SDGNet: A Handover-aware Spatiotemporal Graph Neural Network for Mobile Traffic Forecasting

Yini Fang, Salih Ergüt, and Paul Patras, *Senior Member, IEEE*

***Abstract*—Accurate mobile traffic prediction at city-scale is becoming increasingly important as data demand surges and network deployments become denser. How mobile networks and user mobility are modelled is key to high-quality forecasts. Prior work builds on distance-based Euclidean (grids) or invariant graph representations, which cannot capture dynamic spatiotemporal correlations with high fidelity. In this letter we propose SDGNet, a handover-aware spatiotemporal graph neural network that hinges on Dynamic Graph Convolution and Gated Linear Units to predict traffic consumption over short, medium and long time-frames. Experiments with a real-world dataset demonstrate SDGNet outperforms state-of-the-art neural model, attaining up to 4× lower prediction errors.**

***Index Terms*—mobile traffic forecasting, graph neural networks, deep learning**

I. Introduction

The total global monthly mobile data traffic consumption exceeded 66 exabytes (EB) in Q1/2021 and is expected to surpass 300 EB across different mobile technologies combined by 2026 [1]. This continual rise in demand poses evermore pressing challenges on mobile networks and highlights the need for accurate traffic prediction as a driver for intelligent management of resources, reduced operational expenditure, and smaller network carbon footprint. For instance, traffic forecasts can enable mobile operators to switch off parts of their infrastructure or measurement equipment in order to save both energy and cost. On the other hand, anticipating traffic surges allows provisioning the right resources in advance, so as to maintain user quality of experience.

Making highly-accurate mobile traffic predictions at city-scale is however challenging, as services continues to diversify and exhibit highly dynamic patterns, while spatiotemporal correlations across different parts of a deployment are not straightforward to elucidate given unpredictable user mobility or terrain/coverage irregularities. Previous forecasting solutions build on Euclidean distance-based representations of RAN layouts, which are mapped onto grids, before employing Long Short Term Memories (LSTMs) [2], Convolutional Neural Networks (CNNs) [3], or a combination of these [4], [5]–[7]. Such approaches *(i)* ignore that there may be multiple base stations at each grid cell, each with multiple sectors and multiple carriers/sector, and users are often handed off between these, sometimes among different technologies (e.g. from 4G to 3G), and *(ii)* relationships between these stations change over time.

To be able to capture inter-sector/-base station dependencies at different timestamps and produce more accurate forecasts, in this work we use graphs to represent cellular networks, where vertices correspond to antenna sectors, and weights quantify the dependency between. Recent studies on Graph Convolutional Network (GNN) model such dependency as the distance between two locations [8]; however, location proximity of base stations may not reflect strong dependency due to terrain constraints (users connected to a base station in a valley are unlikely to be handed off to another one on the other side of an edge, but may be handed off to a base station across a stretch of water, given favourable signal propagation conditions), while distant base stations may exhibit strong dependencies because of user daily commute patterns (e.g. along a motorway). Therefore we harness fine-grained handover frequency information, which reflects user mobility to some extent and better captures traffic dynamics.

As such, we propose **SDGNet (Spatiotemporal Dynamic Graph Network)**, a handover-aware spatiotemporal graph neural network for mobile traffic forecasting. We model the cellular network as a directed and weighted dynamic graph, which captures spatiotemporal correlations from both traffic consumption and handover frequency. In particular, vertices keep information about the volume of traffic at a particular sector and time, while the weights are handover frequencies between vertices over fixed observation windows. We leverage Gated Linear Units (GLU) to extract temporal features, and unlike RNN-based models, output traffic predictions at all the base stations in a deployment simultaneously, while the structure is faster to train. We propose a novel approach to dynamic graph convolution, combining dynamic graph spectral convolution and diffusion graph convolution, thereby capturing spatial correlations both short- and long-term. We validate SDGNet with a real-world mobile traffic dataset, demonstrating that our model achieves traffic forecasting with up to 75.2%, 59.4% and 56.0% higher short-, mid- and resepectively long-term accuracy in terms of Mean Absolute Error (MAE), compared with a range of benchmarks.

