Reinforcement Learning-designed LSTM for Trajectory and Traffic Flow Prediction

Mostafa Karimzadeh, Ryan Aebi, Allan M. de Souza, Zhongliang Zhao, Torsten Braun, Susana Sargento, Leandro Villas
Institute of Computer Science, University of Bern, Switzerland
Institute of Computing, University of Campinas, Brazil
Institute de Telecomunicações - Aveiro, Portugal
Email : {mostafa.karimzadeh, zhongliang.zhao, torsten.braun}@inf.unibe.ch, ryan.aebi@students.unibe.ch,
{allanms, leandro}@lrc.ic.unicamp.br, susana@ua.pt

***Abstract*—Trajectory and traffic flow prediction will play an essential role in Intelligent Transportation Systems (ITS) to enable a whole new set of applications ranging from traffic management to infotainment applications. 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 spatiotemporal dependencies. However, these neural networks tend to be over-complex and hard to design due to the broad set of hyper-parameters. We propose an automated framework to predict future trajectories and traffic flows in urban areas without human interventions. We employ Reinforcement Learning (RL) and Transfer Learning (TL) to generate high-performance LSTM predictors, which is referred as RL-LSTM. In addition, we introduce HERITOR (*H*igh ord*E*r t*R*aff*I*c convolu*T*i*O*n *R*l-lstm), a novel deep learning algorithm for traffic flow prediction. Specifically, HERITOR attempts to capture pure spatiotemporal features of urban traffic. The extracted features are fed into the RL-LSTM to realize a high performance LSTM for traffic flow prediction. We examine the proposed trajectory and traffic flow predictors on two real-world, large-scale datasets and observe consistent improvements of 15% - 25% over the state-of-the-art. By using transferred knowledge, we can accelerate the process of searching an optimal architecture of an LSTM by up to 70%.**

***Index Terms*—Trajectory prediction, Traffic Flow prediction, Reinforcement learning, Knowledge transferring, Graph convolution, LSTM.**

I. Introduction

Intelligent Transportation Systems (ITS) will enable more efficient, safer, and greener traffic mobility, which will pave the way to a whole new set of services that will change the way that we live, work, and play [1]. Trajectory and traffic flow prediction will play an important role to improve traffic management decisions, communication protocols, and infotainment applications [1]. A trajectory predictor attempts to estimate the path that a moving object is going to take to travel from one location to another one. The goal of traffic flow predictor is to estimate the number of moving objects in urban areas given historic mobility trace and the underlying trajectories in a city. Vehicular networking is one of the ITS foundations and also will take advantage of trajectory and traffic flow predictions to improve information retrieval, data dissemination, and resource allocations. For instance, knowing in advance the number of vehicles that will be in a region in the next minutes can not only reduce latency in infotainment applications (e.g., multimedia applications, video streaming, etc), but also provide better resource allocation for multi-access edge computing (MEC) services such as virtual machines, bandwidth and etc.

However, due to the spatiotemporal dependencies of the urban environment and the time-varying traffic patterns, predicting the traffic hotspots (e.g., areas with high traffic, congested areas, etc.), the future trajectory of moving objects, and also the traffic flow between to predicted locations are challenging tasks.

Deep learning-based approaches such as Recurrent Neural Networks (RNN) and its variant Long Short Term Memory (LSTM) have excellent performance in traffic prediction due to their ability to learn temporal dependencies [2]. On the other hand, defining a high-performance architecture for neural networks still is a hard task due to the broad set of hyper-parameters [2], [3]. Typically, hyper-parameters are the variables which determine the structure of neural networks (e.g., number of hidden layers).

In this work, we propose an adaptive framework to predict future trajectories and the urban traffic flow between two locations based on historical data. To optimally capture the spatiotemporal dependencies of an urban environment, we used a graph-structured convolution approach, which can learn the interactions between locations and also support better traffic flow forecasting [2]. On the other hand, to remove the human interventions in the neural network architecture design we used recently proposed model [3], which employs a Reinforcement Learning (RL) based method to find the most suitable architecture for a given dataset. Basically, the RL provides a self-learning approach by rewarding the *high-performance architectures* and punishing the *low-performance ones*. In this way, the goal is to change the architecture of a neural network (e.g., the number of hidden layers, the number of neurons in each hidden layer) based on its previous performance (e.g., the reward) towards a most suitable architecture for a given dataset. Nevertheless, due to the high number of possible neural network architectures, the time to find an architecture with the desired performance may be a concern. Thus, Transfer Learning (TL) is used to reduce this time, the idea is transferring knowledge from a trained predictor to a newly suggested predictor. This will help the new algorithm to pass through training phase faster. Moreover, to identify the spatiotemporal traffic hotspots named as Zone of Staying (ZoS), a tuning-free method is also presented. The main contributions of this work can be stated as follows:

- We study a tuning-free approach for eliciting ZoSs of moving objects from spatiotemporal trajectories without any *a-priori* assumptions.
- We design a trajectory predictor that benefits from Reinforcement Learning and transferable knowledge for searching the best architectural description of an LSTM predictor.
- We present HERITOR, a deep learning technique to analyze graph-structured data, to extract and predict complex spatiotemporal dependencies of traffic flows in road networks of urban areas.
- Quantitative experiments on two real-world datasets demonstrate that the proposed trajectory and traffic flow predictors deliver consistently satisfactory results.

The rest of this paper is organized as follows. Section II reviews related work. Section III discusses the problem statement. Section IV introduces the proposed method to discover ZoSs. Section V describes the trajectory predictor using Reinforcement Learning and LSTM, while Section VI describes the urban traffic flow predictor. The evaluation methodology and the performance analysis are presented in Sections VII and VIII, respectively. Finally, Section IX concludes the paper.

II. Related Work

A. Zone of Staying Discovery

Discovering Zones of Staying (ZoS), is an essential component for making cities smart, particularly with regards to traffic management, early congestion warning, mobile network resource allocation, etc. A ZoS refers to a city area, where a moving object (e.g., vehicles, pedestrian) has frequent visits and stays for a considerable amount of time. The initial study [4] uses an iterative clustering approach to extract ZoSs. Clustering is a task in data mining for grouping similar objects into a set known as a cluster. Other clustering-based models, including temporal clustering [5], hybrid clustering [6] [7], k-shape and k-multi-shape clustering [8] have been attempted to discover ZoSs in urban environments. All of the introduced techniques critically rely on some spatial (e.g., acceleration alteration) and temporal (e.g., distance measure) parameters to detect hot-spots. Moving objects are specified by distinct mobility patterns, which leads to having different optimal values of these parameters. Therefore, in this work, we define a novel technique by benefiting from signal processing approaches, which omit the affiliation on the spatial and temporal parameters to detect ZoSs in urban areas.

