Efficient Prediction of Region-wide Traffic States in Public Bus Networks using LSTMs*

Marcos Amaris[1] Mayuri A. Morais[2] Raphael Y. de Camargo[2]

***Abstract*— Public bus systems are impacted by many factors, such as varying traffic conditions, passenger demand, and weather changes. One can combine all those factors that affect bus travel times into a single factor called link state, where a link represents part of a bus route. Several works exist that predict single link states using different statistical and machine learning approaches. More recently, deep learning techniques, such as LSTMs, started to be used to predict the state of entire bus routes. The main problem with this approach is that it uses extensive computational resources.**

In this work, we evaluate the use of LSTMs to predict the state of entire city regions instead of single routes. It has two advantages: (i) the state of each link is evaluated only once for all the bus routes that cross it, and (ii) information from buses from all routes can be used to determine future link states. Using a shallow bidirectional LSTM architecture produced accurate state predictions with an average MAPE of 12.5. Moreover, we show that it can be trained daily and used to predict link states in real-time for a large metropolis, like São Paulo.

I. Introduction

Nowadays, half of the world population lives in about 37 megacities with more of 10 million inhabitants [1], such as São Paulo, with its 12.04 million inhabitants. Large cities have complex interactions among their entities (people, vehicles, objects) and understanding these interactions is still an open challenge [2].

Managing public bus transportation system in these system is complex, specially for cities in developing countries, which have deficient transit infrastructure and frequent traffic jams. This results in highly unpredictable trip times and frequent bunching events. One important way to improve the service level is to use smart city enabling technologies.

In several cities buses operate with different sensors, such GPS (Global Position System) sensors [3]. Using real-time information on bus positions, it is possible to estimate the mean velocity of buses on different roads, allowing better bus travel time estimations. The subject of travel time predictions has been extensively studied, employing from simple techniques, such as [4] to more advance ones, such as deep learning [5]. More sophisticated techniques, such as Long Short-Term Memory (LSTM) networks, can improve predictions [6], since they consider the state of the entire route and generate predictions of the evolution of this state.

But in megacities, scalability in the prediction mechanisms is an essential factor. São Paulo has a fleet of 15000 buses, operating over 2000 routes and 18000 bus stops, generating around 24 million data points per day. Training one model for each route becomes computationally expensive, making the use of LSTM and similar models impractical. Similarly, running predictions is also problematic, since we have to update the state of all bus lines every few seconds.

In this work we propose and evaluate the use of LSTM models to predict the state of entire areas in the city. We model the public bus network as a graph, with routes divided in links and nodes, and multiple routes sharing each link. Each area typically contains a few dozen links and bus routes, which brings two important advantages: (i) the model can use information from buses from multiple routes, improving the estimation the state of each link inside the region, and (ii) each link is part of only a single LSTM model, reducing the number of models required for the city by at least one order of magnitude. We evaluated five LSTM-based architectures to predict link states of individual bus lines and of city regions. We compared they performance in terms of accuracy and computational requirements to define the best strategy to use in a megacity such as São Paulo. This work has three important contributions:

- We propose and evaluate the prediction of future link states by geographical areas of the city using LSTM-based architectures;
- We compare five LSTM-based architectures with simpler predictions models for both route-based and area-based predictions;
- We show the viability of performing accurate future link state predictions in real-time for large cities.

The remaining of the paper is structured as follows. In Section II we present a theoretical background in Neural Networks and links state predictions. We discuss related works in Section III, followed by the methodology in Section IV, and the experimental results in Section V. Finally, we present the conclusions and future work (Section VI).

II. Background and Concepts

The prediction of links states in bus urban networks is fundamental because bus system operators and workers need to take fast decisions without enough information. Simulations permit to create hypothetical situations to evaluate the impact of the interactions of different variables in a

*This research is part of the INCT of the Future Internet for Smart Cities, funded by CNPq (proc. 465446/2014-0), Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Funding Code 001 and FAPESP (procs. 14/50937-1 and 15/24485-9)

[1]Marcos Amaris is with Faculty of Computing Engineering, Federal University of Pará, Campus Tucuruí, Tucuruí-PA, Brazil. amaris@ufpa.br

[2]Mayuri and Raphael is with Department of Computer Science, Center for Mathematics Computing and Cognition, Federal University of ABC, Santo André-SP, Brazil mayuri.ann@gmail.com, raphael.camargo@ufabc.edu.br

controlled environment [7]. In this section, we provide some background about link states predictions in urban transport (see Section II-A) and background about Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM) and derivatives (see Section II-B).

