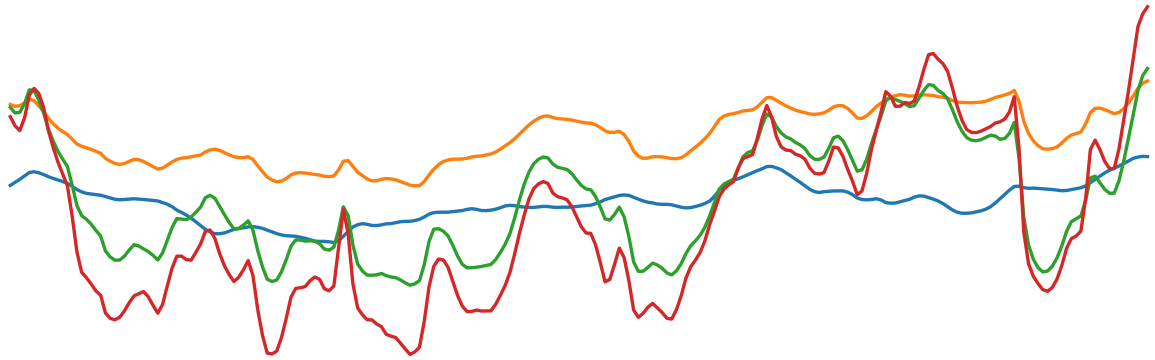


Short-term travel time prediction for public transport

Guillaume Neven (16-832-883)

Student Report

103-0488-00L Seminar in Raumentwicklung und Infrastruktursysteme **2020**

IVT Institut für Verkehrsplanung und Transportsysteme
Institute for Transport Planning and Systems

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Seminar project

Short-term travel time prediction for public transport

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology ETH

Institut für Verkehrsplanung und Transportsysteme
103-0488-00L Seminar in Raumentwicklung und Infrastruktursysteme
Frühjahrssemester 2020

Guillaume Neven (16-832-883)

Abstract

The reliability of public transportation is,unquestionably, of paramount importance in driving a shift in the use of transportation modes. In this respect, innovation in improving the prediction of delays has a major role to play. In the last decade, machine learning and artificial intelligence has been widely deployed in various field, and transportation science is no exception. In view of the importance of the objective, it is not surprising to see data driven model for delays prediction. This project contributes to this effort.

This project proposes a data driven model using a new kind of LSTM, capturing both temporal and spatial aspect to predict the delays of buses using their past delays, boarding/alighting time and travel time. It has been tested for the case of Zürich. This new model, implementing a bus based multi features analysis at a network scale has shown encouraging results. It succeeded to improved the medium-long term predictions for the city of Zürich, even though the network exhibits a high punctuality.

Contents

List of Figures

List of Tables

Acknowledgement

I am particularly grateful for the advice and detailed comments provided by my two supervisors Kimia Chavoshi and Dr. Georgios Sarlas as well as the support from Dr. Anastasios Kouvelas. I also want to thank the community behind the Keras package (Chollet and others (2015)) for their sustained efforts in making machine learning a bit more accessible.

1 Introduction

Public transportation has a major part to play in responding to increasing demand for mobility and achieving sustainable mobility. In this respect, it is important to shift travellers away from private motorized modes and towards public transportation.

To prompt this shift, it is essential to enhance user's perception of quality. A key aspect of quality concerns the reliability of the network Jenelius (2018). Even though disruptions and delays can potentially decrease the reliability of a system, seeking ways of mitigating their impacts lie on the forefront of transport research. To that end, the importance of real time information and accurate service time predictions can be emphasized as they can allow both users and operators to plan accordingly.

To achieve these, it is essential to understand the function of a transport network and the interactions between its different actors, especially during peak hours where the demand and normally the delays are higher. In this project, a new method to predict delays of buses is proposed. More specifically, the application of a Machine Learning (henceforth denoted as ML) algorithm is explored, which is trained on historical data in order to capture the different levels of network complexities that arise. The ML approach is tested in terms of its ability to predict buses' delays in real time. Subsequently, the results could be utilised for enhancing the reliability of the public transport service.

In summary, the objective of this work is to create a data driven model to improve the quality of delay predictions significantly ahead of time.

2 Review of Relevant Literature

ML has become widely used in the last decade, mainly as a result of the explosion of computation power. This has also occurred in the the case of transport systems where ML has been increasingly deployed as passenger and travel data started to be more widely recorded. The most popular algorithm is the Long Short-Term Memory (LSTM). It has been applied to various matters in transport engineering, mostly for travel time prediction purposes (e.g. Ma et al. (2015) Agafonov and Yumaganov (2019) Chen et al. (2019) Kim et al. (2016)), or prediction of "Time-to-green" in intersections (e.g. Genser et al. (2020)).

ML approaches have also found application in the area of delay prediction for public transport. For instance, Petersen et al. (2019) developed an LSTM model to predict travel times along the route of a single bus line in Copenhagen. Their approach employs a link-level analysis allowing to make statements about the delay based on the corresponding travel time predictions. More specifically, they proposed a combination of LSTM and convolution models in order to capture the temporal and spatial correlations. A comparison against different prediction methods showed that their approach outperformed them.

In the same spirit, Huang et al. (2019) developed a model that takes into consideration a higher number of potential variables. In particular, their data set involved a combination of Origin-Destination data based on personalised card logs, and Automated Vehicle Location (AVL) system records for the whole network. The implemented model contains variables describing the travel time and the number of boarding and alighting passengers per stop. In addition, some sub-variables have been added as well, such as weather conditions, day of the week, and whether or not school was taking place at a specific day. In summary, they succeeded to drastically improve the accuracy of the predictions, but at the cost of a high data load, making it difficult to implement everywhere.

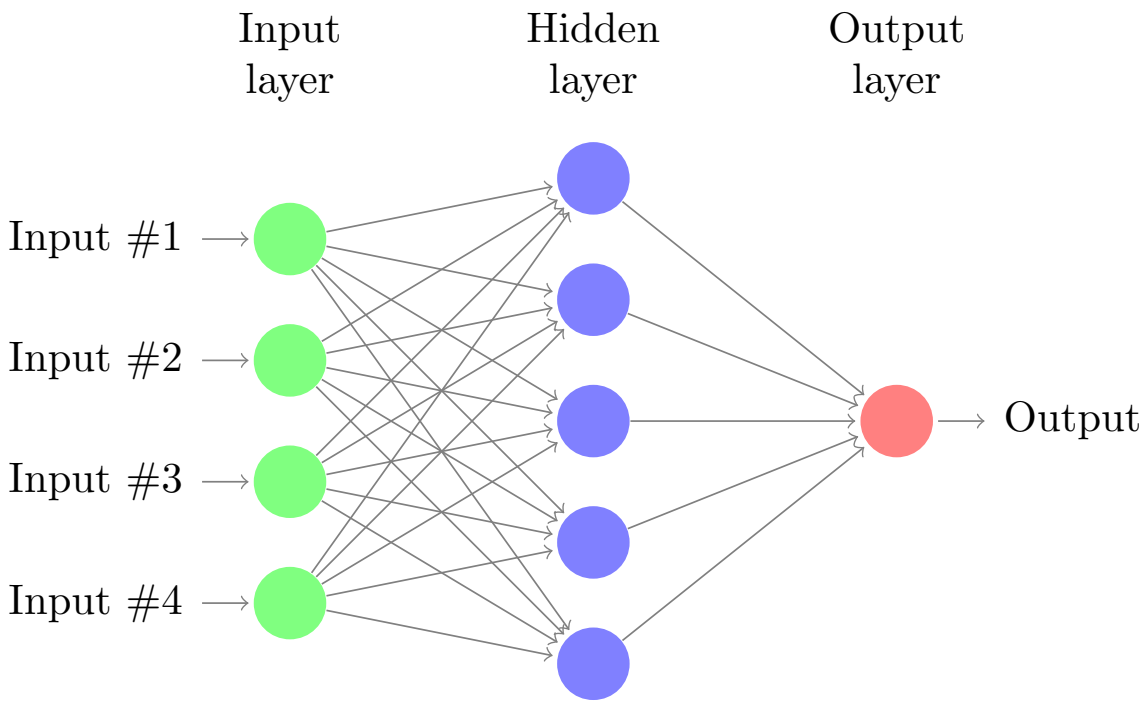
Last, Liu et al. (2020) proposed a travel time prediction approach for a specific line that uses detailed bus GPS probe data, including speed and localisation. In this model, the respective bus travel routes are not analysed as line segments between successive stops but between network intersections. Subsequently, the traffic conditions on those link are predicted and then used to predict the bus delays. Once again they succeed to improve the prediction compared to the system in place.

