Time Series Forecasting Using a Unified Spatial-Temporal Graph Convolutional Network

Yi-Fan Li[*], Yang Gao[*], Yu Lin, Zhuoyi Wang and Latifur Khan
Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75080
{yli, yxg122530, yxl163430, zhuoyi.wang1, lkhan}@utdallas.edu

Abstract

Time series forecasting with additional spatial dependencies is one of the most significant research problems for web information analysis. Recently, Graph Neural Network (GNN) is widely used to better understand the hidden pattern in complex graphs, such as traffic network. Properly capturing dynamics of traffic flows requires the model to incorporate both information from nearby roads (spatial) and past traffic flow records (temporal). Most existing frameworks model spatial and temporal dependencies in two separate steps. However this type of framework may suffer from complex network design, and information loss between the distict steps. In this paper, we propose a unified spatial-temporal GNN framework that captures both spatial and temporal dependencies in only one step. More specifically, for each node in the graph, a unified neural network component is designed to simultaneously extract information from its surrounding nodes and its past records, which enables less information loss with fewer model parameters. We hypothesize that our proposed method could potentially better generalize spatial-temporal data in a more scalable way, compared to state-of-the-art (SOTA) methods.

1 Introduction

The spatial-temporal analytics is one of the most important techniques for web data mining, especially in the context of time-series forecasting with geographical location data. This technique can largely benefit research tasks related to smart city and political science. Such tasks include traffic flow prediction [1], anomalous event detection [2], and local business recommender [3]. Here we expand the case of traffic flow prediction and discuss its details. Estimating the dynamics traffic conditions is a typical spatial-temporal analytics problem. The traffic data are recorded at fixed points in time and at fixed locations on the road distributed in space.

More specifically, the traffic flow is tracked at each sensor with fixed frequencies in a sequential manner, such as t, $t+1$, etc. Indeed, the traffic histories observed at neighboring locations at different time stamps are not independent of each other. Hence, properly extracting both spatial and temporal relationships out of the data is the key to address this problem. Therefore, it could be a major contribution to the smart city research, if both the spatial and temporal dependencies can be captured from the rich data properly.

Deep learning approaches have been widely used for various tasks for spatial-temporal modeling, and have shown their superior performances compared to traditional time-series forecasting methods such as Auto-Regressive Integrated Moving Average (ARIMA) model. However, most existing frameworks are based on a two-step protocol: First, apply Graph Convolutional Networks (GCN)

[*]Equal contribution

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

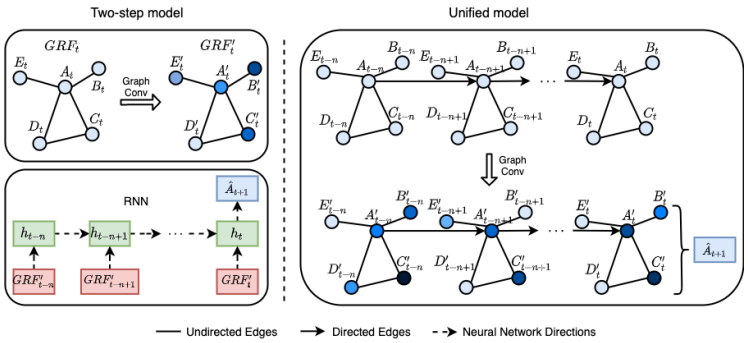

Figure 1: Comparisons between conventional two-step models and our proposed unified model.

module to extract the spatial dependencies out of the graph structured data; Then, incorporate a Recurrent Neural Network (RNN) module [4] or a Convolutional Neural Network (CNN) module to model the temporal dependencies [5], which lead to relatively complex network design.

However, frameworks described above have a number of shortcomings. Overall, even though this two-step paradigm typically goes unquestioned, Lea et al. [6] argue that the model suffers from the loss of valuable information between steps. Regarding the temporal dependency modeling, RNN-based methods typically suffer from time-consuming propagations. In addition, gradient vanishing/explosion may occur when the framework tries to model long sequences [7]. Moreover, CNN-based methods normally require stack multiple 1D convolution layers to properly learn temporal information from long sequences, which could lead to a very deep neural network structure [8]. Some existing work uses a single network structure to model both dependencies, however they treat spatial edges and temporal edges the same way, despite they are heterogeneous features [9, 10].

