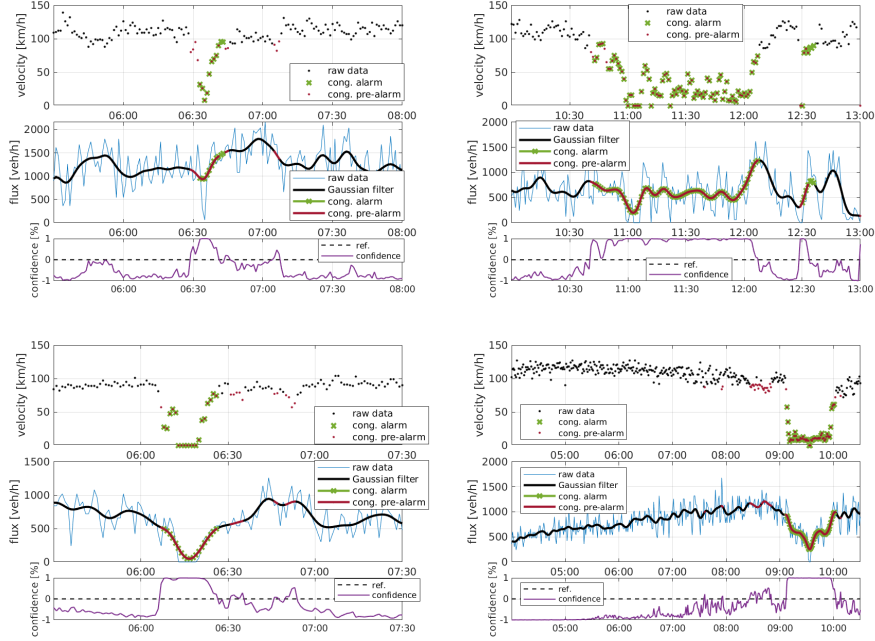


Figure 11: Top-left: at 6:30 a congestion event is correctly pre-alarmed four minutes in advance. Top-right: at 10:45 a congestion event begins intermittently while a solid pre-alarm shows up. Bottom-left: a difficult-to-predict congestion appears at 6:06. Bottom-right: at 8:30 an uncertain situation shows up, characterized by below-average velocity and normal flux, then after 9:00 a clear congestion is formed (pre-alarmed 1 minute in advance).

For what concerns the dataset creation, we selected the data collected during the entire 2021, corresponding to 348 day-wise readings (17 days are missing due to reported reading problems). Due to the high number of available data, we decided to tune a predictor per group of sensors. As for the previous training, we organized the features in day-length sequences, and we built a dataset  $D^{(g)} = \{\mathbf{x}^{(d)}; \mathbf{y}^{(d)}\}$ ,  $1 \leq d \leq 348$ , for each group  $g$ . We extracted roughly one month of samples from the dataset to form the test set while we used the remaining part as the training set.

## 5.2 Training the model

We trained a different model  $\mathfrak{P}^{(g)}$  per group of sensors over its corresponding dataset  $D^{(g)}$ . In particular, we focus on the four groups representing the inflow boundaries of the highway network under consideration: Venice ( $g_1$ ) and Trieste ( $g_2$ ) for the A4, Conegliano ( $g_3$ ) for the A28, and Udine ( $g_4$ ) for the A23, see Figure 2.  $g_1$  is the only group deployed on a three-lane section.

By adopting the same nomenclature as from Figure 7, the choice of the input and output features requires  $N_{\text{IN}} = 2$  and  $N_{\text{PRED}} = 2$ , while  $N_{\text{HID}}$  is a free parameter. Hence, we performed multiple training sessions to estimate the suitable size for the memory  $N_{\text{HID}}$  of the LSTM. Each training session was carried out with the ADAM optimizer, working with a variable learning rate, piece-wise decreasing over 5 progressive learning eras of 50 epochs each (for a total of 250 epochs).

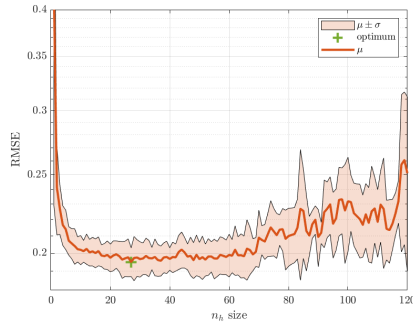


Figure 12: Value of the RMSE, corresponding to the training of  $\mathfrak{P}$ , as a function of the size  $N_{\text{HID}}$ .

Figure 12 shows the value of the RMSE as a function of the free parameter  $N_{\text{HID}}$ . We found  $N_{\text{HID}} = 27$  being the best parameter to correctly capture the trend of the average flux; the average resulting error and the corresponding statistics are reported in Table 1. Note that the RMSE reported in Figure 12 is evaluated on the normalized dataset, hence it is not comparable with results from Table 1.

	Light vehicles		Heavy vehicles	
	2-lane	3-lane	2-lane	3-lane
Mean average-error	42.74	76.60	11.74	25.06
Max average-error	70.80	95.06	19.48	33.44
Mean standard-deviation	41.72	62.96	12.04	23.94
Mean max-error	422.64	744.40	164.46	251.24
Mean average-flux	387.74	768.16	96.52	281.42
Mean max-flux	5320.64	5864.48	1825.30	2047.10

Table 1: Error statistics in [vehicles/h] achieved by multiple trainings (with  $N_{\text{HID}} = 27$ ) over the four datasets  $D^{(g_i)}$  (average is performed on all minutes available in the dataset). Values are sorted by vehicle class and number of lanes. Average and maximum flux are also reported as benchmark values for relative error statistics.

