Long and Short-Term Bus Arrival Time Prediction With Traffic Density Matrix

DANCHO PANOVSKI*, (Member, IEEE), AND TITUS ZAHARIA★
Laboratoire SAMOVAR, ARTEMIS Department, Institut Polytechnique de Paris, Télécom SudParis, 91000 Evry-Courcouronnes, France
Corresponding author: Dancho Panovski (dancho.panovski@telecom-sudparis.eu)
This work has been carried out within the framework of the FUI (Fonds Unique Interministeriel) ETS (Electronic Ticketing System) project, supported by the French Government.

ABSTRACT This article introduces a novel machine learning approach dedicated to the prediction of the bus arrival times in the bus stations over a given itinerary, based on the so-called Traffic Density Matrix (TDM). The TDM constructs a localized representation of the traffic information in a given urban area that can be specified by the user. We notably show the necessity of disposing of such data for successful, both short-term and long-term prediction objectives, and demonstrate that a global prediction approach cannot be a feasible solution. Several different prediction approaches are then proposed and experimentally evaluated on various simulation scenarios. They include traditional machine learning techniques, such as linear regression and support vector machines (SVM), but also advanced, highly non-linear neural network-based approaches. Within this context, various network architectures are retained and evaluated, including fully connected neural networks (FNN), convolutional neural networks (CNN), recurrent neural networks (RNN) and LSTM (Long Short Term Memory) approaches. The experimental evaluation is carried out under two types of different scenarios, corresponding to both long term and short-term predictions. To this purpose, two different data models are constructed, so-called ODM (Operator Data Model) and CDM (Client Data Model), respectively dedicated to long term and short-term predictions. The experimental results obtained show that increasing the degree of non-linearity of the predictors is highly beneficial for the accuracy of the obtained predictions. They also show that significant improvements can be achieved over state of the art techniques. In the case of long-term prediction, the FNN method performs the best when compared with the baseline OLS technique, with a significant increase in accuracy (more than 66%). For short-term prediction, the FNN method is also the best performer, with more than 15% of gain in accuracy with respect to OLS.

INDEX TERMS Machine learning, deep learning, artificial neural networks, public transportation, traffic simulation, intelligent transportation systems, bus arrival time prediction.

I. INTRODUCTION
Traffic jam, pollution, security, unreliable public transportation, limited parking facilities, are the main problems that modern cities are facing today all over the world. Addressing and minimizing such problems could bring some potential benefits for the society in terms of safety for both pedestrians and vehicles, better traffic management, fuel/energy consumption and environmental issues.

In addition, let us underline that vehicular transports in the urban areas performed by individuals or public transportation vehicles contribute heavily on the carbon footprint, called greenhouse gas. The data [1] provided by the French government in 2015 gives an extensive overview of the key numbers related to the production of the greenhouse gas. Transportation, in general, is responsible for 27% of the production of greenhouse gas. Transport operated by road represents 94.8% of them. The logical step to reduce such emissions is to decrease the number of individual vehicles used for transportation, by privileging reliable public transportation facilities. Let us underline that the question of reliability is highly important since the user acceptancy of public transportation strongly depends on. Within this framework, disposing of efficient traffic prediction tools, dedicated to public transportation, is a highly challenging issue, for both users (which want to be informed precisely and in real-time) and transportation operators (which aim at optimizing their transport networks/itineraries).

Different approaches attempt to address this issue. One solution relates to the infrastructure and proposes building
bigger lanes, bridges, roundabouts or underground passages. Such a solution is applicable only in areas where the extension of the infrastructure is possible, excluding hence most of the densely populated cities. In addition, it is also the costliest solution. Another approach concerns the creation of areas restricted to specific types of vehicles (such as public transportation or electrical vehicles) and thus with low-density traffic flow. Even though the proposed solutions are achievable, they are not suitable for numerous urban areas.

Seeking a response to such problems, a new research domain, so-called Intelligent Transportation Systems (ITS) [2] has emerged in early 1970s and continuously evolved since then. ITS aims today at bringing advanced techniques and technologies into the transportation systems [3], [4], such as electronic sensors, data transmission and intelligent control technologies. Under this framework, the required data can be obtained from different and diverse sources such as smart cards, GPS, sensors, video streams, images, social media and so on. The main purpose is to provide better services for both drivers and passengers and to globally improve the whole transportation system [5].

In this article, we are investigating an ITS-related predictive approach, based on computer simulation and machine learning techniques. The proposed methods are applied over existing systems in an artificial environment (traffic simulator software) that can simulate real-life scenarios. The main goal is to develop prediction algorithms that can improve the efficiency of transportation over existing systems/infrastructures. More specifically, our objective is to improve the traffic flow management in public transportation with the help of both short and long-term prediction techniques, based on advanced machine learning algorithms.

The rest of the article is organized as follows. Section II presents the state of the art in the field. In Section III the simulation scenario considered is described in details. Section IV introduces the TDM (Traffic Density Matrix) concept and the associated data models. Section V elaborates extensively on the retained machine learning approaches. In Section VI the experimental results are presented and discussed. Finally, Section VII concludes the article and opens some perspectives of future work.

II. STATE OF THE ART
In the mainstream literature under the scope of ITS and public transportation systems two different types of prediction targets are addressed:

- **Short-term predictions** [6], which usually reflect a prediction horizon of up to one or two hours (typical example: predict the next bus arrival time information at the bus station). Such predictions are usually *client-based*, they take place in consumer applications and board notifications (located at the bus stop stations).

- **Long-term predictions** [7], which correspond to at least one or more days (example: prediction of the bus arrival times at the bus stops for an entire day). Such predictions are mostly *operator-based*, used by the company that provides the transportation services for analysis and system improvements.

Whatever the prediction target considered, two different families of methodologies, so-called *model-driven* and *data-driven*, can be considered.

The *model-driven* approaches attempt, usually through simulations of the whole urban region under analysis, to compute, analyze and predict the behavior or the performance of the observed entities in the system. Among the well-known model-driven frameworks reported in the state of the art [8]–[10] let us notably mention the Dynamic Network Assignment for Management of Transportation to Travelers (DyNAMIT) [11], developed by MIT’s Intelligent Transportation System Laboratory. DyNAMIT requires both offline and online (real-time) data in order to work properly. Performing such simulations is a complex task, which requires a large variety of information as input (traffic, socio-economics, age, gender . . . ).

On the other hand, *data-driven* methods [12] are exploiting both historical and real-time traffic data to perform analysis with data mining-inspired techniques in order to provide possible predictions for future traffic situations. Here, the decisions are based on data analysis and interpretation [13] and can be used for both short-term and long-term prediction.

Well-known forecast techniques [14], [15], still widely used today, mainly include time series analysis [16] exponential smoothening [17], Kalman filtering [18], machine learning (SVMs) [19], ANNs (Artificial Neural Networks) [20], or wavelet-based analysis [21].

On the other hand, let us mention the parametric methods that rely on a relatively reduced set of generic parameters that are governing the states and the output of the model, under a pre-defined data structure [22]. A wide variety of parametric models is today available. Among the most popular ones, let us cite the fully-fledged traffic models like traffic simulators with OD (Origin-Destination) matrices, extended Kalman filters [23], time series analysis approaches [24], or linear regression [25]. Within this context, the Box-Jenkins’ *Autoregressive Integrated Moving Average* (ARIMA) model [26], is one of the most known approach. The proposed model overcomes the issues of incomplete or noisy data by proposing a prediction scheme using SARIMA model for traffic flow prediction that can be applied when only a limited data input is available.

With the rapid data growth in ITS recently, machine learning has become an important tool for solving complex problems like prediction, analytics and pattern identification from large amount of data [27].

Under this framework, various categories of machine learning algorithms have been considered, including supervised [28], unsupervised [29] and reinforcement learning [30] techniques.

Whatever the machine learning algorithm involved, the availability of the training data is a major issue that needs to be carefully taken into account. In our case, we have
considered a simulation approach, able to yield traffic data that is likely to appear in real-life scenarios.

A. TRAFFIC SIMULATION

Traffic simulation can be defined as a mathematical model of transportation systems, implemented through the application of dedicated computer software. The first computer simulation started in 1955 in Los Angeles, California, when D.L. Gerlough published his dissertation “Simulation of freeway traffic on a general-purpose discrete variable computer” [29]. Since then, numerous traffic simulations researches have been conducted.

There are two main categories of traffic simulation models: microscopic [30] and microscopic [31] and one sub-category, called mesoscopic [32], which combines some of the properties from both simulation models. While a microscopic traffic simulator focuses on the mobility of each individual entity in the system, a macroscopic traffic simulator provides a complete traffic flow of the system taking into account more global constraints, such as general traffic density and vehicle distribution.

In our work, we have adopted a microscopic traffic simulation approach, which is more appropriate for our experiments, since it is able to provide a detailed image (with location, time, speed, noise, pollution...) of each vehicle in the system. Table 1 summarizes some of the most advanced microscopic platforms available today, with both open source and commercial solutions.

### TABLE 1. Traffic simulation software.

| Open Source software | Commercial software |
|----------------------|---------------------|
| [33] sumo            | [34] quadetone paramics |
| [35] matsim          | [36] vissim          |
| [37] repast          | [38] vissum          |
| [39] mainsim         | [40] aimson          |
| [41] satsum          |                     |

In the right column of Table 1, solely for information purposes, some commercial solutions are cited. We will not consider them further in our work, despite their advantages (real-life support, enhanced graphical user interfaces) since our purpose is to set up an open framework that can be exploited in the future by the research community.