To address the aforementioned challenges in existing frameworks, we propose a Unified spatial-TEmpoRal (UTTER) graph neural network for time series forecasting. *The key idea is that if we can construct a proper graph over sequences of data, which includes both spatial and temporal information, then a single graph neural network could be established to capture both dependencies simultaneously.* Therefore the main contribution of this work is threefold:

- We design a subgraph construction module to sample the time-series data. Subgraphs generated for each node contains both its structural neighborhood and its past records, which essentially are its temporal neighborhood.
- We propose to use aggregation functions to learn the constructed subgraphs. This aggregation function allows us to learn both the topological structure and the feature distributions from each node's neighborhood.
- We propose to evaluate our framework on real-world datasets compared to other SOTA methods, including temporal-only models and spatial-temporal deep learning models.

2 Method

The primary idea behind our approach is that we convert the time-series forecasting problem to a graph learning problem with spatial-temporal information encoded. First, We formally define the time-series forecasting problem under the graph neural network settings. Second, we describe a subgraph construction algorithm Third, we demonstrate the forward propagation algorithm to generate the embeddings using UTTER given learnable parameters. Finally, we discuss how the parameters in UTTER are learned using conventional backpropagation algorithms in the rest portion of this section.

2.1 Problem Formulation

We formally define time-series forecasting under the graph neural network settings. Given the edge connections between nodes, an undirected graph can be represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set

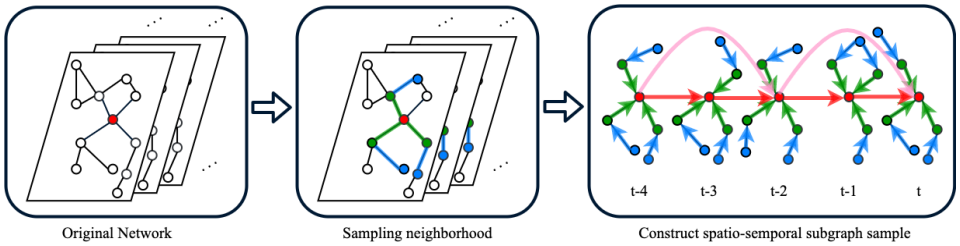

Figure 2: Demonstration of constructing local spatial-temporal subgraph from data sequences. The red/gree/blue dots represent target/1-hop/2-hop nodes respectively. Arrows point out the aggregation directions. Green/blue arrows correspond to 1-hop/2-hop spatial adjacency, and red/pink arrows correspond to direct/skip temporal adjacency. Parameters of the example is: No. of time steps: $P = 5$, No. of spatial aggregations $K = 2$, skip time steps $L = 1$, and No. of skip connections $M = 2$.

of nodes with $|\mathcal{V}| = N$, and $\mathcal{E}$ is a set of edges representing the connectivity between nodes. The overall raw features of graph $\mathcal{G}$ is denoted by $X \in \mathbb{R}^{N \times d}$ where d is the dimension of the features. Indeed, each graph $\mathcal{G}$ represents the relationships between nodes at each time step t. It is worth noticing that we assume that the overall graph structure is static in this work, which means that it doesn't change over time.

A typical time-series forecasting problem aims to predict the value for future length-Q steps based on previous P observations. As we mentioned before, data at each time step t is a graph $\mathcal{G}$ with feature X^t. In other words, we map our training data $X^{(t-P+1):t}$ to forecasted values $X^{(t+1):(t+Q)}$ using some forecast function $f(\cdot)$:

$$[X^{(t-P+1):t}, \mathcal{G}] \xrightarrow{f(\cdot)} [Y^{(t+1):(t+Q)}] \tag{1}$$

where $X^{(t-P+1):t} \in \mathbb{R}^{P \times N \times d}$ and $Y^{(t+1):(t+Q)} \in \mathbb{R}^{Q \times N}$.

