MTL-LSTM: Multi-Task Learning-based LSTM for Urban Traffic Flow Forecasting

Bachelorarbeit
der Philosophisch-naturwissenschaftlichen
Fakultät der Universität Bern

Vorgelegt von

Samuel Martin Schwegler
HS 20

Leiter der Arbeit

Professor Dr. Torsten Braun
Institut für Informatik und angewandte Mathematik

Abstract

Increasing computing power and larger available datasets enable work with machine learning to solve complex problems. This is beneficial for a wide range of traffic flow prediction applications in the environment of large cities. Examples are vehicle navigation services, vehicle routing or traffic congestion management.

Long Short Term Memory Networks (LSTMs) are a specialized variant of Recurrent Neural Networks (RNNs) designed to learn long-term dependencies. Using Multi-Task Learning (MTL) offers the possibility to take advantage of traffic information for multiple trajectories (moving objects) and taking the spatial information shared by neighboring trajectories into account.

In this thesis different MTL architecture based LSTM predictors are used to explore spatio-temporal dependencies among adjacent trajectories hidden in large datasets. The aim was to predict the density (number of present objects during an interval) within rectangular grid cells, based on a data set collected from vehicles in the city of Porto, Portugal.

The thesis shows, that MTL-architectures which are combining dedicated and shared LSTMs performed best among the different tested architectures in this work, benefiting most from shared knowledge. By the state-of-the-art the MTL-LSTM indicated improvements of 10% - 15% in prediction accuracy and time consumption on both weekdays and weekends.

MTL-LSTM: Multi-Task Learning-based LSTM for Urban Traffic Flow Forecasting

Mostafa Karimzadeh[1], Samuel Martin Schwegler[1], Zhongliang Zhao[12], Torsten Braun[1], Susana Sargento[3]
Institute of Computer Science, University of Bern, Switzerland[1]
School of Electronic and Information Engineering, Beihang University, China[2]
Institute de Telecomunicações - Aveiro, Portugal[3]
Email : {mostafa.karimzadeh, torsten.braun}@inf.unibe.ch, samuel.schwegler@students.unibe.ch, zhaozl@buaa.edu.cn, susana@ua.pt

***Abstract*—Predicting traffic flow in large cities is beneficial for a wide range of applications, including vehicle navigation services, vehicle routing, and traffic congestion management. In this scenario, deep learning approaches such as Recurrent Neural Networks (RNN) and its variant Long Short Term Memory (LSTM) are excellent alternatives due to their ability to learn long-term dependencies. However, these neural networks only learn the temporal traffic information for each trajectory (moving object), failing to take advantage of spatial information shared by neighboring trajectories. This paper introduces MTL-LSTM (Multi-Task Learning-based LSTM) traffic flow estimator, which attempts to explore both temporal and spatial dependencies among adjacent trajectories. Specifically, we employ LSTM predictors with the MTL approach to explore traffic flow patterns across urban trajectories. To examine the proposed model, we predict traffic flow in Porto's city using a data set from buses and taxies. Our experiments show improvements of 10% to 15% over the state-of-the-art.**

***Index Terms*—Deep Learning, Multi-Task Learning, Traffic Flow Prediction, and Intelligent Transportation system (ITS).**

I. Introduction

Urban traffic flow forecasting is a significant function of the Intelligent Transportation system (ITS), which plays an important role in city traffic management and public travel safety. Besides, this forecasting type can help people improve daily travel plans and optimize public transport resource allocation. In recent years, due to advances in public infrastructure and data transmission technologies, more traffic flow data is available, promoting forecasting accuracy. Although several models have been deployed, many of them leverage conventional methods that may be unsatisfying to discover the deep correlations like temporal and spatial dependencies hidden in large datasets.

