Graph-Time Convolutional Neural Networks
Architecture and Theoretical Analysis

Sabbaqi, Mohammad; Isufi, Elvin

DOI
10.1109/TPAMI.2023.3311912

Publication date
2023

Document Version
Final published version

Published in
IEEE Transactions on Pattern Analysis and Machine Intelligence

Graph-Time Convolutional Neural Networks: Architecture and Theoretical Analysis

Mohammad Sabbaqi and Elvin Isufi

***Abstract*—Devising and analysing learning models for spatiotemporal network data is of importance for tasks including forecasting, anomaly detection, and multi-agent coordination, among others. Graph Convolutional Neural Networks (GCNNs) are an established approach to learn from time-invariant network data. The graph convolution operation offers a principled approach to aggregate information and offers mathematical analysis by exploring tools from graph signal processing. This analysis provides insights into the equivariance properties of GCNNs; spectral behaviour of the learned filters; and the stability to graph perturbations, which arise from support perturbations or uncertainties. However, extending the convolutional learning and respective analysis to the spatiotemporal domain is challenging because spatiotemporal data have more intrinsic dependencies. Hence, a higher flexibility to capture *jointly* the spatial and temporal dependencies is required to learn meaningful higher-order representations. Here, we leverage product graphs to represent the spatiotemporal dependencies in the data and introduce Graph-Time Convolutional Neural Networks (GTCNNs) as a principled architecture. We also introduce a parametric product graph to learn the spatiotemporal coupling. The convolution principle further allows a similar mathematical tractability as for GCNNs. In particular, the stability result shows GTCNNs are stable to spatial perturbations. owever, there is an implicit trade-off between discriminability and robustness; i.e., the more complex the model, the less stable. Extensive numerical results on benchmark datasets corroborate our findings and show the GTCNN compares favorably with state-of-the-art solutions. We anticipate the GTCNN to be a starting point for more sophisticated models that achieve good performance but are also fundamentally grounded.**

***Index Terms*—Graph convolutional neural networks, graph signal processing, graph-time neural networks, stability to perturbations.**

I. Introduction

LEARNING from *multivariate temporal* data is a challenging task due to their intrinsic spatiotemporal dependencies. This problem arises in applications such as time series forecasting, classification, action recognition, and anomaly detection [2], [3], [4], [5]. The spatial dependencies can be captured by a graph either explicitly such as in transportation networks or implicitly such as in recommender systems [6]. Therefore, graph-based inductive biases should be considered during learning to exploit the spatial dependencies alongside with temporal patterns in a computationally and data efficient manner. Based on advances in processing and learning over graphs [7], [8], a handful of approaches have been proposed to learn from multivariate temporal data [9]. The main challenge is to capture the spatiotemporal dependencies by built-in effective biases in a principled manner [10].

The convolution principle has been key to build learning solutions for graph-based data [11], [12]. By cascading graph convolutional filters and pointwise nonlinearities, graph convolutional neural networks (GCNNs) have been developed as non-linear models for graph-based data [13], [14]. Such a principle reduces both the number of learnable parameters and the computational complexity in the GCNN, ultimately, overcoming the curse of dimensionality [10], [12]. The convolution operation allows also for a mathematical tractability of GCNNs. On the one hand, it is a permutation equivariant operation, implying that it can capture structural symmetries to aid learning [12], [15]. On the other hand, it enjoys a spectral duality by means of a graph signal processing analysis [7], which, in turn, can be used to characterize the stability of GCNNs to perturbations in the support [15], [16], [17], [18]. The overreaching goal of this paper is to leverage the convolution principle to learn from spatiotemporal data as well as study the theoretical properties of this model.

A. Related Works

There are two stream of works related to this paper: one developing architectures; and one studying their stability.

Architectures. Spatiotemporal graph-based learning models can be divided into hybrid and fused models. *Hybrid* models combine distinct learning algorithms for graph and temporal data. They use GCNNs to extract higher-level spatial features and process these features by a temporal RNN, CNN, or variants of them. The works in [19], [20], [21] concatenate a GCNN by an LSTM where the former is applied per timestamp and the latter per node. Instead, the work in [22] uses an RNN followed by a GCNN. Another work in [2] uses a narrower graph convolutional layer in between two temporal gated convolution layers, whereas [23] combines graph diffusion with temporal convolutions. Another approach is to ignore the graph structure and use temporal CNNs augmented by the attention mechanism

Manuscript received 22 June 2022; revised 23 May 2023; accepted 17 August 2023. Date of publication 5 September 2023; date of current version 3 November 2023. This work was supported by the TU Delft AI Labs program and by the TTW-OTP project GraSPA under Grant 19497 financed by the Dutch Research Council (NWO) part of this work has been presented in [1]. Recommended for acceptance by T.-Y. Liu PhD. *(Corresponding author: Mohammad Sabbaqi.)*

The authors are with the Intelligent Systems Department, Delft University of Technology, 2628, CD Delft, The Netherlands (e-mail: m.sabbaqi@tudelft.nl; e.isufi-1@tudelft.nl).

This article has supplementary downloadable material available at https://doi.org/10.1109/TPAMI.2023.3311912, provided by the authors.

Digital Object Identifier 10.1109/TPAMI.2023.3311912

to reduce the number of parameters and ease training [24]. Graph WaveNet uses a gated dilated temporal convolution followed by a first order graph convolution [25]. *Fused* models impose the graph structure into conventional spatiotemporal models to jointly capture the spatiotemporal relationships [3], [26], [27], [28], [29]. The fully connected blocks are replaced by graph convolutions to generate graph-based latent embeddings. The work in [26] proposed graph-based VARMA to learn spatiotemporal embeddings. The authors in [27], [29] employ variants of RNN models with graph convolution-based blocks, while [28] also designs a graph-based gating module. In another approach, the spatial graph is passed through a continuous temporal shift operator to define a spatiotemporal convolution [30]. The work in [3] starts with a graph partitioning and defines a spatiotemporal neighborhood in each partition to perform message passing. The works in [31], [32] use diffusion schemes where the output is a function of the signals of at most one hop neighbors. Since diffusion is considered in a causal fashion the information from the k-hop spatiotemporal neighbors is considered in subsequent layers, which may be limiting to extract patterns.

The advantage of hybrid models lies within their simple and efficient implementation since they benefit from modular spatial and temporal blocks. However, this modularity makes it unclear how to best combine them for the task at hand and analyze theoretically their inner-working mechanisms. Moreover, they are disjoint and sequential in nature (first graph and then temporal processing or vice-versa); thus, are more restrictive and require each node to accumulate global information. Instead, fused models overcome these issues but they mostly use a low-order spatiotemporal aggregation which limits their effectiveness in capturing spatiotemporal patterns. We address this challenge by leveraging product graphs to represent multivariate time series [33]. Product graphs have been widely used for modeling complex data [34], the matrices [35], and developing a graph-time signal processing framework [36]. However, their use for learning from spatiotemporal data has been limited. The work in [37] builds spatiotemporal scattering transforms using product graphs and shows their advantages over alternative data representative solutions. Here, we use product graphs as a platform to run spatiotemporal convolutions and build neural network solutions with it. The work in [3] implicitly uses the Cartesian product graph to build a spatiotemporal neighborhood. Differently, we develop a principled convolutional framework for learning via any product graph, multi-resolution neighborhood information aggregation in each layer ([3] focuses only on immediate neighbors), and introduce a parametric product graph to learn the spatiotemporal coupling.

Stability. Studying the stability of graph neural networks to graph perturbation reveals their potentials and limitations w.r.t. the underlying support. The work in [16] provided a stability scheme to relative perturbations on the graph for GCNNs by enforcing graph convolutional filters to vary smoothly over high graph frequencies. Authors in [38] proposed a linear stability bound to graph arbitrary perturbation for a large class of graph filters named Cayley smooth space. The work in [39] considers an edge rewiring model for perturbation and provides an upper bound for output changes. The authors in [17] proved that GCNNs are stable to stochastic perturbation model where links have a probability of existence in the graph. In case of spatiotemporal graph neural networks, the work in [28] developed a stability analysis to graph perturbation following the model in [16]. The authors in [30] have used the relative perturbation model in [16] to investigate their model stability properties. We also pursue a similar analysis as [16] for the graph-time convolutional neural network to approve its capabilities and observe the effects of time component in the stability and transferability properties of them. The similar base of stability analysis allows us to compare the proposed model with previous works in [28], [30] and collect insights into how different learning solutions for spatiotemporal data handle uncertainty in the spatial support. Differently from the latter, the proposed stability results via product graphs provides direct links with that of GCNNs if time is invariant, highlighting also the impact of how the temporal component is accounted for a joint spatiotemporal learning.

B. Contribution

We consider the convolutional principle over product graphs as an inductive bias to build graph-time convolutional neural network (GTCNN) for learning from multivariate temporal data. Our specific contributions are:

- C1) *Principled architecture:* We develop an architecture that leverages graph-time convolutions and product graphs as a spatiotemporal inductive bias to reduce the computational cost and the number of trainable parameters. We also propose a solution to learn the spatiotemporal coupling and a recursive implementation to tackle the high dimensionality of the product graphs.
- C2) *Theoretical properties:* We prove GTCNNs are permutation equivariant. Then, leveraging concepts from graph-time signal processing, we show the learned filters act in the joint graph and temporal spectral domain as point-wise multiplication between the learnt frequency response and the graph-time Fourier transform of the input.
- C3) *Stability:* We prove the GTCNN is stable to perturbations in the spatial graph. Our result shows the GTCNN becomes less stable for larger graphs and that the temporal window in the product graph affects it more severely than increasing the number of nodes. We provide a thorough discussion of our stability bound w.r.t. baselines and alternative approaches.
- C4) *Numerical performance:* We compare the GTCNN with baseline and state-of-the-art solutions in different tasks about time series classification and forecasting. The GTCNN outperforms the baseline GCNN model that ignores the temporal dimension and compares well with alternatives on benchmark datasets.

The structure of this paper is as follows. Section II, introduces product graphs and signals over them to formulate our problem. In Section III-B, the GTCNN is proposed and its properties are discussed. Section IV is dedicated to stability analysis of GTCNNs. Section V contains the numerical results and experiments

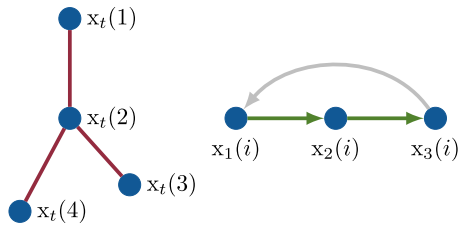


Fig. 1. Spatial and temporal graphs along with signals over their nodes. (Left) Spatial graph and graph signal at time t. Scalar $x_t(2)$ is the signal and node two recorded at time t. It is proximal with signals at nodes one, three, and four. (Right) Temporal graph and ith time series illustrated as graph signal. Edges in green are those of a directed line graph, while edges green and grey are of a cyclic graph.

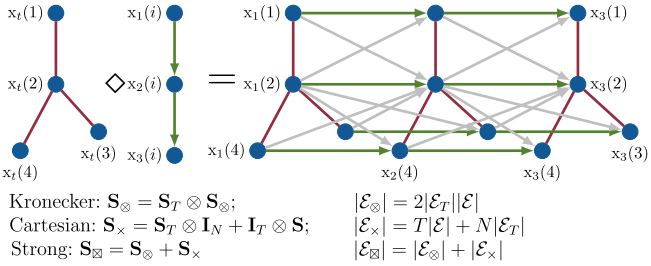


Fig. 2. Different product graphs and the formation of multivariate temporal data over them. (grey edges) Kronecker product $\mathcal{G}_\otimes = \mathcal{G}_T \otimes \mathcal{G}$. (green and red edges) Cartesian product $\mathcal{G}_\times = \mathcal{G}_T \times \mathcal{G}$. (All edges) Strong product $\mathcal{G}_\boxtimes = \mathcal{G}_T \boxtimes \mathcal{G}$..

to corroborate the strengths of GTCNNs. Finally, Section VI concludes the paper.

