Micro-forecasting for Large-scale Traffic Light Optimization

STEFANO GIOVANNI RIZZO, Qatar Computing Research Institute, Qatar
GIOVANNA VANTINI, Qatar Computing Research Institute, Qatar
SANJAY CHAWLA, Qatar Computing Research Institute, Qatar

An optimal traffic light strategy can be seen as a two-steps process: anticipation of the traffic flow and phase selection to accommodate the predicted flow. Recent progress on reinforcement learning has brought promising results to traffic light optimization by implicitly (i) learning to predict future states along with their rewards, and (ii) controlling the phases to maximize the reward. At the same time, these methods require a very large amount of data, either historical or simulated, and tend to overfit on the reward maximization without learning or exposing the model dynamics. In this paper, we separate prediction and control for traffic light optimization, by combining a novel method for micro-forecasting of the flowing vehicles with the effectiveness of the Max Pressure control algorithm. We first introduce the Flow Attention Network (FAN), a micro-forecasting deep learning architecture as a special case of a graph attention network. The FAN uses the actual road network connections as input graph, and the message-passing paradigm as a way of predicting the flow of the vehicles in the network. In the input graph, each road segment of road network is represented as a node and each connection between segments as an edge. The number of vehicles in each movement is predicted based on latent features of source and destination road segments, learned as node embeddings during the training. The proposed method is evaluated on a real road network traffic, reproduced on the SUMO micro-simulator, showing significant improvement in avoiding congestion, increasing throughput and reducing CO2 emissions.

Additional Key Words and Phrases: traffic light optimization, graph attention network, flow prediction, Max Pressure

ACM Reference Format:
Stefano Giovanni Rizzo, Giovanna Vantini, and Sanjay Chawla. 2020. Micro-forecasting for Large-scale Traffic Light Optimization. 1, 1 (November 2020), 21 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The optimization of traffic lights scheduling is a challenging task, with a high potential of positive impact on urban mobility. Employing an efficient traffic light policy reduces average travel time, lowers the emissions, and avoid high congestion.

Several recent studies focus on Reinforcement Learning solutions for traffic lights optimization [1], however it has been shown how simple algorithms such as Max Pressure [2] may outperform very complex deep neural network in terms of congestion avoidance, generalization and traffic flow optimization [3]. Reinforcement learning algorithms tend to overfit on the re-played data [4] and may act in an unexpected way with the only objective of maximizing the defined reward. The problem is exacerbated in multi-agent systems, where the coordination and the optimization of a

Authors' addresses: Stefano Giovanni Rizzo, strizzo@hbku.edu.qa, Qatar Computing Research Institute, Doha, Qatar; Giovanna Vantini, gvantini@hbku.edu.qa, Qatar Computing Research Institute, Doha, Qatar; Sanjay Chawla, schawla@hbku.edu.qa, Qatar Computing Research Institute, Doha, Qatar.

XXXX-XXXX/2020/11-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

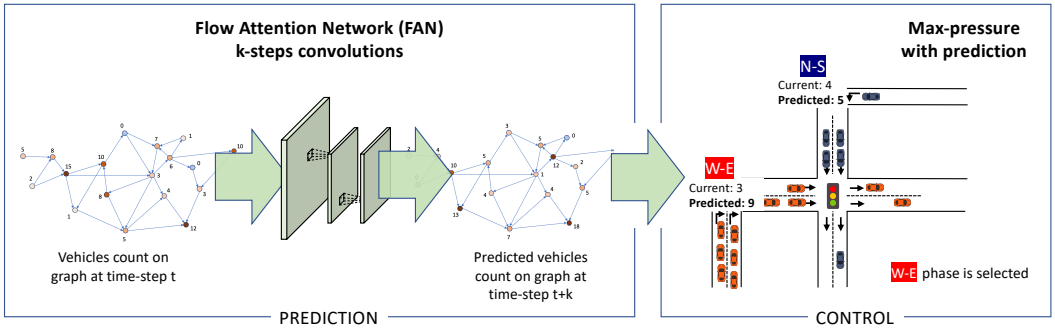

Fig. 1. Prediction and control steps in the FANPressure approach. In the prediction step, given the current counts of vehicles, recurrent convolutions of the FAN network are performed to predict the future state of vehicles on each road segment. In the control step, the Max Pressure algorithm is used with prediction input instead of current counts.

global reward leads to very slow convergence and often to overfitting due to the environments being inherently non-stationary. Another main concern when employing RL agents in real-world intersections is that the choices taken by a deep neural network are still hard to explain [5], in comparison to simpler strategies such as actuated signalization or pressure-based algorithms. Such black-boxes solutions are not acceptable in urban areas, where safety is the primary concern, before traffic flow optimization. Finally, Max Pressure also exhibits global optima properties that guarantee that the choice is not completely competitive.

On the other hand, Max Pressure is a greedy algorithm, that considers only the vehicles count on the road segments adjacent to the intersection and picks the highest pressure at each time step, which may disadvantage the low-volume entries. Above all, Max Pressure can only consider the current count of cars, however, with several entries from nearby roads and fast flows of busy areas, the queue lengths at an intersection tend to vary significantly during the green time. Being able to consider not only the current vehicles in line, but also the vehicles coming in the next few seconds, would provide the Max Pressure algorithm more valued information of the near-future state.

With the aim of traffic light optimization, the traffic light system requires a very short term prediction that strictly depends on the contiguous road status. Simply metering the adjacent road volumes, however, does not account for each possible route that vehicles can take in a complex road network. Current advancements in traffic prediction are coarse-grained in both time and space, and cannot account for the frequent oscillations in the queue length at an intersection. To put this in perspective, the state-of-the-art models for traffic prediction may allow to forecast the volume of cars at an intersection in the following 5 minutes, however they are not able to predict that there are some vehicles approaching in the next 10 seconds. This is crucial for intelligent traffic light control: a coarse prediction can be used as support data for the estimation of proportions between phases in the cycle, however it cannot help in selecting the specific phase at the current time, reacting to the near-future state of traffic.

In this paper, we introduce a deep learning model with the objective of making a short-term prediction of the vehicles flow, exploiting the graph of the road network, then we combine this prediction with the Max Pressure control algorithm to maximize the throughput for the following time interval.

The main contributions of this paper are the following:

- The FAN Graph Convolutional Network, with attention mechanisms, to learn latent features of road segments and junctions and perform recurrent flow operation along the graph, with the objective of micro-forecasting the traffic flow of individual vehicles. To the best of our knowledge this is the first data-driven approach to micro-forecasting.
- A control capability for the Graph Convolutional Network, that allows to change the structure of the graph from an external input, dynamically enabling and disabling connections. This allows to take into account the current state of the traffic lights, where some movement are temporary prevented.
- The combination of prediction and control, using the output of the FAN model in a state-of-the-art traffic light control method, known Max Pressure, to provide an intelligent multi-agent control system.
- The first evaluation of a data-driven method on a non grid-like real case scenario, using a real road network simulated through re-played recorded traffic.

The problem of flow micro-forecasting is formulated as a stochastic flow network. Stochastic flow networks are directed graphs where tokens are injected in input nodes, travel stochastically through edges and depart from the network after reaching output nodes. We specialized the stochastic flow network to cope with a more realistic scenario, in which the number of tokens leaving a node at each step depends on features of the node itself and any node is both an input and output node. The features are not known and only the number tokens on the nodes can be observed. The proposed approach learns the latent features of nodes and edges using a Graph Convolutional Network (GAT) with the objective of predicting the next token observation given the current observation. The network is enhanced with multi-head attention among neighborhood of each node, to harness the features learned from connected neighbors. Finally, the considered scenario provides a mean of controlling the flow network, by selecting groups of nodes whose outgoing flow will be blocked in the next step with the goal of optimizing flow and avoid congestions.