The question arises if it is really needed to train a different LSTM per group of sensors or if one ANN can serve all. To address this question we report in

Figure 13 the error made by  $\mathfrak{P}^{(g_i)}$  tested against datasets  $D^{(g_j)}$ , with  $i, j = 1, \dots, 4$ . We see that it is highly suggested to train as many ANN as groups

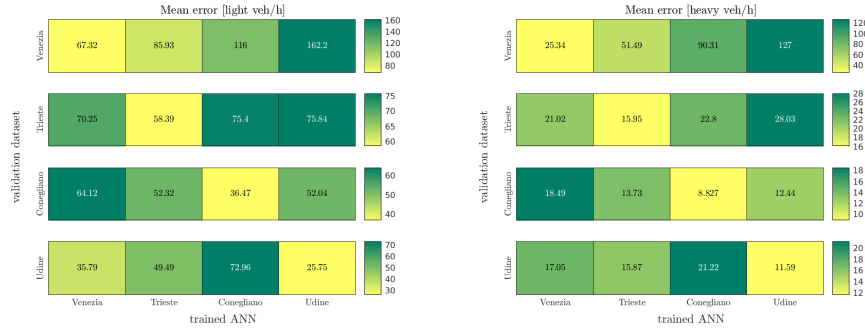


Figure 13: Mean error ( $|y_t - o_t|$ ) in [vehicles/h] (light vehicles on the left, heavy vehicles on the right) obtained by applying the network  $\mathfrak{P}^{(g_i)}$ ,  $i \in \{1, 2, 3, 4\}$  ( $x$ -axis) on the mutual test-sets  $D^{(g_j)}$ ,  $j \in \{1, 2, 3, 4\}$  ( $y$ -axis). As it can be seen, maximum-by-row is achieved on the main diagonal, meaning that applying a specific ANN on a different dataset is not convenient. It is worth noticing that maximum-by-row does not necessarily correspond to the maximum-by-column (see, e.g.,  $i = 1, j = 4$ ); this phenomenon is due to the high regularity of some datasets.

of sensors are considered.

### 5.3 Performance evaluation

We consider the group of sensors  $g_2$  as an example for the performance evaluation. Figure 14 shows the histograms of the errors (in [vehicles/h]) made running the prediction every minute available in the test set. Figure 15

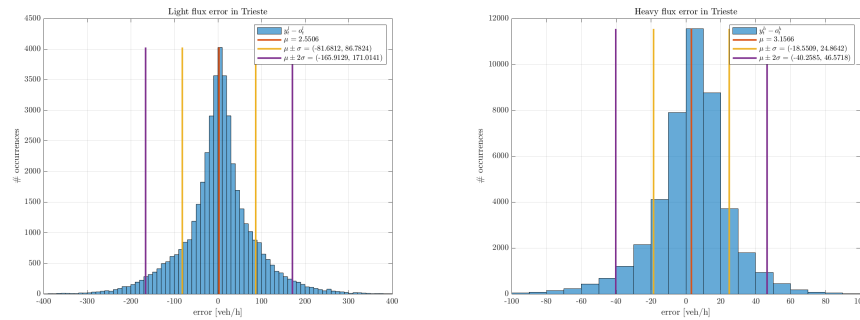


Figure 14: Error histogram of the predictions of  $\mathfrak{P}^{g_2}$  over its test set. Light (left) and heavy (right) vehicles. Error is almost symmetrical w.r.t. 0.

shows instead the result of  $\mathfrak{P}^{g2}$  on an entire day. Every minute, the ground truth (i.e. the total number of vehicles passed in the next 30 minutes, in [vehicle/h]) is compared with its ANN-predicted value, and the offset is evaluated. Real flux  $\mathbf{f}$  is reported along with its *a posteriori* regularization for reference. It can be seen again that the error in prediction is almost symmetrical and it usually corresponds to less than 200 vehicles/h for light vehicles and much less for heavy vehicles.

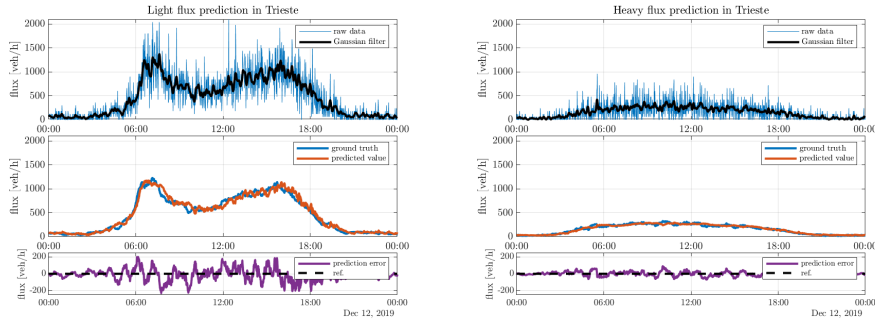


Figure 15:  $\mathfrak{P}^{g2}$ -predicted vs. actually measured flux for 30 mins in the future, minute by minute, for light (left) and heavy (right) vehicles.

Finally, Figure 16 shows an example of a single-minute prediction.

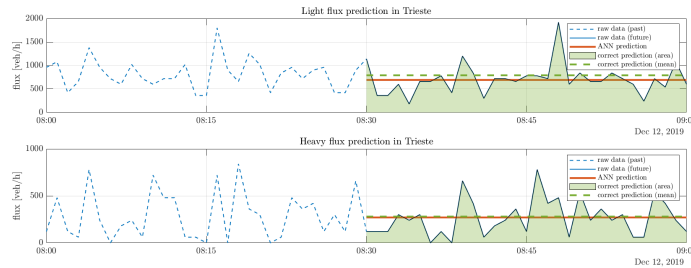


Figure 16: Zoom of a single-minute prediction of the data presented in Figure 15. Prediction is performed at time  $t=8:30$ , where dashed data are available. The predicted value is given by the area beneath the orange line while the target of the prediction is represented by the value of the green shaded area (represented also as the green dashed line for a better comparison).

## 6 Feeding traffic models with ML-enriched data

The tools introduced in the previous sections are useful *per se*, but they can also be used in combination with other, more classical, tools for traffic estimation



and forecast. In this section, we try to get advantage of the pieces of information extrapolated by data in order to improve the accuracy of the macroscopic differential models.