II. Problem Formulation

Here, we first introduce some background material about product graphs and their use to represent multivariate time-series. Then, we motivate the problem of learning from multivariate time-series via an inductive bias perspective.

A. Product Graphs for Spatiotemporal Coupling

Consider an $N \times 1$ multivariate signal $\mathbf{x}_t$ collected over T time instances in matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$. The tth column $\mathbf{x}_t = [x_t(1), \ldots, x_t(N)]^\top$ may be measurements of different sensors in a sensor network at time t, whereas the ith row $\mathbf{x}^i = [x_1(i), \ldots, x_T(i)]^\top$ may be the time series of sensor i over the T time instants. When learning representations from the data in $\mathbf{X}$ we are interested in exploiting both the spatial dependencies in $\mathbf{x}_t$ and the temporal dependencies in $\mathbf{x}^i$. However, despite learning from these spatial and temporal relations is important to extract patterns in $\mathbf{X}$, we need to devise methods that exploit them *jointly* as an inductive bias for learning representations [10].

When data $\mathbf{x}_t$ have a (hidden) underlying structure, we can capture their *spatial* relations through a *spatial graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of N nodes in set $\mathcal{V} = \{1, \ldots, N\}$ and $|\mathcal{E}|$ edges in set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. We represent the structure of $\mathcal{G}$ via its graph shift operator (GSO) matrix $\mathbf{S} \in \mathbb{R}^{N \times N}$ which is a sparse matrix with non-zero entries $[\mathbf{S}]_{ij} \neq 0$ only if $(i, j) \in \mathcal{E}$ or $i = j$ [7]. We can view $\mathbf{x}_t$ as a graph signal with scalar $x_t(i)$ residing on node i. Likewise, we can capture the *temporal* relations in $\mathbf{x}^i$ through a *temporal graph* $\mathcal{G}_T = (\mathcal{V}_T, \mathcal{E}_T)$ of T nodes in set $\mathcal{V}_T = \{1, \ldots, T\}$ and $|\mathcal{E}_T|$ edges in set $\mathcal{E}_T \in \mathcal{V}_T \times \mathcal{V}_T$. Each node represents one time instant t and set $\mathcal{E}_T$ contains an edge if signals at instances t and t' are related. Also, we represent the structure of $\mathcal{G}_T$ with its GSO matrix $\mathbf{S}_T \in \mathbb{R}^{T \times T}$. Data $\mathbf{x}^i$ can be seen as a graph signal with scalar $x_t(i)$ being the value at node t. Examples of $\mathcal{G}_T$ are the directed line graph that assumes signal $x_t(i)$ depends only on the former instance $x_{t-1}(i)$, the cyclic graph that accounts for periodicity, or any other graph that encodes the temporal dependencies in $\mathbf{x}^i$, see Fig. 1.

Given graphs $\mathcal{G}$, $\mathcal{G}_T$, we can capture the spatiotemporal relations in the data $\mathbf{X}$ through product graphs

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

The node set $\mathcal{V}_\diamond = \mathcal{V}_T \times \mathcal{V}$ is the Cartesian product between $\mathcal{V}_T$ and $\mathcal{V}$ with cardinality $|\mathcal{V}_\diamond| = NT$ and node $i_\diamond \in \mathcal{V}$ represents the space-time location (i, t); see Fig. 2. The edge set $\mathcal{E}_\diamond \subseteq \mathcal{V}_\diamond \times \mathcal{V}_\diamond$ connects now space-time locations, which structure and respective GSO $\mathbf{S}_\diamond \in \mathbb{R}^{NT \times NT}$ are dictated by the type of product graph. Typical product graphs are [33]:

- *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 $|\mathcal{E}_\otimes| = 2|\mathcal{E}||\mathcal{E}_T|$. The Kronecker product translates the spatial coupling into a temporal coupling in which spatiotemporal nodes $i_\diamond = (i, t)$ and $j_\diamond = (j, t+1)$ are connected only if the spatial nodes i, j are connected in $\mathcal{G}$ and the temporal nodes t and $t+1$ are connected in $\mathcal{G}_T$; grey edges in Fig. 2.
- *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 $|\mathcal{E}_\times| = T|\mathcal{E}| + N|\mathcal{E}_T|$. The Cartesian product implies a temporal coupling by connecting each spatial node to itself in consecutive instants as dictated by $\mathcal{G}_T$, i.e., the spatiotemporal nodes $i_\diamond = (i, t)$ and $j_\diamond = (i, t+1)$ are connected if time instants t and $t+1$ are adjacent; red and green edges in Fig. 2.
- *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 $|\mathcal{E}_\boxtimes| = |\mathcal{E}||\mathcal{E}_T| + T|\mathcal{E}| + N|\mathcal{E}_T|$. The strong product is the union of the Kronecker and Cartesian products. Here, spatiotemporal nodes $i_\diamond = (i, t)$ and $j_\diamond = (j, t+1)$ are connected if time instants t and $t+1$ are adjacent in the temporal graph $\mathcal{G}_T$ and spatial nodes i and j are either neighbors or $i = j$ in graph $\mathcal{G}$; red, green, and grey edges in Fig. 2.

Column-vectorizing $\mathbf{X}$ into $\mathbf{x}_\diamond = \text{vec}(\mathbf{X}) \in \mathbb{R}^{NT}$ we obtain a product graph signal in which the ith entry is the signal value at the spatiotemporal node $i_\diamond$. Some of such values are illustrated in Fig. 2.

B. Problem Motivation

We are interested in learning representations from the spatiotemporal data in $\mathbf{X}$ for a particular task such as inferring

the source of a signal [40], forecasting future values [2], or classifying time series [3]. While the product graph is a viable mathematical tool to represent the structure in $\mathbf{X}$, we can also ignore it and process $\mathbf{X}$ with standard approaches such as the vector autoregressive (VAR) models [41], multilayer perceptrons (MLPs) [42], or recurrent neural networks (RNNs) [43]. However, such techniques face two main limitations. First, their number of parameters and computational complexity is of order $\mathcal{O}(N^2)$ as they involve dense layers, suffering the curse of dimensionality [12]. Second, they depend on the fixed order of the time series. The latter is a challenge because the learned representations do not permute with the time series and, when a new time series becomes available, the whole system needs to be retrained. Consequently, these models are not transferable.

One strategy to overcome such challenges is to exploit separately the spatial and the temporal structure by first processing $\mathbf{X}$ column-wise with a graph neural network (GNN) [44] and successively with a temporal convolution or RNN [19], [22]. However, such strategy fails to exploit the spatiotemporal coupling in the data and leverages only the coupling in the learned higher-level representation of the GNN, ultimately, limiting the performance and interpretability. One way to overcome this challenge is to embed a graph structure within a recursive model such as VAR [26] or RNN [27], [28]. Working with graph-based VAR models is limiting because they work in the linear space, while graph-based RNN works only with a fixed latent-space dimension of N (number of the nodes) or require an additional pooling step. Both the latter are uncomfortable because RNN-type of algorithms work best with a lower-dimensional latent space and finding the appropriate pooling strategy adds an extra challenge to the whole system [44]. Another aspect of RNNs is that they suffer training issues in their vanilla form; hence, LSTMs [45] or GRUs [46] are needed, which put more emphasis on the temporal dependencies rather than on the joint spatiotemporal ones.

Differently, we propose a first-principle neural network solution that exploits product graphs as an inductive bias for the spatiotemproal coupling, that is modular to any pooling technique, and that puts equal emphasis on the joint spatiotemporal learning. Building on first principles allows for a mathematical tractability akin to that of the principled spatial CNN [11]. To achieve these goals, we leverage the shift-and-sum principle of the convolution operation [47] to propagate information over the product graph but differently from a GCNN [11] we leverage the sparsity of the product graph to handle its large dimensionality. Working with the convolution principle allows also developing an architecture that is equivariant to permutations [11], enjoys a spectral analysis via the graph-time Fourier transform [36], [48] (Section III-B), and that is stable to perturbations in the spatial support (Section IV).

III. Graph-Time Convolutional Neural Networks

In this section, we define the graph-time convolutional neural network (GTCNN) and discuss its properties both in the vertex and in the spectral domain.

A. Graph-Time Convolutional Filters

The key element to build a GTCNN is the *graph-time convolutional filter*. For a product graph $\mathcal{G}_\diamond = (\mathcal{V}_\diamond, \mathcal{E}_\diamond, \mathbf{S}_\diamond)$ and signal $\mathbf{x}_\diamond$, the output $\mathbf{y}_\diamond$ of a graph-time convolutional filter of order K is

$$\mathbf{y}_\diamond = \mathbf{H}(\mathbf{S}_\diamond)\mathbf{x}_\diamond = \sum_{k=0}^{K} h_k \mathbf{S}_\diamond^k \mathbf{x}_\diamond \tag{2}$$

where $h_0, \ldots, h_K$ are the parameters and $\mathbf{H}(\mathbf{S}_\diamond) := \sum_{k=0}^{K} h_k \mathbf{S}_\diamond^k$ denotes the filtering matrix. The qualifier convolution for filter (2) comes from the fact that it shifts the input $\mathbf{x}_\diamond$ up to K times over the product graph $\mathbf{S}_\diamond^0\mathbf{x}_\diamond, \mathbf{S}_\diamond^1\mathbf{x}_\diamond, \ldots, \mathbf{S}_\diamond^K\mathbf{x}_\diamond$ and builds the output $\mathbf{y}_\diamond$ as a scaled sum of these shifts; see Fig. 3. Since $\mathbf{S}_\diamond$ is spatiotemporally local, the shift $\mathbf{S}_\diamond\mathbf{x}_\diamond$ diffuses the signal from the spatiotemporal node $i_\diamond$ to any other immediate spatiotemporal neighbor $j_\diamond$. From recursion $\mathbf{S}_\diamond^k\mathbf{x}_\diamond = \mathbf{S}_\diamond(\mathbf{S}_\diamond^{k-1}\mathbf{x}_\diamond)$, we can see that higher-order powers $\mathbf{S}_\diamond^k$ diffuse the signal to spatiotemporal neighbors that up to $k-$hops. This implies that filter (2) is spatiotemporally local in a neighborhood of radius K. The filter resolution depends on the order K but also on the type of product graph. For instance, the Kronecker product ignores the information present at time t limiting the spatial resolution but has a high temporal one; the Cartesian product brings in the spatial information of time t but accounts for the temporal proximity only for signal values at the same node; the strong product accounts for both and needs a lower order K to cover a particular neighborhood; Fig. 3. The locality of $\mathbf{S}_\diamond$ allows obtaining the output $\mathbf{y}_\diamond$ in (2) with a computational cost of order $\mathcal{O}(K|\mathcal{E}_\diamond|)$. The latter can be achieved by leveraging the recursion $\mathbf{S}_\diamond^k\mathbf{x}_\diamond = \mathbf{S}_\diamond(\mathbf{S}_\diamond^{k-1}\mathbf{x}_\diamond)$. This complexity order is appealing despite we work with a large GSO and input vector. However, it should be noted that $|\mathcal{E}_\diamond|$ is in turn governed by the type of product graph [cf. Fig. 2], thus making the strong product a suitable choice only if the spatial graph is highly sparse. It also follows from (2) that the order of parameters is $\mathcal{O}(K)$ and it is independent on the spatial and temporal dimensions.