B. Trajectory Prediction

Future trajectory prediction helps drivers and pedestrians to have safe and efficient navigation through complex traffic scenarios. The pioneering survey on trajectory prediction [9] classified trajectory predictors into two categories: physics-based and maneuver-based models. Physics-based prediction in crowded scenes takes into account the mutual interaction (e.g., traffic rules, road geometry) between surrounding moving objects to estimate short term trajectories [10], [11]. [12] employed Kalman Filter (KF) to estimate future trajectories. The maneuver-based approach could estimate longer trajectories for the moving objects [13]. Heuristic-based classifiers [14], Markov models [15] [16] and random forest classifiers [17] are maneuver- based approaches, which could predict long trajectories in urban areas. Since future trajectories could be estimated through time-series mobility data, a number of Recurrent Neural Networks (RNNs) and their variants including Gated Recurrent Units (GRU) [18] and Long Short Term Memory (LSTM) [19] have been proven to be very effective for trajectory prediction task. [20], [21] use LSTMs to predict motion of human drivers in a grid map. Authors in [22] propose a *social* LSTM, which predicts the trajectory of pedestrians in crowded spaces through the use of a social pooling layer. Despite the success of the introduced trajectory predictors, describing the architecture of neural networks is an effortful task. We automate the process of developing a high-performance LSTM based trajectory predictor without human intervention.

C. Traffic Flow Prediction

Traffic flow prediction is a crucial task in ITS. Investigations on traffic flow prediction most fall into two main categories [2]: statistical methods and machine learning methods. Statistical methods [23], [24] were developed years ago when traffic systems were less complex in the terms of the number of moving objects and the rate of mobility. The ability of such traditional models to accurately estimate future traffic states is quite limited. In recent years, researchers shifted their attention to the deep learning models, which are more robust and accurate in dynamic and complex urban areas. Deep Belief Networks (DBN) [25], Deep Recurrent Neural Networks (DRNN) [26] and Convolutional Neural Networks (CNN) [27] can effectively learn features of time series data and achieve good prediction performance. [28] introduces the first Graph Convolutional Neural network (GCN), which integrates spectral graph theory with deep neural networks. ChebNet proposed in [29] to improve GCN using fast localized convolutional filters. Authors in [30] developed the idea of the graph Laplacian matrix, which operates on the graph spectrum. [31], [2], [32] propose Diffusion-Convolutional Neural Networks (DCNN) to define convolution as a diffusion process in each vertex of a graph-structured input. The main shortcoming of the mentioned research works is that they do not investigate the effect of neighboring base stations (e.g., RSUs and mobile antennas) and length of trajectories among base stations on the estimated traffic flow. In this work, we introduced an algorithm that employs a high-order convolution operator and adaptive distance adjacency matrix to capture spatiotemporal dependencies of traffic flow in city areas. Besides, we benefit from RL and TL to generate the best possible LSTM architecture to make the traffic flow prediction.

III. Problem Statement

Our work addresses problems of future trajectory prediction and traffic flow prediction in urban areas, which are the core components of Intelligent Transportation Systems (ITS). Hereafter, we split the main problem statement into three sub-problems as described below.

Problem 1: Zone of Staying (ZoS) Discovery

We propose a tuning-free technique for detecting hotspots from spatiotemporal trajectories without any *a-priori* assumption (e.g., constraints on distance, speed of movement, duration of staying, number of collected Global Positioning System (GPS) point, etc.). We eliminate parameter dependence by treating spatiotemporal trajectories as space-time signals and apply signal processing algorithms to discover ZoSs for moving objects (e.g., vehicles and Pedestrians).

• *Requirements and Challenges.*

(i) extracting a moving object's trajectory as an ordered set of visited locations $Tr(lat_n, lon_n, t_n) = [(lat_1, lon_1, t_1), \ldots, (lat_n, lon_n, t_n)]$, where lat_n and lon_n are GPS coordinates and t_n denotes timestamp of the visited location point. *(ii)* transforming extracted trajectory T into a 2D signal $S(t)$. *(iii)* interpreting the space-time signal $S(t)$ in time and frequency domain to detect ZoSs.

Problem 2: Trajectory Prediction with LSTM

Recurrent Neural Networks (e.g., LSTM) are powerful and flexible models that work well for trajectory prediction in city areas. Despite their success, designing an architecture for the LSTM based predictors requires both human expertise and effort. In this research, we study a method to generate a high-performance LSTM for the given learning task automatically.

• *Requirements and Challenges.*

(i) defining a learning agent to suggest the architecture description for the LSTM to have a satisfying accuracy on a validation dataset. *(ii)* transferring the knowledge from a previous model to the new suggested architecture to speed up the experimentation process.

Problem 3: Traffic Flow Prediction

In this research a traffic flow predictor attempts to estimate future traffic states, in terms of the number of moving objects in the trajectories. The trajectories, base stations (e.g., RSUs and mobile antennas), and the collected traffic flow of a city can be represented by a directed graph $G = (V, E, A, W)$, where V is a set of nodes $|V| = N$ (*base stations*), E is a set of ordered pairs of edges (*trajectories*). Directional adjacency matrix presented by $A \in \mathbb{R}^{N\times N}$, in which each element $A_{i,j} = 1$ if there is a path connecting base station i and base station j, otherwise $A_{i,j} = 0$. $W \in \mathbb{R}^{N\times N}$ is a distance adjacency matrix representing base stations mutual influence as a function of their real road distance. The traffic flow collected in a city is shown as $F = \left[f^t \ldots f^{t+T}\right]$, where $f^t \in \mathbb{R}^{N\times P}$ is a set representing number of connected users (P) for each base station (N) 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', given a traffic graph G:

$$L\left[\left(f^{(t)}, \cdots, f^{(t+T)}; G = (V, E, A, W)\right)\right] \equiv \left(f^{(t+1+T)}, \cdots, f^{T'}\right) \tag{1}$$

• *Requirements and Challenges.*

(i) modeling spatiotemporal dependencies of traffic flow, indicating explicitly where and when the traffic happens.*(ii)* learning the impact of traffic flows among adjacent trajectories and neighboring base stations.

IV. From Spatiotemporal Trajectories to ZoSs

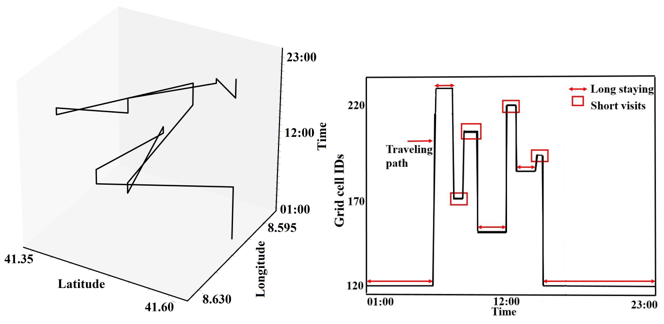
In this section we address *Problem 1.* The Solution includes translating spatiotemporal GPS trajectories into two-dimensional signals and interpreting generated signal in time and frequency time to detect ZoSs for every single moving object.