3 Long short-term memory algorithm

The Long Short-Term Memory (LSTM) is an ML algorithm from the class of Recurrent Neural Network (RNN), which belongs itself to the class of Artificial Neural Networks (ANN). It is largely used for translation and speech recognition purposes but also have found application in transportation(e.g. Huang et al. (2019)).

In figure 1, the simplest form of a neural network is presented. It can have multiple inputs and outputs; in the example the model has 4 inputs and 1 output. Each link has a weight which indicates how much information should be transferred to the next node. Once in the node, a function merges all the incoming links and creates a new value that is transferred further on. A large number of functions exist with the simplest one being a linear activation. At the end, the output is compared to the expected value. Then the model is fed backwards, adjusting the weights in order to get closer to the expected value.

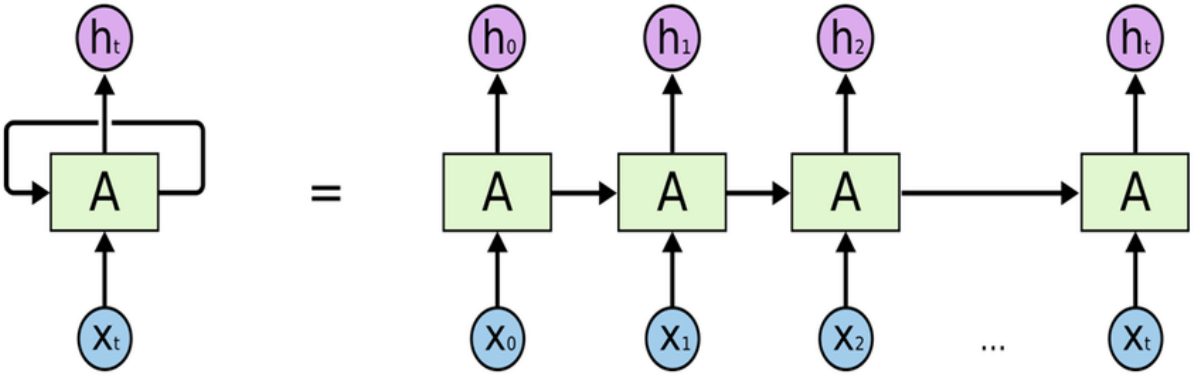
Figure 1: Schema of traditional Artificial Neural Networks (ANN).



Source : http://www.texample.net/tikz/examples/neural-network/

A major limitation of the traditional ANN is the complexity to implement time relations. For instance, one could train a model with the delays of the bus in order to predict the futures delays but the model will not be recursive. RNN overcomes this inherent limitation. Figure 2 illustrates this idea which includes training the model for different time steps, while taking into account information from the past.

Figure 2: Schema of Recurrent Neural Network (RNN).

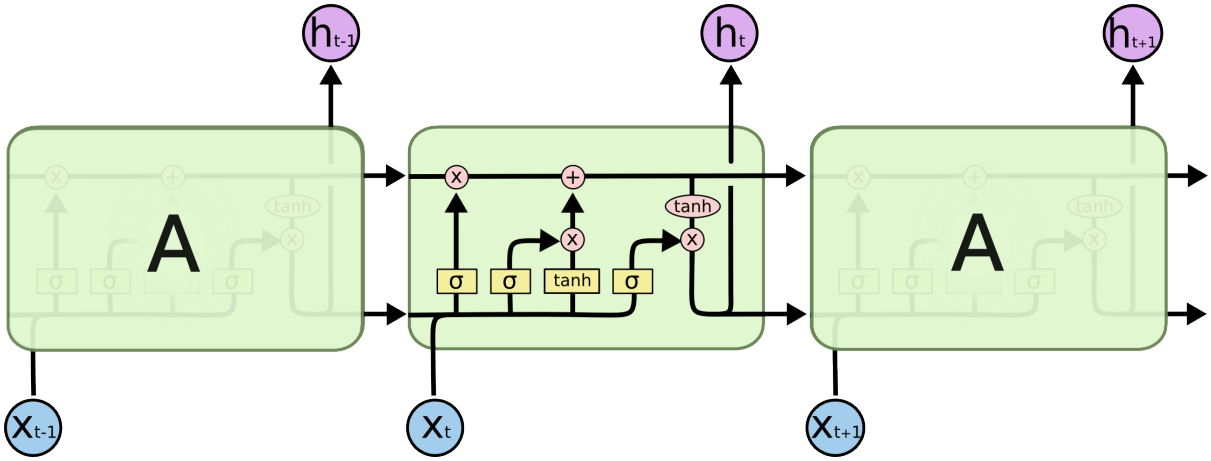


Source : https://colah.github.io/posts/2015-08-Understanding-LSTMs

In this RNN, a time window is selected where the information to predict the output h_t is coming from time step $X_0 - X_t$ and not further backwards. The number of weights, as depicted in 1 can however be very large. If applied to the case study using a simple RNN network to predict the delays of N_{bus}, using the 3 selected features (Total lateness, Travel time lateness and boarding/alighting delays) using the time windows W_{size} of 10 time step behind, one would need an astonishing $2.91e12$ number of weights (equation 1). Here lies the major advantages of the LSTM.

$$N_{\text{weight}} = (N_{\text{bus}} * N_{\text{features}} * W_{\text{size}})^3 = (476 * 3 * 10)^3 = 2.91e12 \quad (1)$$

Figure 3: Schema of Long short-term memory.

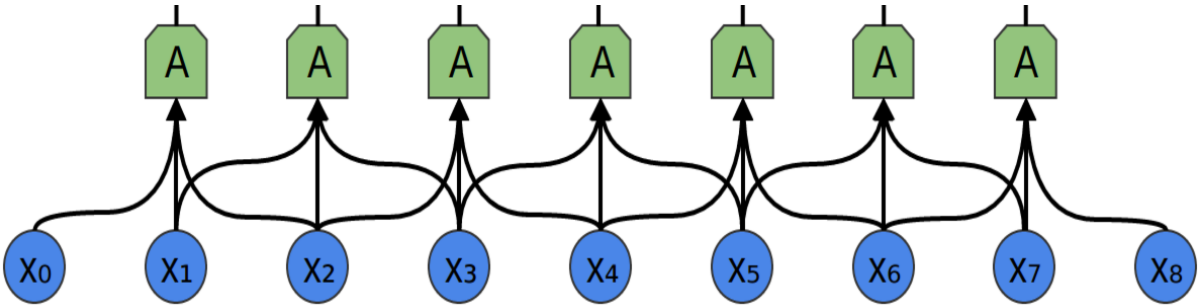


Source : https://colah.github.io/posts/2015-08-Understanding-LSTMs

In figure 3 a basic LSTM model is presented. As one could notice, there is no information travelling through neurons, and therefore through time. Unlike the RNN, the LSTM has two flows of information between cells. The first one, similar to the ANN, is the output of the previous time step. The second one is called the memory, which is independent of the inputs window's size. For example, if a text is given to the LSTM, the memory will keep important information throughout the text, allowing it to understand pronoun referring to object or people defined well before. A detailed presentation and discussion of LSTM can be found in Olah (2015).

