Graph-Time Trend Filtering and Unrolling Network

Sabbaqi, Mohammad ; Isufi, Elvin

DOI
10.23919/EUSIPCO58844.2023.10289885

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the 2023 31st European Signal Processing Conference (EUSIPCO)

Graph-Time Trend Filtering and Unrolling Network

Mohammad Sabbaqi
Delft University of Technology
m.sabbaqi@tudelft.nl

Elvin Isufi
Delft University of Technology
e.isufi-1@tudelft.nl

***Abstract*—Reconstructing missing values and removing noise from network-based multivariate time series requires developing graph-time regularizers capable of capturing their spatiotemporal behavior. However, current approaches based on joint spatiotemporal smoothness, diffusion, or variations thereof may not be effective for time series with discontinuities across the graph or time. To address this challenge, we propose a joint graph-time trend filter operating over a product graph representing spatiotemporal relations. Additionally, we develop a graph-time unrolled neural network to learn the prior from the data, which is based on the alternating direction method of multipliers iterations of the graph-time trend filter and on graph-time convolutional filters. Numerical tests with two synthetic and four real datasets corroborate the effectiveness of both approaches, highlight their inherent trade-offs, and show they compare well with state-of-the-art alternatives.**

***Index Terms*—Graph-time signal processing, graph unrolled networks, trend filtering on graphs.**

I. Introduction

Processing network-based multivariate time series is of interest in a myriad of applications including traffic networks, action recognition, and power networks [1]–[3], to name a few. Two ubiquitous tasks are reconstructing missing values in a spatiotemporal manner and removing noise. These tasks are addressed via graph-time regularization techniques where a joint spatiotemporal prior about the data behavior is forced into the recovery optimization problem. For example, the work in [4] assumes a smoothly evolving model for time-varying graph signals where the regularizer enforces temporal differences to be smooth over the graph. Authors in [5] employ the same assumption but measure the temporal differences smoothness via Sobolev norm to ensure faster convergence and numerical stability. Differently, a stationary time-vertex framework is proposed in [6] and the time-varying graph signals are recovered by solving a Wiener optimization problem. The work in [7] defines a separable smoothness term over temporal and spatial domains as regularizer and unrolls the iterative algorithm for more flexibility. Finally, the authors in [8] use a time-varying kernel and regularize the optimization problem with a kernel-based smoothness to account for both temporal variations in the data and graph itself.

All these works rely on a joint spatiotemporal smoothness, diffusion or variation thereof to reconstruct the signal. The latter are however not useful when the time series present discontinuities either across the graph, time or both. This is for example the case in a traffic network where the velocity of vehicles varies sudden but sparse rather smooth and steady. In these cases, approaches based on sparse differences between adjacent signal values could be of interest as they are suitable to capture sharp transitions in the signal. These approaches have been developed either for univariate time series [9], images [10], or for time invariant graph data [11]. However, extending them to a joint spatiotemporal prior is not straightforward. Here, we use product graphs to create an effective spatiotemporal neighborhood where the nodes can also exchange information with jointly spatial and temporal neighbors. Defining a trend filter over product graph assures piece-wise smoothness in the spatiotemporal domain which enables the model to capture even diffusing trends in the data.

A fundamental challenge with regularization-based approaches is that these priors may be difficult to find or that particular tasks cannot be addressed with simple priors, hence leading to more complex solutions. In these instances, learning the prior from the data or the part it cannot represent allows for sufficient flexibility to address a wider range of tasks while being computationally tractable. A promising approach to achieve the latter goal comprises resorting to model-based neural network solutions, which are deep learning approaches where the propagation rule is based on the algorithm used to solve regularized optimization problem [12]. Such techniques have been widely used for image data [13], seldom for graph time invariant data [14] but remain unexplored for time varying network data. To adopt the latter for our setting, we build upon the ADMM iterations of the graph-time trend filter and employ graph-time convolutional filters in each layer so that to improve flexibility and account for multi-hop spatiotemporal neighboring information, while preserving the inductive biases given by graph structure and the trend filter. In turn, this also reduces the computational complexity of the ADMM solution.