Adjacent trajectories simultaneously influence a moving object's trajectory; each moving object responds to others' stimulus. Traffic time series show strong temporal dynamics, since there are recurring events like rush hours. The road network structure dominates spatial correlation: slower vehicles are slowing down following vehicles on the same road segment; if a traffic jam occurs on a road segment, the adjacent road segments will also be affected [1][2]. The closer the trajectories are, the higher is the distortion; they are spatially dependent. Spatial dependence refers to the degree of spatial auto-correlation between independently measured values observed in geographical space [3]. Consequently, traffic flow prediction accuracy can not profit from sharply increasing traffic data. Therefore, new techniques are eagerly demanded to handle the abundant traffic data at a deep level.

As a subset of machine learning, deep learning has drawn tremendous interest from academic and industrial fields. Deep learning algorithms such as RNNs (Recurrent Neural Networks) and LSTM (Long Short Term Memory) can exploit complex temporal features of urban traffic. In the transportation research area, deep learning is increasingly presented in traffic flow estimation and achieves attractive performance [4]. Most existing traffic flow estimators are STL-based (Single Task Learning) algorithms. In an STL approach, forecasting traffic flow for each trajectory in a city is considered a single prediction task, in which an individual traffic predictor must be built and trained separately for each task (trajectory). The STL-based traffic flow estimators are neglecting the potential benefits of jointly considering traffic flow information (e.g., number of moving objects) across the entire neighboring trajectories in city environments.

To address the aforementioned shortcoming, this paper proposed MTL-LSTM (Multi-Task Learning-based Long-short Term Memory), a framework to forecast the future state of urban traffic in terms of numbers of moving objects (e.g., vehicles) in urban trajectories. Basically, MTL is a machine learning paradigm, and it aims to leverage useful information in multiple related prediction tasks to help improve the prediction performance of all tasks [5]. MTL can take more information into account than STL. By sharing representations between related tasks, we can enable the model to generalize better on the main task. MTL has a better ability to cope with noisy data than STL, which is due to MTLs ability to focus attention. MTL can help the model to determine the relevance or irrelevance of features and set its attention to the relevant features [6]. Those techniques are leading to improved prediction accuracy when comparing MTL and STL.

Our proposed MTL-LSTM is fed simultaneously by historical traffic data (number of vehicles) from neighboring trajectories, where traffic flow estimation in these trajectories is a related prediction task. MTL attempts to improve traffic flow prediction accuracy for one trajectory by utilizing the relations contained in other adjacent trajectories. In this way, the information from one trajectory can help neighboring (related) trajectories to have accurate traffic flow prediction accuracy.

The rest of this paper is organized as follows. Section II reviews related work. Section III discusses the problem statement. Section IV describes the proposed MTL-LSTM traffic flow estimator. The evaluation methodology and the results are presented in Section V. Finally, Section VI concludes the paper.

II. Related Work

Over the past few decades, many traffic flow forecasting models have been proposed to facilitate traffic management. Existing methods can be classified into three categories: parametric, non-parametric, and hybrid approaches. The autoregressive integrated moving average (ARIMA) was first introduced to forecast traffic flows [7]. Over the years, the ARIMA model has been used as the basis for several variants, e.g., seasonal ARIMA and Kohonen ARIMA [8][9]. These parametric approaches are suitable to deal with regular variations, but the performance is undesirable when traffic data shows significant stochastic and non-linear characteristics. To address this problem, non-parametric predictors have shown remarkable advantages in various traffic flow prediction tasks [10]. However, traditional non-parametric approaches are insufficient to model complex relationships and, consequently, fail to improve forecasting accuracy. To maximize the strengths of the introduced parametric and non-parametric approaches, hybrid approaches are explored to achieve attractive results. In [11] authors combine the SARIMA model with a Kalman filter. Hybrid approaches deliver plausible results in various practices, but the prediction accuracy is still limited in some scenarios. The work in [12] states that, when enlarging the training data size, the prediction performance quickly decreases. Therefore, hybrid models fail to take advantage of large datasets.

In recent years, deep learning approaches have increased attention in traffic flow forecasting to tackle the introduced shortcomings. Authors in [13] proposed a novel deep learning-based traffic flow estimator, in which a stacked autoencoder model was used to extract traffic flow features. In [14] it is proposed a deep learning-based traffic flow prediction model with CNN (Convolutional Neural Network) to extract spatial features and GRU (Gated Recurrent Unit) to capture temporal features. These studies show promising performance in estimating traffic flows. However, most existing predictors are STL-based models, which can not take advantage of information sharing among related tasks.

