Public Transportation Prediction with Convolutional Neural Networks

Dancho Panovski(✉) and Titus Zaharia

IP Paris, Télécom SudParis, ARTEMIS Department,
UMR CNRS 5157 SAMOVAR, Évry, France
{dancho.panovski,titus.zaharia}@telecom-sudparis.eu

Abstract. Good, efficient and reliable public transportation systems are of crucial importance for all major cities today. In this paper, we propose a concrete solution to a particular problem: improve the prediction of the bus arrival time at each bus stop station on a given itinerary, by taking to account global and local traffic contexts. The main principle consists of modeling the traffic data as an image structure, adapted for applying CNN deep neural networks. The results obtained shows that the proposed approach outperforms traditional machine learning techniques, such as OLS (Ordinary Least Squares) or SVR (Support Vector Regression) with different kernels (RBF or Polynomial), with more than 18% better accuracy prediction, while being computationally faster.

Keywords: Machine learning · Deep learning · Convolutional neural networks · Public transportation · Traffic prediction · Traffic simulation

1 Introduction

In all urban areas around the world, mobility is of the most crucial importance, and the question that needs to be solved is the following: how to minimize the time spent in transport going from one point of the city to another?

In addition, let us underline that vehicular transports in the urban areas performed by individuals or public transportation vehicles contributed heavily on the carbon footprint, called greenhouse gas. The data [1] obtained from the French government in 2015 gives an extensive overview of the key numbers about production of the greenhouse gas. Transport, in general, is responsible for 27% of the production of greenhouse gas. Transport operated by road represents 94,8% of these emissions. The logical solution to this problem is to decrease the number of vehicles used for private transportation, by proposing more reliable public transportation systems. The question of reliability is here highly important since the user acceptancy strongly depends on. Within this framework, disposing of efficient traffic prediction tools, dedicated to public transportation, is a challenging issue, for both users (which want to be informed precisely and in real-time) and transport operators (which want to optimize their transport networks/itineraries).

A. L. Martins et al. (Eds.): INTSYS 2019, LNICST 310, pp. 150–161, 2020.
https://doi.org/10.1007/978-3-030-38822-5_10

2 Related Work

In this paper, we specifically tackle the issue of bus arrival time prediction in urban areas. The main objective is to minimize the predicted time error of bus arrival at each bus stop while taking to account global and local traffic data at the itinerary of the bus.

In order to obtain this particular data type, we decided to simulate the scenario with the help of a traffic simulation software. Traffic simulation can be defined as the mathematical model of transportation systems, implemented through the application of dedicated algorithms [2]. Two main branches exist for traffic simulation software, including macroscopic [3] and microscopic [4] approaches. Each traffic simulation has its own pros and cons. In our work, a microscopic traffic simulator has been adopted, since it can provide a highly detailed picture (with velocity, location, time, speed…) of each individual entity in the system. More precisely, among the various well-established simulation frameworks [5–7] available, the SUMO [8] traffic simulator has been retained. SUMO (*Simulation of Urban MObility*) [8] is an open-source, microscopic, multimodal traffic simulator that can simulate various scenarios with different type of traffic data and it is supported by a large developer community.

Different algorithms, techniques, and applications have been introduced in the last years in order to address the issue of traffic prediction [9–12]. A conventional commercial application like Google Maps [13] and Citymapper [14] have been widely adopted by the general population. Among other applications, let us mention Transilien [15] for Paris and MVV [16] for Munich. Traditional techniques include time-series analyses [17], ARIMA models [18] with its variations [19] and Kalman Filter [20]. They have been successfully implemented and still intensively exploited today. However, in recent years, with the rapid growth of computational powers, notably GPUs, such approaches have been surpassed in many applications by machine learning techniques [21] and more specifically by deep learning approaches [22]. Different neural networks have also been introduced for solving similar issues, including LSTMs (Long Short Term Memory) [23, 24], CNNs (Convolutional Neural Networks) [25, 26] and GNNs (Graph Neural Networks) [27, 28].