In addition, a modification can be made to the LSTM to capture spatial effects. The idea is to feed the model not merely with an input including a single value but to add a convolution layer just before. The figure 4 shows an example for a one dimension convolutional layer. In the cell *A*, a weighing of the n-inputs is made and given to the LSTM.

Figure 4: Illustration of the conventional layer.

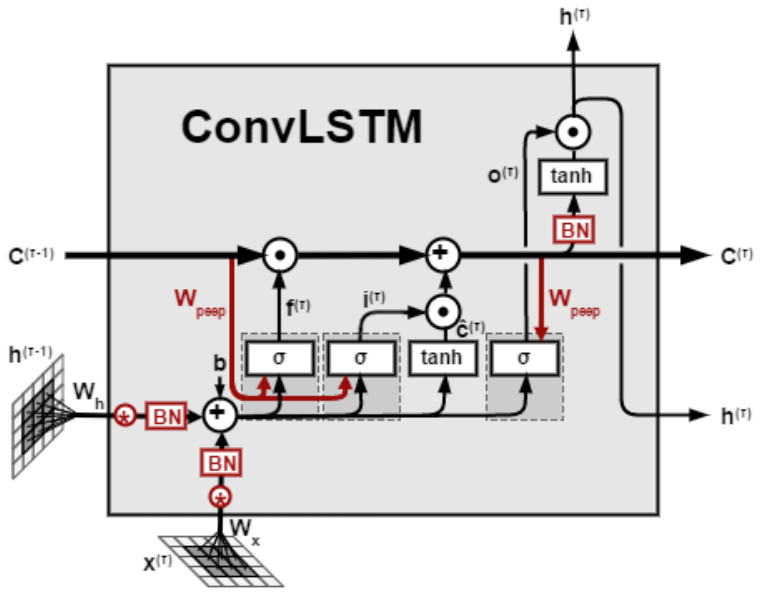


Source : https://colah.github.io/posts/2014-07-Conv-Nets-Modular/

The idea behind the implementation of this convolutional layer is to allow the model to look at the buses behind and ahead of that for which the prediction is performed. To illustrate it conceptually, if a road accident happened ahead, the model will not be trained to this particular scenario, but it might still learn about large delays from the bus(ses) ahead. The ConvLSTM also uses a convolution step inside each gate of LSTM model.

Each step of the ConvLSTM is mathematically defined mathematically below (equation 2 to 7) and schematically in figure 5.

Figure 5: Illustration of the conventional long short-term memory model.



Source : https://medium.com/neuronio/an-introduction-to-convlstm-55c9025563a7

There are 4 different gates in the convLSTM cell. First, the forget gate (equation 2), f_t is computed by passing to as simgoid function (see figure 6) the combination of the previous output $\mathcal{H}_{t-1}$, the memory C_{t-1} and the input $\mathcal{X}_t$ plus a bias b_f. Every W matrix contains the weights that are going to be updated along the process. The idea of this gate is to select (from 0 to 1) the information to be passed to the memory C_t.

$$f_t = \sigma\left(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \circ C_{t-1} + b_f\right) \tag{2}$$

Figure 6: simgoid function

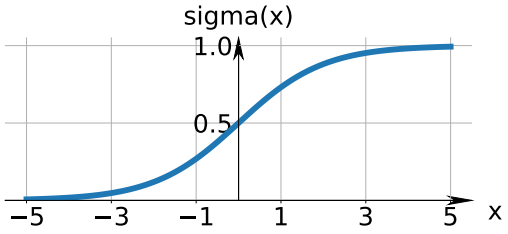
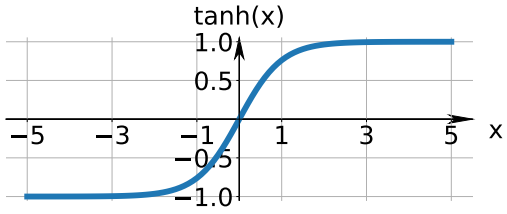


Figure 7: tanh function



$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

The second gate, called update gate, update the candidates. First, (equation 1) selects the candidates to be updated; similarly to the forget gate, a combination of the previous output $\mathcal{H}_{t-1}$, the memory C_{t-1} and the input $\mathcal{X}_t$ plus a bias b_f is made using once again the sigmoid function. Then, the selected candidates from the previous equation are updated using a tanh function (see figure 7).

$$i_t = \sigma\left(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \circ C_{t-1} + b_i\right) \tag{3}$$

$$\widehat{C}_t = \tanh\left(W_{xc} * \mathcal{X}_t + W_{hc} * \mathcal{H}_{t-1} + b_c\right) \tag{4}$$

Then the memory is filled with the updated candidates in equation 5.

$$C_t = f_t \circ C_{t-1} + i_t \circ \widehat{C}_t \tag{5}$$

$$o_t = \sigma\left(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \circ C_t + b_o\right) \tag{6}$$

Afterwards, a final sigmoid function selects parts of the previous output $\mathcal{H}_{t-1}$ and the input $\mathcal{X}_t$ (equation 6) to be fed into the final equation 7. This final equation combines a piece of memory C_t using a hyperbolic tangent (tanh) function and the last output o_t to finally compute the output $\mathcal{H}_t$.

$$\mathcal{H}_t = o_t \circ \tanh\left(C_t\right) \tag{7}$$

4 Implementation of the algorithm

As mentioned before, the objective of this work is to achieve a more accurate prediction of delays. In order to capture the bus delays patterns across a network, a deep model is implemented. A model, is referred to as deep when the information goes through multiple layers before reaching the end. In our case, similarly to Petersen et al. (2019),the model has an encoding part using n encoders convolutional LSTM and n decoding. Note that the internal memory of the LSTM is reset every day, i.e. the model remembers information only from the same day as that for which the prediction is made and no further.

The data has been shaped as depicted in figure 8. For each sample, the model is fed with a three dimensions' matrix, containing for all the buses the features for all selected previous time step. It is selected according to the size of the windows used. The 3 selected features are the total delays of the bus, the delays due to travel time and the increased delays due to boarding and alighting. The relevance of the features is depicted in subsection 4.2.

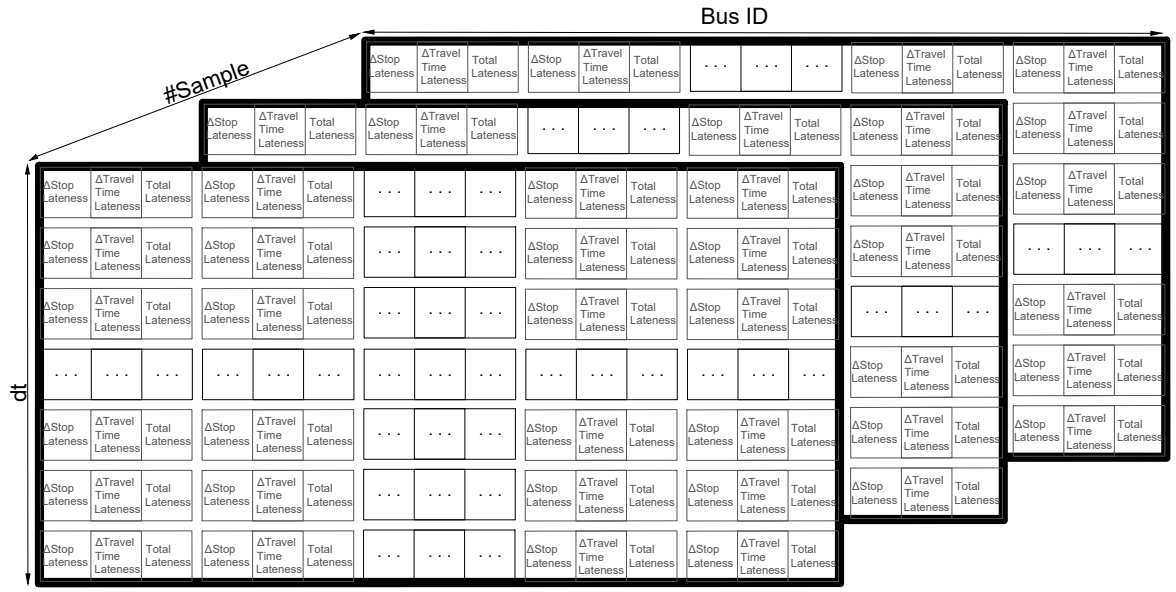
In order to train the model, it should be fed with the same data multiple times; each time is referred to as an epoch. The number of epochs is important; if there are too few, the model is stopped before capturing the complexity. If there are too many, the model will present really good performance, but it will only be due to over-fitting of the data. Essentially ,an over-fitted model makes no prediction, but just copies what he learns. To reduce this over-fitting risk, a dropout is implemented (discussed below) and the error is calculated on another data set, called validation. Seven random days have been selected as validation data set. It is important to select entire day as the validation data and not random sample from different days. Indeed, the internal memory needs to have a complete day to perform at its best.