B. Gtcnns

A graph-time convolutional neural network (GTCNN) is a composition of graph-time convolutional filters with pointwise nonlinearities. Specifically, consider an architecture composed of L layers $\ell = 1, \ldots, L$ and let $\mathbf{H}_\ell(\mathbf{S}_\diamond) = \sum_{k=0}^{K} h_{k\ell}\mathbf{S}_\diamond^k$ be the filter used at layer ℓ. The GTCNN propagation rule is

$$\mathbf{x}_{\diamond,\ell} = \sigma\left(\mathbf{H}_\ell(\mathbf{S}_\diamond)\mathbf{x}_{\diamond,\ell-1}\right) = \left(\sum_{k=0}^{K} h_{k\ell}\mathbf{S}_\diamond^k\mathbf{x}_{\diamond,\ell-1}\right) \tag{3}$$

where $\mathbf{x}_{\diamond,0} := \mathbf{x}_\diamond$ is the GTCNN input. Fig. 4 illustrates a GTCNN of three layers. To augment the representation power, we consider multiple nodal features per layer in matrix $\mathbf{X}_{\diamond,\ell-1} = [\mathbf{x}_{\diamond,\ell-1}^1, \ldots, \mathbf{x}_{\diamond,\ell-1}^F] \in \mathbb{R}^{NT\times F}$ where each column $\mathbf{x}_{\diamond,\ell-1}^g$ is a graph-time signal feature at layer $\ell-1$. These features are passed through a bank of graph-time convolutional filters and

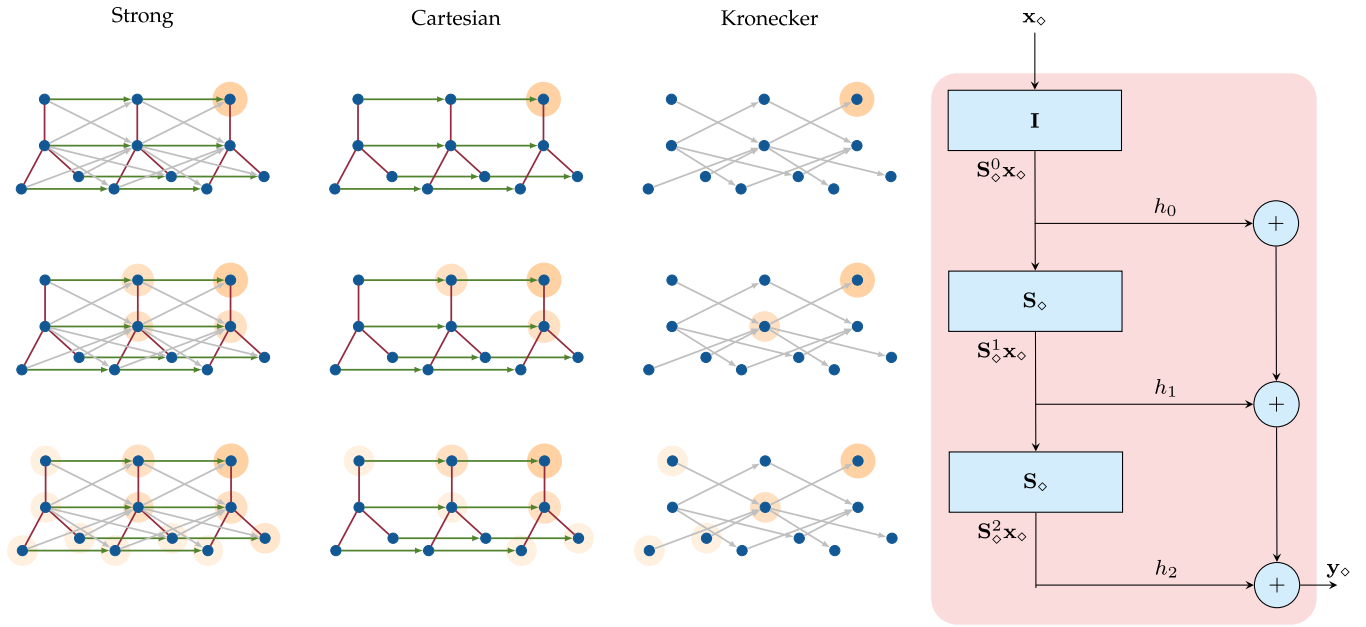


Fig. 3. Graph time convolutional filter of order two illustrated for a Cartesian product graph. Each block shifts the signal $\mathbf{x}_\diamond$ over the product graph by its graph shift operator $\mathbf{S}_\diamond$. Each shift implies a neighborhood coverage highlighted in yellow for a particular node. Each shifted signal $\mathbf{S}_\diamond^k\mathbf{x}_\diamond$ is scaled by its filter coefficient h_k and summed up to build the output $\mathbf{y}_\diamond$. Changing the product graph implies a different spatiotemporal neighborhood coverage.

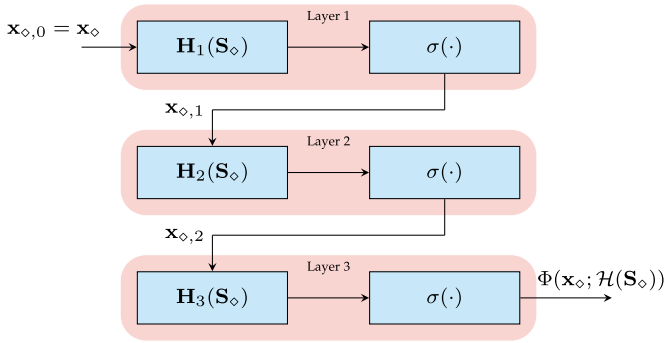


Fig. 4. GTCNN block diagram. On each layer a graph time convolutional filter [cf. (2)] is composed by a point-wise non-linearity, and all these layers are cascaded to generate the output.

pointwise nonlinearities to yield the output features of layer ℓ

$$\mathbf{X}_{\diamond,\ell}=\sigma\left(\sum_{k=0}^{K}\mathbf{S}_\diamond^k\mathbf{X}_{\diamond,\ell-1}\mathbf{H}_{k\ell}\right) \tag{4}$$

where $\mathbf{H}_{k\ell}\in\mathbb{R}^{F\times F}$ is the parameter matrix containing the coefficients of the filter bank at layer ℓ. To ease the analysis of the filer bank in (4), we make explicit the input-output relation between the fth output feature $\mathbf{x}_{\diamond,\ell}^f$ and the gth input feature $\mathbf{x}_{\diamond,\ell-1}^g$ as

$$\mathbf{x}_{\diamond,\ell}^f=\sigma\left(\sum_{g=1}^{F}\mathbf{H}_\ell^{fg}(\mathbf{S}_\diamond)\mathbf{x}_{\diamond,\ell-1}^g\right)=\sigma\left(\sum_{g=1}^{F}\sum_{k=0}^{K}h_{k\ell}^{fg}\mathbf{S}_\diamond^k\mathbf{x}_{\diamond,\ell-1}^g\right) \tag{5}$$

for all $f=1,\ldots,F$ and where $\mathbf{H}_\ell^{fg}(\mathbf{S}_\diamond)=\sum_{k=0}^{K}h_{k\ell}^{fg}\mathbf{S}_\diamond^k$ is one of the F^2 filters used in layer ℓ.

Recursions (4) (resp. (5)) are repeated for all layers. In the last layer $\ell=L$, we consider without loss of generality the number of output features is one, i.e., $\mathbf{x}_{\diamond,L}:=\mathbf{x}_{\diamond,L}^1$. This output is a function of the input $\mathbf{x}_\diamond$ and the collection of all filter banks $\mathbf{H}_\ell^{fg}(\mathbf{S}_\diamond)$ [cf. (2)]. Grouping all filters in the filter tensor $\mathcal{H}(\mathbf{S}_\diamond)=\{\mathbf{H}_\ell^{fg}(\mathbf{S}_\diamond)\}_{\ell fg}$, we can define the GTCNN output as the mapping

$$\mathbf{x}_{\diamond,L}:=\Phi(\mathbf{x}_\diamond;\mathcal{H}(\mathbf{S}_\diamond))\quad\text{with}\quad\mathcal{H}(\mathbf{S}_\diamond)=\{\mathbf{H}_\ell^{fg}(\mathbf{S}_\diamond)\}_{\ell fg}. \tag{6}$$

The GTCNN parameters are learned to minimize a loss $\mathcal{L}(\mathbf{x}_L,\mathbf{Y})$ computed over a training set of input-output pairs $\mathcal{T}=\{\mathbf{x}_\diamond,\mathbf{Y}\}$. As it follows from (2), the number of parameters in each filter is $K+1$. This gets scaled by the number of filters per layer F^2 and the number of layers L, yielding an order of $\mathcal{O}(KF^2L)$ parameters defining the GTCNN. Likewise, the computational complexity for running the filter bank for L layers is of order $\mathcal{O}(KF^2L|\mathcal{E}_\diamond|)$. Such orders are similar to those of conventional GCNNs working with time invariant signals. This is a consequence of exploiting the sparsity of product graphs and the spatiotemporal coupling via graph convolutional filters [cf. (2)]. In the next section, we shall discuss how to *learn* this spatiotemporal coupling.

Alternative Graph-Time Neural Networks. Once the product graph is built, we can employ any aggregation scheme to collect spatiotemporal information. For the sake of completness, we detail here the message passing neural network (MPNN) [8] and the graph attention network (GAT) [49]. *MPNNs* update a spatiotemporal node's latent representation $\mathbf{x}_{\diamond,\ell}$ by passing a

message from its neighborhood as

$$\mathbf{m}^{(i)}_{\diamond,\ell-1} = f(\mathbf{x}_{\diamond,\ell-1}[j] : (i,j) \in \mathcal{E}_\diamond) \tag{7}$$

where $\mathbf{m}^{(i)}_{\diamond,\ell-1}$ is the message vector for node i, $\mathbf{x}_{\diamond,\ell-1}[j]$ is the output of layer $\ell-1$ at node j, and $f(\cdot)$ is a differentiable and permutation equivariant function. Each node's latent vector can be updated based on inferred messages and its current feature vector as

$$\mathbf{x}_{\diamond,\ell} = g(\mathbf{x}_{\diamond,\ell-1}, \mathbf{M}_{\diamond,\ell-1}) \tag{8}$$

where $\mathbf{M}_{\diamond,\ell-1} = \{\mathbf{m}^{(i)}_{\diamond,\ell-1}\}$ collects all the messages for each node and $g(\cdot)$ is an arbitrary differentiable function (i.e., neural networks). Since we are considering the spatiotemporal product graph, the messages are including spatiotemporal information, and the type of product graph dictates which nodes are forwarding these messages.

GATs use an attention mechanism to update latent representation of each node based on its neighborhood as

$$\mathbf{x}_{\diamond,l+1}[i] = \sigma\left(\sum_{(i,j)\in\mathcal{E}_\diamond} \alpha_{ij}\mathbf{H}\mathbf{x}_{\diamond,\ell}[j]\right) \tag{9}$$

where α_{ij} indicates attention coefficients and $\mathbf{H} \in \mathbb{R}^{F\times F}$ is a linear transformation to map the features between layers. The type of product graph rules the attention coefficients α_{ij} and their calculation.