The rest of the letter is organised as follows: §II formalised the problem tackled, §III gives a brief introduction to graph convolution, §IV presents our SDGNet architecture, §V demonstrates the performance of our solution, §VI overviews related work, and §VII concludes the letter.

II. Problem Formulation

Consider a mobile network deployment comprising N base stations. We denote by $\mathbb{F}_t = \{F_{t-T+1}, ..., F_t\} \in \mathbb{R}^{T\times N}$ a sequence of mobile traffic consumption measurements over T

Y. Fang and P. Patras are with the School of Informatics, University of Edinburgh, UK. P. Patras is also with Net AI, UK. E-mail: {yini.fang,paul.patras}@ed.ac.uk. S. Ergüt is with OREDATA, Turkey. E-mail: salih.ergut@oredata.com This work was partially supported by Cisco through the Cisco University Research Program Fund (Grant no. 2019-197006).

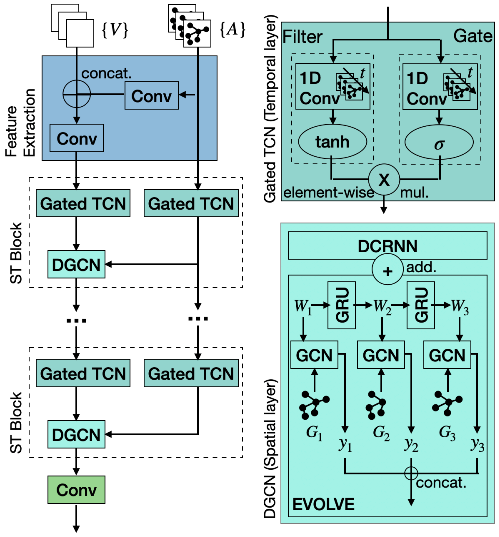

Fig. 1: Proposed SDGNet structure consisting of a feature extraction block, several ST blocks, and a readout 2D Conv layer (left). ST blocks consist of 2 temporal and 1 spatial layer (the structure of each shown on the right).

timestamps up to the current time t, where F_t is the traffic snapshot across all base stations, observed over an interval $[t-\Delta, t]$, i.e., $F_t = \{f_1^t, ..., f_N^t\}$, where f_i^t is the volume of traffic at the i-th base station, and Δ is the temporal granularity of traffic observations configurable by a network administrator.

The mobile traffic network is represented as a directed, weighted and dynamic graph. At the t-th time step, we define a graph $G_t = (v_t, A_t)$, where $v_t \in \mathbb{R}^{N\times C}$ is the graph signal and C is the number of features (i.e., traffic consumption and time information), and $A_t \in \mathbb{R}^{N\times N}$ is the adjacency matrix where an element $a_{i,j}^t$ represents the handover frequency between base stations i and j observed at that time.

Our objective is to predict the most likely mobile traffic consumption in the next H time steps, given the past T observations, i.e.,

$$\hat{F}_{t+1}, .., \hat{F}_{t+H} = \\ \underset{v_{t+1},\dots,v_{t+H}}{\arg\max} \; \log P\left(F_{t+1}, ..., F_{t+H} \mid G_{t-T+1}, \dots, G_t\right).$$

III. Graph Convolution Primer

Bruna et al. propose applying graph convolution (GCN) in the Fourier domain [9], with applications to e.g. molecular modelling [10]. The spectral graph convolution operator $*\mathcal{G}$ is defined as the multiplication of a graph signal $x \in \mathbb{R}^n$ and adjacency matrix $A \in \mathbb{R}^{n\times n}$ with a kernel $g \in \mathbb{R}^n$, i.e.

$$x * \mathcal{G}g = g(L)x = g(U\Lambda U^T)x = Ug(\Lambda)U^T x,$$

where $L \in \mathbb{R}^{n\times n}$ is the normalised graph Laplacian, Λ is the diagonal matrix of eigenvalues of L, and the graph Fourier basis $U \in \mathbb{R}^{n\times n}$ is the matrix of eigenvectors of L. The transition is based on the equation $L = I_n - D^{-\frac{1}{2}}AD^{-\frac{1}{2}} = U\Lambda U^T$, where I_n is an identity matrix and $D \in \mathbb{R}^{n\times n}$ is the diagonal degree matrix with $D_{ii} = \Sigma_j A_{ij}$.