FANPressure is evaluated on a real road network, loaded and simulated in the SUMO micro-simulator [6], replaying 24 hours of real traffic data from vehicles trajectories, augmented in order to reach congestion levels. The experimental results show that our method exhibits higher performances in reducing queues and waiting time, while being able to avoid congestion for the whole simulated period.

The rest of the paper is structured as follows. Section 2 provides related work in the two areas of traffic flow predictions and traffic light optimization. Section 3 gives a brief introduction to Graph Convolutional Networks, attention mechanisms, flow graphs and the Max Pressure algorithm. Section 4 presents the proposed method introducing the components of the FAN network and the control algorithm. In Section 5 we describes the experimental setup and provide the empirical results for the method's effectiveness and Section 6 concludes.

2 RELATED WORK

To the best of our knowledge, no work in literature has combined the vehicles flow prediction with traffic light control methods. Separately, these two problems have been extensively studied in transportation and operational research. However, the problem of micro-forecasting using a predictive model has never been formulated in literature. In previous research the individual movements of vehicles are only modeled using micro-simulators [6]. Micro-simulators requires a lot of manual efforts to calibrate and can take a time longer than real-time to simulate large networks, given that

micro-simulation is not an embarrassingly parallel problem [7].

Prediction. Traffic forecasting is a core component of intelligent transportation systems: being able to predict a change in the traffic flow improves the traffic control, allowing to handle congestions in a preventive way [8]. While there is no study on micro-forecasting with short term prediction for vehicles movement, several studies focused on predicting the changes in traffic speed and traffic flow, with a time granularity usually spanning from 15 minutes [9] to several weeks [8]. As a data-intensive problem, traffic prediction has attracted the interest of data science and lately deep learning communities. Traditional methods employed statistical models mainly based on historical averages to predict traffic flow and traffic speed, however these methods required strong assumptions that could not cope with the complexity of traffic data [10, 11]. With the advent of Deep Learning models it has been possible to take into account complex dependencies, by applying time-series prediction models to the traffic flow detected on the road network. As a straightforward solution to time-series forecasting, authors have applied Recurrent Neural Networks (RNN) to time series of traffic data [12, 13], however several works have considered both temporal and spatial dependencies [14, 15] In more recent works, the spatial dependencies have been take into account representing the collection of detectors as a graph and applying RNNs [16], Graph Convolutional Networks [17] or Diffusion Convolutional Networks (DCNNs)[9, 18]. Lastly recurrent DCNNs have been proposed to better account for both the temporal and the spatial dimensions [8]. The represented graphs in all the aforementioned works are not the actual directed graph of the road network, but rather a graph based on the distance between detectors. The incidence matrix of these graphs is built using a thresholded Gaussian kernel [8], that is by setting a threshold of distance between detectors to decide whether two nodes are connected or not. For the connections that exceed the threshold, the weight is given an inverse function of the distance.

Control. Several timing models have been proposed with the goal of optimizing traffic signal control. These models usually assume that arrival and departure rate of vehicles are known and build an analytical model to derive the time of the phase [19]. Fuzzy logic controllers have also been proposed to respond to real-traffic demand [20]. Recently, novel computational approaches such as genetic algorithms [21], micro-auctions [22] and Reinforcement Learning (RL) algorithms have shown promising improvements [1]. RL approaches model the signalized vehicular network as a Markov Decision Process. The state of the MDP is the representation of the traffic states, such as the discretized vehicles position [23, 24] or a range of traffic volume [25]. More often the traffic state is represented as the occupancy of discretized cells in a grid [26, 27] or the vehicles properties in each grid's cell [28]. Estimating the value of states with tabular approaches requires the exploration of a very high number of trajectories over the distribution of states and actions, and enough computer memory to store the value of each state. For this reason classic RL approaches are computationally bounded by a limited state and action spaces. In [29] for example, tabular Q-learning is applied to an heavily congested environment using the lengths of the four approaching queues as state and the change of phase as action. More recent approaches use neural network as function approximators to estimate the traffic states value (Deep Q-learning [27, 30, 31]) or to learn directly a the policy, as function from state to actions phases (Deep Policy Gradient [32]). Deep learning approaches allowed the use of more complex states representation, such as camera view from above the intersection or, more precisely, a screenshot of the simulator [32, 33]. Decisions taken by a deep RL agent may be difficult to interpret [33], especially if the state space is large and the model is overfitted, however some effort has been given in relating action preferences and observed states [5] While most of RL works are applied on simple grid-like networks, some of these works have been evaluated on real,

complex road network, simulating traffic recorded in real life [34].

Several studies have also explored the possibility of controlling multiple intersections simultaneously using a distributed multi-agent system, by communicating action advises [21, 23] or by using coordination graphs [26]. The Max Pressure control algorithm [2] has been shown to optimize the global throughput in a scenario with multiple intersections. The formulation of pressure as a differential in volume has been used as state representation also in Reinforcement Learning solutions such as Presslight [3]. A more recent version of this work, CoLight [35], also uses a Graph Convolutional Network to include the influence of neighbor intersections, by computing the value for the state-action pair (Q-value). Using Graph Convolutional Networks for RL in traffic light control has fast become a trend to optimize scenario in large grids of intersections [36–38]. One shortcoming of all the above GCN-RL works is that they are only tested on synthetic grid-like networks that cannot capture the complexity of a real network, while at the same time it is known that the lack of generalization in RL networks is still a hard to solve problem [4, 39]. By decoupling prediction and control, in FANPressure we avoid overfitting by relying on a robust control algorithm (Max Pressure), while constraining the prediction of movement to the actual count of vehicles. FANPressure is tested by simulating real traffic on a real road network provided by local authorities.

3 BACKGROUND

3.1 Glossary

$\boldsymbol{A}$	Adjacency matrix for the complete road network graph.
$\boldsymbol{C}$	Control matrix to modify the connections of the adjacency matrix $\boldsymbol{A}$. The control matrix allows to block or unblock traffic movements, to reflect the current state of the traffic lights.
N	Number of nodes in the graph.
i, j	Two connected nodes of the graph. The i-node is a source node, the j-node is a destination node. A vehicles moves from i to j.
$\boldsymbol{I}_n$	Identity matrix with n rows and n columns.
node	A road segment delimited by junctions.
connection	A connection between two nodes. If node i and node j are connected, $A_{i,j} = 1$ and vehicles can move from i to j.
junction	Set of connections between road segments, all in the same position (e.g. an intersection or a side entry).
road network	The complete representation of roads, junctions and traffic lights.
flow	The moving of vehicles from the current node to the subsequent node.
Φ	The flow distribution matrix. Each row Φ_i represents the distribution of vehicles from i. $\Phi_{i,i}$ denotes the fraction of vehicles not moving from i in the next time step. $\Phi_{i,j}$ denotes the fraction of vehicles moving from i to j in the next time step.
phase	The state of a traffic light. In the graph, the phase can be represented as a set of connections. Each phase enables vehicles to move along the related connections.
TL	a traffic light is a special junction where the set of connections of a phase can be enabled or disabled dynamically.
all-red phase	a virtual or real phase in which all the connections of the traffic light are disabled. Any incoming vehicle stops in line.