Fortunately, some attempts were made to forecast traffic states by introducing MTL [15]. MTL is a machine learning paradigm whose aim is to benefit from additional information in auxiliary tasks to predict the main task better. However, these models use deep learning algorithms with only several stacked neural network layers, which may restrict their prediction capacities. In summary, to meet the increasing needs of accurate traffic information in ITS, in this paper, we propose a Multi-Task Learning-based LSTM (MTL-LSTM), which aims to improve urban traffic flow prediction accuracy by jointly training traffic information of multiple adjacent trajectories.

III. Problem Statement

A traffic flow is defined as the total number of moving objects (e.g., vehicles) that pass through an area during a period. The area can be a road segment or a region in the city. In this research, a traffic flow predictor attempts to estimate future traffic states regarding the number of moving objects in the trajectories. To define the city's areas, we use the Python Google S2 geometry library[1]. S2 is a library for working with the geometry of the two-dimensional surfaces of the earth. It enables efficient access to spatial objects (spatial indexing) and helps to partition the geographical space (in our case, Porto, Portugal) into four-corner grid cells.

The coordinates of each RSU (Road-Side Unit) are known. Spatial indexing is used to map these positions of RSUs into grid cells. The grid cells are covering the trajectories of the RSUs, which are situated in the grid cells. In the city center, the RSUs are located denser to each other to cover the higher traffic. The usage of a single RSU can not indicate the number of connections made in the whole area; they only show the workload of one RSU. To get a general prediction for areas, the grid cells are used. With the standardized cell size, it is also easier to compare areas.

In this research, we use grid cells with a 1.27 km^2 area. The traffic flow collected from the trajectories in grid cells is stored as $C = \left[F^1 \ldots F^i\right]$, we refer C as density of the grid cell. Collected traffic flow can be shown as: $F^i = \left[x^t \ldots x^{t+T}\right]$, where $x^t \in \mathbb{R}^{N \times P}$ is a set representing the number of vehicles (V) in each grid cell (Tr) at time interval t. The traffic predictor $L(.)$ attempts to learn patterns of traffic flow at time T and make an estimation for the future time T':

$$L\left(\left(x^{(t)}, \cdots, x^{(t+T)}\right)\right) \equiv \left(x^{(t+1+T)}, \cdots, x^{T'}\right) \quad (1)$$

The extracted traffic information from the dataset for the city of Porto, Portugal [16] is presented in Figure 1. This plot shows the grid cells of the S2 library (brown borders) and the RSUs (colored dots) distributed in the city with their corresponding density. The data represents the situation of an example of Thursday afternoon from 12:00 to 18:00 pm. The density for a RSU is represented by the color of the marker. The density indicates the number of connections made with the RSU during the interval. One vehicle can connect multiple times during this period. In the next section we present the proposed predictor to estimate the urban traffic flow.

[1]http://s2geometry.io/

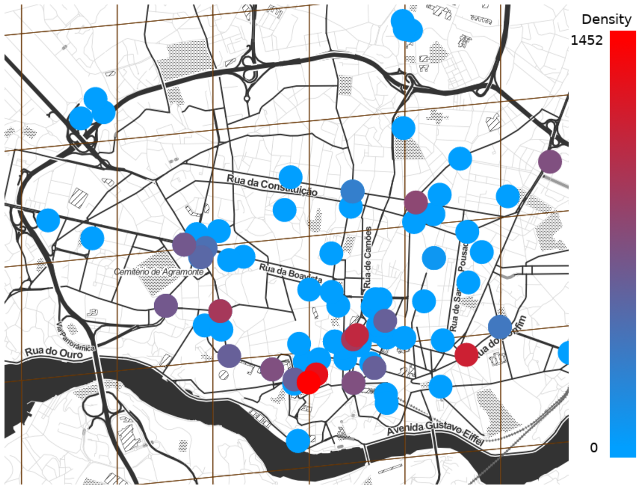


Fig. 1: RSUs with corresponding densities and S2 grid cells (brown borders) in the City centre of Porto, Portugal.

