Multi-output Bus Travel Time Prediction with Convolutional LSTM Neural Networks

Niklas Christoffer Petersen
niklch@dtu.dk

Department of Management Engineering
Technical University of Denmark, 2800 Kongens Lyngby

Public Transport Movia
Gammel Køge Landevej 3, 2500 Valby

Abstract

Accurate and reliable predictions for travel times in 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 empirically evaluated and compared to other popular approaches for link travel time prediction, and currently available services. We find that the proposed model significantly outperforms all the other methods, and is able to detect small irregular peeks in bus travel time very quickly.

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

1. Introduction

Public transport authorities have 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. smartphone travel planner apps, in-vehicle infotainment screens and departure boards. 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. *connection assurance* between low frequent public transport services, where accurate arrival times are needed to decide if one of the services should wait on the other. Another example is *demand-adaptive transit systems*, where prolonged travel time should be taken into account because of the influence on both vehicle and driver utilization, and thus the efficiency of the transit system. Arrival/departure time prediction is commonly approached as a specialization of travel time prediction as illustrated in Figure 1. The predicted travel time for each link 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.

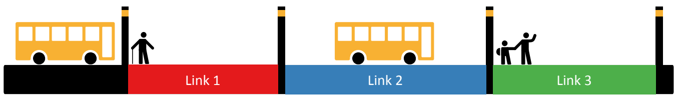


Figure 1: Arrival- and departure using link travel time.

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 influence the traffic flow and passenger demand. As on-board GPS and AVL-systems has become more affordable and common, data has both grown in coverage, i.e. number of vehicles with AVL installed, and frequency, i.e. number of GPS-positions collected per vehicle.

In this paper we present 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 empirically evaluated and compared to other popular approaches for link travel time prediction, including the currently model deployed in production by the public transport authority for the Greater Copenhagen Area: *Movia*. Furthermore the method is compared to Google Traffic, a popular online service for travel time prediction.

This paper is structured in the following manner: In the next section, related work and literature are reviewed. Section 3 introduces Convolutional LSTM neural networks in general and in Section 4 we present the proposed multi-output model in more details, including e.g. network topology, etc. Section 5 introduces the Copenhagen-dataset, which the model has been evaluated on, and our results are presented and discussed in Section 6. Finally we conclude on the work in Section 7.

2. Literature review

Bus link travel time prediction has been explored in research as GPS and AVL data has become increasingly available. The problem overlaps with other research areas such as general traffic flow and speed estimation, but also has unique constraints and opportunities that follows from servicing a fixed route. The improvement in computational power in the recent decades has gradually allowed more complex link travel time models with increased precision.

Early approaches for bus travel time prediction rely on historical averaging models [7, 8], and linear regression [9]. Recent research presents this type of model only for comparison purposes, and the models are in all cases outperformed by the proposed models [10, 11]. A historical averaging model will only slowly converge to changes in the travel time, which of cause is undesired with short, but highly impacting, external influences (e.g. a traffic incident or large event).

Kalman-filters, which by its capabilities of maintaining state between predictions, has shown interest from several studies for capturing *changes* in the travel time, either as an independent model [12, 10] or in combination with other models [13, 14]. The Kalman-filter models still have limited options for capturing and forecasting the complex dynamics of travel and dwell time in a metropolitan bus system. 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], who find artificial neural networks (ANN) to outperform Kalman-filter models. However, the computational challenges of fully connected ANNs is also limiting the depth of the network and thus how complex patterns it can learn to recognize. This has sparked the interest for studying composite or hybrid models. Bai et al. [14] 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, and is still not able to recognize long distance patterns. The proposed model in this work uses *Long short-term memory* cells (LSTM), and differs from existing research in bus link travel time prediction by combining the capability for persisting state-space over multiple time-steps, while allowing the deep neural network to be feasibly trainable.