In summary, the contribution of this work is twofold. First, it proposed a joint graph-time trend filter to process network-based multivariate time series containing abrupt transitions (Section II-B). Second, it develops an unrolled neural network for this setting to make the approach more versatile, benefit from the data, and facilitate scalability to larger graphs (Section III). These approaches are corroborated and compared with competing alternatives on two synthetic and four real datasets from traffic and sensor networks (Section IV).

II. Graph-Time Trend Filtering

A. Product Graph Representation of Multivariate Time Series

Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, named *spatial graph* of N nodes in $\mathcal{V} = \{1, \ldots, N\}$ and $|\mathcal{E}|$ edges in $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Let the graph shift operator (GSO) matrix of this graph be $\mathbf{S} \in \mathbb{R}^{N \times N}$ [15]. On top of this graph we have a multivariate time series $\mathbf{x}_t = [x_t(1), \ldots, x_t(N)]^\top \in \mathbb{R}^N$, where each entry is

This work is supported by the TU Delft AI Labs programme.

the time series of the corresponding node. We collect T such realizations in matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$ where now the ith row $\mathbf{x}^i = [x_1(i), \ldots, x_T(i)]^\top$ is a time series of node i with length T. We can also represent the temporal dependencies in $\mathbf{x}^i$ through a *temporal graph* $\mathcal{G}_T = (\mathcal{V}_T, \mathcal{E}_T)$ of T nodes in $\mathcal{V}_T = \{1, \ldots, T\}$, $|\mathcal{E}_T|$ edges in $\mathcal{E}_T \in \mathcal{V}_T \times \mathcal{V}_T$, and corresponding GSO matrix $\mathbf{S}_T$. Each node of $\mathcal{G}_T$ corresponds to one time instant t and the edges are capture temporal dependencies. Thus, the time series $\mathbf{x}^i$ can be represented as a graph signal over the temporal graph $\mathcal{G}$, which may be a simple directed line graph or a more complicated correlation/causation-based structure.

While each of the graphs can be used to process the corresponding signals individually, we will miss their spatiotemporal dependencies. In the context of trend filtering, this implies that we may be able to capture piecewise signal transitions either spatially or temporally but not both. To capture the latter, we use product graphs and build a joint graph-time trend filter. Specifically, given the spatial graph $\mathcal{G}$ and the temporal graph $\mathcal{G}_T$, their product graph is denoted by

$$\mathcal{G}_\diamond = \mathcal{G}_T \diamond \mathcal{G} = (\mathcal{V}_\diamond, \mathcal{E}_\diamond, \mathbf{S}_\diamond) \tag{1}$$

with node set $\mathcal{V}_\diamond = \mathcal{V}_T \times \mathcal{V}$. Here, a node $v_{\diamond,(i,t)} \in \mathcal{V}_T$ stands for a graph-time location; hence, the edges in $\mathcal{E}_\diamond$ connecting graph-time locations are governed by type of the product graph. Typical product graphs include [16]:

- *Kronecker product*: $\mathcal{G}_\otimes = \mathcal{G}_T \otimes \mathcal{G} = (\mathcal{V}_\otimes, \mathcal{E}_\otimes, \mathbf{S}_\otimes)$ has the GSO $\mathbf{S}_\otimes = \mathbf{S}_T \otimes \mathbf{S}$. The number of edges is $|\mathcal{E}_\otimes| = 2|\mathcal{E}||\mathcal{E}_T|$.
- *Cartesian product*: $\mathcal{G}_\times = \mathcal{G}_T \times \mathcal{G} = (\mathcal{V}_\times, \mathcal{E}_\times, \mathbf{S}_\times)$ has the GSO $\mathbf{S}_\times = \mathbf{S}_T \otimes \mathbf{I}_N + \mathbf{I}_T \otimes \mathbf{S}$. The number of edges is $|\mathcal{E}_\times| = T|\mathcal{E}| + N|\mathcal{E}_T|$.
- *Strong product*: $\mathcal{G}_\boxtimes = \mathcal{G}_T \boxtimes \mathcal{G} = (\mathcal{V}_\boxtimes, \mathcal{E}_\boxtimes, \mathbf{S}_\boxtimes)$ has the GSO $\mathbf{S}_\boxtimes = \mathbf{S}_T \otimes \mathbf{I}_N + \mathbf{I}_T \otimes \mathbf{S} + \mathbf{S}_T \otimes \mathbf{S}$. The number of edges is $|\mathcal{E}_\boxtimes| = |\mathcal{E}||\mathcal{E}_T| + T|\mathcal{E}| + N|\mathcal{E}_T|$.

