Multi-output Bus Travel Time Prediction Using Convolutional LSTM Neural Network

Niklas Christoffer Petersen

Department of Management Engineering
Technical University of Denmark, 2800 Kongens Lyngby
niklch@dtu.dk

Abstract

Accurate and reliable predictions for travel times in a public transport networks is essential for delivering an attractive service. This paper presents a multi-output, multi-time-step, deep neural network for bus travel time prediction using Convolutional and Long short-term memory (LSTM) layers. The method is evaluated and compared to other popular approaches for link travel time prediction, and currently available services.

Keywords: Bus Travel Time Prediction, Intelligent Transport Systems, Convolutional Neural Network (CNN), Long short-term memory (LSTM)

1. Introduction

Public transport authorities has long found that GPS trajectory data from already deployed *Automatic Vehicle Location*-systems (AVL) can be used in *Intelligent Transport Systems* (ITS) [1]. Examples include real-time traffic information for passengers, e.g. departure boards, where studies has shown that reliable real-time information at bus stops has a statistical significant dampening effect on the perceived waiting time [2].

Besides real-time passenger information, robust arrival- and departure time predictions are necessary for operating more sophisticated ITS applications successfully, e.g. demand-adaptive transit systems, and *connection insurance* between different public transport services. Arrival/departure time prediction is commonly approached as a specialization of travel time prediction, where the predicted travel time is simply accumulated downstream the route to yield the arrival/departure time predictions at each stop point of the rest of the current journey. Besides the link travel time, estimations of dwell time (i.e. when a bus is holding at a stop point) are also accumulated downstream.

Producing precise bus travel time predictions in areas with little external influence, e.g. rural areas, can be solved to a large extent with historical averaging or simple regression methods [3, 4]. The problem becomes much more complex in urban areas where congestion, events, road-works, weather, etc. highly influences the traffic flow and passenger demand.

In this paper we presents a multi-output, multi-time-step deep neural network for bus travel time prediction using a combination of *Convolutional* and *Long short-term memory* (LSTM) [5, 6] layers. The goal of this work is to produce precise short-term predictions (e.g. $0-1.5$ hours) for link travel time, specifically for bus traffic in urban areas. The method is evaluated and compared to other popular approaches for link travel time prediction, and currently available services.

The paper is structured in the following manner: In the next section related work and literature is reviewed. Section 2 introduces Convolutional LSTM neural networks in general and in Section 3 we present the proposed multi-output model in more details, including e.g. network topology, etc. Section 4 introduces the Copenhagen-dataset, which the model has been evaluated on, and our results are presented and discussed in Section 5. Finally we conclude on the work in Section 6.

1.1. Literature review

Early approaches presents historical averaging models [7, 8], and linear regression [9]. Recent research presents this type of model only for comparison purpose, and the models are in all cases outperformed by the compared models [10, 11]. Kalman-filters, which by its capabilities of maintaining state between predictions, has shown interest from several studies, either as an independent model [12, 10] or in combination with other models [13, 14].

Both the different regression style models and the Kalman-filter models still has limited options for capturing fluctuations in travel and dwell time in a metropolitan bus system, since they to a large extent still is averaging and smoothing their response. E.g., the Kalman-filter's state is only directly accessible for the leading time-step, and thus not capable of finding long-distance patterns spanning over several links and/or over several time-steps. This is substantiated by [15] and [16] that finds artificial neural networks (ANN) outperforms Kalman-filter models.

The computational challenges of pure ANN approaches i.a. has sparked the interest for studying composite or hybrid models. Zaki et al. [13] uses a two-stage approach by combining offline ANN-models, with an adaptable/online Kalman-filter to yield a dynamic model. The model is not actually learning in the long term, but is able to adapt to temporal variations in the current travel time on a journey.

Some recent research recognizes that several routes can benefit from each other's predictions if they share some partial route segment, e.g. [17, 18, 14], however none of these approaches consider cross temporal correlations between different route segments, and only uses a small window for correlation with upstream links (e.g. max. 3 links).

Finally Yanjie Duan et al. [19] proposes to use LSTM model for general highway travel time prediction, and do predict multiple steps ahead, but only for a single link at a time, i.e. cross link (spacial) correlations are lost.

