Proceedings of Machine Learning Research 123:153–163, 2020 NeurIPS 2019 Competition and Demonstration Track

Graph-ResNets for short-term traffic forecasts in almost unknown cities

Henry Martin MARTINHE@ETHZ.CH
Dominik Bucher DOBUCHER@ETHZ.CH
Ye Hong HONGY@STUDENT.ETHZ.CH
René Buffat BUFFAT@GMAIL.COM
Chair of Geoinformation Engineering, ETH Zurich,
Stefano-Franscini-Platz 5, 8093 Zürich, Switzerland

Christian Rupprecht CHRISR@ROBOTS.OX.AC.UK
Visual Geometry Group, University of Oxford,
Parks Road, Oxford, OX1 3PJ, United Kingdom

Martin Raubal MRAUBAL@ETHZ.CH
Chair of Geoinformation Engineering, ETH Zurich,
Stefano-Franscini-Platz 5, 8093 Zürich, Switzerland

Editors: Hugo Jair Escalante and Raia Hadsell

Abstract

The 2019 IARAI *traffic4cast* competition is a traffic forecasting problem based on traffic data from three cities that are encoded as images. We developed a ResNet-inspired graph convolutional neural network (GCN) approach that uses street network-based subgraphs of the image lattice graphs as a prior. We train the Graph-ResNet together with GCN and convolutional neural network (CNN) benchmark models on Moscow traffic data and use them to first predict the traffic in Moscow and then to predict the traffic in Berlin and Istanbul. The results suggest that the graph-based models have superior generalization properties than CNN-based models for this application. We argue that in contrast to purely image-based approaches, formulating the prediction problem on a graph allows the neural network to learn properties given by the underlying street network. This facilitates the transfer of a learned network to predict the traffic status at unknown locations.

Keywords: traffic prediction, graph neural networks, lattice graphs, image masks, resnet

1. Introduction

Today 55 % of the world's population live in urban areas. This number is expected to rise to 68 % by 2050 (United Nations, 2018). The growing urbanization in combination with population growth and private car ownership threatens to raise the already high levels of congestion which in turn increases pollution and economic costs and accelerates climate change (Reed, 2019). In order to facilitate the transition towards a more sustainable traffic system the growing traffic volume has to be managed in a smart way to reduce its negative impact.

A backbone of smart traffic management are short-term traffic flow predictions. They allow the detection of anomalies, such as accidents or obstacles (Wang et al., 2016), the smart routing of vehicles to optimally use the existing infrastructure (Ringhand and Vollrath, 2018) or the predictive control of the transportation system using traffic lights (Huang

et al., 2018). There is a vast body of literature concerning traffic forecasting available (Vlahogianni et al., 2014; Ermagun and Levinson, 2018), however, *"comparing the forecasting applications across studies is almost impossible"* (Ermagun and Levinson, 2018, p. 791) as studies use different spatio-temporal data resolution and report different error metrics on different aggregation levels.

***Traffic4cast* competition**

In response to this lack of standardization, the Institute of Advanced Research in Artificial Intelligence (IARAI) published a novel, publicly available traffic forecasting benchmark dataset as part of the *traffic4cast* competition[1]. The *Traffic4cast* dataset comprises traffic data from three cities (Berlin, Istanbul and Moscow) and covers one year in 5-minute intervals. The data is given as as three-channel images with normalized information about traffic volume, average speed and average direction, which all range from 0 - 255. An example of the data can be seen in Figure 1. Here the logarithm of the per-city sum of all channels over all training images is shown as a proxy for the activity level per pixel. Only few pixels are always zero over the whole dataset[2], nevertheless the street network is still clearly visible in each city.

The task in the *traffic4cast* competition is to predict the traffic of the next 15 minutes (3 images) based on the last hour (12 images). The results are then evaluated using a pixel-wise calculation of the mean-squared-error (MSE) between the prediction and the ground truth.

