HAL open science

Time-varying Signals Recovery via Graph Neural Networks

John A Castro-Correa, Jhony H Giraldo, Anindya Mondal, Mohsen Badiey, Thierry Bouwmans, Fragkiskos D. Malliaros

To cite this version:

John A Castro-Correa, Jhony H Giraldo, Anindya Mondal, Mohsen Badiey, Thierry Bouwmans, et al.. Time-varying Signals Recovery via Graph Neural Networks. ICASSP 2023 - IEEE International Conference on Acoustics, Speech and Signal Processing, Jun 2023, Rhodes Island, Greece. pp.1-5, 10.1109/ICASSP49357.2023.10096168 . hal-04368767

HAL Id: hal-04368767

https://hal.science/hal-04368767v1

Submitted on 1 Jan 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Time-varying Signals Recovery via Graph Neural Networks

John A. Castro-Correa[1*] , *Jhony H. Giraldo*[2*], *Anindya Mondal*[3*],
Mohsen Badiey[1], *Thierry Bouwmans*[4], *Fragkiskos D. Malliaros*[5]
[1] University of Delaware, USA
[2] LTCI, Télécom Paris, Institut Polytechnique de Paris, France
[3] Jadavpur University, India
[4] Laboratoire Mathématiques, Image et Applications (MIA), La Rochelle Université, France
[5] Université Paris-Saclay, CentraleSupélec, Inria, Centre for Visual Computing (CVN), France

arXiv:2302.11313v3 [eess.SP] 12 Aug 2023

***Abstract*—The recovery of time-varying graph signals is a fundamental problem with numerous applications in sensor networks and forecasting in time series. Effectively capturing the spatio-temporal information in these signals is essential for the downstream tasks. Previous studies have used the smoothness of the temporal differences of such graph signals as an initial assumption. Nevertheless, this smoothness assumption could result in a degradation of performance in the corresponding application when the prior does not hold. In this work, we relax the requirement of this hypothesis by including a learning module. We propose a Time Graph Neural Network (TimeGNN) for the recovery of time-varying graph signals. Our algorithm uses an encoder-decoder architecture with a specialized loss composed of a mean squared error function and a Sobolev smoothness operator.TimeGNN shows competitive performance against previous methods in real datasets.**

***Index Terms*—Graph neural networks, graph signal processing, time-varying graph signal, recovery of signals**

I. Introduction

Recent advances in information technology have led to an accumulation of large amounts of unstructured data. The representation and analysis of such irregular and complex data is a daunting task. Graph Signal Processing (GSP) and Graph Neural Networks (GNNs) are emerging research fields that have proved to be helpful for such tasks in recent years [1]–[6]. In GSP and GNNs, the data is modeled as signals or vectors on a set of nodes of a graph, incorporating both the feature information and the underlying structure of the data. GSP and GNNs thus provide new perspectives on data handling, connecting machine learning and signal processing [7], with profound impact in various fields like semi-supervised learning [3], computer vision [8], [9], and social media [10].

The sampling and reconstruction of graph signals are fundamental tasks that have recently attracted considerable attention from the signal processing and machine learning communities [1], [11]–[18]. Nevertheless, the problem of time-varying graph signal reconstruction[1] has not been widely explored [18]. The reconstruction of time-varying graph signals has significant applications in data recovery in sensor networks, forecasting of time-series, and infectious disease prediction [16], [18], [20]–[22]. Previous studies have extended the definition of smooth signals from static to time-varying graph signals [23]. Similarly, other works have focused on the rate of convergence of the optimization methods used to solve the reconstruction problem [18], [20]. However, the success of these optimization-based methods requires appropriate prior assumptions about the underlying time-varying graph signals, which could be inflexible for real-world applications.

In this work, we propose the Time Graph Neural Network (TimeGNN) model to recover time-varying graph signals. TimeGNN encodes the time series of each node in latent vectors. Therefore, these embedded representations are decoded to recover the original time-varying graph signal. Our architecture comprises: 1) a cascade of Chebyshev graph convolutions [2] with increasing order and 2) linear combination layers. Our algorithm considers spatio-temporal information using: 1) graph convolutions [2] and 2) a specialized loss function composed of a Mean Squared Error (MSE) term and a Sobolev smoothness operator [18]. TimeGNN shows competitive performance against previous methods in real-world datasets of time-varying graph signals.