2. Convolutional LSTM neural networks

A Long short-term memory (LSTM) neural network is a special type of Recurrent Neural Network (RNN) which has proven robust for capturing long-distance dependencies [5, 6]. The important feature of a LSTM network is its capability to persist cell state, c_t, from previous observations across sequences of input (e.g. time), but also eliminate information which is considered unimportant. To allow this mechanism the persistence of information is controlled by three gates: *input gate*, *forget gate*, and *output gate*. Each gate yields a state value at time t, respectively i_t, f_t, and o_t, along with the cell output, h_t, cf. Equation (1), where $\circ$ denotes the entry-wise product.

$$\begin{aligned} i_t &= \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci} \circ c_{t-1} + b_i\right) \\ f_t &= \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf} \circ c_{t-1} + b_f\right) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right) \\ o_t &= \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co} \circ c_t + b_o\right) \\ h_t &= o_t \circ \tanh\left(c_t\right) \end{aligned} \tag{1}$$

Figure 1 illustrates the inner structure of a LSTM cell with peephole as proposed by [20]. Is has especially grown popular for predicting time series using methods evolved from [21], where fixed-length windows of time-series are generated and feed into a LSTM network. Multiple LSTMs can be stacked such more complex patterns of sequential information (e.g. temporal patterns) can be learned.

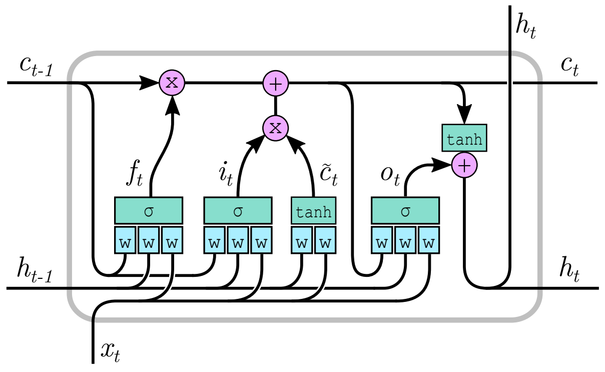


Figure 1: Structure of LSTM cell with peephole.

Convolutional Neural Networks (CNN) on the other hand has been widely used for capturing spacial relationships, e.g. importance of neighboring pixels in an image. As opposed to fully connected layers, where each unit, i, in the layer, has a dedicated scalar weight, w_{ij}, for all input value, x_j, convolutional units are only locally connected and reuses the same weights for producing several outputs. Instead of considering the entire input-vector, only a fixed sized window, or *convolution*, around each input is considered. The weights are therefore referred to as the *filters* or *kernels* of the layer. Figure 2 illustrates a single convolutional filter of size 3 being applied to 1-dimensional data.

Special care needs to be taken on the boundaries,

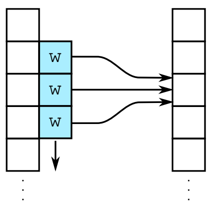


Figure 2: Application of Convolutional filter onto 1D data.

i.e. where the convolutional filter will exceed the input. A popular approach to avoid the size of the output decreases is to pad the input, e.g. with zeros. This ensures that the output shape of each convolutional unit will always be identical to the input shape, which is often desirable. One of the key benefits of convolutional networks is that the number of weights that needs to be learned is considerably reduced compared to fully connected networks, and that learned patterns can be transfered across space. I.e. the convolutional filters become feature detectors, that in our case can detect spacial patterns across links, e.g. congestion forming, etc.

Shi et al. [22] introduced the novel combination of Convolutional and LSTM layers into a single structure, the *Convolutional LSTM*, or simply *ConvLSTM*. Specifically the method applies convolutional filters in the *input-to-state* and *state-to-state* transitions the LSTM cf. Equation (2), where $*$ denotes the convolution operator.