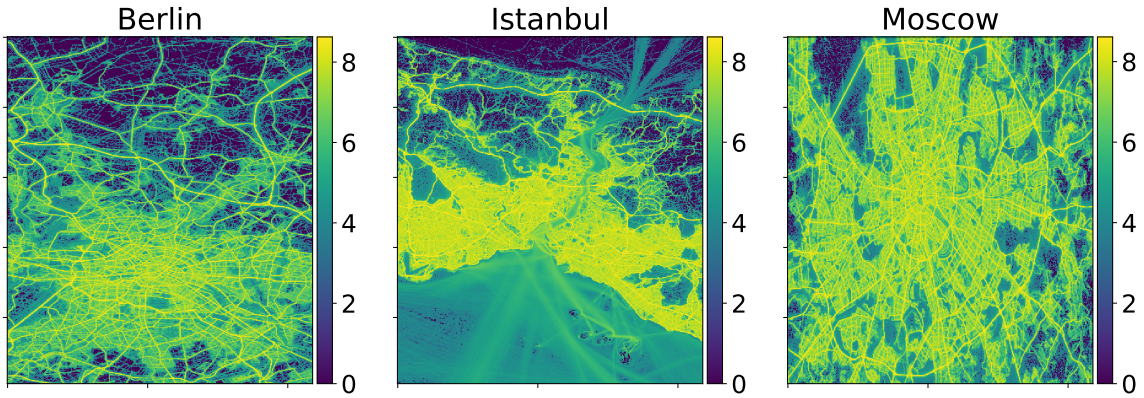

Figure 1: Logarithm of the per city sum of all training images of all channels.

Graph convolutional neural networks (GCNs)

The competition specifically encoded the traffic data as images to facilitate the usage of deep convolutional neural networks (CNN). However, the image-based representation omits explicit information about the street network and therefore disregards that the movement of cars is usually restricted to the road network. While a CNN-based approach will likely be able to extract the street network and store it in its weights, the street network could be provided explicitly to greatly reduce the problem complexity.

1. www.iarai.ac.at/traffic4cast
2. 29 % for Berlin, 11 % for Moscow and 23 % for Istanbul.

GCNs have been developed over the past couple of years in an attempt to generalize the success of CNNs to irregularly structured domains that can often be described via graphs (Bronstein et al., 2017; Defferrard et al., 2016). By now, there are several propositions for graph (convolutional) neural network architectures that are described in the available review papers (Wu et al., 2019; Zhou et al., 2018). GCNs have already been applied to many different domains (Bronstein et al., 2017; Wu et al., 2019; Zhou et al., 2018) but there are only few examples that are related to the movement of people (Martin et al., 2018) or short-term traffic forecasting (Yu et al., 2018; Cui et al., 2019; Zhang et al., 2019).

Contribution
In this work use the image-based U-Net approach of team MIE-Lab (Martin et al., 2019)[3] that won the second place in the *traffic4cast* competition and compare it with an alternative approach based on GCNs. We use well-known GCN architectures as well as suitable modifications such as the Graph Residual Network (Graph-ResNet) which is inspired by the residual learning network (ResNet) presented in (He et al., 2016a). We provide evidence that while the U-Net approach outperforms the GCN approach on known cities, the GCN approach generalizes better to unknown cities. The code to reproduce all experiments is publicly available[4].

2. Graph based traffic forecasting

2.1. Preprocessing

We introduce the ordered set of timestamps $T = \mathbb{N}_{12\times24\times365}$ to index all the 5-minute intervals throughout the year[5]. We denote all available training data as a 5-dimensional tensor $\mathbf{P}$ with shape $(|U|, |T|, h, w, |C|)$, where an individual value $\mathrm{P}_{u,t,i,j,c} \in \mathbb{N}_{255}$ denotes the integer value of a single pixel with $u \in U = \{B, I, M\}$ (for *Berlin*, *Istanbul* and *Moscow*), $t \in T$ the timestamp throughout the year, $i \in \mathbb{N}_{495}$ and $j \in \mathbb{N}_{436}$ the pixel coordinates, $c \in C = \{V, S, H\}$ the channel (corresponding to *Volume*, *Speed* and *Heading*), and $h = 495$ and $w = 436$ are the height and width of a single image.