3.2 Graph Convolutional Operation

Given X the features matrix of all nodes, W the parameters matrix, A the adjacency matrix of the graph and D the degree matrix, a Graph Convolutional Network (GCN) layer is defined by

$$Z = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}XW \tag{1}$$

where Z is the convolved signal matrix [40]. The $\tilde{}$ operation on A and D is a renormalization trick to avoid numerical instabilities when used in a deep learning model, where self-loops are added to the matrix: $I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \rightarrow \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$, with $\tilde{A} = A + I_N$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$.

3.3 Attention mechanisms in GCNs

In a standard GCN, the convolution operation aggregates the output of the previous layer from the neighbors node using a normalized sum:

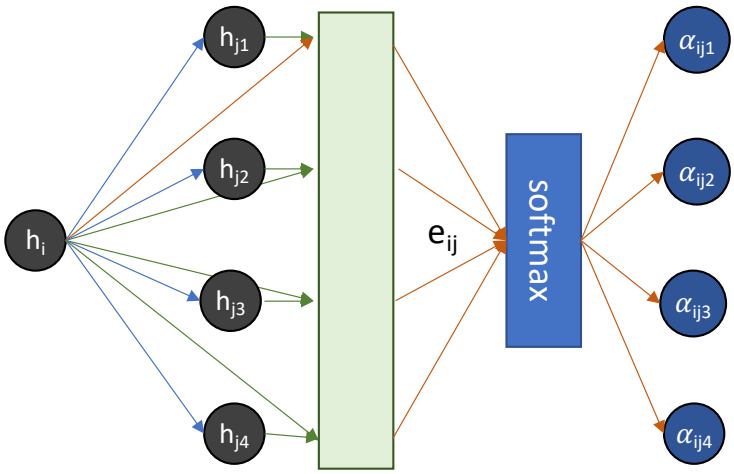

Fig. 2. A layer in a Graph Attention Network (GAT).

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \frac{1}{c_{ij}} W^{(l)} h_j^{(l)}\right) \tag{2}$$

It has been shown that attention mechanisms applied to Graph Convolution Networks increases the accuracy of node embeddings [41]. In Figure 2 is shown how attention replaces the statically normalized convolution operation.

The convolutional layer with attention is defined as following:

$$z_i^{(l)} = W^{(l)} h_i^{(l)} \tag{3}$$

$$e_{ij}^{(l)} = ReLU(\vec{a})^T \left(z_i^{(l)} \| z_j^{(l)}\right) \tag{4}$$

$$\alpha_{ij}^{(l)} = \frac{\exp\left(e_{ij}^{(l)}\right)}{\sum_{k \in \mathcal{N}(i)} \exp\left(e_{ik}^{(l)}\right)} \tag{5}$$

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(l)} z_j^{(l)}\right) \tag{6}$$

In the FAN model, α_{ij} will represent the probability of moving from node i to node j, with z_i being the predicted number of vehicles in node i.

3.4 Stochastic Flow Networks

A stochastic flow network [42] is a directed graph which consists of a set of nodes, including source nodes and sink nodes, and a collection of directed arcs which join pairs of nodes and are associated with a probability. In a stochastic flow network tokens enter through the source nodes, travel stochastically in the network, and can exit the network through the sink nodes. Each node

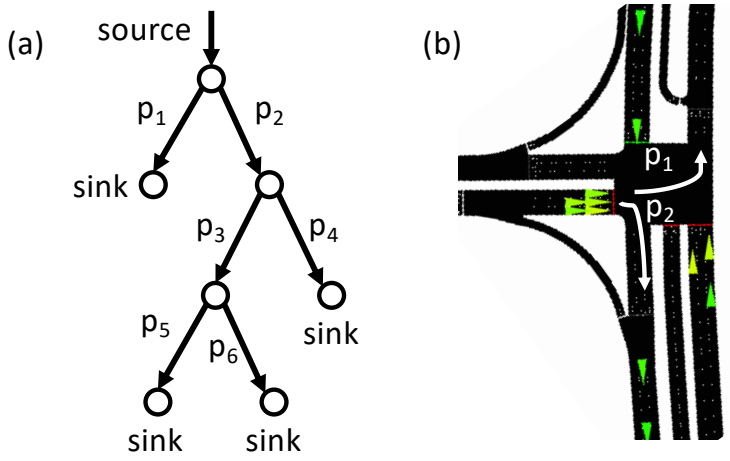

Fig. 3. Stochastic flow network (a) and traffic flow on a road network (b).

is a splitter: a token can enter a node and exits through one of the output edges, according to a predefined probability distribution.

As shown in Figure 3, the traffic flow in the road network (Figure 3b) can be represented as a stochastic flow network (Figure 3a). In FAN, we extend the stochastic flow network representation with a probability of the node being a source or sink (cars leaving or entering a parking).

3.5 Max Pressure

Max Pressure is a control algorithm for networks of signalized intersections which provably maximizes throughput globally by acting locally [2]. In Max Pressure the time horizon is divide into time steps of fixed length T. Within a time step, excluding the amber light length L, there exists $T - L$ time units available for selecting a phase, in which one or more connections will be enabled from incoming nodes to outgoing nodes. The pressure $P(p_i)$ of a phase is defined as the number of vehicles on incoming lanes that have green light in that phase, minus the number of vehicles on the outgoing lanes of the phase. After computing the pressure of each phase, the next phase is set as

$$\arg\max_{p_i \in \mathrm{TL}} P(p_i) \tag{7}$$

The Max Pressure control is limited to the current observation of approaching and leaving vehicles, from the adjacent incoming and outgoing lanes. By applying the proposed Flow Attention Network k times on the current observation, we are able to consider the k^{th}-order road connections from the traffic light and to predict the number of vehicles that will approach the intersection during the $T - L$ interval.

3.6 ϵ-Greedy Max Pressure

The Max Pressure control is a greedy algorithm. At each time instance it will assign the green light to the phase with the maximum pressure. In certain settings, this can often lead to assignment of the red light to other phases for a prolonged and even indefinite period of time. Thus from a behavioral aspect (where actually drivers are involved), this is not feasible. One simple way to address the obvious discrepancy of Max Pressure is to design an ϵ-greedy version of the protocol. Thus with a small probability ϵ, the algorithm will select a phase which does not have the maximum

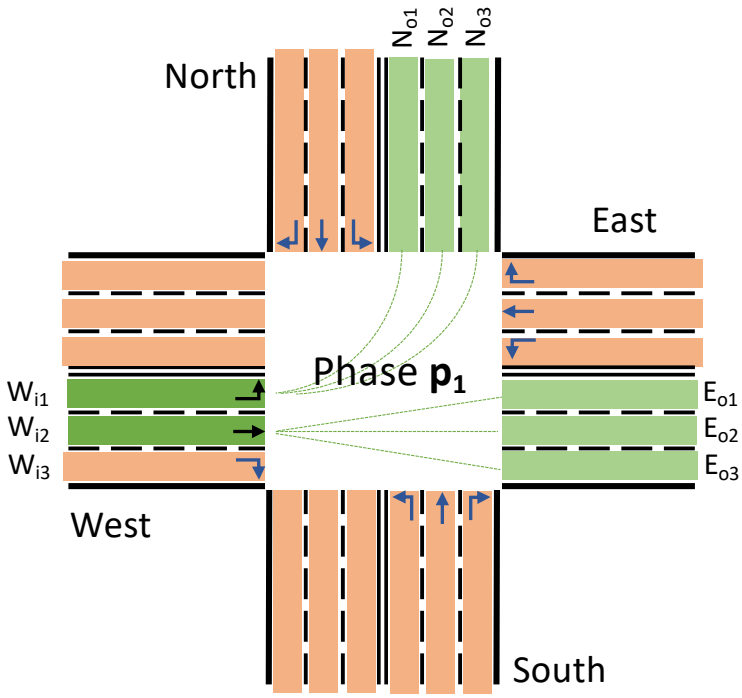

Fig. 4. Example of a phase with active incoming and outgoing nodes (road segments). Pressure of phase p_1 is given by $(W_{i1} + W_{i2}) - (N_{o1} + N_{o2} + N_{o3} + E_{o1} + E_{o2} + E_{o3})$, where W_{i1} is the count on vehicles currently on the first node from West.