## 6.1 Nowcast

In this section we explore the possible advantages of inverting the fundamental diagram when estimating the current status of traffic density by means of a macroscopic differential model. As mentioned in Section 1, we split the whole road  $[x_{\text{MIN}}, x_{\text{MAX}}]$  into consecutive segments delimited by fixed sensors. We aim at estimating the traffic density at the current time  $t_0$ , therefore we start the simulation at a previous time  $t_0 - \Delta t_{\text{PAST}}$ , with  $\Delta t_{\text{PAST}} > 0$ , assuming an empty road at that time. Let us denote by  $N_S$  the number of road segments (sensors are  $N_S + 1$ ). Each road segment  $S^k := (s^k, s^{k+1})$ ,  $k = 1, \dots, N_S$ , is delimited by two fixed sensors, located at  $x = s^k$  and  $x = s^{k+1}$ , respectively.

We begin with a simplified setting, then we move to the real one.

### 6.1.1 Simplified setting

We consider the classical single-lane single-class LWR model on each road segment  $S_k$

$$\begin{cases} \partial_t \rho^k(x, t) + \partial_x f(\rho^k(x, t)) = 0, & x \in S^k, \quad t \in (t_0 - \Delta t_{\text{PAST}}, t_0) \\ \rho^k(x, t_0) \equiv 0, & x \in S^k \\ \rho^k(s_k, t) = \rho_{\text{IN}}^k(t), & t \in [t_0 - \Delta t_{\text{PAST}}, t_0] \\ \rho^k(s_{k+1}, t) = \rho_{\text{OUT}}^k(t), & t \in [t_0 - \Delta t_{\text{PAST}}, t_0] \end{cases} \quad (10)$$

where  $\rho^k \in [0, \rho_{\text{MAX}}]$  is the vehicle density for some maximal density  $\rho_{\text{MAX}}^k > 0$ , and  $\rho \mapsto f(\rho)$  is the fundamental diagram. Let us assume, as usual, that  $\rho \mapsto f(\rho)$  is concave and denote by  $\sigma$  the argmax of  $f$ , i.e.  $f(\sigma) = \max_{\rho} f(\rho)$  (see Figure 1). Equation (10) is defined independently on each segment. Let us also recall that the relation (1) holds true.

Equation (10) is usually solved by numerical approximation. Let us introduce a grid in the domain  $S^k \times [t_0 - \Delta t_{\text{PAST}}, t_0]$ , with space step  $\Delta x$  and time step  $\Delta t$ . The time interval is divided into  $N_t$  intervals, while each segment  $S^k$  is divided into  $N_x^k$  cells of length  $\Delta x$  and the approximate average density in cell  $C_j^k$ ,  $j = 1, \dots, N_x^k$  at time  $n$  is denoted by  $\rho_j^{k,n}$ . Any conservative numerical scheme [28] for (10) has the form

$$\rho_j^{k,n+1} = \rho_j^{k,n} - \frac{\Delta t}{\Delta x} \left( F(\rho_j^{k,n}, \rho_{j+1}^{k,n}) - F(\rho_{j-1}^{k,n}, \rho_j^{k,n}) \right), \quad j = 1, \dots, N_x^k, \quad (11)$$

where  $F$  is the *numerical flux* (i.e. an approximation of the flux  $f$  at the interface between two consecutive cells). For example, in the case of the Godunov scheme,

we have

$$F(\rho_-, \rho_+) := \begin{cases} \min\{f(\rho_-), f(\rho_+)\} & \text{if } \rho_- \leq \rho_+ \\ f(\rho_-) & \text{if } \rho_- > \rho_+ \text{ and } \rho_- < \sigma \\ f(\sigma) & \text{if } \rho_- > \rho_+ \text{ and } \rho_- \geq \sigma \geq \rho_+ \\ f(\rho_+) & \text{if } \rho_- > \rho_+ \text{ and } \rho_+ > \sigma \end{cases} \quad (12)$$

Let us consider, e.g., the right boundary condition of a given segment  $S^k$ , which corresponds to the left boundary condition of the following segment  $S^{k+1}$ . Let us also drop the index  $k$  from  $\rho$  for readability: if  $j = N_x^k$ , one could follow a

- *flux-based approach*: directly inject in the scheme the flux datum  $f_{s^k}$  measured by the sensor across the interface  $S^k|S^{k+1}$  in place of the numerical outgoing flux  $F(\rho_{N_x}^n, \rho_{\text{OUT}}^n)$ , without estimating the density  $\rho_{\text{OUT}}$ ;
- *density-based approach*: evaluate the numerical flux  $F(\rho_{N_x}^n, \rho_{\text{OUT}}^n)$ , by estimating the density  $\rho_{\text{OUT}}$ .

The two ways are in principle both correct and the first one seems to be more practical since sensors provide flux data only (i.e. density is not available at all). On the other hand, flux data  $f_s$  provided by the sensor are not always compatible with the solution carried by the numerical scheme. In fact, any flux  $f_s$  outside the set of admissible values

$$\{F(\rho_{N_x}^n, \rho) : \rho \in [0, \rho_{\text{MAX}}]\} \quad (13)$$

is not compatible with (10) and leads to negative densities, since more mass comes out than it is available in the road, see Figure 17-left. The question arises how to enforce the compatibility of the sensor data: a natural solution is to project the sensor data into the admissible set of flux data (13).

Regarding the density-based approach, the idea is to use the algorithm introduced in Section 4 to help invert the concave fundamental diagram, i.e. passing from the flux data to the density data by duly distinguishing between free and congested regimes. More precisely, given the sensor flux datum  $f_s < f_{\text{MAX}}$ , the choice between  $\rho$  and  $\rho'$ , with  $\rho' \neq \rho$ ,  $f(\rho) = f(\rho') = f_s$ , is taken depending on the presence or not of the congestion, see Figure 17-right.

In order to discuss the difference between the two approaches, we devise a simple numerical test. We consider an infinitely-long single-lane road. As it is usually done in the mathematical literature, we normalize both density and velocity in the interval  $[0, 1]$  and we choose the fundamental diagram as  $f(\rho) = \rho(1 - \rho)$ . In this case, the maximal flux is 0.25 and it is achieved for  $\rho = \sigma := 0.5$ . Two sensors are located at  $x = s^k := 0.45$  and  $x = s^{k+1} := 0.8$ .