Our main contribution is twofold and concerns the development of the data model involved as well as the prediction approach exploiting the dedicated data model. Thus, the vehicular traffic data is in our case modeled and represented as an image. This makes it possible to exploit an original, dedicated convolutional neural network (CNN) that yields the predicted arrival times of the bus as a regressor.

The rest of the paper is organized as follows, Sect. 3 describes the simulation scenario and the data model. In Sect. 4, the proposed CNN method is described in details. Experimental results are presented and discussed in Sect. 5. Finally, Sect. 6, concludes the paper and opens some perspectives of future work.

3 Simulation Scenario and Data Model

3.1 Simulation of a Real-Live Scenario

Obtaining data that contains information about vehicles, public transportation, pedestrians…. is not obvious and easy to acquire, for various reasons, including economic barriers and regulation issues. In our work, we have considered instead synthetic data, that we have created with the help of the SUMO (*Simulation of Urban MObility*) [8] open platform.

The simulation scenario under consideration concerns two real bus lines, that are fully operational in the city of Nantes, France. They are illustrated in Fig. 1. The decision of choosing these lines is based on the geographical location of the bus itineraries, which globally cover the same geographical region. This makes it possible to include two different bus lines in a single simulation scenario. The bus data, including itineraries, location and names of the bus stops have been recovered from TAN (*Transport de l'Agglomeration Nantaise*) [29] and Nantesmetropole [30] – open data repository, while the map was loaded from OSM (*Open Street Map*) [31].

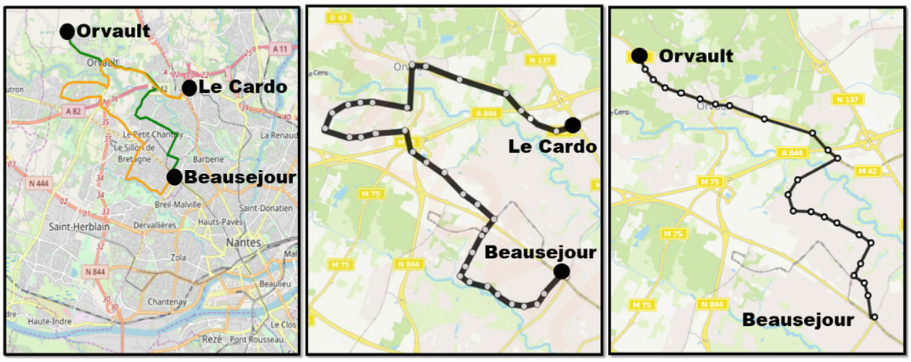

Fig. 1. A geographical area in city of Nantes, with Bus line 89 (middle) – 36 bus stops and 13,4 km, bus line 79 (right) – 25 bus stops and 9,64 km

The simulation scenario has been developed and executed by the SUMO traffic simulator with the following tools and parameters:

- The 2D map of the considered region of the city of Nantes has been converted and imported from OSM.
- The bus itineraries and the bus stop have been manually added to the simulation.
- The total number of simulations performed is 12000.
- The total number of vehicles inserted into the system varies between [8000, 25000].
- Each vehicle itinerary is calculated using the shortest path algorithm of Dijkstra [32], presented with 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 simulation time was set to 19000 s, which ensures that all the vehicles will get in and out of the system, as presented in Fig. 2(A).
- The bus time spends at each bus stop was fixed to 20 s.

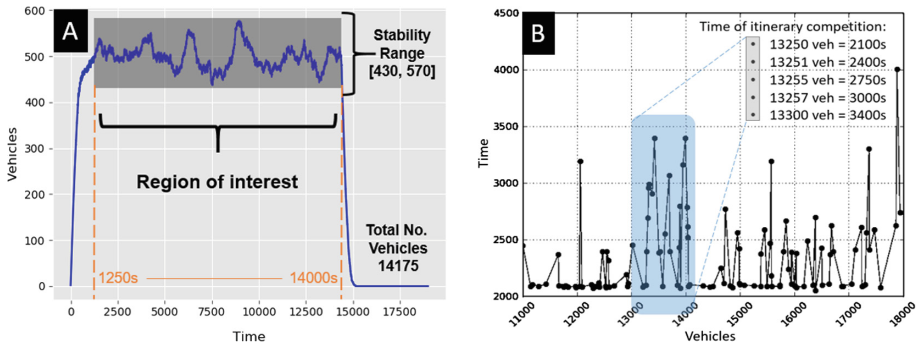

Fig. 2. A – region of interest and stability range; B - time needed to compute the whole itinerary