All these product graphs can be seen as particular instances of a parametric product graph with GSO

$$\mathbf{S}_\diamond = \sum_{i=0}^{1} \sum_{j=0}^{1} s_{ij} \left(\mathbf{S}_T^i \otimes \mathbf{S}^j \right). \tag{2}$$

The scalars $\{s_{ij}\}$ could be set at the outset if a prior about the connectivity exits or can be treated as trainable parameters so as to learn also the spatiotemporal dependencies in the data for the task at hand [17], [18].

Column-vectorizing $\mathbf{X}$ into $\mathbf{x}_\diamond = \mathrm{vec}(\mathbf{X}) \in \mathbb{R}^{NT}$ we obtain a product graph signal in which the ith entry is the signal value at the spatiotemporal node $i_\diamond = (i, t)$.

B. Graph-Time Trend Filtering

When the time series in $\mathbf{X}$ present a spatiotemporal piecewise smooth structure, we could use the latter as a prior to process such signals. For example, this is the case in a social network where people in a community share a similar political inclination different from the other communities and their opinion varies smoothly over time. another example can be a traffic network where velocity time series in each node is piecewise smooth while their variation over the road network is smooth. Trend filtering concepts developed over the product graph $\mathcal{G}_\diamond$ could be of interest here to capture the spatiotemporal signal jumps.

Specifically, we column-vectorize $\mathbf{X}$ into $\mathbf{x}_\diamond = \mathrm{vec}(\mathbf{X}) \in \mathbb{R}^{NT}$ to obtain a product graph signal in which the ith entry is the signal value at the spatiotemporal node $i_\diamond = (i, t)$. Then, denoting the incidence matrix of the product graph by $\mathbf{B}_\diamond^{(1)} := \mathbf{B}$ and the measurements by $\mathbf{y}_\diamond = \mathrm{vec}(\mathbf{Y})$, a kth order graph-time trend filter comprises solving

$$\hat{\mathbf{x}}_\diamond = \underset{\mathbf{x}_\diamond \in \mathbb{R}^{NT}}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{y}_\diamond - \mathbf{x}_\diamond\|_2^2 + \gamma \|\mathbf{B}_\diamond^{(k+1)} \mathbf{x}_\diamond\|_1, \tag{3}$$

where γ is the regularization weight and $\mathbf{B}_\diamond^{(k+1)}$ is a higher-order difference operator defined as

$$\mathbf{B}^{(k+1)} = \begin{cases} \mathbf{B}_\diamond \mathbf{B}_\diamond^{(k)} = \mathbf{L}_\diamond^{\frac{k+1}{2}} & k \text{ is odd} \\ \mathbf{B}_\diamond^\top \mathbf{B}_\diamond^{(k)} = \mathbf{B}_\diamond^\top \mathbf{L}_\diamond^{\frac{k}{2}} & k \text{ is even} \end{cases} \tag{4}$$

where $\mathbf{B}_\diamond^\top = \mathbf{B}_\diamond^{(1)}$ is the incidence matrix and $\mathbf{L}_\diamond = \mathbf{S}_\diamond$ is the normalized Laplacian matrix of the product graph. To understand the graph-time trend filtering operation, consider that operation

$$\|\mathbf{B}_\diamond^{(1)} \mathbf{x}_\diamond\|_1 = \sum_{(i_\diamond, i'_\diamond) \in \mathcal{E}_\diamond} |\mathbf{x}_{\diamond, i_\diamond} - \mathbf{x}_{\diamond, i'_\diamond}| \tag{5}$$

measures the spatiotemporal local difference over the product graph $\mathcal{G}_\diamond$. If such differences are sparse (e.g., in spatiotemporal piece-wise signals), this acts as a spatiotemporal regularizer to obtain the signal from noisy and/or incomplete measurements. The kth order graph-time trend filter instead performs differently depending on the order. For an odd order, it performs a difference operation over nodes of the graph. Instead, for an even order it applies the difference operator on the edges. The higher orders of graph-time trend filter perform similar operation but on the shifted time-varying graph signal. Hence, minimizing non-zero values of this spatiotemporal difference leads to a time-varying signal which is piecewise smooth either in spatial or temporal domain. The spatiotemporal resolution to measure the smoothness mainly depends on the type of product graph and imposed spatiotemporal couplings.