At initial time  $t = t_0 - \Delta t_{\text{PAST}}$  the road has constant density  $\rho = 0.45$  for  $x < 0.52$  while the rest of the road is empty ( $\rho = 0$ ). Immediately after the initial time, an accident occurs at  $x = b := 0.6$  and a bottleneck is formed between the two sensors.

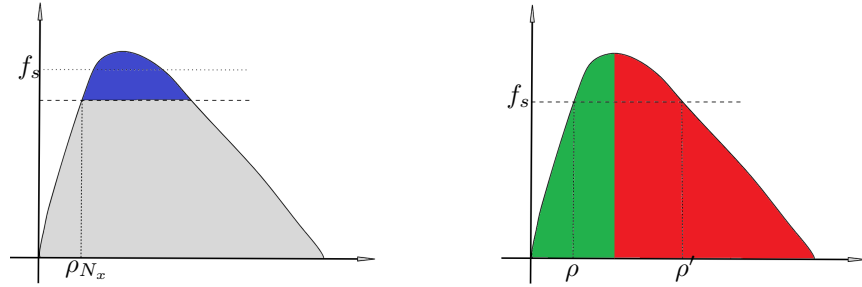


Figure 17: Fundamental diagram  $f = f(\rho)$ . (Left) Example of nonadmissible sensor flux datum for the numerical flux 12: no flux  $f_s > f(\rho_{N_x})$  is compatible with any boundary condition  $\rho_{OUT}$ . (Right) Given the sensor flux datum  $f_s$ , the corresponding density  $\rho$  or  $\rho'$  is chosen depending on the presence or not of the congestion.

Figure 18 shows the reference simulation, i.e. what we assume it is really happening on the road and that we would ideally like to reproduce with data at our disposal: at the initial time, a rarefaction fan immediately appears at  $x = 0.52$  and the right part of the road starts populating. When enough vehicles have reached the bottleneck at  $x = b$ , a queue appears and starts back-propagating. The queue reaches the sensor at  $s^k$  and continues back-propagating. From the bottleneck on, vehicles set off at maximal flux ( $\rho = \sigma$ ) and proceed normally.

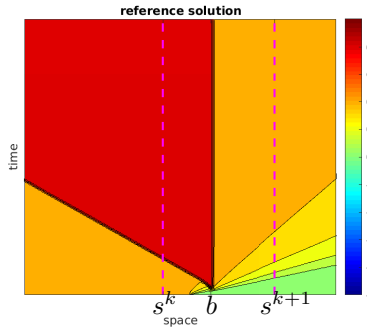


Figure 18: Academic test. Reference solution

It is plain that, if the traffic is observed only at sensors, we cannot be able to perceive the accident and the formation of the bottleneck in its actual position. The effects of the accident will be visible only when the queue reaches the sensor at  $x = s^k$ .

Figure 19-left show the result obtained by the *density-based* approach. When the queue is perceived at  $x = s^k$ , the recorded flux  $f_{s^k} = 0$  is correctly translated into the maximal density  $\rho = 1$  inverting the fundamental diagram. Therefore, the queue continues back-propagating while, for  $x > s^k$ , the traffic restarts with

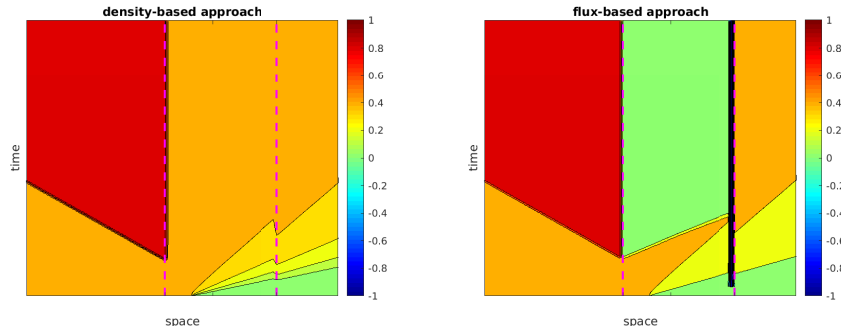


Figure 19: Academic test. Left: density-based approach. Right: flux-based approach

maximal flux. At  $x = s^{k+1}$  whatever approach is used, the solution is the same and the traffic keeps going with the same dynamics.

Figure 19-right show instead the result obtained by the *flux-based* approach. As before, when the queue reaches  $x = s^k$  the sensor registers a null flux there, and a queue starts back-propagating. Since the flux through the sensor is null, no one moves from the sensor on, and the space between the sensors becomes empty in a short time. At this point, the flux data at the right sensor becomes incompatible with the traffic condition (the road is empty but the sensor perceives moving vehicles). Therefore, a negative density appears at  $x = s^{k+1}$ . For  $x > s^{k+1}$  traffic dynamics restart correctly with the measured flux.

In conclusion, in this case, the density-based approach is preferable since it catches with better precision the real scenario, and this is achieved since the density-based approach actually puts in the simulation additional information about the system other than the naked flux data, i.e. the discrimination free/congested scenario.