In both MPNN and GAT models, we can split the rule over space and time to have a different message passing or attention scheme. All in all, these represent different principles to learn spatiotemporal representation with product graphs. We continue with convolutional principle since it provides theoretical tools for their analysis and leave the MPNNs and GATs to interested readers.

C. Learning the Spatiotemporal Coupling

The GTCNN output in (6) is influenced by the type of product graph but it is unclear which form is most suitable for a specific problem. To avoid imposing a wrong inductive bias, we consider the parametric product graph GSO

$$\mathbf{S}_\diamond = \sum_{i=0}^{1}\sum_{j=0}^{1} s_{ij}\left(\mathbf{S}_T^i \otimes \mathbf{S}^j\right), \tag{10}$$

where by learning the four scalars $\{s_{ij}\}$ we learn the spatiotemporal coupling. The parametric product graph generalizes the three product graphs seen in Section II-A. For instance, setting $s_{11} = 1$ and the rest zero we get the Kronecker product. But allowing each s_{ij} to take any real value weigh accordingly the impact of the spatial and temporal graphs into the final product graph. Note also that $s_{00} \neq 0$ implies the presence of spatiotemporal self-loops. If all s_{ij}'s are non-zero, the parametric product graph has $|\mathcal{E}_\diamond| = |\mathcal{E}_\boxtimes| + NT$ edges, which are NT more edges than the strong product because of these self-loops.

Building a GTCNN with the graph-time convolutional filter (2) but with the GSO (10) matches the spatiotemporal coupling with the task at hand. However, treating $\{s_{ij}\}$ as learnable parameters is practical only if we fix the filter order to $K = 1$.[1] For higher orders $K \geq 2$ this implies pre-computing the powers $\mathbf{S}^k_\diamond$ of (10) and storing them in the memory as the computational complexity of each multiplication is of order $\mathcal{O}((NT)^3)$. Yet, storing the powers in the memory is still impractical for large spatial graphs. Another way to mitigate these issues is to rearrange the whole filter expression as shown next.

Proposition 1: Consider the spatial graph $\mathcal{G}$ with shift operator $\mathbf{S}$ and the temporal graph $\mathcal{G}_T$ with shift operator $\mathbf{S}_T$. The graph-time convolutional filter in (2) operating with the parametric product graph in (10) is a particular case of

$$\mathbf{H}(\mathbf{S},\mathbf{S}_T) = \sum_{k=0}^{\bar{K}}\sum_{l=0}^{\tilde{K}} h_{kl}(\mathbf{S}_T^l \otimes \mathbf{S}^k) \tag{11}$$

where $\bar{K}$ and $\tilde{K}$ are the orders over the spatial and temporal graphs respectively and $\{h_{kl}\}$ are the parameters.

Proof: See Appendix A, available online. □

Building a GTCNN with a filter tensor $\mathcal{H}(\mathbf{S},\mathbf{S}_T) = \{\mathbf{H}_\ell^{fg}(\mathbf{S},\mathbf{S}_T)\}_{\ell fg}$ yields an architecture that is more flexible than (6). This is because of:

1) Filter $\mathbf{H}(\mathbf{S},\mathbf{S}_T)$ [cf. (11)] has more freedom to control the spatial and temporal resolution in each layer through orders $\bar{K}$ and $\tilde{K}$. Differently, filter $\mathbf{H}(\mathbf{S}_\diamond)$ [cf. (2)] does not allow for such an independent control, in which order K influences the spatiotemporal resolution in a coupled manner. This allows exploiting more resolution in a particular domain.
2) Filter $\mathbf{H}(\mathbf{S},\mathbf{S}_T)$ learns how the spatiotemporal coupling influences the graph-time convolution through parameters $\{h_{kl}\}$. This happens for each layer and feature. This is also more powerful than learning $\{s_{ij}\}$ and $\{h_k\}$ disjointly as in $\mathbf{H}(\mathbf{S}_\diamond)$ because it matches the importance of the multihop resolutions with the spatiotemporal links.
3) Filter $\mathbf{H}(\mathbf{S},\mathbf{S}_T)$ enjoys the same computational cost of $\mathbf{H}(\mathbf{S}_\diamond)$. Computing the output $\mathbf{y}_\diamond = \mathbf{H}(\mathbf{S},\mathbf{S}_T)\mathbf{x}_\diamond$ requires computing all shifts $\mathbf{x}^{(k,l)}_\diamond = (\mathbf{S}_T^l \otimes \mathbf{S}^k)\mathbf{x}_\diamond$ for all $k \in [\bar{K}]$ and $l \in [\tilde{K}]$. These shifts can be written in a recursive manner as

$$\mathbf{x}^{(k,l)}_\diamond = (\mathbf{S}_T\mathbf{S}_T^{l-1} \otimes \mathbf{S}\mathbf{S}^{k-1})\mathbf{x}_\diamond. \tag{12}$$

Exploiting the properties of the Kronecker product[2], we can write the latter as

$$\begin{aligned}\mathbf{x}^{(k,l)}_\diamond &= (\mathbf{S}_T \otimes \mathbf{S})(\mathbf{S}_T^{l-1} \otimes \mathbf{S}^{k-1})\mathbf{x}_\diamond \\ &= (\mathbf{S}_T \otimes \mathbf{I}_N)(\mathbf{I}_T \otimes \mathbf{S})(\mathbf{S}_T^{l-1} \otimes \mathbf{S}^{k-1})\mathbf{x}_\diamond.\end{aligned} \tag{13}$$

Thus, we can compute $\mathbf{x}^{(k,l)}_\diamond$ recursively as

$$\begin{aligned}\mathbf{x}^{(k,l)}_\diamond &= (\mathbf{S}_T \otimes \mathbf{I}_N)\mathbf{x}^{(k,l-1)}_\diamond \\ &= (\mathbf{S}_T \otimes \mathbf{I}_N)(\mathbf{I}_T \otimes \mathbf{S})\mathbf{x}^{(k-1,l-1)}_\diamond\end{aligned} \tag{14}$$

where $\mathbf{x}^{(k,l-1)}_\diamond = (\mathbf{I}_T \otimes \mathbf{S})\mathbf{x}^{(k-1,l-1)}_\diamond$ is the spatially shifted signal and $\mathbf{x}^{(0,0)}_\diamond := \mathbf{x}_\diamond$. Recursion (14) implies

[1] This would be the spatiotemporal form of GCN [50].
[2] $(\mathbf{A}\otimes\mathbf{B})(\mathbf{C}\otimes\mathbf{D}) = \mathbf{AC}\otimes\mathbf{BD}$ [51].

that we can compute $\mathbf{x}_\diamond^{(k,l)}$ from $\mathbf{x}_\diamond^{(k-1,l-1)}$ with a cost of $\mathcal{O}(T|\mathcal{E}| + N|\mathcal{E}_T|)$. Since we need to perform the latter for all $k \in [\bar{K}]$ and $l \in [\tilde{K}]$, we have a computational cost of order $\mathcal{O}(\bar{K}T|\mathcal{E}| + \bar{K}N|\mathcal{E}_T|)$, which is linear in the product graph dimension. The GTCNN cost is $F^2\,L$ times the latter.

D. Properties

The convolution principle allows studying two fundamental properties: equivariances to permutations in the vertex domain and the filters' frequency response in the frequency domain. We shall discuss these properties for a GTCNN with the more general filter $\mathbf{H}(\mathbf{S}, \mathbf{S}_T)$ [c.f. (11)] but the results hold also for filter $\mathbf{H}(\mathbf{S}_\diamond)$ [c.f. (2)].

Permutation equivariance allows assigning an arbitrary ordering to the nodes. This is a desirable property since a graph does not change by the permutation of its nodes, hence the GTCNN output should stay unchanged up to a reordering. The GTCNN is permutation equivariant as shown by the following proposition.

Proposition 2: Let $\mathbf{x}_\diamond = \text{vec}(\mathbf{X})$ be the signal over the product graph $\mathcal{G}_\diamond$ with shift operator $\mathbf{S}_\diamond = \mathbf{S}_T \diamond \mathbf{S}$. Consider also the output of a graph-time filter $\mathbf{H}(\mathbf{S}_T, \mathbf{S})\mathbf{x}_\diamond$. Then, for a permutation matrix $\mathbf{P}$ belonging to the set

$$\mathcal{P} = \{\mathbf{P} = \{0,1\}^{N\times N} : \mathbf{P}\mathbf{1} = \mathbf{1}, \mathbf{P}^\top\mathbf{1} = \mathbf{1}\}$$

it holds that

$$\mathbf{P}^\top \text{vec}^{-1}(\Phi(\mathbf{x}_\diamond; \mathcal{H}(\mathbf{S}_T, \mathbf{S}))) = \Phi(\mathbf{x}_\diamond; \mathcal{H}(\mathbf{S}_T, \mathbf{P}^\top\mathbf{S}\mathbf{P}))\text{vec}(\mathbf{P}^\top\mathbf{X}). \tag{15}$$

Proof: See Appendix B, available online. □

That is, the GTCNN operating on a multivariate time-series over a spatial graph $\mathbf{S}$ has the equally permuted output of the GTCNN operating on permuted spatial graph $\mathbf{P}^\top\mathbf{S}\mathbf{P}$. This property allows the GTCNN to exploit the spatiotemporal data dependencies and symmetries encoded in the product graph to enhance learning [12].

Spectral analysis of filter (11) can help us to study their behaviors and what spectral features they learn for a specific task. It also provides a fundamental framework for further analyzing the robustness of the GTCNNs to perturbations as we shall detail in the next section. To do so, consider the eigendecomposition of the spatial GSO $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\mathsf{H}$ with eigenvevtors $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_N]$ and eigenvalues $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_N)$. Consider also the eigendecomposition of the temporal GSO $\mathbf{S}_T = \mathbf{V}_T\mathbf{\Lambda}_T\mathbf{V}_T^\mathsf{H}$ where $\mathbf{V}_T = [\mathbf{v}_{T,1}, \ldots, \mathbf{v}_{T,T}]$ is the matrix of temporal eigenvectors and $\mathbf{\Lambda}_T = \text{diag}(\lambda_{T,1}, \ldots, \lambda_{T,T})$ is that of temporal eigenvalues. Then, the GSO of the product graph $\mathbf{S}_\diamond$ can be eigendecomposed as

$$\mathbf{S}_\diamond = \mathbf{V}_\diamond\mathbf{\Lambda}_\diamond\mathbf{V}_\diamond^\mathsf{H} = (\mathbf{V}_T \otimes \mathbf{V})(\mathbf{\Lambda}_T \diamond \mathbf{\Lambda})(\mathbf{V}_T \otimes \mathbf{V})^\mathsf{H} \tag{16}$$

with eigenvectors are $\mathbf{V}_\diamond = \mathbf{V}_T \otimes \mathbf{V}$ and eigenvalues $\mathbf{\Lambda}_\diamond = \mathbf{\Lambda}_T \diamond \mathbf{\Lambda}$ dictated by the product graph. Then, we can define the graph-time Fourier transform of signal $\mathbf{x}_\diamond$ as $\tilde{\mathbf{x}}_\diamond = (\mathbf{V}_T \otimes \mathbf{V})^\mathsf{H}\mathbf{x}_\diamond$ [33], [36]. This transform represents the variation of the multivariate time-series over the product graph in terms of product graph eigenvectors. Using these concepts, the following proposition characterizes the spectral behavior of filter (11).