$$\begin{aligned}
i_t &= \sigma\left(W_{xi} * x_t + W_{hi} * h_{t-1} + W_{ci} \circ c_{t-1} + b_i\right)\\
f_t &= \sigma\left(W_{xf} * x_t + W_{hf} * h_{t-1} + W_{cf} \circ c_{t-1} + b_f\right)\\
c_t &= f_t \circ c_{t-1} + i_t \circ \tanh\left(W_{xc} * x_t + W_{hc} * h_{t-1} + b_c\right)\\
o_t &= \sigma\left(W_{xo} * x_t + W_{ho} * h_{t-1} + W_{co} \circ c_t + b_o\right)\\
h_t &= o_t \circ \tanh\left(c_t\right)
\end{aligned} \tag{2}$$

The output dimensionality of a *ConvLSTM* layer is like the traditional CNN-layer determined by the number of filters applied. However *ConvLSTM* requires a total of eight filters for each desired output, i.e. four *input-to-state* filters (W_{xi}, W_{xf}, W_{xc}, and W_{xo}) and four *state-to-state* filters (W_{hi}, W_{hf}, W_{hc}, and W_{ho}).

3. Multi-output model

In this section we present the multi-output, multi-time-step model for bus travel time prediction, which uses the *ConvLSTM* introduced in the previous section.

3.1. Network topology

Figure 3 shows the overall network topology, where blue boxes illustrates *input-to-state* convolutions and yellow *state-to-state* convolutions. The network uses an encoder/decoder technique that to a large extent follows [22], where the encoder-block consists of two *ConvLSTM* layers. The result is feed into a decoder, or prediction block, also consisting of two *ConvLSTM* layers. The architecture allows unequal w and k, e.g. predict the next 3 time-steps based on a window size of 20.

Thus convolutional filters are applied to each input, at each time-step, to the respective LSTM-cell, and also between LSTM-cells in the state-transition. Since the time-steps are one-dimensional (i.e. link travel times across links), the filters are also one-dimensional. In each of the two blocks, the *ConvLSTMs* are arranged with filter sizes of respectively 10×1 and 5×1 for each of the layers in the block. This size is used both for the *input-to-state* and *state-to-state* convolutional filters. Finally each *ConvLSTM* layer 64 outputs, yielding a total of 512 convolutional filters.

To avoid over-fitting doing training *Dropout* [23] is performed between the *ConvLSTM* layers, and *Batch Normalization* [24] are also performed before each *ConvLSTM* layer to ensure reasonable input for activations. The dropout factor is adjusted to respectively 20%, 10% and 10%.

Each of the *ConvLSTM* layers uses linear activation functions, and the output from the last layer in the decoder-block is feed into a fully connected (FC) layer using the *ReLU* activation function, which also ensures only positive travel times are predicted.

3.2. Data preparation

We expect link travel times from AVL-systems to be available in a tabular form, where each link travel time measurement has a timestamp, and a reference to the link as illustrated in Table 1.

For the *ConvLSTM* model to be able to capture the desired spatial-temporal patterns, the input data must be arranged in a suitable manner: I.e. in N samples, each with a window of the w lagging time-steps $t-w+1,\ldots,t$, and each time-step with u link travel times $1,\ldots,u$ cf. Figure 4.

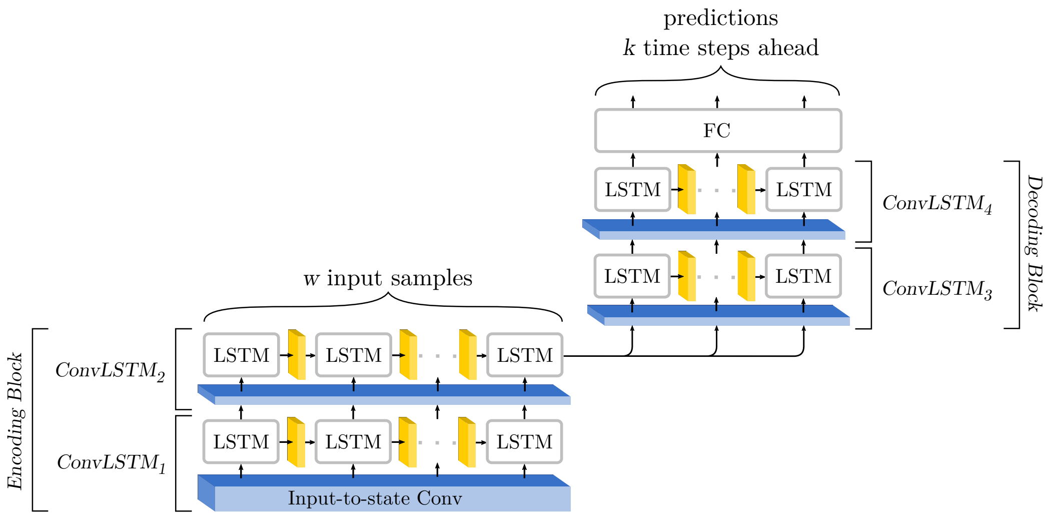


Figure 3: Convolutional LSTM network topology.