In order to select the optimal model, an optimisation needs to be performed on different criteria (see table 1). A more in depth description of the tuning is given in subsection 4.1.

Table 1: Tuning and range of the different hyper-parameters.

Parameter	Range	Best
Number of encoders (n)	[1 2 3]	1
Size of filters	[16 32 64]	32
Convolution window	[1 5 10]	10
Dropout	[0.05 0.1 0.2 0.35 0.5]	0.5

Figure 8: Representation of the shape of the data.



The code can be found in the following link: `https://github.com/elingenior/LSTM_for_bus_lateness_prediction` (The code will be further cleaned)

4.1 Parameter Tuning

In order to find the best model, multiple models have been trained with different hyper parameters and only for the line 32. The mean absolute error (see equation 8) on the validation data and the computational time to perform 30 iterations have been used as metrics and a combination of the metrics will be used to select the model.

The hyper parameters are, the kernel size, which is the number of neighbouring bus taken into account; the filter size, that defines for the convulotional layer the number of kernels used in the gate. Last, the dropout rate is a factor, randomly dropping nodes in order to prevent over-fitting by droping some random node at each iteration.

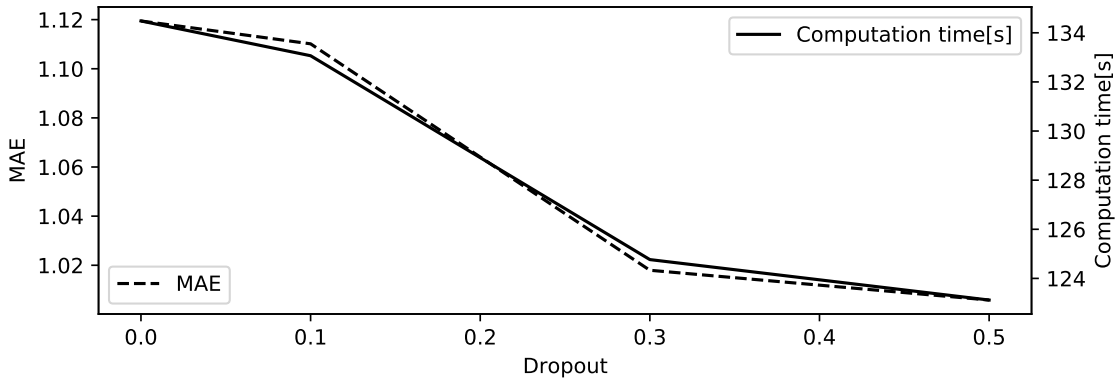
$$\text{MAE} = \frac{\sum_{i=1}^{n} \left| y_{\text{pred}_i} - y_{\text{true}_i} \right|}{n} \tag{8}$$

The numbers of layers n have been selected between 1 and 3 .The number of layers is limited to 3, since increasing the number of layers has negative impact on computation time.

The kernel is another name for the window. The range has been selected from 1 to 10. Computationally, it make more sense to have a filter as a power of 2.

In light of the results of figure 9, the dropout indeed reduces the error. As some nodes are dropped it also reduce the computation time, but on a really small scale. For later calculation a dropout of 0.5 is selected. Due to risk of instability in the model it is conventionally rare to select a dropout above 0.5.

Figure 9: Tuning of the dropout.

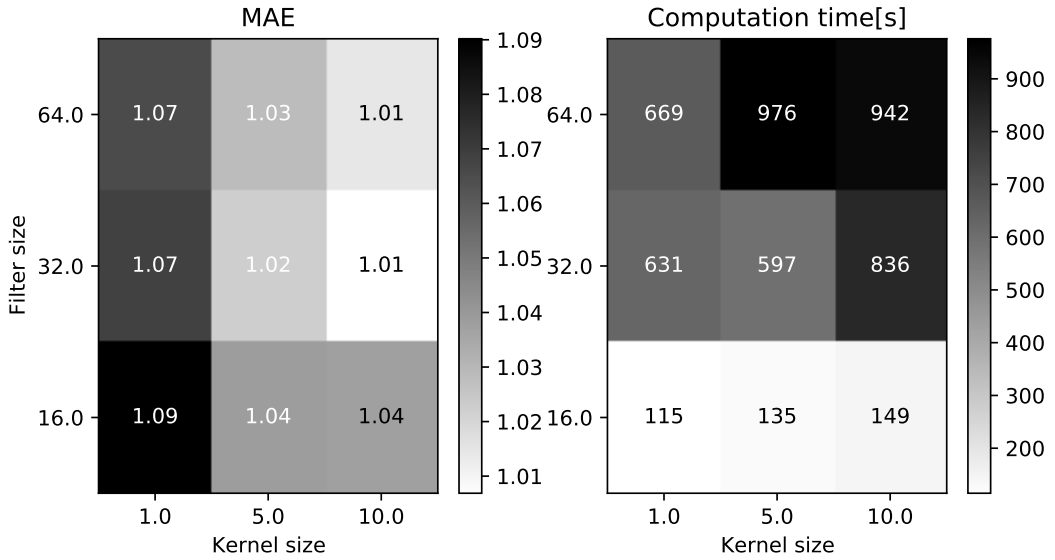


The optimisation takes place as follow; first all possible models for $n = 1$ have been trained on 30 epochs and then the Mean Absolute Error (MAE) for the validation data set has been computed.

In figure 10 the MAE values for the different kernel and filter size are presented.A reduction on the computation time as the number of filters decrease can be observed. It is not really surprising, as less computation effort is needed.

Unlike the first hyper-parameters, it seems that the Filter and Kernel size have a very strong influence on the Time as well as on the MAE. The best combination seems to be a Kernel size of 10 and a Filter size of 32.

Figure 10: Dependency of the Kernel and Filter size.



From those results, deeper models have also been computed. However, as shown in table 2, the model with a unique layer performs better. It was expected to be faster but is also surprisingly gives a lower MAE. This finding is discussed in a later section (section 5.5). Figures illustrating the tuning for the others n can be found in Appendix A.1.