The main contributions of our work are summarized as follows: 1) we exploit GNNs to recover time-varying graph signals from their samples, 2) we relax the strict prior assumption of previous methods by including some learnable modules in TimeGNN, and 3) we perform experimental evaluations on natural and artificial data, and compare TimeGNN to four methods of the literature. The rest of the paper is organized as follows. Section II introduces the proposed TimeGNN model. Section III presents the experimental framework and results. Finally, Section IV shows the conclusions.

II. Time Graph Neural Network

A. Preliminaries

We represent a graph with $G = (\mathscr{V}, \mathscr{E}, \mathbf{A})$, where $\mathscr{V}$ is the set of nodes with $|\mathscr{V}| = N$, $\mathscr{E} \subseteq \{(i,j) \mid i,j \in \mathscr{V} \text{ and } i \neq j\}$ is the set of edges, and $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the weighted adjacency matrix with $\mathbf{A}(i,j) = a_{i,j} \in \mathbb{R}_+$ if $(i,j) \in \mathscr{E}$ and 0 otherwise. In this work, we consider connected, undirected, and weighted

*Authors have equal contributions
Corresponding author: jcastro@udel.edu

[1]One can think of the recovery of time-varying graph signals as a matrix completion task where each column (or row) is associated with time, and each row (or column) is associated with a vertex of a graph.

graphs. We also define the symmetrized Laplacian as $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, where

$\mathbf{D} = diag(\mathbf{A}\mathbf{1})$ is the diagonal degree matrix of the graph. Finally, a node-indexed real-valued graph signal is a function $x : \mathscr{V} \to \mathbb{R}$, so that we can represent a one-dimensional graph signal as $\mathbf{x} \in \mathbb{R}^N$.

B. Reconstruction of Time-varying Graph Signals

The sampling and recovery of graph signals are crucial tasks in GSP [11], [12]. Several studies have used the smoothness assumption to address the sampling and recovery problems for static graph signals. The notion of global smoothness was formalized using the *discrete p-Dirichlet form* [24] given by:

$$S_p(\mathbf{x}) = \frac{1}{p}\sum_{i\in\mathscr{V}}\left[\sum_{j\in\mathscr{N}_i}\mathbf{A}(i,j)[\mathbf{x}(j)-\mathbf{x}(i)]^2\right]^{\frac{p}{2}}, \tag{1}$$

where $\mathscr{N}_i$ is the set of neighbors of node i. When $p = 2$, we have $S_2(\mathbf{x})$ which is known as the graph Laplacian quadratic form $S_2(\mathbf{x}) = \sum_{(i,j)\in\mathscr{E}}\mathbf{A}(i,j)[\mathbf{x}(j)-\mathbf{x}(i)]^2 = \mathbf{x}^\mathsf{T}\mathbf{L}\mathbf{x}$ [24].

For time-varying graph signals, some studies assumed that the temporal differences of time-varying graph signals are smooth ([18], [23]). Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M]$ be a time-varying graph signal, where $\mathbf{x}_s \in \mathbb{R}^N$ is a graph signal in G at time s. Qiu *et al.* [23] defined the smoothness of $\mathbf{X}$ as:

$$S_2(\mathbf{X}) = \sum_{s=1}^{M}\mathbf{x}_s^\mathsf{T}\mathbf{L}\mathbf{x}_s = tr(\mathbf{X}^\mathsf{T}\mathbf{L}\mathbf{X}). \tag{2}$$

$S_2(\mathbf{X})$ only computes the summation of the individual smoothness of each graph signal $\mathbf{x}_s \;\forall\; s \in \{1,2,\ldots,M\}$, so we do not consider any temporal information. To address this problem, we can define the temporal difference operator $\mathbf{D}_h$ as follows [23]:

$$\mathbf{D}_h = \begin{bmatrix} -1 & & & \\ 1 & -1 & & \\ & 1 & \ddots & \\ & & \ddots & -1 \\ & & & 1 \end{bmatrix} \in \mathbb{R}^{M\times(M-1)}. \tag{3}$$