Timestamp	Linkref.	Link travel time (s)
2017-10-10 00:20:02	29848:1254	63
2017-10-10 00:21:07	1254:1255	65
2017-10-10 00:21:51	1255:10115	44
⋮	⋮	⋮

Table 1: Example of raw travel time measurements.

Likewise is the output arranged with N predictions of k time steps ahead, $t+1,\ldots,t+k$. Thus the input is a 4D-tensor with shape $(N,w,u,1)$, and the output a 4D-tensor with shape $(N,k,u,1)$ – in both bases the last one refers to the single link travel time for each time-step/link combination. It is emphasized that each prediction consists of travel time predictions for all links for the next k time-steps, i.e. multi-output, multi-time-step prediction.

The N samples are sampled at a fixed time resolution, since we need a shared time reference across all links. Section 4 elaborates on some of the considerations in choosing a sound resolution.

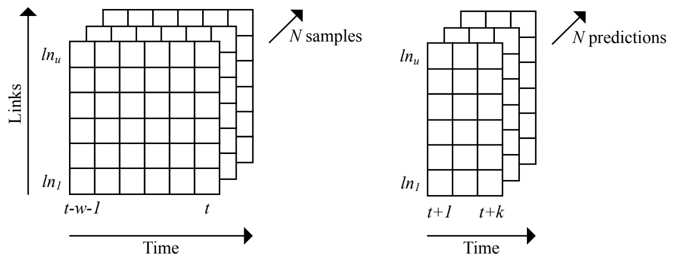


Figure 4: Shapes of the input and output data.

3.3. Detrending

Urban bus travel times varies throughout the time of the day, and the day of the week due to *recurring congestion*. In order to reduce the need for the network to learn this recurring variation, link travel for link $ln \in \{1,\ldots,u\}$, at time-step t, $x_{ln,t}$, is normalized to focus on deviations from the normal and expected pattern. Travel times are centered with the mean for each link, at the time of day, and day of week, $\bar{x}_{ln,dow,tod}$, and scaled with the standard deviation for each link, σ_{ln}, cf. Equation (3).

$$x'_{ln,t} = \frac{x_{ln,t} - \bar{x}_{ln,dow,tod}}{\sigma_{ln}} \tag{3}$$

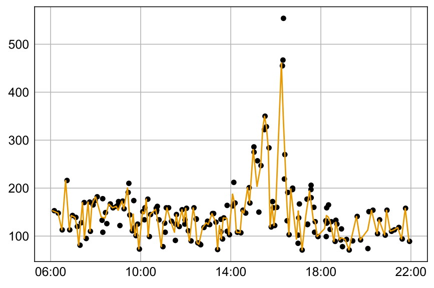
A similar normalization is applied to the predicted travel times, $y_{ln,t}$, but only using the historical mean and standard deviation, since the true mean and standard deviation obviously is unavailable in real-time prediction scenarios.

When calculating the mean and standard deviation it can be beneficial to exclude extreme outliers, since both mean and standard deviation are higly sensitive to such measurements. A suggested method is to apply *absolute deviation around the median* (MAD) cf. [25] when calculating $\bar{x}_{ln,dow,tod}$ and σ_{ln}.

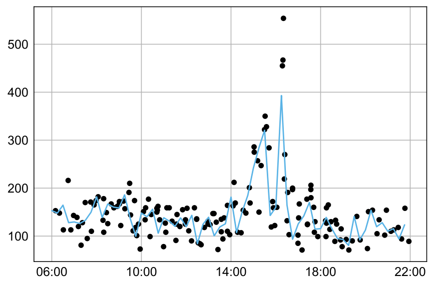
3.4. Training

The network is trained using the *RMSprop*-algorithm [26].