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 use a small window for correlation with upstream links (e.g. max. 3 links). Likewise, Yanjie Duan et al. [19] proposes to use a LSTM model for general highway travel time prediction, and do predict multiple time-steps ahead, but only for a single link at a time, i.e. cross link (spatial) correlations are lost. In contrast the combination of both *LSTM* cells and *Convolutional* filters for bus travel time prediction, proposed in this paper, allows the learned patterns to generalize beyond a single link and time, i.e. multi-output and multi-time-step. Furthermore this reduces the computational complexity by magnitudes compared to fully connected ANNs capable of capturing similar complex patterns.

3. 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, $\mathbf{c}_t$, from previous observations across sequences of input (e.g. time), but also eliminate information which is considered irrelevant. 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 $\mathbf{i}_t$, $\mathbf{f}_t$, and $\mathbf{o}_t$, along with the cell output, $\mathbf{h}_t$, cf. eq. (1), where $\circ$ denotes the entry-wise product.

$$\begin{aligned}
\mathbf{i}_t &= \sigma\left(\mathbf{W}^i\mathbf{x}_t + \mathbf{R}^i\mathbf{h}_{t-1} + \mathbf{U}^i \circ \mathbf{c}_{t-1} + \mathbf{b}^i\right) \\
\mathbf{f}_t &= \sigma\left(\mathbf{W}^f\mathbf{x}_t + \mathbf{R}^f\mathbf{h}_{t-1} + \mathbf{U}^f \circ \mathbf{c}_{t-1} + \mathbf{b}^f\right) \\
\mathbf{c}_t &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tanh\left(\mathbf{W}^c\mathbf{x}_t + \mathbf{R}^c\mathbf{h}_{t-1} + \mathbf{b}_c\right) \\
\mathbf{o}_t &= \sigma\left(\mathbf{W}^o\mathbf{x}_t + \mathbf{R}^o\mathbf{h}_{t-1} + \mathbf{U}^o \circ \mathbf{c}_t + \mathbf{b}^o\right) \\
\mathbf{h}_t &= \mathbf{o}_t \circ \tanh\left(\mathbf{c}_t\right)
\end{aligned} \tag{1}$$

Figure 2 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 that more complex patterns of sequential information (e.g. temporal patterns) can be learned.

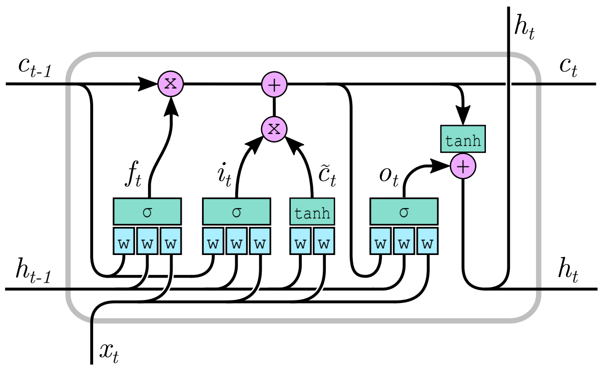


Figure 2: Structure of LSTM cell with peephole.

Convolutional Neural Networks (CNN) on the other hand have 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 reuse the same weights for producing several outputs. Instead of considering the entire input-vector, only a fixed-size window, or *convolution*, around each input is considered. The weights are therefore referred to as the *filters* or *kernels* of the layer. Figure 3 illustrates a single convolutional filter of size 3 being applied to 1-dimensional data.

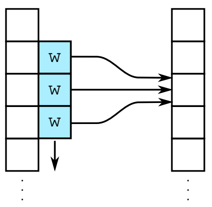


Figure 3: Application of Convolutional filter onto 1D data.

Special care needs to be taken on the boundaries, 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 spatial 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 of the LSTM cf. eq. (2), where $*$ denotes the convolution operator.