IV. System Model

In this section, we present the proposed MTL-LSTM model as an urban traffic flow estimator. The predictor is a deep learning-based multi-task learning (MTL) algorithm. MTL-LSTM employs multi-task learning to explore the Spatio-temporal correlations of traffic flows between neighboring trajectories and then predict the future number of moving objects in the trajectories. The proposed MTL-LSTM is presented in Figure 3. In the following subsections, we first introduce the concept of Dedicated and Shared LSTMs. We describe how to integrate those learning architectures into a unified Multi-Task Learning framework to estimate urban traffic flows. Note that, for simplicity, we only present a two grid cell scenario.

A. Single-Task, Dedicated and Shared LSTM

The RNN (Recurrent Neural Network) is mainly used for tasks that are involved with sequential inputs, such as time series prediction. The authors in [17] show that RNNs can not support long-term dependencies. Thus, an improved RNN called a Long Short Term Memory network (LSTM) is proposed, which uses special hidden units (i.e., memory cells) to remember inputs for a long time. LSTM models can learn long-term temporal sequences and make accurate predictions. As an LSTM naturally explores the temporal dependencies, here, we mainly focus on adopting an LSTM to capture spatial dependencies of traffic flow across neighboring trajectories. In this way, first, we introduce *Dedicated LSTM* and *Shared LSTM*, which can capture spatial dependencies of urban traffic flow among adjacent areas. Then, to improve the performance gains, we merge these two learning architectures to generate our MTL-LSTM model.

1) Single-Task LSTM: To use as a baseline for performance comparison, we consider a Single-Task LSTM. It predicts traffic for one grid cell using only data from this cell, not taking advantage of spatial dependencies. This prediction can be expressed as:

$$\hat{Y}^{i}_{t+1} = STL^{i}\left(F^{i}\right) \tag{2}$$

F^{i} represents collected traffic flows from trajectories in grid cell i used by the STL predictor to predict future traffic flow $\hat{Y}^{i}_{t+1}$.

2) Dedicated LSTM: In this prediction model, the traffic flow forecasting task for each grid cell, which includes specific trajectories, will be served by its Dedicated LSTM predictor. Dedicated means that each LSTM has only the purpose to predict traffic for one grid cell. Therefore, for every grid cell a LSTM exists. However, the full set of collected traffic from all grid cells is provided to each predictor for joint exploration of the spatial-dependencies of traffic flow between adjacent trajectories (see Figure 4-a). The prediction process for each Dedicated LSTM can be formulated as:

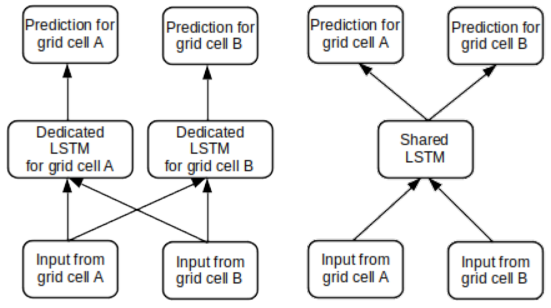
$$\hat{Y}^{i}_{t+1} = DL^{i}\left(C = \left[F^{1} \dots F^{j}\right]\right) \tag{3}$$

$\hat{Y}^{i}_{t+1}$, DL^{i} are predicted traffic flow and the Dedicated-LSTM predictor, respectively. C presents accumulated traffic flow from all j grid cells.

3) Shared LSTM: As shown in Figure 4-b, different from the previous architecture, in this model there is no Dedicated LSTM predictor. Instead, a Shared LSTM is adopted. Again, collected traffic flows from all grid cells are fed to this shared predictor to learn spatial and temporal dependencies for all trajectories at the same time. But in contrary to the dedicated LSTM, the shared architecture uses one LSTM to predict traffic for all grid cells. The forecasting process for the Shared LSTM can be presented as:

$$\hat{Y}^{i}_{t+1} = SL\left(C = \left[F^{1} \dots F^{j}\right]\right) \tag{4}$$

$\hat{Y}^{i}_{t+1}$, depicts the predicted traffic flow for all trajectories in the city areas. SL is the Shared-LSTM.