The trend filtering operation is directly controlled by the type of product graph. For example, for the Kronecker product operation (5) measures the difference with neighbor previous value. Instead, for the Cartesian product we have either temporal or spatial difference. Finally, if we consider a parametric product graph, its incidence matrix has the form

$$\mathbf{B}_\diamond = \big[\sqrt{s_{00}}\, \mathbf{I}_{NT} \;\big|\; \sqrt{s_{01}}\, \mathbf{I}_T \otimes \mathbf{B} \;\big|\; \sqrt{s_{10}}\, \mathbf{B}_T \otimes \mathbf{I}_N \;\big|\; \sqrt{s_{11}}\, \mathbf{B}_T \otimes \mathrm{ReLU}([1, -1] \otimes \mathbf{B}) \big] \tag{6}$$

with $\mathbf{B}$ and $\mathbf{B}_T$ are the incidence matrices of the spatial graph $\mathcal{G}$ and temporal graph $\mathcal{G}_T$, respectively and $\mathrm{ReLU}(x) = \max\{0, x\}$ is applied element-wise. Therefore, when substituted in (5) the incidence matrices considers local differences

in different ways. The first set of edges $s_{00}\mathbf{I}_{NT}$ stand for self-loops and balance the ratio between changing and keeping the value of each node. The second set $s_{01}\mathbf{I}_T \otimes \mathbf{B}$ represents temporal connections and captures time series smoothness on time axis, whereas the third set $s_{10}\mathbf{B}_T \otimes \mathbf{I}_N$ stands for the spatial dependencies and captures the variation over the graph. Finally, the fourth set $s_{11}\mathbf{B}_T \otimes \mathrm{ReLU}([1,-1] \otimes \mathbf{B})$ represents the spatiotemporal relations and extracts the difference between a node value and its neighbors history.

With this in place, the graph-time trend filtering optimization problem (3) is convex and can be solved via the alternating method of multipliers (ADMM). To this end, we define $\mathbf{z} = \mathbf{B}_\diamond^\top \mathbf{x}_\diamond$ to segregate variables in non-differentiable function and constraint the optimization problem to its residual. Considering the scaled ADMM variable $\mathbf{u}$, the augmented Lagrangian for the optimization problem in (3) is

$$\mathcal{J}(\mathbf{x}_\diamond, \mathbf{z}, \mathbf{u}) = \frac{1}{2}\|\mathbf{y}_\diamond - \mathbf{x}_\diamond\|_2^2 + \gamma\|\mathbf{z}\|_1 + \frac{\rho}{2}\|\mathbf{B}_\diamond^\top \mathbf{x}_\diamond - \mathbf{z} + \mathbf{u}\|_2^2 - \frac{\rho}{2}\|\mathbf{u}\|_2^2 \tag{7}$$

where $\mathbf{z}$ and $\mathbf{u}$ are intermediate variables, and ρ is the ADMM parameter. Alternating the minimization over its variables solves the optimization problem comprising the following equations

$$\mathbf{x}_\diamond^{\ell+1} = (\mathbf{I}_{NT} + \rho\mathbf{S}_\diamond)^{-1}(\mathbf{y}_\diamond + \rho\mathbf{B}_\diamond(\mathbf{z}^\ell - \mathbf{u}^\ell)) \tag{8a}$$

$$\mathbf{z}^{\ell+1} = \mathrm{soft}_\gamma(\mathbf{B}_\diamond^\top \mathbf{x}_\diamond^{\ell+1} + \mathbf{u}^\ell) \tag{8b}$$

$$\mathbf{u}^{\ell+1} = \mathbf{u}^\ell + \mathbf{B}_\diamond^\top \mathbf{x}_\diamond^{\ell+1} - \mathbf{z}^{\ell+1} \tag{8c}$$

where $\mathrm{soft}_\gamma(\cdot)$ is a standard soft threshold function as the proximal operator of l_1-norm term and enforces piecewise smoothness. Both intermediate variables $\mathbf{z}$ and $\mathbf{u}$ are initialized by zero vectors.