pressure. More formally,

$$p = \begin{cases} \arg\max_{p_i \in \text{TL}} P(p_i) & \text{w.p } 1 - \epsilon \\ \text{uniformly at random } p \in TL & \text{w.p. } \epsilon \end{cases} \tag{8}$$

ϵ-greedy algorithms have been used extensively in multi-armed bandits and reinforcement learning problems to balance between exploitation and exploration where the latter is controlled by the parameter ϵ. However, in our case it more use a fairness criterion to ensure that no vehicle is made to wait for a disproportionate amount of time at an intersection. However, our experiments suggest that the ϵ-greedy approach actually is a better algorithm, i.e., it reduces overall congestion in a much more efficient manner than the Max Pressure algorithm.

4 FAN: FLOW ATTENTION NETWORK

The full architecture of the proposed FAN is shown in Figure 5, which can be logically separated in road embedding layers to learn latent features of the nodes, entering and leaving prediction for vehicles that completely leave or enter the network without moving to adjacent nodes (e.g. parking), and flow distribution prediction. The model can be applied in a recurrent fashion to allow vehicles flowing through multiple connections. Finally, the prediction is used to estimate the future pressure and select the optimal phase.

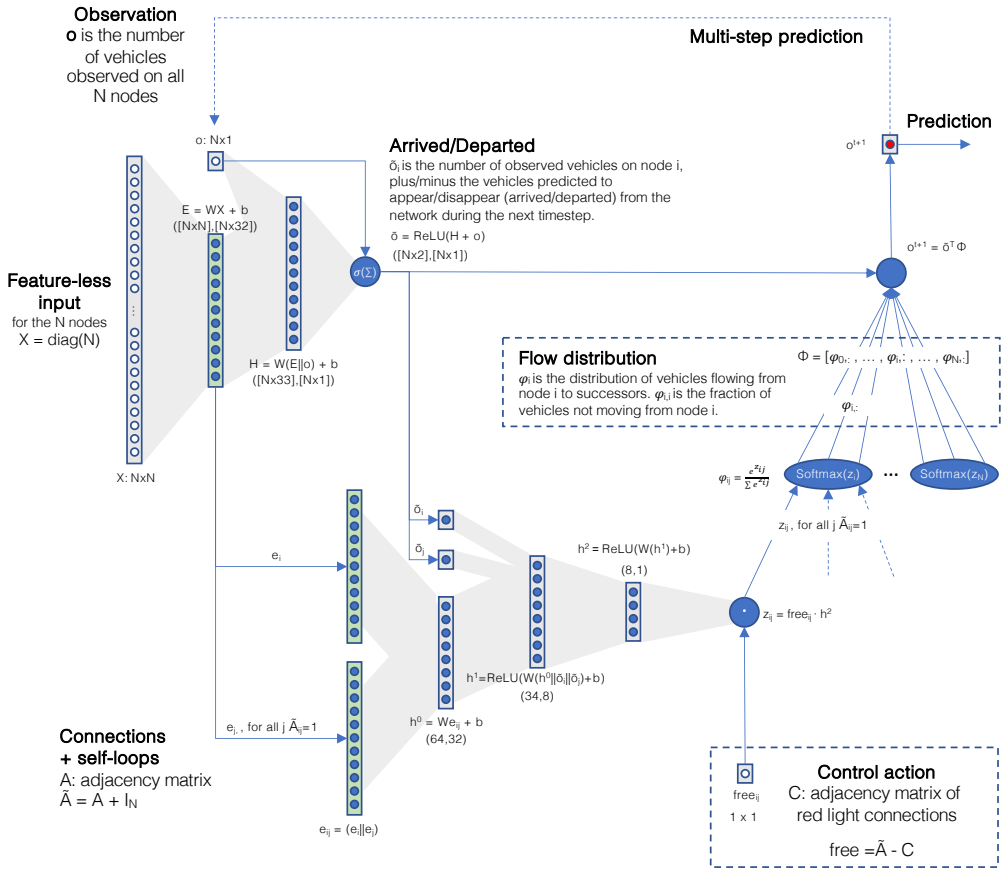

Fig. 5. Full architecture of a FAN, with nodes features X, graph A, current observation o and control matrix C as input.

4.1 Initial graph of road connections

The initial graph is given as an adjacency matrix A, where each vertex is a road segment between two junctions. Each road segment has only one direction, so for example if a road has two directions it will be represented with two different nodes in the graph. To simplify the lane-changing dynamics, the road segment may include multiple lanes, thus multiple lanes of the same direction are represented as a single node of the graph. This way, vehicles are allowed to freely change lanes to follow their routes or to overtake, while no additional complexity is required to predict vehicle movements between one lane to another. We also assume that each road segment has a detector (e.g. camera, loop detector, or area detector) able to estimate the count of cars on it. The connection between two road segment results in a connection between the related two nodes in the graph, therefore each junction in the road network is an edge in the graph. In the case of an intersection, an incoming road can be connected to multiple outgoing roads. Following this structuring, it is possible to represent a traffic light phase as a set of connections between the nodes that represents incoming lanes and the nodes that represent outgoing lanes.

4.2 Road embeddings

We don't make any assumption on the specific features of the road segments or the connections that the model can have access to. Instead, our goal is to learn latent features of the road segments, as well as the connections between road segments. For this reason, in the input space, each road segment and each connection has a one-hot-encoding as a unique feature-less representation.

$$X = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & 0 & \ddots & 0 & \vdots \\ 0 & \dots & 0 & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix} \begin{matrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{matrix}$$

In the above diagonal matrix, the j^{th}-row is the feature-less representation of the node j.

During the training of the FAN, an embedding of road segments is learned with the objective of predicting:

(1) Vehicles parking and entering the network: this will depend on the latent features of the road segment and on the current count of vehicles.
(2) Flow propagation: the number of vehicles moving to an adjacent road segment through a connection, as well as the number of vehicles staying in the road segment (through self-loop connection)

The idea behind learning latent features of the road segments is to capture properties such as the capacity and the average speed and the likelihood of the road segment being the start or the end of a route, features that are functional to the prediction of the two above predictions. These features are not explicitly given or modeled, instead they are implicitly learned during training with the downstream task objective of predicting the vehicles movements.

The embedding of the node is given by the parameters of the first hidden layer of the FAN, $E = W^{(1)}X$, where $W_{N\times l}$ is a matrix of trainable parameters and l is the embedding dimension.