Prediction for grid cell A
Prediction for grid cell B
Dedicated LSTM for grid cell A
Dedicated LSTM for grid cell B
Input from grid cell A
Input from grid cell B
Prediction for grid cell A
Prediction for grid cell B
Shared LSTM
Input from grid cell A
Input from grid cell B

(a) Dedicated LSTM (b) Shared LSTM

Fig. 2: $LSTM$ architectures for spatio-temporal traffic forecasting.

B. Multi Task Learning LSTM

The introduced Dedicated and Shared LSTM architectures attempt to capture spatio-temporal features of urban traffic flows among neighboring trajectories. However, these predictors still have a simple architecture, which implies that correlations between trajectories can not be explored effectively. In order to fully capture the pure spatial and temporal information of the urban traffic among adjacent trajectories, we introduce the MTL (Multi-Task Learning) based architecture, which combines the Shared and Dedicated LSTM architectures to improve prediction performances (e.g., accuracy, training time, etc.) (see Figure 3). Again, every LSTM uses information from all grid cells. Additionally, the prediction for each grid cell combines a shared and dedicated LSTM.

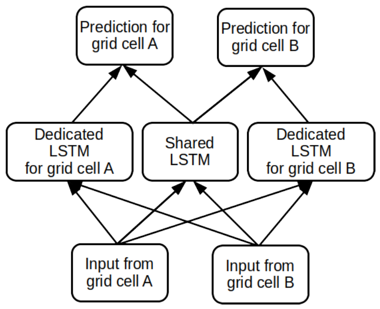

Prediction for grid cell A
Prediction for grid cell B
Dedicated LSTM for grid cell A
Shared LSTM
Dedicated LSTM for grid cell B
Input from grid cell A
Input from grid cell B

Fig. 3: The system architecture of MTL-LSTM designed for traffic flow estimation using dedicated and shared LSTM.

Basically, given the learning tasks (e.g., traffic flow estimation in urban trajectories), where all the tasks or a subset of them are related, MTL aims to improve the learning of each prediction task by using the knowledge contained in all or some of the tasks. The prediction function for the proposed MTL-LSTM can be expressed as follows:

$$\hat{Y}^i_{t+1} = \left[DL^i\left(C = \left[F^1 \dots F^j\right]\right), SL\left(C\right)\right] \tag{5}$$

$\hat{Y}^i_{t+1}$ represents the predicted traffic flow, F^i and $C = \left[F^1 \dots F^j\right]$ denote the collected traffic flow information from a single trajectory and from all trajectories in the city of Porto, respectively.

C. Feature Engineering

The success of machine learning predictions depends heavily on the choice of features [18]. In this subsection, the process of extracting features from raw data is explained.

Mobility traces collected from moving objects as vehicles consist of spatio-temporal points. The spatio-temporal points are tuples with GPS coordinates (latitude and longitude) and time of presence ($RSU\ ID$, $timestamp$). They represent to which RSUs (Road Site Units) and OBU (On-Board Unit) in the City of Porto were connected at a certain time. For each RSU, the coordinates are known. The mobility traces of OBUs include the $GPS\ coordinates\ of\ connected\ RSU$ and their $RSU\ ID$ as well as the $timestamp$. Those sequences of connected RSUs for OBUs are our used trajectories. The sampling frequency of the mobility traces is an average of 10 seconds.

With the S2 library, every RSU can be efficiently assigned to a unique grid cell. We define traffic density in a grid cell as the number of connected OBUs to an RSU, located inside the corresponding grid cell, during an interval of 30 minutes. If an OBU leaves a cell and returns during the interval, it gets counted multiple times. We filter out days where some grid cells have no collected traffic data.