### 6.1.2 Real setting

In this section, we consider real sensor data provided by Autovie Venete, see Section 2. The major difference with respect to the previous test is that now we deal with a three-lane highway and two classes of vehicles (light and heavy), with coupled dynamics. The LWR-like model is generalized to this case by a system of PDEs

$$\begin{cases} \partial_t \rho_L^k + \partial_x f_L(\rho_L^k, \rho_H^k) = 0, & x \in S_k, \quad t \in (t_0 - \Delta t_{\text{PAST}}, t_0) \\ \partial_t \rho_H^k + \partial_x f_H(\rho_L^k, \rho_H^k) = 0, & x \in S_k, \quad t \in (t_0 - \Delta t_{\text{PAST}}, t_0) \end{cases} \quad (14)$$

where  $\rho_L$ ,  $f_L$  and  $\rho_H$ ,  $f_H$  are the density and flux of light and heavy vehicles, respectively. Equation (14) is complemented with initial and boundary conditions as in (10). Moreover, the two classes of vehicles do not share the road in the same manner, being heavy vehicles not allowed in the fastest lane. The coupled

dynamics, with uneven space occupancy, was already derived in [6] and we refer the reader to that paper for both the mathematical and numerical details. Here we just recall that we consider a *phase transition* (cf. [11, 12, 15]) due to the presence of two states of the system, see Figure 20:

- The *partial-coupling phase*, when heavy vehicles influence the dynamics of light ones but not vice versa. Light vehicles are then mainly in the fast lane and heavy vehicles are independent of them. In this case, the two equations in the system (14) are partially coupled, i.e.  $f_H$  only depends on  $\rho_H$ ,

$$\begin{cases} \partial_t \rho_L + \partial_x f_L(\rho_L, \rho_H) = 0, \\ \partial_t \rho_H + \partial_x f_H(\rho_H) = 0. \end{cases}$$

- The *full-coupling phase*, when light vehicles are too much to fit the fast lane only and then invade the slow lane(s), influencing the dynamics of heavy vehicles. In this case, the two equations are fully coupled and fall in the general form of the system (14).

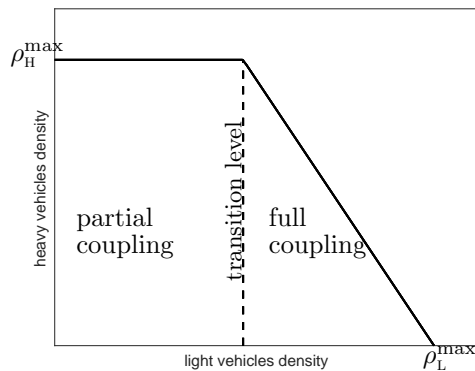


Figure 20: The two phases of the model (14): the density of light vehicles determines if the dynamics is partially or fully coupled.

Figure 21 shows the families of fundamental diagrams devised for taking into account the flux-density dependence for each class of vehicles given the density of the other class, see [6] for more details.

We are now ready to describe the test. Let us consider a stretch of road of length 32 km, across two segments. A sensor is located at the interface at  $x = 20.5$  km. A permanent bottleneck caused by the transition from 3 to 2 lanes is located at  $x = 29.5$  km.

In the real scenario, confirmed by direct observation of Autovie Venete personnel, the bottleneck causes congestion for heavy vehicles only, which propagates back and, in turn, slows down the light vehicles. As in the academic test, the model perceives the congestion only when it reaches the sensor.

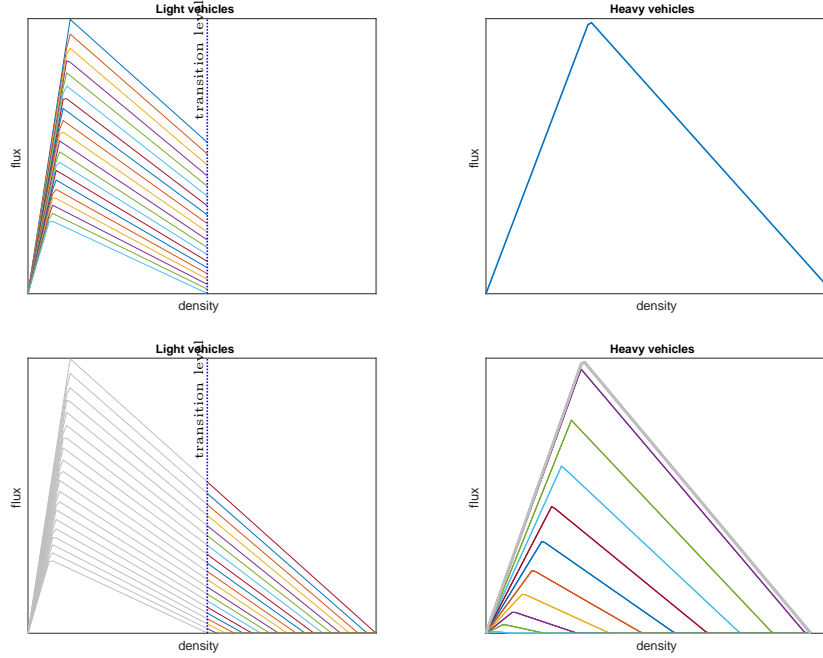


Figure 21: Families of fundamental diagrams in the two phases. Left: function  $\rho_L \rightarrow f_L(\rho_L, \rho_H)$  for several values of  $\rho_H$  in the case of partial (top) and full (bottom) coupling. Right: function  $\rho_H \rightarrow f_H(\rho_L, \rho_H)$  for several values of  $\rho_L$  in the case of partial (top) and full (bottom) coupling.

Figure 22 shows the density and the velocity of light and heavy vehicles in the case of flux- and density-based approaches. If the flux-based approach is used, a piece of congestion for heavy vehicles propagates upstream the sensor and it keeps slowing down the light vehicles (their speed is about 60 km/h), while downstream traffic is free. If instead the density-based approach is used, the dynamics are more complicated because the corrections  $\rho \rightarrow \rho'$  for each class changes the fundamental diagram of the other class, see Figure 21: downstream, we observe a slowdown for both vehicle classes (velocity is about 20 km/h for heavy vehicles and 100 km/h for light vehicles); upstream, both vehicle classes are totally congested (velocity is zero). Again, the density-based approach better matches the real scenario especially between the sensor and the bottleneck.

