Multi-output bus travel time prediction with convolutional LSTM neural network

Niklas Christoffer Petersen [a,b,*], Filipe Rodrigues [b], Francisco Camara Pereira [b]

[a] *Public Transport Movia, Gammel Køge Landevej 3, Valby 2500, Denmark*
[b] *Department of Management Engineering, Technical University of Denmark, Kongens Lyngby 2800, Denmark*

ARTICLE INFO

ABSTRACT

Accurate and reliable travel time predictions in public transport networks are essential for delivering an attractive service that is able to compete with other modes of transport in urban areas. The traditional application of this information, where arrival and departure predictions are displayed on digital boards, is highly visible in the city landscape of most modern metropolises. More recently, the same information has become critical as input for smart-phone trip planners in order to alert passengers about unreachable connections, alternative route choices and prolonged travel times. More sophisticated *Intelligent Transport Systems* (ITS) include the predictions of connection assurance, i.e. an expert system that will decide to hold services to enable passenger exchange, in case one of the services is delayed up to a certain level. In order to operate such systems, and to ensure the confidence of passengers in the systems, the information provided must be accurate and reliable. Traditional methods have trouble with this as congestion, and thus travel time variability, increases in cities, consequently making travel time predictions in urban areas a non-trivial task. This paper presents a system for bus travel time prediction that leverages the non-static spatio-temporal correlations present in urban bus networks, allowing the discovery of complex patterns not captured by traditional methods. The underlying model is a multi-output, multi-time-step, deep neural network that uses a combination of 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, including the currently deployed model at Movia, the regional public transport authority in Greater Copenhagen. We find that the proposed model significantly outperforms all the other methods we compare with, and is able to detect small irregular peaks in bus travel times very quickly.

© 2018 Published by Elsevier Ltd.

1. Introduction

One of the most visible applications of *Intelligent Transport Systems* (ITS), within the field of public transportation, is the display of real-time traffic information. This has happened, traditionally, in the form of arrival and departure times on digital departure boards at stops and stations, and more recently in smart-phone apps and in-vehicle infotainment screens. It is widely deployed in most major cities, and is now considered a standard method to deliver an attractive and competitive public transport service. To an increasing extent, real-time information channels constitute the only source of passenger information.

Public transport authorities have long found that GPS trajectory data from already deployed *Automatic Vehicle Location* systems (AVL) can be used in the production of arrival and departures times (Schweiger, 2003).

Our motivation is to improve the accuracy yielded by current prediction methods by exploiting spatio-temporal correlations present in public transport networks. Our focus is especially on urban bus networks that often share considerable parts of the infrastructure with other modes of transport, and therefore are prone to ripple effects. The proposed system, and the information it produces, can be integrated into ITSs in various ways with different applications. In its most basic form, the system can simply substitute current methods as a data source in passenger information systems, presenting real-time arrival and departure times. Passengers presented with reliable travel times can make use of this information in their decision-making (Cats, Koutsopoulos, Burghout, & Toledo, 2011), e.g. choose alternative routes or modes of trans-

---

* Corresponding author at: Department of Management Engineering, Technical University of Denmark, 2800 Kongens Lyngby, Denmark.
   E-mail addresses: niklch@dtu.dk (N.C. Petersen), rodr@dtu.dk (F. Rodrigues), camara@dtu.dk (F.C. Pereira).
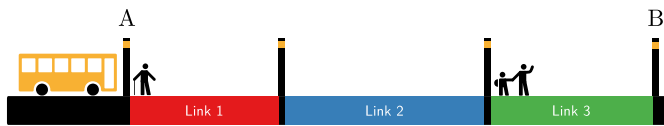
Fig. 1. Arrival- and departure using link travel time.

port to avoid prolonged travel time on their current route. The availability of the information produced by the system to awaiting passengers can also simply function as a comforting assurance, as studies have shown that reliable real-time information at bus stops has a statistically significant dampening effect on the perceived waiting time (Fan, Guthrie, & Levinson, 2016).

A more *intelligent* use of the information can be in the context of automated trip planners. These already accept this kind of real-time information, e.g. using the General Transit Feed Specification (GTFS). This allows for alerting passengers or proposing alternative routes earlier on the passenger's trip when a connecting service might be unreachable due to prolonged travel times.

Operating more sophisticated ITS applications successfully requires, to an even larger extent, accurate and reliable travel time predictions, since the cost of making erroneous decisions based on the predictions increases. Lo and Chang (2012) present a decision support system for bus holding that requires accurate estimated arrival times to function optimally. Other advanced ITS examples include *connection assurance* between two low frequency public transport services, where an expert system advises the driver or the traffic management system that one of the services should wait for the other, based on the arrival time predictions for both services. If the travel time predictions are too optimistic, the expert system ends up advising to hold the connecting service for longer than anticipated, introducing a prolonged delay for the other service and the passengers already present on that service. The use of our proposed system in this context can be achieved with a simple rule-based decision engine on top of the travel time model presented in detail in this paper.

1.1. Bus travel time prediction

Arrival/departure time prediction is commonly approached as a specialization of travel time prediction as illustrated in Fig. 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. Thus, in the example, to predict when the bus at stop A will arrive at stop B, we would just sum up our predicted link travel times for Links 1 to 3. Besides the link travel time, estimations of dwell time (i.e. when a bus is holding at a stop point) should also be accumulated downstream.

Producing precise bus travel time predictions in areas with little external influence, e.g. rural areas, can to a large extent be solved with historical averaging or simple regression methods (Altinkaya & Zontul, 2013; Williams & Hoel, 2003). The problem becomes much more complex in urban areas where congestion, special events, roadworks, weather, etc. highly influence the traffic flow and passenger demand. As on-board GPS and AVL systems have 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 for each vehicle per time-unit.

Using geofencing techniques the raw GPS trajectory data can be converted into arrivals and departures at stop points, and subsequently, travel times on the links between the stop points. The objective is an intelligent expert system that utilizes this data in order to produce precise short-term predictions (e.g. up to 0–1.5 h in the future) for link travel time, specifically for bus traffic in urban areas.