A. Trajectory Extraction

A trajectory is an observed path of a moving object when it travels from one ZoS to another one. Both pedestrians and vehicles can take multiple routes to move among different locations. To discover trajectories, we explore two rich real-world datasets (see Section VII). To extract trajectories for each moving object, we need the list of sequentially connected base stations (e.g., RSUs, antennas) and GPS coordinates of each connected station. We use the Google Maps API Direction Service[1] to discover all possible routes between two consecutive base stations. The discovered trajectory per each individual moving object between ZoS_i and ZoS_j is stored as a 3D signal $T(lat_n, lon_n, t_n) = [(lat_1, lon_1, t_1), \ldots, (lat_n, lon_n, t_n)]$. After discovering the paths the next step is to partition the trajectories into grid cells, for which we use the Python Google S2 Geometry Library [2]. Each grid cell is a four-corner cell, which covers a specific region. Each observed path t_k is partitioned into a sub-list of grid cells. In this work, the coverage area of each grid cell is set to be 300 m^2. The resulting partitioned path can be shown as a 2D signal $r(c_l, t_l) = \{(c_1, t_1), (c_2, t_2), \ldots, (c_l, t_l)\}$, where c_l is the grid cell ID and t_l is the time stamp of visiting the grid cell. Figure 1(a) visualizes a moving object's trajectory as a 3D trajectory and the transformation to a 2D signal is presented in Figure 1(b). In the next subsection, we will analyze the 2D signal in time and frequency domain to extract ZoSs for each single moving object.

1) Time Domain Analysis: Interprets the signal concerning time. As shown in Figure 1(b), a 2D signal in time domain reveals two main features of a moving object's mobility pastern: *(i)* staying at locations, *(ii)* traveling along paths. The local maxima/minima of the signal are interpreted as staying locations of a moving object. A set of distinct staying locations can be discovered by selecting the maxima/minima with distinct cell IDs. The staying location areas, where the

[1]https://cloud.google.com/maps-platform/
[2]http://s2geometry.io/



(a) Visualizing one day trajectory as a 3D space-time signal (b) Visualizing one day trajectory as a 2D space-time signal

Fig. 1. Visualizing the OBU's movements as space-time signal.

moving objects spend a long duration of time, can directly assume to be their ZoSs (e.g., home, workplace and congested city center, etc.). Moreover, travel paths depict the routes that moving objects take to travel from one staying location to another one. The rest of the detected locations have a shorter staying time duration. Therefore, to be able to transform them as a ZoS, it is necessary to know how often a user visits these staying locations. This information can be obtained by analyzing the signal in the frequency domain, which is the second step of ZoS detection.

2) Frequency Domain Analysis: A periodic signal $S(t)$ is typically represented as $S(t) = S(t+T)$ for all time stamps t, where T is the period of the signal. It represents the smallest duration of time that the signal needs to repeat itself. Analyzing a signal in frequency domain reveals visiting periodicity of each location by a moving object. High periodicities mean that the user visits the location frequently so that the location can be assumed as a ZoS. On the other hand, low periodicity shows that users visit the location infrequently. As explained in Equation 2, applying *Discrete Fourier Transform* (DFT) converts a signal from the space-time domain to a representation in the frequency domain.

$$P[l] = \sum_{l=0}^{L-1} r(c_l, t_l) e^{-jl2\pi/L} \tag{2}$$

$r(c_l, t_l)$ is the 2D trajectory of length L composed of grid cells (c_l). $P[l]$ is the computed visiting periodicity for grid cell c_l. From calculated periodicities for the grid cells, we select the first four dominant periodicities. To interpret the selected visiting periodicities, we applied the *Inverse Discrete Fourier Transform* (IDFT) to convert the signal back from the frequency domain to the time domain (see Equation 3).

$$ZoS(c_l) = (1/N) \sum_{n=0}^{N-1} P\prime[k] e^{jn2\pi/N} \tag{3}$$

$ZoS(c_l)$ denotes detected ZoSs including grid cell c_l and $P\prime[l]$ is the selected dominant periodicity. Figure 2 illustrates an example of discovered ZoSs for a vehicle in the city of Porto, which are represented by a set of rectangular grid cells.

V. Trajectory Predictor

In this section, we address *Problem 2*. We introduce RL-LSTM, a trajectory predictor based on Reinforcement Learning (RL) to automatically realize a high-performing LSTM

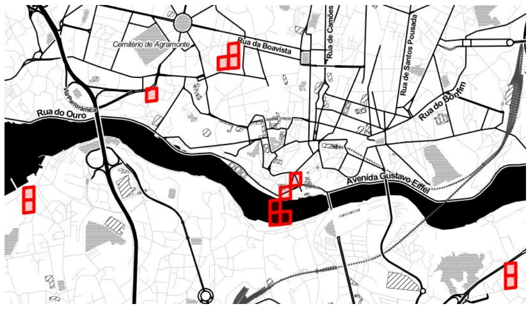


Fig. 2. Discovered ZoSs for OBU ID = 2599

predictor for a given learning task. Besides, to accelerate the architecture search process, we benefit from Transfer Learning (TL). Using TL the knowledge of the pre-trained architecture (*teacher LSTM*) to estimate the trajectory of a moving object is used as the starting point of the newly suggested architecture (*student LSTM*) for the same task. The leading search method that we use in this work is the Neural Architecture Search (NAS) framework [3]. In NAS, the RL-based controller generates architectures for the predictor. Basically, architecture of a neural network refers to the number of hidden layers, the number of neurons per each layer, and how they are connected. Then, the predictor is trained to make predictions on a validation dataset. The outputs of the algorithm are used to update the controller so that it will generate better architectures over time. Our proposed RL-LSTM has three main units: *(i)* Long Short Term Memory (LSTM) as a student predictor to grow up to get a satisfying accuracy for the prediction task *(ii)* Q-learning as the controller to propose better architectures for the student LSTM to maximize the expected prediction results and *(iii)* the Transfer Learning (TL) unit to accelerate the architecture search process. Details of each unit and how they are integrated to predict future trajectories are explained in the following subsections.

A. Long Short Term Memory (LSTM)

A special kind of Recurrent Neural Network (RNN) that can be applied to time series forecasting is the popular Long Short Term Memory (LSTM) [2]. This architecture represents only the most common implementation of the LSTM as a predictor. To have a highly accurate LSTM predictor the learning agent in Section V-B explores a search space, which includes: (i) *Action space* refers to a set of constraints that restrict the learning agent from taking certain actions. First, we allow the agent to terminate the iterations if the student LSTM can deliver a satisfying prediction accuracy (e.g., 90%). Otherwise, the process will terminate when the learning agent has explored the whole search space. Besides, we force the learning agent to have a dropout layer [33] after each hidden layer. (ii) *Parameter space* is defined as a set of all relevant layer parameters that the learning agent can take. The number of hidden layers is an integer value selected from $(0, 150]$. For each hidden layer, the number of neurons is chosen from $\{1, 5, 10, 20, 40, 60, 80, 100, 150, 200\}$ and the dropout ratio is chosen from $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. Additionally, we use Rectified Linear Unit (ReLU) as non-linearities [34] for each dense layer. Note that defining the *Action* and *Parameter* spaces must have a faster convergence because of limited hardware resources, and it is not a limitation of the method itself.