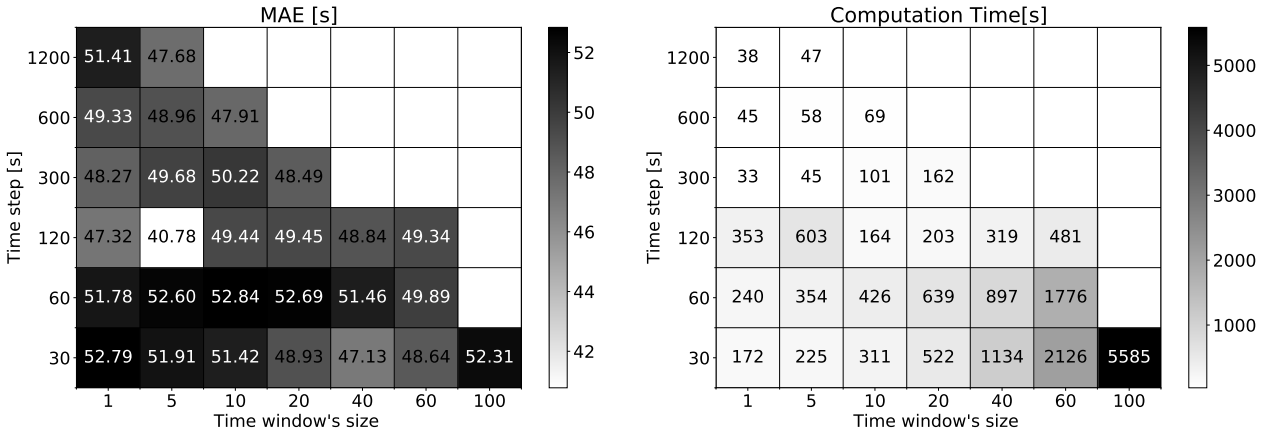
Table 2: Error and time for the best of each n.

Model	MAE	Time [s]
Number of layers (n) = 1	0.697	2.798e+03
Number of layers (n) = 2	0.723	1.493e+03
Number of layers (n) = 3	0.704	8.655e+03

The question about the time step also has to be discussed, as well as the number of previous time steps used to predict the next one. These two parameters depend on the horizon of the prediction. Indeed, different time steps are needed to predict 2 or 20 minutes ahead. Once again different model have been computed to select the best parameters. The objective was getting the best $t + 20[min]$ prediction. Results are presented in figure 11. The 2 minutes time step with a window size of 10 is much better than any other time step. This surprising results can be difficult to explain; it would appear that 2 minutes is enough to captures the important behaviours without actually be overloaded by information. If the models would have been trained with more epochs maybe the results would have been different.

It has to be noted that for each prediction time different time step are optimal. It is also true for every hyper parameters. This make the tuning of the model really demanding and time consuming.

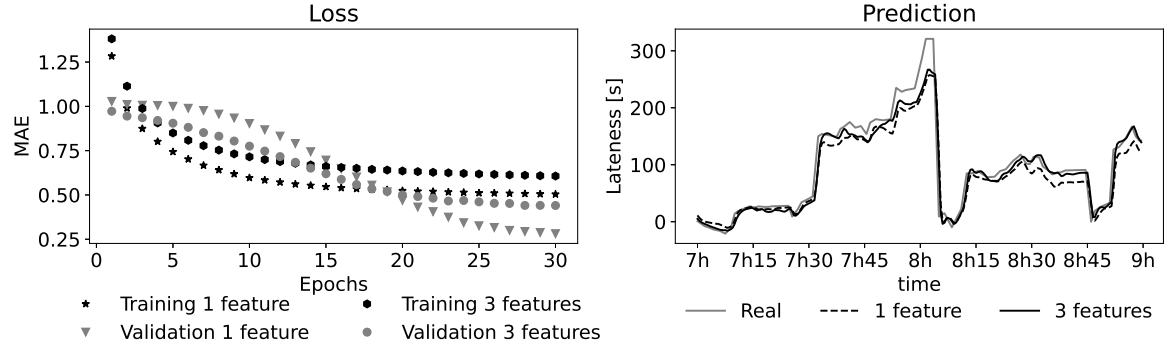
Figure 11: Performance for the different time step.



4.2 Features relevance

As stated previously, 3 features have been selected. In the literature and especially in Petersen et al. (2019), only the travel time delays were computed. The question arises whether using 3 features improves the model. The results in figure 12 show that the implementation of multiple features increase the accuracy of the model. It is still interesting to note that the 1 feature model converges faster. The reduced number of variables is the most likely explanation.

Figure 12: Relevance of 3 features compared to the literature.

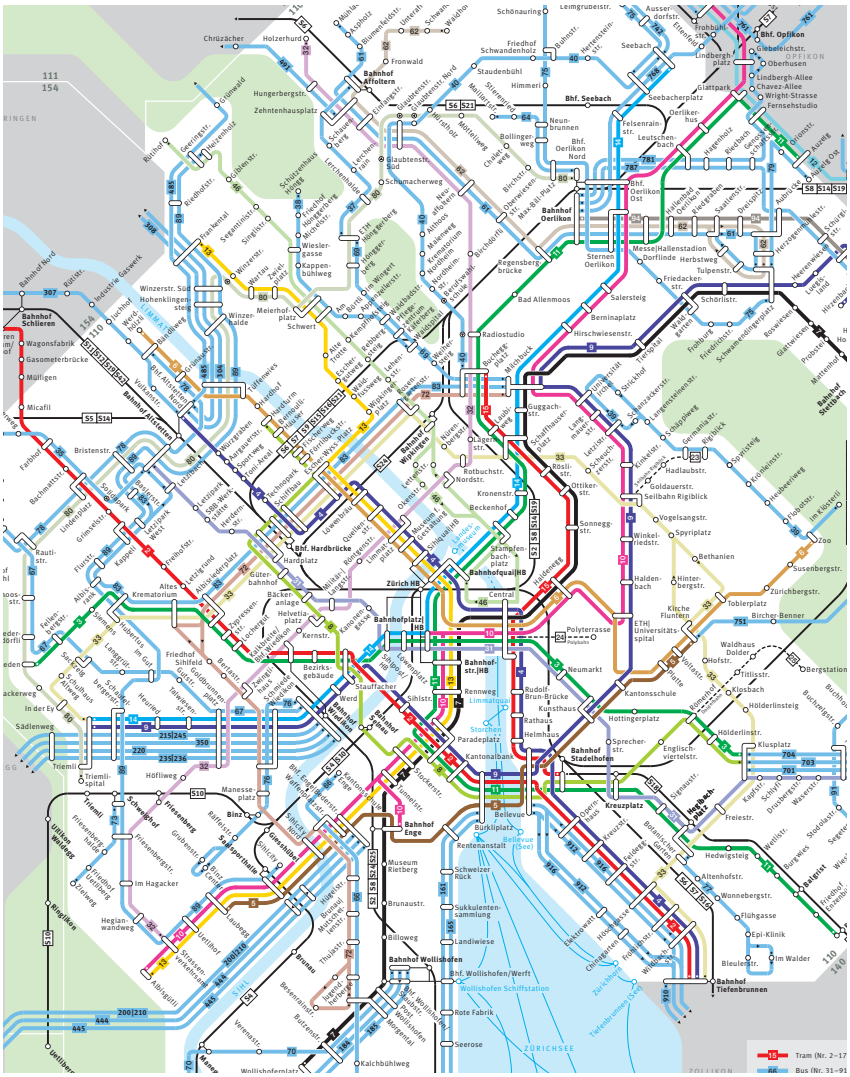


5 Case study

In order to test the ability of the proposed model to provide short-term delay predictions, a case study is designed. More specifically, publicly available AVL data for the city of Zürich (2020) are used for that purpose. The data contains the arrival and departure time at each stop for every bus in Zürich. Consequently, from this raw data the delay of each trip can be inferred.

The data set is made of 35 working day from the 1st of January to the 21st of February. The weekend days and holidays are discarded as the focus lies on typical weekdays. Only the peak hours are selected when the delays are the most likely to occur. The model has been trained on 29 working day and 6 days have been used as a validation set to prevent over-fitting.