Kipf and Welling propose a first-order approximation of spectral graph convolution [11], which aggregates and exploits a node's neighbourhood information, given by:

$$x * \mathcal{G}g = \theta(I_n + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})x,$$

where $\theta \in \mathbb{R}^K$ are polynomial coefficients, given that the filter function $g(\Lambda)$ can be expressed by a function of a polynomial of Λ. This optimization localizes the filter and significantly reduces the computational complexity.

In contrast, Diffusion Graph Convolution (DCRNN) models information flow as a diffusion process, and captures spatial dependencies using bidirectional random walks on the graph [12]. The DCRNN operator $*\mathcal{D}$ over a graph signal $x \in \mathbb{R}^n$ and adjacency matrix $A \in \mathbb{R}^{n\times n}$ with a kernel $g \in \mathbb{R}^n$ is:

$$x * \mathcal{D}g = \sum_{k=0}^{K}(P_f^k x\theta_{k,1} + P_b^k x\theta_{k,2}),$$

where K is the number of finite steps in the diffusion process, $\theta \in \mathbb{R}^{K\times 2}$ are the parameters of the filter and P^k represents the power series of the transition matrix. For directed graphs, the diffusion process has a forward and backward direction, with forward transition matrix $P_f = A/\operatorname{rowsum}(A)$ and backward transition matrix $P_b = A^T/\operatorname{rowsum}(A^T)$.

Wu et al. propose Graph WaveNet [13] to produce road traffic predictions, which uses a self-adaptive adjacency matrix to learn hidden spatial dependencies. The graph convolution layer in Graph WaveNet applies DCRNN for pre-defined spatial dependencies and spectral graph convolution for self-learned hidden dependencies. This model refines the graph topology automatically in the case that such dependency information is difficult to model or obtain.

IV. SDGNet

Inspired by Graph WaveNet [13], we propose SDGNet, a deep neural network that solves the mobile traffic forecasting problem posed in Sec. II. SDGNet captures spatiotemporal correlations among traffic consumption at different locations and dynamic adjacency matrices modelled from handover data. The proposed model consists of a feature extraction block followed by several spatiotemporal (ST) blocks, as shown in Fig. 1. Each ST block comprises two temporal layers that handle graph signals and adjacency matrices, and a spatial layer for dynamic graph convolution. In what follows we explain in detail the inner workings of these different modules.

a) Spatiotemporal feature extraction: The first block generates feature maps for the next module by capturing spatiotemporal correlations from graph signals $\mathbf{V} \in \mathbb{R}^{T\times N\times C}$ and dynamic adjacency matrix $\mathbf{A} \in \mathbb{R}^{T\times N\times N}$. The feature dimension of $\mathbf{A}$ is first reduced by a Conv layer before concatenating with $\mathbf{V}$, so that features pertaining to $\mathbf{A}$ do not dominate in the concatenated matrix. We then pass the results through another Conv layer to extract feature maps that will be handled by the subsequent spatiotemporal block.

b) Gated TCN: Each ST blocks encompasses two gated temporal convolution networks (TCNs) and a Dynamic Graph Convolution Network (DGCN). We adopt gated 1-D dilated causal convolution [13] as Temporal convolution layer to capture complex temporal dependencies. Dilated causal convolution can handle long-term sequences in a non-recursive manner and converges faster compared to traditional RNN-based models. The structure works by sliding over inputs and skipping elements with a given step, which increases with the layer depth. Given an input $\mathbf{x} \in \mathbf{R}^T$ and filter $\mathbf{g} \in \mathbf{R}^K$, the dilated causal convolution operation step is represented as:

$$\mathbf{x} \star \mathbf{g}(t) = \sum_{s=0}^{K-1} \mathbf{g}(s)\mathbf{x}(t - d \times s),$$

where d is the dilation factor determining the length of the skipping step. Several dilated causal convolution layers are stacked with increasing receptive fields.

