Matias Williams and Ehab Ebeid
11.S955
Fall 2019
Project Revised Report

Link-Level Bus Travel Time Prediction using ConvLSTM: Application to the MBTA System

"Give the pupils something to do, not something to learn; and the doing is of such a nature as to demand thinking; learning naturally results."

— John Dewey

Introduction

Public transit users are attracted to accurate and reliable information about the service they are using. Most important among these is information about when buses or trains will arrive at a particular stop that the passengers intend to use. Historically, this information was displayed mainly in digital passenger information system and was pulled from scheduling data alone.

Advances in computing in the transportation field has changed both the source from which arrival and departure information is inferred and the ways in which it is communicated. Increasingly, this information is communicated primarily through smartphone applications relying on automated trip planning software, such as Google Maps and Transit. The source of the information is also increasingly not static timetables, as is the case with the original General Transit Feed Specification (GTFS) protocol. Transit agencies have found that their existing automatic vehicle location (AVL) systems can be used to provide dynamic real-time information about vehicle arrival and departure. These are now most commonly formatted following the standards of such protocols as GTFS Real-time and the Service Interface for Real Time Information (SIRI).

Real-time transit feeds rely both on the vehicle's live location as well as information about expected travel time in order to predict a bus's arrival time. Travel times used for the production of transit feeds, including real-time ones, mostly consist of historical average. They do not effectively capture historical dependencies that can more accurately predict a bus's arrival time.

We propose a more intelligent model, following Petersen et al. (2018), for the prediction of bus travel times at the link level, that takes into consideration both spatial and temporal correlations to a greater extent than in the industry standard. We decided to use state-of-the-art techniques in deep learning, namely convolutional neural networks (CNN) and Long short-term memory (LSTM) recurrent neural network architecture, and

their combination into ConvLSTM, to allow us to discover patterns across both time and space.

We propose to conduct this analysis with data obtained from the Massachusetts Bay Transit Authority's AVL repository, which is a rich source of bus transit data.

We have chosen to demonstrate the model MBTA Bus Route 1, a high-frequency, highly used bus route among the MBTA's routes, and have isolated the northbound direction for the purposes of our analysis.

Data Preprocessing

Link-level travel times are not readily available from the MBTA in tabular form, where each link on a bus route is identified by the timestamps of the bus' entering and exiting that link, the travel time through it, and the dwell time at the stops.

The MBTA does, however, collect information about stop events within their AVL reporting systems, which has been routinely available since January 2012. This dataset can be converted to link-level travel times, though they cannot accurately provide such information for all the links on all bus trips, since buses, unlike rapid transit systems, commonly skip stops.