Our contribution is an intelligent model for bus travel time prediction that takes advantage of the non-static spatio-temporal correlations present in urban bus traffic. We leverage on recent state-of-the-art techniques from *machine learning* by combining *convolutional* and *long short-term memory* (LSTM) (Gers, Schmidhuber, & Cummins, 2000; Hochreiter & Schmidhuber, 1997) neural networks, thus allowing the discovery of patterns across both time and space. Our proposed model is also multidimensional in its output with respect to both spatial and temporal aspects, i.e. we predict travel times for all links, for multiple time-steps ahead.

The proposed method is empirically evaluated and compared to other popular approaches for link travel time prediction, including the model currently deployed in production by the public transport authority for the Greater Copenhagen Area, *Movia*. Furthermore, the method is compared to Google Traffic (part of Google Maps), a popular online service for travel time prediction.

This paper is structured in the following manner: In the next section, related work and relevant 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 detail, including e.g. network topology, and data preparation. 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. Related work

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 the problem has also unique constraints and opportunities that follow from servicing a fixed route with fixed stop points. The improvement in computational power in recent decades has gradually allowed more complex link travel time models with increased precision.

Early approaches for bus travel time prediction rely on historical average models (Dailey & Wall, 1999; Sun et al., 2007), and linear regression (Patnaik, Chien, & Bladikas, 2004). Recent research presents this type of models only for comparison purposes, and in all cases, these are outperformed by the proposed alternatives (Jeong & Rilett, 2005; Shalaby & Farhan, 2004). The major disadvantage of historical average models is that they will only slowly converge to changes in the travel time, which of course is undesired with short, but highly impacting, external influences (e.g. a traffic incident or a large event). However, their simplicity, both with respect to computational cost and need of input data, has made them widely used in the industry. In rural areas, where traffic patterns are quite static, they can actually perform reasonably.

By their capabilities of maintaining state between predictions, Kalman filters (KF) have been the topic of several studies either as an independent model (Chen & Chien, 2001; Shalaby & Farhan, 2004), or in combination with other models (Bai, Peng, Lu, & Sun, 2015; Yu, Yang, Chen, & Yu, 2010). In all cases, the applied filters are traditional linear KFs, and applied independently to each link. Because of the linearity, these models are computationally still quite cheap, but likewise, their disadvantage is that they are very limited in capturing and forecasting the complex non-linear dynamics of travel and dwell time in a metropolitan bus system. For example, the KF's state is only directly accessible for the leading time-step and thus is not capable of finding long-distance patterns spanning over several links and/or over several time-steps. In order to overcome this, KFs can be generalized to *extended Kalman filters* (EKFs), allowing nonlinearities, but they still do not consider

multiple links simultaneously. Making EKFs output travel times for all links simultaneously, with possible nonlinear interactions between them, would dramatically increase the computational cost.

The above analysis is substantiated by Lin, Yang, Zou, and Jia (2013) and Kumar, Kumar, Vanajakshi, and Subramanian (2014) who find *artificial neural networks* (ANN) to outperform independent Kalman filter models. However, the computational challenges of fully connected ANNs are also limiting the number of neurons of the network, and thus the complexity of the patterns it can learn to recognize. This has sparked the interest in studying *composite* or *hybrid* models. Bai et al. (2015) use a two-stage approach by combining an offline ANN model with an adaptable/online Kalman filter to yield a dynamic model. The advantage is the balance between computational complexity and the ability to adapt to smaller deviations quickly. The model is able to adapt to temporal variations in the current travel time on a journey, but it is still not able to recognize long distance patterns. The model proposed in this work uses *long short-term memory* cells (LSTM), a special form of recurrent neural network (RNN) cells. Ma, Tao, Wang, Yu, and Wang (2015) use LSTM cells for highway speed prediction, and find it to significantly outperform KFs. Our proposal differs from existing research in bus link travel time prediction by combining the capability for maintaining state-space over multiple time-steps, while allowing the deep neural network to be efficiently trained.

Some recent research recognizes that several routes can benefit from each other's predictions if they share some partial route segment, e.g. (Bai et al., 2015; Gal, Mandelbaum, Schnitzler, Senderovich, & Weidlich, 2017; Yu, Lam, & Tam, 2011). However, none of these approaches consider cross-temporal correlations between different route segments, and they only use a small window for correlation with upstream links (e.g. max. 3 links). Likewise, Duan, Lv, and Wang (2016) propose the use of an LSTM model for general highway travel time prediction, and to predict multiple time-steps ahead, but only for a single link at a time, i.e. cross link (spatial) correlations are lost. Another non-public transport study estimates travel times on road segments (Tang, Chen, & Khattak, 2018), and actually incorporates the spatial correlation, but the temporal aspect is very coarse and does not predict multiple time-steps ahead. 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 orders of magnitude compared to fully connected ANNs capable of capturing similar complex patterns.

We can identify the following strengths of the proposed system compared to existing approaches:

  i) Unlike previous contributions in bus arrival prediction, it has the ability to learn spatio-temporal correlations as a coherent structure. The learned patterns can generalize over time and network links since the *convolutional* filters are shared.
  ii) We predict multiple time-steps ahead using a recurrent structure and an encoder-decoder architecture that allows the time-steps ahead to follow more complex patterns compared to existing approaches that just use a fully connected ANN layer as the final layer to split the prediction into multiple output time-steps.
  iii) The input data needed for the method is easily obtained from the raw GPS traces that the AVL systems output, given the relatively fixed road network and location of bus stop points and stations.