A gating mechanism is applied to control the information flow through layers [14], leading to the following output:

$$\mathbb{H}(\mathbf{x}) = z\left(\mathbf{x} \star \mathbf{g}_1(t) + \mathbf{b}\right) \odot \sigma\left(\mathbf{x} \star \mathbf{g}_2(t) + \mathbf{c}\right),$$

where $\mathbf{b}$ and $\mathbf{c}$ are model weights, $\odot$ is the element-wise multiplication, $z(\cdot)$ is an activation function, and $\sigma(\cdot)$ is the sigmoid function which controls the information passed to the next layer. We apply Gated TCNs on both inputs $\mathbf{V}$ and $\mathbf{A}$, to learn their temporal dependencies while reducing the temporal dimension of the propagated output.

c) Dynamic Graph Convolution Network: To obtain accurate forecasts both short- and long-term, we combine spectral graph convolution and DCRNN into dynamic graph convolution network. Intuitively, we can adopt spectral graph convolution by applying the multiplication of $\mathbf{x} \in \mathbb{R}^{T \times N \times C}$ and the adjacency matrix $\mathbf{A} \in \mathbb{R}^{T \times N \times N}$ along the first dimension. This acts as T copies of spectral graph convolution, each one working on one snapshot of the sequence. One drawback is that these T copies share one weight, and have no temporal correlations. To circumvent this issue, we adopt EvolveGCN [15], where we assign a weight to each snapshot, and these weights are temporally related. Mathematically, for every snapshot $\mathbf{x}_t$ and its corresponding adjacency matrix $\mathbf{A}_t$,

$$\begin{aligned} \mathbf{x}_t &= \mathbf{x_t} * \mathcal{G}\mathbf{w}_t = \sigma\left(\tilde{\mathbf{A}}_t \mathbf{x}_t \mathbf{w}_t\right), \\ \mathbf{w}_t &= \mathrm{GRU}(\mathbf{w}_{t-1}), \end{aligned}$$

where $\tilde{\mathbf{A_t}} = \mathbf{I_n} + \mathbf{D^{-1/2} A_t D^{-1/2}}$ and $\mathbf{w_t}$ is the weight of t-th snapshot. Each GCN operation has a weight, which is generated from the weight in the last snapshot using a Gated recurrent unit (GRU). The GRU is used to capture long-term temporal correlations. Compared to the Long Short-Term Memory (LSTM), it has fewer gates and therefore is faster to train and uses less memory. We initialize $\mathbf{w}_1$ at the beginning. For the following recurrent steps, we use the last output as both hidden state and the input to the GRU. Finally we concatenate the output from every snapshot as the final output of EvolveGCN. We denote the EvolveGCN operator as $*\mathcal{E}$. We express the DGCN operation in matrix form:

$$\begin{aligned} \mathbb{Y}(\mathbf{x}, \mathbf{A}) =& \mathbf{x} * \mathcal{E}\mathbf{w^e} + \mathbf{x} * \mathcal{D}\mathbf{w^d} \\ =& \|_{t=1}^{T} \mathbf{\tilde{A}_t x w_t^e} + \sum_{k=0}^{K} (\mathbf{P}_f^k \mathbf{x w^d_{k,1}} + \mathbf{P}_b^k \mathbf{x w^d_{k,2}}), \end{aligned}$$

where $\|$ denotes concatenation, $\mathbf{\tilde{A}} = \mathbf{I_n} + \mathbf{D^{-1/2} A D^{-1/2}}$, $\mathbf{w^e} \in \mathbb{R}^{T \times C \times C'}$, $\mathbf{w^d} \in \mathbb{R}^{C \times C'}$. C' is the dimension of the hidden states. Finally, the formula of the l-th ST block given the input graph signal $v^l \in \mathbb{R}^{T \times N \times C'}$ and input adjacency matrix $A^l \in \mathbb{R}^{T \times N \times N}$, is given by:

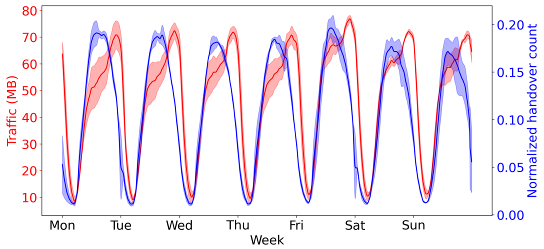

Fig. 2: Avg. and std deviation of weekly traffic consumption and normalised handover count, over the entire deployment.

$$v^{l+1} = \mathbb{Y}(\mathbb{H}(v^l), \mathbb{H}(A^l)); \quad A^{l+1} = \mathbb{H}(A^l).$$

d) Training Loss: We seek to minimise the Mean Square Error (MSE) when training the overall neural model, i.e.

$$\mathbb{L}(\hat{F}_{t+1}, .., \hat{F}_{t+H}) = \frac{1}{H \times N} \sum_{i=1}^{H} \sum_{n=1}^{N} (\hat{f}_n^i - f_n^i)^2,$$

where H is the number of prediction steps and N is the number of base stations in the deployment.

V. Evaluation

We implement SDGNet using PyTorch, empirically choosing a configuration with 8 ST blocks, following experiments with different numbers and aiming to strike a good performance/complexity trade-off. We train using the Adam optimizer with learning rate $\lambda = 10^{-4}$ and a batch size of 128.

A. Dataset

We evaluate our SDGNet using real-world mobile traffic data collected by Turkcell between 5–29 March 2020 in a medium-size city. The commercial-sensitivity of the dataset precluded disclosure of antenna sector locations and its public release. As such, our comparison is confined to graph neural models and excludes solutions that involve location remapping to a grid [2]–[6]. Fig. 2 shows the weekly average and standard deviation of the volume of traffic and respectively the normalised handover count, across the entire deployment. Traffic volume and handover count have peaks at slightly different times due to their different scope: the former reflects consumption and the latter user mobility. To ensure sufficient data samples for training, we augment the dataset via bicubic interpolation, enhancing the granularity from 1 h to 10 min. This enhances the sample size by a factor of 6×, while also aligning with finer time-frames typical of resource allocation decisions in software-based networks. We split the data into training, validation and testing with a 6:2:2 ratio. We confine consideration to 100 base stations that exhibit the highest traffic consumption.

To speed up training, we normalize the traffic consumption data by $\log(1 + x)/\bar{x}$, where x is a traffic consumption measurement and $\bar{x}$ is the mean of $\log(1 + x)$ across all data points. We further normalize the adjacency matrices by a Softmax function (i.e. handovers counts from a base station to other base stations sum up to 1). We add absolute time