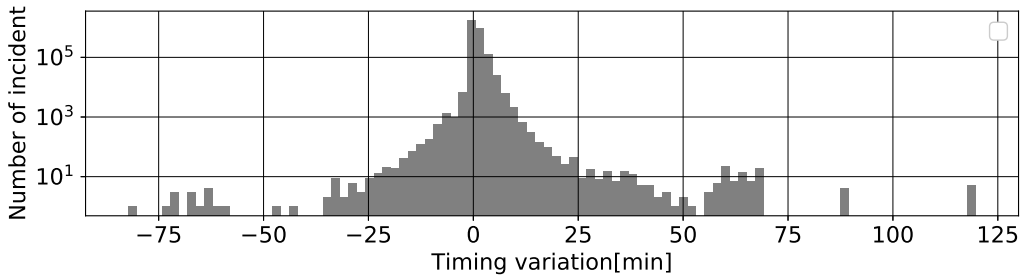
Figure 13: Map of the Zürich's network.



5.1 Description of data set

The data set corresponds to the lines of the greater Zürich area, namely, 663 stops serviced by 69 lines (roughly the zone 110 of the VBZ network, highlighted in white in figure 13).

Figure 14: Probability distribution function (PDF)



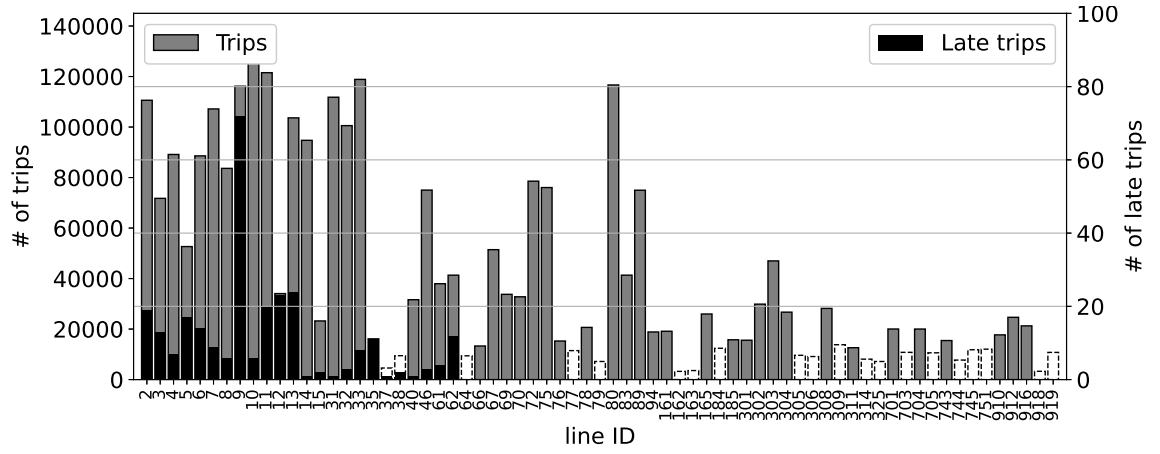
In figure 14 the delays in the network are presented. Taking note of the log scale, it appears that small delays dominate. Indeed, if one counts a time difference between the actual and planned arrival time of more than 2 minutes as a delay, in our data only 0.009% of all the trips are subject to delay (where a trip is defined as a route between two stops).

Concerning the observed large delay values, they can be attributed to either reporting error, or technical problems with the vehicle. Nevertheless, given the very sparse existence of such events, it is not necessary to discard them as the model will recognize them as such.

Figure 15 shows the line distribution in the data set. One can observe the under-representation of some lines, highlighted in white.

The majority of those underrepresented lines correspond to lines leaving the study area and therefore stop recorded data, and as such incomplete information concerning their trips may resolved in model malfunctioning. Therefore, those lines are discarded from the respective data sets.

Figure 15: Distribution of the line set and their delays.



5.2 Data processing

Some modifications have been made to the data set, prior to the model estimation. First, all the runs to the depot have been deleted. Then a unique run ID has been created; according to the following equation;

$$uniqueID = 100 * line_{ID} + line_{kursID} \tag{9}$$

Subsequently, a normalisation on all 3 features is performed. The model is surprisingly sensible to the chosen normalisation. In this regard, a min-max normalisation was tested, i.e. value between -1 and 1 only. However, this type of normalisation yields models with unreasonable results. To overcome this issue, the following normalisation has been chosen instead:

$$\mathcal{D} = \frac{\mathcal{D} - \bar{\mathcal{D}}}{\sqrt{\sigma_{\mathcal{D}}}} \tag{10}$$

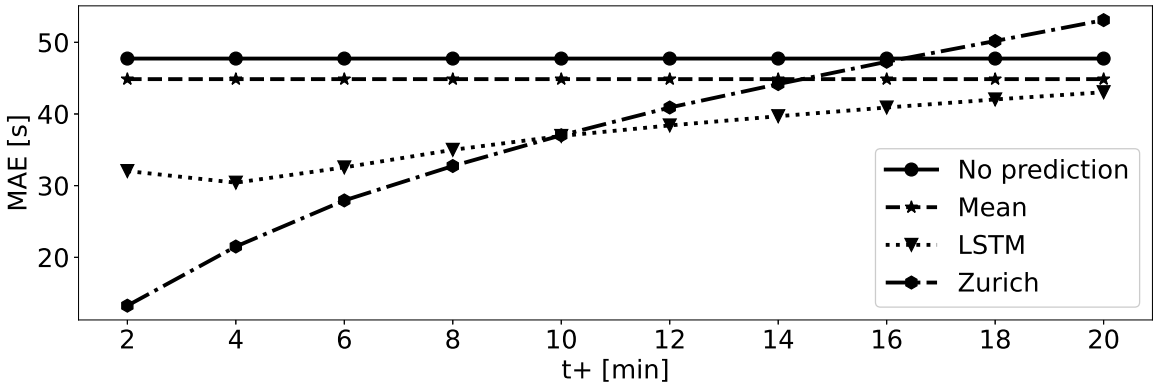
With $\mathcal{D}$ being the data set, $\bar{\mathcal{D}}$ its mean and $\sqrt{\sigma_{\mathcal{D}}}$ its standard deviation.

5.3 Time prediction performance

In this section, the performance of the proposed model is compared to 3 different delay prediction approaches. The first approach involves no prediction,i.e. the buses are assumed to be always on time. In the second approach, the delay of a bus is assumed to be equal to the mean delay of all the previous days, i.e. the bus always has the same delay everyday. Last, in the third approach that is presumed to be the one utilized currently by the operator, no adaptation of the delays through time is considered, i.e. it is assumed the bus will have the same delays in the future time steps without any correction.

The predictions for the next 20 minutes with a time step of 2 minutes are compared in figure 16

Figure 16: Time prediction for different time step.

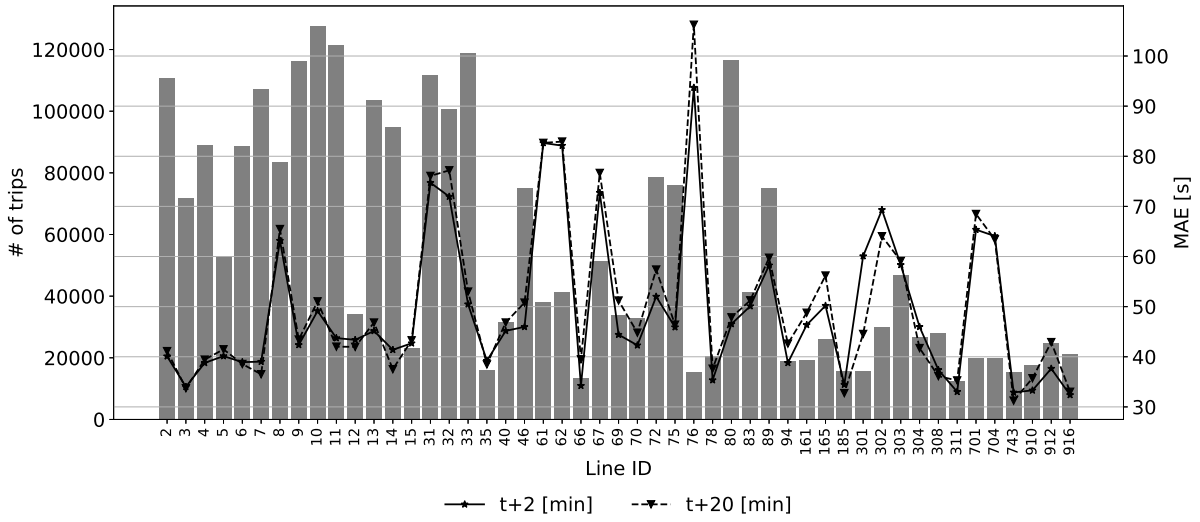


First, it is interesting to notice the performance of the method currently used. Indeed, for short prediction, Zürich's model is performing far better than our model. However, for medium to long term prediction, our model could improve the accuracy of the predictions. It is also interesting to notice that the LSTM performed better than the no prediction case, even in the case of Zurich which features so few delays. .