We define a sample as $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$ where $\boldsymbol{x}^i$ is a short movie of 12 consecutive traffic images with the timestamps $[t^{(i)}, t^{(i)} + 1, ..., t^{(i)} + 12]$ and $\boldsymbol{y}^{(i)}$ consists of the images with the next three consecutive timestamps $[t^{(i)} + 13, t^{(i)} + 14, t^{(i)} + 15]$ which represent the prediction target. As it is described in Martin et al. (2019) we collapse the time dimension into the channel dimension. $\boldsymbol{x}^{(i)}$ is then of shape $(12 \cdot |C|, h, w)$ and $\boldsymbol{y}^{(i)}$ of shape $(3 \cdot |C|, h, w)$. The official test set in the *traffic4cast* competition contains only a small subset of timestamps for every day, namely *01:30, 04:45, 09:30, 14:30, 18:30*. In this work, all experiments are performed using only the subset of the data that corresponds to the available test timestamps (e.g, we use all training images corresponding to *01:30, 04:45, 09:30, 14:30, 18:30* but we omit training images corresponding to different time stamps). This approach leads to only a slight loss in performance while greatly decreasing training time.

3. Code and pretrained networks are available under https://github.com/mie-lab/traffic4cast.
4. https://github.com/mie-lab/traffic4cast-Graph-ResNet
5. We define $\mathbb{N}_w = \{1, 2, ..., w\}$ as the set of natural numbers up until w.

2.2. Graph extraction

To enable the usage of GCNs we follow the workflow shown in Figure 2. (1) a city-specific mask is used to extract pixels that lie on the street network. The mask is generated based on the training images of the city. (2) the remaining pixels are transformed into a graph. In this graph representation, data is represented as a vector of node features and a sparse adjacency matrix to store their connectivity. (3) we use GCNs to generate node level predictions. (4) the data is transformed back from the graph domain into the image domain where the error is calculated based on the competition rules.

To create a graph from the traffic images, we define an image as a regular grid with pixels as nodes. The pixel-nodes are connected to all directly adjacent and diagonally adjacent pixel-nodes via undirected edges with weight 1. The pixel values of all available channels are stored as a vector of node features. For a given graph, the node feature vectors form a feature matrix with dimension $n \times 12 \cdot |C|$. We introduce sparsity in the above defined grid graph by deleting nodes corresponding to low activity pixels. A low-activity pixel $\hat{\mathrm{P}}_{i,j}$ is defined as a pixel for which the sum over all available training images (of a single city u) does not exceed a certain user-defined threshold μ:

$$P_{LA,u} = \{\hat{\mathrm{P}}_{i,j} | \sum_{t \in T,\, c \in C} \sum \mathrm{P}_{u,t,i,j,c} < \mu\} \tag{1}$$

A visualisation of the sum (before applying the threshold) is given in Figure 1.

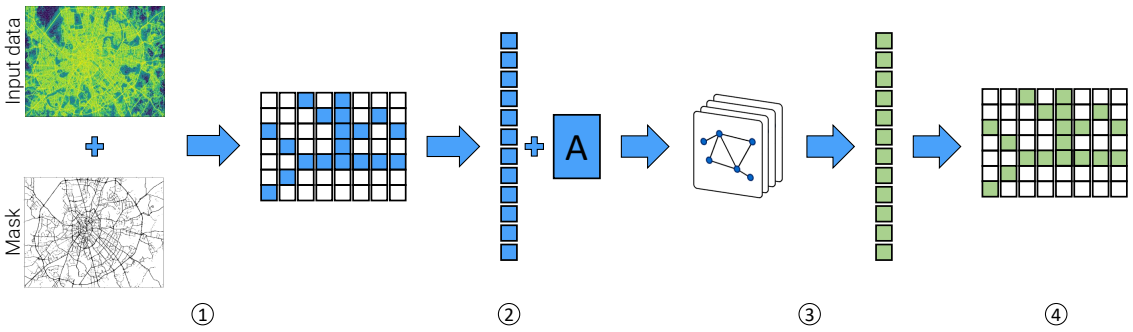

Figure 2: The graph-based traffic forecasting workflow. The input images are transformed into graphs by filtering out inactive pixels, after which the Graph ResNet learns to predict traffic on this graph.