4.3 Entering & Leaving prediction

Entering vehicles are vehicles that were not part of the total count of vehicles in the total road network, and enter in the current time step. Leaving vehicles are vehicles that were part of the total count of vehicles in the total road network, that leave the network in the current time step. In the real scenario, vehicles are inserted in the network when they leave a parking and begin their trip. Once the vehicles reach their destination, their trip ends entering some parking and the vehicles are removed from the network. From the perspective of an observer who has access to the detector counts, these vehicles will randomly appear or disappear at specific nodes. Unlike in the normal flow process, a vehicle that reaches its destination will be removed from one node without being added to the next node. In order to account for this "inconsistency", a sub-network of the FAN model is devoted to predict the number of vehicles appearing on the node, without coming from any connected node, and the number of vehicles disappearing from the node, without moving to a succeeding node.

The embedding E_j of a node j is merged with the current observed o_j of vehicles in j. A linear layer is applied to learn a linear regression, predicting the sum of cars leaving or entering: $H = W^{(2)}(E||o) + b$. We want to enforce that this prediction will not leave the node with a negative number of cars. Therefore, we apply a ReLu to obtain $\tilde{o}$ as the observed vehicles adjusted with our

entering/leaving prediction: $\tilde{o} = ReLU(H + o)$.

4.4 Attention-based Flow Distribution

Given the embeddings of adjacent nodes i and j, the ratio of vehicles to be moved from one node to another is computed as an attention score. Note that in the common GCN the convolution operation is applied to learn a new node-level representation, while in FAN the convolution operation is applied to obtain a flow probability. First layer for attention score is computed using the embeddings of the two nodes:

$$e_{i,j}^{(0)} = ReLU(\vec{a}^T(e_i|e_j)) \tag{9}$$

Then the current observation of vehicles on the two nodes is taken into account in the next layer, to capture the dependency between occupancy of the adjacent roads and the flow between them

$$e_{i,j}^{(1)} = ReLU(W(e^{(0)}|\tilde{o}_i|\tilde{o}_j) + b) \tag{10}$$

$$e_{i,j}^{(2)} = ReLU(W(e^{(1)}) + b) \tag{11}$$

$$z_{i,j} = C_{i,j} \cdot e_{i,j}^{(2)} \tag{12}$$

The last layer is required to take into account the **control** matrix C. The control-matrix is an adjacency matrix updated with the current control actions. That is, if a phase of a traffic light controlling the connection between node i and node j is red, this will be denoted by having i, j-entry equal to zero.

$$C_{ij} = \begin{cases} 0 & i \rightarrow j \text{ connection is a phase with red light} \\ \tilde{A}_{ij} & otherwise \end{cases}$$

Note that the adjacency matrix is a fixed input in Graph Neural Networks and so it is in the FAN model, in fact A is still used in the convolutional operation, however the resulting predicted movements are blocked by the new topology, the one enforced by the control matrix C. The control-matrix is crucial both for dynamically modifying the graph connections during the training of the FAN, and for predicting queue lines for control purposes. For example, all of the phases of a traffic light can be set to red in order to predict the number of cars that the traffic light will face on each direction.

Finally, the unnormalized attention score $z_{i,j}$ is normalized by applying a softmax on all the outgoing connections of each node, including the self-loop.

$$\phi_{i,j} = \frac{e^{z_{i,j}}}{\sum_{\forall k|A_{i,k}=1} e^{z_{i,k}}} \tag{13}$$

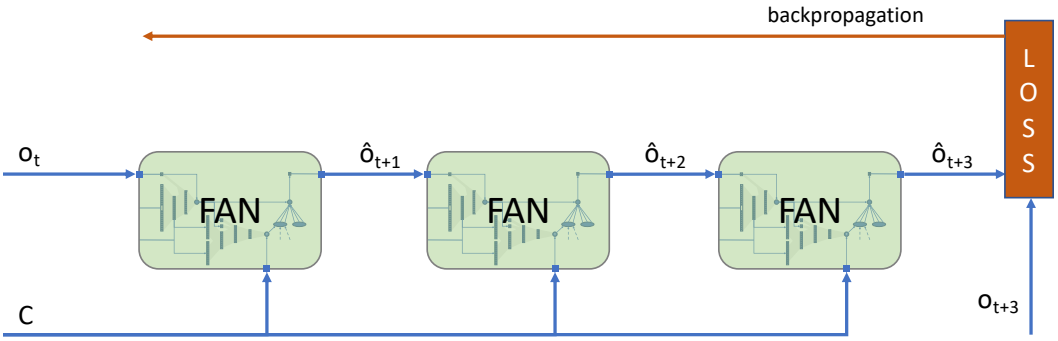

Fig. 6. Recurrent steps applying multiple FAN convolutions to the previous predicted observation. Recurrent steps enable to predict movements through multiple consecutive road segments.

For example, given a node i and its outgoing connections and the self-loop connection, $i \rightarrow m$, $i \rightarrow p$, $i \rightarrow q$ and $i \rightarrow i$, the flow probabilities are distributed among the 4 connections, and it sum is equal to 1.

The complete prediction of the vehicles on all nodes in the road network, given the current observation updated with entering and leaving vehicles, $\tilde{o}^t$, is

$$o^{t+1} = \tilde{o}^{t^T} \Phi \tag{14}$$

4.5 Recurrent predictions

Given the current observation, the FAN predict the observation of the next time-step. Both a training and inference time this can be performed in a recurrent fashion. Successive applications of the FAN effectively convolve the k^{th}-order neighborhood of a node, where k is the number of successive prediction operations, given that there is only one convolutional layer in the network. In other words, by using the current prediction as an input of the subsequent prediction, it is possible to obtain multi-steps prediction at inference time, as shown in. Figure 6. At training time, the loss function on the prediction is applied on the output after multiple steps, and the backpropagation can run back from the last step to the initial observation.

The recurrence of the FAN is very important as it enables for multi-hop predictions. Consider the case in which a vehicle is fast enough to traverse more than one node in line in the duration of a time step, where each traversed node is a short road segment. In this case the movement of the vehicle can only be predicted for the first subsequent node in line, therefore the predicted node location of the vehicle would stay behind with respect to the actual one. Conversely, by applying multiple recurrent steps during one time step, such multi-hop prediction becomes possible.

4.6 FANPressure

In the phase selection process, we first predict the state of the traffic after k steps, then we apply the Max Pressure algorithm to this predicted state. However one problem emerges when combining this two steps: in order to produce a prediction using the FAN model, we need to know what is the current topology of the network, that is modified by the control matrix C, which in turn requires

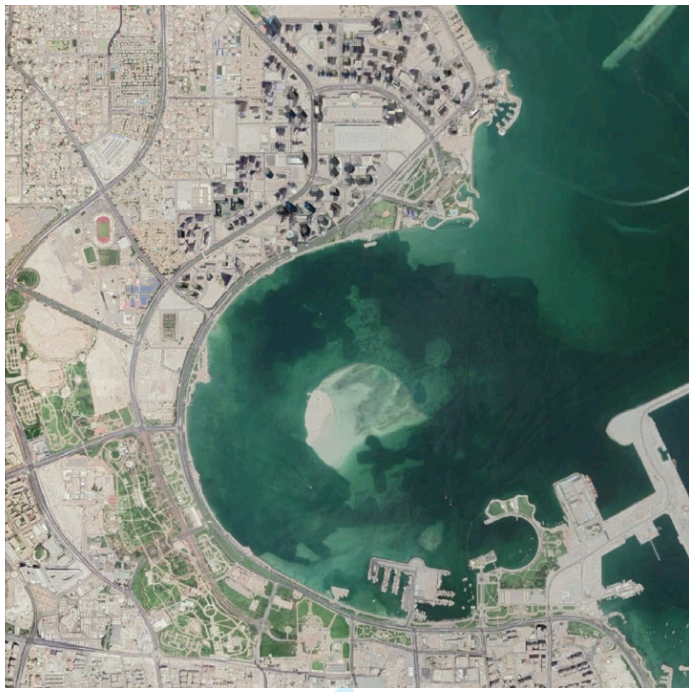

Fig. 7. Aerial view of the road network used for evaluation, including an area of 30 km^2 with 29 traffic lights.

to know the traffic lights phases for the following period. Because the selection of the traffic lights phase is our goal, we would have to make a prediction for each possible combination of phases. In a scenario with multiple intersection, this will quickly incur in a curse of dimensionality problem.