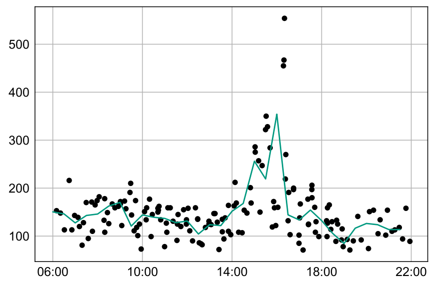
During training the variables $\bar{x}_{ln,dow,tod}$ and σ_{ln} should be calculated solely based on the training set, to emulated real-world application.



(a) 5 min



(b) 15 min



(c) 30 min

Figure 5: Examples of travel time for a single link at various time resolutions.

4. Experiments

For evaluation the method is applied to a dataset from Copenhagen's public transport authority, "Movia". The dataset consists of 1,2 M travel time observations for the "4A" bus line in the period January 1 to August 31 2017. The data points were collected using the real-time AVL-system installed in all vehicles of the line.

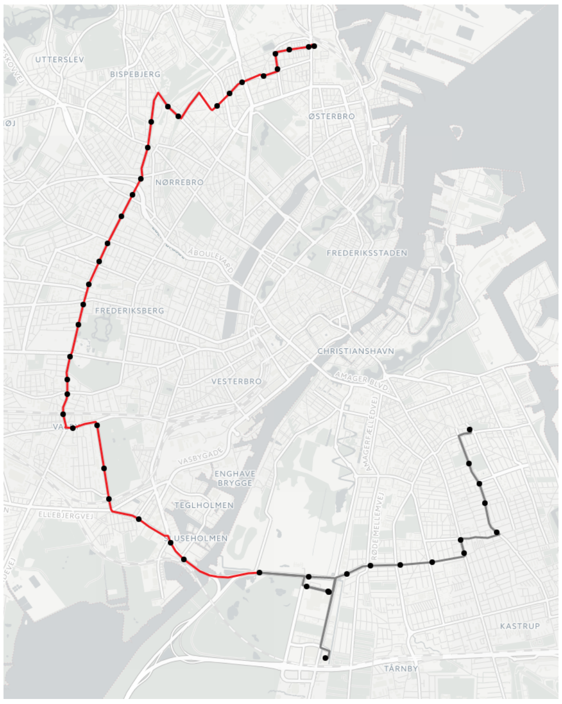


Figure 6: Geography of the 4A bus line in Copenhagen.

The geography of the route is shown in Figure 6. As the line circles Central Copenhagen, it is potentially highly sensitive to congestion to/from the city as it intersects with several large corridors along its route. Southeast of the city center the lines splits in different destination patterns (gray), therefore only the first 32 links are considered in this experiment (red).

4.1. Time resolution

To allow predictions for fixed time-steps ahead, data is aggregated at a fixed resolution. Figure 5 shows examples of travel time for a single link and a single day at various time resolutions. The black dots are actual measurements, and the lines the aggregated link travel time at the given resolution. The choice depends of the *expected* frequency of the line, and is a balance between capturing the details and still having a reasonal number of measurements in each time-step.

For this expirement data was aggregated into 15-minute time-steps and normalized cf. Section 3.

The implementation of the proposed model was done in Python using Keras [27].

4.2. Evaluation

In order to evaluate the presented model and comparisons, the following measures is used: *mean absolute error* (MAE), *root mean square error* (RMSE), and *mean absolute percentage error* (MAPE) cf. Equations (4) to (6), where X_i is the true travel time for sample i and $\widehat{X}_i$ is the predicted travel time.

$$\text{MAE}(X, \widehat{X}) = \frac{\sum_{i=1}^{n} \left| X_i - \widehat{X}_i \right|}{n} \tag{4}$$

$$\text{RMSE}(X, \widehat{X}) = \sqrt{\frac{\sum_{i=1}^{n} \left(X_i - \widehat{X}_i \right)^2}{n}} \tag{5}$$

$$\text{MAPE}(X, \widehat{X}) = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{X_i - \widehat{X}_i}{X_i} \right| \tag{6}$$

5. Results and discussion

The model is trained on the prepared data using a sliding window approach to simulate real-world conditions where real-time travel time measurements arrives as a continuously data stream. Figure 8 shows the evolution of training and validation loss for each epoch for such a window, where the loss is the MAE if the individual links.

d

Table 2 shows the results of the presented method and compares to two other methods: 1) a nave historical average model, i.e. equivalent of just prediction the normalized value, $\bar{x}_{l,dow,tod}$ and 2) a pure LSTM model as proposed by [19]. Predictions are accumulated downstream on journey level to simulate for use in real-time bus arrival/departure time prediction.