$$\begin{aligned}
\mathbf{i}_t &= \sigma\left(\mathbf{W}^i * \mathbf{x}_t + \mathbf{R}^i * \mathbf{h}_{t-1} + \mathbf{U}^i \circ \mathbf{c}_{t-1} + \mathbf{b}^i\right) \\
\mathbf{f}_t &= \sigma\left(\mathbf{W}^f * \mathbf{x}_t + \mathbf{R}^f * \mathbf{h}_{t-1} + \mathbf{U}^f \circ \mathbf{c}_{t-1} + \mathbf{b}^f\right) \\
\mathbf{c}_t &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tanh\left(\mathbf{W}^c * \mathbf{x}_t + \mathbf{R}^c * \mathbf{h}_{t-1} + \mathbf{b}_c\right) \\
\mathbf{o}_t &= \sigma\left(\mathbf{W}^o * \mathbf{x}_t + \mathbf{R}^o * \mathbf{h}_{t-1} + \mathbf{U}^o \circ \mathbf{c}_t + \mathbf{b}^o\right) \\
\mathbf{h}_t &= \mathbf{o}_t \circ \tanh\left(\mathbf{c}_t\right)
\end{aligned} \tag{2}$$

The output dimensionality of a *ConvLSTM* layer is like the traditional CNN-layer determined by the number of filters applied. However, *ConvLSTMs* require a total of eight filters for each desired output, i.e. four *input-to-state* filters ($\mathbf{W}^i$, $\mathbf{W}^f$, $\mathbf{W}^c$, and $\mathbf{W}^o$) and four *state-to-state* filters ($\mathbf{R}^i$, $\mathbf{R}^f$, $\mathbf{R}^c$, and $\mathbf{R}^o$). Still it is important to emphasize that application of convolutional filters to the LSTM model greatly reduces the number of weights that needs to be learned, compared to a *pure LSTM* approach. This allows for even deeper networks.

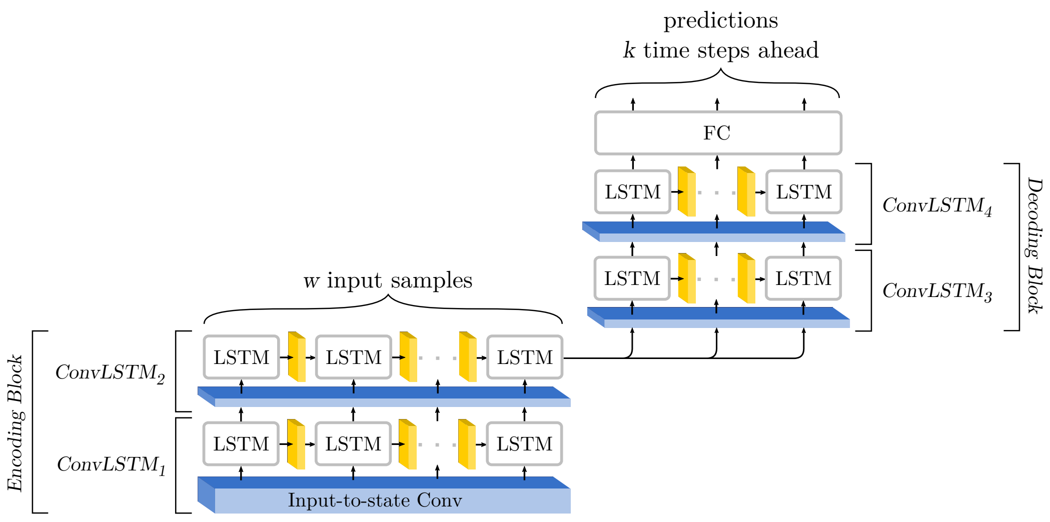


Figure 4: Convolutional LSTM network topology.

4. 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.

4.1. Network topology

Figure 4 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 fed 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 during training *Dropout* [23] is performed between the *ConvLSTM* layers, and *Batch Normalization* [24] is also performed before each *ConvLSTM* layer to ensure reasonable inputs for the activations. The dropout probability 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 fed into a fully connected (FC) layer using the *ReLU* activation function, which also ensures that only positive travel times are predicted.

4.2. 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.

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.

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 $t - w + 1, \dots, t$, and each time-step with u link travel times $1, \dots, u$ cf. Figure 5.

Likewise, the output is arranged with N predictions of k time steps ahead, $t + 1, \dots, t + k$. Thus the input is a 4D-tensor, $\mathbf{X}$ with dimensionality $N \times w \times u \times 1$, and the output, $\mathbf{Y}$, a 4D-tensor with dimensionality $N \times k \times u \times 1$ – in both cases 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 5 elaborates on some of the considerations for choosing a sound resolution.