Therefore, we have that $\mathbf{X}\mathbf{D}_h = [\mathbf{x}_2-\mathbf{x}_1, \mathbf{x}_3-\mathbf{x}_2, \ldots, \mathbf{x}_M-\mathbf{x}_{M-1}]$. Some studies [18], [23] have found that $S_2(\mathbf{X}\mathbf{D}_h)$ shows better smoothness properties than $S_2(\mathbf{X})$ in real-world time-varying data, *i.e.* $\mathbf{x}_s - \mathbf{x}_{s-1}$ exhibits smoothness in the graph even if $\mathbf{x}_s$ is not smooth across the graph. Qiu *et.al.* [23] used $S_2(\mathbf{X}\mathbf{D}_h)$ to present a Time-varying Graph Signal Reconstruction (TGSR) method as follows:

$$\min_{\tilde{\mathbf{X}}} \frac{1}{2}\|\mathbf{J}\circ\tilde{\mathbf{X}} - \mathbf{Y}\|_F^2 + \frac{\upsilon}{2}tr\left((\tilde{\mathbf{X}}\mathbf{D}_h)^\mathsf{T}\mathbf{L}\tilde{\mathbf{X}}\mathbf{D}_h\right), \tag{4}$$

where $\mathbf{J} \in \{0,1\}^{N\times M}$ is a sampling matrix, $\circ$ is the Hadamard product between matrices, υ is a regularization parameter, and $\mathbf{Y} \in \mathbb{R}^{N\times M}$ is the matrix of observed values. The optimization problem in (4) has some limitations: 1) the solution of (4) could lose performance if the real-world dataset does not satisfy the smoothness prior assumption, and 2) (4) is solved with a conjugate gradient method in [23], which has a slow convergence rate because $S_2(\tilde{\mathbf{X}}\mathbf{D}_h)$ is ill-conditioned ([18]). Our algorithm relaxes the smoothness assumption by introducing a learnable module. Similarly, TimeGNN is fast once the GNN parameters are learned.

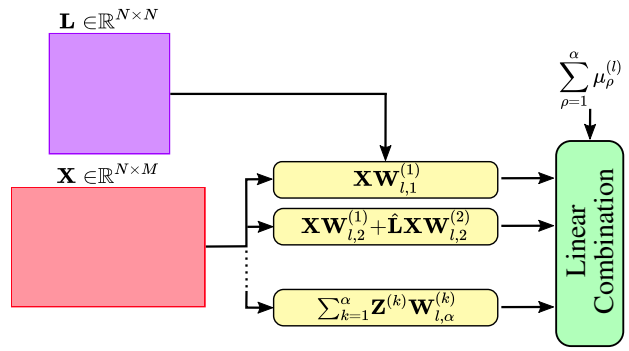

Fig. 1: Cascade of Chebyshev graph convolutions.

C. Graph Neural Network Architecture

TimeGNN is based on the Chebyshev spectral graph convolutional operator defined by Defferrard *et.al.* [2], whose propagation rule is given as follows:

$$\mathbf{X}' = \sum_{k=1}^{K}\mathbf{Z}^{(k)}\mathbf{W}^{(k)}, \tag{5}$$

where $\mathbf{W}^{(k)}$ is the kth matrix of trainable parameters, $\mathbf{Z}^{(k)}$ is computed recursively as $\mathbf{Z}^{(1)} = \mathbf{X}$, $\mathbf{Z}^{(2)} = \hat{\mathbf{L}}\mathbf{X}$, $\mathbf{Z}^{(k)} = 2\hat{\mathbf{L}}\mathbf{Z}^{(k-1)} - \mathbf{Z}^{(k-2)}$, and $\hat{\mathbf{L}} = \frac{2\mathbf{L}}{\lambda_{\max}} - \mathbf{I}$. We use the filtering operation in (5) to propose a new convolutional layer composed of: 1) a cascade of Chebyshev graph filters, and 2) a linear combination layer as in Fig. 1. More precisely, we define the propagation rule of each layer of TimeGNN as follows:

$$\mathbf{H}^{(l+1)} = \sum_{\rho=1}^{\alpha}\mu_\rho^{(l)}\sum_{k=1}^{\rho}\mathbf{Z}^{(k)}\mathbf{W}_{l,\rho}^{(k)}, \tag{6}$$

where $\mathbf{H}^{(l+1)}$ is the output of layer $l+1$, α is a hyperparameter, $\mu_\rho^{(l)}$ is a learnable parameter, $\mathbf{Z}^{(k)}$ is recursively computed as in (5), and $\mathbf{W}_{l,\rho}^{(k)}$ is the kth learnable matrix in the layer l for the branch ρ. The architecture of TimeGNN is given by stacking n cascade layers as in (6), where the input is $(\mathbf{J}\circ\mathbf{X})\mathbf{D}_h$. Finally, our loss function is such that:

$$\begin{aligned}\mathscr{L} = \frac{1}{|\mathscr{S}|}\sum_{(i,j)\in\mathscr{S}}(\mathbf{X}(i,j)-\bar{\mathbf{X}}(i,j))^2 \\ +\lambda tr\left((\bar{\mathbf{X}}\mathbf{D}_h)^\mathsf{T}(\mathbf{L}+\varepsilon\mathbf{I})\bar{\mathbf{X}}\mathbf{D}_h\right),\end{aligned} \tag{7}$$

where $\bar{\mathbf{X}}$ is the reconstructed graph signal, $\mathscr{S}$ is the training set, with $\mathscr{S}$ a subset of the spatio-temporal sampled indexes given by $\mathbf{J}$, and $\varepsilon \in \mathbb{R}^+$ is a hyperparameter. The term $tr\left((\bar{\mathbf{X}}\mathbf{D}_h)^\mathsf{T}(\mathbf{L}+\varepsilon\mathbf{I})\bar{\mathbf{X}}\mathbf{D}_h\right)$ is the Sobolev smoothness ([18]).

We can think of TimeGNN as an encoder-decoder network with a loss function given by an MSE term plus a Sobolev smoothness regularization. The first layers of TimeGNN encode the term $(\mathbf{J}\circ\mathbf{X})\mathbf{D}_h$ to an H-dimensional latent vector that is then decoded with the final layer. As a result, we

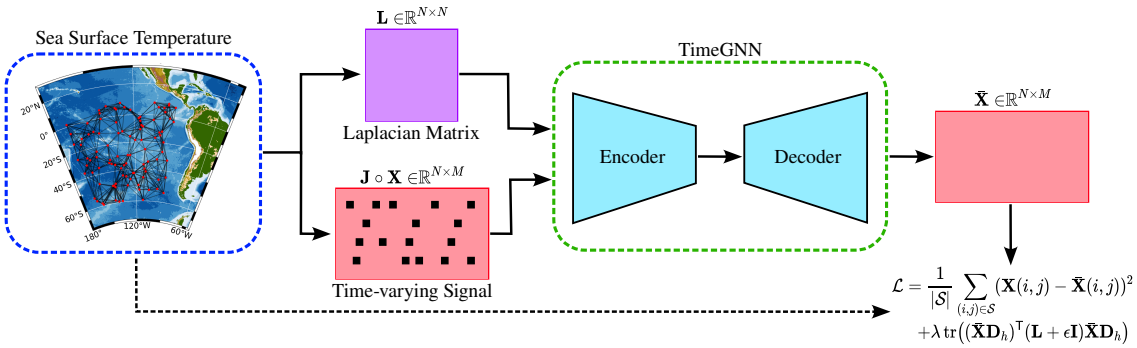

Fig. 2: Pipeline of our Time Graph Neural Network (TimeGNN) for the recovery of time-varying graph signals.

capture the spatio-temporal information using the GNN, the temporal encoding-decoding structure, and the regularization term $tr\left((\bar{\mathbf{X}}\mathbf{D}_h)^\top(\mathbf{L}+\varepsilon\mathbf{I})\bar{\mathbf{X}}\mathbf{D}_h\right)$ where we use the temporal operator $\mathbf{D}_h$. The parameter λ in (7) weighs the importance of the regularization term against the MSE loss. Figure 2 shows the pipeline of our TimeGNN applied to a graph of the sea surface temperature in the Pacific Ocean.