From the table it is found that the Convolutional LSTM performs better than the compared methods. Although the advantage might seem small it is emphasized that evaluation measurements are averaging their response, and thus the increased accuracy can be much higher on individual journeys, especially if they experienced very irregular travel times.

5.1. Future work

For the prediction accuracy to be increased further it is proposed to apply ensemble/multi-model approaches. In this case the presented model can be included and used as a sub-model for the ensemble. Further research should be invested in the more rare, but highly impacting deviations, e.g. traffic incidents, extreme weather conditions, etc.

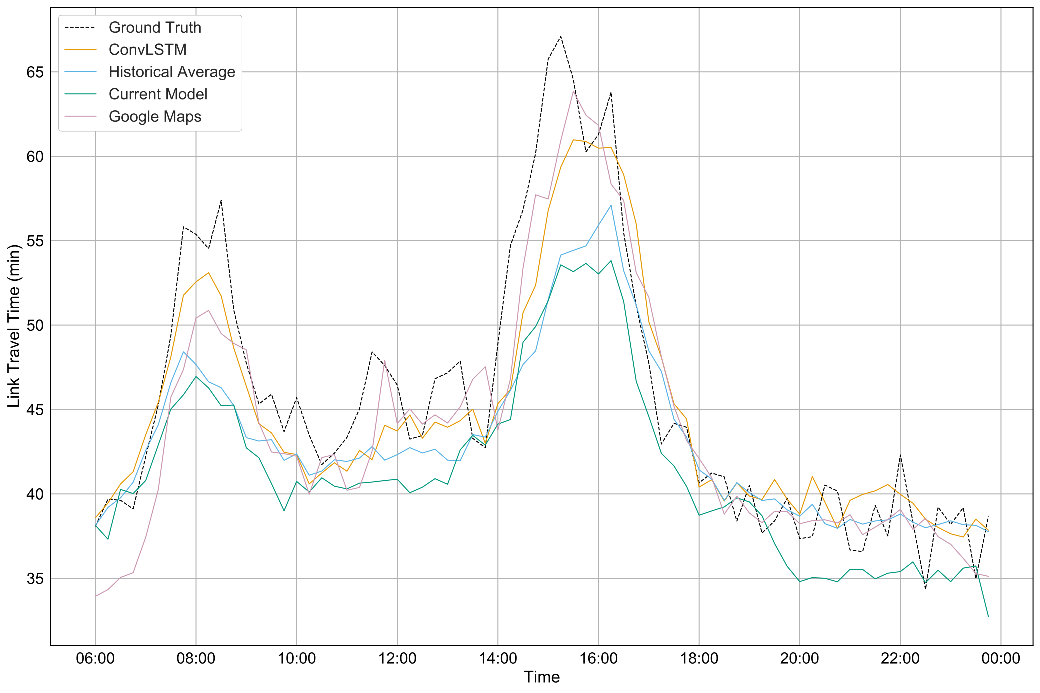


Figure 7

Table 2: Results of the presented and other popular models

Model	Time ahead	MAE (min)	RMSE (min)	MAPE (%)
Historical average		4.90	6.32	7.38 %
Current Model	t + 1 (15 min)			%
Current Model	t + 2 (30 min)			%
Current Model	t + 3 (45 min)			%
Pure LSTM	t + 1 (15 min)	3.71	3.71	5.87 %
Pure LSTM	t + 2 (30 min)	4.31	5.63	6.63 %
Pure LSTM	t + 3 (45 min)	4.97	6.41	7.52 %
Convolutional LSTM	t + 1 (15 min)	2.96	3.94	5.10 %
Convolutional LSTM	t + 2 (30 min)	3.04	4.08	5.21 %
Convolutional LSTM	t + 3 (45 min)	3.17	4.24	5.37 %