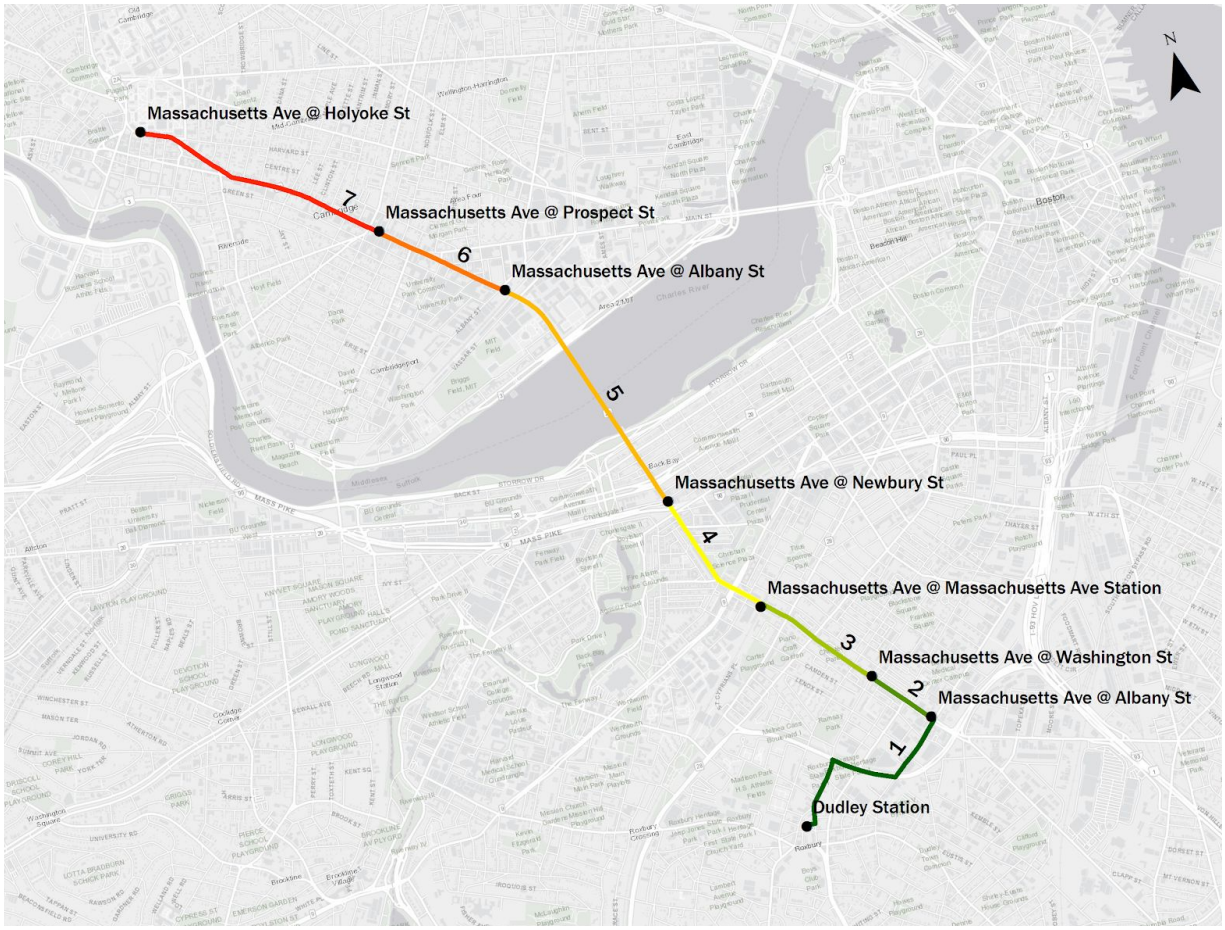
To resolve this issue, we elected to divide the route into links based on our definition of the significant stops, which are most often stopped at by the bus and are skipped the least often. The links are summarized in the following table:

Link Number	**Start Stop**	**End Stop**
1	Dudley Square Station	Mass. Ave at Albany St (Boston)
2	Mass. Ave at Albany St (Boston)	Mass. Ave at Washington St
3	Mass. Ave at Washington St	Mass. Ave Station
4	Mass. Ave Station	Mass. Ave at Newbury St
5	Mass. Ave at Newbury St	Mass Ave at Albany St (Cambridge)
6	Mass Ave at Albany St (Cambridge)	Mass. Ave at Prospect St (Central T)
7	Mass. Ave at Prospect St (Central T)	Mass. Ave at Holyoke St (Harvard T)

Table 1: Definition of links along the northbound direction of MBTA Route 1.

This enables the conversion of the stop event data into link-level information as long as the bus has served a pair of stops that define a link. For each link, travel time were calculated based on the bus's departure from the starting stop and arrival at the end stop.

Map 1 shows the seven links with the bus stops that delineate their borders.



Map 1: The defined links and boundary stops on the MBTA 1 Bus northbound.