## 6.2 Forecast

In this section, we explore the possible advantage of estimating the incoming traffic volume for traffic forecast. The idea is to run again the simulator based on the model (14) from time  $t_0$  to time  $t_0 + \Delta t_{\text{FUT}}$ . Conversely to the previous case, here we aim at forecasting the traffic distribution in the whole road  $[x_{\text{MIN}}, x_{\text{MAX}}]$

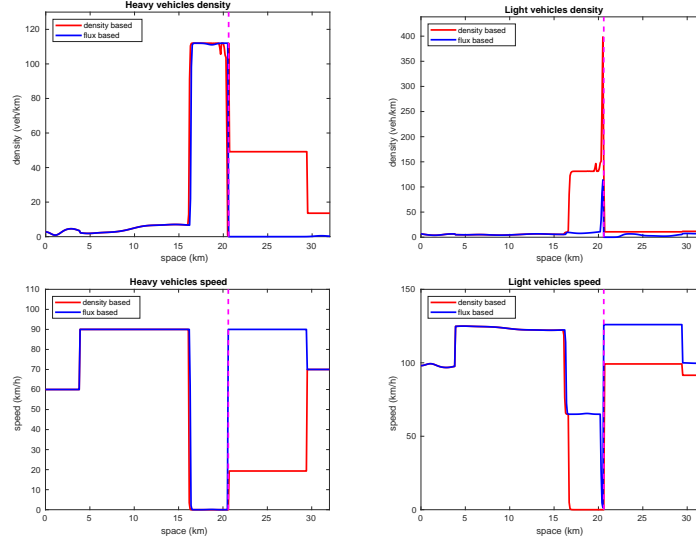


Figure 22: Test with real data. Density and velocity of light and heavy vehicles for the flux- and density-based approach at time  $t = t_0$ .

(with no interruption at sensors), starting from the outcome of the nowcast procedure as initial condition for the densities  $(\rho_L, \rho_H)$ . Obviously, sensor data cannot be longer used here since they are not yet available, and the problem arises which boundary conditions should be used. Ideally, one should estimate the correct future inflow and outflow at each time step  $\Delta t$ , but this is extremely difficult considering the high variability of the traffic dynamics. On the opposite side, the simplest solution is to assume  $\rho_{IN} = 0$  so as to assume that nobody enters the road, and  $\rho_{OUT} = 0$  so as to guarantee maximal outflow, but this leads to a gradual emptying of the road starting from the inflow boundary. A possible compromise is to set  $\rho_{OUT} = 0$  for maximal outflow and keep a *constant* inflow, equal to the last available datum, or an average of the last minutes, or equal to a certain value predicted by a separate procedure. Here we consider the outcome of the ANN set up in Section 5, which estimates the traffic volume for 30 min in the future, directly injecting the flux datum in the numerical scheme.

In order to measure the error of the simulation, we compare the forecast traffic density distribution  $x \rightarrow \rho_{\{L,H\}}^F(x, t_0 + \delta)$  at any future time  $t_0 + \delta$ ,  $\delta \in [0, \Delta t_{FUT}]$ , with the estimated nowcast  $x \rightarrow \rho_{\{L,H\}}^N(x, t_0 + \delta)$  computed as soon as the data becomes available (i.e. at time  $t_0 + \delta$ ). Relative  $L^1$ -distance between traffic densities is computed as usual

$$E_{\{L,H\}}(t) := \frac{\|\rho_{\{L,H\}}^F - \rho_{\{L,H\}}^N\|_{L^1}}{\|\rho_{\{L,H\}}^N\|_{L^1}} \quad \text{where} \quad \|\rho\|_{L^1} := \int_{x_{MIN}}^{x_{MAX}} |\rho(x, t)| dx.$$

Figure 23 shows the error as a function of time for light and heavy vehicles

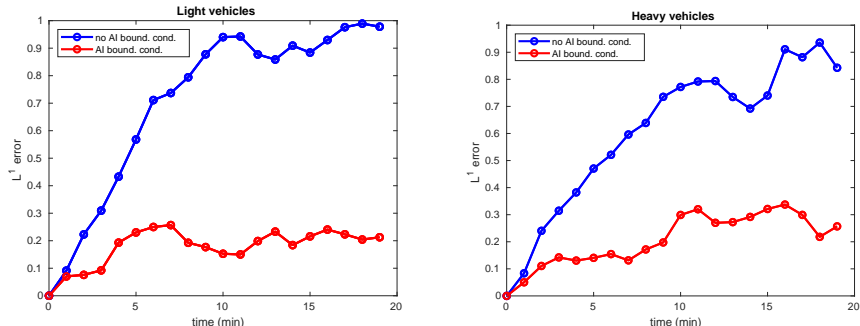


Figure 23: Normalized  $L^1$  error as a function of time with ANN-generated inflow boundary condition (red line) and with null inflow (blue line), for light (left) and heavy (right) vehicles.

separately, on a stretch of road of length 16 km with two lanes. We compare the error made by using the predicted value of the incoming flux with the one made by simply assuming a null inflow (the simplest choice). It is interesting to note that in the ANN-aided traffic prediction, the error initially increases and then stays bounded below about 30% for both light and heavy vehicles. This means that, although the traffic distribution can be shifted horizontally with respect to the real one, the total mass (i.e. the amount of vehicles) does not differ excessively. On the contrary, using a null inflow the error rapidly increases up to 100% and then stabilizes, as expected.

## 7 Conclusions

In this paper, we have proposed a hybrid model-/data-driven method for reconstructing and predicting traffic distributions on extra-urban roads and highways. Similar to [41, 23], the idea is to use an ANN as a joining link between real data and the mathematical model, avoiding using the latter in the training phase of the ANN. This approach allows exploiting the power of ML in extrapolating information from real-time and historical data and then passing to the model a piece of processed information that can be immediately incorporated. We have also observed that data-driven approaches based on data measured by fixed sensors can hardly extrapolate any kind of spatial information, i.e. information about the spatial distribution of traffic *between* sensors. This is the reason why we think that the mathematical model is essential in TSE since it catches the right causality of traffic dynamics in space and time.



## Acknowledgments

The authors want to warmly thank Andrea Appella and Giulia Tatafiore for their support in analyzing and understanding data, as well as improving the quality of the labeled dataset. The authors want also to thank Paolo Ranut who supported this research on behalf of the motorway company Autovie Venete S.p.A.

## Funding

This work was carried out within the research project “SMARTOUR: Intelligent Platform for Tourism” (No. SCN\_00166) funded by the Ministry of University and Research with the Regional Development Fund of European Union (PON Research and Competitiveness 2007–2013).