III. Experiments and Results

We compare TimeGNN with Graph Convolutional Networks (GCN) [3], Natural Neighbor Interpolation (NNI) [25], TGSR [23], and Time-varying Graph signal Reconstruction via Sobolev Smoothness (GraphTRSS) [18].

A. Implementation Details

We implement TimeGNN and GCN using PyTorch and PyG [26]. We define the space search for the hyperparameters tuning of TimeGNN as follows: 1) number of layers $\{1,2,3\}$, 2) hidden units $\{2,3,\ldots,10\}$, 3) learning rate $[0.005, 0.05]$, 4) weight decay $[1e-5, 1e-3]$, 5) $\lambda \in [1e-6, 1e-3]$, 6) $\alpha \in \{2,3,4\}$. Similarly, we set the following hyperparameters: 1) $\varepsilon = 0.05$, and 2) the number of epochs to $5,000$. The graphs are constructed based on the coordinate locations of the nodes in each dataset with a k-Nearest Neighbors (k-NN) algorithm as in [18]. NNI, TGRS, and GraphTRSS are implemented using the code in [18] in MATLAB® 2022b. The hyperparameters of the baseline methods are optimized following the same strategy as with TimeGNN.

B. Datasets

Synthetic Graph and Signals: We use the synthetic graph dataset developed in [23]. The graph contains 100 nodes randomly generated from a uniform distribution in a 100×100 square area using k-NN. The graph signals are generated with the recursive function $\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{L}^{-1/2}\mathbf{f}_t$, where $\mathbf{x}_1$ is a low frequency graph signal with energy 10^4, $\mathbf{L}^{-1/2} = \mathbf{U}\lambda^{-1/2}\mathbf{U}^\top$, where $\mathbf{U}$ is the matrix of eigenvectors, $\lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N)$ is the matrix of eigenvalues, $\lambda^{-1/2} = \mathrm{diag}(0, \lambda_2^{-1/2}, \ldots, \lambda_N^{-1/2})$, and $\mathbf{f}_t$ is an i.i.d. Gaussian signal.

PM 2.5 Concentration: We use the daily mean concentration of PM 2.5 in the air in California, USA[2]. Data were collected from 93 sensors over 220 days in 2015.

[2]https://www.epa.gov/outdoor-air-quality-data

Sea-surface Temperature: We use the sea-surface temperature data, which are measured monthly and released by the NOAA PSL[3]. We use a sample of 100 locations in the Pacific Ocean over a duration of 600 months.

Intel Lab Data: We use the data captured by the 54 sensors deployed at the Intel Berkeley Research Laboratory [4]. The data consists of temperature readings between February 28th and April 5th, 2004.

C. Evaluation Metrics

We use the Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) metrics, as defined in [18], to evaluate our algorithm.

D. Experiments

We construct the graphs using k-NN with the coordinate locations of the nodes in each dataset with a Gaussian kernel as in [18]. We follow a random sampling strategy in all experiments. Therefore, we compute the reconstruction error metrics on the non-sampled vertices for a set of sampling densities. We evaluate all the methods with a Monte Carlo cross-validation with 50 repetitions for each sampling density. For the synthetic data, $k = 5$ in the k-NN, and the sampling densities are given by $\{0.1, 0.2, \ldots, 0.9\}$. For PM2.5 concentration, $k = 5$ and the sampling densities are $\{0.1, 0.15, 0.2, \ldots, 0.45\}$. For the sea-surface temperature, we keep $k = 5$ and the sampling densities are set to $\{0.1, 0.2, \ldots, 0.9\}$. For Intel Lab data, we set $k = 3$ and the sampling densities at $\{0.1, 0.3, 0.5, 0.7\}$.