2.3. GCN architectures

Based on the graphs extracted in Section 2.2, we use a GCN as proposed in (Kipf and Welling, 2016) as baseline and introduce three variations based on this baseline and on the ideas of the ResNet proposed by (He et al., 2016a).

Simple graph convolutional neural network

Similar to (Kipf and Welling, 2016), we define a graph convolutional processing block as the

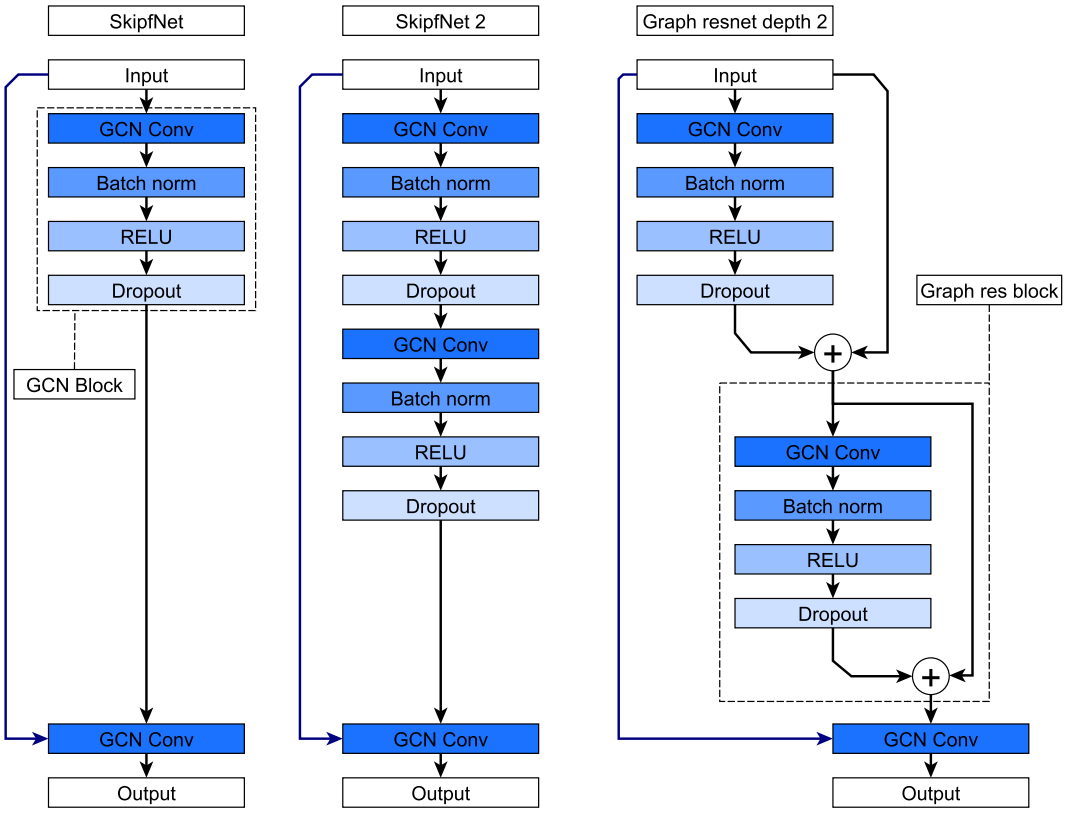

Figure 3: The different GCN architectures used and introduced in this paper.

sequential application of a graph convolution and a ReLU activation function (we optionally consider dropout and batch normalization layers but do not explicitly denote them in the formalization below):

$$\boldsymbol{H}_{G_u}^{(l+1)} = \sigma(\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}^{(u)}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}_{G_u}^{(l)}\boldsymbol{W}^{(l)}) \tag{2}$$