B. LSTM Architecture Design With Reinforcement Learning

In this subsection, we seek to automate the process of LSTM architecture selection through a searching procedure based on RL. We create a controller using Q-learning, that attempts to define high-performance architectural description of an LSTM that performs well to predict the future trajectory of moving objects without human intervention. As explained in Section V-A, by limiting the layer parameters (e.g., number of hidden layers, number of neurons in each hidden layer and dropout ratio) and actions to choose from, the controller has a finite but large space of possible architectures to search from. The controller as a learning agent trains through random exploration and slowly begins to exploit its finding to select higher-performance architectures employing the ϵ-greedy strategy [35]. The learning agent receives the computed accuracy for the estimated future trajectory as the reward. Based on the reward signal, the learning agent suggests better architectures over time. The whole procedure is shown on the right side of Figure 3. As explained, we benefit from the Q-learning approach as a learning agent to propose architectures for the *student LSTM*. We now summarize the theoretical formulation of Q-learning, as adapted to our problem. We construct an environment where an agent interacts with a discrete and finite *Parameter space* S, which includes a set of all relevant parameters (see Section V-A) that the learning agent is allowed to take. Moreover, we define *Action space* A, which refers to a set of all possible actions that the agent should consider (see Section V-A). At each iteration $t \in \{0, 1, 2, \ldots\}$, the agent in state $s \in S$ will take an action $a \subseteq A(s)$ to pass into next state s'. At each iteration t, computed accuracy is given to the agent as a reward signal ($r_t \in \mathbb{R}$), which depends on the transition from state s to the next state s'. The ultimate objective of the agent is to maximize the total cumulative reward over all possible iterations. Although we limit the agent to a finite search space, there is still a large number of possible architectures, which motivates the use of RL. We define the reward maximization problem recursively in terms of sub-problems as follows; for any state $s \in S$ and action $a \in A(s)$, we define the maximum total expected reward over all possible iterations to be $Q'(s, a)$. $Q'(\cdot)$ is named as the action-value function and individual $Q'(s, a)$ is known as Q-value. The recursive maximization equation, which is known as Bellman's Equation, can be written as:

$$Q'(s,a) = \mathbb{E}_{s'|s,a}\left(\mathbb{E}_{r|s,a,s'(r|s,a,s')+\gamma \max_{a' \in \mathcal{A}(s')} Q'(s',a')}\right) \quad (4)$$

In our problem, the learning agent does not know a priori what are the effects of each suggested architecture. The agent only knows what the set of possible parameters and actions are. In this case, the agent has to learn through the output of the suggested architectures. Therefore, we can write the Bellman's Equation as an iterative formula (see Equation 5):

$$Q_{t+1}(s,a) = (1-\alpha) Q_t(s,a) + \alpha \left(r_t + \gamma \max_{a' \in \mathcal{R}(s')} Q_t(s',a')\right) \quad (5)$$

$\alpha \in (0, 1]$ is the Q-learning rate, which determines the weight given to new information over old information and $\gamma \in (0, 1]$ is the discount factor, which determines the weight given to immediate rewards over future rewards. Q-learning is *off policy* RL, i,e. the learning agent could explore the environment randomly, and despite of this, it can find the optimal architecture for the *student LSTM*. In this research, we use the ϵ-greedy exploration/exploitation strategy. The learning agent begins to suggest a new architecture to the *student LSTM* and it tries some random architectures, which refers to the exploration phase. However, as soon as the agent gets better and suggests high-performance architectures to the LSTM, the agent starts to converge, which refers to the exploitation phase. With ϵ-greedy [35], the learning agent at each iteration suggests a random architecture including a set of possible parameters with probability ϵ, $0 \leq \epsilon \leq 1$. At the beginning of the architecture search, we start with $\epsilon = 1.0$ to ensure that the learning agent has enough time for the exploration phase and we slowly decay ϵ to 0.01 (and not to $\epsilon = 0$) to move toward the exploitation phase.

C. Accelerated Training with Transfer Learning

When training an LSTM, it is not very efficient to train every suggested architecture from scratch. Transfer Learning (TL) offers a solution to this, as it offers possibilities on how to transfer knowledge from a trained predictor (*teacher*), to a new predictor (*student*). Typically, this will help the *student LSTM* to pass through the learning phase faster.

If two LSTMs have a similar architecture (in terms of layers and connectivity) partly, we can call them *semi-homogeneous*. Now given that *teacher* and *student* LSTMs are *semi-homogeneous*, then we can transfer the knowledge from the *teacher* to the *student*. This is achieved by extracting the learned knowledge, which is saved as weights from the *teacher* predictor and initializing the new *student* predictor with those weights. Specifically, we use an adaptation of the Net2Net research [36], where the authors attempt to transfer knowledge from the pre-trained predictor at iteration $t-1$ to the new one at iteration t. In [36] , for a newly added hidden layer, it must have more hidden units than the previous hidden layer, otherwise initializing weights (transferring knowledge) from the previous hidden layer to the new hidden layer will not work.

As a solution for this deficiency, we introduce a new technique to transfer knowledge from the *teacher LSTM* to the *student LSTM*. Let the layers of the pre-trained LSTM be $L = \{l_1, l_2, \ldots, l_n\}$, where the layers l_1 and l_n represent input and output layers, respectively. Suppose that a new layer l'_i is proposed by the learning agent, and then implanted into the *student LSTM* between index $n-1$ and n before the output layer; thus its layers would be $\tilde{L} = \{\tilde{l}_1, \tilde{l}_2, \ldots, \tilde{l}_{n-1}, l'_i, \tilde{l}_n\}$. Then, we define the function $\upsilon(l_j) \in \mathbb{N}{>}\ 0$ to represent the number of hidden neurons for each layer l_j, where $1 \leq j \leq n$. Further, we define a weight function $\omega(l_j) \in \mathbb{R}^{n \times m}$, where $n, m \in \mathbb{N} > 0$, to form the weight matrix. Now, we can find hidden layers with the same number of units in each layer between L and $\tilde{L}$ by defining $L \bigcap \tilde{L} := \{l_i \mid v(l_i) = v(\tilde{l}_i)$ for $i = 1, \ldots, n-1$. So the first $n-1$ layers of both networks are found to be similar; thus we can transfer knowledge from l_i to $\tilde{l}_i$ for $i = 1, \ldots, n-1$. Further, the transfer learning can then be defined as applying $\omega(\tilde{l}_i) := \omega(l_i), \forall i = 1, \ldots, n-1$. This describes copying the first $n-1$ weights from the teacher to the student. With our proposed approach we can choose the number of hidden units n in l'_i freely, so $v(l'_i) = n$ for any $n \in \mathbb{N}$. Therefore, we could effectively add the layer l'_i to the *student* and then perform transfer knowledge on the layers l_j for $j = 1, \ldots, n-1$.

VI. Urban Traffic Flow Prediction

In this Section, we address *Problem 3*. We introduce HERITOR (*H*igh ord*E*r t*R*aff*I*c convolu*T*i*O*n *R*l-lstm), a novel deep learning algorithm to estimate future states of urban traffic flow in terms of the number of moving objects in the trajectories. HERITOR employs a high order convolution operator and an adaptive distance adjacency matrix to extract rich spatiotemporal features of urban traffics. Then, the RL-LSTM is fed by the extracted features to generate the best possible LSTM to predict traffic flows. Details of the proposed urban traffic flow predictor are described in the following subsections.