Instead of predicting a different state for each possible phase selection, we proceed as follow:

(1) The traffic light is set to an *all red* phase. That is, for each phase $i \rightarrow j$ of the traffic light TL, we set the relative control matrix item $C_{ij} = 0$.
(2) A multi-step prediction is performed through the recurrent FAN, with vehicles eventually reaching and stopping at each node i that face the traffic light.
(3) The predicted pressure of each phase is computed as the difference between the predicted count of vehicles in the source nodes and the current count of vehicles in the destination nodes.
(4) Phase with the maximum predicted pressure is selected for the next time step.

The *all-red* trick converts all the approaching roads to dead-end nodes, so that all the vehicles that would pass through the node are instead accumulated, avoiding multiple computation for each phase of the traffic light.

5 EXPERIMENTS

We compare the proposed method against the following baselines:

- **Fixed time-policy**: in this policy the phase cycle is fixed thorough the day, however the timing of each phase is designed by an expert to (i) accommodate traffic volume ratios and (ii) prioritize arterial traffic for green-waves.

Traffic Volume	Method	Avg Cumulative Waiting Time	Avg. Queue	Max Queue	Avg. CO_2 [kg]	Max CO_2 [kg]
High	Fixed time policy	6 months	4286	8393	11644	22069
	Max Pressure	11 months	6074	8798	16112	23118
	Max Pressure ϵ-greedy	8 weeks	1108	6462	3710	16980
	FANPressure	**3 days**	**90**	**485**	**1124**	**3058**
Low	Fixed time policy	12 m 42 s	44	146	568	1321
	Max Pressure	11 m 3 s	18	68	489	**1144**
	Max Pressure ϵ-greedy	**6 m 16 s**	21	85	500	1183
	FANPressure	10 m 49 s	**15**	**64**	**474**	1148

Table 1. Performance comparison for traffic light control methods over a 24 hours simulation. Avg Cumulative Waiting Time refers to the sum of cumulated waiting time over all the vehicles in the network, averaged across all timesteps. Queue is the total number of halted vehicles (speed less than 0.1 m/s).

- **Max Pressure**: the original Max Pressure algorithm. In order to compute the pressure, we consider the vehicle counts on the incoming and outgoing lanes directly connected to the intersection.
- **Max Pressure ϵ-greedy**: a non-greedy version of the Max Pressure algorithm. At each time-step, it gives green to the highest pressure phase with the $1-\epsilon$ probability, and pick a random phase with ϵ probability. In the following experiments, a random phase is picked with a probability $\epsilon = 0.2$.

The FAN model is implemented using the Deep Graph Library [43] and PyTorch. Source code with hyperparameters settings and resources needed for reproducibility are publicly available[1]. The model is first trained on several episodes of simulated traffic until no further improvement in prediction is obtained: we noticed a small difference in micro-forecasting accuracy between the episode 18 and the episode 100, therefore we used the FAN network trained for 18 episodes. Each training episode spans for 24 hours. The traffic used for training is a low traffic setting, representing the traffic volume of typical day. The same fixed FAN model is then used for testing also on a high traffic volume setting. We train the model in a recurrent fashion with 4 recurrent steps. Every 10 seconds, the output of the 4th step is compared with the current observation of vehicles counts, to compute the loss of the model. In this way, the FAN is trained to forecast the observation after 10 seconds, while each recurrent step predicts the next 2.5 seconds.

5.1 Data

We evaluate the model on real data, using the official road network of Qatar, by cropping a sub-area of 30 km^2 with 29 traffic lights, related to the Corniche area in Doha, shown in Figure 7. The GIS Shapefile is loaded into the SUMO micro-simulator [6]. Each traffic light has variable number of adjacent road segments, thus a variable number of phases, each one corresponding to a set of non-conflicting movements in the intersection. For each transition between any pair of phases, we pre-defined and enforced a yellow phase to allow the vehicles to safely traverse the junction. The corresponding graph has 1868 nodes, one for each road segment in the network, and 4670 edges, one for connection between road segments, excluding the self-loop connections. The traffic data, replayed in SUMO, is extracted from a collection of taxi vehicles trajectories, recorded for 24

[1]Full source code can be found here: `https://github.com/Strizzo/FANPressure`

Fig. 8. Complete graph of the considered area with 1868 road segments (nodes) and 4670 junctions (edges).

hours during a weekday. The volume of the vehicles is proportionally augmented to reach high congestion levels: in the high congestion setting, the taxi traffic volume is scaled by a factor of 9, while in the low congestion setting, this is scaled by a factor of 5.

5.2 Results

The results of the evaluation of FANPressure, in comparison with the implemented baselines, are shown in Table 1. The following measures are evaluated:

- Average Cumulative Waiting Time. At each timestep, the cumulative waiting time of each vehicle is recorded. We first sum the cumulative waiting time for all the vehicles in the network in the specific timestep, then we average this cumulative value for each time step over the whole episode of 24 hours.
- Average and Maximum Queue. The queue is the number of vehicles that are halted in the whole network at a specific timestep. We show the average and the maximum for the whole episode.
- Average and Maximum CO_2. The CO_2 emissions, estimated by SUMO, are the total emissions in the network. Average and maximum over the 24 hours are shown.

The FANPressure method shows a drastic improvement in presence of high congestion. This advantage is due to the method's capability of considering the volume of vehicles in the network, and discharging flows that are accumulating in connected road segments to a higher degree. The

other methods instead, can only consider the volume of vehicles directly facing the traffic light. When the traffic volume is lower the effect of the flow prediction is less noticeable in the whole period of 24 hours. With no congestion, the method produces comparable waiting times, while having the lowest CO_2 emissions and accumulated queues.

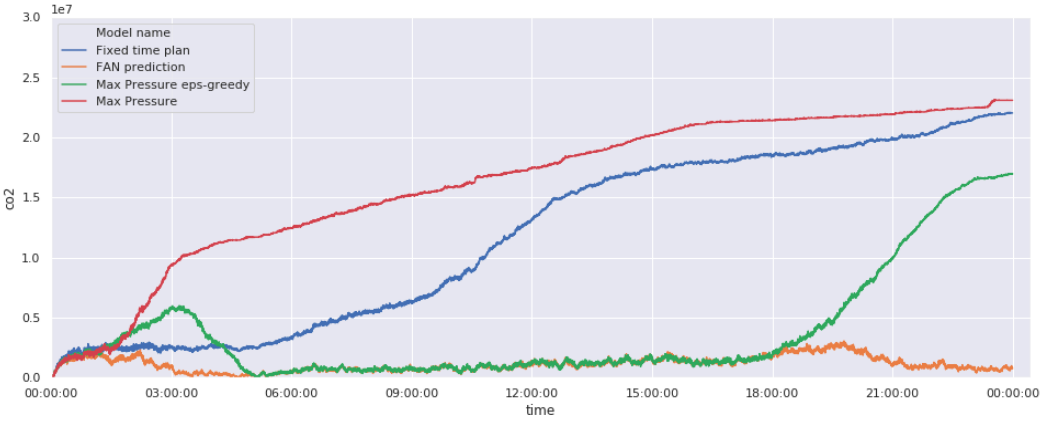

Fig. 9. CO_2 emissions measure over an interval of 24 hours in high traffic volume conditions. When congestion is reached, CO_2 emissions increase drastically as vehicles are halted in queue. FANPressure manages to avoid congestion until the end of the episode, with significant reduction in CO_2.