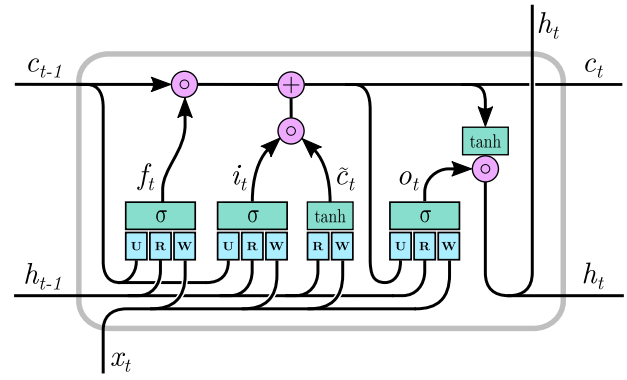In contrast, the following possible weakness should also be considered:

Fig. 2. Structure of LSTM cell with peephole.

  i) The computational complexity of the training is still a concern. Even though the computational complexity is reduced greatly with *convolutional* filters compared to pure ANN models, it is still time-consuming to train the proposed model. That said, we have successfully trained models for complete routes using commodity-grade hardware within reasonable time. With our test setup we could do retraining on a daily basis without computational complications. The training can easily be distributed across multiple computational instances, so we argue that the scalability issue can be overcome.

3. Convolutional LSTM neural networks

A long short-term memory (LSTM) neural network is a special type of Recurrent Neural Network (RNN) which has been proven robust for capturing long-term dependencies (Gers et al., 2000; Hochreiter & Schmidhuber, 1997). The important feature of an LSTM network is its capability to maintain a cell state, $\mathbf{c}_t$, from previous observations across sequences of input (e.g. time), but also to eliminate information considered irrelevant. To allow this mechanism, the maintenance of information is controlled by three gates: *input gate, forget gate*, and *output gate*. Each gate yields a state variable 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 element-wise product.

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

Fig. 2 illustrates the inner structure of an LSTM cell with peephole as proposed by Gers and Schmidhuber (2000). It has especially grown popular for predicting time series using methods evolved from Gers, Eck, and Schmidhuber (2001), where fixed-length windows of time-series are generated and feed into an LSTM network. Multiple LSTMs can be stacked such that more complex patterns of sequential information (e.g. temporal patterns) can be learned.

Convolutional neural networks (CNNs), on the other hand, have been widely used for capturing spatial relationships, e.g. the 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 values x_j, convolutional units are only locally connected and reuse the same weights to produce 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
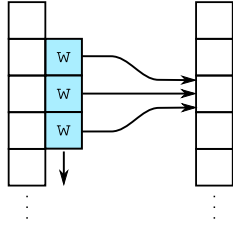
Fig. 3. Application of convolutional filter onto 1D data.

layer. Fig. 3 illustrates a single convolutional filter of size 3 being applied to one-dimensional data.

Special care needs to be taken at the boundaries, i.e. where the convolutional filter will exceed the input. To avoid that the size of the output decreases, an approach 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 also that learned patterns can be transferred 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. (2015) 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.

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

As with traditional CNN layers, the output dimensionality of a *ConvLSTM* layer is 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 the application of convolutional filters to the LSTM model greatly reduces the number of parameters/weights that need to be learned, compared to a *pure LSTM* approach. This allows for even deeper networks.

4. Multi-output model

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

4.1. Network topology

Fig. 4 shows the overall network topology, where blue boxes illustrate *input-to-state* convolutions and yellow boxes *state-to-state* convolutions. The network uses a sequence encoder/decoder technique, which is an extension of the encoder/decoder presented by Shi et al. (2015). The encoder block consists of two *ConvLSTM* layers, where the resultant sequence (last k values of the sequence) is fed into a decoder, or prediction block. The decoder block also consists of two *ConvLSTM* layers, and a fully connected (FC) layer. The proposed architecture allows unequal w and k, e.g. it predicts the next 3 time-steps based on a window size of 20 previous time-steps.

Therefore, 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. Lastly, each *ConvLSTM* layer has 64 outputs, yielding a total need of 512 convolutional filters.

In order to avoid over-fitting during training *Dropout* (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) is used between the *ConvLSTM* layers, and *Batch Normalization* (Ioffe & Szegedy, 2015) is also performed before each *ConvLSTM* layer to ensure reasonable inputs for the activations and speed-up learning. The dropout probability is adjusted to 20%, 10%, and 10%, respectively.

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.
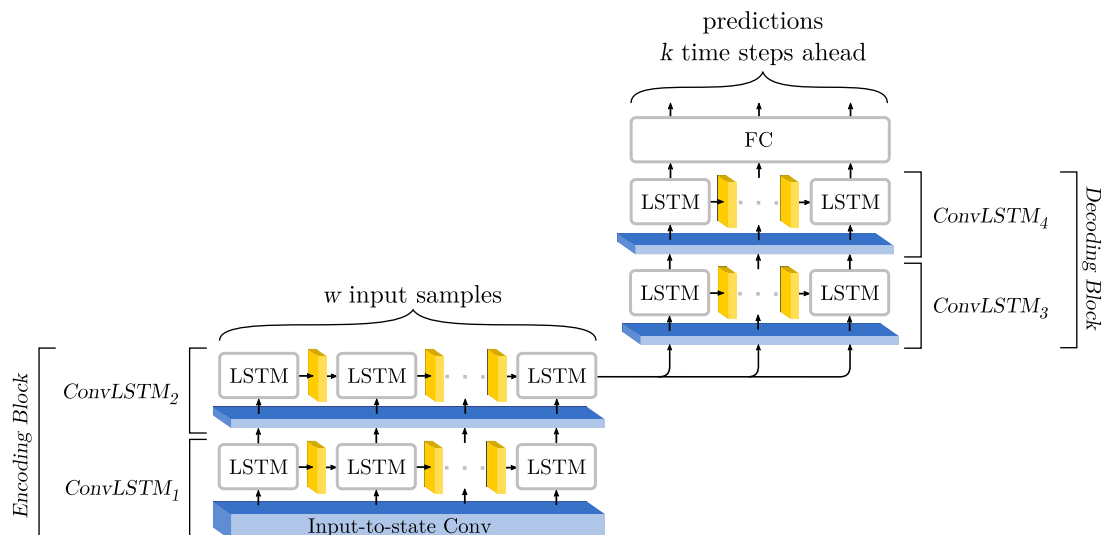

Fig. 4. Convolutional LSTM network topology. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Example of raw travel time measurements.