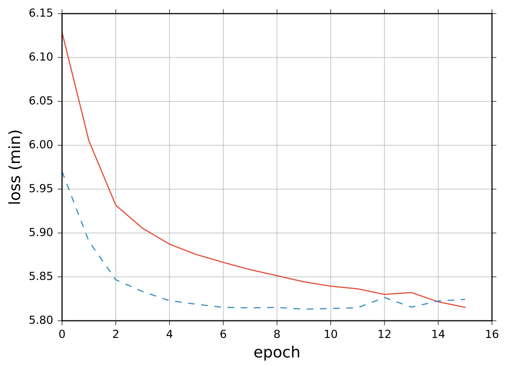


Figure 8: Training and validation loss evolution. TODO: Needs to be updated ...

6. Conclusion

This paper has presented a multi-output, multi-time-step deep neural network for bus travel time prediction using Convolutional and Long short-term memory (LSTM) layers. Results shown the presented network outperforms other popular and recent methods.

Future research opportunities include using the model a part of a multi-model ensemble, and focus in the more rare, but highly impacting deviations.

References

[1] C. Schweiger, TCRP Synthesis 48: Real-Time Bus Arrival Information Systems, ISBN 0309069653, 2003.

[2] Y. Fan, A. Guthrie, D. Levinson, Perception of Waiting Time at Transit Stops and Stations, Tech. Rep. 9, Center for Transportation Studies, University of Minnesota, 2016.

[3] B. M. Williams, L. a. Hoel, Modeling and Forecasting Vehicular Traffic Flow as a Seasonal ARIMA Process: Theoretical Basis and Empirical Results, Journal of Transportation Engineering 129 (6) (2003) 664–672, ISSN 0733-947X, doi:10.1061/(ASCE)0733-947X(2003)129:6(664).

[4] M. Altinkaya, M. Zontul, Urban Bus Arrival Time Prediction: A Review of Computational Models, International Journal of Recent Technology and Engineering 2 (4) (2013) 164–169.

[5] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural Computation 9 (8) (1997) 1735–80, ISSN 0899-7667, doi:10.1162/neco.1997.9.8.1735, URL http://www.ncbi.nlm.nih.gov/pubmed/9377276.

[6] F. A. Gers, J. Schmidhuber, F. Cummins, Learning to Forget: Continual Prediction with LSTM, Neural Computation 12 (10) (2000) 2451–2471, ISSN 0899-7667, doi:10.1162/089976600300015015, URL http://www.mitpressjournals.org/doi/10.1162/089976600300015015.

[7] D. J. Dailey, Z. Wall, An Algorithm for Predicting the Arrival Time of Mass Transit, in: Transportation Research Board 78th Annual Meeting, 990870, Transpotation Research Board, Washington DC., doi:10.1.1.579.2083, 1999.

[8] D. Sun, H. Luo, L. Fu, W. Liu, X. Liao, M. Zhao, Predicting Bus Arrival Time on the Basis of Global Positioning System Data, Transportation Research Record: Journal of the Transportation Research Board 2034 (2034) (2007) 62–72, ISSN 0361-1981, doi:10.3141/2034-08, URL http://trrjournalonline.trb.org/doi/10.3141/2034-08.

[9] J. Patnaik, S. Chien, A. Bladikas, Estimation of Bus Arrival Times Using APC Data, Journal of Public Transportation 7 (July 2017) (2004) 1–20, ISSN 1077-291X, doi:10.5038/2375-0901.7.1.1.

[10] A. Shalaby, A. Farhan, Prediction model of bus arrival and departure times using AVL and APC data, Journal of Public Transportation 7 (2004) 41–61, ISSN 1077-291X, doi:10.1.1.170.9999, URL http://www.nctr.usf.edu/wp-content/uploads/2010/03/JPT-7-1.pdf{#}page=46.

[11] R. Jeong, L. R. Rilett, Prediction Model of Bus Arrival Time for Real-Time Applications, Transportation Research Record: Journal of the Transportation Research Board 1927 (1927) (2005) 195–204, ISSN 0361-1981, doi:10.3141/1927-23, URL http://trrjournalonline.trb.org/doi/10.3141/1927-23.