E.C. and M.B. would also like to thank the Italian Ministry of Instruction, University and Research (MIUR) to support this research with funds coming from PRIN Project 2017 (No. 2017KKJP4X entitled “Innovative numerical methods for evolutionary partial differential equations and applications”).

This work was also partially funded by Autovie Venete S.p.A.

All the authors are members of the INdAM Research group GNCS.

## References

- [1] A. Aw and M. Rascle. Resurrection of “second order” models of traffic flow. *SIAM Journal on Applied Mathematics*, 60(3):916–938, 2000.
- [2] M. Barreau, M. Aguiar, J. Liu, and K. H. Johansson. Physics-informed learning for identification and state reconstruction of traffic density. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 2653–2658. IEEE, 2021.
- [3] N. Bellomo and C. Dogbe. On the modeling of traffic and crowds: A survey of models, speculations, and perspectives. *SIAM Review*, 53(3):409–463, 2011.
- [4] G. Bretti, M. Briani, and E. Cristiani. An easy-to-use algorithm for simulating traffic flow on networks: Numerical experiments. *Discret. Contin. Dyn. Syst. Ser. S*, 7:379–394, 2014.
- [5] M. Briani and E. Cristiani. An easy-to-use algorithm for simulating traffic flow on networks: Theoretical study. *Netw. Heterog. Media*, 9:519–552, 2014.
- [6] M. Briani, E. Cristiani, and P. Ranut. Macroscopic and multi-scale models for multi-class vehicular dynamics with uneven space occupancy: A case study. *Axioms*, 10(2):102, 2021.

- [7] Bundesanstalt für Straßenwesen Bergisch Gladbach (BAST). Technische Lieferbedingungen für streckenstationen. Technical report, Bundesministerium für Verkehr, Bau und Stadtentwicklung (BMDV), Aug 2012. [PDF], [bast.de/tls](http://bast.de/tls).
- [8] G. Carlier and F. Santambrogio. A continuous theory of traffic congestion and Wardrop equilibria. *Journal of Mathematical Sciences*, 181(6):792–804, 2012.
- [9] W. Chen, F. Guo, and F.-Y. Wang. A survey of traffic data visualization. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):2970–2984, 2015.
- [10] A. H. Chow, Y. Li, and K. Gkiotsalitis. Specifications of fundamental diagrams for dynamic traffic modeling. *Journal of Transportation Engineering*, 141(9):04015015, 2015.
- [11] R. M. Colombo. Hyperbolic phase transitions in traffic flow. *SIAM J. Appl. Math.*, 63(2):708–721, 2002.
- [12] R. M. Colombo, P. Goatin, and B. Piccoli. Road networks with phase transitions. *J. Hyperbolic Differ. Eq.*, 7(1):85–106, 2010.
- [13] E. Cristiani and F. S. Priuli. A destination-preserving model for simulating Wardrop equilibria in traffic flow on networks. *Networks & Heterogeneous Media*, 10(4):857, 2015.
- [14] C. F. Daganzo. The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B: Methodological*, 28(4):269–287, 1994.
- [15] M. L. Delle Monache, K. Chi, Y. Chen, P. Goatin, K. Han, J. Qiu, and B. Piccoli. Three-phase fundamental diagram from three-dimensional traffic data. *Axioms*, 10:17, 2021.
- [16] W. Du, Q. Zhang, Y. Chen, and Z. Ye. An urban short-term traffic flow prediction model based on wavelet neural network with improved whale optimization algorithm. *Sustainable Cities and Society*, 69:102858, 2021.
- [17] S. Fan and D. B. Work. A heterogeneous multiclass traffic flow model with creeping. *SIAM J. Appl. Math.*, 75(2):813–835, 2015.
- [18] W. Fang, W. Zhuo, J. Yan, Y. Song, D. Jiang, and T. Zhou. Attention meets long short-term memory: A deep learning network for traffic flow forecasting. *Physica A: Statistical Mechanics and its Applications*, 587:126485, 2022.
- [19] A. Ferrara, S. Sacone, and S. Siri. *Freeway Traffic Modelling and Control*. Springer, 2018.

- [20] M. Garavello and B. Piccoli. *Traffic Flow on Networks*. American Institute of Mathematical Sciences, 2006.
- [21] Y. Gu, W. Lu, L. Qin, M. Li, and Z. Shao. Short-term prediction of lane-level traffic speeds: A fusion deep learning model. *Transportation Research Part C: Emerging Technologies*, 106:1–16, 2019.
- [22] D. Helbing. Traffic and related self-driven many-particle systems. *Reviews of Modern Physics*, 73(4):1067, 2001.
- [23] M. Herty and N. Kolbe. Data-driven models for traffic flow at junctions. preprint arxiv:2212.08912, 2022.
- [24] Y.-L. Hsueh and Y.-R. Yang. A short-term traffic speed prediction model based on LSTM networks. *International Journal of Intelligent Transportation Systems Research*, 19(3):510–524, 2021.
- [25] A. J. Huang and S. Agarwal. Physics-informed deep learning for traffic state estimation: Illustrations with LWR and CTM models. *IEEE Open Journal of Intelligent Transportation Systems*, 3:503–518, 2022.
- [26] F. Kessels. *Traffic Flow Modelling*. Springer, 2019.
- [27] K. Lee, M. Eo, E. Jung, Y. Yoon, and W. Rhee. Short-term traffic prediction with deep neural networks: A survey. *IEEE Access*, 9:54739–54756, 2021.
- [28] R. J. LeVeque. *Numerical Methods for Conservation Laws*. Birkhäuser, 1992.
- [29] M. J. Lighthill and G. B. Whitham. On kinematic waves II. A theory of traffic flow on long crowded roads. *Proc. R. Soc. Lond. Ser. A*, 229:317–345, 1955.
- [30] S. Maerivoet and B. De Moor. Cellular automata models of road traffic. *Physics Reports*, 419(1):1–64, 2005.
- [31] E. L. Manibardo, I. Laña, and J. Del Ser. Deep learning for road traffic forecasting: Does it make a difference? *IEEE Transactions on Intelligent Transportation Systems*, 2021.
- [32] A. Messner and M. Papageorgiou. METANET: A macroscopic simulation program for motorway networks. *Traffic Engineering & Control*, 31(8-9):466–470, 1990.
- [33] Z. Mo, R. Shi, and X. Di. A physics-informed deep learning paradigm for car-following models. *Transportation Research Part C: Emerging Technologies*, 130:103240, 2021.