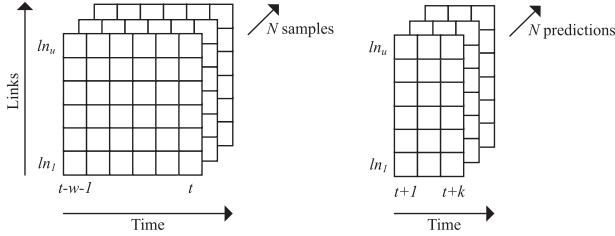| Timestamp	Linkref.	Link travel time (s)
2017-10-10 00:20:02	29848:1254	63
2017-10-10 00:21:07	1254:1255	65
2017-10-10 00:21:51	1255:10115	44
⋮	⋮	⋮

Fig. 5. Shapes of the input and output data.

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. This output is standard for most AVL systems used in public transport systems, thus allowing the proposed system to generalize to other regions.

For the *ConvLSTM* model to be able to capture the desired spatio–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 $1,\ldots,u$, as illustrated in Fig. 5.

As for the output, it consists of N predictions for each of the k time-steps ahead, $t+1,\ldots,t+k$. Thus the input is a 4D-tensor, **X** with dimensionality $N \times w \times u \times 1$, and the output, **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-ahead 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 an adequate resolution.

4.3. Detrending

Urban bus travel times vary throughout the day, and the day of the week due to *recurring congestion*. In order to reduce the need for the deep neural network to learn this recurring variation, the travel times for link $ln \in \{1,\ldots,u\}$, at time-step t, $x_{ln,\,t}$, are 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}:

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

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 are obviously 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 deviations are highly sensitive to such measurements. A sug-
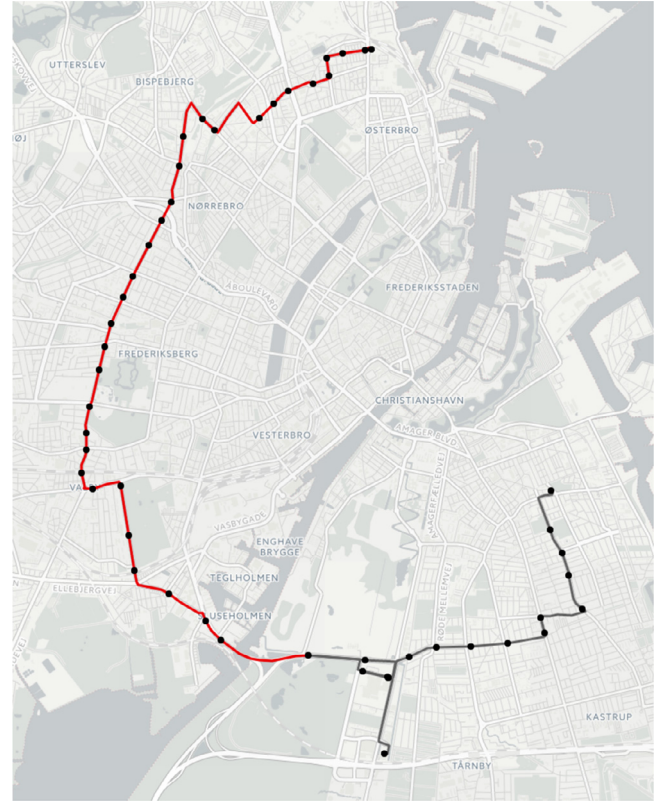

Fig. 6. Geography of the 4A bus line in Copenhagen. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

gested method is to apply *absolute deviation around the median* (MAD; see Olewuezi, 2011) 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 (Chollet & Others, 2015), and trained using the *RMSprop* algorithm (Hinton & Tieleman, 2017). The source code for the proposed method is publicly available at GitHub: Petersen, Rodrigues, and Pereira (2017).

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 1,2M 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.

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

5.1. Time resolution

In order to allow predictions for fixed time-steps ahead, the data is aggregated at a fixed time resolution. The choice of time
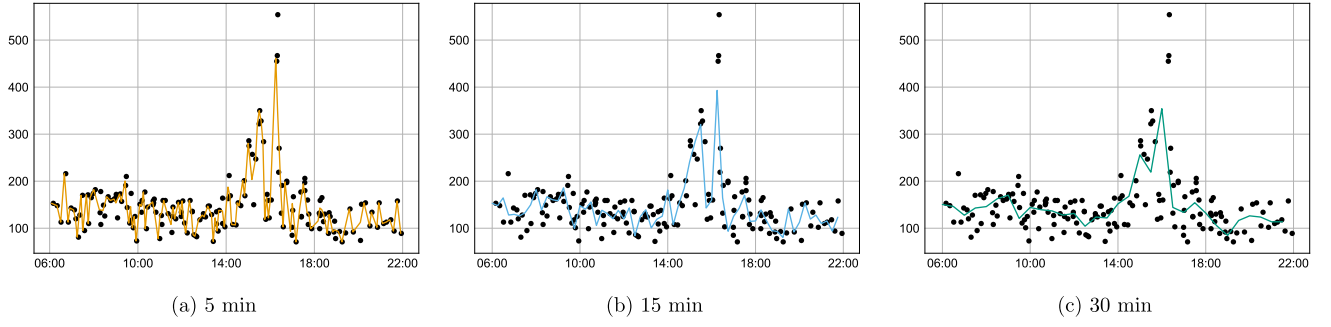
Fig. 7. Examples of travel time for a single link over a single day, at various time resolutions.
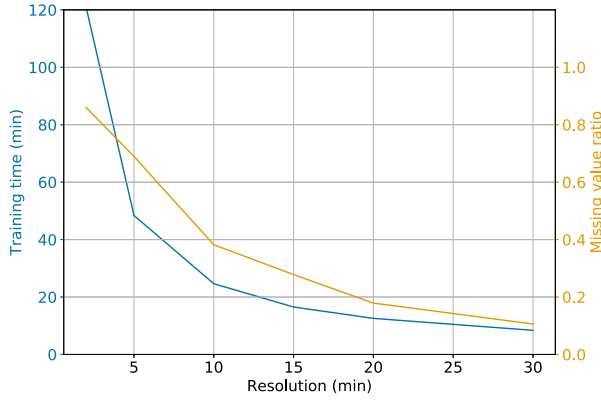

Fig. 8. Choice of resolution influence on training time and missing values. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

resolution is a hyper-parameter for the proposed system, and should be tuned for the specific dataset. Fig. 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 aggregated mean link travel time at the given resolution. Several considerations should be made when choosing the time resolution:

- The *expected* frequency of the line, since a choice far from this will lead to either 1) sparse measurements, and low probability of actually using a prediction, because no service runs in the predicted time step; or 2) an overly smooth time-series, with too much detail about variability being lost. Thus it is a balance between capturing the details and still having a reasonable number of measurements of each time-step to avoid overfitting.
- The computational cost of training the system, since smaller time-steps will require further iterations over the training data and larger values of w and k to include the same lagging time window, and time horizon for predictions.