2.2 spatial-Temporal Subgraph Sampling

Our proposed framework aims to model the spatial and temporal dependencies in a unified module. Therefore, in each training example, multiple graph networks at distinct time steps need to be considered. Inspired by GraphSage [11], we design our algorithm following a "Sample and Aggregate" fashion. Here we describe the sampling method used to build each spatial-temporal subgraph example. It is worth noting that the subgraph generated contains heterogeneous information, and we need a learning algorithm to address this challenge later.

To model the relationships along space and time simultaneously, each example need to include a target node with its nearby nodes to preserve the spatial relationships, also consecutive time steps of neighborhood are demanded to preserve the temporal relationships. *The subgraph construction algorithm is presented in the supplemental materials, and is demonstrated in Fig. 2.*

As shown in this figure, the first step is to sample the neighborhood for the target node at each time step from the original network. In this work, a fixed-size set of neighborhood is sampled for each node at each. There are two major benefits of applying this sampling method: First, it is more computationally efficient to use a small set of nodes in each training example, instead of using the full graph node set (as discussed in in Kipf et al. [12]). Next, a fixed-size set of samples guarantees the same number of node dimensions of each training example. Thus, for $v \in \mathcal{V}$, a set of neighborhood nodes $u \in \mathcal{V}$ is sampled by a function $\mathcal{N}$ for K iterations. This also means that upto K-hop spatial dependencies can be considered in our sampled subgraph. There are multiple sampling approaches, such as rule-based sampling, uniform sampling and random degree node sampling [13]. For example, at the sampling neighborhood phase of Fig. 2, it demonstrates a uniform sampling with $K = 2$ (2-hop relationship), with sample size of 3 in each iteration.

The second step is to link spatial subgraphs between distinct time steps. One intuitive way is to connect the target node with directed edges between different subgraphs. Notably, the directed edges between time steps represent the asymmetry property of time, which means that subgraphs at a certain moment is only impacted by its previous steps, not next moments. Since the topological structure of

constructed subgraph contains both spatial and temporal information, the dependencies between the target node and its spatial-temporal neighbor can be extracted in one step.

However, connecting only adjacent neighbors along time is not enough. Normally information from 1-hop neighbors are combined together using the aggregation function once. Therefore, a huge number of layers is needed to model the temporal dependencies from the whole sequence under this setting, which is not quite feasible. Inspired by ResNet [14], we add shortcut connections along time, which turn the graph structure to its residual version regarding temporal dependencies modeling (see line 5-7 at Alg. 2). Two additional parameters are defined for the ease of future demonstration: L and M. More specifically, L represents the number of skip time steps, while M corresponds to the total number of such skip connections. For instance, in Fig. 2, since the skip connection links $t-4$ and $t-2$, which override $t-3$, the number of skip time step here is $L=1$. And we have a total number of 2 skip connections in this subgraph, which makes $M=2$. Suppose P, L, M are all integers, by setting these parameters properly, we can have a relationship between these parameters:

$$P = M \times (L+1) + 1 \tag{2}$$

2.3 Forward Propagation Algorithm

Assuming all parameters in UTTER are fixed, now we demonstrate the forward propagation algorithm, which aims to generate the node embedding for all nodes in the graph.

Here, we consider the spatial neighborhood by K-hop dependencies, and number of skip connections M derived from Eq. 2. To model the node embedding v_t including both spatial and temporal information from itself P step ahead, a total number of $K+M$ aggregation functions is needed. Similarly, $K+M$ weight matrices are desired to propagate information between different layers of model. We denote the neighborhood aggregation function and weight matrices as $\mathbf{AggNbh}_i$ and $\mathbf{W}_i$ respectively, where $i \in \{1, \cdots, M+L\}$. Besides, as we mentioned in previous subsection, the subgraph generated contains heterogeneous information. Here we refer $\mathbf{AggNbh}_i$ to as the general neighborhood aggregation function, and the specific function to use is demonstrated in Sec. 2.4.