A. High Order Traffic Graph Convolution Operator

The convolution operator at a specific node v_j in graph G can be generally expressed as:

$$Convolution(j) = \sum_{i \in N_j} w_{ij} f^i \tag{6}$$

$f^t \in F$ is the extracted feature for node v_j, $w_{i,j}$ is the weight and N_j is the set of nodes that are adjacent to v_j. In this section we introduce the high order convolution operator in the graph. The high order (k-th order) neighborhood can be defined as $N_j = \{v_i \in V | d(v_i, v_j) \leq k\}$ for node v_j. The 1-hop neighborhood matrix for a graph G is exactly the adjacency matrix. Then, the k-hop adjacency matrix of n stations can be obtained by calculating the k-th product of $A \in \mathbb{R}^{N \times N}$. Therefore, we can define the k-th order convolution operator for a specific time interval t of the traffic graph as follows:

$$\widetilde{L}^{(k,t)}_{Convolution} = \left(W_k \odot \widetilde{A}^k\right) f^t \tag{7}$$

Here, $\odot$ refers to element-wise matrix product. $\widetilde{A}^k$ is obtained by adding identity matrix I to the k-hop directional adjacency matrix A^k that creates a self-loop for each node to make them self-accessible in the graph. This means that the moving object can stay connected to the base stations. Otherwise, they are forced to make a transition in each time step. $f^t \in F$ is the feature matrix, to show the number of connected users to each station at a specific time interval t. $W \in \mathbb{R}^{N \times N}$ is a directional distance adjacency matrix computed based on the real distance among stations in the traffic network. The k-order convolution operator for the time interval t ($\widetilde{L}^{(k,t)}_{Convolution}$) takes the k-hop directional adjacency matrix, k-hop directional distance adjacency matrix, and the feature matrix as the input. Its output is the weighted average of the feature matrix with the same dimension as f^t.

The introduced k-order convolutional operator in Equation 7 takes into account only the number of hops around each base station that fails to capture pure spatiotemporal features of traffic flows. In urban areas, nearby base stations are more related than distant base stations. Following this idea, we aim to give large weights to the short trajectories between nodes. Therefore, we propose the adaptive directional distance adjacency matrix in Equation 8. More specifically, we use the real length of trajectories to assign the weight between two base stations. So, closer nodes will be linked with higher weights.

$$\widetilde{W}_k = \sigma\left(\widetilde{A}^k f^t\right) \tag{8}$$

$\widetilde{A}^k$ and f^t are k-hop directional adjacency matrix and feature matrix, respectively. Sigmoid non-linearity is applied to map elements of the $\widetilde{W}_k$ into a range between $[-1, 1]$. By adding the adaptive directional distance adjacency matrix (Equation 8) into the high order convolution operator, we introduce our convolution operator in Equation 9.

$$\widetilde{L}^{(k,t)} = \left(\widetilde{W}_k \circ \widetilde{A}^k\right) f^t \tag{9}$$

Extracted information about traffic flow in urban road network by the graph convolution within k-hops adjacent nodes with respect to time t are concatenated together as follows:

$$\widetilde{L}^{(k,t)}_{Total} = \left[\tilde{L}^{(1,t)}, \ldots, \widetilde{L}^{(k,t)}\right] \tag{10}$$

$\widetilde{L}^{(k,t)}_{Total}$ is a set of k-order traffic graph convolutional feature, that can be fed to the predictor described in the following subsection.

B. High Order Traffic Convolution RL-LSTM (HERITOR)

The proposed traffic flow predictor on a directional graph is a holistic approach that aims to capture patterns of traffic dynamics in urban areas by taking both node features and graph connections into account. The extracted features by incorporating both k-hop convolution operator and adaptive directional distance adjacency matrix are fed into the predictor. We leverage the LSTM predictor to estimate the future spatiotemporal state of urban traffic. In particular, to design the most efficient architecture for the LSTM, similar to Section V-B, we applied Q-learning as a controller to suggest architectures to the *student LSTM*. The model architecture of HERITOR is illustrated in Figure 3. The RL-LSTM as a component of the model is shown on the right side and the high order traffic convolution operator is the left side, where k-hop convolution orders for each time interval t are represented with respect to a red node. HERITOR can learn and predict spatiotemporal dependencies in directional graph-structured data for various forecasting problems (e.g., pedestrians and vehicles)

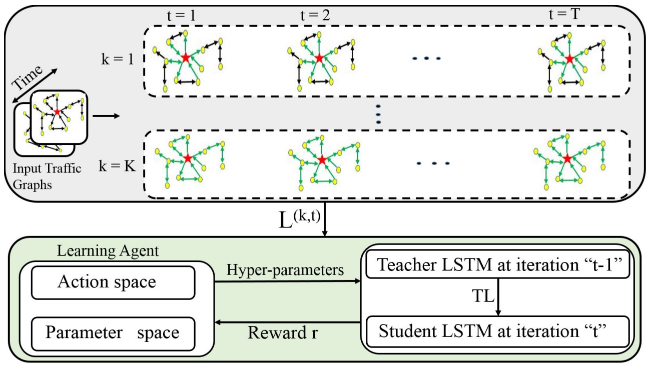


Fig. 3. The system architecture of HERITOR designed for spatiotemporal traffic flow forecasting in urban areas.

VII. Evaluation

In this section, we present an evaluation methodology to validate the proposed moving object trajectory predictor and urban traffic flow estimator.

A. Dataset

We conduct experiments on two rich real-world datasets: (i) *MDC dataset:* This dataset includes reach context information from the smartphones of around $100+$ users connected to 500 mobile antennas around the Lake Geneva region in Switzerland from October 2009 to March 2011 [37]. (ii) *Aveiro dataset:* This dataset includes real vehicle traces collected from the VANET testbed deployed in the city of Porto in Portugal from October 2016 until August 2017. This urban-scale testbed consists of $100+$ networked vehicles connected to the infrastructure through 120 RSUs [38]. In both of these datasets, we select mobility traces of 100 users, which include connected base station IDs, GPS coordinates of each base station, and the time stamps of the connections. Using this information, the introduced estimators can discover movement patterns of the users and predict the future behaviors of them.

B. Evaluation Metrics

To interpret the prediction success of the proposed RL-LSTM algorithm as a trajectory predictor, we use F_1-Score (11), which is the harmonic mean of precision and recall:

$$\mathrm{F}_1 = \left(\frac{\text{recall}^{-1} + \text{ precision }^{-1}}{2}\right)^{-1} \tag{11}$$

Here, the precision is the part of the predicted trajectory that genuinely belongs to the observed trajectory. The recall refers to the part of the observed trajectory that is correctly estimated. The performance of the introduced traffic flow predictor is examined by the Mean Absolute Error (MAE) (12):

$$\mathrm{MAE}(\boldsymbol{p}, \hat{\boldsymbol{p}}) = \frac{1}{|\boldsymbol{L}|} \sum_{l \in \mathrm{L}} |p_l - \hat{p}_l| \tag{12}$$

$\boldsymbol{p} = p_1, \cdots, p_l$ represents the number of users connected to each base station, which is extracted directly from the dataset, $\hat{\boldsymbol{p}} = \hat{p}_1, \cdots, \hat{p}_l$ represents the estimated values, and L denotes the number of base stations in each dataset.