Each consecutive simulation differs from the previous one by the number of vehicles inserted within the system. This may be questionable because of the small number of vehicles inserted into the system between two consecutive simulations. However, huge disparities from one simulation to another can appear, even though the number of vehicles is increased by small units. This phenomenon is illustrated in Fig. 2 (B), which show the time of completion of the bus itinerary with respect to the total number of vehicles that are inserted in the system. This can be explained as a result of the random distribution of the initial vehicle starting point to the systems, and also by the fact that for each vehicle the shortest route algorithm is calculated, taking to account the global situation.

In the same time, the results in Fig. 2(B) demonstrate that attempting to make a global prediction of the time of completion of an itinerary is impossible.

3.2 Data Model

Our initial goal is to take into account the traffic situation around the bus itinerary. Each simulation is used to create a specific traffic density map that later on can be transformed into one channel (grayscale) image as presented in Fig. 3.

In order to represent traffic as a grayscale image, first, we need to compute and create a so-called TDM (*Traffic Density Matrix*). This matrix represents the traffic situation at each measurement station. Measurement stations are points in space that detects and counts how many vehicles pass in a given period of time. In real-life scenarios, such information can be acquired with the help of loop detectors. In our work, for the convenience of the simulation, each bus stop location has been considered as a measurement station. In this particular scenario, the detection radius around the considered measurement station was set to 100 m. This representation can be interpreted as a traffic density matrix evolving over time, which is traversed by the considered buses. The last step is to normalize the values of the traffic density matrix by the maximum value of 255 as one channel (grayscale) image.

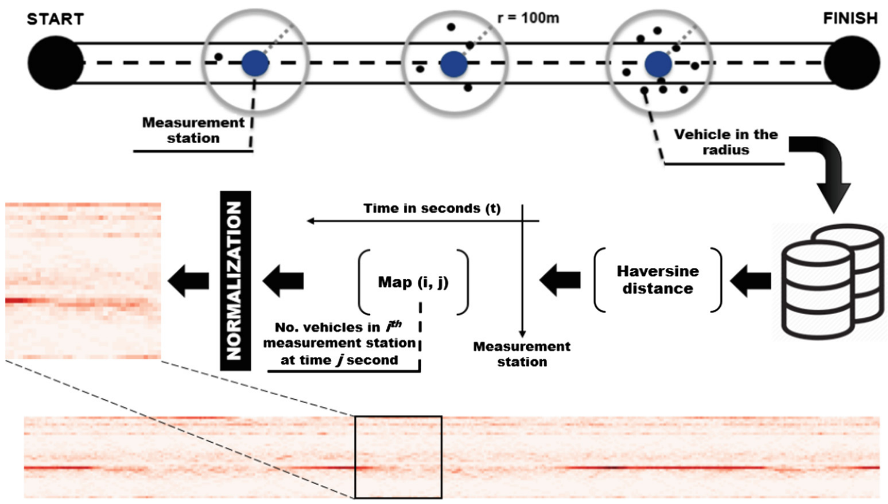

Fig. 3. Traffic density matrix, computation, normalization and traffic data visualized as an image

For each bus generated at each simulation, we create in this way a local density map simulation matrix, denoted by D, of size (Mstations x T), where Mstations is the number of measurement stations and T is the number of time instances measured. In order to simplify and speed up the process, we have evenly sampled the simulation interval with a step of 10 s. This helps reduce 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 total number of vehicles injected within the system:

- A vector $a = \left(a_1, a_2, a_{\text{Nstops}}\right)$ of size N_{stops}, storing the bus arrival times in all the stations of the itinerary
- The density matrix D, of size $\left(M_{\text{staions}} \times T\right)$
- M_{stations} is (25, 36) for bus 79 and 89 respectively
- T - the time is 4000 s, but since is measured every 10 s, $T = 400$
- The final image was developed by the grayscale (min/max) normalization procedure:

$$I = 255\frac{X - \min(X)}{\max(X) - \min(X)} \tag{1}$$

- *X:* the value of the original image
- *I:* the output - grayscale value of the image after the normalization procedure.