- [34] S. Modi, J. Bhattacharya, and P. Basak. Multistep traffic speed prediction: A deep learning based approach using latent space mapping considering spatio-temporal dependencies. *Expert Systems with Applications*, 189:116140, 2022.
- [35] D. Ni, H. K. Hsieh, and T. Jiang. Modeling phase diagrams as stochastic processes with application in vehicular traffic flow. *Appl. Math. Model.*, 53:106–117, 2018.
- [36] H. J. Payne. Model of freeway traffic and control. *Mathematical Model of Public System*, pages 51–61, 1971.
- [37] P. I. Richards. Shock waves on the highway. *Oper. Res.*, 4(1):42–51, 1956.
- [38] Z. (Sean) Qian, J. Li, X. Li, M. Zhang, and H. Wang. Modeling heterogeneous traffic flow: A pragmatic approach. *Transp. Res. Part B*, 99:183–204, 2017.
- [39] T. Seo, A. M. Bayen, T. Kusakabe, and Y. Asakura. Traffic state estimation on highway: A comprehensive survey. *Annual Reviews in Control*, 43:128–151, 2017.
- [40] R. Shi, Z. Mo, and X. Di. Physics-informed deep learning for traffic state estimation: A hybrid paradigm informed by second-order traffic models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 540–547, 2021.
- [41] R. Shi, Z. Mo, K. Huang, X. Di, and Q. Du. A physics-informed deep learning paradigm for traffic state and fundamental diagram estimation. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):11688–11698, 2022.
- [42] D. A. Tedjopurnomo, Z. Bao, B. Zheng, F. Choudhury, and A. K. Qin. A survey on modern deep neural network for traffic prediction: Trends, methods and challenges. *IEEE Transactions on Knowledge and Data Engineering*, 2020.
- [43] B. T. Thodi, Z. S. Khan, S. E. Jabari, and M. Menéndez. Incorporating kinematic wave theory into a deep learning method for high-resolution traffic speed estimation. *IEEE Transactions on Intelligent Transportation Systems*, 2022.
- [44] F. van Wageningen-Kessels. Framework to assess multiclass continuum traffic flow models. *Transp. Res. Rec.*, 2553(1):150–160, 2016.
- [45] F. van Wageningen-Kessels, H. Van Lint, K. Vuik, and S. Hoogendoorn. Genealogy of traffic flow models. *EURO Journal on Transportation and Logistics*, 4(4):445–473, 2015.

- [46] H. Wang, D. Ni, Q.-Y. Chen, and J. Li. Stochastic modeling of the equilibrium speed-density relationship. *J. Adv. Transp.*, 47:126–150, 2013.
- [47] K. Wang, C. Ma, Y. Qiao, X. Lu, W. Hao, and S. Dong. A hybrid deep learning model with 1DCNN-LSTM-Attention networks for short-term traffic flow prediction. *Physica A: Statistical Mechanics and its Applications*, 583:126293, 2021.
- [48] S. Wang, J. Cao, and P. Yu. Deep learning for spatio-temporal data mining: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3681–3700, 2022.
- [49] Y. Wang, M. Zhao, X. Yu, Y. Hu, P. Zheng, W. Hua, L. Zhang, S. Hu, and J. Guo. Real-time joint traffic state and model parameter estimation on freeways with fixed sensors and connected vehicles: State-of-the-art overview, methods, and case studies. *Transportation Research Part C: Emerging Technologies*, 134:103444, 2022.
- [50] J. G. Wardrop. Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers*, 1(3):325–362, 1952.
- [51] G. B. Whitham. *Linear and nonlinear waves*. John Wiley & Sons, 2011.
- [52] W. Xiangxue, X. Lunhui, and C. Kaixun. Data-driven short-term forecasting for urban road network traffic based on data processing and LSTM-RNN. *Arabian Journal for Science and Engineering*, 44(4):3043–3060, 2019.
- [53] J. Xing, W. Wu, Q. Cheng, and R. Liu. Traffic state estimation of urban road networks by multi-source data fusion: Review and new insights. *Physica A: Statistical Mechanics and its Applications*, page 127079, 2022.
- [54] H. Yan, L. Fu, Y. Qi, D.-J. Yu, and Q. Ye. Robust ensemble method for short-term traffic flow prediction. *Future Generation Computer Systems*, 133:395–410, 2022.
- [55] Y. Yuan, Z. Zhang, and X. T. Yang. Macroscopic traffic flow modeling with physics regularized Gaussian process: Generalized formulations, preprint. <https://arxiv.org/abs/2007.07762>, 2022.
- [56] Y. Yuan, Z. Zhang, X. T. Yang, and S. Zhe. Macroscopic traffic flow modeling with physics regularized Gaussian process: A new insight into machine learning applications in transportation. *Transportation Research Part B: Methodological*, 146:88–110, 2021.
- [57] H. M. Zhang. A non-equilibrium traffic model devoid of gas-like behavior. *Transportation Research Part B: Methodological*, 36(3):275–290, 2002.
- [58] Z. Zhang, Y. Yuan, and X. Yang. A hybrid machine learning approach for freeway traffic speed estimation. *Transportation Research Record*, 2674(10):68–78, 2020.

- [59] Z. Zhao, W. Chen, X. Wu, P. C. Chen, and J. Liu. LSTM network: A deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems*, 11(2):68–75, 2017.
- [60] G. Zheng, W. K. Chai, and V. Katos. A dynamic spatial–temporal deep learning framework for traffic speed prediction on large-scale road networks. *Expert Systems with Applications*, 195:116585, 2022.