Here, l denotes the current layer in the network, $\boldsymbol{H}_{G_u}^{(l)}$ the input from the previous layer (where $\boldsymbol{H}_{G_u}^{(0)} = \boldsymbol{x}^{(i)}$ is the input, $\boldsymbol{A}^{(u)}$ is the adjacency matrix, $\boldsymbol{D}$ is the diagonal node degree matrix of $\boldsymbol{A}^{(u)}$ (used to normalize the adjacency matrix), and $\boldsymbol{W}^{(l)}$ is a layer-specific (learned) weight matrix. $\sigma(\cdot)$ is an appropriate activation function; in this work, we use rectified linear units (ReLU) for all experiments. In addition to the network proposed by (Kipf and Welling, 2016), we add a skip connection that concatenates the input $\boldsymbol{x}^{(i)}$ with the output from the last GCN block before applying a last graph convolution (i.e., $\boldsymbol{y}^{(i)} = \boldsymbol{H}_{G_u}^{(m+1)} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}^{(u)}\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{H}_{G_u}^{(m)} \oplus \boldsymbol{H}_{G_u}^{(0)})\boldsymbol{W}^{(m)}$ with m being the number of GCN blocks). The combined SkipfNet has the advantage that it can very well learn functions resembling the identity function $f(x) = x$ due to the direct availability of the non-convoluted input (cf. He et al. (2016b)). A central hyperparameter of the SkipfNet is the number of GCN blocks; Figure 3 shows exemplary networks with one and two blocks.

Graph ResNet

Inspired by the popular ResNet architecture (He et al., 2016a), we introduce a Graph ResNet to further improve the predictive power of the SkipfNet. We define a graph residual block similar to the GCN block of the SkipfNet, but add input of the previous layer to the output of the current layer before passing it on to the next graph residual block (cf. Figure 3; again, we do not mathematically represent dropout and batch normalization below):

$$\boldsymbol{H}_{G_j}^{(l+1)} = \sigma(\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}^{(u)}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}_{G_j}^{(l)}\boldsymbol{W}^{(l)}) + \boldsymbol{H}_{G_j}^{(l)} \tag{3}$$

While in the original ResNet architecture, the layer size varied throughout the network using pooling. However, as pooling for graphs is more complex and still under active research (Ying et al., 2018; Lee et al., 2019), we keep the layer size constant.

3. Experiments and results

We perform two groups of experiments. At first we train different graph networks to solve the *traffic4cast* prediction task in Moscow and compare the results to U-Nets of different depth. As a second set of experiments we use the GCN and U-Nets that were trained only on Moscow and use them to predict the traffic in Berlin and Istanbul.

All U-Nets are trained using the parameters and training schedule from (Martin et al., 2019) with a varying depth between 2 and 6 layers. Additionally, we include the original KipfNet with 16 hidden units as used in the original paper for the CORA dataset (Kipf and Welling, 2016) and with 128 hidden units, which is the maximum that our GPU allows. We fit 36 SkipfNets of depth 1, 29 SkipfNets of depth 2 and a total of 42 Graph-ResNets using random search on the hyperparameter space shown in Table 1. All Networks are fitted with the ADAM optimizer, learning rate 0.01 and weight decay of 0.0001 for a maximum of 10 epochs with early stopping if the validation error is not decreasing for two epochs in a row. The learning rate is divided by 10 after 5 epochs. We use a batch size of 2 and the mean squared error as a loss function. All graph networks receive the pixel coordinates of the nodes as two additional channels. These additional channels were omitted for the U-Nets as they degraded their performance. Training was performed using a Tesla P100 GPU with 12 GB of RAM. All experiments were implemented in Pytorch (Paszke et al., 2019) and Pytorch-Geometric (Fey and Lenssen, 2019).