III. Graph-Time Trend Unrolling

The graph-time trend filter has three critical aspects. First, its prior should be strongly present in the data since the solution of (3) is biased by the regularizer. Second, it is useful for limited spatial graphs and a small temporal windows since the inverse in (8a) has a cubic cost $\mathcal{O}((NT)^3)$. Third, the ADMM may require thousands of iterations to reach a local minima, which adds to the computational effort. When these aspects become a challenge, we can resort to unrolled neural networks, which exploit a recursion used to solve the optimization problem for building a data-driven solution [12].

In the graph-time trend filtering case, we build upon the ADMM iterations (8a)-(8c) and replace the inverse in (8a) with a graph-time convolutional filter (GTConv. filter) of the form

$$\mathbf{H}(\mathbf{S}_\diamond) = \sum_{k=0}^{K} h_k \mathbf{S}_\diamond^k \tag{9}$$

where $\mathbf{h} = [h_1, \ldots, h_K]^\top$ is the filter coefficient vector that we estimate for the task at hand [19]. The filter order K controls the spatiotemporal resolution of the filter and allows gathering at each node $i_\diamond = (i,t)$ signal information from its neighbors and itself that is at most K hops away in the product graph; see [19] for a detailed discussion. Replacing filter (9) into (8a) and adding also a nonlinearity to enhance its representation power, the graph-time unrolled network has the propagation rule at layer ℓ

$$\mathbf{x}_\diamond^{\ell+1} = \sigma\big(\mathbf{H}_\ell(\mathbf{S}_\diamond)(\mathbf{x}_\diamond^\ell + \rho\mathbf{B}_\diamond(\mathbf{z}^\ell - \mathbf{u}^\ell))\big) \tag{10a}$$

$$\mathbf{z}^{\ell+1} = \mathrm{soft}_\gamma(\mathbf{B}_\diamond^\top \mathbf{x}_\diamond^{\ell+1} + \mathbf{u}^\ell) \tag{10b}$$

$$\mathbf{u}^{\ell+1} = \mathbf{u}^\ell + \mathbf{B}_\diamond^\top \mathbf{x}_\diamond^{\ell+1} - \mathbf{z}^{\ell+1} \tag{10c}$$

where $\sigma(\cdot)$ is an activation function and the measurement variable $\mathbf{y}_\diamond$ is replaced by the previous layer input $\mathbf{x}_\diamond^k$. Notice that computing the update $\mathbf{x}_\diamond^{\ell+1}$ in (10a) has now a computational complexity of order $\mathcal{O}(KT(N + 2|\mathcal{E}|))$ in a worst case scenario assuming the strong product graph and a cyclic temporal graph. The convolutional filter has reduced the computational complexity from a cubic order into a linear one w.r.t. product graph dimension as it operates locally in the vertex domain over a commonly sparse set of edges.

The graph-time unrolled network has as input a vectorized time-varying graph signal $\mathbf{x}_\diamond^0 = \mathbf{y}_\diamond$, which, for instance, is a noisy observation in the denoising task. This input is propagated following (10a)-(10b) for $\ell = 1, \ldots, L-1$. The output of the final layer L constitutes the output of the network, which can be represented in the compact form $\mathbf{y}_\diamond \to \Phi(\mathbf{y}_\diamond; \mathbf{S}_\diamond, \mathcal{H}) := \mathbf{x}_\diamond^\star$, where set $\mathcal{H}$ collects the KL trainable parameters. These parameters are estimated by solving an end-to-end empirical risk minimization problem

$$\mathcal{H}^\star = \underset{\mathcal{H}}{\mathrm{argmin}} \ \frac{1}{m}\sum_{i=1}^{m} \|\mathbf{x}_\diamond^\star - \mathbf{x}_\diamond\|_2^2 \tag{11}$$

where we have m observation pairs $(\mathbf{x}_\diamond, \mathbf{y}_\diamond)$ in the dataset. The intermediate variable $\mathbf{z}$ and $\mathbf{u}$ can be initialized as zero vectors, and the hyperparameters ρ and γ need to be selected by fine-tuning. The fact that the unrolled network is build upon the ADMM iterations and the product graph limits the function the search space and serves as a strong inductive bias, thus requiring a limited number of parameters to solve the task. This is quite desirable when we have limited training samples. Moreover, the fact that we now learn multi-hop resolution information with the filters improves the flexibility w.r.t. the trend filter (3) when the bias is not strongly present in the data.