The objective is then to predict the arrival vector a, given as an input the image I. In order to solve such a problem, we have adopted a CNN architecture, described in the following section.

4 Proposed CNN Architecture

The (Convolutional Neural Network) CNN architecture proposed is illustrated in Fig. 4.

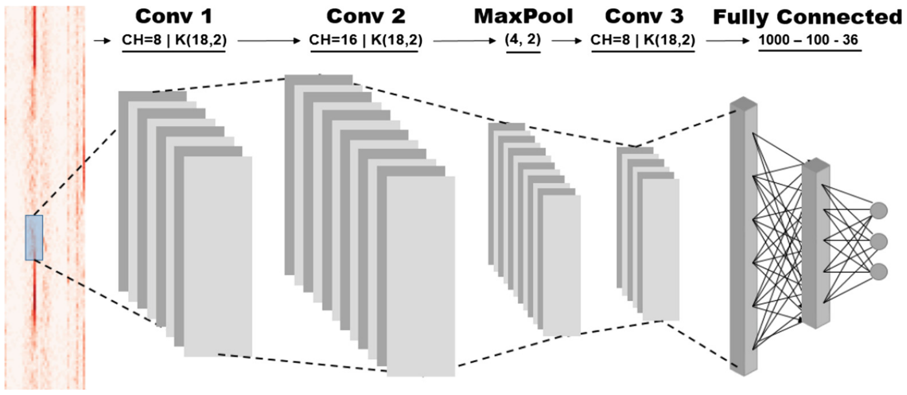

Fig. 4. A proposed convolutional neural network with 3 convolutional layers, one max-pooling and 3 fully connected layers, ReLU as an activation function

It includes the following elements:

- 3 convolutional layers with kernel size (18, 2) and channels (8, 16, 8) respectively,
- 1 max-pooling layer of size (4, 2),
- 3 fully connected layers with size (1000, 100, 36).

The specific kernel size was chosen because of the particular elongated shape of the image data considered (*cf.* Section 3.2). The size and structure of the fully connected layer depend heavily on the vector that is received after the convolutions and downsizing step. Padding was chosen as 0, so it was not used in this scenario because the traffic information at the beginning and the end of the image is not relevant for the prediction process. The SGD (*Stochastic Gradient Descent*) algorithm (with learning rate of 0.001) has been considered for the learning stage.

For implementation purposes, we have considered the PyTorch [33] deep neural network framework for the CNNs.

5 Experimental Results

For comparison purposes, we have considered several machine learning techniques: Linear Regression (OLS) [34], Support Vector Regression (SVR) [35] (with different kernels). We have retained the Scikit-learn [36] machine learning toolbox with their respective implementations. All the data has been structured in .csv format and compressed with the MsgPack [37] library. The next step is to split the data randomly into three different sub-set and performing 10-fold cross-validation. The training/test split was performed as follows:

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

Our first approach is based on Linear Regression also known as OLS (Ordinary Least Squares). Figure 5 illustrates a snapshot of the simulation from the bus during the scheduled itinerary, together with the prediction curves obtained by the linear regression model (versus the ground truth), for booth buses 79 and 89.