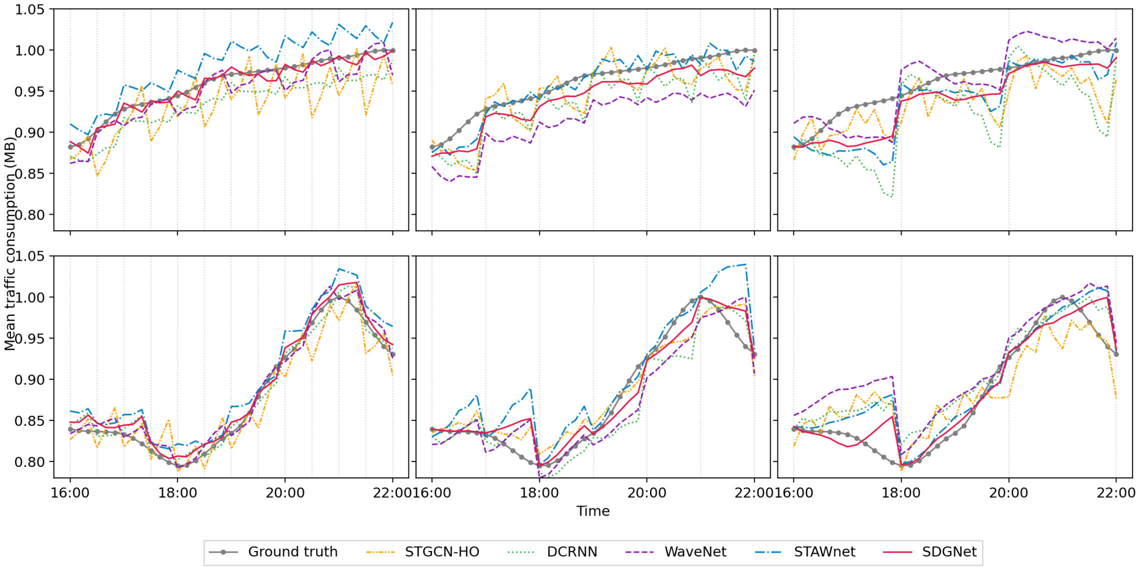

Fig. 3: Normalised traffic predictions between 16:00 and 23:00, averaged over 5 days, when forecasting window is 3 steps (30 min, left), 6 steps (60 min, middle), and 12 steps (120 right), at a heavily-loaded (top) and lightly-loaded (bottom) base station.

TABLE I: Model configurations

Model	Configuration	
	Spatial operation	Temporal operation
STGCN-HO	GCN	1D CNN
DCRNN	DCRNN	Gated TCN
WaveNet	GCN with self-adaptive adjacency matrix +DCRNN	Gated TCN
STAWnet	GCN+DCRNN, both with self-adaptive adjacency matrix	Gated TCN
SDGNet	EvolveGCN+DCRNN	Gated TCN

information as additional features of the graph signals, namely day of the week, hour of the day, and minute of the hour, which we map to the $[-1, 1]$ range. We use historical two-hour windows as inputs to model input, i.e. 12 snapshots.

B. Result

We measure the prediction accuracy of our SDGNet over 3, 6 and 12 steps respectively (30, 60, and 120 min), in terms of mean absolute area (MAE) and root mean squared error (RMSE), which we compare against that of the following graph neural network benchmarks (configurations in Table I):

- STGCN-HO [16] – also models dependencies through handover frequency, but uses the average handover frequency computed over four random days, without capturing any handover dynamics.
- DCRNN [12] – as introduced before, DCRNN models the traffic flow as a diffusion process on a directed graph.
- WaveNet [13] – introduces an adaptive adjacency matrix which is initialised with the adjacency matrix and learned in the training process.
- STAWnet [17] – introduces self-learned node embedding by a dynamic attention mechanism; no prior knowledge of the graph is needed; the adjacency matrix is not used in attention learning.

All models considered are trained and evaluated with the same dataset. The results are summarised in Table II. Observe that our solution outperforms the existing models, reducing the MAE of the best performing benchmark by 6.0% (DCRNN), 7.2% (WaveNet) and 8.3% (STAWnet), when making 3-step, 6-step, and 12-step predictions respectively. Short-term, our SDGNet provides 4× smaller errors than STGCN-HO (which uses fixed adjecncy matrices), while over 2 hours our model offers up to 36% lower MAE.