A. Time series, Geo-location and Link states Predictions

Currently, the number of buses of public transport in the city of São Paulo is around 14.500; Buses are distributed in 2271 bus lines approximately. All buses operate with different sensors on them, collecting data from passengers and other information. One of these sensors is the GPS (Global Position System), used for the geo-location of each bus in the network [3]. Nowadays, measurements coming from GPSs offer the opportunity to investigate interactions and behavior of the buses during their travels. This can be done with high temporal and spatial resolution and studied independently as time series.

Time series are characterized as a set of observations collected over time in an ordered and sequential manner. The mathematical representation of an uni-dimensional time series is a vector Z, of order $n \times 1$, where n is the number of observations. Multiple time series can be analyzed jointly in order to use the statistical information between them and thereby to find patterns which can be used for spatial forecasting [8]. Regression techniques based on machine learning are used on data series for temporal or spatial analysis, to try to predict the future values of a series or to predict the future values or otherwise of another series, respectively.

Geo-location information can be studied using graph algorithms. A graph G is a set of vertices V and a set of edges E, comprising an ordered pair $G = (V, E)$. Analogously in a bus urban network, each vertex is represented as a bus stop or a terminal and each edge is represented as the distance between two continuous bus stops. A network can be represented as a graph $G = (V, E)$. A graph can be described by the adjacency matrix. This matrix describes how the vertexes of the graph are linked and the distance or weight between all the vertexes.

All models are abstractions of a reality and simulations can be used to evaluate the correctness of such models. To build a model aiming to evaluated future link states of the bus urban network of São Paulo city is a real problem of Big Data and data analytics. GPS of each one of the buses is used for the geo-location and this data is used as historical information of the buses to analyze the link state of the network. Thus, travel times of the buses crossing the links are characterized by discretizing the temporal and spatial data from each one of the GPS's of the buses.

Figure 1 presents a idea where different stops are shared by different routes. When links travel time are organized and analyzed by line buses in these routes, link travel time of the same links are computed several times. Figure 2 corroborate this idea, showing 4 different time series of the same links when they are computed with data from buses of the same bus line or trip.

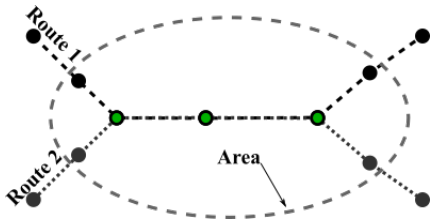

Fig. 1. Graphical representation of several intersections of a bus urban network. Bus stops in green are shared by route 1 and 2.

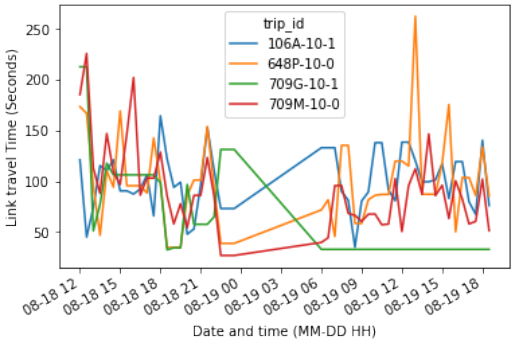

Fig. 2. 4 different time series of link travel time of a link shared by 4 trips.

Different set of time series about the states of each link of the bus urban network can be computed with the geo-localization of each bus. These series can be organized and analyzed in different groups. Thus, one purpose of this research is to explore the relationship of link states between different links when they are grouped in two ways: by geographical areas and by bus route (trip).

B. RNN, LSTM and derivatives

Artificial Neural Network (ANN) architectures are classified in two mains categories: feed-forward and recurrent. In the feed-forward algorithm the output of one layer is used as input in the next layer and so on, and the connections between the nodes do not form a cycle. In recurrent neural networks (RNNs), the outputs of each layer are feedback as input to the same layer. These networks can store internal states, which are useful in problems involving time series [9].