It is also interesting to see performance per line; as shown in figure 17, the model's performance varies substantially between the lines. For some lines the prediction for $t + 20$ [min] is better than for $t + 2$ [min]. Those line are lines leaving the central area, therefore the delays become stable once the vehicle has left the zone. It is then rather easy for the model to predict that the delay will be constant.

A good result that can also be seen in figure 17 is the consistency of the delays prediction error for the tram lines (from 2-14). Indeed, the prediction is rather constant while increasing the prediction lead time. As trams are the backbone of the urban transportation, having a good reliability on those lines is important.

Figure 17: Model performance per line.



5.4 Performance under extreme condition

Among all the days in the data set, one had a higher number of late trips, see figure 18. After verification, it appeared that it snowed that day. It is a good opportunity to see how the model reacts in that kind of more extreme situations.

Figure 18: Distribution of lateness for each day of the full data set.

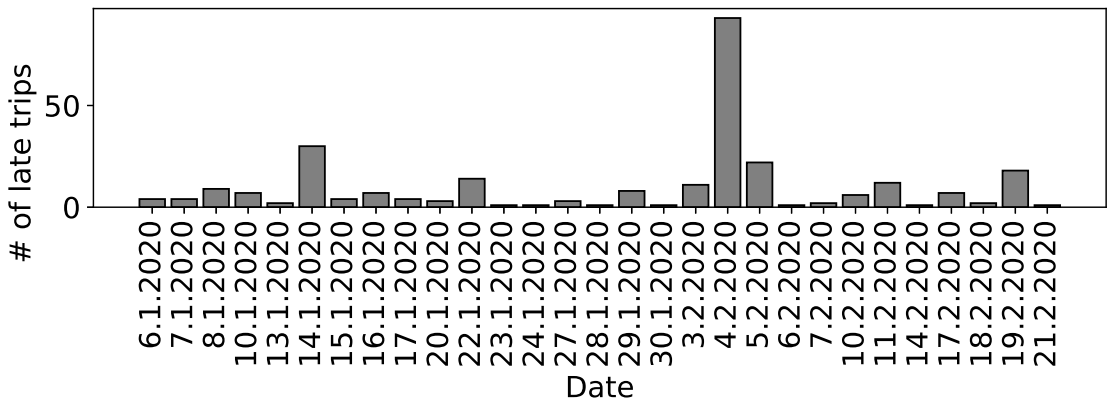
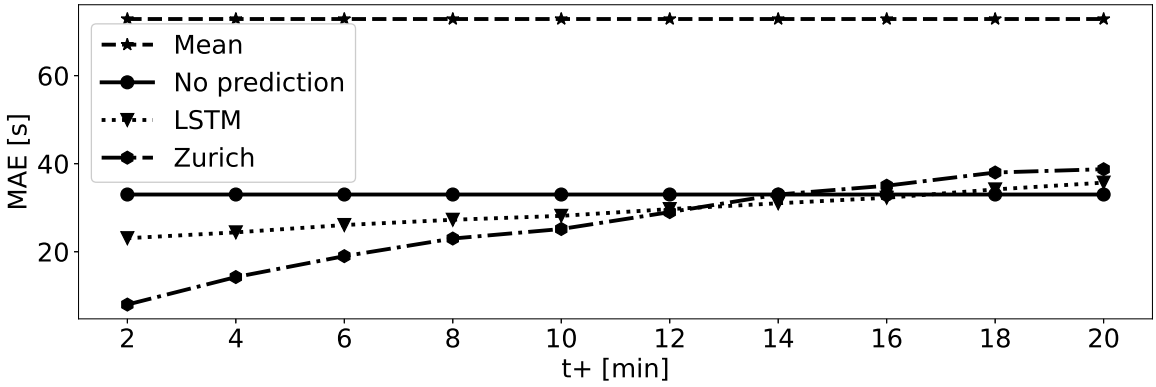


Figure 19 shows that the model fails to predict better when there are more delays in the network. An explanation is given in subsection 5.5. However, it is impressive to see how in those circumstances the mean techniques is by far the worst, even worse than the "no prediction". An explanation might be that some buses have a mean negative delays (so that they are on average in advance on the schedule), explaining why the mean technique is even worst than the "no prediction" one.

Figure 19: Time prediction under extreme condition.



5.5 Discussion

The methodology that we have developed leads to an improvement in the accuracy of delay predictions for the case study. However, due to the punctuality of public transport in the city of Zurich, the improvement is not huge. Indeed, as most of the trips are on-time the model has trouble to identify a pattern of delays. A solution to further test the potential of the methodology could be to select another network as case study (which has not been done due to time restrictions), or to perform an unbalanced data set correction. The idea would be to either create some realistic high delays from the few one already collected, or delete a range of really low delays. By doing so, the data will not be overwhelmed anymore by the majority of small delays. Haibo He and Garcia (2009) developed an approach to cope with that kind of issues. Another solution could be to simply train the model where days with high delays are as represented as "normal" day. With either of these approaches, the model could then be used as an emergency model, i.e. when the network does not follow an every day routine but was affected by some external factors like snow or public demonstrations.

6 Conclusion

The results that have been shown in this study provide some insights about the functionality of the proposed methodologies. The multi-features analysis manages to overcome previous methodology from the literature. It also confirmed that LSTM models can be applied for delays prediction not only by inferring them from congestion but directly from the delays themselves.

However, due to its sensitivity, each prediction time have different optimal parameters, making it difficult to have a single model for a large time prediction. Having one model per time prediction would be feasible and interesting but will require a higher computational effort.

The case study in Zürich also shows the possibility to implement the model in a well performing network. Indeed, the upside compared to easier methods is not very large but still significant.

For future work, a comparison between a stop based model and a bus based model could be performed on the same data set . A development relevant for machine learning, could be to run the model backwards, i.e. to understand on what basis is the model making prediction. Indeed, the lack of information about its actual understanding is one of the biggest weakness of machine learning algorithms. Some research have already been done to try to solve this black box problem (e.g. Lundberg (2018)).

7 References

Agafonov, A. A. and A. S. Yumaganov (2019) Bus Arrival Time Prediction Using Recurrent Neural Network with LSTM Architecture, *Optical Memory and Neural Networks*, **28** (3) 222–230, ISSN 1060-992X, 1934-7898.

Chen, C., H. Wang, F. Yuan, H. Jia and B. Yao (2019) Bus travel time prediction based on deep belief network with back-propagation, *Neural Computing and Applications*, ISSN 0941-0643, 1433-3058.

Chollet, F. and others (2015) *Keras*, GitHub.

Genser, A., L. Ambühl, K. Yang, M. Menedez and A. Kouvelas (2020) Time-to-Green: A framework to enhance SPaT messages using machine learning (working paper), 23rd IEEE International Conference on Intelligent Transportation Systems, September 2020.