Fig. 8 shows how the choice of resolution influences the training time of our proposed deep neural network architecture on commodity hardware (blue). It also shows how the portion of time-steps with missing values (yellow) also increases as more fine-grained resolutions are considered. For instance, using a 2-min resolution will cause 89% of all time steps to not include any measurements.

For this experiment, the AVL data was aggregated into 15-min intervals and normalized as described in Section 4. This resolution was chosen based on the above-mentioned considerations. The "4A" bus line had a measured mean *headway* (the time between two vehicles during daytime) of 7.5 min between 06:00 and 22:00, and thus there is a reasonably high probability that 15-min time-

steps will include 1–2 measurements. Indeed, the average number of measurements in each time step was 1.7 for the training set.

Given the time resolution, we set the fixed window size, $w = 32$, equivalent to 8 h, to allow patterns in the morning peak to affect patterns in the afternoon peak. We set $k = 3$ to allow predictions of up to 45 min into the future.

5.2. Evaluation

The evaluation of the proposed model and all the considered baselines is based on the following statistics: *mean absolute error* (MAE), *root mean square error* (RMSE), and *mean absolute percentage error* (MAPE), as formalized in Eqs. (4)–(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 the same 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:

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

This is equivalent to predicting the total travel time of all 32 links, and follows the initial approach for arrival/departure time prediction by accumulating link travel times.

The output of each of the evaluation functions is thus simply a 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 in order to simulate real-world conditions, in which real-time travel time measurements arrive as a continuous data stream. We use 23 weeks of data for training, and one week of data for testing. The window is advanced for 1 week at a time for a total of 4 test weeks. The trained models are available, alongside the source code, at Petersen et al. (2017) and include a test dataset (4 weeks). For replicating our results, the full dataset used in this experiment is available from Movia upon request.

Table 2
Results of the proposed and the baseline models.

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 3
Results: Morning peak (7h–9h).

Model	RMSE	MAE	MAPE
Historical Average	6.40	5.57	10.62%
Current Model	6.69	5.88	11.22%
Pure LSTM	3.80	3.16	6.01%
Google Traffic	5.25	4.62	9.17%
ConvLSTM	2.64	2.09	4.04%

Table 4
Results: Afternoon peak (14h–18h).

Model	RMSE	MAE	MAPE
Historical Average	5.90	4.65	8.28%
Current Model	6.28	5.20	9.37%
Pure LSTM	5.26	3.97	7.08%
Google Traffic	4.16	3.34	6.21%
ConvLSTM	3.79	3.02	5.61%

6. Results and discussion

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

1. a naïve historical average model, i.e. equivalent to just predicting the normalized value, $\bar{x}_{ln,dow,tod}$;
2. the 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 model and the baseline methods. Predictions are limited to daytime, i.e. between 06:00 and 22:00, and are accumulated downstream on a journey level to simulate 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 in time it predicts, as it just represents a weekly cycle of mean link travel times.

6.2. Current model

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

6.3. Pure LSTM

The pure LSTM model for link travel time prediction is similar to the model proposed by Duan et al. (2016). 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 (Google Developers, 2017). Google uses crowd-sourced road congestion data collected from smart-phones with the *Google Maps App* installed (Barth, 2009). While the exact model powering the service is not publicly described in detail, the documentation states 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". Furthermore, it states that "live traffic becomes more important the closer the departure time is to now" (Google Developers, 2017).

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

Another important aspect is that the Google Traffic model is primarily designed for estimating car travel times, and therefore it can be biased and not ideal for estimating the bus travel times used in our experiments. Since we only consider link travel times and collect 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 baseline models mentioned above. The overall results from Table 2 show that the *ConvLSTM* model outperforms all the other methods. The *current model* performs the worst, even compared to the *historical average* model, on which it is based on. This is most likely due to the rule-based enforcement put on top of the historical average.

Although the difference in performance might seem small, it should be emphasized that the evaluation measurements are averaging their errors, 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. during morning and afternoon peaks.

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

The peak hour evaluation shows that the *ConvLSTM* model increases its performance over the baseline models when the trans-
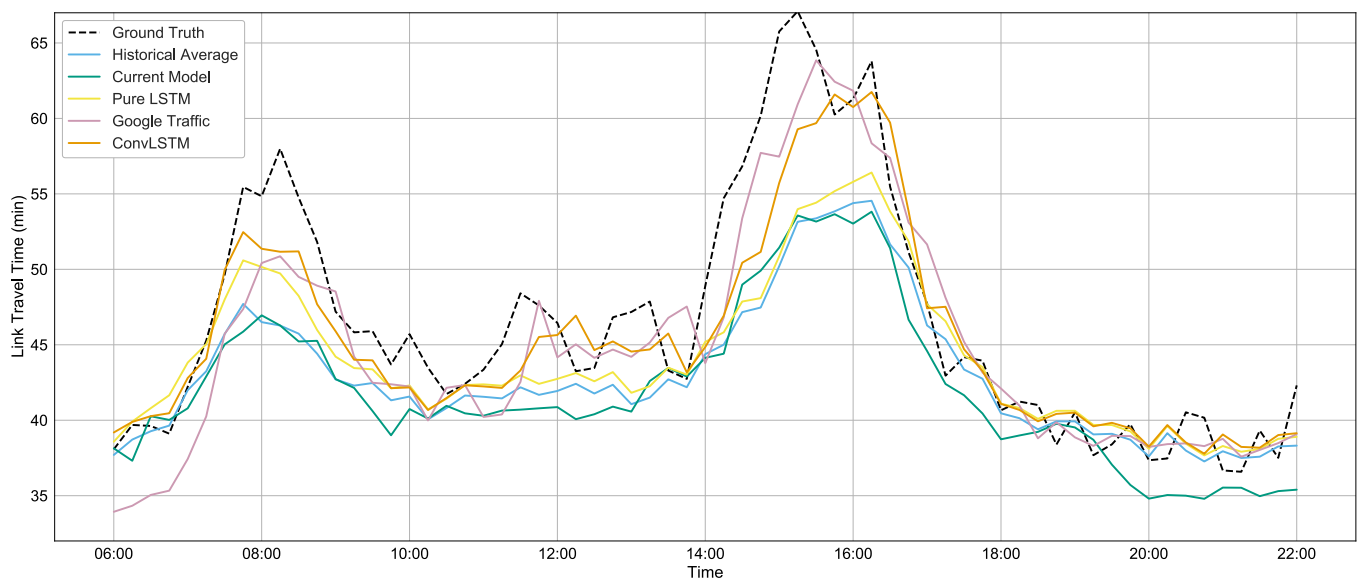
Fig. 9. Accumulated link travel time over a single day (a Thursday) in the test set.