Network	H 1	H 2 / depth	K	K mix	sc	bn	dropout
SkipfNet	8, 16, 32, 48, **64**	-	2, 4, **6**, 8	1,2,4,**6**	**0**, 1	**0**	**0.5**
SkipfNet2	8, 16, 32, 48, **64**	8, 16, **32**, 48	2, 4, **6**, 8	1,**2**,4	0, **1**	**0**	**0.5**
Graph ResNet	16 - 100 (**100**)	2 - 60 (**4**)	2 - 6 (**4**)	1, 2 (**2**)	0, 1 (**1**)	0, **1**	**0**, 0.5

Table 1: Hyperparameter space explored during graph network training. H=Hidden layer; K and K mix=number of terms for the Chebyshev polynomial from (Defferrard et al., 2016) for the convolutional layers and the last mixing layer; sc=usage of optional skip connection (blue in Figure 3); bn=usage of batchnorm; dropout=used dropout probability. Hyperparameters of best model are shown in bold.

3.1. Traffic prediction capacity

The out-of-sample errors of the different models for the Moscow validation dataset are shown in the top graph of Figure 4 and in Table 2. The Figure shows a clear dependency between the number of model parameters and the performance. In general there are two major ways to add complexity to the models: increasing the depth of the model or increasing the number of channels of the convolutional layers. The model complexity of the graph networks is limited because deeper networks lead to over-smoothing of the prediction (Li et al., 2018) and because the lack of suitable graph pooling operations forces us to add channels at the full-size graph which quickly drains GPU memory. In our work the original KipfNet architecture reaches the memory limit of our GPU at 128 output channels of the convolution. The Skipfnet architecture with the variable K parameter and the skip connection allows adding additional parameters without over-smoothing. Finally the Graph ResNet allows us to increase the performance even further. None of the graph networks reaches the performance of the deeper U-Net versions, however, all of the methods are below the official *traffic4cast* baseline[6] that is shown as a dashed line.

	KipfNet nh16	KipfNet nh128	Graph-ResNet	SkipfNet1	SkipfNet2
Nb. of params	$1.5 \cdot 10^3$	$1.2 \cdot 10^4$	$1.7 \cdot 10^5$	$1.8 \cdot 10^4$	$2.8 \cdot 10^4$
MSE Moscow	867	836	814	836	828
MSE Berlin	492	475	**468**	474	471
MSE Istanbul	671	645	**633**	644	639

	U-Net d2	U-Net d3	U-Net d4	U-Net d5	U-Net d6
Nb. of params	$4.2 \cdot 10^5$	$1.8 \cdot 10^6$	$7.7 \cdot 10^6$	$3.1 \cdot 10^7$	$1.2 \cdot 10^8$
MSE Moscow	813	797	**791**	794	794
MSE Berlin	478	500	520	519	501
MSE Istanbul	653	713	731	727	704

Table 2: Number of parameters and the resulting mean squared error for all models in the different cities. The content of the table is visualized in Figure 4.

3.2. Generalization capacity

In a second step we use the models trained on Moscow to predict the traffic in Berlin and Istanbul. To make up for the GCN's additional knowledge of the of the street network we mask the U-Net predictions by multiplying them with the binary mask of the street network. The results of this experiment are shown in Figure 4.

The first result is that all methods generalize surprisingly well to unknown cities, as all methods are able to beat the baseline that is described in Footnote 6 and shown as a dashed

6. The baseline is a MSE of 1032 for Moscow, 789 for Istanbul and 582 for Berlin. It consists of predicting the per-channel mean of the last three images for the next three images. Due to the high regularity in traffic, this can be considered a strong baseline. The implementation of the baseline is available under https://github.com/iarai/NeurIPS2019-traffic4cast.