Remark 1: The role of the GTConv. filter (9) in (10a) could also be seen as a more flexible way to represent the Neumann expansion of the inverse. Specifically, we have that $(\mathbf{I}_{NT} + \rho\mathbf{S}_\diamond)^{-1} = \sum_{k=0}^{\infty}(-1)^k\rho^k\mathbf{S}_\diamond^k$ with $\rho\|\mathbf{S}_\diamond\| \ll 1$. Hence, the GTConv. filter could be seen as a more flexible truncated inverse series where instead of using the kth power of a single scalar $(-1)^k\rho^k$ to weigh the information of k-hop neighbors it learns specific parameters h_k.

IV. Numerical Experiments

In this section, we evaluate the proposed model performance on denoising and reconstruction tasks over synthetic and real-world data. We compare graph-time unrolling network with the following baselines:

- **BatchRTVG:** Batch reconstruction of time-varying graph signals [4] uses gradient method to solve an optimization problem that its cost function minimizes the smoothness of all the graph signals one-step shifted over time.
- **GUTF:** Graph unrolling trend filtering proposed in [14] which solves and unrolls the trend filtering problem over a spatial graph. The temporal data will be fed to this model as distinct features.
- **GraphTRSS:** Time-varying graph signal reconstruction via Sobolev smoothness [20] solves an optimization problem regularized with a smoothness term over dilated graph signals to reconstruct time-varying graph signals.

The experiments contain two tasks of reconstruction and denoising of time-varying graph signals. All the experiments are applied on the following datasets:

- **Synthetic data:** For the synthetic data, we generate an undirected stochastic block model with $C = 5$ communities and $N = 100$ nodes. The edges are independent random variables with probability of p_e for nodes in different communities and $p_i = 0.8$ for nodes in a community. The time-varying graph signals are originated from a smooth random signal over each community $\mathbf{x}_c = \mathcal{N}(\mu_c, L_c^{\dagger})$ and diffused over the graph with operator $\exp^{-t\mathbf{L}}$.
- **Traffic data:** PEMS-BAY and METR-LA datasets are used in this category. PEMS-BAY contains six months of traffic measurements over 325 nodes in Bay area with a 5 minutes resolution. METR-LA includes the traffic data of 207 nodes located on the highways of Los Angeles County with same resolution. The shift operator is a directed adjacency matrix extracted from the road distance matrix.
- **Weather data:** We evaluated our model over two benchmark weather datasets, Molene and NOAA. The Molene dataset contains 744 weather recordings over 32 stations in a region of France with an hourly resolution. The NOAA dataset consists of 8579 hourly temperature measurements across 109 stations in the U.S.. The adjacency matrix is a normalized Laplacian matrix deployed by using a Gaussian kernel over the distance matrix.

In all the experiments, the ADAM optimizer is used to train the model and an unweighted directed graph is the temporal graph $\mathcal{G}_T$. The performance is measured using root normalized mean square error (rNMSE).

Reconstruction In this task, we mask the original data randomly to generate missing values where each entry of the mask $\mathbf{J}$ is an independent Bernoulli random variable with probability p. Having the masked data, we divide it into $50\%-20\%-10\%$ splits for train, validation, and test, respectfully. We generate the dataset by extracting $\mathbf{X} \in \mathbb{R}^{N\times T}$ data points in each split. Finally, we train our model with 5 layers, GTConv. filter order $K = 3$, and temporal window $T = 5$.