We illustrate details about the proposed forward propagation algorithm for UTTER. Pseudocode of this algorithm is demonstrated in Alg. 1. Inputs of the algorithm are the graph structure $\mathcal{G}$, and features at time t: $\{x_v^{(t-P+1):t}, \forall v \in \mathcal{V}\}$. As discussed before, we need to aggregate the node representations for $K+M$ times in order to use all information K hops away and P steps ahead. This operation is demonstrated by the loop starting from Line 2. Inside this loop, every node in $\mathcal{G}$ is iterated, and the neighborhood embedding $\mathrm{h}_{\mathcal{N}_{\mathrm{tmp}},i}^{(t-P+1):t}$, is updated. More specifically, the neighbors here refer to the spatial-temporal subgraph generated by **GenSub** algorithm. It is worth noticing that in Line 5, the new embedding with regard to aggregating neighborhood information uses the embedding updated from last iteration $i-1$. Then, both the current representations of the target node and its neighborhood are concatenated together, and fed into a fully connected (fc) layer with activation function σ. Eventually, the output generated by the fc layer could be used as inputs then for the next aggregation step.

2.4 Aggregator Architecture

Due to the nature of our spatial-temporal subgraph sampling algorithm (**GenSub**), there are some basic requirements for proper aggregation functions to combine all neighborhood dependencies together. First, the generated subgraph contains two distinct types of directed edges: spatial edges and temporal edges. Therefore we should treat them differently during the aggregation. Second, the aggregator should be permutation invariant, since neighbors of nodes do not have a certain order. Additionally, it would be better if trainable parameters are preserved within this aggregation function, because they can greatly increase the express power of our proposed framework.

Inspired by Graph Isomorphism Network (GIN) [15], we propose a multi-layer proceptron (MLP) with two layers to achieve the best performance. Especially, we follow the GIN-0 protocol with sum operation to aggregate the neigborhood information altogether, as it outperforms other methods, such as mean aggregators or max aggregators. First, representations of target node and its spatial neighborhood are added together to form a spatial embedding. Then, representations of target node and its temporal neighborhood are added together to form a temporal embedding. Two embedding are

Algorithm 1: Forward propagation algorithm

Input: Graph $\mathcal{G}$; iterations $K+M$; trainable matrix $W_i, \forall i \in \{1, \cdots, K+M\}$; input features at time t $\{x_v^{(t-P+1):t}, \forall v \in \mathcal{V}\}$; activation function σ;

Output: spatial-temporal subgraph node representation at time t $\mathbf{emb}^t$

$\mathbf{h}_{v,0}^{(t-P+1):t} \leftarrow x_v^{(t-P+1):t}, \forall v \in \mathcal{V}$;

for $i = 1, \cdots, (K+M)$ **do**

  /* iterate through all nodes in the graph */

  for $v \in \mathcal{V}$ **do**

    /* generate spatial-temporal subgraph */

    $\mathcal{N}_{\text{tmp}} \leftarrow \mathbf{GenSub}(v)$;

    /* aggregate target node with neighborhood */

    $\mathbf{h}_{\mathcal{N}_{\text{tmp}},i}^{(t-P+1):t} \leftarrow \mathbf{AggNbh}_i\left(\{\mathbf{h}_{u,(i-1)}^{(t-P+1):t}, \forall u \in \mathcal{N}_{\text{tmp}}\}\right)$;

    $\mathbf{h}_{v,i}^{(t-P+1):t} \leftarrow \sigma\left(W_i \cdot \mathbf{Concat}(\mathbf{h}_{v,(i-1)}^{(t-P+1):t}, \mathbf{h}_{\mathcal{N}_{\text{tmp}},i}^{(t-P+1):t})\right)$;

  end

  Normalize $\mathbf{h}_{v,i}^{(t-P+1):t}$;

end

$\mathbf{emb}_v^t \leftarrow \mathbf{h}_{v,K+M}^t, \forall v \in \mathcal{V}$;