Proposition 3: Consider the eigendecomposition of the spatial GSO $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\mathsf{H}$ and the temporal GSO $\mathbf{S}_T = \mathbf{V}_T\mathbf{\Lambda}_T\mathbf{V}_T^\mathsf{H}$. Consider also the graph-time Fourier transform of the output signal $\tilde{\mathbf{y}}_\diamond = (\mathbf{V}_T \otimes \mathbf{V})^\mathsf{H}\mathbf{y}_\diamond$ and the input signal $\tilde{\mathbf{x}}_\diamond = (\mathbf{V}_T \otimes \mathbf{V})^\mathsf{H}\mathbf{x}_\diamond$. Then, the filtering operation (11) operates in the graph-time frequency domain as

$$\tilde{\mathbf{y}}_\diamond = h\left(\mathbf{\Lambda}_T, \mathbf{\Lambda}\right)\tilde{\mathbf{x}}_\diamond \tag{17}$$

with the filter frequency response matrix

$$h\left(\mathbf{\Lambda}_T, \mathbf{\Lambda}\right) = \sum_{k=0}^{\bar{K}}\sum_{l=0}^{\tilde{K}} h_{kl}(\mathbf{\Lambda}_T^l \otimes \mathbf{\Lambda}^k) \tag{18}$$

with diagonal entries $[h(\mathbf{\Lambda}_T, \mathbf{\Lambda})]_{kk} = h(\lambda_{Tt}, \lambda_i)$ and $k = N(t-1) + i$ for $i = 1, \ldots, N$ and $t = 1, \ldots, T$.

Proof: See Appendix C, available online. □

That is, the graph-time filtered version $\mathbf{y}_\diamond$ of $\mathbf{x}_\diamond$ with the filter $\mathbf{H}(\mathbf{S}_T, \mathbf{S})$ corresponds to an element-wise multiplication in the graph-time frequency representation, i.e., $\hat{\mathbf{y}}_{ti} = h([\mathbf{\Lambda}_T]_t, [\mathbf{\Lambda}]_i)\hat{\mathbf{x}}_{ti}$. This is a direct extension of the convolution theorem [47] to the spatiotemporal setting, ultimately, justifying the qualifier convolution for filter (11). It shows that by learning the parameters h_{kl} we not only learn spatiotemporal coupling in the vertex domain with desirable properties but also the frequency response of these filters to extract relevant spectral patterns.

IV. Stability Analysis

We now investigate the stability properties of GTCNNs w.r.t. perturbations on the spatial graph to characterize its learning capabilities. Analyzing stability is important as we often may not have access to the ground truth spatial graph. In some cases, the inferred graph may be imperfect and we have to train the GTCNN over a perturbed graph; in other cases, practical physical networks (e.g., water or power networks) slightly differ from the one we train the GTCNN, e.g., because of model mismatches. So, having a stable GTCNN is desirable to perform the task reliably and allow transference [15]. We analyze the stability w.r.t. the *relative perturbation* over the spatial graph

$$\hat{\mathbf{S}} = \mathbf{S} + (\mathbf{E}\mathbf{S} + \mathbf{S}\mathbf{E}) \tag{19}$$

where $\hat{\mathbf{S}}$ is the perturbed GSO and $\mathbf{E}$ is the perturbation matrix [16]. Such model suggests that the graph perturbation depends on its structure, i.e., a node with more connected edges is relatively more prone to more perturbation. To characterize the GTCNN stability, we consider graph-time convolutional filters with a frequency response that varies slightly in the high spatial frequencies (eigenvalues $\mathbf{\Lambda}$) and arbitrarily in the temporal one, see Fig. 5.

Definition 1: Given a graph-time filter with an analytic frequency response $h(\lambda_T, \lambda)$ [cf. (18)]. We say this filter is graph integral Lipschitz if there exists constant $C > 0$ such that for all

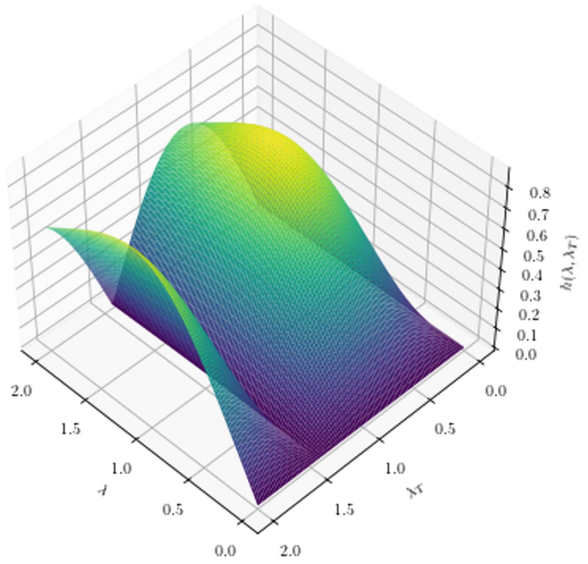


Fig. 5. Spatial Lipschitz graph-time filter. The frequency response varies smoothly over high graph frequencies, while it has sudden changes on temporal frequencies.

graph frequencies $\lambda_1, \lambda_2 \in \mathbf{\Lambda}$, it holds that

$$|h(\lambda_T, \lambda_2) - h(\lambda_T, \lambda_1)| \leq C \frac{|\lambda_2 - \lambda_1|}{|\lambda_2 + \lambda_1|/2}. \tag{20}$$

Expression (20) implies that the filter's two-dimensional frequency response is Lipschitz in any interval (λ_1, λ_2) of graph frequencies where the Lipschitz constant depends on their gap $|\lambda_2 + \lambda_1|/2$. The latter is similar to the integral Lipschitz property for GCNNs working on time invariant signals [16] since we treat spatial perturbations, but it does not restrict the temporal behavior of the filter.[3] For the two-dimensional frequency response, this implies that its partial derivative is restricted as

$$\left| \lambda \frac{\partial h(\lambda_T, \lambda)}{\partial \lambda} \right| \leq C, \tag{21}$$

i.e., the filters in a GTCNN have a frequency response that cannot vary drastically on high graph frequencies for all temporal frequencies but the filter can vary arbitrary over the temporal frequencies. Fig. 5 illustrates this property.

Definition 2: Consider a graph-time convolutional filter with an analytic frequency response $h(\lambda_T, \lambda)$ [cf. (18)]. We say this filter has a normalized spectral response if $|h(\lambda_T, \lambda)| \leq 1$ for all λ, λ_T.

This implies that the filter's gain $B = \|\mathbf{H}(\mathbf{S}_T, \mathbf{S})\|$ to be less or equal to 1, $B \leq 1$, w.r.t. an l_2-measure. Definition 2 is required as the output of each layer should not magnify the input norm, otherwise, the system will become less stable as the number of layers increase, even in absence of perturbation. Otherwise, constant B would also appear in the following stability result of the GTCNN.

Theorem 1: Let $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\mathsf{H}}$ and $\mathbf{S}_T\mathbf{V}_T\mathbf{\Lambda}_T\mathbf{V}_T^{\mathsf{H}}$ be spatial and temporal graph shift operators, respectively. Let also $\hat{\mathbf{S}}$ be the relatively perturbed graph shift operator [cf. (19)]. Consider the error matrix has the eigendecomposition $\mathbf{E} = \mathbf{U}\mathbf{M}\mathbf{U}^{\mathsf{H}}$ where $\mathbf{U}$ are the eigenvectors and $\mathbf{M}$ is the diagonal matrix of eigenvalues. Assume the error matrix has a bounded operator norm $\|\mathbf{E}\| \leq \epsilon$. Consider a GTCNN with L layers, F features and integral Lipschitz spatial-temporal graph filters [cf. Definition 1] with normalized spectral responses [cf. Definition 2]. Let also its nonlinearities be 1-Lipschitz, i.e., $|\sigma(a) - \sigma(b)| < |a - b|$, such as ReLU. Then, the distance between the GTCNN outputs $\Phi(\mathbf{x}_\diamond; \mathbf{S}_T, \mathbf{S}, \mathcal{H})$ on the nominal graph and $\Phi(\mathbf{x}_\diamond; \mathbf{S}_T, \hat{\mathbf{S}}, \mathcal{H})$ on the perturbed graph is upper bounded by

$$\|\Phi(\mathbf{x}_\diamond; \mathcal{H}(\mathbf{S}_T, \mathbf{S})) - \Phi(\mathbf{x}_\diamond; \mathcal{H}(\mathbf{S}_T, \hat{\mathbf{S}}))\|_2 \leq LF^{L-1}\Delta\epsilon\|\mathbf{x}_\diamond\|_2 \tag{22}$$

where $\Delta = 2C(1 + \delta T\sqrt{N})$, $\delta = (\|\mathbf{U} - \mathbf{V}\|^2 + 1)^2 - 1$ indicates the eigenvector misalignment between the spatial graph shift operator $\mathbf{S}$ and error matrix $\mathbf{E}$, N is the size of spatial graph, and T is the size of temporal graph.

Proof: See Appendix D, available online. □

Result (22) generalizes the findings in [16] to the spatiotemporal domain and states that GTCNNs are stable to relative perturbations in the spatial graph for integral Lipschitz graph-time convolutional filters in the graph frequency domain (modifying of [16] for product graphs enforces this assumption on both time and graph frequency domains). Together with the permutaiton equivariance [Proposition 2], results (22) shows that GTCNNs allow for inductive learning and that are transferable architectures. Such result provides three main insight on its stability/transferability:

1) The GTCNN is less stable for larger graphs ($\sqrt{N}$) as more nodes are exchanging information over a perturbed graph. Instead, its stability is more affected by the temporal resolution as increasing T implies replicating the entire perturbed spatial graph.
2) The GTCNN is less stable if it is more discriminative in the nominal graph. This is reflected by term LF^{L-1} and it is due to the larger number of filters operating on the perturbed graphs. Such a observation shows also an inherit trade-off between stability and discriminability of GTCNNs.
3) Finally, we see the impact of each individual filter via the Lipschitz constant C. The latter in turn controls the filter discriminability on high graph frequencies to improve stability.

Comparison With Alternative Bounds. We discuss now that exploring the product graph sparsity with GTCNN provides more stable solutions compared with baselines that ignore it. We discuss also the relation of result (22) with the stability result for spatiotemporal learning [28], [30].

Product graph GCNN: One trivial learning solution on product graphs is to ignore what different edges represent and naively deploy a GCNN over this large graph of NT nodes. Generalizing

[3] If we would ignore the product graph structure and discuss the stability results using [16] straightforwardly the filter needed be spatiotemporal integral Lipschitz. This in turn affects discriminability [15].

the results in [16] for the product graph with NT nodes yields a stability bound of $\Delta_{\text{GCNN}} = 2C\sqrt{T}(1+\delta\sqrt{NT})$ since it assumes the perturbation over all the edges in the product graph including temporal ones, so, the noise energy scales by $\sqrt{T}$ and it applies over NT nodes. The latter leads to a stability bound $2CT$ times looser than (22). Moreover, Theorem 1 restrains the graph-time filter variability only on spatial frequencies, while if we apply directly the results of [16], we restrain also high temporal frequency variations.

GCNNs: Another way to approach spatiotemporal learning is to treat the time series as features over the nodes and deploy a conventional GCNN. Leveraging again the result of [16], such a solution will have a stability bound $\Delta_{\text{GCNN}} = 2CT^L(1+\delta\sqrt{N})$ as we replicate filters in each layer T times for each feature, including the input layer. This bound is magnificently large due to factor T^L.