C. Experimental Details

During the exploration phase, which refers to searching the highly accurate architecture for the *student LSTM*, we train each algorithm with 70% of the trace data and 30% of the data is used for testing each suggested predictor. After convergence and starting the exploitation phase, the discovered LSTM predictor was trained for 100 epochs. An epoch represents one iteration over the entire dataset. To speed up the training over defined epochs, we use a method called *Early Stopping*. Early Stopping monitors the training progress within epochs by checking the computed accuracy of each training epoch. If the fluctuation of accuracy over the patience epochs (e.g., $P_{epochs} = 10$) is less than $\Delta_{min} = 0.1$, we stop the training. Besides, the batch sizes are set to 200, and the initial learning rate of the LSTM is set to 0.002. We set the Q-learning rate (α) and discount factor (γ) to 0.01 and 1, respectively to prioritize rewards in the distant future. The predictors are trained and evaluated on a High Performance Computing Cluster at the University of Bern in Switzerland (HPC Cluster - UBELIX [3]) with Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz.

VIII. Performance Analysis

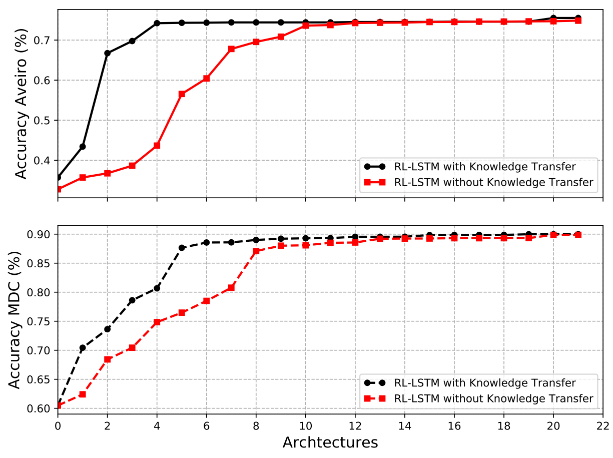
We evaluate the performance of the proposed RL-LSTM and the advantages provided by the TL to predict the trajectory of moving objects and also to predict the traffic flow in urban environments. To evaluate adaptability of the RL-LSTM, we analyze the prediction in the MDC and Aveiro datasets. The results of the advantages provided by TL is presented in Subsection VIII-A. Subsection VIII-B shows the trajectory prediction results while Subsection VIII-C shows the results for the traffic flow prediction.

A. Transfer Learning Results

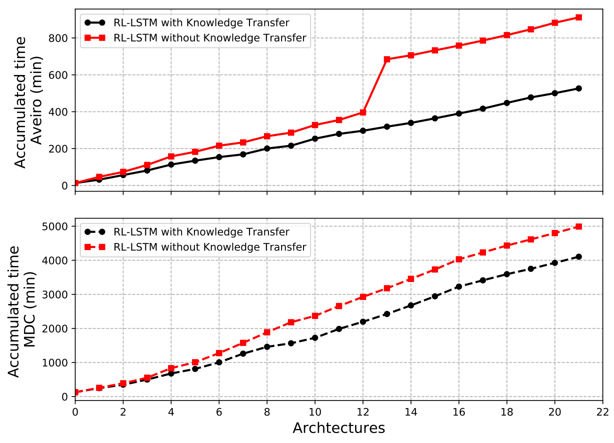
Due to the large number of different architectures suggested by the RL-LSTM, the convergence time (e.g., reaching to a highly accurate architecture for the *student LSTM*) can be an issue. Therefore, to have a faster convergence, we employed knowledge transfer between pre-trained LSTM at iteration $t-1$ and newly suggested LSTM at iteration t by the controller. The results of such a reduction for trajectory prediction for both datasets (e.g., MDC and Aveiro) are shown in Figure 4(a). In particular, Figure 4(a) shows the trajectory accuracy results in function of the architectures suggested by the RL-LSTM, while Figure 4(b) shows the accumulated training time of each architecture comparing the RL-LSTM with knowledge transfer against RL-LSTM without it.

When using the TL the RL-LSTM can reach the desired performance (trajectory prediction accuracy of 75% in the Aveiro dataset and 90% in MDC) earlier than RL-LSTM without knowledge transfer. Therefore, RL-LSTM with knowledge transfer converges at the 5-th and at the 7-th suggested architecture considering the Aveiro and MDC datasets, respectively (see Figure 4(a)). On the other hand, the RL-LSTM without

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



(a) Convergence time



(b) Exploration time

Fig. 4. Reinforcement learning-designed LSTM results

knowledge transfer converges at the 11-th and 13-th suggested architecture considering the Aveiro and MDC datasets, respectively (see Figure 4(a)). In terms of exploration time (e.g., time to find the most suitable architecture), the knowledge transfer provides a reduction of 70% in the accumulated time for both datasets (see Figures 4(b)).

B. Trajectory Prediction Results

In this analysis, we evaluate the performance of the most suitable LSTM architecture found in the previous section to predict the trajectory of moving objects in an urban environment based on the Aveiro and MDC datasets. We have compared the RL-LSTM with the following literature solutions: Hybrid Markov Chain (HMC) [39], Markov Chain (MC), Random Forest (RF), J48, and the LSTM proposed in [32]. Figure 5 shows the F_1-Score results of each solution. Figure 5(a) shows the average F_1-Score considering business days and weekends, while Figure 5(b) shows the results for all predictions as a Cumulative Distribution Function (CDF).

The results show that J48 and RF present the worst results in both datasets, reaching an average F_1-Score of approximately 0.4 during business days and weekend for both datasets (see Figure 5(a)). Also, for 80% of the predictions, both J48 and RF present an F_1-Score lower than 0.6 in both datasets (see Figure 5(b)). In turn, the performance of MC depends on the quality of the input data. Therefore, MC presents better results in the MDC dataset achieving an average F_1-Score of about 0.6 during business days and weekends (see Figure 5(a)), while in the Aveiro dataset the MC reaches an average F_1-Score of around 0.4 for both business days and weekends. To improve prediction performance of the MC predictor, the authors in [39] proposed the HMC, which can switch dynamically between the first and second order Markov Chain based on the quality of the input data, consequently improving the performance of the predictor. The HMC was specifically designed for performing prediction tasks (e.g., mobility and trajectory estimation) for the Aveiro and MDC datasets. The results show that HMC provides an average F_1-Score of approximately 0.6 in both datasets considering business days and weekends (see Figure 5(a)). In addition, for 40% of the predictions, HMC provides a F_1-Score higher than 0.7 in both datasets (see Figure 5(b)).