Based on the city's trajectories, we predict traffic density inside a grid cell for the next 30 minutes. We decided to approach this as a classification problem and defined four categories: $none$, low, $medium$ and $high$ density. The $none$ category stands for no present OBUs, e.g., in the early morning (03:00 - 04:00 am). The three remaining categories have their borders chosen in a way that all of them have the same count of elements. We incorporate temporal information $time\ in\ a\ day$, $weekday$, $month$, and $density$ as input features.

V. Performance Analysis

To examine the prediction performance of the proposed MTL-LSTM traffic flow estimator, we experiment it on a large scale VANET dataset. Section V-A describes the dataset and experiments details. The results are presented in section V-B.

A. Experiment Setup

This work considers real vehicle traces collected from the VANET testbed deployed in Porto's city from October 2016 until August 2017 [16]. This urban-scale testbed consists of 600+ networked vehicles and 70+ RSUs scattered along the city.

To evaluate the success of our urban traffic flow estimator, we rely on the prediction accuracy. We trained the RL-LSTM for 250 epochs, where each epoch indicates the number of iterations over the entire dataset. We set the learning rate to 0.1, which indicates the amount that the weights are updating during training.

The basic STL-LSTM consists of an input layer, two hidden LSTM layers of size 15 and activation function $tanh$, two dropout layers, and a dense layer. For the MTL, the functional API of Keras [19] is used with concatenation layers for merging. The built-in Adam optimizer [20] is used. The loss function for the classification problem is "categorical_crossentropy."

1) Test and train split: A good cross-validation scheme emulates the test distribution well; it has to cope with future unseen data. Cross-validation divides data into n parts with equal number of elements. Then, $n - 1$ parts are used to train the predictor, and the remaining part is used to test the algorithm. Most general cross-validation techniques as K-Fold are not correct when it comes to time-series data/problems. The methods are based on shuffling the data-set. This has to be avoided with time-series problems because it means predicting the past with data from the future. Ignoring the order of time-series can lead to inaccurate validation scores. Most cross-validation techniques do not ensure the same coherence in the validation folds compared to the actual test set. There is also a risk of overfitting on parts of the data set. Overfitting is the phenomenon of fitting the Neural Network (NN) to the training data and failing to fit on additional data or future observations. It means that the NN fails to find a general predictive rule. We want to maximize the prediction accuracy on new data points and not necessarily on the training data [21].

To take time-series into account, we use walk-forward cross-validation or expanding window cross-validation. With this test/train, split data remains in chronological order. The training starts on a small set of batches and is tested against the next upcoming batch. In the next iteration, the batch used for testing is added to the training set. With this split test, data can be reused for training; the training data is expanding in each fold [22].

2) Frameworks and hardware: We use the Keras library [19], a deep learning API, written in Python, running on top of TensorFlow. TensorFlow is an open-source software library for numerical computation using data flow graphs. Nodes in the graph are standing for mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) that flow between them [23].

The predictors are trained and evaluated on a High-Performance Computing Cluster at the University of Bern in Switzerland (HPC Cluster - UBELIX [2]) with Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz.

B. Evaluation Results

In this subsection, we examine the performance of the traffic flow prediction considering the proposed MTL-LSTM. We conducted detailed experiments to compare the performance of MTL-LSTM with Shared-LSTM, Dedicated LSTM, and STL-LSTM. STL-LSTM is a pure temporal model, where the traffic flow in each grid cell is predicted using its own LSTM. This model is mainly used as a benchmark [24].
Figure 5 presents the computed urban traffic flow prediction accuracy across the city for business days (Monday to Friday) and weekends (Saturday and Sunday). The accuracy measures if the right category (level of traffic) is predicted (see IV-C). The results show the efficiency of MTL-LSTM, which outperforms other predictors for both business days and weekends. MTL-LSTM can deliver an average prediction accuracy of 73% and 71% for business days and weekends, respectively. Among the predictors in Figure 5, the pure temporal model (STL-LSTM) performs worst. Another important observation is that the Shared-LSTM outperforms the Dedicated-LSTM. Shared-LSTM and MTL-LSTM are fitted to other grid cells and can therefore take advantage of spatial dependencies. This shows that, instead of training an LSTM for each grid cell, attempting to estimate the traffic flow for all nearby grid cells could provide us with even better results. Moreover, considering the availability of data collected over a longer period of time (multiple years), seasonal differences could be discovered and exploited.