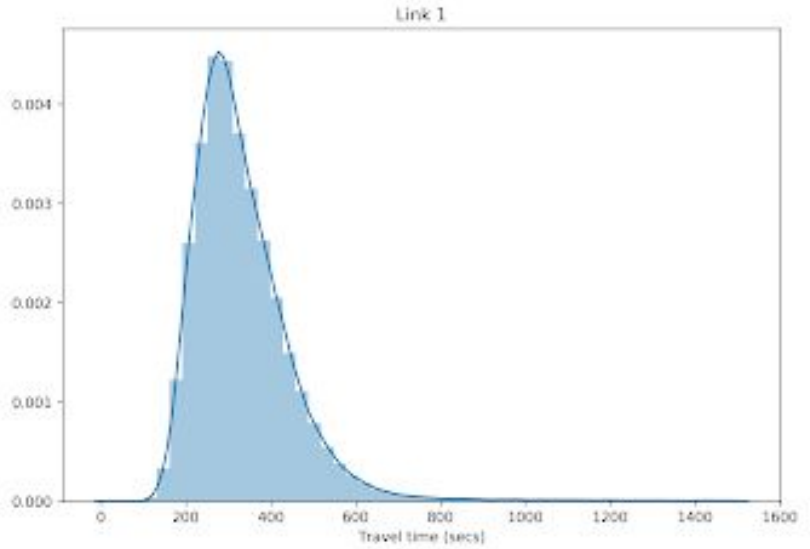


Figure 1. Distribution of travel time (in seconds) of Link 1. Note that dwell is not included.

The links were found to have travel times that seemed normally distributed but with long, thin tails, indicating a positive skew. This makes sense as negative and very small values are not possible, but arbitrary large values can occur due to traffic or vehicle malfunction.

To limit outliers and establish a maximum value for the purposes of normalizing, 1500 seconds (or 25 minutes) was chosen as the cutoff.

See figure 1 for an illustration of a typical link's travel time distribution. Note that it excludes dwell time, however, which when included substantially increases the stochasticity of the travel time for the first link especially.

We decided that we were going to predict the travel time of the bus between 6am to 10pm in intervals of 15 min. We then created time intervals from [0,..,55] where each represents a time slot of 15 minutes from 6am to 8pm. For example, time interval 0 will be 6am to 6.15am and time interval 1 is 6.15am to 6.30am.

Model

Input and Output

Since our model is centered on real-time application, it must be responsive to the dependencies found in information from the recent past, in order to predict travel times in the real future.

Thus, we propose the use of time windows of length equal to 8 time periods (of 15 minutes each), meaning a total of two hours, as the lookback for the model. These 8 time periods will then be used to predict the next four time periods (meaning the next hour of travel times along the route).

The input x consists of the travel times for eight time intervals, while the corresponding y consists of the travel times for the four succeeding time intervals.

We thus arranged our link level travel time data in samples of *N* matrices for each of x and y. An x matrix is of shape *x time intervals* $\times$ *links,* meaning 8×7 whereas a y matrix is of shape *y time intervals* $\times$ *links,* meaning 4×7.

Each observation of travel time in the dataset is assigned to its appropriate link, the time interval in which it occurred, its place within the sample set *N*. In cases where there is more than one data point for a given fifteen-minute interval over a given link, we calculated the mean of those link travel times and stored it in the matrix.

In cases where there are no observations for a given fifteen-minute interval and a given link, the average of the nearest time intervals on the same link is used to fill the missing data point. Other methods of filling in the missing data should be considered, including, for example, a three-dimensional interpolation.

We normalize our input values by dividing the total link travel time by 1,500 (the maximum value per our outlier removal), resulting in a 3D input matrix composed of size (39634, 8, 7) for x and corresponding y matrix of (39634, 4, 7).

Model Architecture

Our new model considered all the lessons learnt we previously talked about and only two ConvolutionalLSTM as we see no need in using more than that. As our activation functions in every layer in ‘relu’ we decided to add a BatchNormalization layer after

each other layer and a DropOut layer to prevent overfitting. Then, we flatten the output and added a dense layer to get the output as expected.

Figure 6 shows a graphic representation of our model and Figure 7 its summary.