In the left column of Table 1 are presented open source, free software packages that are usually adopted by the research community and individual developers.

In our work, we have retained the SUMO [42] open source platform. SUMO (Simulation of Urban MOBility) is a microscopic, multi-modal traffic simulator that is able to simulate different types of traffic data. SUMO allows simulating a given traffic demand which consists of individual vehicle moves through a given road network. The simulation permits addressing a large set of traffic management topics. It is purely microscopic: each vehicle is modeled explicitly, has an own route, and moves individually through the network. Simulations are deterministic by default but there are various options for introducing randomness. The SUMO package allowed us to develop our synthetic data further used for prediction purposes.

B. MACHINE LEARNING APPROACHES

With the rapid data growth and expansion of ITS in the recent years, machine learning becomes an important tool for solving complex problems like prediction, analytics and deriving patterns from a large amount of data [27].

In our work, we have considered supervised machine learning algorithms that include both traditional approaches (linear regression, support vector regression–SVR) and neural network-based approaches.

1) TRADITIONAL MACHINE LEARNING (ML) ALGORITHMS

Let us first recall some well-known ML algorithms involved in the public transportation prediction domain.

As the simplest approach, let us mention the case of linear regression [43], also known as OLS (Ordinary Least Squares). Here, the prediction model is based on linear combinations of its covariates, and the parameters indicate the contribution of the covariate to the outcome. The model is computationally light, and in some relatively simple cases can produce satisfactory results.

Support Vector Machines (SVMs) and Support Vector Regression (SVR) [44] are well-known machine learning algorithms for solving classification and regression problems respectively. Over the years, they have been continuously gaining in popularity and adoption, for various applications, since they offer convenient implementation and testing facilities.

SVMs algorithms are still today one of the most popular and widely adopted methods. In many cases, they achieve good and accurate prediction results that can be used as standalone solutions or as a baseline for further improvements.

Let us cite some recent use cases and implementations reported in the literature. In [45], different features including road length, weather and speed have been used in order to successfully predict the bus arrival time. In a different manner, in [46], the SVM algorithm has been used to predict multiple bus arrival times at one bus stop station. An application for bus time prediction was also proposed in [47]. Here, the distance between each bus stop station and the time needed for the bus to reach the station have been exploited.

Among other algorithms used for public transportation prediction, let us also mention the k-nearest neighbors [48], or random forests [49].

Traditional ML algorithms have proven to be powerful tools for solving particular problems. Nevertheless, there are some key issues and challenges that need to be overcome. First, traffic measurement data is not available for all traffic links so the data may be incorrect or incomplete. Second, traffic flows are complex and dynamic so they exhibit highly non-linear behaviors. For this reason, it is essential to look for solutions that can overcome such problems and produce better results. One potential solution concerns the neural network.
approaches, which have made in recent years a spectacular breakthrough in various fields of applications, including computer vision, object recognition, natural speech processing and so on. The following section examines how such artificial intelligence-related approaches have been employed for the objectives of public transportation prediction.

2) NEURAL NETWORK APPROACHES
Artificial neural networks [50] algorithms are specific types of machine learning algorithms that are able to learn both the model structure and the corresponding parameters directly from the data. Such models manage to achieve a certain robustness and prove to be less sensitive to incorrect or incomplete data. The main advantage is that the complexity, dynamics, and non-linearity encountered in various traffic conditions can be intrinsically taken into account. The development, in recent years, of affordable GPUs (Graphical Processing Units) with high processing power and intrinsic parallelization abilities, required for accomplishing the learning stage, has largely catalyzed the emergence of deep neural-networks approaches. Let us detail some recent achievements and research works carried out in the field of public transportation prediction.

In [51], authors employ a standard, Back Propagation Neural Network (BPNN) [52] for traffic forecast purposes. Geo-spatial bus data from the city of Guangzhou as well as weather report information have been here exploited. The obtained results are satisfying and, through a ten-fold cross-validation experimental procedure, the model demonstrates its suitability for traffic forecast purposes.

Zheng et al. [53] introduce the so-called Bayesian-combined neural network (BCNN), which combines radial basis functions and a BPNN. The network assigns a credit value to each predictor and adapts accordingly its behavior. The obtained results show that the hybrid model outperforms a single neural network most of the time.

More recently, a new deep-learning-based traffic flow prediction method [54] has been proposed, which inherently takes into account both spatial and temporal correlations. A stacked auto-encoder, trained in a greedy, layer-wise fashion, is here used to learn generic traffic flow features.

In [55], the proposed method focuses on building a framework not for “normal” but for what they call “extreme” traffic conditions. A Deep LSTM (Long Short Term Memory) neural network is here used to forecast peak hours and identify unique characteristics for the traffic data. Further enhancing the model for post-accident forecasting by mixing Deep LSTMs provides a joint model for both normal traffic conditions and patterns of accidents. The experimental evaluation shows significant improvements over the baseline.

Ma et al. [48] propose a Convolutional Neural Network (CNN)-based method that predicts large-scale, network-wide traffic speed. The proposed method is benchmarked against 4 different predictive methods, including ordinary least squares, k-nearest neighbors, artificial neural networks and random forests. Concerning the neural networks, three deep learning architectures, including a stacked auto-encoder, a recurrent neural network, and a LSTM have been retained. The obtained results show that the CNN-based approach manages to predict the traffic more accurately.

In a less conventional manner, in [56] authors model the traffic flow as a diffusion process performed on a directed graph, with the help of the so-called DCRNN (Diffusion Convolutional Recurrent Neural Network) approach. The method incorporates both spatial and temporal dependencies of the traffic flow. The spatial dependencies are captured with the help of bidirectional random walks on the graph. The temporal dependency is taken into account by using the encoder-decoder architecture with scheduled sampling. The framework has been evaluated on two real-world datasets and shows significant improvement over the state of the art.

C. CONSUMER APPLICATIONS AND FRAMEWORKS
Some mainstream, conventional commercial applications have also integrated real-time traffic conditions and provide prediction facilities for public transportation. Among the most popular solutions, let us cite Google Maps [57], Bing Maps [58] and Citymapper [59], which have been widely adopted by the general public. Among other applications, let us also mention some local applications for specific cities, like Transilien [60] for Paris, MVV [61] for Munich, or RTC [62] for Las Vegas.

Even though such applications and software solutions are widely available and highly accepted, there is still a lot to be done (particularly for buses) in terms of accuracy for both long- and short-term prediction.

The following section details the considered simulation scenario considered in our work.

III. SIMULATION SCENARIO
The simulation scenario has been developed and computed with the help of the SUMO traffic simulation framework.

In order to set up and evaluate the predictive algorithms in a manner that is close to real-life conditions, the scenario needs to be carefully planned and executed. Our initial goal was to create a scenario that is complex, close to real-life situations, not excessively computationally heavy, and that can still leave space for future extensions and improvements.

More precisely, two real bus lines from the bus network of the French city of Nantes have been considered. The bus lines retained are numbered by 79 and 89, and lead to 4 different itineraries in total (both directions for each line), with 25 and 36 bus stops, respectively. The real bus stop locations have been recovered from the TAN (Transports de l’Agglomération Nantaise) [63] public traffic provider in Nantes and with the help of the OSM (Open Street Map) transport editor [64]. The retained bus itineraries are illustrated in Figure 1.

Figure 1 depicts the two bus lines (79 and 89), which are located in the same geographical area. This choice allows us to create a simulation in such a way that the bus lines are computed in a single simulation instead of two different ones. So,
from the beginning, the simulation time and computational effort are divided by two.

The simulation scenario has been prepared, created and executed with the SUMO traffic simulator. More precisely, the following parameters have been considered:

- The 2D map of the considered region in the city of Nantes, imported from OSM (Open Street Map) [64], has been first converted to the SUMO format.
- The total number of simulations performed was set to 4000.
- Each simulation performs 3 bus runs per bus itinerary in different time slots, as illustrated in Figure 2. Then, each bus run per simulation is considered as a separate run. This leads to a total number of 4000 simulations × 3 buses = 12000 bus runs.
- The simulation is controlled by a global macro-parameter, which is the total number of vehicles inserted into the system (i.e., the total number of both public and private vehicles that are present in the city). In order to simulate various traffic conditions, this parameter ranges within the [11000, 18000] interval with a random sample step (between 1 and 10) of vehicles.
- Each vehicle itinerary is calculated using the shortest path Dijkstra algorithm [65], applied on the O/D (origin/destination) matrix.
- The total number of pedestrians was set for all simulations to 3600. Let us note that it is important to use pedestrians in the simulation due to the impact of pedestrian traffic lights. This makes it possible to increase the degree of realism of the entire simulation.
- The bus waiting time at each bus stop station was fixed to 20 seconds.
- The time for each bus run was limited to 4000 seconds since after extensive measurement it was concluded that 99% of all bus runs complete the itinerary under 4000 seconds.
- The position of each bus was sampled every 10 seconds.
- The total simulation time was set to 19000 seconds, which ensures that all the vehicles will enter and exit the system, as illustrated in Figure 2.