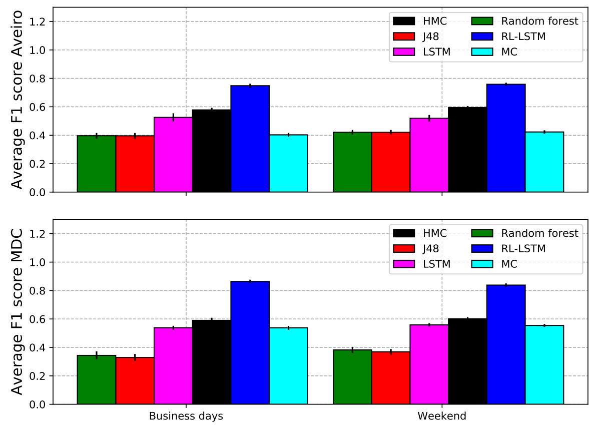
The LSTM predictor presented in [32] does not achieve a satisfying prediction performance in both datasets. This is the result of the hyper-parameters adjustments that need to be tuned specifically for each dataset, but the predictor has not such a capability. In this way, by using the LSTM present in [32], the average F_1-Score decreases by 15% when compared to the HMC in both datasets. Moreover, only 20% of the predictions present a F_1-Score is greater than 0.6 in both datasets (see Figure 5(b)). Finally, the efficiency of the architecture suggested by the RL-LSTM can be seen by analyzing the substantial improvements over the HMC results. Therefore, by using the most suitable architecture for each dataset, the RL-LSTM increases the average F_1-Score by 33% in the Aveiro dataset and by 50% in the MDC dataset compared to HMC. Also, for 80% of the predictions, the F_1-Score is higher than 0.7 in both datasets. Therefore, by employing the RL to realize high-performance architectures for the LSTM, we overcome the performance of all predictors, even the HMC, which is a predictor designed explicitly for the MDC and Aveiro datasets.

C. Density Prediction Results

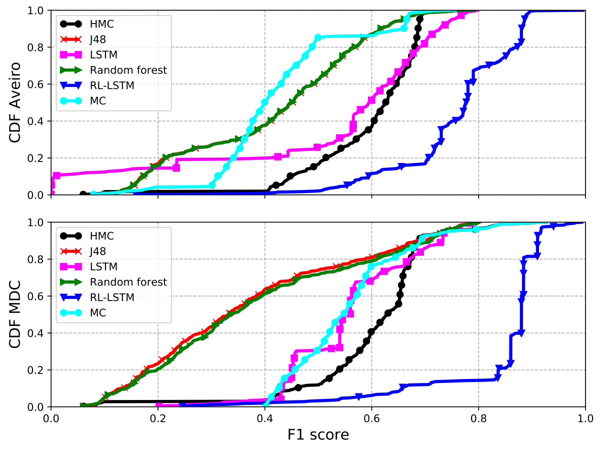
TABLE I
HIGH ORDER CONVOLUTION RESULTS

	Convolution orders				
k-order	1-st	2-nd	3-rd	4-th	5-th
MAE-Aveiro	0.3782	0.3280	*0.2567*	0.3378	0.3976
MAE-MDC	1.7348	*1.1203*	1.4351	1.5644	1.8421

In this subsection, we evaluate the performance of HERITOR to predict traffic flow dynamics in comparison with the novel probabilistic model proposed in [39]. In this way, first, we need to find the most efficient convolution order (e.g., the k-th order) for HERITOR. Table I shows the MAE results for 5 different orders in which $k \in \{1, 2, 3, 4, 5\}$. The convolution order represents the area (e.g., number of hops)

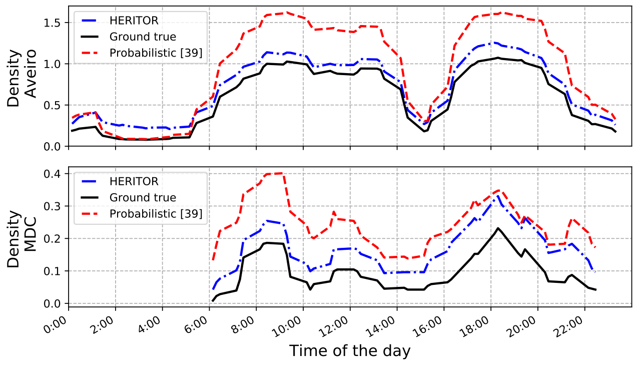


(a) Average F_1 score

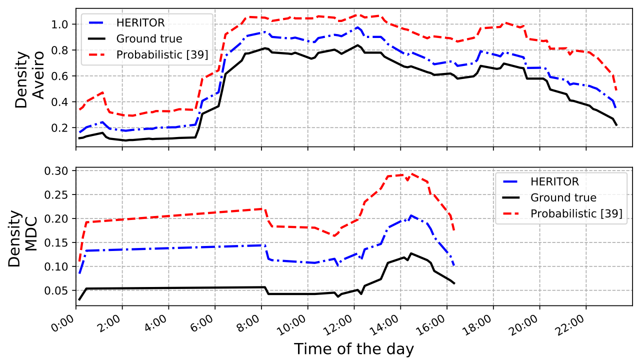


(b) F_1 score distribution

Fig. 5. Spatiotemporal trajectory prediction results.



(a) Business days



(b) Weekends

Fig. 6. Traffic flow prediction results.

around each node that will be taken into account during the convolution task. For the given datasets HERITOR reaches its best prediction performance with $k = 3$ and $k = 2$, which represents the optimal spatiotemporal structure based on the road network used. These results are presented in Table I. Therefore, the HERITOR will be fed using the 3-rd convolution order in the Aveiro dataset and 2-nd in the MDC dataset.

Figure 6 shows the urban traffic flow prediction results as an average density over the city throughout the day comparing HERITOR and the solution proposed in [39]. Figure 6(a) represents the predictions and the ground truth for business days while Figure 6(b) shows the results during weekends. The results show the efficiency of HERITOR, which provides a traffic flow prediction very similar to the real traffic flow during business days and weekends (see Figures 6(a) and 6(b)). This is due to the high order graph convolution operator and adaptive distance adjacency matrix that capture the pure spatiotemporal dependencies and also due to the efficient LSTM designed using RL. Specifically, in the worst case scenario HERITOR introduces an average error of 10% when compared to the real traffic, while the solution proposed in [39] can introduce an error of approximately 50% in the traffic flow prediction when compared to the real traffic flow.

With these results, we can conclude that *(i)* the TL methods employed by the framework proposed in this work can speed up the time to find the most efficient LSTM architecture; *(ii)* the LSTM designed using Reinforcement Learning is highly adaptive and outperforms literature solutions for trajectory and traffic flow prediction; and *(iii)* the graph convolution methods extract optimal spatiotemporal dependencies of a road network and provide better support for traffic flow prediction.

IX. Conclusions

In this work, use LSTM to estimate future trajectories and traffic flows of moving objects in urban areas. The main challenges are (i) realizing a high-performance architecture for the LSTM to have satisfying prediction performance for the given task (ii) capturing the rich spatiotemporal dependencies of traffic flows over trajectory networks in urban areas. To address the first challenge, we propose an RL-based method for training a learning agent as a controller within a large search space to automatically generate high-performance LSTMs for the given prediction task. To accelerate this process, we introduced a TL approach to transfer knowledge from a pre-trained LSTM to the new suggested LSTM. Besides, we represent network traffic as graph-structured data and introduced a high-order convolution operator and adaptive directional distance adjacency matrix to learn spatiotemporal dependencies in the traffic network. Experimental evaluations on two real-world datasets show that the proposed predictors provides better prediction performance over state-of-the-art works. Using transferred knowledge, we speed up the process of searching an optimal architecture of an LSTM by up to 70%.