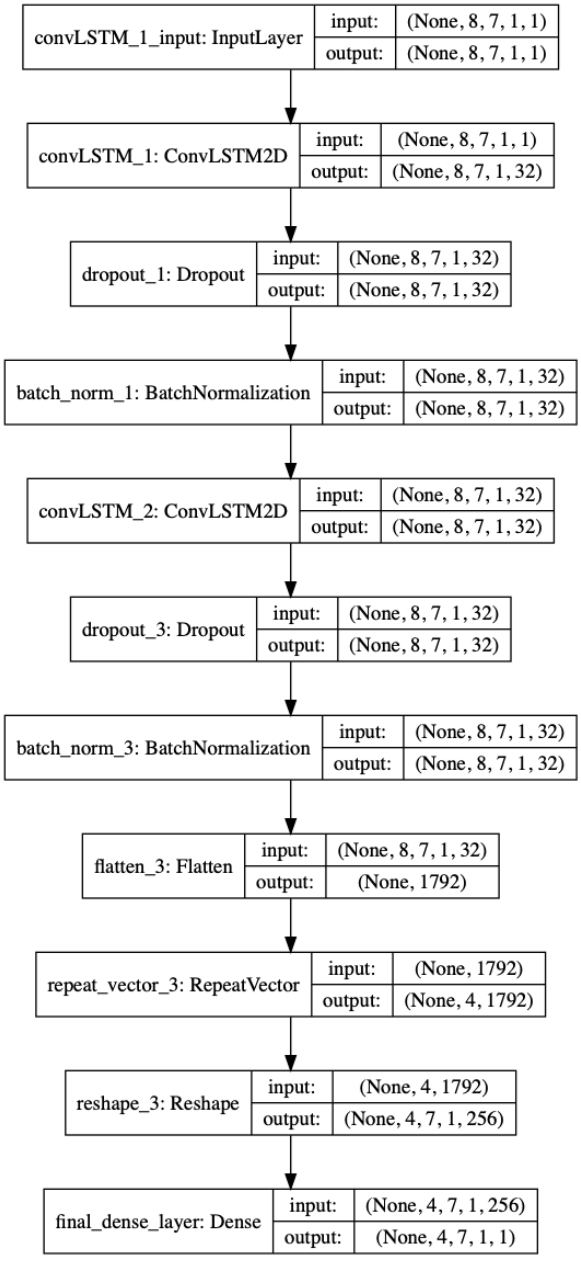


Figure 6. Model architecture.