Please note that the simulation time considered ensures that the simulations are performed within a certain stability range, where the number of total vehicles in the system is relatively stable. This aspect is illustrated in Figure 2, where the actual, real number of vehicles that are present in the system over time is presented. We can observe that, in the beginning, the number of vehicles is gradually increasing, the vehicles being successively inserted within the system. On the contrary, at the end of the simulation interval, the number of vehicles is rapidly decreasing, since the vehicles that are going out of the system are not replaced by new ones. This behavior is due to the intrinsic functioning mode of the SUMO simulator. A stability range, with a relatively constant number of vehicles that are present in the system, is achieved within the time interval [2500 - 15000] seconds. For this reason, for a given simulation, 3 different buses are launched for each of the 4 itineraries at the following starting times: 3000, 7000 and 11000 seconds. This makes it possible to obtain consistent simulations.

Once the simulation data available, we have started to investigate the prediction techniques. In a first stage, we have conducted a relatively simple experiment, concerning a global prediction approach. The question that we have addressed here is the following: given the total number of vehicles inserted in the system, would it be possible to predict, for each bus, the time of arrival at the terminus station (i.e., time of completion of the itinerary)?

A priori, it is a reasonable assumption that knowing the total number of vehicles in the system we can successfully predict the bus arrival time at the final destination.

In order to investigate the validity of this assumption, we have studied the potential correlation between the time of completion of a given itinerary and the total number of vehicles inserted in the system. The results obtained, presented in Figure 3, are quite surprising.

The itinerary completion times are varying in a chaotic manner, and no correlation between the total number of vehicles within the system and the time of completion of the bus itinerary can be established.
We can observe that there are huge disparities from one simulation to another even when the number of vehicles is increased by small units. Moreover, in some cases, the time of completion of the itinerary is inferior when a significantly superior number of vehicles is present in the system. This phenomenon can be explained by the fact that a certain number of singularities can appear. They correspond to traffic jams occurring locally in some parts of the region under analysis. Such singularities are completely out of control, and they depend more on the initial, random distribution of the vehicles instead of the global number of vehicles that are present in the system.

This situation is illustrated in Figure 4. Here, the same round-about is perfectly fluid when a total number of 15450 vehicles (4/A) is present in the system, but completely saturated for a lower number of 11990 vehicles (4/B).

This initial analysis shows that attempting to perform a global prediction of the time of completion of the itinerary, based on the global number of vehicles is not possible. Considering the initial simulation results and such an irregular traffic behavior it is clear that a more elaborate and finely granular solution is necessary in order to be able to perform relevant predictions. The analysis of the results also confirms that no reliable prediction can be performed when solely a global parameter is taken into account. This is mainly due to some singular events, which mainly concern localized traffic jams hazardously occurring in some given points. It is then necessary to consider a finer level of granularity, in order to characterize the state of the system in a more local and reliable manner. The following section describes in greater detail the proposed TDM (Traffic Density Matrix) solution.

IV. TRAFFIC DENSITY MATRIX (TDM) AND DATA MODELS

In order to overcome the problem of global traffic prediction, a new solution is needed. This solution should consider a more granular approach, which can offer a localized representation of the traffic over time.

Each simulation is used to create a specific, image-like TDM (Traffic Density Matrix) data structure [66] that later on can be transformed in different data models for prediction purposes. The TDM concept plays a central role in our approach and is detailed in the following section.

A. TRAFFIC DENSITY MATRIX (TDM)

The Traffic Density Matrix (TDM) is illustrated in Figure 5. For each bus generated at each simulation (total number of simulations $S_{total} = 12000$), we create a local density matrix per simulation, denoted by $iTDM$, of size $(M_{stations} \times T)$, where $M_{stations}$ is the number of measurement stations and $T$ is the number of time instances measured. In order to simplify and speed-up the simulation process, we have evenly sampled the simulation interval with a step of 10 seconds. This helps reducing the amount of data and consequently the related storage/computational requirements.

The simulation yields the following two outputs, controlled by a global parameter which is the number of vehicles injected within the system:

- A vector $a = (a_1, a_2, a_{M_{stations}})$ of size $M_{stations}$, storing the bus arrival times in all the stations of the itinerary.
- The individual density matrix $iTDM$, of size $(M_{stations} \times T)$.

In our case, the value $M_{stations}$ is 25 and 36 for buses 79 and 89 respectively. Parameter $T$ defines the number of instant measurements. For a simulation time of 4000 seconds, sampled every 10 seconds, we obtain $T = 400$. 

We can observe that there are huge disparities from one simulation to another even when the number of vehicles is increased by small units. Moreover, in some cases, the time of completion of the itinerary is inferior when a significantly superior number of vehicles is present in the system. This phenomenon can be explained by the fact that a certain number of singularities can appear. They correspond to traffic jams occurring locally in some parts of the region under analysis. Such singularities are completely out of control, and they depend more on the initial, random distribution of the vehicles instead of the global number of vehicles that are present in the system.

This situation is illustrated in Figure 4. Here, the same round-about is perfectly fluid when a total number of 15450 vehicles (4/A) is present in the system, but completely saturated for a lower number of 11990 vehicles (4/B).

This initial analysis shows that attempting to perform a global prediction of the time of completion of the itinerary, based on the global number of vehicles is not possible. Considering the initial simulation results and such an irregular traffic behavior it is clear that a more elaborate and finely granular solution is necessary in order to be able to perform relevant predictions. The analysis of the results also confirms that no reliable prediction can be performed when solely a global parameter is taken into account. This is mainly due to some singular events, which mainly concern localized traffic jams hazardously occurring in some given points. It is then necessary to consider a finer level of granularity, in order to characterize the state of the system in a more local and reliable manner. The following section describes in greater detail the proposed TDM (Traffic Density Matrix) solution.

IV. TRAFFIC DENSITY MATRIX (TDM) AND DATA MODELS

In order to overcome the problem of global traffic prediction, a new solution is needed. This solution should consider a more granular approach, which can offer a localized representation of the traffic over time.

Each simulation is used to create a specific, image-like TDM (Traffic Density Matrix) data structure [66] that later on can be transformed in different data models for prediction purposes. The TDM concept plays a central role in our approach and is detailed in the following section.

A. TRAFFIC DENSITY MATRIX (TDM)

The Traffic Density Matrix (TDM) is illustrated in Figure 5. For each bus generated at each simulation (total number of simulations $S_{total} = 12000$), we create a local density matrix per simulation, denoted by $iTDM$, of size $(M_{stations} \times T)$, where $M_{stations}$ is the number of measurement stations and $T$ is the number of time instances measured. In order to simplify and speed-up the simulation process, we have evenly sampled the simulation interval with a step of 10 seconds. This helps reducing the amount of data and consequently the related storage/computational requirements.

The simulation yields the following two outputs, controlled by a global parameter which is the number of vehicles injected within the system:

- A vector $a = (a_1, a_2, a_{M_{stations}})$ of size $M_{stations}$, storing the bus arrival times in all the stations of the itinerary.
- The individual density matrix $iTDM$, of size $(M_{stations} \times T)$.

In our case, the value $M_{stations}$ is 25 and 36 for buses 79 and 89 respectively. Parameter $T$ defines the number of instant measurements. For a simulation time of 4000 seconds, sampled every 10 seconds, we obtain $T = 400$. 

We can observe that there are huge disparities from one simulation to another even when the number of vehicles is increased by small units. Moreover, in some cases, the time of completion of the itinerary is inferior when a significantly superior number of vehicles is present in the system. This phenomenon can be explained by the fact that a certain number of singularities can appear. They correspond to traffic jams occurring locally in some parts of the region under analysis. Such singularities are completely out of control, and they depend more on the initial, random distribution of the vehicles instead of the global number of vehicles that are present in the system. 

This situation is illustrated in Figure 4. Here, the same round-about is perfectly fluid when a total number of 15450 vehicles (4/A) is present in the system, but completely saturated for a lower number of 11990 vehicles (4/B).

This initial analysis shows that attempting to perform a global prediction of the time of completion of the itinerary, based on the global number of vehicles is not possible. Considering the initial simulation results and such an irregular traffic behavior it is clear that a more elaborate and finely granular solution is necessary in order to be able to perform relevant predictions. The analysis of the results also confirms that no reliable prediction can be performed when solely a global parameter is taken into account. This is mainly due to some singular events, which mainly concern localized traffic jams hazardously occurring in some given points. It is then necessary to consider a finer level of granularity, in order to characterize the state of the system in a more local and reliable manner. The following section describes in greater detail the proposed TDM (Traffic Density Matrix) solution.

IV. TRAFFIC DENSITY MATRIX (TDM) AND DATA MODELS

In order to overcome the problem of global traffic prediction, a new solution is needed. This solution should consider a more granular approach, which can offer a localized representation of the traffic over time.

Each simulation is used to create a specific, image-like TDM (Traffic Density Matrix) data structure [66] that later on can be transformed in different data models for prediction purposes. The TDM concept plays a central role in our approach and is detailed in the following section.

A. TRAFFIC DENSITY MATRIX (TDM)

The Traffic Density Matrix (TDM) is illustrated in Figure 5. For each bus generated at each simulation (total number of simulations $S_{total} = 12000$), we create a local density matrix per simulation, denoted by $iTDM$, of size $(M_{stations} \times T)$, where $M_{stations}$ is the number of measurement stations and $T$ is the number of time instances measured. In order to simplify and speed-up the simulation process, we have evenly sampled the simulation interval with a step of 10 seconds. This helps reducing the amount of data and consequently the related storage/computational requirements.