port network is put under stress. In the morning peak, the *ConvLSTM* model does not degrade in performance compared to the overall daytime results, whereas all the baseline models experience a decrease in performance of up to several minutes according to both RMSE and MAE, and an increase in MAPE of roughly one third.

Similarly, the afternoon peak evaluation shows improvements with respect to the baseline models, even though the *ConvLSTM* model also decreases its performance when compared to the overall results. However, in this case, the difference in performance with the baseline methods is not as significant as in the morning peak. We can also observe that the *Google Traffic* model performs rather well in the afternoon peak, which reduces the gap in error to less than a minute to the proposed ConvLSTM-based approach.

To obtain a more detailed view of how the different models perform at 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 Fig. 9 which shows the accumulated travel time of all 32 links and the predicted travel time at time-step $t + 1$, both for the proposed model and the baseline model.

On this particular day (a Thursday), the peak hour traffic was worse than normal, which leads both the *historical average* model and the *current model* to underestimate travel time in the peak periods. Please recall that the *current model* is based on the *historical average* model. Therefore, it is not surprising that they perform similarly. There is also a small peak in travel time in the afternoon, which none of the historical average models is 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 peak hours. The *Google Traffic* model seems to predict more accurately than the *ConvLSTM* model in the afternoon peak, whereas the opposite occurs in the morning peak. However, both models are able to detect the irregular peak in the afternoon and adjust to it, at least to some degree.

Fig. 10 shows another example day - a Friday. Here the difference between the proposed model and the historical average and current model baselines is slightly less significant, simply because the day to a larger degree follows the average pattern for a "normal" Friday (especially around the afternoon peak). Nonetheless, the proposed model still performs the best, and this also supports our 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.

Finally, we compare the computational complexity of training the different models. Obviously, we cannot include metrics for the *Google Traffic*, as the model is not public. Likewise, it is not sensible to compare with the *Current Model*, since it is "trained" on a dataset of different size and on hardware using in production at the transport authority. But, since we know it is essentially an *historical average* approach, we can expect a similar computational complexity. The historical average can be calculated within seconds for the full 23-week training dataset. The training of the *Pure LSTM* and *ConvLSTM* model can be achieved in both cases, for the full 23-week training dataset and the full 32-links, in less than 20 min on commodity hardware (8 cores, 64GB RAM, GTX 1070 GPU). This might indicate why the *historical average* models are still popular in the industrial systems, but we, however, argue that the more complex models are indeed scalable and the improved accuracy desirable, even though it is more computationally expensive.

7. Conclusion

This paper proposed a multi-output, multi-time-step system for bus travel time prediction. The proposed system uses a deep neural network model consisting of convolutional and long short-term memory (LSTM) layers, that is able to capture the non-static spatio-temporal correlations of variability in urban bus travel times. This allows the model to generalize patterns learned in predictions across space and time. Also, our approach for multi-time-step prediction using an encoder/decoder architecture is, to the best of our knowledge, new in the context of bus travel time prediction. The proposed approach allows accurate predictions further into the future compared to traditional approaches where subsequent time-steps are predicted independently. Our empirical results demonstrate that the proposed model outperforms other popular and recent methods from the state-of-the-art. This includes Google's Traffic model based on crowd-sourced live traffic data, and the current model deployed by Movia, the public transport authority in the Greater Copenhagen Area. The increased accuracy when compared to the baseline approaches is even more significant in the peak hours, where the urban bus transport network is under stress. The data required for the proposed system is simply the standard output that most AVL systems used in the pub-
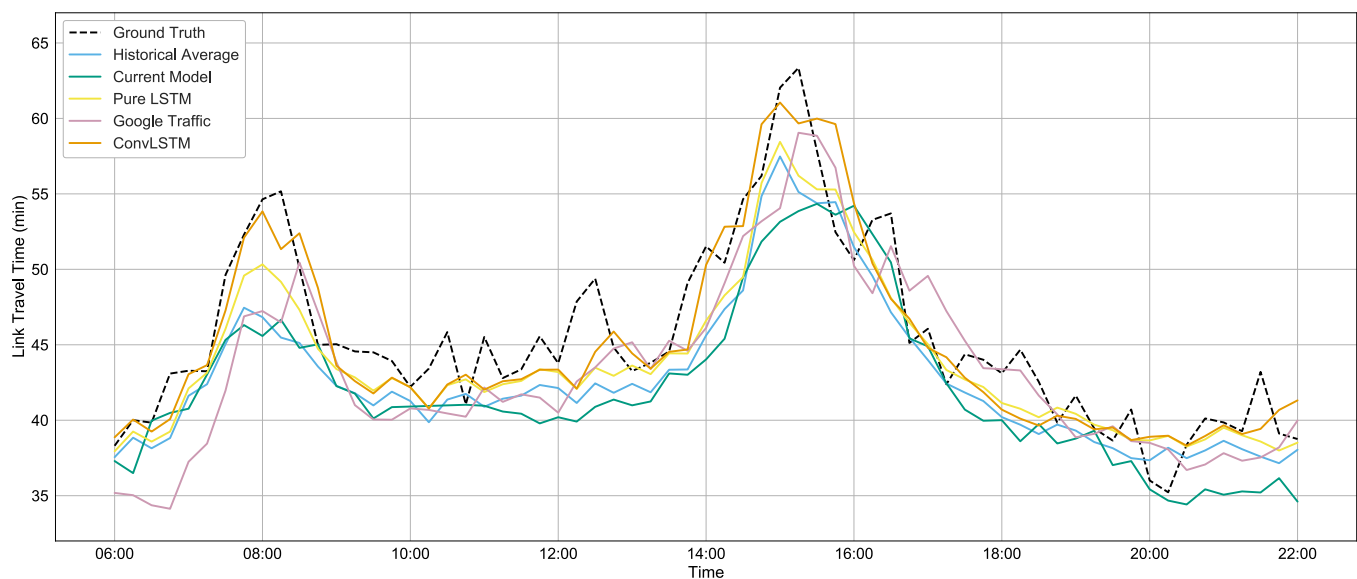
Fig. 10. Accumulated link travel time over a single day (a Friday) in the test set.