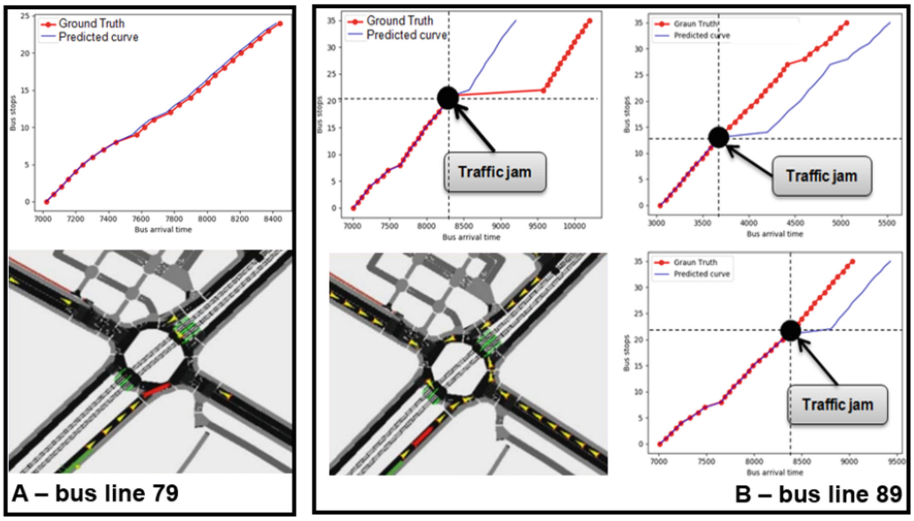

Fig. 5. Prediction results and snapshot from the simulation software. A - bus line 79, the traffic situation is normal and fluid. B - bus line 89 on the right with a traffic jam.

The corresponding MAE (Mean Absolute Error) scores are summarized in Table 1.

Table 1. Prediction results for bus 79 and 89 both directions

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

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

For bus line 79, the results are highly accurate, with an MAE inferior to 5 s. In this case, we can observe that the traffic is mostly fluid and the OLS model is perfectly adapted for prediction purposes. However, the prediction results degrade significantly in the case of bus line 89. We can observe from Fig. 5(B), that some localized singularities events (traffic jams) occur in some places and introduce a certain non-linearity. As a consequence, the MAE value increases up to 134,18 s.

In order to investigate this behavior, we have considered a second approach, based on SVR (kernel = ‘Poly’ and ‘RBF’). The polynomial kernel is of second degree known as Quadratic kernel, and the RBF (Radial Basis Function) kernel with gamma = 1/n_features and C = 1.0. The obtained prediction curves are presented in Fig. 6.

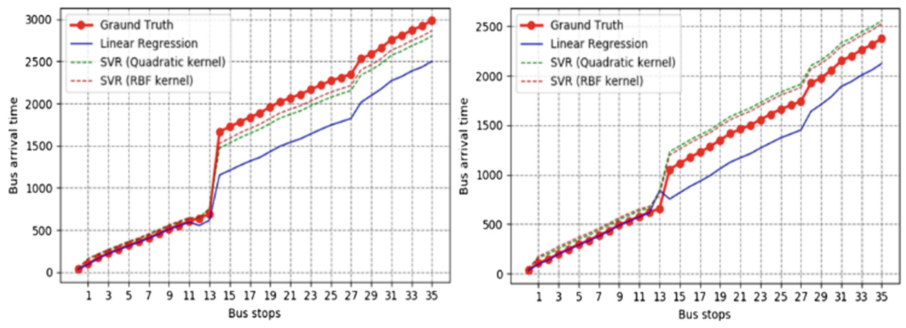

Fig. 6. Support vector regression (‘Poly’ and ‘Rbf’) vs linear regression

We can observe that the SVR method with both kernels significantly outperforms the Linear Regression model. These results are also confirmed by the global MAE values and computational time reported in Table 2. We can observe that the CNN training process is faster than the other two methods.

Table 2. Prediction error and computational time for bus 89 direction Beausejour > Le Cardo

	OLS	SVR (rbf)	SVR (poly)	CNN
MAE	134.18	71.23	72.06	59.38
cTime	818	12063	11727	768

MAE = Mean Absolute Error (s), cTIME = Computational Time in (s), OLS = Ordinary Least Squares, Poly = Polynomial, Rbf = Radial Basis Function

The results obtained confirm our original intuition: more complex, non-linear machine learning techniques are required, in order to be able to obtain more accurate predictions while taking to account this localized, non-linear phenomena. Let us now investigated if the proposed CNN can further confirm this intuition.