The simulation yields the following two outputs, controlled by a global parameter which is the number of vehicles injected within the system:

- A vector $a = (a_1, a_2, a_{M_{stations}})$ of size $M_{stations}$, storing the bus arrival times in all the stations of the itinerary.
- The individual density matrix $iTDM$, of size $(M_{stations} \times T)$.

In our case, the value $M_{stations}$ is 25 and 36 for buses 79 and 89 respectively. Parameter $T$ defines the number of instant measurements. For a simulation time of 4000 seconds, sampled every 10 seconds, we obtain $T = 400$. 

We can observe that there are huge disparities from one simulation to another even when the number of vehicles is increased by small units. Moreover, in some cases, the time of completion of the itinerary is inferior when a significantly superior number of vehicles is present in the system. This phenomenon can be explained by the fact that a certain number of singularities can appear. They correspond to traffic jams occurring locally in some parts of the region under analysis. Such singularities are completely out of control, and they depend more on the initial, random distribution of the vehicles instead of the global number of vehicles that are present in the system.

This situation is illustrated in Figure 4. Here, the same round-about is perfectly fluid when a total number of 15450 vehicles (4/A) is present in the system, but completely saturated for a lower number of 11990 vehicles (4/B).

This initial analysis shows that attempting to perform a global prediction of the time of completion of the itinerary, based on the global number of vehicles is not possible. Considering the initial simulation results and such an irregular traffic behavior it is clear that a more elaborate and finely granular solution is necessary in order to be able to perform relevant predictions. The analysis of the results also confirms that no reliable prediction can be performed when solely a global parameter is taken into account. This is mainly due to some singular events, which mainly concern localized traffic jams hazardously occurring in some given points. It is then necessary to consider a finer level of granularity, in order to characterize the state of the system in a more local and reliable manner. The following section describes in greater detail the proposed TDM (Traffic Density Matrix) solution.
A global 3D matrix, denoted by TDM is finally constructed, with size \( M_{\text{stations}} \times T \times S_{\text{total}} \), where \( S_{\text{total}} \) is the total number of simulations.

In order to compute and successfully predict the bus arrival time at each bus stop station, a suitable data model needs to be developed. In our case, we have considered two different data models, a first one dedicated to the operators (ODM – Operator Data Model) and a second one focusing on the client needs (CDM – Client Data Model). They are described in the following sections.

### B. OPERATOR DATA MODEL (ODM)

The operator data model (ODM) is structured in a manner that attempts to maximize the input information in the learning process. The ODM data model uses the Traffic Density Matrix as a whole input.

Thus, the ODM contains the TDM information from all 12000 simulations under the form of a 3D matrix, illustrated in Figure 6. The horizontal axis presents the bus stations \( (M_{\text{stations}}) \) that are used to measure the traffic density on the itinerary. The vertical axis is the temporal one and presents the time in seconds (with a 10 seconds sampling unit). The last axis represents the number of simulations, which in our case was set to 12000. Basically, the TDM stores an integer number for each measurement station at a certain period of time. Each individual simulation is stored in an image-like structure called iTDM. The so-called iTDM contains the spatiotemporal data \( (M_{\text{stations}} \times T) \) obtained per individual simulation.

The objective is then to predict the arrival vector \( a \), given as input the iTDM. The input data is a matrix of size \( (M_{\text{stations}} \times T) \), and the output is a vector of size \( M_{\text{stations}} \) (with \( M_{\text{stations}} \) being equal to 25 and 36, for buses 79 and 89, respectively), containing the bus arrival times at all the bus stop stations.

### C. CLIENT DATA MODEL (CDM)

The client data model (CDM) concerns the prediction of the bus arrival time at a given station, based on the current position of the bus over the itinerary and taking into account the traffic situation of the whole itinerary, as illustrated in Figure 7.

In order to develop the client data model, some additional data processing and development must take place. More precisely, for this particular predictive scenario, data modeling has been created in the following way:

- 5 different bus stop stations have been selected, evenly sampling the itinerary, and denoted by \( R_{\text{stop}} \). In the case of the bus line 89, the bus stops no 11, 16, 21, 26 and 31, evenly sampling the itinerary, are considered for predicting the corresponding bus arrival times, so \( R_{\text{stop}} \in \{11, 16, 21, 26, 31\} \).
- For each selected bus stop, 10 different bus runs are extracted from the TDM. This leads to a total number of 12000 (simulations) \( \times (R_{\text{stop}} = 5) \times 10 \) (runs) = 600000 inputs.

The first part of the CDM data model concerns the vectors generated from the TDM. The bus runs are stored in the so-called called input vector, illustrated in Figure 8. A second value is a scalar one, called \( R_{\text{stop}} \), which corresponds to the current bus stop station \( (i.e., \text{the bus station where the user requested the information for}) \). The last part is also a scalar value denoted by \( C_{\text{stop}} \) (closest bus stop). This station is defined as the bus stop station that is the closest to the current bus position at the given, requested time \( t_{\text{crt}} \).

During the construction of the CDM data model, we make sure to retain only valid trials, \( i.e., \text{values for which in the simulations, at the current time, there exists a bus prior to the current bus stop location. In this way, we make sure to retain valid runs for which the C}_{\text{stop}} \) (current bus position) is anterior to the considered bus station where the prediction is required \( (R_{\text{stop}}) \).

To summarize, the input vector (Figure 8) includes three different variables, including the TDM_{vector} (which includes the real traffic information over the whole itinerary), the C_{stop} for the current time \( t_{\text{crt}} \), and the R_{stop} (the bus stop station used to predict the bus arrival time). This input vector is used for predicting the bus arrival time at the requested station. The output value is a scalar, representing the expected arrival time, denoted by \( b \), in seconds.
In order to solve such a problem, we have considered various machine learning techniques and algorithms, described in the following section.

V. RETAINED MACHINE LEARNING APPROACHES

The data obtained from the simulation process needs to be prepared in a suitable format called the data model. The data containing the traffic information under the form of the TDM matrix, also called input data (in the machine learning process) was processed and stored in either 2D (per individual simulation – iTDM matrix) or 3D (for all simulations – TDM matrix) data structures, as discussed in the previous section.

The bus arrival time at each bus stop station (which represents the output data) was stored either as a 1D vector or as a scalar value. It contains the timestamps as values of each bus arrived at the bus stop stations.

In order to reduce the data size and the corresponding memory requirements, all the data was compressed with the MsgPack [67] standard library.

The next step is of crucial importance for the learning stage. It involves the data split and the cross-validation, which were achieved as follows:

- Total number of simulations 12000.
- Training dataset - 80 % (9600 simulations).
- Test dataset – 20 %, (2400 simulations), from which: 10 % (240) were used for validation, and 90 % (2160) for evaluation purposes.

A data normalization procedure was included based on the following equation (1):

\[ \gamma_{\text{normalized}} = \frac{\gamma - \mu}{\sigma} \]

where:

- \( \gamma \): the data considered (denoting both the TDM matrix \( D \) and and the arrival time vector \( a \)),
- \( \mu \): is the mean of \( (\gamma) \),
- \( \sigma \): is the standard deviation of \( (\gamma) \).

The data normalization procedure has two benefits. On one hand, in many cases, it makes it possible to speed up the convergence rate of the training procedure. On the other hand, in the case of neural networks-based approaches, it allows increasing the number of neurons in the hidden layers. For this reason, the data normalization procedure will be used in all of the machine learning techniques considered, that are detailed in the following sections.

A. OLS

The first, basic machine learning technique that we have adopted and implemented is the Linear Regression [43] also known as OLS (Ordinary Least Squares). Here, we have used the implementation available in the Scikit-learn machine learning toolbox [68].

The OLS approach fits a linear model to minimize the residual sum of squares between the observed targets in the considered dataset.

Ordinary Least Squares (OLS) is the simplest machine learning technique retained.

B. SVR

The SVR approach [69] extends in a certain sense the linear regression, with two fundamental differences. A first one concerns the concept of margin, defined as the distance between support vectors (i.e., vectors that are closest to the decision boundary) and the decision hyperplane. The SVR approach maximizes the margin, increasing in this way the robustness of the estimation and maximizing the generalization capability.

Another fundamental aspect related to SVR concerns the so-called kernel trick. The kernel functions make it possible to seamlessly map the initial data onto a higher dimensional feature space. In this way, problems that are not linearly separable in the original representation spaces become separable, while computations continue to be performed in the initial, lower dimensional space. Different kernel functions can be specified.

In our case, we have adopted the SVR implementation available in the Scikit-learn library and used it for both operator (ODM) and client (CDM) data models. Two different SVR kernels have been considered: RBF (Radial Basis Function), and Polynomial (2nd degree – quadratic kernel). We have not considered the SVR with a linear kernel since in this case, the results are equivalent to those achieved by the OLS technique.

C. FNN (FEEDFORWARD FULLY CONNECTED NEURAL NETWORK)

Historically, the so-called Feedforward Neural Networks are the oldest type of artificial neural networks that are designed to mimic the neural connections in the human brain.

A fully connected neural network connects all the possible combinations of neurons between successive layers. One important distinction is that the neurons in the same layer do not connect with each other. The number and the size of the hidden layers then define the network architecture.

Each connection between neurons is controlled by a weight parameter. The set of weights needs to be learned and governs then the behavior of the whole network. There are various techniques that make it possible to learn the weights based on a supposed available learning data set [20]. The objective is to minimize a loss function, measuring the error between the predictions performed by the neural network and the ground truth values from the learning data set.