8. See Footnote 6

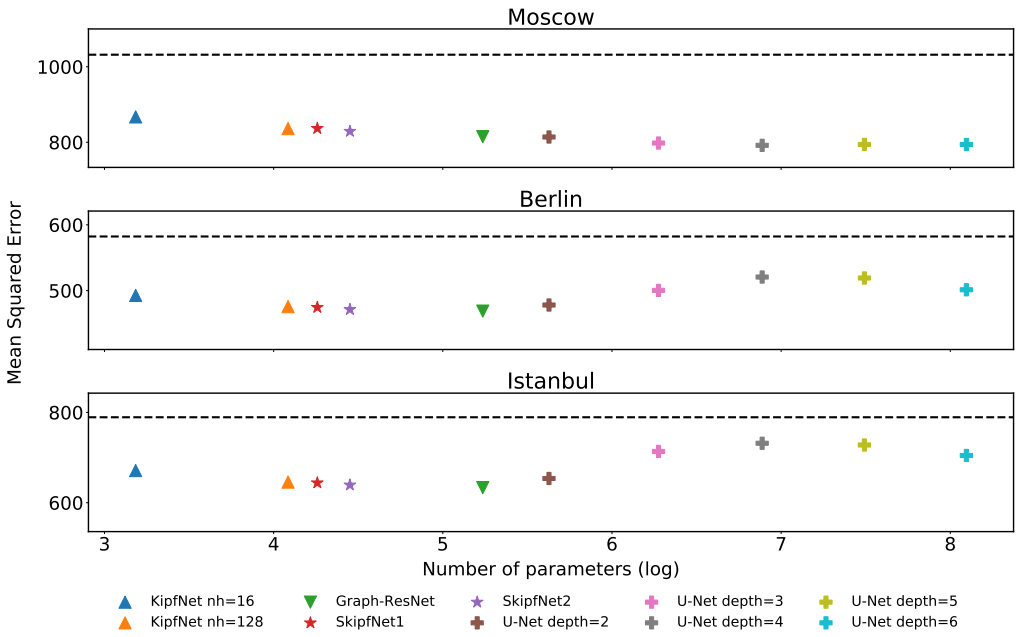

Figure 4: Traffic forecasting results. All models are trained on Moscow (top) and tested on Berlin (middle) and Istanbul (bottom). The competition baseline is shown as dashed line[8].

line for each city. Next, the results show that the relationship between performance and number of parameters does not hold anymore. For the U-Nets we observe that the shallow U-Net has the best prediction results and that the U-Net loses performance the deeper it gets. This trend is not as clear for U-Nets for depth 5 and 6 which might be a sign that they have not been trained with enough data or not long enough. To support this claim, we repeated this experiment with the pretrained U-Net of depth five used in the *traffic4cast* competition by Team MIE-Lab[9]. This competition network was tuned more carefully and trained with more data and indeed does fail miserably in the generalization task. It achieves a MSE of 774 for Moscow where it outperforms all other networks but a MSE of 798 for Berlin and an MSE of 3357 for Istanbul (both off the charts).

For the GCNs, the relationship between performance and number of parameters is still intact with the Graph-ResNet now being the overall best performer. An explanation for these observations is that the more parameters the conventional CNNs have and the more data they get during training, the better they can extract the (non-transferable) street network and store it in their weights. Whereas with few parameters they will simply learn a baseline like the conditional mean of the input data, which is to a large extent generalizable to other cities. The graph-based networks however already know the street network as it is provided as an explicit input and can therefore learn transferable rules based on the propagation of traffic through the network.

9. See Footnote 3.

4. Conclusions and future work

In this work, we presented a graph-based approach for traffic forecasting and apply it to the publicly available dataset from the *traffic4cast* competition. We define ResNet-inspired GCN networks with skip connections that allow the training of deeper GCNs without over-smoothing. We train all models on traffic data from Moscow and validate them on data from Berlin and Istanbul. Our results suggest that the GCN based models generalize better than the state-of-the-art CNN models from the *traffic4cast* competition. We think that the large performance difference between U-Nets and GCNs in the known city prediction is largely because the image-based competition design heavily favors conventional CNNs. Especially GPS positioning noise in combination with the MSE evaluation which favors blurry predictions (Lotter et al., 2016) is hard for the graph networks which can, by design, only predict values exactly on the graph. To still have a good performance under competition rules[10], the threshold for extracting the street network from Equation 1 was chosen rather low. However, a low threshold almost reproduces a grid graph where conventional CNNs are known to outperform GCNs. Therefore follow-up work should evaluate the performance of graph models under conditions that do not disadvantage graph-based approaches (e.g., by only evaluating pixels that lie on the graph). Furthermore, the relationship between parameters and model performance for graph models on unknown cities should be further explored by creating more complex graph models. A key role for this will be the successful integration of graph pooling methods in the model architecture. These experiments will be important to validate the findings of this work that, opposed to CNN-based models, GCN-based models continue to increase their performance with increasing model complexity when forecasting traffic in unknown cities.