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
convLSTM_1 (ConvLSTM2D)      (None, 8, 7, 1, 32)       12800
_________________________________________________________________
dropout_1 (Dropout)          (None, 8, 7, 1, 32)       0
_________________________________________________________________
batch_norm_1 (BatchNormaliza (None, 8, 7, 1, 32)       128
_________________________________________________________________
convLSTM_2 (ConvLSTM2D)      (None, 8, 7, 1, 32)       24704
_________________________________________________________________
dropout_3 (Dropout)          (None, 8, 7, 1, 32)       0
_________________________________________________________________
batch_norm_3 (BatchNormaliza (None, 8, 7, 1, 32)       128
_________________________________________________________________
flatten_3 (Flatten)          (None, 1792)              0
_________________________________________________________________
repeat_vector_3 (RepeatVecto (None, 4, 1792)           0
_________________________________________________________________
reshape_3 (Reshape)          (None, 4, 7, 1, 256)      0
_________________________________________________________________
final_dense_layer (Dense)    (None, 4, 7, 1, 1)        257
=================================================================
Total params: 38,017
Trainable params: 37,889
Non-trainable params: 128
_________________________________________________________________
```

Figure 7. Model summary.

Hyperparameter Tuning

To define the model, we tested different hyperparameters, before settling on some chosen optimal ones, as seen in Table 2 below.

Hyperparameter	Tested Values	Chosen Optimal
Learning rate	0.0001, 0.0005, 0.001	0.0005
Filters (ConvLSTM)	16, 32, 64	32
# ConvLSTM layers	2, 3, 4	2
Activation functions	ReLu, Sigmoid, Softmax	ReLu
Dropout rate	0.1, 0.2	0.2
Look-back time intervals	8	8
Prediction time interval	4	4

Table 2: Tested versus chosen hyperparameters.

Training, validation and testing

We trained our model using 39,634 samples, collected as described in the data processing section. We decided to split our samples into 70% for training, 15% for validation and 15% for testing.

Loss function

We chose mean-squared error as our loss function.

In Figure 8, we can see the how loss has evolved for our new model. Even though the loss is decreasing over each epoch, it does not seem to be getting low enough to consider that the model is deeply learning from our samples. We have tried different models, showing similar results to this one. These results seem to be promising but there is a lot of room for improvement, mainly by better understanding the input data.

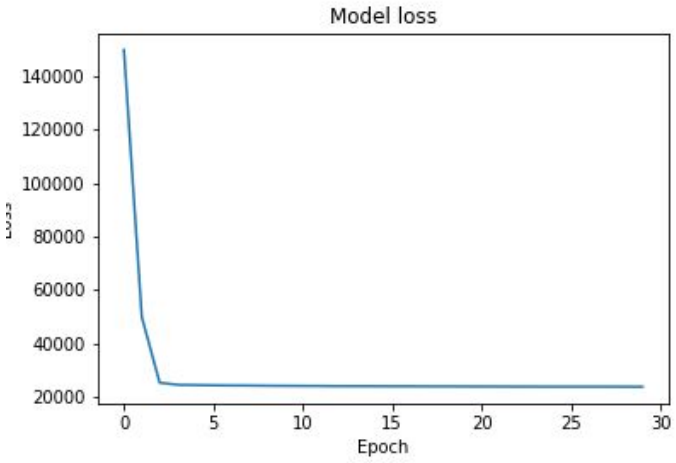


Figure 8. Training loss value for our model after 30 epochs.

In Figure 9, we can see the validation loss for our new model. There was no overfitting which might seem as a good results but the loss is still quite large, which means that the model performance might not be very strong.

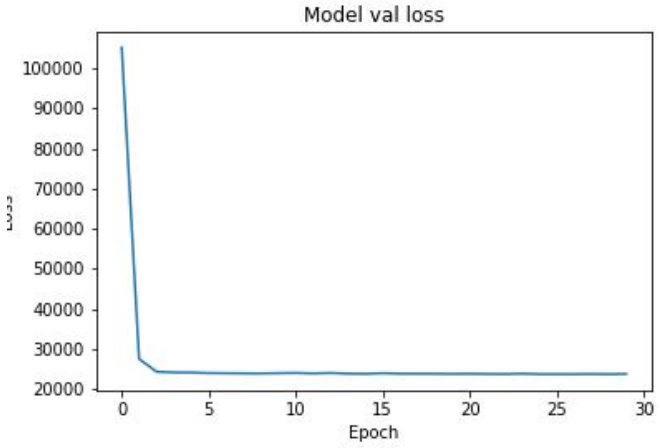


Figure 9. Validation loss value for our model after 30 epochs.

Prediction accuracy

To analyze the prediction accuracy we decided to compare our results to the historical average for each combination of link and two-hour, one-hour x, y pair of travel times. This type of historical average is most similar to what transit agencies usually use to feed the real-time transit feeds and was thus used as the benchmark to determine whether the ConvLSTM model provides an edge over the traditional methods used by transit agencies.

The model's overall performance is marginally better than the use of historical average, with lower values for mean absolute error, mean absolute percentage error, and root-mean-square error.

	MAE (sec)	RMSE (sec)	MAPE (%)
ConvLSTM Model	81.432	114.546	10.83%
Historical Average	101.000	138.884	13.43%

Table 3: Overall model performance versus the historical average.

When analyzing the model performance by link or by time interval, the general picture becomes more partial, however.

Table 4 shows that the ConvLSTM model is a better real-time predictor of travel times for links 2 to 6, but not for the first and last link. For the first link, this makes sense as its total travel time is especially stochastic. This may also be the case with the last link.

	ConvLSTM Model			Historical Average		
Links	**MAE (sec)**	**RMSE (sec)**	**MAPE (%)**	**MAE (sec)**	**RMSE (sec)**	**MAPE (%)**
1	*91.68*	*106.19*	*12.19%*	*74.28*	*85.59*	*9.88%*
2	12.77	17.93	1.70%	18.97	22.96	2.52%
3	162.67	179.42	21.63%	265.79	276.10	35.34%
4	72.38	107.16	9.62%	77.70	101.39	10.33%
5	95.25	139.08	12.67%	155.33	157.98	20.66%
6	94.64	122.71	12.58%	94.91	123.00	12.62%
7	*40.64*	*46.60*	*5.40%*	*20.03*	*23.89*	*2.66%*

Table 4: Model performance versus historical average by route link.

Meanwhile, Table 5 shows that the ConvLSTM model outperforms the historical averages for all but the first predicted time interval (*T+15 minutes),* in which it performs slightly worse.