In the case of feedforward Fully Connected Neural Network (FNN), two different, hand-crafted architectures have been considered, dedicated to the ODM and CDM data models respectively. The networks have been developed while considering the shape and structure of each individual data model.

The proposed FNN for the ODM case is illustrated in Figure 9. It includes 5 fully connected layers, with 4 hidden layers of sizes (1000 – 100 – 1000 - 100). A ReLU activation function was included.
FIGURE 9. The proposed FNN architecture for ODM.

function has been considered for all of them. The input layer iTDM is of size \((M_{\text{stations}} \times T)\) and the output is a vector of size \(M_{\text{stations}}\), representing the bus arrival times at all the bus stop stations considered.

For example, for bus 89, the iTDM is a 2D matrix of size \([36 \times 400]\), which corresponds to 36 measurement stations and 400 temporal instants. The output is a 36-valued vector that represents bus arrival times at each bus stop station.

In this particular configuration, the hidden layers aim at gradually reducing the dimensionality of the features, up to obtaining the output vector of size 36 (time at each bus stop station). The double alternation between 1000 and 100 layers makes it possible to spread and mix as much as possible the data between layers. Let us also note that adding supplementary hidden layers and thus obtaining a deeper architecture is possible. However, in our case, we have privileged a simpler architecture with only 4 hidden layers. As it will be shown in experimental results section, this is sufficient for obtaining accurate prediction results.

A MSE (Mean Square Error) loss function has been adopted and the SGD (Stochastic Gradient Descent) algorithm (with a learning rate of 0.001) used for learning. The number of epochs for training this model was set to 416 and the batch size to 10.

The second proposed FNN architecture concerns the CDM data model and is presented in Figure 10. It includes 3 hidden layers of size \((20 - 200 - 20)\). The ReLU activation function was also considered in this case. The input vector is of size \([1, M_{\text{stations}} + 2]\) and the output value is a scalar (time), representing the expected bus arrival time at the considered bus stop station.

The MSE metric was used as a loss function and the SGD (Stochastic Gradient Descent) algorithm (with a learning rate of 0.001) for optimization in the learning stage. The number of epochs was set to 25. This network is obviously smaller than the previous one due to the reduced complexity of the data.

D. CNN (CONVOLUTIONAL NEURAL NETWORK)

Convolutional Neural Networks (ConvNet / CNN) are a specific type of neural network architectures that are adapted for image-like structures. The specificity of the images comes from the high dimension of the data. It would be inconceivable to build fully connected neural networks for such data since this would require billions of neural connections and thus parameters that need to be learned.

In order to deal with such complexity-related issues, the principle consists of replacing the fully-connected layers by a set of convolution operators that govern the interactions between successive layers in the network. Each convolution operator is defined by a filtering kernel function, that has to be learned, and that replaces the weights encountered in the case of fully connected networks. The number of convolutional filters used at a given layer defines the depth of the network at the considered layer.

Additionally, pooling operators can be used in order to lower the dimensionality of the data. Such pooling operators correspond to sub-sampling processes and allow also to achieve invariance with respect to different forms of variability in the input data.

A Convolutional Neural Network (CNN) has also been proposed. Particularly adapted for an image-like structure, the CNN aims at taking into account the particular structure of the iTDM data. In particular, let us underline that it makes fully sense to perform convolutions on both dimensions (time and number of stations) of the iTDM structure, as illustrated in Figure 11.

Here, 4 different randomly selected simulations are presented. Each individual grayscale image (iTDM) depicts the traffic condition of one simulation, which is used as input data for the ODM data. The Y-axis represents the measurement stations, while the X-axis concerns the temporal dimension (in seconds).

The proposed CNN architecture is presented in Figure 12. This particular CNN network includes two successive convolutional layers (denoted by Conv 1 and Conv 2), followed by one maxPool layer, a third convolutional layer (Conv 3) and finally three fully connected layers, at the end of the chain. The corresponding parameters are the following:

- The kernel sizes of the three convolutional layers are of \((18, 2)\), with 8, 16, and 8 channels respectively.
FIGURE 11. Input data, traffic data as an image-like structure iTDM.

FIGURE 12. Proposed CNN architecture [70].

- The max-pooling layer is of size (4, 2).
- The two fully connected layers are of sizes 1000 and 100.
- The output layer is of size 10 stations.

Here are some architectural decisions that we have considered when designing this particular CNN architecture. The specific kernel size was chosen because of the particular elongated shape of the image/data (iTDM matrix) used for the ODM data model. The inclusion of the fully connected layers at the end of the chain is a commonly used step, necessary to go down from the convolutional layer to the output decision layer. The padding parameter was set to 0 because the traffic information at the beginning and the end of the image is not essential for the prediction process.

The MSE measure was used as a loss function and the SGD (Stochastic Gradient Descent) algorithm (with a learning rate of 0.001) has been considered for the learning stage. The algorithm was trained on 105 epochs.

E. RNN (RECURRENT NEURAL NETWORK) AND LSTM (LONG-SHORT TERM MEMORY)

Recurrent Neural Network (RNNs) represent a family of ANNs specifically designed to recognize the data sequential characteristics. They use the output from the previous step to be fed as an input to the current step, as illustrated in Figure 13(a). This feature makes them algorithms with internal memory.

On the other hand, Long Short-Term Memory (LSTM), like the name suggests is a special kind of recurrent network, capable of capturing long-term dependencies in the data. First introduced in 1997 by Hochreiter and Schmidhuber [71], the LSTMs have been recently popularized and refined. They were designed specifically to avoid long-term dependency problems, by remembering the information for a longer period of time.

Similarly, to RNNs, the LSTMs also have a chain-like structure, but the repeating model has a different structure, as illustrated in Figure 13(b). Here, four neural network layers are interacting in a specific way.

The equations that govern the forward pass of an LSTM are the following:

\[
\begin{align*}
    f_t &= \sigma (W_f \cdot [h_{t-1}, x_t] + b_f) \\
    i_t &= \sigma (W_i \cdot [h_{t-1}, x_t] + b_i) \\
    O_t &= \sigma (W_o \cdot [h_{t-1}, x_t] + b_o) \\
    \tilde{C}_t &= \tanh (W_C \cdot [h_{t-1}, x_t] + b_C) \\
    C_t &= f_t \ast C_{t-1} + i_t \ast \tilde{C}_t \\
    h_t &= O_t \ast \tanh (C_t) \\
\end{align*}
\]

where:
- \(x_t\) : input vector to LSTM unit,
- \(C_t\) : cell state vector,
- \(h_t\) : hidden state vector also known as output vector of the LSTM unit,
- \(f_t\) : forget gate activation vector,
- \(i_t\) : input gate activation vector,
- \(O_t\) : output gate activation vector,
- \(W\) : weight matrix,
- \(b\) : bias vector.

The core concepts behind the LSTMs concern the cell states and their various gates. In theory, the cell state can carry relevant information through the entire processing of the sequence (Figure 13(b)). One LSTM cell consists of four gates as follows:

- **Forget gate**, after getting the input from previous state \(h_{t-1}\) takes the decision what must be removed or forget, thus keeping only the relevant information.
- **Input gate**, add new information form the input to the preset cell.
D. Panovski, T. Zaharia: Long and Short-Term Bus Arrival Time Prediction With TDM

FIGURE 14. Proposed RNN architecture.

FIGURE 15. Proposed LSTM architecture.

- **Update (g) gate**, tanh layer that creates a vector for new candidate $C_t$.

- **Output gate**, provides the output from the cell state.

RNNs and LSTMs are machine learning algorithms that have unique characteristics and capabilities that can be exploited for our purposes.

The configuration of the RNN approach proposed is presented in Figure 14.

The proposed RNN configuration includes 2 RNN layers of size 200 and 2 fully connected layers of size (200 - 100) with ReLU as an activation function, and output layer (36).

The SGD (Stochastic Gradient Descent) algorithm (with a learning rate of 0.001) has been considered for the learning stage. The number of trained epochs was 166 and the batch size was 10.

The adaptation of the LSTM architecture to our particular objectives is presented in Figure 15. Here, each LSTM layer corresponds to the representation presented in Figure 13(b).

The proposed LSTM configuration includes 2 LSTM layers of size 300, 3 fully connected layers of size (300 - 200 - 100) and the output layer (of size $M_{\text{stations}}$).

The SGD (Stochastic Gradient Descent) algorithm (with a learning rate of 0.01) has been considered for the learning stage. The number of trained epochs was 125 and the batch size was 10.

Let us now present the experiments conducted and results obtained.

VI. EXPERIMENTAL RESULTS

The experimental evaluation has been carried out over both ODM and CDM data models. In order to randomly and evenly sample the data, a data distribution per simulation with an appropriate train/test data split needs to be performed. The considered data partition is illustrated in Figure 16.

The total number of simulations was set to 4000. Each simulation was computed with a random number of vehicles in the system, within the range [11000, 18000]. On the other hand, the randomness of the data split function was calculated with 10-fold cross-validation (80% train and 20% test), as previously explained. Here, we may notice that the data split and the vehicle distribution functions have been performed in a uniform manner. This is an important issue because the uniformity of the data is crucial for avoiding any potential biases.

A. EXPERIMENTAL RESULTS FOR ODM

In the first stage, we have considered for evaluation the OLS and SVR (with different kernels) approaches. The algorithms have been used for predicting the bus arrival times at each bus stop station (long-term prediction) for both bus lines 79 and 89.

In a first example, Figure 17 plots the predicted bus arrival times per station, under various traffic conditions (which are controlled by global the number of vehicles inserted in the system), for both bus lines.