E. Results and Discussion

Figure 3 shows the performance of TimeGNN against the previous methods for all datasets using RMSE. Furthermore, Table I shows the quantitative comparisons using the averages of all metrics along the set of sampling densities. We do not plot the performance of GCN in Fig. 3 because this network performs considerably worse than the other methods, as shown in Table I. GCN was implemented using the same input and loss function as in TimeGNN. Our algorithm outperforms previous methods for several metrics in PM2.5 concentration

[3]https://psl.noaa.gov

[4]http://db.csail.mit.edu/labdata/labdata.html

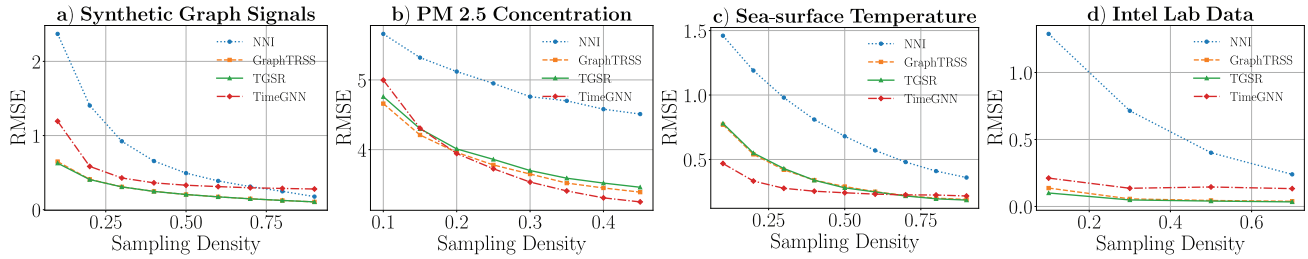

Fig. 3: Comparison of TimeGNN to baseline methods in one synthetic and three real-world datasets (RMSE).

TABLE I: Quantitative comparison of TimeGNN with the baselines in all datasets using the average error metrics.

Method	Synthetic Graph Signals			PM2.5 Concentration			Sea-surface Temperature			Intel Lab Data		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
GCN (Kipf and Welling [3])	11.296	8.446	1.123	4.657	2.959	0.550	3.766	2.922	0.548	2.998	2.327	0.120
NNI (Kiani *et. al.* [25])	0.775	0.436	0.255	4.944	2.956	0.593	0.772	0.561	0.067	0.661	0.291	0.015
GraphTRSS (Giraldo *et. al.* [18])	**0.260**	***0.256***	***0.178***	***3.824***	***2.204***	***0.377***	***0.357***	***0.260***	***0.029***	**0.056**	**0.023**	**0.001**
TGSR (Qiu *et. al.* [23])	***0.263***	**0.193**	**0.144**	3.898	2.279	0.394	0.360	0.263	0.030	***0.069***	***0.037***	***0.002***
TimeGNN (ours)	0.452	0.323	0.226	**3.809**	**2.172**	**0.362**	**0.275**	**0.203**	**0.023**	0.156	0.095	0.005

The best and second-best performing methods on each dataset are shown in **red** and ***blue***, respectively.

and sea-surface temperature datasets. The synthetic data were created to satisfy the conditions of smoothly evolving graph signals (Definition 1 in [23]), while here, we relaxed that prior assumption by adding a trainable GNN module. Therefore, TGRS and GraphTRSS are better suited for that artificial dataset, as shown in Fig. 3 and Table I. Similarly, the Intel Lab dataset is highly smooth. Some of the reasons behind our model's success in real-world datasets are: 1) its ability to capture spatio-temporal information, 2) its encoding-decoding structure, and 3) its powerful learning module given by a cascade of Chebyshev graph convolutions.

IV. Conclusions

In this paper, we introduced a GNN architecture named TimeGNN for the recovery of time-varying graph signals from their samples. Similarly, we proposed a new convolutional layer composed of a cascade of Chebyshev graph filters. TimeGNN includes a learning module that relaxes the requirement of strict smoothness assumptions. We found that our framework shows competitive performance against several approaches in the literature for reconstructing graph signals, delivering better performance in real datasets. Our algorithm could help solve problems like recovering missing data from sensor networks, forecasting weather conditions, intelligent transportation systems, and many others.

For future work, we plan to extend our framework to other graph filters like transformers [27], and alternative compact operators as introduced in [28]. Similarly, we will explore TimeGNN in highly dynamic 4D real datasets [29], [30].

Acknowledgments: This work was supported by DATAIA institute as part of the "Programme d'Investissement d'Avenir", (ANR-17-CONV-0003) operated by CentraleSupélec, by ANR (French National Research Agency) under the JCJC project GraphIA (ANR-20-CE23-0009-01), and by the Office of Naval Research, ONR (Grant No. N00014-21-1-2760).