Figure 7 presents the prediction curves between the different machine learning techniques considered.

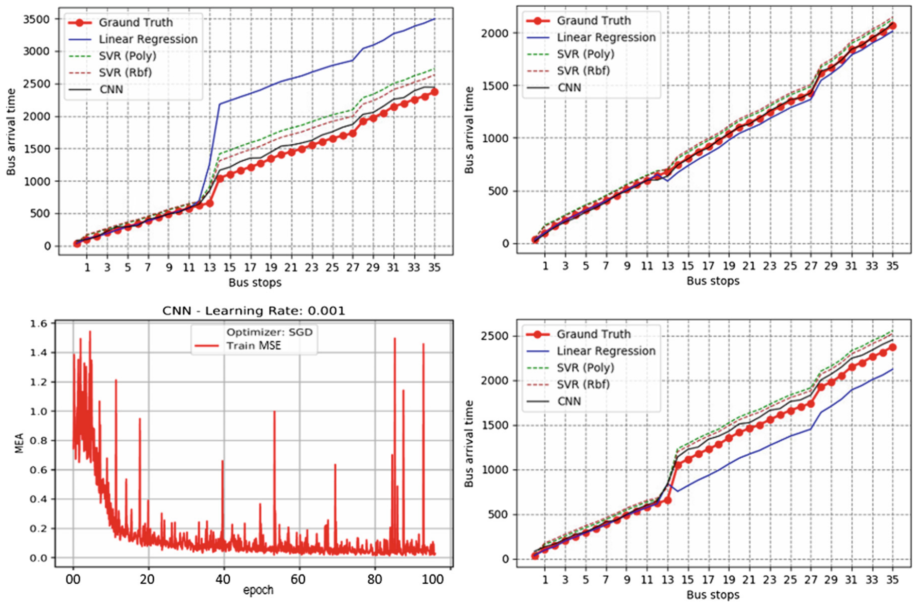

Fig. 7. OLS vs SVR (Poly and Rbf) vs CNN, the Learning curve for CNN

The proposed CNN method outperforms the two previously considered techniques, with overall gains accuracy (Table 2) of 18,5% with respect to SVR and 77,7% better over OLS. The learning curve evolution of the CNN training model (for 105 epochs) is also illustrated here.

In Fig. 8, the histogram and cumulative histogram plots are presented.

From the plot (left) we can observe that 50% of the simulations have an MAE lower than 34 s (which corresponds to the median of the MAE). Moreover, in 80% of cases the MAE is inferior to 86 s.

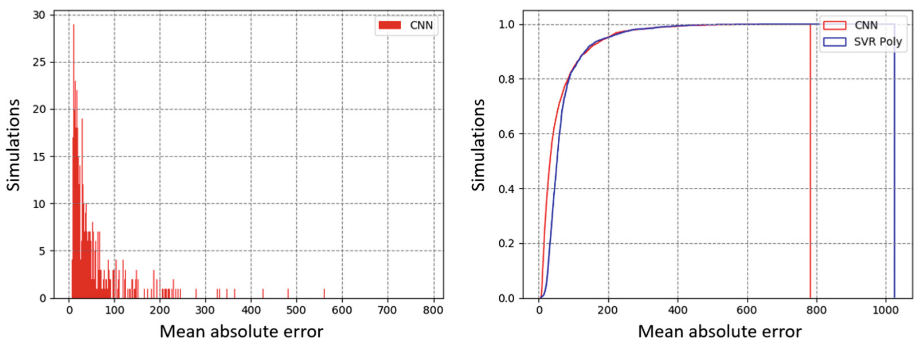

Fig. 8. CNN histograms (left), CNN and SVR Poly cumulative histogram (right)

6 Conclusion and Perspectives

In this paper, we have proposed a new approach for the prediction of bus arrival times in the various bus stops over their itinerary.