$\hat{y}_v^{(t+1):(t+Q)} \leftarrow \sigma\left(W^{\text{fc}} \cdot \mathbf{emb}_v^t\right)$

concatenate together for each of the instances, and this concatenation is fed into the designed MLP for information fusion. For the ease of demonstration, we denote the *spatial neighborhood* within the graph to the target node v as $\mathcal{N}_{\text{tmp}}^{\text{sp}}$, while the *temporal neighborhood* as $\mathcal{N}_{\text{tmp}}^{\text{te}}$. Consequently, the update rule for the target node is demonstrated as follows:

$$\mathrm{h}_{v,i}^{(t-P+1):i} = \text{MLP}\left(\mathbf{concat}(\mathrm{h}_{\text{tmp}}^{\text{sp}}, \mathrm{h}_{\text{tmp}}^{\text{te}})\right) \tag{3}$$

where $\mathrm{h}_{\text{tmp}}^{\text{sp}}$ corresponds to the spatial embedding, and $\mathrm{h}_{\text{tmp}}^{\text{te}}$. They are calculated as follows:

$$\mathrm{h}_{\text{tmp}}^{\text{sp}} = \mathrm{h}_{v,(i-1)}^{(t-P+1):i} + \sum_{u \in \mathcal{N}_{\text{tmp}}^{\text{sp}}} \mathrm{h}_{u,(i-1)}^{(t-P+1):i} \tag{4}$$

$$\mathrm{h}_{\text{tmp}}^{\text{te}} = \mathrm{h}_{v,(i-1)}^{(t-P+1):i} + \sum_{u \in \mathcal{N}_{\text{tmp}}^{\text{te}}} \mathrm{h}_{u,(i-1)}^{(t-P+1):i} \tag{5}$$

2.5 Learning Parameters of UTTER

After getting hidden representations of all nodes at time t, denoted as $\mathbf{emb}^t$, we flatten this hidden embeddings and feed it into two fully connected (fc) layers for prediction. Notice that the number of output nodes for the fc layer is Q. Therefore, unlike the previous work which generate the output at each time step recursively [4], predictions of UTTER are generated as a whole. We use Mean Squared Error (MSE) as the training objective function. This objective function represents the average squared differences between predictions for each target node at each time step to the ground truth. Mathematically, this loss function is defined as:

$$L = \frac{1}{QN} \sum_{i=1}^{Q} \sum_{j=1}^{N} \left(\hat{y}_j^{(t+i)} - y_j^{(t+i)}\right)^2 \tag{6}$$

3 Experiment Protocol

3.1 Datasets

This study considers time-series forecasting problem with additional spatial information in the dataset. Therefore, two traffic network datasets (METR-LA & PEMS-BAY [4]), and a conflict dataset

(ViEWS [16]) will be used here to evaluate our proposed method. We follow previous work, which divides this dataset along time, and uses a ratio of 7:1:2 for train, validation and test to evaluate our results.

- **ViEWS**[2] This political science dataset contains PRIO-Grid level [17] spatial-temporal conflict data concentrating on Africa, which includes records on monthly basis such as number of fatalities and geographical information. There are a total number of $10,677$ grid cells for Africa, and in this study we use data covering a period of 20 years, from January, 2000 to December, 2019.
- **METR-LA**[3] This traffic dataset consists of spatial-temporal transportation data, e.g., traffic speed, in Los Angeles county road network [18]. This record is updated every 5 minutes. Following the experiment setting in Li et al. [4], we select 207 sensors in the LA County, and in this study we use data covering a period of 4 months, from March 1, 2012 to June 30, 2012.
- **PEMS-BAY**[4] This traffic dataset contains spatial-temporal transportation data for Bay Area. Similar to METR-LA dataset, the data is recorded every 5 minutes. We select 325 sensors from the road network, and we use data covering a period of 6 months, from January 1, 2017 to June 30, 2017.

3.2 Experiment Design

There are three major research questions that we want to address by the experiment:

- **RQ1:** What is the best sampling method to construct the spatial-temporal graph?
- **RQ2:** How effective our proposed framework is for spatial-temporal modeling?
- **RQ3:** Is proposed method efficient when it comes to large real-world applications?