References

[1] H. Ye, L. Liang, G. Y. Li, J. Kim, L. Lu, and M. Wu, "Machine learning for vehicular networks: Recent advances and application examples," *IEEE Vehicular Technology Magazine*, vol. 13, no. 2, pp. 94–101, June 2018.

[2] 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

[3] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *CoRR*, vol. abs/1611.01578, 2016. [Online]. Available: http://arxiv.org/abs/1611.01578

[4] D. Ashbrook and T. Starner, "Using gps to learn significant locations and predict movement across multiple users," *Personal Ubiquitous Comput.*, vol. 7, no. 5, pp. 275–286, Oct. 2003. [Online]. Available: http://dx.doi.org/10.1007/s00779-003-0240-0

[5] B. Chapuis, A. Moro, V. Kulkarni, and B. Garbinato, "Capturing complex behaviour for predicting distant future trajectories," in *Proceedings of the 5th ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems*, ser. MobiGIS '16. New York, NY, USA: ACM, 2016, pp. 64–73. [Online]. Available: http://doi.acm.org/10.1145/3004725.3004730

[6] M. Karimzadeh, Z. Zhao, F. Gerber, and T. Braun, "Pedestrians complex behavior understanding and prediction with hybrid markov chain," in *2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct 2018, pp. 200–207.

[7] ——, "Mobile users location prediction with complex behavior understanding," in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, Oct 2018, pp. 323–326.

[8] J. Paparrizos and L. Gravano, "Fast and accurate time-series clustering," *ACM Trans. Database Syst.*, vol. 42, no. 2, pp. 8:1–8:49, Jun. 2017. [Online]. Available: http://doi.acm.org/10.1145/3044711

[9] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *ROBOMECH Journal*, vol. 1, no. 1, p. 1, Jul 2014. [Online]. Available: https://doi.org/10.1186/s40648-014-0001-z

[10] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1549–15 498, 2018.

[11] J. Schulz, C. Hubmann, J. Löchner, and D. Burschka, "Interaction-aware probabilistic behavior prediction in urban environments," *CoRR*, vol. abs/1804.10467, 2018. [Online]. Available: http://arxiv.org/abs/1804.10467

[12] C. Barrios, Y. Motai, and D. Huston, "Trajectory estimations using smartphones," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 12, pp. 7901–7910, Dec 2015.

[13] G. Xie, H. Gao, L. Qian, B. Huang, K. Li, and J. Wang, "Vehicle trajectory prediction by integrating physics- and maneuver-based approaches using interactive multiple models," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5999–6008, July 2018.

[14] A. Houenou, P. Bonnifait, V. Cherfaoui, and W. Yao, "Vehicle trajectory prediction based on motion model and maneuver recognition," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 4363–4369.

[15] N. Deo, A. Rangesh, and M. M. Trivedi, "How would surround vehicles move? A unified framework for maneuver classification and motion prediction," *CoRR*, vol. abs/1801.06523, 2018. [Online]. Available: http://arxiv.org/abs/1801.06523

[16] M. Karimzadeh, F. Gerber, Z. Zhao, and T. I. Braun, "Pedestrians trajectory prediction in urban environments," in *2019 International Conference on Networked Systems (NetSys) (NetSys'19)*, Garching b. München, Germany, Mar. 2019.

[17] J. Schlechtriemen, F. Wirthmueller, A. Wedel, G. Breuel, and K. Kuhnert, "When will it change the lane? a probabilistic regression approach for rarely occurring events," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, June 2015, pp. 1373–1379.

[18] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014. [Online]. Available: http://arxiv.org/abs/1412.3555

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[20] D. J. Phillips, T. A. Wheeler, and M. J. Kochenderfer, "Generalizable intention prediction of human drivers at intersections," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 1665–1670.

[21] B. Kim, C. M. Kang, S. Lee, H. Chae, J. Kim, C. C. Chung, and J. W. Choi, "Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network," *CoRR*, vol. abs/1704.07049, 2017. [Online]. Available: http://arxiv.org/abs/1704.07049

[22] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[23] D. R Drew, "Traffic flow theory and control / donald r. drew," *SERBIULA (sistema Librum 2.0)*, 06 2019.

[24] G. E. P. Box and G. Jenkins, *Time Series Analysis, Forecasting and Control*. San Francisco, CA, USA: Holden-Day, Inc., 1990.

[25] W. Huang, G. Song, H. Hong, and K. Xie, "Deep architecture for traffic flow prediction: Deep belief networks with multitask learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2191–2201, Oct 2014.

[26] N. Laptev, J. Yosinski, E. L. Li, and S. Smyl, "Time-series extreme event forecasting with neural networks at uber," 2017.

[27] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," *CoRR*, vol. abs/1610.00081, 2016. [Online]. Available: http://arxiv.org/abs/1610.00081

[28] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, "Spectral networks and locally connected networks on graphs," in *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, 2014.

[29] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *CoRR*, vol. abs/1609.02907, 2017.

[30] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 1993–2001. [Online]. Available: http://papers.nips.cc/paper/6212-diffusion-convolutional-neural-networks.pdf

[31] ——, "Diffusion-convolutional neural networks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. USA: Curran Associates Inc., 2016, pp. 2001–2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=3157096.3157320

[32] D. Andreoletti, S. Troia, F. Musumeci, S. Giordano, G. Maier, and M. Tornatore, "Network traffic prediction based on diffusion convolutional recurrent neural networks," 02 2019.

[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=2627435.2670313

[34] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10. USA: Omnipress, 2010, pp. 807–814. [Online]. Available: http://dl.acm.org/citation.cfm?id=3104322.3104425

[35] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14236

[36] T. Chen, I. J. Goodfellow, and J. Shlens, "Net2net: Accelerating learning via knowledge transfer," *CoRR*, vol. abs/1511.05641, 2016.

[37] J. K. Laurila, D. Gatica-Perez, I. Aad, B. J., O. Bornet, T.-M.-T. Do, O. Dousse, J. Eberle, and M. Miettinen, "The mobile data challenge: Big data for mobile computing research," 2012. [Online]. Available: http://infoscience.epfl.ch/record/192489

[38] C. Ameixieira, A. Cardote, F. Neves, R. Meireles, S. Sargento, L. Coelho, J. Afonso, B. Areias, E. Mota, R. Costa, R. Matos, and J. Barros, "Harbornet: a real-world testbed for vehicular networks," *IEEE Communications Magazine*, vol. 52, no. 9, pp. 108–114, Sep. 2014.

[39] Z. Zhao, L. Guardalben, M. Karimzadeh, J. Silva, T. Braun, and S. Sargento, "Mobility prediction-assisted over-the-top edge prefetching for hierarchical vanets," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 8, pp. 1786–1801, Aug 2018.