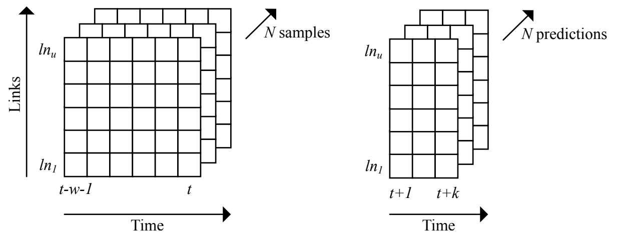


Figure 5: Shapes of the input and output data.

4.3. Detrending

Urban bus travel times vary 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. eq. (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 highly 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}.

4.4. Implementation and training

The proposed network model was implemented in Python using the Keras Framework [26], and trained using the *RMSprop*-algorithm [27].

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

5. Experiments

For the purpose of evaluation, the proposed 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 May to October 2017. The data points were collected using the real-time AVL-system installed in every vehicle servicing the line.

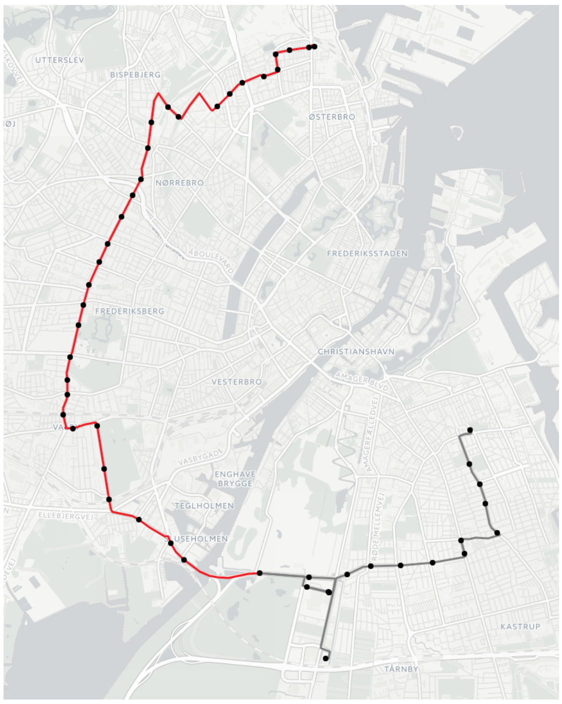
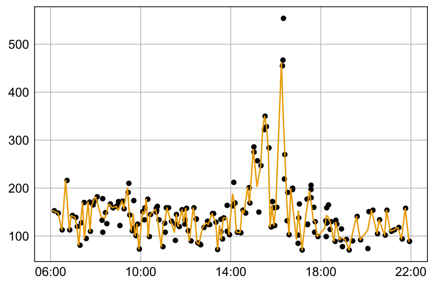


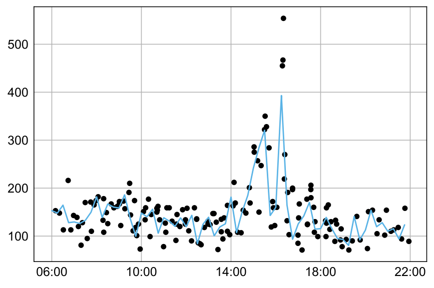
Figure 6: Geography of the 4A bus line in Copenhagen.

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 line splits in different destination patterns (gray), therefore only the first 32 links are considered in this experiment (red).

5.1. Time resolution

To allow predictions for fixed time-steps ahead, data is aggregated at a fixed time resolution. Figure 7 shows examples of travel time for a single link over a single day at various time resolutions. The black dots are actual measurements, and the lines the


(a) 5 min


(b) 15 min

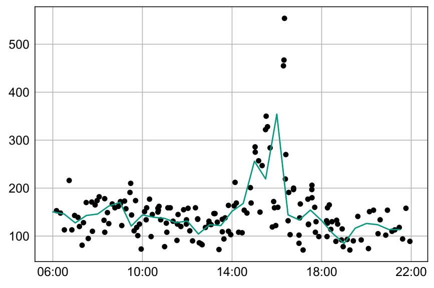

(c) 30 min

Figure 7: Examples of travel time for a single link over a single day, at various time resolutions.

aggregated mean 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 reasonable number of measurements of each time-step to avoid over-fitting.