The plot in the upper-left corner (red boundary box) represents bus line 79. In this case, the OLS approach...
TABLE 2. OLS prediction results for bus 79 and 89 both directions.

|          | 79 Beau > | 79 Ovra > | 89 Beau > | 89 Le Cr > |
|----------|------------|------------|------------|------------|
| MAE      | 4.96       | 3.18       | 134.18     | 133.2      |

OLS = Mean Absolute Error (s), Beau = Beausejour, Ovra = Ovrault, Le Cr = Le Cardo

TABLE 3. Prediction results for bus 89 beausejour > le cardo.

|          | OLS       | SVR Rbf   | SVR Poly   |
|----------|-----------|-----------|------------|
| MAE      | 134.18    | 71.23     | 72.06      |

OLS = Mean Absolute Error (s), OLS = Ordinary Least Squares, Poly = Polynomial, Rbf = Radial Basis Function

is perfectly capable of predicting accurately the bus arrival time per station.

This behavior for bus 79 has been confirmed by all the other experiments that we have conducted, under various traffic conditions. On this itinerary, no major traffic congestion zones (traffic jams), which may cause problems, are encountered and a linear approach perfectly manages to predict correctly the arrival times.

To further confirm this claim, let us refer to Table 2, which summarizes the MAE scores, globalized over the entire test data set. In the case of bus 79, the obtained MAE scores range between 3 and 5 seconds, which corresponds to a highly accurate prediction.

However, the behavior is completely different in the case of bus line 89. As we can see from Figure 17 (plots in the blue boundary box), the predicted curve deviates from the ground true significantly (Table 2) with an MAE score of 133 and 134 seconds (for the two directions, respectively). A finer analysis allowed us to understand what happens in this case. Actually, some very punctual, localized traffic jams are occurring in some intersections and are responsible of the stair-wise character of the curves.

The OLS approach is too elementary to take into account such complex, non-linear behavior. As a result, the prediction fails.

Let us now examine, for the same bus line 89, the behavior of the SVR approaches, when compared to the OLS technique. The obtained global MAE scores are reported in Table 3.

The obtained results show that the MAE in the case of OLS is significantly higher than those obtained by the SVR approaches. The performance of SVR decreases the corresponding MAE down to 71 seconds, which by itself is impressive. This result represents almost a 47% error decrease, with respect to OLS.

The analysis of these first results makes us think that more complex and highly non-linear approaches are necessary for taking into account such localized traffic jams events. In order to validate this intuition, let us analyze the results obtained by the various deep learning approaches (FNN, CNN, RNN and LSTM) considered. They are presented in Figure 18, for different groups represented by a letter (from A to H). Such groups differ one from another by the total number of vehicles inserted in the system, indicated next to the small car sign that is present in the bottom right corner of each plot.

The predicted times are expressed in seconds and the considered bus line is number 89, direction Beausejour – Le Cardo. The obtained results show that, globally, all the considered approaches yield good prediction results. In most of the cases, the FNN technique performs best and is closest to the ground truth. However, in some cases, like those illustrated in Figure 18(B), RNN shows better results than FNN. In Figure 18(H), the LSTM techniques show better performances than FNN. Finally, there are some cases, such as those illustrated in Figure 18(E), where the best performer is the CNN.

In order to objectively present and compare the results, in addition to the MAE score, we have considered the following set of supplementary criteria:

- minMAE presents the lowest MAE value that is achieved by the algorithm compared to the ground truth, while the maxMAE is the opposite (highest MAE value),
- The MEDIAN MAE value,
- The MAE standard deviation (STD). A low standard deviation indicates that the values tend to be close to the average of the set, while a high standard deviation indicates that the values are spread out over a wider range.
TABLE 4. Prediction results for bus 89 Beausejour > Le Cardo.

| Method      | OLS  | SVR  | FNN  | CNN  | RNN  | LSTM  |
|-------------|------|------|------|------|------|-------|
| MAE         | 134.2| 72.1 | 44.6 | 59.4 | 65.5 | 56.4  |
| minMAE      | 12.3 | 5.9  | 3.4  | 5.5  | 10.7 | 3.3   |
| maxMAE      | 1563 | 1024.8| 973.8| 782.7| 983.3| 1084  |
| MEDIAN      | 109.8| 53.6 | 16.2 | 33.5 | 36.6 | 35.1  |
| STD         | 109.8| 67.5 | 73.6 | 73.1 | 75.2 | 73.9  |
| cTime       | 818  | 12193| 1500 | 1569 | 1654 | 1332  |
| EPOCHs      | -    | -    | 416  | 104  | 166  | 125   |

MAE = Mean Absolute Error (s), minMAE = Minimum MAE, maxMAE = Maximum MAE, MEDIAN = Separating higher from lower half in dataset, STD = Standard deviation, cTIME = Computational Time (s), EPOCHs = One whole dataset iteration, OLS = Ordinary Least Squares, SVR = Support Vector Regression, FNN = Feedforward Fully connected Neural Network, CNN = Convolutional Neural Network, RNN = Recurrent Neural Network, LSTM = Long-Short Term Memory

- cTime, denoting the computational time of the learning stage,
- EPOCH is the number of iterations that requires a full pass through the whole data set.

Table 4 summarizes the performances of the various machine/deep learning techniques considered, according to this set of criteria.

The lowest MAE (44.6 seconds) is by far achieved by the FNN algorithm, while the highest by OLS (134.1 seconds). The FNN approach also leads to the lowest median MAE value, which is quite remarkable (16.2 seconds).

The lowest minMAE number is achieved by the LSTM technique (3.3 seconds) and the minimum maxMAE is achieved by the CNN approach (782.6 seconds). Concerning the standard deviations obtained, they are quite equivalent to all the algorithms considered, with the exception of the OLS approach, which is significantly higher.

In order to better illustrate these results, Figure 19 presents the MAE distributions obtained over the various simulations performed.

Here, the dotted vertical lines illustrate the median MAE values. The best median is achieved by FNN (16.2), which is an incredible result. This means that in 50% of the obtained results, the MAE is lower than 17 seconds.

In addition, Figure 20 presents the cumulative MAE histogram obtained by the various approaches retained. The vertical drop (line) corresponds to the maximum MAE obtained for each of the methods involved.

To benchmark the performance of each algorithm in terms of computational time and effort, we have considered the cTime parameter (Table 4), which provides the time requested for completing the learning stage. The best computational performance cTIME is achieved by OLS (with a high penalty of overall performance), with 818 seconds. The LSTM approach comes in the second position, with 1332 seconds. Let us also note that all the deep learning approaches considered are significantly more efficient in terms of computational effort (with cTIME values ranging in the [1332-1659 seconds] interval, than the SVR techniques (with cTIME values exceeding 12000 seconds).

The EPOCH parameter presents the total number of times where the whole dataset was browsed during the learning stage. The smallest number of epochs to train successfully the model was CNN with 104 epochs in total, as illustrated in Figure 21. Here, the learning curves between different models and their performance are also presented.

The training loss is presented with the red line and the validation loss is represented by the blue dotted line. The training curve presents the learning performance of the model, which is easy to observe (the convergence of the learning curve.). On the other hand, the validation curve is also important when a deep learning model is developed. It can serve to measure the evolution and performance of the model in real-time (very useful) and also can give an intuition if the model overfits the training data.

The deep learning curves presented in Figure 21, converge successfully for all the considered algorithms. The validation curve shows that there is no overfitting.

From all the presented results so far, a conclusion about the best machine learning algorithm is not obvious, since they present quite equivalent results. However, the best performer is the FNN approach, which offers the lowest MAE (44.6 seconds) and median MAE (16.2 seconds).
B. EXPERIMENTAL RESULTS FOR CDM

This section presents the experimental results obtained for the CDM (Client Data Model) [72]. Let us underline that in this case, the prediction window is very narrow in the moment of the observation (real-time).

For comparison, three different machine learning algorithms have been considered: OLS (Ordinary Least Squares), SVR (Support Vector Regression) with 2nd-degree polynomial kernel and FNN (Feedforward Fully Connected Neural Network).

The obtained prediction results are presented in Figure 22. Here, 5 different bus stop stations were considered for prediction, namely: 11th, 16th, 21st, 26th, and 31st. The corresponding MAE scores are presented in seconds and represent the error time of the bus arriving at the desired station. On the other hand (Table 5), the global MAE (gMAE) score measures the average MAE over the whole set of 5 bus stop stations. In addition, the cTime parameter presents the time needed to compute each model during the learning process.

Table 5 shows that the FNN approach yields the best short-term prediction results (with 15.76 % improvement over OLS and 7.62 % over SVR). The FNN performance is also better for all predicted bus stops, with the exception of only one instance which corresponds to the bus stop 11. Here, the SVR technique has an MAE of 43.5 seconds against 54.8 seconds for the FNN. This result can be explained by the fact that during the first bus stops, there is poorly probable to encounter traffic jam problems.

The best cTime needed for computing the algorithms during the learning process (time needed to learn and build the model from the data) is achieved by the OLS approach (with only 43 seconds). However, the OLS-related accuracy is the worst.

Figure 23 presents the learning and validation curves of the FNN method for the CDM data model. Here, the algorithm was trained for only 25 epochs, which was sufficient to successfully obtain reasonable performance.

Globally, we can observe that here again, the non-linear solutions are more appropriate for performing accurate predictions.