GGRNN [28]: Graph gated recurrent neural networks (GGRNNs) replace the linear transformations in a recurrent neural network with graph filters to learn spatiotemporal patterns. They have a stability bound $\Delta_{\text{GGRNN}} = C(1+\sqrt{N}\delta)(T^2+3T)$. Comparing with (22), Δ_{GGRNN} expands at a higher rate by a factor of T, but note that the term T^2+3T also implicitly contains the effect of layers L in itself. The assumptions behind the stability theorem in [28] are similar to Theorem 1, i.e., spatial integral Lipschitz filters and 1-Lipschitz nonlinearity. In conclusion, Theorem 1 and [28] present closely related stability bounds based similar conditions but for different models.

ST-GNN [30]: Space-time graph neural network (ST-GNN) linearly composes a GSO with a continuous time shift operator to define a space-time shift operator. The stability to relative perturbation on the spatial graph is equal to GCNN [11], i.e., $\Delta_{\text{STGNN}} = 2C(1+\delta\sqrt{N})$. This bound suggests that ST-GNN can process temporal properties of times series robustly under spatial perturbation. The result (22) reflects the effect of time series length on stability of GTCNN as it involves time directly into convolutional filters to capture spatiotemporal patterns in the data.

V. Numerical Results

This section evaluates the GTCNN performance in different scenarios and compares it with other state-of-the-art algorithms. In all experiments, the ADAM optimizer is used to train the model and an unweighted directed line graph is selected as the temporal graph. A complete hyperparameter selection is reported in Appendix E, available online.

A. Source Localization

The task is to detect the source of a diffusion process over a graph by observing the time-series of length T. The graph is an undirected stochastic block model with $C=5$ communities and $N=100$ nodes. The edges are randomly and independently drawn with probability 0.8 for nodes in a same community and 0.2 for nodes in different communities. At $t=0$, a random node takes a unitary value and diffuses it throughout the network 30 times as $e^{-t\mathbf{L}}$. The model is fed with a multivariate time series $\{\mathbf{x}_{t-T},\dots,\mathbf{x}_{t-1}\}$ randomly selected after at least $t=15$ diffusion and the goal is to detect the community corresponding to the source node. Our aim here is to study the role of the different product graphs; hence, we compare different GTCNNs with the baseline GCNN that ignores the temporal connections and treats time as feature.

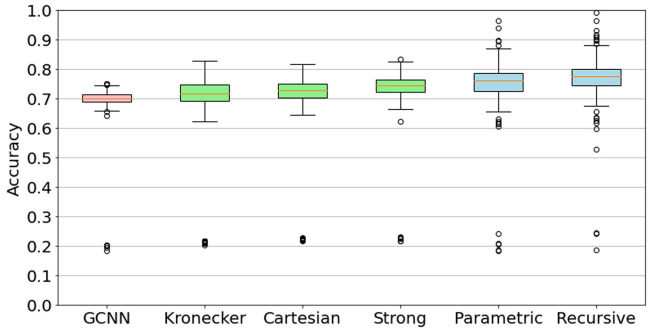


Fig. 6. Comparison of GCNN with parametric and non-parametric GTCNN performances on source localization task to emphasize on the role of spatiotemporal couplings.

The dataset contains 2000 samples with an $80/10/10$ split. All architectures have two layers with two second order filters. The cross-entropy loss is used for all models except for the parametric which its cost is also regularized by an l_1-norm of product graph parameters $\{s_{ij}\}$ with regularization weight $\beta=0.05$ to enforce sparse spatiotemporal connections. The temporal windows are selected from $T\in\{2,3,4,5\}$ and the features from $F_1,F_2\in\{4,8,16,32\}$. The models are trained for 1000 epochs with 100 batch size. Each experiment is done for 10 different random graphs and 10 different dataset realization.

The results in Fig. 6 suggest that accounting for the temporal connections even through a fixed product graph improves the performance. Considering consecutive times as features in the GCNN can be translated as a fully connected temporal graph, so, the improved performance by enforcing a reasonable structure to temporal samples was expected. Better results are achieved by using parametric product graph as it learns the temporal connections for the specific task. This flexibility reduces also the number of failed training attempts compared with the fixed product graphs and the GCNN.

B. Multivariate Time-Series Forecasting

We applied the GTCNN to address traffic and weather forecasting on four benchmark datasets. On the traffic forecasting task, we used parametric product graph while the recursive model is applied for the weather forecasting problem. The goal is to show that both solutions compare well with alternatives. The results are compared with baseline methods to provide insights into the GTCNN capabilities and limitations. For baselines, we considered:

- *ARIMA:* auto-regressive integrated moving average model using Kalman filter [52]. This model treats each time series individually.
- *G-VARMA:* graph vector auto-regressive moving average model [26]. This model works upon statistical assumptions on the data and takes the spatial graph into account.

TABLE I
PERFORMANCE COMPARISON OF GTCNN AND OTHER BASELINE MODELS FOR DIFFERENT PREDICTION HORIZONS. THE BEST PERFORMANCE IS SHOWN IN BOLD AND THE SECOND BEST IS UNDERLINED. THE STANDARD DEVIATION OF ALL MODELS ARE OF THE ORDER 10^{-3} AND ARE OMITTED TO AVOID AN OVERCROWDED TABLE

Data	Models	3-Steps			6-Steps			12-Steps		
		MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
METR-LA	ARIMA [53]	3.99	8.21	9.60%	5.15	10.45	12.70%	6.90	13.23	17.40%
	G-VARMA [26]	3.60	6.89	9.62%	4.05	7.84	11.22%	5.12	9.58	14.00%
	GP-VAR [26]	3.56	6.54	9.55%	3.98	7.56	11.02%	4.87	9.19	13.34%
	FC-LSTM [53]	3.44	6.30	9.60%	3.77	7.23	10.90%	4.37	8.69	13.20%
	Graph Wavenet [25]	2.69	**5.15**	6.90%	3.07	6.22	8.37%	**3.53**	7.37	**10.01%**
	GMAN [54]	2.94	5.89	7.51%	3.22	6.61	8.92%	3.68	7.49	10.25%
	STGCN [2]	2.88	5.74	7.62%	3.47	7.24	9.57%	4.59	9.40	12.70%
	GGRNN [28]	2.73	5.44	7.12%	3.31	6.63	8.97%	3.88	8.14	10.59%
	GTCNN (this work)	**2.68**	5.17	**6.85%**	**3.02**	**6.20**	**8.30%**	3.55	**7.35**	10.21%
PEMS-BAY	ARIMA [53]	1.62	3.30	3.50%	2.33	4.76	5.40%	3.38	6.50	8.30%
	G-VARMA [26]	1.88	3.96	4.28%	2.45	4.70	5.42%	3.01	5.83	7.10%
	GP-VAR [26]	1.74	3.22	3.45%	2.16	4.41	5.15%	2.48	5.04	6.18%
	FC-LSTM [53]	2.05	4.19	4.80%	2.20	4.55	5.20%	2.37	4.74	5.70%
	Graph Wavenet [25]	1.30	2.74	2.73%	1.63	3.70	3.67%	1.95	4.52	4.63%
	GMAN [54]	1.34	2.82	2.81%	**1.62**	3.72	**3.63%**	**1.86**	**4.32**	**4.31%**
	STGCN [2]	1.36	2.96	2.90%	1.81	4.27	4.17%	2.49	5.69	5.79%
	GGRNN [28]	1.33	2.81	2.83%	1.68	3.94	3.79%	2.34	5.14	5.21%
	GTCNN (this work)	**1.25**	**2.66**	**2.61%**	1.65	**3.68**	3.82%	2.27	4.99	5.11%

- *GP-VAR:* graph polynomial auto-regressive model [26]. It has fewer parameters than G-VARMA yet considers the spatial graph.
- *FC-LSTM* [52]: fully connected LSTM performing independently on time series, i.e., one LSTM per time series.
- *Graph WaveNet:* A hybrid convolutional model for time-series over graph [25].
- *GMAN:* A multi-attention graph-based network designed for traffic prediction [53]
- *STGCN:* spatial-temporal graph convolution network which uses graph convolution module alongside with 1D convolution [2].
- *GGRNN:* gated graph recurrent neural network which replace linear transforms in a RNN by graph convolutional filters [28].

Traffic forecasting. We considered the traffic network datasets METR-LA and PEMS-BAY. METR-LA contains four months of recorded traffic data over 207 nodes on the highways of Los Angeles County with 5 minutes resolution [52]. PEMS-BAY includes six months of traffic information over 325 nodes in Bay Area with similar resolution of METR-LA. We considered the same setting as in [54]. The shift operator is a directed adjacency matrix constructed by applying a Gaussian threshold kernel over the road network distance matrix. The goal is to predict the traffic load in time horizons $15-30-60$ minutes having the time series for last 30 minutes, i.e., $T=6$.

Both datasets are divided into an $80/10/10$ split chronologically. The GTCNN is fixed and contains two layers with third order filters and the parametric product graph. We evaluated the number of features in each layer from $F \in \{4, 8, 16\}$. The objective function is the regularized mean squared error (MSE) via the l_1-norm on the product graph parameters $\mathbf{s}$, i.e., $\mathcal{L} = \text{MSE}(\hat{\mathbf{x}}_{t+1}, \mathbf{x}_{t+1}) + \beta\|\mathbf{s}\|_1$. The regularization weight is chosen from $\beta \in \{0, 0.05, 0.1\}$. For the GGRNN, we evaluated features $F \in \{4, 8, 16\}$ and filter orders $K \in \{3, 4, 5\}$. For the other models, the parameters have been set similar to [25]. The evaluation metrics are the mean absolute error (MAE), the root mean squared error (RMSE), and the mean absolute percentage error (MAPE).

Table I compares the performance of GTCNN and other baseline models. The GTCNN outperforms the other models in a short horizon while Graph WaveNet and GMAN work better for longer horizons. The benefits in the short term are due to high order spatiotemporal aggregation in the GTCNN which allows capturing efficiently the spatiotemporal patterns in the data. On the longer term, the Graph WaveNet works better because it captures longer term patterns by increasing the receptive field of the model through dilated convolutions. Graph Wavenet can also be fed with a longer time series as it works only with the spatial graph. GMAN also captures long-term pattern in the data through its multi-attention mechanism among different time stamps. Differently, the GTCNN does not capture them and an efficient implementation remains an interesting extension.