A simple RNN compute the output of y_t as $f(W_y h_t)$, where h_t is $\sigma(W_h h_{t-1} + W_x x_t)$, and W_y, W_h and W_x are matrices; for hidden layers output h_t, past hidden activity h_{t-1} and the input x_t. The logistic function $\sigma(\cdot)$ perform a nonlinear relation in the recurrence. Long Short-Term Memory (LSTM) and derivatives belongs to RNN approaches. It maintains an internal state with accumulated information from previous data.

LSTM introduce linear relations between this recurrence adding memory cells c_t and c_{t-1}. LSTMs also have gates for input and output data in the cells. The input gate, forget gate, output gate, and memory cell in an single LSTM cell are calculated using the following equations

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1}) \quad (1)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1}) \quad (2)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1}) \quad (3)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t) \quad (4)$$

$$h_t = o_t \odot \tanh(c_t) \quad (5)$$

where i_t, f_t, o_t and h_t are the input gate, forget gate, output gate and the layer output. $\odot$ is the element-wise vector/matrix multiplication operator.

A single LSTM architecture is comprised of a single hidden LSTM layer followed by a standard feed-forward layer. Multiple LSTMs can be stacked and temporally concatenated to create more complex architectures. Recently, variants of LSTM architectures were created, such as the use of a bidirectional training process [10] or the inclusion of convolutive processes in the LSTM gates [11]. Bidirectional LSTM (Bi-LSTM) contains two separate LSTM hidden layers that receive the input data in the forward and backward directions. ConvLSTMs try to capture spatio-temporal correlations using convolutional structures in the training process. These convolutional processes are applied in equations (1), (2) e (4), affecting the input gate i_t, forget gate f_t, and output gate o_t respectively. LSTMs and derivatives are describe in detail elsewhere [9], [11]. In Section IV-B, we describe the architectures used in this work.

III. Related Works

In the literature we find two distinct types of predictions for bus travel time: predictions of total travel time and predictions of time / arrival time at a specific point, where efforts are more concentrated in this second type of prediction. In this section we discuss some recent works using these techniques [12], [13], [14], [15], [16], [17], [6]. Most of the work on predicting link states are based on ARIMA (AutoRegressive Integrated Moving Average), Markov Processes and neural networks. In fact, LSTM and recurrent neural networks are based on similar techniques than hidden Markov Models. In this sense, while hidden Markov models ignore past inputs, RNN works with this principle.

Ma *et. al* [12] performed a stochastic analysis using Markov chains. They focused their method in two major phases: Markov chain identification and probability distribution estimation. The outputs of the first phase are probabilities of link traffic states, link TTDs (conditional on states), and time-space dependent transition probabilities. They used a Gaussian Mixture model (GMM) based clustering algorithm to define the states of the model and a logit model formulation to estimate the transition probabilities. The authors also considered correlations in time and space, and created a heuristic based on the statistical information of the data (i.e. probability distributions, transition probabilities and well-defined states).

Kumar *et. al* [13] used kNN algorithm to classify data input. They used a Euclidean distance to classify input data in relation to historical data. They performed a sensitivity analysis based on the Mean Absolute Percentage Error (MAPE) to determine the optimum number inputs to be used for the kNN algorithm. Once, the inputs are identified, the estimation was carried out by using a hybrid time discretized model. Authors used Auto Regressive Integrated Moving Average (ARIMA) models to combine samples from time series to make estimation, following the historical data autocorrelation profiles. Their model can not deal with long-term travel time estimations and with a variety of weather parameters which were difficult to calculate in real-time. To deal with this, they used an exponential smoothing technique with recursive estimation scheme based on the Kalman Filtering (KF) technique. To test their model, they used data from the Metropolitan Transport Corporation (MTC) bus of the city of Chennai, India.

Kun Tang *et. al* [14] used large-scale and sparse GPS trajectories generated by taxicabs. Authors created a tensor-based Bayesian probabilistic model for citywide and personalized travel time estimation for taxicabs. This model comprised three components: map matching, travel time tensor construction and travel time tensor factorization. After map-matching, each GPS trajectory was converted into a road trajectory, which was represented by the travel time of a driver traversing a road segment in a time slot. They included a Markov Chain Monte Carlo (MCMC) method in the last component travel time tensor factorization. MCMC was applied to sample directly posterior distributions, because, they considered the distribution of a missing travel time as a multi-dimentional integral. MCMC sampled from a distribution based on constructing a Markov chain that has desired distribution as its equilibrium distribution. This model was applied in different case studies on the citywide road network of Beijing, China. They performed also correlations in time-space of the information.