	ConvLSTM Model			Historical Average		
Time prediction	**MAE (sec)**	**RMSE (sec)**	**MAPE (%)**	**MAE (sec)**	**RMSE (sec)**	**MAPE (%)**
T + 15 min	128.90	165.00	17.14%	119.01	165.14	15.83%
T+ 30 min	83.72	117.92	11.13%	119.94	163.26	15.95%
T + 45 min	62.24	72.45	8.28%	84.52	112.04	11.24%
T + 60 min	53.93	85.68	7.17%	85.59	112.66	11.38%

Table 5: Model performance versus historical average by predicted time interval.

We conclude that excluding the first and last link, the model performance may be significantly better than the use of historical averages by transit agencies to predict real-time vehicle arrival.

Conclusions

Our main conclusion is that our ConvLSTM model performs better than historical average given optimal hyperparameters. Nevertheless, when thinking about the practical implications of these results, due to long execution time, historical average may be best for transit agencies on a day-to-day basis.

Comparing time intervals performance between historical average and our model, only in the first time interval historical average performs better. The cause of this difference is not clear enough to understand why ConvLSTM model is not performing better in all the time interval.

Last, the first and the last link are the ones where historical average performs better than ConvLSTM. This could be related to the sparsity of data for total link travel times for those two links.

Improvements to Model

Our model could be improved by some changes either in data preprocessing or having more information as an input. For instance, we could consider identifying data as weekday or weekend day, so that our model could learn that as well. Also, we could separate peak hours from non-peak hours, because they are actually very different from one another, and are one of the reasons for the sparsity of link total travel time.

Further research

Analysis should be continued in order to assess whether Deep Learning algorithms could provide a better estimation tool than historical average time. In fact, using other Deep Learning models such as Sequence to Sequence, could show better results. Also, changing the amount of look back time intervals needed in order to predict the next time intervals should be studied.

Lessons Learnt

While doing our work for this paper, it helped us better understand deep learning models, and its components.

First, the most evident finding is that data preprocessing is a huge part of the work and should be carried out careful enough, to avoid misrepresenting data, to get a more efficient working model, to be able to predict more accurately, among others. For instance, it was very important to correctly normalize data. We started training our model without normalizing the inputs, a problem that generates a large error in our solution. That was the reason to include a BatchNormalization layer as the first one. The problem with this solution is that it only does a normaliziation along the batch values, which was 32, instead of doing that through the entire dataset. So our two first improvements were to take out that first layer, and also do some normalization over our input data (not output).

Second, we attempted to use tuples without having a timestep or time correlation with a Convolutional LSTM thinking that would enable the model to recognize the logic of temporal relation within the tuples. This was an assumption that we made, thinking that a Deep Learning model will be able to understand it in some way. After realizing this was not working, we decided to change our approach. We rearrange our data to have time intervals to enable a robust temporal-space correlation in the model. The problem with this approach was that data was only organized by date, thinking that we need to be able to predict, for example, what would happen at 8am and should be different from predicting what would happen at 5pm. So, we decided to break the samples group by date into samples group by 2 hours period. This allows us to go from more than 2,000 samples to almost 40,000, and still keep the spatio temporal relationship due to the use of ConvLSTM.

Third, we used many layers because we were following a model similar to that described in Petersen et al. that was quite complex and had a vast amount of data and samples to train a large network. Our mistake was trying to simulate that model instead of understanding that we have less samples and a simple model should work better in our approach. So we tried different architectures, also using different amounts of ConvLSTM layers and we reached an optimal architecture that shows the results described before.

References

Petersen, N.C., Rodrigues, F., Pereira, F.C., 2019. Multi-output Bus Travel Time Prediction with Convolutional LSTM Neural Network. Expert Syst. Appl. 120, 426–435. https://doi.org/10.1016/j.eswa.2018.11.028