Weather forecasting. We considered two benchmark datasets, Molene and NOAA. The Molene dataset contains 744 hourly temperature measurement across 32 stations in a region of France. The NOAA dataset contains 8579 hourly temperature measurement across 109 stations in the U.S.. The same setting

TABLE II
THE PERFORMANCE OF THE GTCNN COMPARED WITH BASELINE METHODS ON THE WEATHER FORECASTING TASK. THE BEST PERFORMANCE IS SHOWN IN BOLD AND THE SECOND BEST IS UNDERLINED. THE STANDARD DEVIATION OF ALL MODELS ARE OF THE ORDER 10^{-2} FOR THE MOLENE DATASET AND 10^{-3} FOR THE NOAA DATASET AND THEY ARE OMITTED TO AVOID AN OVERCROWDED TABLE

Data	Models	1-Steps			3-Steps			5-Steps		
		MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
Moelene	ARIMA [53]	2.26	4.74	5.24%	3.56	7.95	12.37%	6.15	13.59	17.51%
	G-VARMA [26]	**2.09**	**4.40**	**4.97%**	**3.45**	**7.69**	**12.07%**	**6.00**	**13.26**	**17.42%**
	GP-VAR [26]	2.13	4.47	5.01%	3.52	7.85	12.10%	6.05	13.38	17.46%
	LSTM	3.84	8.06	12.44%	4.72	10.52	15.54%	7.84	17.33	21.09%
	Graph Wavenet [25]	3.52	7.41	11.39%	4.49	10.02	15.11%	7.54	16.66	20.14%
	GMAN [54]	3.68	7.82	12.09%	4.82	11.04	14.91%	7.73	16.86	19.74%
	STGCN [2]	3.78	7.95	15.20%	4.56	10.18	9.57%	7.50	16.59	20.12%
	GGRNN [28]	3.17	6.67	15.20%	4.55	10.16	8.97%	7.35	16.26	19.89%
	GTCNN (this work)	3.55	7.47	11.45%	4.43	9.87	14.98%	6.54	14.46	18.53%
NOAA	ARIMA [53]	0.39	0.98	1.68%	1.60	3.86	4.21%	2.89	6.75	8.47%
	G-VARMA [26]	0.36	0.89	1.67%	1.39	3.35	3.56%	2.59	5.48	6.92%
	GP-VAR [26]	0.41	1.02	1.70%	1.40	3.37	3.56%	2.59	6.04	7.31%
	LSTM	0.29	0.75	1.58%	0.66	1.59	2.31%	**1.39**	**3.24**	**3.46%**
	Graph Wavenet [25]	0.33	0.83	1.65%	0.95	2.28	2.84%	1.42	3.31	3.51%
	GMAN [54]	0.39	0.90	1.68%	0.99	2.31	2.81%	1.54	3.68	3.73%
	STGCN [2]	0.35	0.87	1.65%	0.82	1.97	2.44%	1.68	3.91	3.92%
	GGRNN [28]	**0.26**	**0.64**	**1.54%**	0.84	2.02	2.47%	1.63	3.80	3.89%
	GTCNN (this work)	0.28	0.70	1.57%	**0.63**	**1.53**	**2.28%**	1.53	3.57	3.65%

as [26] is used in this experiment. Our aim is to make a prediction of the temperature for $1-3-5$ hours ahead having the time series for last 10 hours.

The model is fixed and consists of two recursive GTCNN [c.f. (11)] layers with temporal locality selected from $\tilde{K} \in \{3,4,5\}$ and spatial locality from $\bar{K} \in \{3,5,7\}$. For the neural network models, the number of features is chosen via grid search from $F \in \{4,8,16\}$. The GGRNN filter orders are varied among $K \in \{3,4,5\}$. For the statistical models we set the parameters similar to [26]. The number of LSTM hidden units are selected from $\{8,16,32,64\}$. The loss function is the MSE at 1-step prediction.

Table II indicates the model performance for different prediction horizons. In the Molene dataset, graph-based statistical methods outperform the rest as the dataset contains fewer samples and the time series have clear patterns in their temporal variation. Hence, leveraging a statistical assumption for the data compensates for the lack of samples and leads to a better performance. Among these methods, G-VARMA performs the best which indicates the importance of inducing the graph in the model. Due to the temporal connections, the GTCNN still performs better than the neural network counterparts. In the NOAA dataset, the abundance of training data allows the neural network models to learn complicated patterns and outperform statistical-based models. All the neural network alternatives perform closely, however, in higher forecasting horizons LSTM starts to take over the other variants while GTCNN performs better in short horizons. Overall, the GTCNN can be considered as a valid alternative for learning spatiotemporal representations in both cases where the training set is limited or large.

C. Stability Analysis

To investigate the stability of the GTCNN, we tested the trained models in the source localization and weather forecasting experiments under perturbed graphs with different signal to noise ratios (SNR)

$$\text{SNR} = 10 \log_{10} \frac{\|\mathbf{S}\|_F^2}{2\|\mathbf{E}\|_F^2}. \tag{23}$$

The noise energy is doubled as it appears twice in the relative perturbation model [c.f. (19)].

Fig. 7(a) shows the average classification accuracy for different amounts of perturbations. The GTCNN performs decently even in noisy scenarios around $5\,dB$. Fig. 7(b) represents the performance for different number of nodes in the graph. We trained the GTCNN over different graph sizes and evaluated the results using a perturbed graph to observe the effect of graph size on stability. The accuracy reduces steadily with the graph size as Theorem 1 suggests. To investigate GTCNN stability itself and its relation with bound (22), we analyzed the output embeddings of a fixed GTCNN network between the nominal and the perturbed graph. Fig. 7(c) and (d) depicts theoretical bound versus GTCNN empirical performance. We can observe that the bound reflects the same behavior of the empirical results with respect to both noise energy and time series length. In

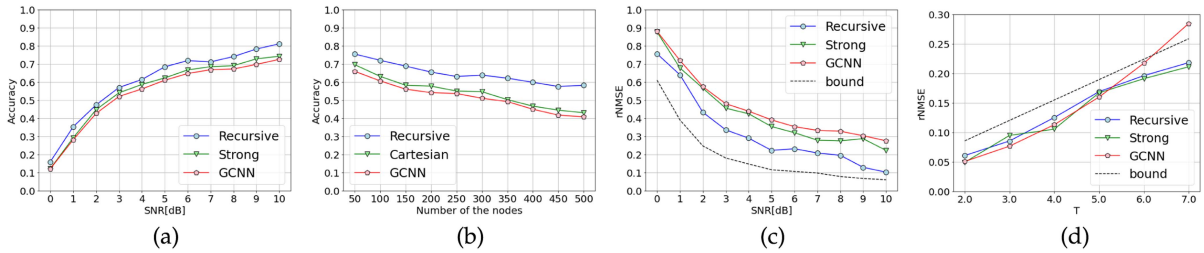


Fig. 7. Stability results for different scenarios of the parametric GTCNN and the best alternative for a fixed product graph (Kronecker, Cartesian, Strong). We also consider the vanilla GCNN as a baseline. (a) Results for different SNRs. (b) Performance for different graph sizes in 5 dB perturbation. (c) Embedding difference in terms of rNMSE between the trained GTCNN on the nominal graph and the perturbed one. (d) Embedding difference for different time series lengths in 5 dB perturbation.

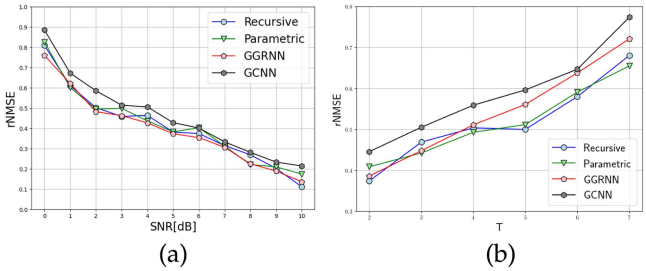


Fig. 8. Stability results for different scenarios of the parametric GTCNN, recursive GTCNN, baseline GCNN, and GGRNN on NOAA dataset. (a) Results for different SNRs. (b) Performance for different time series lengths in 5 dB perturbation.

TABLE III
ABLATION STUDY OF THE GTCNN. THE PERFORMANCE METRIC FOR SOURCE LOCALIZATION TASK IS ACCURACY, TRAFFIC PREDICTION IS RNMSE, AND WEATHER FORECASTING IS RNMSE. COMPUTATION IS THE AVERAGE TIME (SECOND) PER EPOCH FOR ALL THE EXPERIMENTS

Model	Souce Localization	METR-LA	NOAA	Time per epoch
Baseline GCNN	0.69	0.3984	0.2718	64.52
Fixed	0.74	0.3912	0.2702	69.39
Parametric	0.76	0.3624	0.2451	88.94
Shared parametric	0.76	0.3641	0.2369	58.11
Recurssive	0.78	0.3477	0.2119	63.97

addition, from all results we see the GTCNN offers a more stable performance compared with the vanilla GCNN by using the temporal data as features. This in turn highlights our theoretical insights after Theorem 1.

To compare the GTCNN stability with baseline models, we replicated the weather forecasting experiment on NOAA dataset and employed perturbed graphs with different SNRs in the testing phase. Fig. 8(a) illustrates the performance of GTCNN alongside with GGRNN and the baseline GCNN for different noise levels. All models show a similar empirical performance but most importantly we corroborate the findings of Theorem 1. The proposed models (recursive and parametric) become looser with the rate shown by result (22). Fig. 8(b) studies the effect of temporal window T by depicting the performance for fixed noise and different time series lengths. The GTCNN performs better than the rest and it also loses performance on a lower rate than alternatives as the time window increases. The latter also aligns with the theoretical analysis in Theorem 1.

Finally, in Fig. 9 we investigate the frequency response of the learned filters. The frequency responses vary smoothly over high graph frequencies for the trained filters while the temporal frequencies have more variations throughout of the spectrum. This also shows the integral Lipschitz property of said filters.

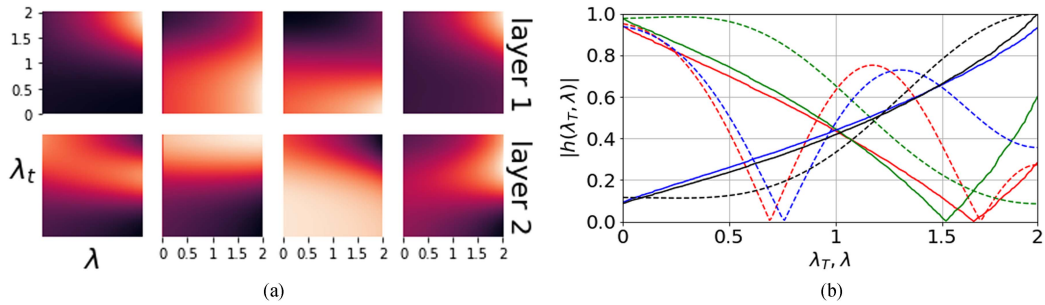
D. Ablation Study

To investigate the role of each component in the GTCNN and the difference between its variations, an ablation study is performed. The experiments include source localization on synthetic data, traffic prediction on METR-LA dataset, and weather forecasting on NOAA dataset. The performance metric for the first is accuracy and the others are measured with rNMSE. Notice that the design of the models are similar to the performed applications. The GTCNN variants studied include:

- *Baseline GCNN:* The vanilla GCNN where each time stamp is considered as a specific feature over the graph.
- *Fixed:* Best performance of a fixed product graph (Cartesian, Kronecker, Strong) used in the GTCNN.
- *Parametric:* Uses the parametric product graph in the GTCNN alongside a l_1-norm regularizer for parameters $\mathbf{s}$ with weight β.
- *Recursive:* The GTCNN structure using the recursive graph-time convolutional filter (11).
- *Shared parametric:* Recursive model where parameters h_{kl} are shared in a certain way yielding the parametric product graph model.

Table III shows the performance of different variations of the GTCNN. As the results suggest, using a product graph, even a fixed one, improves the model's performance and increases computational cost slightly. Instead, using a regularized parametric product graph further improves the performance in exchange of considerable model complexity. The recursive model not only reduces the computational cost of parametric model, but it also improves the performance slightly by providing more degrees of freedom. Shared parametric variation reduces the computational complexity, however, the performance drops near the parametric model as Proposition 1 claims.

Fig. 10 shows the performance of parametric model on weather forecasting application for different regularizer weights



(a) (b)

Fig. 9. (a) Normalized frequency response of a trained recursive GTCNN with two layers and four filters per layer. The bright color represents 1 while the dark color stands for 0. (b) Examples of the filter frequency response variation to normalized frequencies. continuous line belongs to graph frequencies λ and dashed ones are related to temporal frequencies λ_T.