According to Kumar *et. al* [13], time series analysis is identified to be suitable for bus travel time estimation under different traffic conditions, because of their ability in capturing the variations in travel time. Ma *et. al* [12] used Markov chain to predict the changes of the link states while buses travel for those links. In our proposal, we will use Neural Networks to model the evolution of the links as a model of the city, without concerning about the buses.

Some recent work has used LSTM for traffic state predictions. Cui *et. al.* used stacked bidirectional and unidirectional LSTM for Forecasting Network-wide Traffic State with Missing Values [17]. They also proposed a data imputation process in the structure of the LSTM. They designed an imputation unit to infer missing values. Although, Cui *et. al.* presented optimum results they did not present experiments about a high scale prediction model.

Petersen *et. al.* [6] compared the accuracy of LSTM techniques and used them to predict multiple step in a single bus line of the city of Copenhagen in Denmark. They presented good results in the accuracy of the models, however these models neither were not tested in a huge scale experiment and the behavior or statistics of the link states

were not as complexes as in the city of São Paulo in Brazil.

To the best of our knowledge, this is the first work that tackles the prediction of link travel time of a public service bus in a big region-wide size. This approach has the potential to improve the trade-off between accuracy and efficiency when LSTMs models are created by geographical regions and not by bus lines.

IV. Methodology

Here, we describe the data acquisition and its preparation process, followed by the methodology, where LSTM architectures and the experimental platform are presented.

A. Acquisition and Preparation of the Dataset

We use historical data from the public bus fleet of São Paulo, from June 21^{th} to September 28^{th} of 2017. We processed the GTFS[1] data from São Paulo using a framework developed at our research group [18], which provides scalable processing of historical and real-time bus GPS data. GTFS data is transformed into a graph, where each node represents a point equidistant from two consecutive bus stops in a route, and each link the bus trajectory between two nodes. Each link may shared among multiple bus routes that have the same trajectory. The link state prediction is part of the framework and is useful for improving travel time (TT) predictions.

We selected five geographical areas from the city to perform the area-based link state prediction. Table I shows the number of links inside the areas and bus routes that crosses the areas. The areas are located in the different parts of the city (Figure 3). We also selected three bus lines from each area to perform the route-based predictions.

TABLE I
GEOGRAPHICAL AREAS SELECTED FOR THE EXPERIMENTS

No.	Name	Links	Lines	Route Codes
1	Lapa	33	14	809N-10-0, 7181-10-1, 875H-10-0
2	Paulista	25	90	930P-10-0, 715M-10-1, 719R-10-1
3	Butantã	20	41	778J-10-1, 106A-10-1, 648P-10-0
4	Itaim Bibi	61	82	709G-10-1, 709M-10-0, 5175-10-1
5	Ibirapuera	28	43	5614-10-1, 5611-10-1, 177Y-10-0

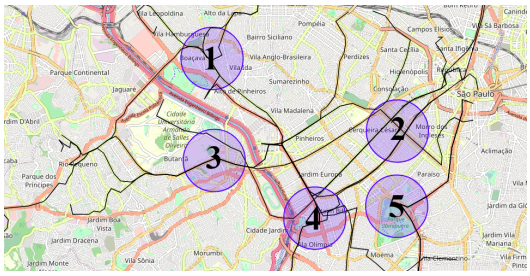

Fig. 3. Map with the 5 geographical areas and the selected bus lines that cross each area.

We used historical data to compute the link travel time (ltt) for each bus and computed the link state as the mean ltt of all buses that crossed the link during a sliding window of 30 minutes. The obtained data is noisy, with some incorrectly reported GPS data, and we excluded buses with ltt smaller than 10 seconds and larger than 4.5 times the interquartile range (IQR) above the median. We filled missing values, resulting from no buses passing in the link for 30 minutes, with a forward-fill to propagate the last observed value, followed by backfilling for missing values in the series beginning. We removed periods between midnight and 6 am. Finally, we performed the following data transformation steps: (i) log-transformation, (ii) z-score normalization, and (iii) min-max normalization between 0 and 1.

[1]General Transit Feed Specification.