References

[1] A. Ortega et al., "Graph signal processing: Overview, challenges, and applications," in Proceedings of the IEEE, 2018. 1
[2] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in NeurIPS, 2016. 1, 2
[3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in ICLR, 2017. 1, 3, 4
[4] V. N. Ioannidis, A. G. Marques, and G. B. Giannakis, "A recurrent graph neural network for multi-relational data," in IEEE ICASSP, 2019. 1
[5] F. Gama et al., "Aggregation graph neural networks," in IEEE ICASSP, 2019. 1
[6] A. Duval and F. D. Malliaros, "Higher-order clustering and pooling for graph neural networks," in ACM CIKM, 2022. 1
[7] M. M. Bronstein et al., "Geometric deep learning: going beyond euclidean data," in IEEE Signal Process. Mag., 2017. 1
[8] J. H. Giraldo, S. Javed, and T. Bouwmans, "Graph moving object segmentation," IEEE T-PAMI, 2022. 1
[9] A. Mondal et al., "Moving object detection for event based vision using graph spectral clustering," in IEEE ICCV-W, 2021. 1
[10] A. Benamira et al., "Semi-supervised learning and graph neural networks for fake news detection," in IEEE/ACM ASONAM, 2019. 1
[11] A. G. Marques et al., "Sampling of graph signals with successive local aggregations," in IEEE TSP, 2015. 1, 2
[12] D. Romero et al., "Kernel-based reconstruction of graph signals," in IEEE TSP,2016. 1, 2
[13] D. Ramirez, A. G. Marques, and S. Segarra, "Graphsignal reconstruction and blind deconvolution for diffused sparse inputs," in IEEE ICASSP, 2017. 1
[14] A. Parada-Mayorga et al., "Blue-noise sampling on graphs," in IEEE T-SIPN, 2019. 1
[15] B. Guler et al., "Robust graph signal sampling," in IEEE ICASSP, 2019. 1
[16] B. Girault, A. Ortega, and S. S. Narayayan, "Graph vertex sampling with arbitrary graph signal Hilbert spaces," in IEEE ICASSP, 2020. 1
[17] J. Hara et al., "Design of graph signal sampling matrices for arbitrary signal subspaces," in IEEE ICASSP, 2021. 1
[18] J. H. Giraldo et al., "Reconstruction of time-varying graph signals via Sobolev smoothness," in IEEE T-SIPN, 2022. 1, 2, 3, 4
[19] B. Girault, "Stationary graph signals using an isometric graph translation," in EUSIPCO, 2015.
[20] J. H. Giraldo and T. Bouwmans, "On the minimization of Sobolev norms of time-varying graph signals: Estimation of new Coronavirus disease 2019 cases," in IEEE MLSP, 2020. 1
[21] S. Chen and Y. C. Eldar, "Time-varying graph signal inpainting via unrolling networks," in IEEE ICASSP, 2021. 1
[22] A. Mondal et al., "Recovery of missing sensor data by reconstructing time-varying graph signals," in EUSIPCO, 2022. 1

[23] K. Qiu et al., "Time-varying graph signal reconstruction," in IEEE Journal of Selected Topics in Signal Processing, 2017. 1, 2, 3, 4
[24] D. I. Shuman et al., "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," in IEEE Signal Process. Mag., 2013. 2
[25] K. Kiani and K. Saleem, "K-nearest temperature trends: A method for weather temperature data imputation," in ICISDM, 2017. 3, 4
[26] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in ICLR-W, 2019. 3
[27] S. Yun et al., "Graph transformer networks," in NeurIPS, 2019. 4
[28] F. Ji and W. P. Tay, "A hilbert space theory of generalized graph signal processing," in IEEE TSP, 2019. 4
[29] M. Badiey, L. Wan, and A. Song, "Three-dimensional mapping of evolving internal waves during the Shallow Water 2006 experiment," in J. Acoust. Soc. Am., 2013. 4
[30] J. A. Castro-Correa et al., "Supervised classification of sound speed profiles via dictionary learning," in J. Atmos. Ocean. Technol., 2022. 4