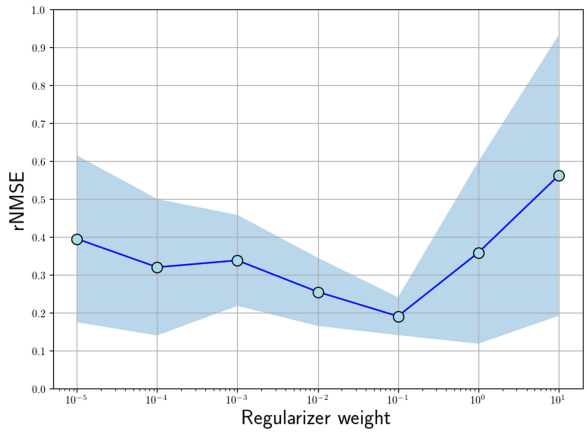


Fig. 10. The performance of parametric model for different values of regularization weight β on NOAA dataset.

β. The result shows that the parametric GTCNN is highly sensitive to the regularizer weight β as it varies considerably with respect to this parameter. The main reason is polynomial relation of product graph parameters $\mathbf{s}$ with the model output which magnifies the effect of regularizer weight. Hence, another added merit of recursive model is eliminating polynomial parameters and improving model's sensitivity.

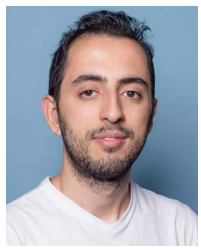
VI. Conclusion

We introduced graph-time convolutional neural networks as a model to learn from spatiotemporal data. The GTCNN uses product graph to convert dynamic data over network into static data over a larger graph. Afterward, a shift-and-sum convolution mechanism conveys the information over the product graph to exploit spatiotemporal dependencies in the data. The product graph itself can also be parametric to enable the model to learn temporal relationships directly from data, and also allows us to implement the GTCNN recursively and avoid storing and processing large graphs. We proposed a spectral domain analysis for graph-time convolutional filters and showed they operate as point-wise multiplication between the filters frequency response and the time series graph-time Fourier transform. Such a spectral analysis allowed us to study the stability of the GTCNN to perturbations in the spatial support. We showed that the GTCNN becomes linearly less stable as the time series length increases. However, it is yet more stable than the case where we consider time as distinct features and approach the problem via GCNNs. The numerical results also approved that GTCNNs performs better than a baseline GCNN model. Moreover, GTCNN compares well on benchmark datasets to state-of-the-art graph-based models but it suffered to capture long term patterns in the data. This is because the GTCNN works with large graphs and cannot maintain long input time series. The presented recursive model overcomes the memory consumption problem for large graphs and long time series while computational complexity needs to be improved yet as a direction for future works.

References

[1] E. Isufi and G. Mazzola, "Graph-time convolutional neural networks," in *Proc. IEEE Data Sci. Learn. Workshop*, 2021, pp. 1–6.
[2] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," 2017, *arXiv: 1709.04875*.
[3] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 1–9.
[4] M. Kadous et al., *Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series*. Princeton, NJ, USA: Citeseer, 2002.
[5] C. Zhang et al., "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 1409–1416.
[6] S. Wang et al., "Graph learning based recommender systems: A review," *arXiv:2105.06339*, 2021.
[7] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, May 2018.
[8] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," 2017, *arXiv: 1709.05584*.
[9] S. Wang, J. Cao, and P. Yu, "Deep learning for spatio-temporal data mining: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 8, pp. 3681–3700, Aug. 2022.
[10] P. W. Battaglia et al., "Relational inductive biases, deep learning, and graph networks," 2018, *arXiv: 1806.01261*.
[11] F. Gama, E. Isufi, G. Leus, and A. Ribeiro, "Graphs, convolutions, and neural networks: From graph filters to graph neural networks," *IEEE Signal Process. Mag.*, vol. 37, no. 6, pp. 128–138, Nov. 2020.
[12] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges," 2021, *arXiv:2104.13478*.
[13] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro, "Convolutional neural network architectures for signals supported on graphs," *IEEE Trans. Signal Process.*, vol. 67, no. 4, pp. 1034–1049, Feb. 2019.
[14] X. DefferrardBresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, D. Lee, M.U. Sugiyama, I. LuxburgGuyon, and R. Garnett Eds., Red Hook, NY, USA: Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper/2016/file/04df4d434d481c5bb723be1b6df1ee65-Paper.pdf

[15] L. Ruiz, F. Gama, and A. Ribeiro, "Graph neural networks: Architectures, stability, and transferability," *Proc. IEEE*, vol. 109, no. 5, pp. 660–682, May 2021.
[16] F. Gama, J. Bruna, and A. Ribeiro, "Stability properties of graph neural networks," *IEEE Trans. Signal Process.*, vol. 68, pp. 5680–5695, 2020.
[17] Z. Gao, E. Isufi, and A. Ribeiro, "Stability of graph convolutional neural networks to stochastic perturbations," *Signal Process.*, vol. 188, 2021, Art. no. 108216.
[18] R. Levie, W. Huang, L. Bucci, M. Bronstein, and G. Kutyniok, "Transferability of spectral graph convolutional neural networks," *J. Mach. Learn. Res.*, vol. 22, no. 272, pp. 1–59, 2021.
[19] D. Chai, L. Wang, and Q. Yang, "Bike flow prediction with multi-graph convolutional networks," in *Proc. 26th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2018, pp. 397–400.
[20] F. Manessi, A. Rozza, and M. Manzo, "Dynamic graph convolutional networks," *Pattern Recognit.*, vol. 97, 2020, Art. no. 107000.
[21] Y. Sun, Y. Wang, K. Fu, Z. Wang, C. Zhang, and J. Ye, "Constructing geographic and long-term temporal graph for traffic forecasting," 2020, *arXiv: 2004.10958*.
[22] M. Khodayar and J. Wang, "Spatio-temporal graph deep neural network for short-term wind speed forecasting," *IEEE Trans. Sustain. Energy*, vol. 10, no. 2, pp. 670–681, Apr. 2019.
[23] Y. Wang, P. Li, C. Bai, and J. Leskovec, "Tedic: Neural modeling of behavioral patterns in dynamic social interaction networks," in *Proc. Web Conf.*, New York, NY, USA: Association for Computing Machinery, 2021, pp. 693–705.
[24] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 922–929.
[25] X. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," 2019, *arXiv: 1906.00121*.
[26] E. Isufi, A. Loukas, N. Perraudin, and G. Leus, "Forecasting time series with VARMA recursions on graphs," *IEEE Trans. Signal Process.*, vol. 67, no. 18, pp. 4870–4885, Sep. 2019.
[27] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in *Proc. Int. Conf. Neural Inf. Process.*, 2018, pp. 362–373.
[28] L. Ruiz, F. Gamao, and A. Ribeiro, "Gated graph recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 68, pp. 6303–6318, 2020.
[29] C. Si, W. Chen, W. Wang, L. Wang, and T. Tan, "An attention enhanced graph convolutional LSTM network for skeleton-based action recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1227–1236.
[30] S. Hadou, C. I. Kanatsoulis, and A. Ribeiro, "Space-time graph neural networks," 2021, *arXiv:2110.02880*.
[31] A. Pareja et al., "EvolveGCN: Evolving graph convolutional networks for dynamic graphs," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 5363–5370.
[32] E. Hajiramezanali, A. Hasanzadeh, K. Narayanan, N. Duffield, M. Zhou, and X. Qian, "Variational graph recurrent neural networks," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, Eds., Red Hook, NY, USA: Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/a6b8deb7798e7532ade2a8934477d3ce-Paper.pdf
[33] 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 Process. Mag.*, vol. 31, no. 5, pp. 80–90, Sep. 2014.
[34] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: An approach to modeling networks," *J. Mach. Learn. Res.*, vol. 11, no. 2, pp. 985–10422010.
[35] W. Huang, A. G. Marques, and A. R. Ribeiro, "Rating prediction via graph signal processing," *IEEE Trans. Signal Process.*, vol. 66, no. 19, pp. 5066–5081, Oct. 2018.
[36] F. Grassi, A. Loukas, N. Perraudin, and B. Ricaud, "A time-vertex signal processing framework: Scalable processing and meaningful representations for time-series on graphs," *IEEE Trans. Signal Process.*, vol. 66, no. 3, pp. 817–829, Feb. 2018.
[37] C. Pan, S. Chen, and A. Ortega, "Spatio-temporal graph scattering transform," 2020, *arXiv: 2012.03363*.
[38] R. Levie, E. Isufi, and G. Kutyniok, "On the transferability of spectral graph filters," in *Proc. IEEE 13th Int. Conf. Sampling Theory Appl.*, 2019, pp. 1–5.
[39] H. Kenlay, D. Thano, and X. Dong, "On the stability of graph convolutional neural networks under edge rewiring," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2021, pp. 8513–8517.
[40] H. Zhao et al., "Multivariate time-series anomaly detection via graph attention network," in *Proc. IEEE Int. Conf. Data Mining*, 2020, pp. 841–850.
[41] H. Lütkepohl, *New Introduction to Multiple Time Series Analysis*. Berlin, Germany: Springer, 2005.
[42] J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Berlin, Germany: Springer, 2017.
[43] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.
[44] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.
[45] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[46] K. Cho et al., "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014, *arXiv:1406.1078*.
[47] A. V. Oppenheim, *Discrete-Time Signal Processing*. Noida, Uttar Pradesh, India: Pearson Education India, 1999.
[48] E. Isufi, G. Leus, and P. Banelli, "2-dimensional finite impulse response graph-temporal filters," in *Proc. IEEE Glob. Conf. Signal Inf. Process.*, 2016, pp. 405–409.
[49] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, *arXiv: 1710.10903*.
[50] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.
[51] K. B. Petersen et al., "The matrix cookbook," *Tech. Univ. Denmark*, vol. 7, no. 15, 2008, Art. no. 510.
[52] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," 2017, *arXiv: 1707.01926*.
[53] C. Zheng, X. Fan, C. Wang, and J. Qi, "GMAN: A graph multi-attention network for traffic prediction," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 1234–1241.
[54] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.



Elvin Isufi received the BSc and MSc degrees in electronic and telecommunication engineering from the University of Perugia, Italy, in 2012 and 2014, respectively, and the PhD degree in electrical engineering from the Delft University of Technology, The Netherlands, in 2019. He is an assistant professor with the Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, The Netherlands and co-director of AIdroLab -the TU Delft AI Lab on water networks. Prior to this, he was a postdoctoral researcher with the Department of Electrical and Systems Engineering, University of Pennsylvania. His research interests include intersection of signal processing, mathematical modeling, machine learning, and network theory, particularly in processing and learning from data on graphs and higher-order structures. He received the 2022 IEEE Signal Processing Society (SPS) Best PhD Dissertation Award and paper awards at the IEEE CAMSAP 2017, DSLW 2021, and DSLW 2022. Elvin Isufi is a member of the IEEE SPS Technical Committee on Signal Processing for Communication and Networking and serves as associate editor for Elsevier Signal Processing.

Mohammad Sabbaqi recieved the MSc degree in communication systems from the Sharif University, Tehran, Iran, in 2019. He is currently working toward the PhD degree with the Multimedia Computing Group, Delft University of Technlogy, Delft, The Netherlands. His research interests include graph neural networks, graph signal processing, and their applications on multivariate time series over networks.