Haibo He and E. Garcia (2009) Learning from Imbalanced Data, *IEEE Transactions on Knowledge and Data Engineering*, **21** (9) 1263–1284, ISSN 1041-4347.

Huang, Z., Q. Li, F. Li and J. Xia (2019) A Novel Bus-Dispatching Model Based on Passenger Flow and Arrival Time Prediction, *IEEE Access*, **7**, 106453–106465, ISSN 2169-3536.

Jenelius, E. (2018) Public transport experienced service reliability: Integrating travel time and travel conditions, *Transportation Research Part A: Policy and Practice*, **117**, 275–291, ISSN 0965-8564.

Kim, Y., S. Choi, S. Briceno and D. Mavris (2016) *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*., IEEE., S.l., ISBN 978-1-5090-2523-7. OCLC: 1096705549.

Liu, H., H. Xu, Y. Yan, Z. Cai, T. Sun and W. Li (2020) Bus Arrival Time Prediction Based on LSTM and Spatial-Temporal Feature Vector, *IEEE Access*, **8**, 11917–11929, ISSN 2169-3536.

Lundberg, S. (2018) Explainers — SHAP latest documentation, `https://shap.readthedocs.io/en/latest/#`.

Ma, X., Z. Tao, Y. Wang, H. Yu and Y. Wang (2015) Long short-term memory neural network for traffic speed prediction using remote microwave sensor data, *Transportation Research Part C: Emerging Technologies*, **54**, 187–197, ISSN 0968090X.

Olah, C. (2015) LSTM, Understanding LSTM Networks – colah's blog, `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

Petersen, N. C., F. Rodrigues and F. C. Pereira (2019) Multi-output bus travel time prediction with convolutional LSTM neural network, *Expert Systems with Applications*, **120**, 426–435, ISSN 09574174.

Zürich, O. D. (2020) Fahrzeiten der VBZ im SOLL-IST-Vergleich - data.stadt-zuerich.ch, `https://data.stadt-zuerich.ch/dataset/vbz_fahrzeiten_ogd`. Library Catalog: data.stadt-zuerich.ch.

A Appendix

A.1 Tuning for deeper network

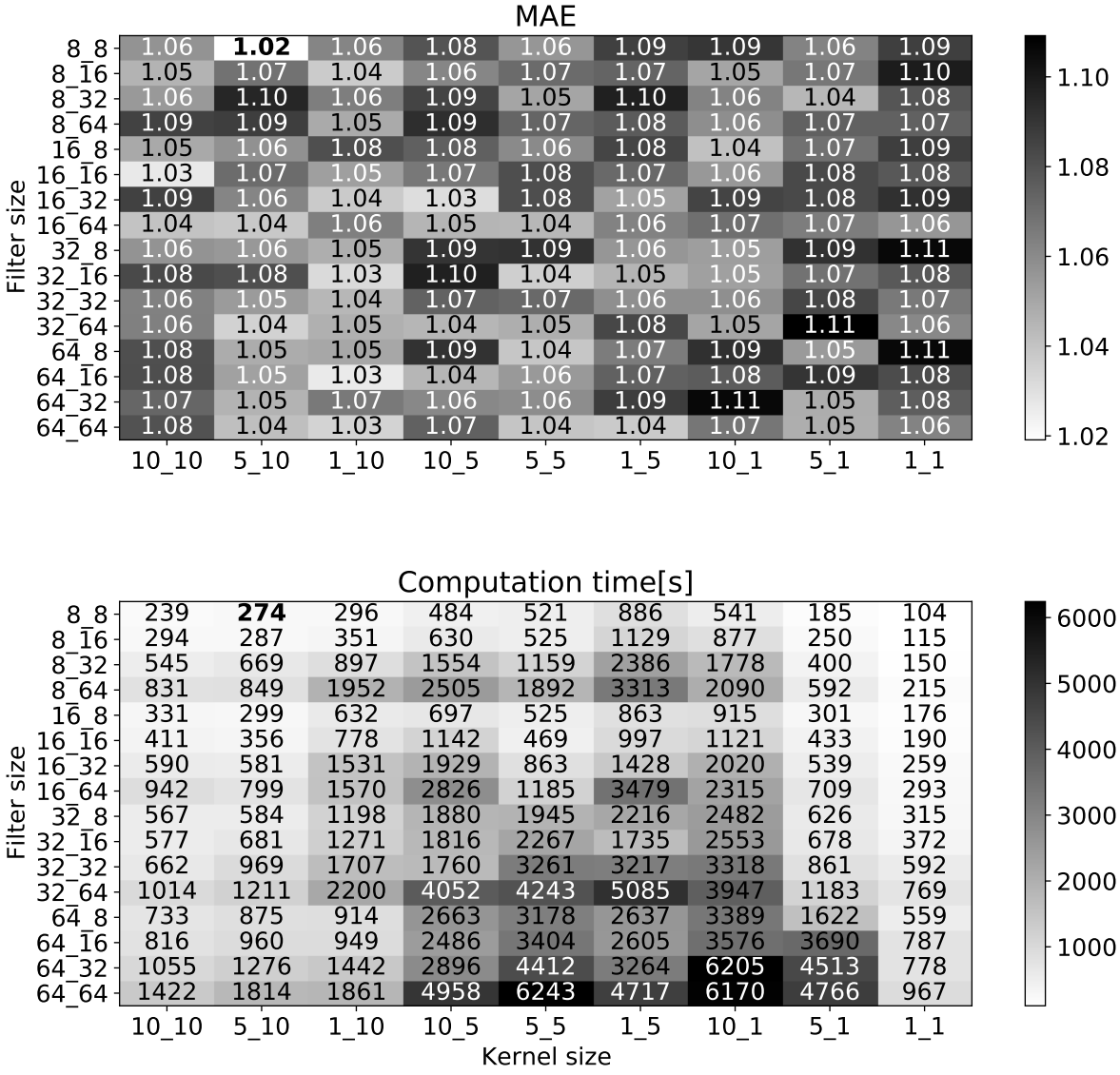
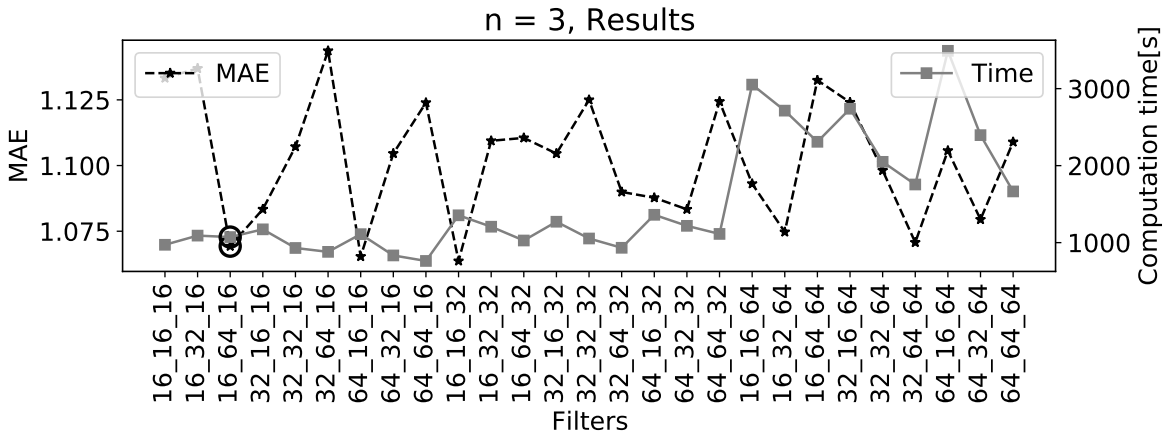


Figure 20: Tuning of the hyper parameters for 2 layers.

Selected values in bold

Figure 21: Tuning of the hyper parameters for 3 layers.



Selected values encircling