B. Prediction Algorithms

We applied the prediction algorithms over the link travel time series, each containing 3850 samples. We grouped time series by geographical areas and by bus lines and used them predict up to three forward time step, each with 30 minutes. For all the experiments, we used the last 7 seven days (252 samples) as test set, and the remainder as train set.

Our baseline comparison was to repeat state of a given link l state for the next three time steps. We also applied Linear Regression and Random Forests, using as input the last s states from link l and from its o nearest neighbors from each side, resulting in $(2o+1)*s$ input variables. We predicted the three future states (T_1, T_2, and T_3) of link l, and repeated the process for each monitored link. We applied these models only to data from individual bus lines. After evaluating all combinations of s and o values, we decided to use the last $s=4$ states from $o=1$ neighbors from each side, which provided the best predictions.

For the LSTM architectures, we provide as input the links states from all links from the line/region for n_steps_in time steps. The network then predicts the next three states (T_1, T_2, and T_3) for all links from the line/region. We used the same architecture from [6] for LSTM and Conv2DDeep. The Bi-LSTM and Conv2D are simplified architectures, with a single layer, which should be faster to train. If they provide comparable results to the their deeper counterparts, they should be preferable. We now describe the 4 LSTM architectures using a block syntax:

- **LSTM**: LSTM and Conv2DDeep have an encoder/decoder network topology. The decoder and decoder are created both by two hidden LSTM layers: LSTM → LSTM → RepeatVector → LSTM → LSTM → DENSE
- **Bi-LSTM**: This architecture allows the RNN to have both backward and forward information about the series at every time step. We used a single layer of Bidirectional LSTM: Bi-LSTM → DENSE
- **Conv2DDeep**: Similar than LSTM, but it has ConvLSTM2D layers. Follow a block diagram of this architecture: ConvLSTM2D → ConvLSTM2D → RepeatVector → ConvLSTM2D → ConvLSTM2D → DENSE
- **Conv2D**: In this architecture, we employed a single layers of ConvLSTM2D: ConvLSTM2D → DENSE

All LSTM techniques used 64 units in each layer, except Conv2D which uses 32 to reduce training time. LSTM and Conv2DDeep used `BatchNormalization` and `Dropout` layers between LSTM layers. We used the *Adam* optimizer, mean square error (*mse*), and batch size of 128 samples. We trained the LSTM architectures for 50, 250 and 500 epochs, with an early_stop mechanism, applied when the *mse* did not change for 5, 10 and 10 steps, respectively. We used as inputs the last 16, 24, and 36 time steps of the link states. Each LSTM architecture used 80% of data as training set and 20% for validation of the models.

For the experiments, we used an machine with Intel Core i7 with 8 cores at 3.60GHz, 64 GB of RAM, and a GPU Nvidia Geforce RTX 2080, with 4352 CUDA cores. We used Ubuntu 20.04 LTS 64-bit distribution with Linux kernel 5.8 and NVIDIA drivers 450.102.04. Experiments used Python 3 with `Tensorflow` 2.3.1 and `Keras` 2.4.3 packages, among others. Collected data, experimental results, and source codes are publicly available at a Github repository.[2]

V. Experiments and Evaluations

We evaluated the Mean Absolute Percentage Error (MAPE) to measure the accuracy of the different models. MAPE is equal to $\frac{100}{N}\sum_{n=1}^{N}\left|\frac{M_n - P_n}{M_n}\right|$, where M_n are the measured values and P_t are the predicted values, and N is the number of all the predicted samples.

Table II shows the MAPE of the experiments using the actual state T_0, to predict T_1, T_2 and T_3 of all the 15 bus lines. Results with a single link neighbor represented the best accuracy with Linear regression and Random Forests techniques. We used the default algorithm of Linear Regression, without modification of its initial values. For the Random Forests algorithm, the variables `n_estimators`, `max_depth`, `random_state` and `max_features` were initialized as 100, 10, 5 and 'auto', respectively. Table II, show the MAPE for LR and RF with a single link neighbor, models with more of two neighbors finished in worst predictions.