Table I indicates the results for $p = 0.75$. On the synthetic data, the proposed method and GUTF outperform the others as they both use piecewise smoothness. However, in $p_e = 0.8$ case, this assumption is violated and their performances decrease, yet, they perform better than the rest due to having learnable parameters. In the traffic dataset, learning methods outperform the others again, but the proposed model performs better than GUTF as it considers temporal dependencies in the data. The parametric and recursive model are having better results since they can adapt the spatiotemporal structure during the training phase. Finally, similar pattern to traffic data occurs in the NOAA dataset while Molene results differ. The learnable models underperform as the amount of available data in this experiment is not enough and also the data is highly smooth over both graph and time.

Denoising

In this task, we add white Gaussian noise to our data and the goal is removing the noise and reconstructing the original data. The learning setup is similar to the reconstruction task but for 8 layers.

Table 2 presents the denoising performance for signal to noise ratio $\mathrm{SNR} = 5dB$. It can be observed the unrolling models outperform others as they learn parameters and adapt themselves to the task at hand. The only exception is the Molene dataset where the unrolling models fail due to the lack of data. The proposed model mostly performs better as it considers the spatiotemporal connections in the data and the parametric model even learns the spatiotemporal coupling while training. It should be noted that the unrolling model succeeds to detect pievewise smooth patterns in the data through its structure while the objective function is the MSE between the output and the label. This can be seen in synthetic data experiment on $p_e = 0.8$ where the violation of piecewise smoothness leads to the reduction of performance.

Product graph's role: We investigated the proposed model performance for different types of product graph on distinct applications. As Table III suggests, the unrolling models perform better than optimization ones since they adapt learning parameters to the task during training. The difference accentuates in the NOAA dataset where the piece-wise smoothness assumption breaks, yet, unrolling models manage to reconstruct while ADMM ones fail. Results also indicate that the type of product graph highly depends on the task so performance is independent of the product graph type in general. This observation supports the performance of parametric product graph which adapts the product graph structure to the application.

V. Conclusion

This work proposes a framework to process network-based multivariate time series with abrupt transitions over the graph, time, or both. This is achieved by first using product graphs to represent the spatiotemporal signal proximities and then developing a joint graph-time trend filter on this structure. When such a prior is not present in the data or computationally demanding, we put forward a graph-time unrolled neural network. This network builds its layers following the ADMM iterations used to solve the graph-time trend filter and replaces matrix inverse operations with graph-time convolutional filters. Our experiments on signal interpolation and denoising on six

TABLE I
RECONSTRUCTION ERROR IN TERMS OF RNMSE FOR DIFFERENT DATASET AND METHODS FOR $p = 0.75$.

Reconstruction (rNMSE)	Synthetic		Traffic		Weather	
	$p_e = 0.2$	$p_e = 0.8$	PeMS-Bay	METR-LA	NOAA	Molene
BatchRTVG [4]	0.2037	0.1891	0.2408	0.2472	0.2973	0.1880
GraphTRSS [20]	0.2011	0.1904	0.2376	0.2341	0.2510	**0.1829**
GT-TF [Ours]	0.1644	0.2015	0.1861	0.1899	0.2711	0.2163
GUTF [14]	0.1473	0.1610	0.1629	0.1601	0.1553	0.2239
GT-TFU [Ours]	**0.1297**	**0.1382**	**0.1495**	**0.1471**	**0.1429**	0.1992

TABLE II
DENOISING ERROR IN TERMS OF RNMSE FOR DIFFERENT DATASET AND METHODS FOR $\mathrm{SNR} = 5dB$.

Denoising (rNMSE)	Synthetic		Traffic		Weather	
	$p_e = 0.2$	$p_e = 0.8$	PeMS-Bay	METR-LA	NOAA	Molene
BatchRTVG [4]	0.1978	0.1847	0.2112	0.2235	0.2569	0.1903
GraphTRSS [20]	0.1899	0.1824	0.2146	0.2092	0.2473	**0.1815**
GT-TF [Ours]	0.1572	0.1913	0.1733	0.1856	0.2694	0.1996
GUTF [14]	0.1349	0.1510	0.1477	0.1393	0.1509	0.2189
GT-TFU [Ours]	**0.1084**	**0.1095**	**0.1156**	**0.1254**	**0.1219**	0.1927

TABLE III
RECONSTRUCTION ERROR IN TERMS OF RNMSE FOR DIFFERENT PRODUCT GRAPH TYPES WHERE $p = 0.75$.