In addition to prediction accuracy, we also measure the training time for each traffic flow predictor. The results can be found in Figure 4. In general, the training time refers to the time duration that a learning algorithm spends to learn features and discover patterns in the training dataset. Shared-LSTM and MTL-LSTM benefit from gaining knowledge of other grid cells, decreasing the time consumption for updating weights compared to Dedicated-LSTM and STL-LSTM.

[2]https://docs.id.unibe.ch/ubelix

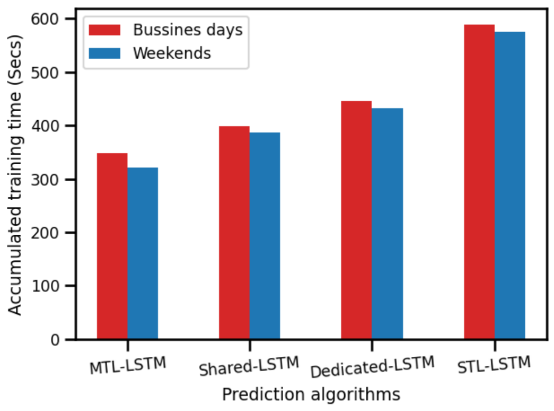


Fig. 4: Total training time of traffic flow predictors for business days and weekends.

The obtained results, as shown in Figure 4, show that the MTL-LSTM can learn the spatio-temporal features and discover the movement patterns of the vehicles faster than other algorithms. In total, MTL-LSTM spends 350 seconds and 320 seconds to train collected traffic data during business days and weekends, respectively. STL-LSTM is the one requiring more training time compared to the others. These observations conclude that, by feeding the MTL-based predictor using collected traffic flow from multiple neighboring grid cells simultaneously, the prediction process will be accelerated.

VI. Conclusions

This paper proposed the MTL-LSTM model to estimate the traffic flow of vehicles in urban trajectories. The proposed predictor attempts to explore both temporal and spatial dependencies of traffic patterns among neighboring trajectories to improve traffic flow prediction accuracy and improve speed. Based on a real-world and large-scale dataset, we provided a detailed evaluation of different learning models. We showed that the spatial information of urban traffic among nearby trajectories can indeed provide valuable information to improve prediction accuracy and reduce time consumption.

Future work will use this approach to provide information to the city management platform, and advise for possible problems and required interventions in the city, including the effects of traffic changes in the neighboring parts of the city.

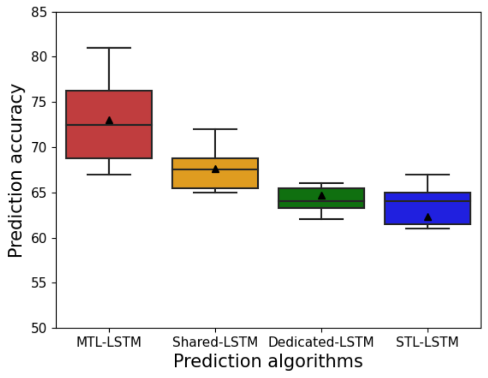
References

[1] M. Bando, K. Hasebe, A. Nakayama, A. Shibata, and Y. Sugiyama, "Dynamical model of traffic congestion and numerical simulation," *Phys. Rev. E*, vol. 51, pp. 1035–1042, Feb 1995. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.51.1035

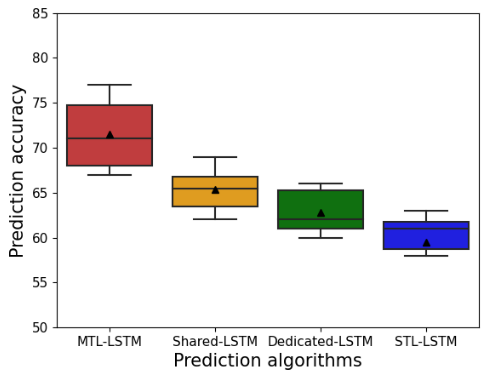
[2] J. Qiu, L. Du, D. Zhang, S. Su, and Z. Tian, "Nei-tte: Intelligent traffic time estimation based on fine-grained time derivation of road segments for smart city," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2659–2666, 2020.