lic transport industry produce. We are aware that public transport agencies in Singapore, London, New York, Stockholm, Oslo, and Helsinki all have deployed AVL systems that fulfill this requirement, and thus the proposed system indeed generalizes trivially across different cities in different countries.

Although the proposed method is more computationally expensive than simple *historical average* models, given the state of modern computational hardware, it is indeed scalable to be applied to an urban bus network for independent routes. Even with commodity hardware, we are able to retrain the route used in this experiment in less than 20 min, and we can thus easily retrain the model on a daily basis. Given the results of our proposed model, we are currently actively pursuing deployment of the model in the Greater Copenhagen region, in close collaboration with the transport authority - Movia. We do however consider this route-independent approach a limitation of the current system, and below we provide some research opportunities to extend the proposed system by handling correlations between different routes.

7.1. Future work

As future work, we would like to extend the presented systems in the following directions:

i) The integration of our proposed system to different control strategies for enhancing the regularity and reliability of the bus service, e.g. as suggested by Lo and Chang (2012). This would create a possible feedback loop from the predicted travel times that could possibly affect the travel times on a short-term basis. This is a non-trivial task, since it requires either simulation, which is complex for urban public transport networks in the detail needed here, or integration directly into currently running services, which is organizationally and technically challenging.

ii) In order for the prediction accuracy to be increased further, it would be interesting to include more contextual features in the input data and not only the observed travel times. This could include features from the road network that the link consists of, e.g. whether intersections on the link are regulated by a traffic signal or not. In order to achieve this, map matching of at least the link geometry to the road network is necessary. However, this should be easily overcome,

and many interesting crowd-sourced data are freely available (e.g. Open Street Map). Additional data sources such as weather conditions have shown to impact bus travel time (Chen, Liu, Xia, & Chien, 2004) and could also be included. More rare, but highly impacting deviations, such as traffic incidents, service-outage, holidays and large events, could also be considered in this research direction.

iii) Currently, the model only uses convolutions over a single bus route, i.e. 1D-convolutions. We believe that it would be interesting to see the effect on the accuracy of the system if this was generalized to a network of bus routes. This would require extending the convolutions into a multi-dimensional space. Popular approaches used traditionally in conjunction with convolutions, such as overlaying the geographical map with a 2D-grid, have not shown good results. The challenge seems to be that bus networks are relatively sparse, and that travel times do not aggregate well in cells, e.g. compared to travel demand. Recent state-of-the-art proposes *graph convolutional neural networks* (Li, Yu, Shahabi, & Liu, 2017), i.e. where the convolutions are done over graph structures. We plan to pursue this approach - with the complications and development needed - to adapt the method to bus networks and the bus travel time prediction problem.

iv) A final direction we have identified is to include the proposed system in an ensemble/multi-model approach. In this case, the proposed model can be included and used as a sub-model for the ensemble. The challenge here is the coordination between the different sub-models that can be seen as autonomous agents in an expert and intelligent system context. Especially, how to solve disagreements. Different approaches have been proposed by Weng, Lu, Wang, Megahed, and Martinez (2018), and we expect to explore these approaches in our research.

CRediT authorship contribution statement

Niklas Christoffer Petersen: Conceptualization, Methodology, Software, Writing - original draft, Writing - review & editing. **Filipe Rodrigues:** Conceptualization, Writing - review & editing, Supervision. **Francisco Camara Pereira:** Conceptualization, Writing - review & editing, Supervision.

References

Altinkaya, M., & Zontul, M. (2013). Urban bus arrival time prediction: A review of computational models. *International Journal of Engineering & Technology, 2*(4), 164–169.

Bai, C., Peng, Z. R., Lu, Q. C., & Sun, J. (2015). Dynamic bus travel time prediction models on road with multiple bus routes. *Computational Intelligence and Neuroscience, 2015*. doi:10.1155/2015/432389.

Barth, D. (2009). The bright side of sitting in traffic: Crowdsourcing road congestion data. Retrieved 2017-11-06 from https://googleblog.blogspot.dk/2009/08/bright-side-of-sitting-in-traffic.html.

Cats, O., Koutsopoulos, H., Burghout, W., & Toledo, T. (2011). Effect of real-time transit information on dynamic path choice of passengers. *Transportation Research Record: Journal of the Transportation Research Board, 2217*, 46–54. doi:10.3141/2217-06.

Chen, M., & Chien, S. (2001). Dynamic freeway travel-time prediction with probe vehicle data: Link based versus path based. *Transportation Research Record, 1768*(1), 157–161. doi:10.3141/1768-19.

Chen, M., Liu, X., Xia, J., & Chien, S. I. (2004). A dynamic bus-arrival time prediction model based on APC data. *Computer-Aided Civil and Infrastructure Engineering, 19*(5), 364–376. doi:10.1111/j.1467-8667.2004.00363.x.