To answer RQ1, essentially ablative studies are needed to compare the model performance using different sampling methods, such as rule-based sampling, uniform sampling, and importance score-based sampling. The observation of this research question may unveil how neighborhood information may lead to learn the aggregation function properly.

Regarding RQ2, three evaluation metrics and two sets of baselines are designed to evaluate the effectiveness of our proposed framework. More specifically, for evaluation metrics, we plan to use MAE, RMSE and MAPE to evaluate our method, as they provide different insights into the prediction accuracy, and are commonly used in previous research [8, 19]. Regarding benchmark methods, the first set of baselines are classic time-series modeling methods, which only considers temporal dependencies, e.g. ARIMA, CNN, and LSTM models. Comparing to these methods could demonstrate whether the proposed method actually extract additional spatial information out of data. The second set of baselines are SOTA deep learning based spatial-temporal models, such as STGCN [8], Graph WaveNet [5], DCRNN [4]. By comparing to these methods, we can conclude if the proposed method is effective enough for spatial-temporal modeling.

RQ3 will be addressed by comparing the efficiency among deep-learning-based spatial-temporal models. In practice, the size of the spatial-temporal graph could be very large. For example, the conflict dataset contains 10,677 nodes and 41,711 edges. In industrial applications such as social network, the data scale is definitely even larger. Therefore we want to explore whether our proposed method is efficient enough to model the spatial and temporal dependencies for a sequence of very large graphs. More specifically, training time and inference time will be used to evaluate time efficiency of our model, and No. of model parameters will be used to evaluate the space efficiency.

4 Acknowledgement

This material is based upon work supported by NSF awards DMS-1737978, OAC-1828467, and OAC-1931541.

[2] https://pcr.uu.se/research/views/

[3] https://tinyurl.com/y6jytq9e

[4] https://tinyurl.com/y544mh6e

References

[1] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. Traffic flow prediction via spatial temporal graph neural network. In *Proceedings of The Web Conference 2020*, pages 1082–1092, 2020.

[2] Chao Zhang, Liyuan Liu, Dongming Lei, Quan Yuan, Honglei Zhuang, Tim Hanratty, and Jiawei Han. Triovecevent: Embedding-based online local event detection in geo-tagged tweet streams. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 595–604, 2017.

[3] Ruirui Li, Jyun-Yu Jiang, Chelsea J-T Ju, and Wei Wang. Corals: Who are my potential new customers? tapping into the wisdom of customers' decisions. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 69–77, 2019.

[4] 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.

[5] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 1907–1913. AAAI Press, 2019.

[6] Colin Lea, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks: A unified approach to action segmentation. In *European Conference on Computer Vision*, pages 47–54. Springer, 2016.

[7] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *arXiv preprint arXiv:1803.07294*, 2018.

[8] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3634–3640, 2018.

[9] Yujun Cai, Liuhao Ge, Jun Liu, Jianfei Cai, Tat-Jen Cham, Junsong Yuan, and Nadia Magnenat Thalmann. Exploiting spatial-temporal relationships for 3d pose estimation via graph convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2272–2281, 2019.

[10] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. *arXiv preprint arXiv:1801.07455*, 2018.

[11] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

[12] 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.

[13] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636, 2006.

[14] 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, 2016.

[15] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.

[16] Håvard Hegre, Marie Allansson, Matthias Basedau, Michael Colaresi, Mihai Croicu, Hanne Fjelde, Frederick Hoyles, Lisa Hultman, Stina Högbladh, Remco Jansen, et al. Views: a political violence early-warning system. *Journal of peace research*, 56(2):155–174, 2019.

[17] Andreas Forø Tollefsen, Håvard Strand, and Halvard Buhaug. Prio-grid: A unified spatial data structure. *Journal of Peace Research*, 49(2):363–374, 2012.

[18] Hosagrahar V Jagadish, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M Patel, Raghu Ramakrishnan, and Cyrus Shahabi. Big data and its technical challenges. *Communications of the ACM*, 57(7):86–94, 2014.

[19] 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.