TABLE II: Forecasting performance comparison

	30 min		60 min		120 min	
Model	MAE	RMSE	MAE	RMSE	MAE	RMSE
STGCN-HO	3.14	4.78	3.23	4.85	4.29	6.12
DCRNN	1.03	1.70	2.17	3.53	3.81	5.92
WaveNet	0.83	1.38	1.99	3.16	3.23	5.03
STAWnet	1.16	1.74	1.85	3.11	3.00	4.84
SDGNet	**0.78**	**1.28**	**1.31**	**2.42**	**2.75**	**4.49**

We delve deeper into the behaviour of SDGNet and that of the benchmarks considered, and plot the predicted traffic consumption using these models between times of the day with low demand (16:00 hrs) and high demand (23:00 hrs), zooming in on a heavily-loaded and lightly-loaded base station and examining different forecasting windows. The results are illustrated in Fig. 3, where dashed line stands indicate the beginning of a prediction window as we apply a sliding window with different numbers of predicted steps – 3 (left), 6 (middle), 12 (right). Observe that DCRNN always underestimates the traffic volume at the heavily-loaded base station, while WaveNet tends to overestimate the traffic consumption at both base stations. STAWnet performs worse when making 6-step predictions at the heavily-loaded base station. STGCN-HO fails to capture the traffic patterns correctly, which results in significant deviations from the ground truth. In contrast, our SDGNet forecasts the traffic consumption accurately irrespective of the load regime or forecasting window length, following closely the ground truth.

Long-term performance: We take a closer look at how the different approaches compare when making long-term predictions. To this end, in Fig. 4 we plot the average MAE at each prediction step of 12-step (120 min) forecasting windows. STGCN-HO performs modestly at the beginning

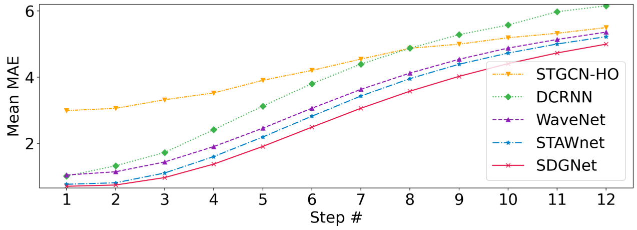

Fig. 4: Average MAE of SDGNet and different benchmarks at each step of 12-step forecasting windows.

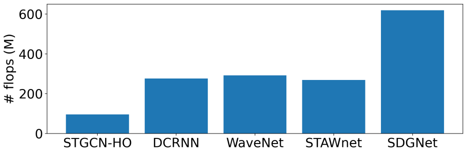

Fig. 5: Time complexity of SDGNet and different benchmarks.

but approaches the performance of most models in the final steps, while DCRNN predicts reasonably well at the beginning but much worse in the final steps. This indicates that spectral graph convolution is helpful for long-term prediction while DCRNN structures benefit short-term forecasting. This is also seen in the behaviour of the other three models where spectral convolution-based and DCRNN are combined. Our SDGNet yields the smallest error at every step, which confirms the benefit of capturing handover dynamics when making predictions.

Time complexity: Fig. 5 shows the number of FLOPs per inference instance of SDGNet and the benchmark models considered. Because SDGNet works with adjacency matrices of larger dimensionality and encompasses several GRUs that process the data sequentially, the accuracy gains it achieves come at the price of higher inference times. However, given that off-the-shelf GPUs routinely handle over 20 TFLOPs per second, the cost is affordable.

VI. Related Work

Applying graph convolution to spatiotemproal tasks is gaining wide interest. Seo et al. introduce a Graph Convolutional Recurrent Network (GCRN) to predict structured sequences of data [18]. The proposed model combines graph convolution operations to identify spatial structures and RNN to capture dynamic patterns. The authors found that simultaneously exploiting graph spatial and dynamic information about data can improve both prediction accuracy and training speed. Based on this principle, Li et al. propose Diffusion Convolutional Recurrent Neural Network (DCRNN) for traffic prediction, modelling traffic flows as a diffusion process [12]. In contrast, Spatiotemporal Graph Convolutional Networks (STGCNs) consists of spatiotemporal convolutional layers and a fully-connected output layer, with a residual connection and bottleneck strategy applied inside each block [19].
where the neighbouring base stations are limited to those in

The performance of the GCN models relies on the structure of the input graph [11], and which makes it difficult to capture the complex spatial dependencies specific to mobile traffic. In graphs, this dependency is reflected in the adjacency matrix, as weight $w(i, j)$ reflect how the traffic from an i-th base station is related to that an j-th. This adjacency is modelled as a function of distance between these two base station in [8], close proximity. This approach harms prediction performance because spatial correlations among traffic also exist between distant base stations [20]. Zhao et al. consider the effect of handover on the spatial characteristics of the traffic and build graphs for base stations based on their handover frequencies [16]. Kalander et al. use hybrid GCNs with three types of binary adjacency matrices: spatial proximity, functional similarity (i.e. similarity of average weekly traffic), and recent similarity trends [21]. This requires substantial memory and becomes impractical as the number of base stations is grows.