For this experiment data was aggregated into 15-minute time-steps and normalized cf. Section 4. This resolution was chosen based on a measured mean *headway* of 7.5 minutes between 06:00 and 22:00, i.e. the time between two vehicles during daytime.

Given the time resolution, we set the fixed windows size, $w = 32$, equivalent of 8 hours, to allow pattern in the morning peek affect patterns in the afternoon peek. We set $k = 3$ to allow predictions of up to 45 minutes into the future.

5.2. Evaluation

In order to evaluate the proposed model and comparisons, the following measures are used: *mean absolute error* (MAE), *root mean square error* (RMSE), and *mean absolute percentage error* (MAPE) cf. eqs. (4) to (6), where $\mathbf{Y}_i$ is the true link travel times for sample i and $\widehat{\mathbf{Y}}_i$ is the predicted travel times. Since the multi-output, multi-time-step model predicts link travel times for all u links for the next k time steps, $\mathbf{Y}_i$ and $\widehat{\mathbf{Y}}_i$ have both the dimensionality $w \times u \times 1$.

$$\text{MAE}(\mathbf{Y}, \widehat{\mathbf{Y}}) = \frac{\sum_{i=1}^{N} \left| \mathbf{Y}_i - \widehat{\mathbf{Y}}_i \right|}{N} \tag{4}$$

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

$$\text{MAPE}(\mathbf{Y}, \widehat{\mathbf{Y}}) = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{\mathbf{Y}_i - \widehat{\mathbf{Y}}_i}{\mathbf{Y}_i} \right| \tag{6}$$

To allow a clear comparison, we reduce $\mathbf{Y}_i$ and $\widehat{\mathbf{Y}}_i$ by summing over all links cf. eq. (7). This is equivalent of predicting the total travel time of all 32 links, and follows the initial approach for arrival/departure time prediction by accumulating link travel times.

$$\mathbf{Y}'_i = \sum_{ln=1}^{u} \mathbf{Y}_{i,ln} \qquad \widehat{\mathbf{Y}}'_i = \sum_{ln=1}^{u} \widehat{\mathbf{Y}}_{i,ln} \tag{7}$$

The output of each of the evaluation functions is thus simply vector of size k, i.e. the evaluation of the different time steps for all links accumulated.

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. We use 23 weeks of data for training, and one week of data for testing.

6. Results and discussion

The performance of our proposed model, based on *ConvLSTM*, for link travel time prediction is compared against several other baseline models and services using the presented evaluation:

1. a naïve historical average model, i.e. equivalent of just predicting the normalized value, $\bar{x}_{ln,dow,tod}$
2. the current traffic prediction model currently deployed by Movia.
3. a pure LSTM-model for link travel time prediction, i.e. without applying convolutional filters in state transitions.
4. travel time predictions from Google Traffic (part of Google Maps).

Table 2 shows the overall performance of the proposed and the baseline models. Predictions are limited to daytime, i.e. between 06:00 and 22:00 and accumulated downstream on a journey level to simulate

Model	Time ahead	RMSE (min)	MAE (min)	MAPE (%)
Historical average		4.35	3.23	6.51 %
Current model	t + 1 (15 min)	4.92	3.90	8.05 %
	t + 2 (30 min)	4.91	3.46	6.82 %
	t + 3 (45 min)	5.47	4.15	8.68 %
Pure LSTM	t + 1 (15 min)	3.48	2.48	5.02 %
	t + 2 (30 min)	3.56	2.51	5.08 %
	t + 3 (45 min)	3.68	2.62	5.34 %
Google Traffic	t + 1 (15 min)	3.67	2.96	6.32 %
ConvLSTM	t + 1 (15 min)	2.66	1.99	4.19 %
	t + 2 (30 min)	2.89	2.11	4.44 %
	t + 3 (45 min)	3.11	2.27	4.75 %

Table 2: Results of the proposed and the baseline models

the use for real-time bus arrival/departure time prediction cf eq. (7).