TABLE 5. CDM data model performance.

|     | 11   | 16   | 21   | 26   | 31   | gMAE | cTime |
|-----|------|------|------|------|------|------|-------|
| OLS | 73.5 | 212.7| 215.9| 244.0| 226.5| 194.52| 43    |
| SVR | 43.5 | 209.2| 208.3| 212.1| 214  | 177.42| 2985  |
| FNN | 54.8 | 191  | 183.1| 189.4| 201.2| 163.9 | 399   |

MAE = Mean Absolute Error, CDM = Client Data Model, OLS = Ordinary Least Squares, SVR = Support Vector Regression, FNN = Feedforward Fully Connected Neural Network, gMAE = global MAE, cTime = Computational Time.

VII. DISCUSSION AND SOA COMPARISON

In this article, we have elaborated and studied extensively the issue of public transportation prediction. The objective was to predict the bus arrival times at the bus stop stations, under the framework of both long-term (ODM) and short-term (CDM) prediction.

In order to further understand the performance of the algorithms, a comparative analysis with recently published algorithms is proposed.

Table 6 presents the results obtained by some recent research works for the long-term prediction of public transportation (buses). The approach proposed in this article is presented in the first row, in gray color.
TABLE 6. Comparative performance with different SoA approaches (odm long-term prediction).

| Article            | Year | Algorithms ML/DL | MAE   | Improve in % |
|--------------------|------|------------------|-------|--------------|
| OLS                |      | OLS              | 134.186 | 0            |
| SVR poly           |      | SVR              | 72.063 | 66.3%        |
| SVR rbf            |      | SVR              | 71.062 | 47.04%       |
| RNN                | 2020 | RNN              | 65.474 | 51.21%       |
| CNN                |      | CNN              | 59.389 | 55.74%       |
| LSTM               |      | LSTM             | 56.383 | 57.98%       |
| FNN                |      | FNN              | 44.629 | 66.74%       |
| Zhang and Liu [73] | 2019 | FVDM             | 28.04  | 4.79%        |
|                    |      | BPNN             | 78.11  | -160.23%     |
|                    |      | CK-means         | 19.27  | 34.57%       |
| Ho et al. [74]     | 2018 | HA               | 29.45  | 0            |
|                    |      | LR               | 4.220  | 0            |
|                    |      | SRV              | 3.684  | 12.7%        |
|                    |      | FCNN             | 4.093  | 3.01%        |
|                    |      | PCF              | 1.978  | 53.13%       |
| Treethidaphat et al. [75] | 2017 | OLS              | None   | 0%           |
|                    |      | FCNN             | None   | 55%          |
| Yamaguchi et al. [76] | 2018 | LR               | None   | 0%           |
|                    |      | ANN              | None   | 0%           |
|                    |      | RF               | None   | 0%           |
|                    |      | SVR              | None   | 0%           |
|                    |      | GBDT             | None   | 0%           |
| Yu et al. [77]     | 2017 | OLS              | 1.631  | 0%           |
|                    |      | RR               | 1.629  | 0.12%        |
|                    |      | LASSO            | 1.445  | 11.4%        |
|                    |      | FCNN             | 3.643  | -123.30%     |
|                    |      | LSTM             | 1.063  | 38.5%        |
|                    |      | Mix LSTM         | 0.970  | 40.53%       |
| Pan et al. [78]    | 2012 | HA               | None   | 0%           |
|                    |      | BPNN             | None   | 5.7%         |

TABLE 7. Comparison with SoA approaches (cdm short-term prediction).

| Article            | Year | Algorithms ML/DL | MAE   | Improve in % |
|--------------------|------|------------------|-------|--------------|
| CDM approach       | 2019 | LR               | 194.52 | 0%           |
|                    |      | SVR              | 177.42 | 8.79%        |
|                    |      | FNN              | 163.9  | 15.74%       |
| Liu and Xiao [79]  | 2019 | HA               | 78.2   | 0%           |
|                    |      | OPFA             | 71.67  | 8.35%        |
| Bohan and Yun [80] | 2019 | LSTM             | 0.682  | 0%           |
|                    |      | GRU              | 0.761  | 7.84%        |
| Fu et al. [81]     | 2016 | GRU              | 18.127 | 5.47%        |
|                    |      | LSTM             | 17.211 | 10.24%       |
| Wu and Tan [82]    | 2016 | GBRT             | 22.52  | 0%           |
|                    |      | FCNN             | 20.61  | 8.48%        |
|                    |      | SAЕ              | 20.36  | 9.59%        |
| Lam et al. [83]    | 2019 | SMA              | None   | 0%           |
|                    |      | ANN              | None   | 17%          |
|                    |      | IHM              | None   | 0%           |
| Wu et al. [84]     | 2018 | LASSO            | 29.586 | 0%           |
|                    |      | StoS             | 27.562 | 6.96%        |
|                    |      | SAЕ              | 27.173 | 8.16%        |
|                    |      | BBPN             | 25.672 | 13.23%       |
|                    |      | DNN-BTF          | 25.189 | 14.86%       |
| Huang et al. [52]  | 2019 | HOD              | None   | 0%           |
|                    |      | GA               | None   | 7.5%         |
|                    |      | LSTM             | None   | 14%          |

MAE · Mean Absolute Error, HA · Historical Average, LR · Linear Regression, RR · Ridge Regression, LASSO · Least Absolute Squares Regression, OLS · Ordinary Least Squares Regression, ARIMA · AutoRegressive Integrated Moving Average, SVR · Support Vector Regression, SVM · Sequence to sequence learning, SMA · Simple Moving Average, FVDM · Front Vehicle Data Model, PCF · Partitioning and Combination Framework, GBRT · Gradient Boosting Regression Tree, OME · Stacked AutoEncoder, GA · Genetic Algorithm, GBDT · Gradient Boosting Decision Tree, K-means · K means Clustering, OPFA · Optimized Particle Filtering Algorithm, Hybrid Model, ANN · Artificial Neural Network, DNN-BTF · Deep Neural Network Based Traffic Flow, GRU · Gated Recurrent Unit, FCNN · Fully Connected Neural Network, Feedforward Fully Connected Neural Network, BPNN · Back-Propagation Neural Network, LSTM · Long-Short Term Memory

The table is divided into 5 columns, including article citation, year of publishing, algorithms implemented, MAE values and achieved improvement. Each method has an improvement score expressed in percentages with respect to the reference baseline method (RBM) retained. The RBM techniques are indicated in red (and show a 0% improvement score over themselves).

A first observation is that the FNN proposed in this article objectively leads to the best performances, with an improved score of 66% over the baseline (in our case OLS). The other deep learning algorithms proposed perform similarly, with improvement over baseline around 55%. This result is approached solely by the FCNN-based method introduced by Treethidaphat et al. in [75]. Another remark is that in some cases, such as those reported by Zhang and Liu [73] and Yu et al. [77], the deep learning techniques perform very poorly, and even worse than the baseline approach. This can be due to some specific implementation or because of a lack of sufficient or diverse data.

As in the case of the ODM data model presented in the previous section, in order to further understand the performance of the algorithms, we propose a comparison with the recent state of the art techniques (Table 7) that are targeting short-term prediction of public transportation (buses). The CDM proposed in this article (our approach) is presented in the first row with gray color. The reference baseline method considered by each approach is represented in red.

Here, the performance of the proposed FNN deep learning technique shows a 15% improvement over the baseline. The highest achieved score was 17% improvement in Lam et al. [83], but this score was calculated against the SMA (Simple Moving Average) baseline method, which is not as powerful as OLS. The slight improvement of SVR and
FNN over the baseline OLS was expected since this data model (CDM) is limited in terms of the robustness of the input data. In this case, the data contains less information when compared with the ODM data model, so the model can learn less.

VIII. CONCLUSIONS AND PERSPECTIVES

In this work, we have considered the issue of public transportation prediction (buses in particular). Our main objective was to predict the bus arrival times at the bus stop stations (short- and long-term prediction).

All the proposed prediction methods are based on a novel concept, so-called TDM (Traffic Density Matrix) that was first introduced. The TDM technique introduces localized information about the traffic conditions in a given city area, with the help of a set of measurement stations, which capture the number of vehicles that are present in the vicinity of the considered measurement point. The resulting TDM data is presented under the form of an image-like structure that represents the evolving vehicle density over a period of time.

A real-life scenario has been conducted in a virtual environment, with the help of the SUMO traffic simulation platform. It concerns a sub-part of the city of Nantes, France for two real bus lines numbered by 79 and 89 (which leads to four itineraries in total). The total number of simulations was 4000 for both bus lines since the scenario was constructed in a way that allows us to simulate both lines in a single simulation.

In order to predict the bus arrival times in the stop stations, various techniques have been proposed and explored. Ordinary Least Square and Support Vector Regressors (with both Polynomial and RBF kernels) have been retained as baseline methods. Then, we have proposed a set of dedicated deep learning architectures, including a Feed-Forward Fully Connected Neural Network, a Convolutional Neural Network, a Recurrent Neural Network and a Long-Short Term memory.

The experimental results obtained showed that the best performances are achieved by the FNN approach, which yields the lowest MAE and median MAE scores among all of the considered techniques. In a general manner, all the deep learning techniques proposed outperform, in terms of prediction accuracy, the traditional OLS and SVR machine learning approaches. This shows that increasing the degree of non-linearity of the methods makes it possible to obtain superior results, in particular in the case where local singularities that are caused by traffic jams occur.