	Reconstruction (rNMSE)	$p_e = 0.2$	PeMS-Bay	NOAA
ADMM	Cartesian	0.1701	0.1861	0.3043
	Kronecker	0.1644	0.1879	0.2899
	Strong	0.1729	0.2083	0.2711
Unrolling	Cartesian	0.1577	0.1848	0.2309
	Kronecker	0.1603	0.1822	0.2164
	Strong	0.1694	0.1935	0.2291
	Parametric	0.1297	0.1495	0.1429

synthetic and real-world datasets corroborate the effectiveness of the proposed prior, as well as show superior performance when compared with alternative solutions. Future work will focus on characterizing the convergence and stability of the unrolled network.

REFERENCES

[1] W. Jiang and J. Luo, "Graph neural network for traffic forecasting: A survey," *Expert Systems with Applications*, vol. 207, p. 117921, 2022.

[2] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018.

[3] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 1409–1416, Jul. 2019.

[4] K. Qiu, X. Mao, X. Shen, X. Wang, T. Li, and Y. Gu, "Time-varying graph signal reconstruction," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 870–883, 2017.

[5] 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 *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2020, pp. 1–6.

[6] N. Perraudin, A. Loukas, F. Grassi, and P. Vandergheynst, "Towards stationary time-vertex signal processing," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 3914–3918.

[7] S. Chen and Y. C. Eldar, "Time-varying graph signal inpainting via unrolling networks," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 8092–8097.

[8] D. Romero, V. N. Ioannidis, and G. B. Giannakis, "Kernel-based reconstruction of space-time functions on dynamic graphs," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 856–869, 2017.

[9] S.-J. Kim, K. Koh, S. Boyd, and D. Gorinevsky, "ℓ_1 trend filtering," *SIAM Review*, vol. 51, no. 2, pp. 339–360, 2009.

[10] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin, "An iterative regularization method for total variation-based image restoration," *Multiscale Modeling & Simulation*, vol. 4, no. 2, pp. 460–489, 2005.

[11] Y.-X. Wang, J. Sharpnack, A. Smola, and R. Tibshirani, "Trend Filtering on Graphs," in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, vol. 38. PMLR, 09–12 May 2015, pp. 1042–1050.

[12] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Processing Magazine*, vol. 38, no. 2, pp. 18–44, 2021.

[13] Y. Li, M. Tofighi, J. Geng, V. Monga, and Y. C. Eldar, "Efficient and interpretable deep blind image deblurring via algorithm unrolling," *IEEE Transactions on Computational Imaging*, vol. 6, pp. 666–681, 2020.

[14] S. Chen, Y. C. Eldar, and L. Zhao, "Graph unrolling networks: Interpretable neural networks for graph signal denoising," *IEEE Transactions on Signal Processing*, vol. 69, pp. 3699–3713, 2021.

[15] E. Isufi, F. Gama, D. I. Shuman, and S. Segarra, "Graph filters for signal processing and machine learning on graphs," *https://arxiv.org/abs/2211.08854*, 2022.

[16] A. Sandryhaila and J. M. Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 80–90, 2014.

[17] A. Natali, E. Isufi, M. Coutino, and G. Leus, "Learning time-varying graphs from online data," *IEEE Open Journal of Signal Processing*, vol. 3, pp. 212–228, 2022.

[18] M. Sabbaqi, R. Taormina, A. Hanjalic, and E. Isufi, "Graph-time convolutional autoencoders," in *Proceedings of the First Learning on Graphs Conference*, ser. Proceedings of Machine Learning Research, B. Rieck and R. Pascanu, Eds., vol. 198. PMLR, 09–12 Dec 2022, pp. 24:1–24:20.

[19] M. Sabbaqi and E. Isufi, "Graph-time convolutional neural networks: Architecture and theoretical analysis," *https://arxiv.org/abs/2206.15174*, 2022.

[20] J. H. Giraldo, A. Mahmood, B. Garcia-Garcia, D. Thanou, and T. Bouwmans, "Reconstruction of time-varying graph signals via sobolev smoothness," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 8, pp. 201–214, 2022.