Before going into a direct comparison, it is important to understand some aspects of the baseline models, and how measurements were collected.

6.1. Historical average

The performance of the historical average is independent with respect to the number of time steps ahead it predicts, as it just represents a weekly cycle of mean link travel times.

6.2. Current model

Measurements for the currently deployed bus prediction model were collected at a 5-minute frequency using a non-publicly accessible endpoint at the transport authority. The model is based on an historical average model, but has rule based mechanism on top, which can override or adjust the historical link travel times. For instance it will assume that a delayed vehicle can recover (partially or completely) from its delay by traversing links faster. Of course such an assumption can be problematic in an urban area with lots of external traffic effects.

6.3. Pure LSTM

The pure LSTM-model for link travel time prediction is similar to the model proposed by Yanjie Duan et al. [19]. The model was trained on the exact same dataset as the *ConvLSTM* model, and has a similar architecture, but without the convolutional filters.

6.4. Google Traffic

Measurements from the Google Traffic model were collected using the Google Maps Distance Matrix API [28]. Google uses crowd-sourced road congestion data collected from smart-phones with the *Google Maps App* installed [29]. While the exact model powering the service is not publicly described in detail, the documentation states that "that the returned *duration in traffic* should be the best estimate of travel time given what is known about both historical traffic conditions and live traffic". And further, "that live traffic becomes more important the closer the departure time is to now" [28].

Because there is a limit on the number of requests that one freely can make to the API over a 24-hour period, it has only been possible to collect link travel times for the $t + 1$ time step (i.e. next 15 minutes). Travel times for each link were collected at a 15-minute interval between 06:00 and 22:00.

Another important aspect is that the Google Traffic model is primary designed for estimating car travel times, and can thus be biased away from the bus travel times used in this experiment. Since we only consider link travel time, and collection data for each link individually, the bus dwell time will not be an issue, as it is not included in either measurement.

6.5. Comparison

We compare the performance of the proposed *ConvLSTM* model for bus link travel time prediction against the presented baseline models. The overall results from Table 2 shows that the *ConvLSTM* model outperforms the other methods. The *current model* performs the worst, even compared to the *historical*