For future analyzes and development, we may explore different machine learning techniques, as well as different bus lines where the possibilities of traffic jams are highly possible. One interesting future possibility may concern the change of the location of the measurement station in the bus itinerary, from the bus stop stations to specific road intersections. More generally, the optimization of the number and position of the considered measurement stations is a promising axis of research.

Another interesting axis of future research concerns the inclusion within the prediction process of various other parameters, that may concern the population density in given areas and the degree of occupancy of the buses, or moreover the introduction of unexpected events (infrastructure works, accidents...).

ACKNOWLEDGMENT

This work has been carried out within the framework of the FUI (Fonds Unique Interministériel) ETS (Electronic Ticketing System) project, supported by the French Government.

REFERENCES

[1] Mars 2015 Chiffres clés du Transport, Ministère de la Transition écologique—Commissariat général au développement durable SOeS, 2015.
[2] S. An, B.-H. Lee, and D.-R. Shin, “A survey of intelligent transportation systems,” in Proc. 3rd Int. Conf. Comput. Intell., Commun. Syst. Netw., 2011, pp. 332–337.
[3] L. Qi, “Research on intelligent transportation system technologies and applications,” in Proc. Workshop Power Electron. Intell. Transp. Syst., Aug. 2008, pp. 529–531.
[4] N.-E. El Faouzi, “Data fusion in intelligent transportation systems: Progress and challenges—A survey,” Inf. Fusion, vol. 12, no. 1, pp. 4–10, Jan. 2011.
[5] R. Ghosh, R. Pragathis, S. Ullas, and S. Borra, “Intelligent transportation systems: A survey,” in Proc. Int. Conf. Circuits, Controls, Commun. (CCUBE), 2017, pp. 160–165.
[6] J. Barros, M. Araujo, and R. J. E. Rossetti, “Short-term real-time traffic prediction methods: A survey,” in Proc. Int. Conf. Models Technol. Intell. Transp. Syst. (MT-ITS), vol. 2015, pp. 132–139.
[7] F. Su, H. Dong, L. Jia, Y. Qin, and Z. Tian, “Long-term forecasting oriented to urban expressway traffic situation,” Adv. Mech. Eng., vol. 8, no. 1, Jan. 2016, Art. no. 168781401662839.
[8] T. Schwedtfeger, “DYNEMO: A model for the simulation of traffic flow in motorway networks,” presented at the 9th Int. Symp. Transp. Traffic Theory, Delft, The Netherlands, Jul. 11–13, 1984.
[9] M. Rickert and K. Nagel, “Dynamic traffic assignment on parallel computers in TRANSIMS,” Future Gener. Comput. Syst., vol. 17, no. 5, pp. 637–648, Mar. 2001.
[10] A. K. Ziliaskopoulos and S. T. Waller, “An Internet-based geographic information system that integrates data, models and users for transportation applications,” Transp. Res. C, Emerg. Technol., vol. 8, nos. 1–6, pp. 427–444, Feb. 2000.
[11] M. Ben-Akiva, H. N. Koutsopoulos, C. Antoniou, and R. Balakrishna, Traffic Simulation With DynaMIT. New York, NY, USA: Springer, 2010, pp. 363–398.
[12] K. Friso, L. J. J. Wismans, and M. B. Tijink, “Scalable data-driven short-term traffic prediction,” presented at the 9th Int. Conf. Models Technol. Intell. Transp. Syst. (MT-ITS), Jun. 2017, pp. 687–692.
[13] D. Solomatine, L. M. See, and R. J. Abrahart, “Data-driven modelling: Concepts, approaches and experiences,” in Practical Hydroinformatics, Berlin, Germany: Springer, 2009, pp. 17–30.
[14] B. Ghosh, B. Basu, and M. O’Mahony, “Multivariate short-term traffic flow forecasting using time-series analysis,” IEEE Trans. Intell. Transp. Syst., vol. 10, no. 2, pp. 246–254, Jun. 2009.
[15] J. Guo and B. M. Williams, “Real-time short-term traffic speed level forecasting and uncertainty quantification using layered Kalman filters,” Transp. Res. Rec. J. Transp. Res. Board, vol. 2175, no. 1, pp. 28–37, Jan. 2010.
[16] M. Karimpour, A. Karimpour, K. Kompania, and A. Karimpour, “Online traffic prediction using time series: A case study,” in Integral Methods in Science and Engineering, vol. 2. Cham, Switzerland: Springer, 2017, pp. 147–156.
[17] K. Y. Chan, T. S. Dillon, J. Singh, and E. Chang, “Traffic flow forecasting neural networks based on exponential smoothing method,” in Proc. 6th IEEE Conf. Ind. Electron. Appl., Jun. 2011, pp. 376–381.
[18] Z. H. Mir and F. Filali, “An adaptive Kalman filter based traffic prediction algorithm for urban road network,” in Proc. 12th Int. Conf. Innov. Intell. Technol. (IIT), Nov. 2016, pp. 1–6.
[72] D. Panovski and T. Zaharia, “Real-time public transportation prediction with machine learning algorithms,” in IEEE Int. Conf. Consum. Electron. Dig. Tech. Papers, Jan. 2020, pp. 1–4.

[73] X. Zhang and Z. Liu, “Prediction of bus arrival time based on GPS data: Taking no. 6 bus in Huangdao district of Qingdao city as an example,” in Proc. Chin. Control Conf. (CCC), Jul. 2019, pp. 8789–8794.

[74] P. He, G. Jiang, S. K. Lam, and D. Tang, “Travel-time prediction of bus journey with multiple bus trips,” IEEE Trans. Intell. Transp. Syst., vol. 20, no. 11, pp. 4192–4205, Nov. 2019.

[75] W. Treethidaphat, W. Pattara-Atikom, and S. Khaimook, “Bus arrival time prediction at any distance of bus route using deep neural network model,” in Proc. IEEE 20th Int. Conf. Intell. Transp. Syst. (ITSC), Oct. 2017, pp. 988–992.

[76] T. Yamaguchi, M. As, and T. Mine, “Prediction of bus delay over intervals on various kinds of routes using bus probe data,” in Proc. IEEE/ACM 5th Int. Conf. Big Data Comput. Appl. Technol. (BDCAT), Dec. 2018, pp. 97–106.

[77] R. Yu, Y. Li, C. Shahabi, U. Demiryurek, and Y. Liu, Deep Learning: A Generic Approach for Extreme Condition Traffic Forecasting. Philadelphia, PA, USA: SIAM, 2017, pp. 777–785.

[78] J. Pan, X. Dai, X. Xu, and Y. Li, “A self-learning algorithm for predicting bus arrival time based on historical data model,” in Proc. IEEE 2nd Int. Conf. Cloud Comput. Intell. Syst. (IEEE CICS), Oct. 2012, pp. 1112–1116.

[79] J. Liu and G. Xiao, “Efficient bus arrival time prediction based on spark streaming platform,” in Proc. IEEE 23rd Int. Conf. Comput. Supported Cooperat. Work Design (CSCWD), May 2019, pp. 416–421.

[80] H. Bohan and B. Yun, “Short-term traffic flow prediction based on bus floating car,” in Proc. IEEE 9th Int. Conf. Electron. Inf. Emergency Commun. (ICEIEC), Jul. 2019, pp. 303–306.

[81] R. Fu, Z. Zhang, and L. Li, “Using LSTM and GRU neural network methods for traffic flow prediction,” in Proc. 31st Youth Academic Annu. Conf. Chin. Assoc. Autom. (YAC), Nov. 2016, pp. 324–328.

[82] Y. Wu and H. Tan, “Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework,” 2016, arXiv:1612.01022. [Online]. Available: https://arxiv.org/abs/1612.01022

[83] C.-T. Lam, B. Ng, and S. H. Leong, “Prediction of bus arrival time using real-time on-line bus locations,” in Proc. IEEE 19th Int. Conf. Commun. Technol. (ICCT), Oct. 2019, pp. 473–478.

[84] Y. Wu, H. Tan, L. Qin, B. Ran, and Z. Jiang, “A hybrid deep learning based traffic flow prediction method and its understanding,” Transp. Res. C, Emerg. Technol., vol. 90, pp. 166–180, May 2018.

DANCHO PANOVSKI (Member, IEEE) received the M.S. degree in computer science and information technology from Paris Saclay University, Paris, France, in 2016, and the Ph.D. degree in informatics, data science and AI from the Institute Polytechnique de Paris, Telecom SudParis, Paris, in 2020.

From 2013 to 2016, he was a Research Engineer at ARTEMIS Department, Telecom SudParis. He has contributed on various domestic (French) and European projects during this period. His research interests include public transportation systems, traffic simulation programs, machine learning, deep learning algorithms, data science, and implementation of artificial intelligence in different domains.

TITUS ZAHARIA received the Engineering degree in electronics and telecommunications, and the M.Sc. degree from University Politehnica, Bucharest, Romania, in 1995 and 1996, respectively, and the Ph.D. degree in mathematics and computer science from University Paris V—Rene Descartes, Paris, France, in 2001. He joined the ARTEMIS Department, Institut Mines-Télécom, Télécom SudParis, in 2002, as an Associate Professor, and has become a Full Professor, in 2011.

Since 1998, he has been actively contributing to the ISO/MPEG-4 and MPEG-7 standards. His research interests include visual content representation methods, with 2D/3D compression, reconstruction, recognition, and indexing applications, but also computer vision and deep learning methodologies.

226284