[12] M. Chen, S. Chien, Dynamic Freeway Travel-Time Prediction with Probe Vehicle Data: Link Based Versus Path Based, Transportation Research Record 1768 (1) (2001) 157–161, ISSN 0361-1981, doi:10.3141/1768-19.

[13] M. Zaki, I. Ashour, M. Zorkany, B. Hesham, Online Bus Arrival Time Prediction Using Hybrid Neural Network and Kalman filter Techniques, International Journal of Modern Engineering Research (IJMER) 3 (2013) 2035–2041, URL http://ijmer.com/papers/Vol3{_}Issue4/BC3420352041.pdf.

[14] C. Bai, Z. R. Peng, Q. C. Lu, J. Sun, Dynamic bus travel time prediction models on road with multiple bus routes, Computational Intelligence and Neuroscience 2015, ISSN 16875273, doi:10.1155/2015/432389.

[15] Y. Lin, X. Yang, N. Zou, L. Jia, Real-Time Bus Arrival Time Prediction: Case Study for Jinan, China, Journal of Transportation Engineering 139 (11) (2013) 1133–1140, ISSN 0733-947X, doi:10.1061/(ASCE)TE.1943-5436.0000589, URL http://ascelibrary.org/doi/10.1061/{%}28ASCE{%}29TE.1943-5436.0000589.

[16] V. Kumar, B. A. Kumar, L. Vanajakshi, S. C. Subramanian, Comparison of Model Based and Machine Learning Approaches for Bus Arrival Time Prediction, TRB 93rd Annual Meeting Compendium of Papers .

[17] B. Yu, W. H. K. Lam, M. L. Tam, Bus arrival time prediction at bus stop with multiple routes, Transportation Research Part C: Emerging Technologies 19 (6) (2011) 1157–1170, ISSN 0968090X, doi:10.1016/j.trc.2011.01.003, URL http://dx.doi.org/10.1016/j.trc.2011.01.003.

[18] A. Gal, A. Mandelbaum, F. Schnitzler, A. Senderovich, M. Weidlich, Traveling time prediction in scheduled transportation with journey segments, Information Systems 64 (2014) 266–280, ISSN 03064379, doi:10.1016/j.is.2015.12.001, URL http://dx.doi.org/10.1016/j.is.2015.12.001.

[19] Yanjie Duan, Yisheng +Lv, Fei-Yue Wang, Travel time prediction with LSTM neural network, 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC) (2016) 1053–1058doi:10.1109/ITSC.2016.7795686, URL http://ieeexplore.ieee.org/document/7795686/.

[20] F. A. Gers, J. Schmidhuber, Recurrent nets that time and count, in: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, IEEE, ISBN 0-7695-0619-4, 189–194 vol.3, doi:10.1109/IJCNN.2000.861302, URL http://ieeexplore.ieee.org/document/861302/, 2000.

[21] F. A. Gers, D. Eck, J. Schmidhuber, Applying LSTM to time series predictable through time-window approaches, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 2130, ISBN 3540424865, ISSN 16113349, 669–676, doi:10.1007/3-540-44668-0_93, 2001.

[22] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, W.-c. Woo, Convolutional LSTM network: A machine learning approach for precipitation nowcasting, Advances

in Neural Information Processing Systems 28 (2015) 802–810ISSN 10495258.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Journal of Machine Learning Research 15 (2014) 1929–1958, ISSN 15337928, doi:10.1214/12-AOS1000.

[24] S. Ioffe, C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, Arxiv (2015) 1–11ISSN 0717-6163, doi:10.1007/s13398-014-0173-7.2, URL `http://arxiv.org/abs/1502.03167`.

[25] N. Olewuezi, Note on the Comparison of Some Outlier Labeling Techniques, Journal of Mathematics and Statistics 7 (4) (2011) 353–355, ISSN 15493644, doi:10.3844/jmssp.2011.353.355.

[26] G. Hinton, T. Tieleman, Lecture - Rmsprop: Divide the gradient by a running average of its recent magnitude, 2017.

[27] F. Chollet, Others, Keras, URL `https://github.com/fchollet/keras`, 2015.