Acknowledgments

This research was supported by the Swiss Data Science Center (SDSC) and by the Swiss Innovation Agency Innosuisse within the Swiss Competence Center for Energy Research (SCCER) Mobility. Christian Rupprecht is supported by ERC IDIU-638009.

References

M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, July 2017.

Zhiyong Cui, Kristian Henrickson, Ruimin Ke, and Yinhai Wang. Traffic Graph Convolutional Recurrent Neural Network: A Deep Learning Framework for Network-Scale Traffic Learning and Forecasting. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–12, 2019. Conference Name: IEEE Transactions on Intelligent Transportation Systems.

10. In this context *competition rules* refers to the fact that all pixels are evaluated, not just pixels that are on the street graph; cf. Figure 1.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems 29*, pages 3844–3852, 2016.

Alireza Ermagun and David Levinson. Spatiotemporal traffic forecasting: review and proposed directions. *Transport Reviews*, 38(6):786–814, November 2018.

Matthias Fey and Jan Eric Lenssen. Fast Graph Representation Learning with PyTorch Geometric. April 2019. arXiv: 1903.02428.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016b.

Wei Huang, Lubing Li, and Hong K. Lo. Adaptive traffic signal control with equilibrium constraints under stochastic demand. *Transportation Research Part C: Emerging Technologies*, 95, October 2018.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-Attention Graph Pooling. June 2019. arXiv: 1904.08082.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. 2016. arXiv: 1605.08104.

Henry Martin, Dominik Bucher, Esra Suel, Pengxiang Zhao, Fernando Perez-Cruz, and Martin Raubal. Graph convolutional neural networks for human activity purpose imputation. In *NIPS spatiotemporal workshop at the 32nd Annual conference on neural information processing systems (NIPS 2018)*, 2018.

Henry Martin, Ye Hong, Dominik Bucher, Christian Rupprecht, and René Buffat. Traffic4cast-traffic map movie forecasting–team mie-lab. 2019. arXiv: 1910.13824.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8026–8037. Curran Associates, Inc., 2019.

Trevor Reed. INRIX Global Traffic Scorecard. February 2019.

Madlen Ringhand and Mark Vollrath. Make this detour and be unselfish! Influencing urban route choice by explaining traffic management. *Transportation Research Part F: Traffic Psychology and Behaviour*, 53:99–116, February 2018.

United Nations. 2018 revision of world urbanization prospects, 2018.

Eleni I. Vlahogianni, Matthew G. Karlaftis, and John C. Golias. Short-term traffic forecasting: Where we are and where we're going. *Transportation Research Part C: Emerging Technologies*, 43:3–19, June 2014.

Shen Wang, Soufiene Djahel, Zonghua Zhang, and Jennifer McManis. Next Road Rerouting: A Multiagent System for Mitigating Unexpected Urban Traffic Congestion. *IEEE Transactions on Intelligent Transportation Systems*, 17(10):2888–2899, October 2016.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A Comprehensive Survey on Graph Neural Networks. January 2019. arXiv: 1901.00596.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical Graph Representation Learning with Differentiable Pooling. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 4800–4810. Curran Associates, Inc., 2018.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 3634–3640, July 2018. arXiv: 1709.04875.

Yang Zhang, Tao Cheng, Yibin Ren, and Kun Xie. A novel residual graph convolution deep learning model for short-term network-based traffic forecasting. *International Journal of Geographical Information Science*, 0(0):1–27, December 2019.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph Neural Networks: A Review of Methods and Applications. December 2018. arXiv: 1812.08434.