TABLE II
MAPEs of Simple techniques of Time Travel Predictions

Technique	T_1	T_2	T_3	**Average**
Actual **ltt** State T_0	19.85	23.47	26.11	23.14
Linear Regression	20.74	20.79	20.67	20.73
Random Forest	20.88	20.92	20.81	20.87
LSTM	12.70	12.48	12.53	12.92
Bi-LSTM	**12.31**	**12.36**	**12.24**	**12.5**
Conv2D	16.53	16.70	16.54	16.82
Conv2Deep	11.88	11.79	11.99	12.13

For LSTM techniques, the accuracy of the models improved when the number of input steps increased. The LSTM technique with the best MAPE were those with 32 input steps. Results show that Bi-LSTM presented the best accuracy when models are organized by bus lines, resulting in an average error of 12.5 to predict the 3 next steps in all the links of different bus lines.

Figure 4 presents the average of T_1, T_2, T_3, MAPE of the links which are in the geographical models, and they are also in the bus lines models. In this way, we compare the results of the links which make part of both approaches. In Figure 4 is possible to see that geographical area models presented better accuracy than models using all data of single bus lines. This experiment also tested the models when they were trained with 50, 250 and 500 epochs. The number of epochs does not demonstrated improvements of accuracy and they represented in the increasing of time to perform the training process. MAPE with 250 epochs and 500 was larger than 50, this happened because over-fitting in the training processes, this can be avoided using 5 epochs as patience. Geographical models presented a higher correlation between the links than models by bus lines. Geographical models had more information about the states of the links, due to the integration of many trips by link. Area with

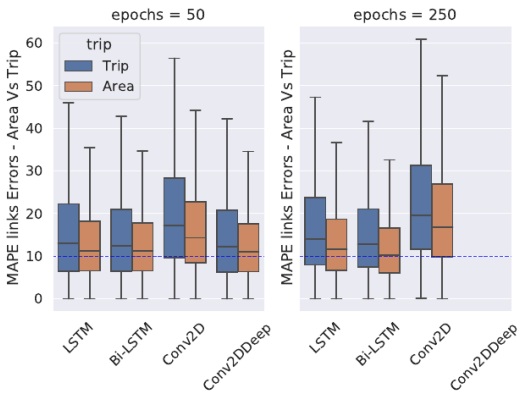

Fig. 4. MAPE of link state predictions for the area and trip models, using 32 time steps as input.

São Paulo city has more than 2000 bus lines and the graph representing the whole city has 28.242 links. Grouping in sets of 40 links representing regions of the city results in 700 models. Table III presents estimated time necessary to train these models on both approaches, geographical area and bus lines or trips. To reach the total time when using the line approach, we multiplied the average training time for the 15 selected bus lines by the total number of bus lines in São Paulo. Similarly, for the area approach, we multiplied the mean training time for the five selected areas by 700, representing the 700 models. **Bi-LSTM** with 50 training epochs presents the best trade-off between accuracy and training time, since it required the least amount of training time while presenting a MAPE value only 1% above those obtained with 250 training epochs. The LSTM also had similar results in both metrics and could also be used. Both require at most 8 hours for training in a single GPU, which could be done at night. With 2 GPUs or next generations ones, this value could be reduced to a few hours.

[2] `https://github.com/marcosamaris/links_states_prediction_LSTM`

TABLE III

ESTIMATED TIME (IN HOURS) REQUIRED TO TRAIN ALL MODELS FOR THE ENTIRE CITY, USING THE AREA AND TRIP APPROACHES, FOR DIFFERENT LSTM ARCHITECTURES, NUMBERS OF TRAINING EPOCHS AND PAST TIME STEPS.

No. Epochs	Input steps	Ave. Epoch	LSTM		Bi-LSTM		Conv2D		Conv2Deep	
			Area	Trips	Area	Trips	Area	Trips	Area	Trips
50	16	48.7	5.62	16.07	**2.42**	6.92	3.33	9.51	25.75	73.58
	24	48.1	6.92	19.76	**6.37**	18.19	6.32	18.05	45.03	128.66
	32	48.5	7.36	21.03	**8.14**	23.25	7.64	21.82	57.23	163.52
250	16	228.4	24.73	70.66	9.63	27.51	11.07	31.63		
	24	218.8	31.10	88.86	26.18	74.81	17.79	50.84		
	32	204.0	32.89	93.98	35.95	102.70	18.04	51.54		
500	16	347.1	48.80	139.42	12.18	34.80	15.07	43.05		
	24	323.9	61.15	174.71	32.57	93.05	18.02	51.49		
	32	309.4	65.04	185.84	37.99	108.55	17.08	48.80		