We have shown that modeling the traffic data as a 2D image can offer multiple benefits for structuring the information in a meaningful and exploitable manner.

A dedicated CNN network architecture has been proposed, which makes it possible to predict the bus arrival time while taking into account the current traffic situation. The experimental results obtained showed that the proposed CNN outperforms the traditional prediction techniques based on linear regression or SVR approaches, in both prediction accuracy and computational complexity.

Our perspectives of future work concern the consideration of different neural network-based approaches able to deal with the temporal nature of the data, such RNN (Recurrent Neural Networks) and LSTM (Long Short Time Memory).

Acknowledgment. Part of this work has been carried out within the framework of the French FUI project ETS (*Electronic Ticketing System*), supported by the Conseil Départemantal de l'Essonne and the Systematic Paris Region competitivity cluster.

References

1. SOeS: Mars 2015 Chiffres clés du transport
2. Azlan, N.N.N., Rohani, M.M.: Overview of application of traffic simulation model. In: MATEC Web Conference, vol. 150, p. 03006 (2018). https://doi.org/10.1051/matecconf/201815003006
3. Geroliminis, N., Daganzo, C.F.: Macroscopic modeling of traffic in cities (2007)
4. Toledo, T., Koutsopoulos, H., Ben-Akiva, M., Jha, M.: Microscopic traffic simulation: models and application. In: Simulation Approaches in Transportation Analysis, pp. 99–130. Springer, New York. https://doi.org/10.1007/0-387-24109-4_4
5. Balmer, M., Rieser, M.: MATSim-T: architecture and simulation times. In: Multi-Agent Systems for Traffic and Transportation Engineering (2009). https://doi.org/10.1140/epjb/e2008-00153-6

6. Fellendorf, M., Vortisch, P.: Microscopic traffic flow simulator VISSIM. In: Fundamentals of Traffic Simulation (2010). https://doi.org/10.1007/978-1-4419-6142-6_2
7. Rickert, M., Nagel, K.: Dynamic traffic assignment on parallel computers in TRANSIMS. Future Gener. Comput. Syst. **17**, 637–648 (2001). https://doi.org/10.1016/S0167-739X(00)00032-7
8. Krajzewicz, D., Erdmann, J., Behrisch, M., Bieker, L.: Recent development and applications of SUMO - Simulation of Urban MObility. Int. J. Adv. Syst. Meas. **5**, 128–138 (2012)
9. Panovski, D., Zaharia, T.: Simulation-based vehicular traffic lights optimization. In: 2016 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), pp. 258–265. IEEE (2016). https://doi.org/10.1109/SITIS.2016.49
10. Lin, Y., Yang, X., Zou, N., Jia, L.: Real-time bus arrival time prediction: case study for Jinan, China. J. Transp. Eng. **139**, 1133–1140 (2013). https://doi.org/10.1061/(ASCE)TE.1943-5436.0000589
11. As, M., Mine, T.: Dynamic bus travel time prediction using an ANN-based model. In: Proceedings of the 12th International Conference on Ubiquitous Information Management and Communication - IMCOM 2018, pp. 1–8. ACM Press, New York (2018). https://doi.org/10.1145/3164541.3164630
12. Petersen, N.C., Rodrigues, F., Pereira, F.C.: Multi-output bus travel time prediction with convolutional LSTM neural network. Expert Syst. Appl. **120**, 426–435 (2019). https://doi.org/10.1016/j.eswa.2018.11.028
13. Google Maps. https://www.google.com/maps
14. Citymapper - The Ultimate Transport App. https://citymapper.com/paris
15. Schanzenbacher, F., Chevrier, R., Farhi, N.: Fluidification du traffic Transilien : approach prédictive et optimisation quadratique, 2p (2016)
16. Munich Transport and Tariff Association | MVV. https://www.mvv-muenchen.de/
17. Karimpour, M., Karimpour, A., Kompany, K., Karimpour, A.: Online traffic prediction using time series: a case study. In: Constanda, C., Dalla Riva, M., Lamberti, P.D., Musolino, P. (eds.) Integral Methods in Science and Engineering, Volume 2, pp. 147–156. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59387-6_15
18. Kumar, S.V., Vanajakshi, L.: Short-term traffic flow prediction using seasonal ARIMA model with limited input data. Eur. Transp. Res. Rev. **7**, 21 (2015). https://doi.org/10.1007/s12544-015-0170-8
19. Zhang, N., Zhang, Y., Lu, H.: Seasonal autoregressive integrated moving average and support vector machine models. Transp. Res. Rec. J. Transp. Res. Board. **2215**, 85–92 (2011). https://doi.org/10.3141/2215-09
20. Mir, Z.H., Filali, F.: An adaptive Kalman filter based traffic prediction algorithm for urban road network. In: 2016 12th International Conference on Innovations in Information Technology (IIT), pp. 1–6. IEEE (2016). https://doi.org/10.1109/INNOVATIONS.2016.7880022
21. Panovski, D., Scurtu, V., Zaharia, T.: Simulation and prediction of public transportation with maps of local density blobs. In: 2019 IEEE International Conference on Consumer Electronics (ICCE), pp. 1–4. IEEE (2019). https://doi.org/10.1109/ICCE.2019.8661921
22. Panovski, D., Scurtu, V., Zaharia, T.: A neural network-based approach for public transportation prediction with traffic density matrix. In: 2018 7th European Workshop on Visual Information Processing (EUVIP), pp. 1–6. IEEE (2018). https://doi.org/10.1109/EUVIP.2018.8611683
23. Ke, J., Zheng, H., Yang, H., Chen, X.M.: Short-term forecasting of passenger demand under on-demand ride services: a spatio-temporal deep learning approach. Transp. Res. Part C Emerg. Technol. **85**, 591–608 (2017). https://doi.org/10.1016/J.TRC.2017.10.016