The evolution of CO_2 emissions is shown in Figure 9. Level of CO_2 are proportional to the number of halted cars. The plot shows how, once a critical point of congestion is reached, it is hard to return to a normal flow. Max Pressure ϵ-greedy is able to revert the congestion levels back to a stable situation at `t=03:00:00`, until new peaks of traffic are reached at `t=18:00:00`. During the interval `05:00:00 - 18:00:00`, FANPressure follows the same trend as Max Pressure ϵ-greedy, nevertheless it avoids entering a critical congestion, as a consequence the emissions level are kept low until the end of the episode.

We also evaluate the effect on traffic of using a different number of recurrent steps in the prediction. The experiments are performed on the high-level congestion scenario. In Figure 10 we show the state of the traffic with four different models at the same time step of the 24 hours simulation. In the figure, vehicles are rendered with a different color depending on their current speed: red vehicles are halter or low speed vehicles, yellow and green are vehicles with medium speed, blue are vehicles traveling at the maximum speed. Using $\epsilon-$greedy Max Pressure without any prediction, most of the traffic is at a low speed or halted state. This improves when the prediction from the FAN model is used instead, with 2 steps of the recurrent FAN. The state of traffic improves even more as we increase the number of recurrent step, while the duration of the complete time step is fixed to 10 seconds. Note that as the speed of vehicles decrease, the total number of vehicles in the network increases, since it takes more time for each trip to reach the destination.

Finally, the FANPressure approach exhibits promising generalization properties, since the results obtained on the high congestion dataset are a great improvement over the baselines, despite the FAN network being trained on a low traffic volume simulation.

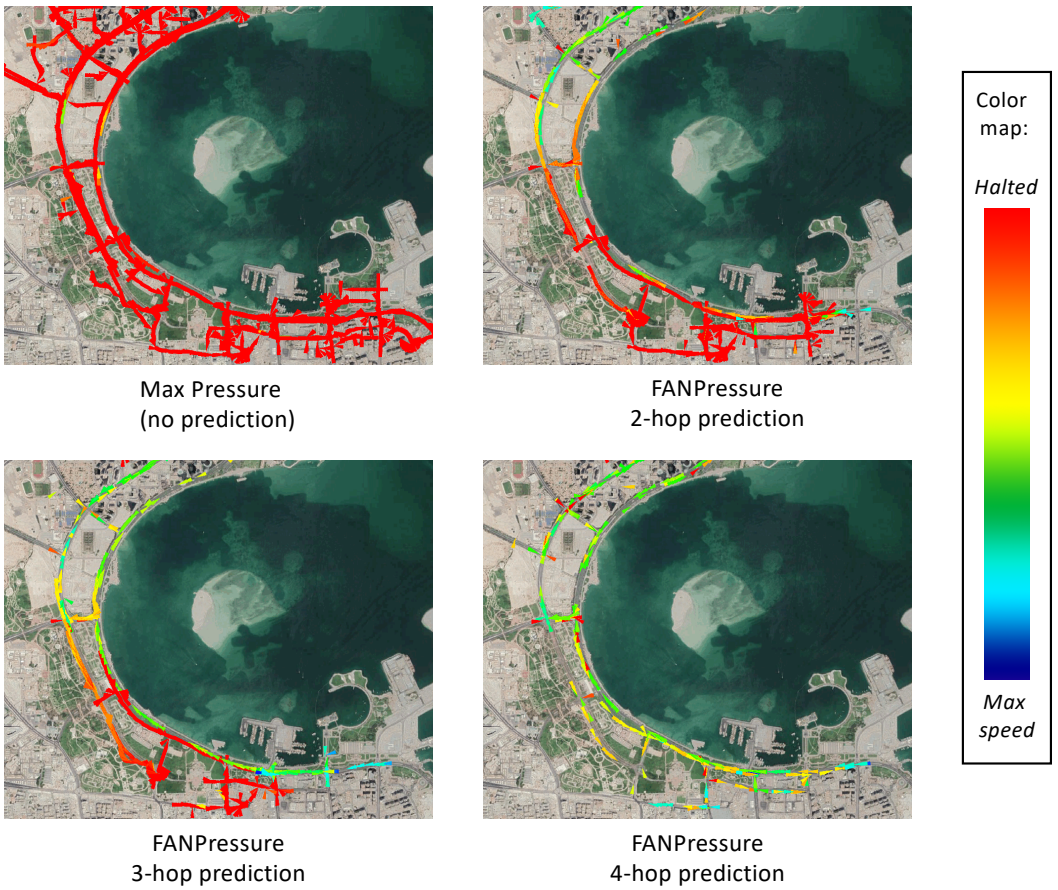

Fig. 10. State of the traffic for a fixed time step in the simulation (20:50:00), showing the speed of vehicles using a color map from red (halted vehicles) to blue (vehicles at maximum speed). The traffic gradually improves from Max Pressure (no prediction) to multi-hop prediction, where one hop is one recurrent application of the FAN model.

6 CONCLUSION

This paper introduces a new deep learning architecture for vehicle flow micro-forecasting. The overall method, named FANPressure, combines the vehicle flow prediction with the effectiveness of Max Pressure control.

The presented problem of micro-forecasting is tailored for the traffic light optimization problem, where short-term prediction are required to take into account approaching vehicles from connected roads. This new type of forecasting differs from the common task of traffic flow forecasting, where the traffic flow, measured as number of vehicles per time unit and average speed, is predicted with a much coarser granularity. Solutions for the classic traffic flow forecasting problem do not well adapt to the high-frequency dynamics of traffic light control, where a quick reaction is needed to

near future, near distance observations.

Using Graph Convolutional Network with attention mechanisms as a generic framework, we have designed a new Deep Learning architecture, named Flow Attention Network, able to learn node embeddings for each road segment in the road network and to perform recurrent steps of vehicles flowing through the graph, predicting the vehicles movement in the presence of a junction. The model takes into account the complexities of fine grain flows of traffic, including the prediction of vehicles that enter or leave the road network, and the possibility of dynamically change the topology of the graph both at training and inference time, allowing the input of a control matrix to modify the original adjacency matrix. At training time, the latter feature has been used to reflect the current status of network including the traffic light states. At inference time, the control matrix is used to stop the incoming traffic during the prediction step, in order to count the number of vehicles that would pass through each incoming lane during the next time step.

The Flow Attention Network micro-forecasting model is combined with a well-known control algorithm, Max Pressure, to select the phase that minimize the overall pressure at each intersection. FANPressure is the resulting prescriptive-learning model, combining prediction and control. The MaxPressure algorithm can only take into account the current number of vehicles, without considering the vehicles incoming in the near future. Conversely, FANPressure takes this into account, as the pressure considered is the micro-forecast pressure.