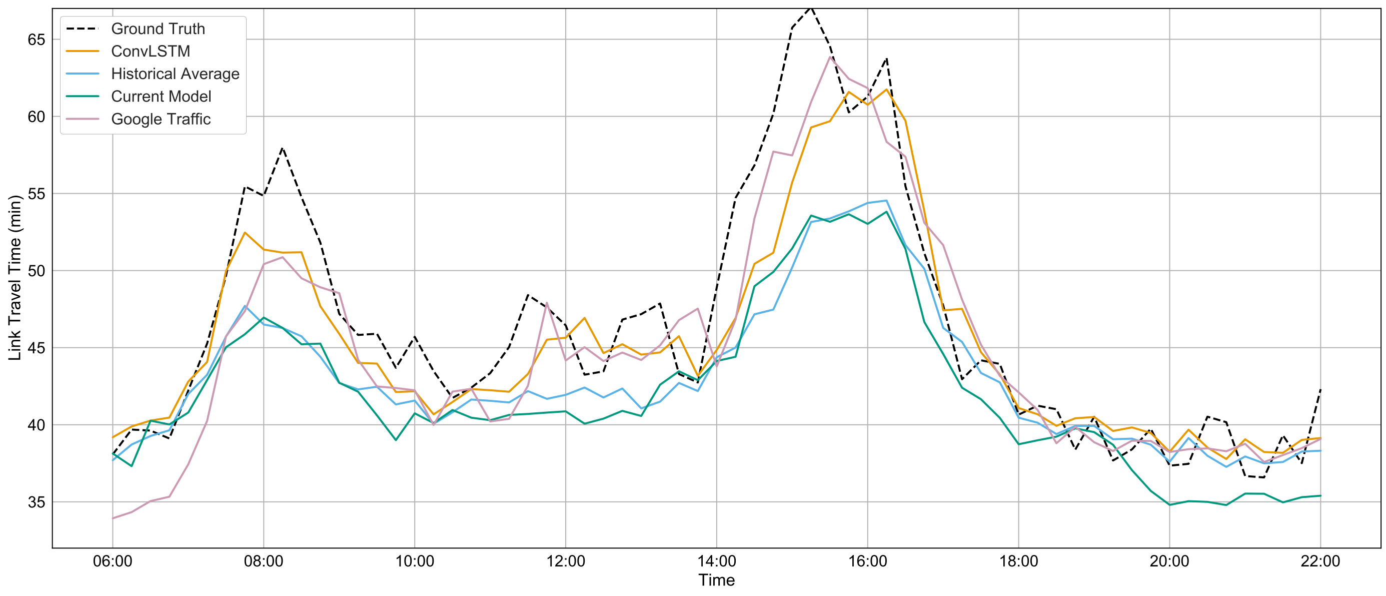


Figure 8

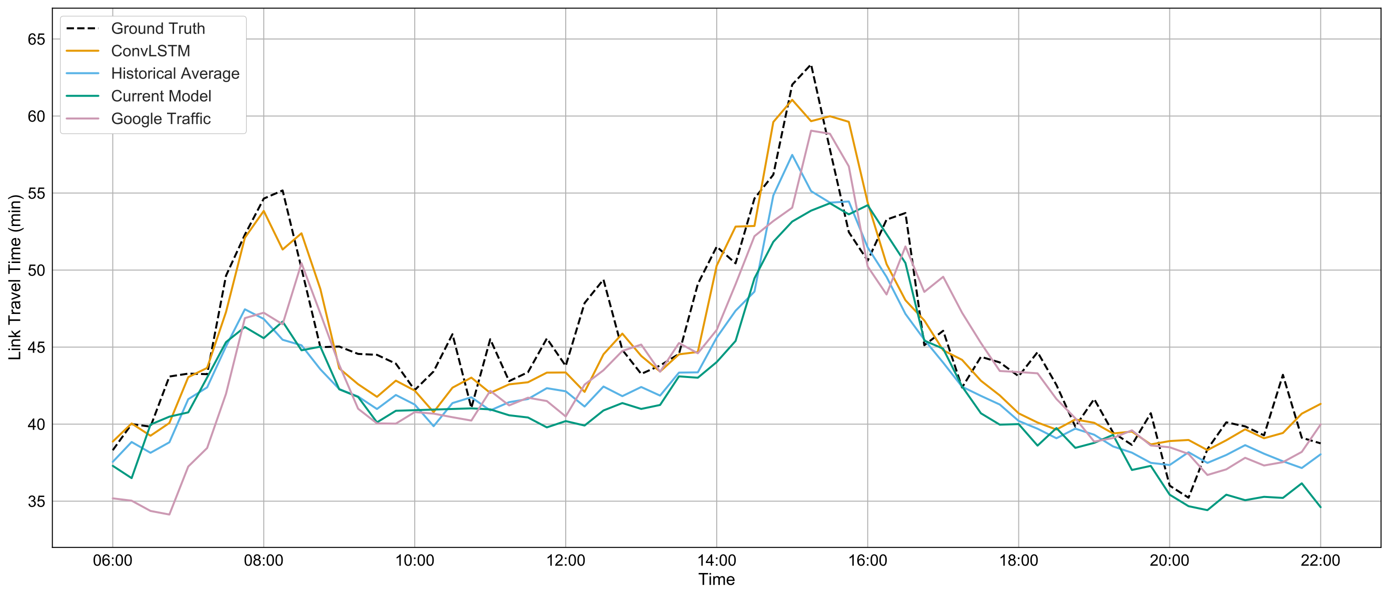


Figure 9

average model, on which it is based. This is most likely due to the rule-based enforcement of recovery from delays, even though the data weights against a recovery.

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 experience very irregular travel times. To investigate this, we focus our analysis on periods when the transport system is most vulnerable, and even small changes in regularity can propagate, since recovery is not an option, i.e. morning and afternoon peeks.