24. Yu, R., Li, Y., Shahabi, C., Demiryurek, U., Liu, Y.: Deep learning: A generic approach for extreme condition traffic forecasting. In: Proceedings of the 2017 SIAM International Conference on Data Mining, pp. 777–785. Society for Industrial and Applied Mathematics (2017)
25. Ma, X., Dai, Z., He, Z., Na, J., Wang, Y., Wang, Y.: Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction. Sensors **17**(4), 818 (2017)
26. Fouladgar, M., Parchami, M., Elmasri, R., Ghaderi, A.: Scalable deep traffic flow neural networks for urban traffic congestion prediction. In: 2017 International Joint Conference on Neural Networks (IJCNN), pp. 2251–2258. IEEE (2017). https://doi.org/10.1109/IJCNN.2017.7966128
27. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S.: A comprehensive survey on graph neural networks (2019)
28. Li, Y., Yu, R., Shahabi, C., Liu, Y.: Diffusion convolutional recurrent neural network: data-driven traffic forecasting (2017)
29. Webmaster: Tan - Ma vie sans arrêt - tan.fr
30. Accueil—Open Data Nantes Métropole. https://data.nantesmetropole.fr/pages/home/
31. Jokar Arsanjani, J., Zipf, A., Mooney, P., Helbich, M.: An introduction to OpenStreetMap in geographic information science: experiences, research, and applications. In: Jokar Arsanjani, J., Zipf, A., Mooney, P., Helbich, M. (eds.) OpenStreetMap in GIScience. LNGC, pp. 1–15. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-14280-7_1
32. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**(1), 269–271 (1959)
33. Paszke, A., et al.: Automatic differentiation in PyTorch (2017). https://openreview.net/forum?id=BJJsrmfCZ
34. Seber, G.A.F., Lee, A.J.: Linear Regression Analysis. Wiley, Hoboken (2003)
35. Undefined: support vector regression (2004). cmlab.csie.ntu.edu.tw. Science, M.W.-D. of C., of, U
36. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
37. Ching, T., Eddelbuettel, D.: RcppMsgPack: messagepack headers and interface functions for R (2018)