Chollet, F. (2015). Keras. https://github.com/fchollet/keras.

Dailey, D. J., & Wall, Z. (1999). An algorithm for predicting the arrival time of mass transit. *Transportation research board 78th annual meeting*. Washington DC.: Transpotation Research Board. doi: 10.1.1.579.2083.

Duan, Y., Lv, Y., & Wang, F.-Y. (2016). Travel time prediction with LSTM neural network. *2016 IEEE 19th international conference on intelligent transportation systems*, 1053–1058. doi:10.1109/ITSC.2016.7795686.

Fan, Y., Guthrie, A., & Levinson, D. (2016). Perception of waiting time at transit stops and stations. *Technical Report 9*. Center for Transportation Studies, University of Minnesota, Minneapolis.

Gal, A., Mandelbaum, A., Schnitzler, F., Senderovich, A., & Weidlich, M. (2017). Traveling time prediction in scheduled transportation with journey segments. *Information Systems, 64*(C), 266–280. doi:10.1016/j.is.2015.12.001.

Gers, F., & Schmidhuber, J. (2000). Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS international joint conference on neural networksIJCNN 2000. Neural computing: New challenges and perspectives for the new millennium* (pp. 189–194 vol.3). IEEE. doi:10.1109/IJCNN.2000.861302.

Gers, F. A., Eck, D., & Schmidhuber, J. (2001). 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): 2130* (pp. 669–676). doi:10.1007/3-540-44668-0_93.

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation, 12*(10), 2451–2471. doi:10.1162/089976600300015015.

Google Developers (2017). Google maps distance matrix API. https://developers.google.com/maps/documentation/distance-matrix/.

Hinton, G., & Tieleman, T. (2017). Lecture - Rmsprop: Divide the gradient by a running average of its recent magnitude. https://www.coursera.org/learn/neural-networks/lecture/YQHki/rmsprop-divide-the-gradient-by-a-running-average-of-its-recent-magnitude.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780. doi:10.1162/neco.1997.9.8.1735.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. Arxiv, (pp. 1–11). doi:10.1007/s13398-014-0173-7.2.

Jeong, R., & Rilett, L. R. (2005). Prediction model of bus arrival time for real-time applications. *Transportation Research Record: Journal of the Transportation Research Board, 1927*(1927), 195–204. doi:10.3141/1927-23.

Kumar, V., Kumar, B. A., Vanajakshi, L., & Subramanian, S. C. (2014). Comparison of model based and machine learning approaches for bus arrival time prediction. *TRB 93rd annual meeting compendium of papers*.

Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2017). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. http://arxiv.org/abs/1707.01926. doi:10.15662/IJAREEIE.2015.0501067.

Lin, Y., Yang, X., Zou, N., & Jia, L. (2013). Real-Time bus arrival time prediction: Case study for Jinan, China. *Journal of Transportation Engineering, 139*(11), 1133–1140. doi:10.1061/(ASCE)TE.1943-5436.0000589.

Lo, S. C., & Chang, W. J. (2012). Design of real-time fuzzy bus holding system for the mass rapid transit transfer system. *Expert Systems With Applications, 39*(2), 1718–1724. doi:10.1016/j.eswa.2011.08.111.

Ma, X., Tao, Z., Wang, Y., Yu, H., & Wang, Y. (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. doi:10.1016/j.trc.2015.03.014.

Olewuezi, N. (2011). Note on the comparison of some outlier labeling techniques. *Journal of Mathematics and Statistics, 7*(4), 353–355. doi:10.3844/jmssp.2011.353.355.

Patnaik, J., Chien, S., & Bladikas, A. (2004). Estimation of bus arrival times using APC data. *Journal of Public Transportation, 7*(July 2017), 1–20. doi:10.5038/2375-0901.7.1.1.

Petersen, N. C., Rodrigues, F., & Pereira, F. C. (2017). Multi-output bus travel time prediction with convolutional LSTM neural networks source code. https://github.com/niklascp/bus-arrival-convlstm.

Schweiger, C. (2003). *TCRP synthesis 48: Real-time bus arrival information systems*.

Shalaby, A., & Farhan, A. (2004). Prediction model of bus arrival and departure times using AVL and APC data. *Journal of Public Transportation, 7*, 41–61. doi: 10.1.1.170.9999.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-k., & Woo, W.-c. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Proceedings of the 28th international conference on neural information processing systems – Volume 1* (pp. 802–810). MIT Press.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research, 15*, 1929–1958. doi:10.1214/12-AOS1000.

Sun, D., Luo, H., Fu, L., Liu, W., Liao, X., & Zhao, M. (2007). Predicting bus arrival time on the basis of global positioning system data. *Transportation Research Record Journal of the Transportation Research Board, 2034*(2034), 62–72. doi:10.3141/2034-08.

Tang, K., Chen, S., & Khattak, A. J. (2018). Personalized travel time estimation for urban road networks: A tensor-based context-aware approach. *Expert Systems With Applications, 103*, 118–132. doi:10.1016/j.eswa.2018.02.033.

Weng, B., Lu, L., Wang, X., Megahed, F. M., & Martinez, W. (2018). Predicting short-term stock prices using ensemble methods and online data sources. *Expert Systems With Applications, 112*, 258–273. doi:10.1016/j.eswa.2018.06.016.

Williams, B. M., & Hoel, L. a. (2003). Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results. *Journal of Transportation Engineering, 129*(6), 664–672. doi:10.1061/(ASCE)0733-947X(2003)129:6(664).

Yu, B., Lam, W. H. K., & Tam, M. L. (2011). Bus arrival time prediction at bus stop with multiple routes. *Transportation Research Part C: Emerging Technologies, 19*(6), 1157–1170. doi:10.1016/j.trc.2011.01.003.

Yu, B., Yang, Z.-Z., Chen, K., & Yu, B. (2010). Hybrid model for prediction of bus arrival times at next station. *Journal of Advanced Transportation, 44*(3), 193–204. doi:10.1002/atr.136.