[3] T. Crawford, "Scale analytical," in *International Encyclopedia of Human Geography*, R. Kitchin and N. Thrift, Eds. Oxford: Elsevier, 2009, pp. 29 – 36. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780080449104003990

[4] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Graph convolutional recurrent neural network: Data-driven traffic forecasting," *CoRR*, vol. abs/1707.01926, 2017. [Online]. Available: http://arxiv.org/abs/1707.01926



(a) Business days



(b) Weekends

Fig. 5: Traffic flow prediction results.

[5] Y. Zhang and Q. Yang, “A survey on multi-task learning,” *CoRR*, vol. abs/1707.08114, 2017. [Online]. Available: http://arxiv.org/abs/1707.08114

[6] S. Ruder, “An overview of multi-task learning in deep neural networks,” *CoRR*, vol. abs/1706.05098, 2017. [Online]. Available: http://arxiv.org/abs/1706.05098

[7] s. m. ahmed and r. a. cook, “Analysis of freeway traffic time-series data by using box-jenkins techniques,” *Transportation Research Record*, 1979.

[8] B. Williams and L. Hoel, “Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results,” *Journal of Transportation Engineering*, vol. 129, pp. 664–672, 11 2003.

[9] M. van der Voort, M. Dougherty, M. Dougherty, and S. Watson, “Combining kohonen maps with arima time series models to forecast traffic flow,” *Transportation research. Part C: Emerging technologies*, vol. 4, no. 5, pp. 307–318, 1996.

[10] Z. Zhu, B. Peng, C. Xiong, and L. Zhang, “Short-term traffic flow prediction with linear conditional gaussian bayesian network,” *Journal of advanced transportation*, 06 2016.

[11] J. Ou, J. Xia, Y.-J. Wu, and W. Rao, “Short-term traffic flow forecasting for urban roads using data-driven feature selection strategy and bias-corrected random forests,” *Transportation Research Record*, vol. 2645, no. 1, pp. 157–167, 2017. [Online]. Available: https://doi.org/10.3141/2645-17

[12] M. Lippi, M. Bertini, and P. Frasconi, “Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 871–882, 2013.

[13] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Wang, “Traffic flow prediction with big data: A deep learning approach,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.

[14] W. Yuankai, H. Tan, B. Ran, and Z. Jiang, “A hybrid deep learning based traffic flow prediction method and its understanding,” *Transportation Research Part C: Emerging Technologies*, vol. 90, 05 2018.

[15] F. Jin and S. Sun, “Neural network multitask learning for traffic flow forecasting,” *CoRR*, vol. abs/1712.08862, 2017. [Online]. Available: http://arxiv.org/abs/1712.08862

[16] P. Santos, J. Rodrigues, T. Lourenço, P. D’Orey, Y. Luís, S. Sargento, A. Aguiar, and J. Barros, “Portolivinglab: an iot-based sensing platform for smart cities,” *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 523–532, April 2018.

[17] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature Cell Biology*, vol. 521, no. 7553, pp. 436–444, May 2015.

[18] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[19] F. Chollet *et al.*, “Keras,” https://keras.io, 2015.

[20] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014.

[21] T. Dietterich, “Overfitting and undercomputing in machine learning,” *ACM computing surveys (CSUR)*, vol. 27, no. 3, pp. 326–327, 1995.

[22] germayne. (2019) Time series cross-validation — a walk forward approach in python. [Online]. Available: https://medium.com/eatpredlove/time-series-cross-validation-a-walk-forward-approach-in-python-8534dd1db51a

[23] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[24] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.