FANPressure is evaluated on a real dataset, using official updated city road network data, and reproducing real recorded traffic from a collection of vehicles trajectories. The results show the importance of flow prediction in particular for high volumes of traffic, where greedy local decisions may lead to a congested network with large-scale effects. FANPressure is able to anticipate high volumes of traffic incoming in the near feature, therefore to avoid that this traffic clogs the dense urban networks. Even when the traffic volume is low, FANPressure leads to reduction in queue lengths and consequently emissions. FANPressure has also shown good generalization properties, when tested on high volume traffic while being trained on low level traffic. We are working with the local government to test FANPressure in a real environment consisting of multiple intersections.

REFERENCES

[1] Hua Wei, Guanjie Zheng, Vikash Gayah, and Zhenhui Li. A survey on traffic signal control methods, 2019.

[2] Pravin Varaiya. Max pressure control of a network of signalized intersections. *Transportation Research Part C: Emerging Technologies*, 36:177–195, 2013.

[3] Hua Wei, Chacha Chen, Guanjie Zheng, Kan Wu, Vikash Gayah, Kai Xu, and Zhenhui Li. Presslight: Learning max pressure control to coordinate traffic signals in arterial network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1290–1298, 2019.

[4] Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018.

[5] Stefano Giovanni Rizzo, Giovanna Vantini, and Sanjay Chawla. Reinforcement learning with explainability for traffic signal control. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3567–3572. IEEE, 2019.

[6] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[7] Hao Chen, Ke Yang, Stefano Giovanni Rizzo, Giovanna Vantini, Phillip Taylor, Xiaosong Ma, and Sanjay Chawla. Qarsumo: A parallel, congestion-optimized traffic simulator. In *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*, pages 578–588, 2020.

[8] Tanwi Mallick, Prasanna Balaprakash, Eric Rask, and Jane Macfarlane. Graph-partitioning-based diffusion convolution recurrent neural network for large-scale traffic forecasting, 2019.

[9] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018.

[10] Brian L Smith and Michael J Demetsky. Traffic flow forecasting: comparison of modeling approaches. *Journal of transportation engineering*, 123(4):261–266, 1997.

[11] Anthony Stathopoulos and Matthew G Karlaftis. A multivariate state space approach for urban traffic flow modeling and prediction. *Transportation Research Part C: Emerging Technologies*, 11(2):121–135, 2003.

[12] Nikolay Laptev, Jason Yosinski, Li Erran Li, and Slawek Smyl. Time-series extreme event forecasting with neural networks at uber. In *International Conference on Machine Learning*, volume 34, pages 1–5, 2017.

[13] Rose Yu, Yaguang Li, Cyrus Shahabi, Ugur Demiryurek, and Yan Liu. Deep learning: A generic approach for extreme condition traffic forecasting. In *Proceedings of the 2017 SIAM international Conference on Data Mining*, pages 777–785. SIAM, 2017.

[14] Nicholas G Polson and Vadim O Sokolov. Deep learning for short-term traffic flow prediction. *Transportation Research Part C: Emerging Technologies*, 79:1–17, 2017.

[15] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, 2014.

[16] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 5308–5317, 2016.

[17] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 922–929, 2019.

[18] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in neural information processing systems*, pages 1993–2001, 2016.

[19] B. De Schutter and B. De Moor. Optimal traffic light control for a single intersection. *European Journal of Control*, 4(3):260 – 276, 1998.

[20] Zhiyong Liu. A survey of intelligence methods in urban traffic signal control. *IJCSNS International Journal of Computer Science and Network Security*, 7(7):105–112, 2007.

[21] Kenneth Tze Kin Teo, Wei Yeang Kow, and YK Chin. Optimization of traffic flow within an urban traffic light intersection with genetic algorithm. In *Computational Intelligence, Modelling and Simulation (CIMSiM), 2010 Second International Conference on*, pages 172–177. IEEE, 2010.

[22] Michele Covell, Shumeet Baluja, and Rahul Sukthankar. Micro-auction-based traffic-light control: Responsive, local decision making. In *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*, pages 558–565. IEEE, 2015.

[23] Marco Wiering. Multi-agent reinforcement learning for traÆc light control. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1151–1158, 2000.

[24] Li Chun-Gui, Wang Meng, Sun Zi-Gaung, Lin Fei-Ying, and Zhang Zeng-Fang. Urban traffic signal learning control using fuzzy actor-critic methods. In *Natural Computation, 2009. ICNC'09. Fifth International Conference on*, volume 1, pages 368–372. IEEE, 2009.

[25] PG Balaji, X German, and Dipti Srinivasan. Urban traffic signal control using reinforcement learning agents. *IET Intelligent Transport Systems*, 4(3):177–188, 2010.

[26] Lior Kuyer, Shimon Whiteson, Bram Bakker, and Nikos Vlassis. Multiagent reinforcement learning for urban traffic control using coordination graphs. In Walter Daelemans, Bart Goethals, and Katharina Morik, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 656–671, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[27] Juntao Gao, Yulong Shen, Jia Liu, Minoru Ito, and Norio Shiratori. Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network. *arXiv preprint arXiv:1705.02755*, 2017.

[28] X. Liang, X. Du, G. Wang, and Z. Han. A deep q learning network for traffic lights' cycle control in vehicular networks. *IEEE Transactions on Vehicular Technology*, pages 1–1, 2019.

[29] Baher Abdulhai, Rob Pringle, and Grigoris J. Karakoulas. Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering*, 129(3):278–285, 2003.

[30] Wade Genders and Saiedeh Razavi. Using a deep reinforcement learning agent for traffic signal control. *arXiv preprint arXiv:1611.01142*, 2016.

[31] Li Li, Yisheng Lv, and Fei-Yue Wang. Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica*, 3(3):247–254, 2016.

[32] Seyed Sajad Mousavi, Michael Schukat, and Enda Howley. Traffic light control using deep policy-gradient and value-function-based reinforcement learning. *IET Intelligent Transport Systems*, 11(7):417–423, 2017.

[33] Hua Wei, Guanjie Zheng, Huaxiu Yao, and Zhenhui Li. Intellilight: A reinforcement learning approach for intelligent traffic light control. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 2496–2505, New York, NY, USA, 2018. ACM.

[34] Stefano Giovanni Rizzo, Giovanna Vantini, and Sanjay Chawla. Time critic policy gradient methods for traffic signal control in complex and congested scenarios. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1654–1664, 2019.

[35] Hua Wei, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Zhang, Yanmin Zhu, Kai Xu, and Zhenhui Li. Colight: Learning network-level cooperation for traffic signal control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1913–1922, 2019.

[36] Yanan Wang, Tong Xu, Xin Niu, Chang Tan, Enhong Chen, and Hui Xiong. Stmarl: A spatio-temporal multi-agent reinforcement learning approach for cooperative traffic light control. *IEEE Transactions on Mobile Computing*, 2020.

[37] Chenguang Zhao, Xiaorong Hu, and Gang Wang. Pdlight: A deep reinforcement learning traffic light control algorithm with pressure and dynamic light duration. *arXiv preprint arXiv:2009.13711*, 2020.

[38] Pengyuan Zhou, Xianfu Chen, Zhi Liu, Tristan Braud, Pan Hui, and Jussi Kangasharju. Drle: Decentralized reinforcement learning at the edge for traffic light control. *arXiv preprint arXiv:2009.01502*, 2020.

[39] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1282–1289, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[40] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[41] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[42] David K Smith. Network flows: theory, algorithms, and applications. *Journal of the Operational Research Society*, 45(11):1340–1340, 1994.

[43] Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, et al. Deep graph library: Towards efficient and scalable deep learning on graphs. *arXiv preprint arXiv:1909.01315*, 2019.