Model	RMSE	MAE	MAPE
Historical Average	6.40	5.57	10.62 %
Current Model	6.69	5.88	11.22 %
Pure LSTM	0.00	0.00	0.00 %
Google Traffic	5.25	4.62	9.17 %
ConvLSTM	2.64	2.09	4.04 %

Table 3: Results: Morning peek (7–9)

Model	RMSE	MAE	MAPE
Historical Average	5.90	4.65	8.28 %
Current Model	6.28	5.20	9.37 %
Pure LSTM	0.00	0.00	0.00 %
Google Traffic	4.16	3.34	6.21 %
ConvLSTM	3.79	3.02	5.61 %

Table 4: Results: Afternoon peek (14–18)

Table 3 and Table 4 shows the evaluation results for *morning peeks (weekdays, 7–9)* and *afternoon peeks (weekdays, 14–18)* respectively for the time-step $t+1$.

The peek hour evaluation shows that the *ConvLSTM* model increases its performance over the baseline models when the transport network is put under stress. In the morning peek, the *ConvLSTM* model performs similar as the overall, whereas the baseline models all experience decreased performance of up to several minutes in both the RMSE and MAE, and an increased MAPE of roughly one third.

Likewise the afternoon peek evaluation shows improvements with respect to the baseline models, even though the *ConvLSTM* model also decreases its performance compared to the overall results. The outperformance is neither as significant as in the morning peek, and as the *Google Traffic* model also performs quite well leaving the gap less than a minute.

The get a little more detailed view of how the different models perform on the micro-level (i.e. the specific journey) we can inspect a single day of predictions. A random weekday from the test-dataset is plotted in Figure 8, which shows the accumulated travel time of all 32 links and the predicted travel time at time-step $t + 1$ both for the proposed and baseline models.

In this particular day (a Thursday) the peek hour traffic was worse than normal, which leads both the *historic averaging* model and the *current model* to both underestimate travel time in the peek periods. Remember that the *current model* is based on the *historic averaging* model, so it is not unexpected that they perform very similar. There is also an small peek in travel time in the midday hours, which none of these historic averaging models are able to predict.

On the other hand, both the *Google Traffic* model and the proposed *ConvLSTM* model get much closer to the ground truth in the peek ours. The *Google Traffic* model seems to predict more accurate then *ConvLSTM* model in the afternoon peek, whereas the opposite occurs in the morning peek. Even more interesting is the detection of the irregular peek in the midday hours, which both the *Google Traffic* model and the proposed *ConvLSTM* model are able to adjust to, at least to some degree.

Figure 9 shows another example (a Friday). Here the difference between the proposed model and the historical averaging and current model baselines are a bit weaker, simply because the the day to a larger degree follows the average pattern for a Friday (especially around the afternoon peek). Even though, the proposed model still performs the best, and it also supports the claim, that the proposed model is strongest when the traffic pattern deviates from the normal pattern, i.e. when the transport network is under stress.

7. Conclusion

This paper has proposed a multi-output, multi-time-step deep neural network for bus travel time prediction using Convolutional and Long short-term memory (LSTM) layers. Results show, that the proposed model network outperforms other popular and recent methods. This includes Google's Traffic model based on crowd-sourced live data, and the current model deployed by Movia, the public transport authority in the Greater Copenhagen Area.

For the prediction accuracy to be increased further it is proposed to apply ensemble/multi-model approaches. In this case the proposed 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.

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] B. Yu, Z.-Z. Yang, K. Chen, B. Yu, Hybrid model for prediction of bus arrival times at next station, Journal of Advanced Transportation 44 (3) (2010) 193–204, ISSN 01976729, doi:10.1002/atr.136, URL `http://onlinelibrary.wiley.com/doi/10.1002/atr.144/fullhttp://doi.wiley.com/10.1002/atr.136`.

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

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

[28] Google Developers, Google Maps Distance Matrix API, URL `https://developers.google.com/maps/documentation/distance-matrix/`, 2017.

[29] D. Barth, The bright side of sitting in traffic: Crowdsourcing road congestion data, URL `https://googleblog.blogspot.dk/2009/08/bright-side-of-sitting-in-traffic.html`, 2009.