VII. Conclusion

In this letter we proposed SDGNet, a handover-aware dynamic graph neural network for mobile traffic prediction. Our architecture captures accurately dynamic spatiotemporal correlations within both mobile traffic consumption and handovers among base stations. We experimented with a real-world mobile traffic data set, demonstrating that our solution outperforms state-of-the art forecasting models and excels in making long-term predictions. Improvements remain to be further explored, such as hybrid graph structures that exploit more detailed information about the cellular network without becoming computationally heavy.

References

[1] "Ericsson Mobility Report," June 2021.

[2] K. Zhang, *et al.*, "A new method for traffic forecasting in urban wireless communication network," *EURASIP J. Wireless Comms & Netw.*, 2019.

[3] C. Zhang, *et al.*, "Citywide cellular traffic prediction based on densely connected convolutional neural networks," *IEEE Comms Let.*, 2018.

[4] C. Zhang and P. Patras, "Long-term mobile traffic forecasting using deep spatio-temporal neural networks," in *ACM MobiHoc*, 2018.

[5] C. Zhang, *et al.*, "Deep transfer learning for intelligent cellular traffic prediction based on cross-domain big data," *JSAC*, vol. 37, no. 6, 2019.

[6] W. Shen, *et al.*, "Time-wise attention aided convolutional neural network for data-driven cellular traffic prediction," *IEEE W. Comms Let.*, 2021.

[7] A. Montieri, *et al.*, "Packet-level prediction of mobile-app traffic using multitask deep learning," *Computer Networks*, 2021.

[8] L. Fang, *et al.*, "Mobile demand forecasting via deep graph-sequence spatiotemporal modeling in cellular networks," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3091–3101, 2018.

[9] J. Bruna, *et al.*, "Spectral networks and locally connected networks on graphs," *ICLR*, 2014.

[10] C. Shang, *et al.*, "Multi-view spectral graph convolution with consistent edge attention for molecular modeling," *Neurocomputing*, (445), 2021.

[11] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[12] Y. Li, *et al.*, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *ICLR*, 2018.

[13] Z. Wu, *et al.*g, "Graph WaveNet for Deep Spatial-Temporal Graph Modeling," in *IJCAI*, 2019.

[14] Y. N. Dauphin, *et al.*, "Language modeling with gated convolutional networks," in *ICML*, 2017.

[15] A. Pareja, *et al.*, "Evolvegcn: Evolving graph convolutional networks for dynamic graphs," in *AAAI*, 2020.

[16] S. Zhao, *et al.*, "Cellular network traffic prediction incorporating handover: A graph convolutional approach," in *IEEE SECON*, 2020.

[17] C. Tian and W. K. Chan, "Spatial-temporal attention wavenet: A deep learning framework for traffic prediction considering spatial-temporal dependencies," *IET Intelligent Transport Systems*, vol. 15, no. 4, 2021.

[18] Y. Seo, *et al.*, "Structured sequence modeling with graph convolutional recurrent networks," in *ICONIP*, 2018.

[19] B. Yu, *et al.*, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," *IJCAI*, 2018.

[20] M. Z. Shafiq, *et al.*, "Geospatial and temporal dynamics of application usage in cellular data networks," *IEEE TMC*, vol. 14, no. 7, 2014.

[21] M. Kalander, *et al.*, "Spatio-temporal hybrid graph convolutional network for traffic forecasting in telecommunication networks," *arXiv preprint arXiv:2009.09849*, 2020.