VI. CONCLUSIONS

In this work, we evaluate the use of LSTMs to predict the state of entire city regions instead of single routes. We compared four LSTM architectures, in respect to accuracy and computational time. The Bidirectional LSTM demonstrated the best trade-off between accuracy and efficiency. It showed an average MAPE of 12.5 and it would spend around of 8.14 hours to train 700 models to predict whole the links of São Paulo city when links are grouped by geographical areas. This shows that it is viable to deploy more complex and accurate models in large cities like São Paulo.

The work has some limitations, since we just estimated the total time to train the models. Performing real deployments could bring unexpected difficulties. We also did not deploy the model for real-time travel time predictions of single bus trips and this is our next planned step. Finally, although it is clear that having better link state estimates leads to improved travel time predictions, we did not compare travel time predictions accuracy with other methods.

REFERENCES

[1] W. Cox, "Demographia world urban areas," *14th Annual Edition ed. St. Louis: Demographia. Available: http://www. demographia. com/db-worldua.pdf. Date of access May, 30th of 2018*, p. 120, 2018.

[2] V. Albino, U. Berardi, and R. M. Dangelico, "Smart cities: Definitions, dimensions, performance, and initiatives," *Journal of Urban Technology*, vol. 22, no. 1, pp. 3–21, 2015.

[3] E. Mazloumi, G. Currie, and G. Rose, "Using GPS data to gain insight into public transport travel time variability," *Journal of Transportation Engineering*, vol. 136, no. July, pp. 623–632, 2009.

[4] F. Alrukaibi, R. Alsaleh, and T. Sayed, "Applying machine learning and statistical approaches for travel time estimation in partial network coverage," *Sustainability*, vol. 11, no. 14, 2019.

[5] M. Abdollahi, T. Khaleghi, and K. Yang, "An integrated feature learning approach using deep learning for travel time prediction," *Expert Systems with Applications*, vol. 139, p. 112864, 2020.

[6] N. C. Petersen, F. Rodrigues, and F. C. Pereira, "Multi-output bus travel time prediction with convolutional lstm neural network," *Expert Systems with Applications*, vol. 120, pp. 426–435, 2019.

[7] P. Wepulanon, A. Sumalee, and W. H. Lam, "A real-time bus arrival time information system using crowdsourced smartphone data: a novel framework and simulation experiments," *Transportmetrica B*, vol. 6, no. 1, pp. 34–53, 2018.

[8] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[9] F. A. Gers and 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*, vol. 3, 2000, pp. 189–194 vol.3.

[10] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[11] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," *CoRR*, vol. abs/1506.04214, 2015.

[12] Z. Ma, H. N. Koutsopoulos, L. Ferreira, and M. Mesbah, "Estimation of trip travel time distribution using a generalized markov chain approach," *Transportation Research Part C: Emerging Technologies*, vol. 74, pp. 1 – 21, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X16302248

[13] B. A. Kumar, L. Vanajakshi, and S. C. Subramanian, "A hybrid model based method for bus travel time estimation," *Journal of Intelligent Transportation Systems*, vol. 2450, no. December, pp. 1–17, 2017.

[14] T. Kun, S. Chen, Z. Liu, and A. J. Khattak, "A tensor-based Bayesian probabilistic model for citywide personalized travel time estimation," *Transportation Research Part C*, vol. 90, no. March, pp. 260–280, 2018.

[15] N. Ranjan, S. Bhandari, H. P. Zhao, H. Kim, and P. Khan, "City-wide traffic congestion prediction based on cnn, lstm and transpose cnn," *IEEE Access*, vol. 8, pp. 81 606–81 620, 2020.

[16] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.

[17] Z. Cui, R. Ke, Z. Pu, and Y. Wang, "Stacked bidirectional and unidirectional lstm recurrent neural network for forecasting network-wide traffic state with missing values," *Transportation Research Part C: Emerging Technologies*, vol. 118, p. 102674, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X20305891

[18] M. A. Morais and R. Y. Camargo, "A framework for scalable data analysis and model aggregation for public bus systems," in *Anais do III Workshop de Computação Urbana*. SBC, 2019, pp. 83–96.