Nonlinear State-Space Generalizations of Graph Convolutional Neural Networks

Ruiz, Luana; Gama, Fernando ; Ribeiro, Alejandro; Isufi, Elvin

DOI
10.1109/ICASSP39728.2021.9414672

Publication date
2021

Document Version
Accepted author manuscript

Published in
ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)

NONLINEAR STATE-SPACE GENERALIZATIONS OF GRAPH CONVOLUTIONAL NEURAL NETWORKS

Luana Ruiz[*], *Fernando Gama*[†], *Alejandro Ribeiro*[*] *and Elvin Isufi*[‡]

[*]Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, USA
[†]Electrical Engineering and Computer Sciences Department, University of California, Berkeley, USA
[‡]Department of Intelligent Systems, Delft University of Technology, Netherlands

ABSTRACT

Graph convolutional neural networks (GCNNs) learn compositional representations from network data by nesting linear graph convolutions into nonlinearities. In this work, we approach GCNNs from a state-space perspective revealing that the graph convolutional module is a minimalistic linear state-space model, in which the state update matrix is the graph shift operator. We show that this state update may be problematic because it is nonparametric, and depending on the graph spectrum it may explode or vanish. Therefore, the GCNN has to trade its degrees of freedom between extracting features from data and handling these instabilities. To improve such trade-off, we propose a novel family of nodal aggregation rules that aggregate node features within a layer in a nonlinear state-space parametric fashion allowing for a better trade-off. We develop two architectures within this family inspired by the recurrence with and without nodal gating mechanisms. The proposed solutions generalize the GCNN and provide an additional handle to control the state update and learn from the data. Numerical results on source localization and authorship attribution show the superiority of the nonlinear state-space generalization models over the baseline GCNN.

Index Terms— graph neural networks, state-space models, nonlinear systems, graph signal processing

1. INTRODUCTION

Graph convolutional neural networks (GCNNs) learn a parametric map from high-dimensional data whose dependencies can be represented by a graph, e.g., biological data, financial data, and social network data [1–3]. The GCNN map is a compositional layered function of simpler functions, where each layer is composed of a linear graph convolutional filter nested into a nonlinearity [4]. The graph serves as the prior about the data structure and restricts the space of functions to those exploiting this prior so to ease learning.

GCNNs have been developed from different yet equivalent viewpoints either in the graph spectral domain or in the vertex domain. The work in [5] leveraged spectral graph theory to convolve the data with a learnable filter as a pointwise multiplication in the Laplacian eigenspace. Subsequently, [6–9] built upon the shift-and-sum structure to perform convolutions directly in the vertex domain; an operation also known as finite impulse response (FIR) graph filtering [10, 11]. Changing the filter type to an autoregressive moving average (ARMA) form [12], the works in [13–15] implemented GCNNs with a rational spectral response in the convolutional layer. Differently, the work in [16] followed the attention idea [17] to aggregate nodal features, which, as it turns out, is also a FIR graph convolutional filter of order one on a graph whose edge weights are learned from data [15].

Since the graph filter is the tool that exploits the *graph-data coupling* within GCNNs, most of the contributions proposed filters operating with different graph representation matrices or with different implementations. While beneficial in specific applications, this strategy focuses only on linear nodal feature aggregation, and, as such, it overshadows the implicit state-space model present in graph convolutions. Unveiling and analyzing this state-space model can bring new insight into how graph convolutions operate and allows coming up with more general nodal aggregation schemes. In fact, state-space models have been fundamental in advancing Markov chains, Kalman filtering [18], and recurrent neural networks [19].

Inspired by the coupling between state-space models and sequential statistical learning, we put forth a similar interplay for graph convolutional filters. The state-space model considers the graph representation matrix (e.g., adjacency, Laplacian) as the state transition matrix while the intermediate nodal aggregations as system outputs (Section 2). We then show that the GCNN state-space convolutional module is rather limiting and propose appropriate generalizations towards a full-fledged non-linear state-space propagation rule (Section 3). Concretely, the contributions of this paper are twofold. First, it proposes a state-space analysis of graph convolutions, revealing that the GCNN is limited to linear nodal aggregations where the input signal of a specific layer vanishes/explodes with the filter order. Consequently, the filter coefficients have to mitigate such effect. Second, it develops a new family of graph neural networks (GNNs), which considers non-linear nodal aggregations within a layer and have intra-layer residual bridges to account for the layer input signal in the higher-order aggregations. By making parallelisms with nonlinear state-space models and with conventional RNNs, we further introduce a gating mechanism to increase the nonlinear filter order but still account for multi-resolution information in a data-driven manner. These contributions have been corroborated with numerical results in source localization and authorship attribution (Section 4). Conclusions are drawn in Section 5.

2. GRAPH CONVOLUTIONAL NEURAL NETWORKS

Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with node set $\mathcal{V} = \{1, \ldots, N\}$, edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, and shift operator matrix $\mathbf{S} \in \mathbb{R}^{N \times N}$ such that entry (i, j) satisfies $S_{ij} \neq 0$ if $(j, i) \in \mathcal{E}$ or $i = j$. Common choices for $\mathbf{S}$ include the graph adjacency matrix $\mathbf{A}$ or the graph Laplacian matrix $\mathbf{L}$. Along with the graph, we are interested in learning from signals $\mathbf{x} = [x_1, \ldots x_N]^\top$ residing on the vertices $\mathcal{V}$, in which entry x_i corresponds to the signal at node i. The GSO $\mathbf{S}$ plays a role into learning from this signal because the graph encodes pairwise relationships between signal components, which in turn serves as an inductive prior. In particular, if we consider a vector of coefficients $\mathbf{h} = [h_0, \ldots, h_K]^T$, we can use $\mathbf{S}$ to define graph convolutional filters as [11]

$$\mathbf{y} = \mathbf{h} *_{\mathbf{S}} \mathbf{x} = \sum_{k=0}^{K} h_k \mathbf{S}^k \mathbf{x} := \mathbf{H}(\mathbf{S})\mathbf{x} \tag{1}$$

where $\mathbf{y}$ is the filter output and $\mathbf{H}(\mathbf{S}) := \sum_{k=0}^{K} h_k \mathbf{S}^k$ is the filter matrix representation. The convolutional filter in (1) leverages the graph-data coupling locally. To see this, consider operation $\mathbf{w}_1 = \mathbf{S}\mathbf{x}$, which diffuses the input to neighboring vertices to produce another graph signal whose value w_{1i} at node i is a linear combination of signal values at the 1-hop neighbors. Likewise, operation $\mathbf{w}_k = \mathbf{S}^k\mathbf{x}$ shifts

Supported by USA NSF CCF 1717120, ARL DCIST CRA W911NF-17-2-0181.

the input k times to produce a graph signal whose value w_{ki} at node i is a linear combination of the input signal on neighbors that are at most k hops away. But since $\mathbf{w}_k$ can also be obtained from $\mathbf{w}_{k-1}$ as $\mathbf{w}_k = \mathbf{S}^k\mathbf{x} = \mathbf{S}\mathbf{w}_{k-1}$, it can be seen as the result of an aggregation from one-hop neighbor values of the former shifted signal $\mathbf{w}_{k-1}$.

Leveraging the graph convolutional filter in (1), we can define graph convolutional neural networks (GCNNs) as a compositional architecture of L convolutional filters and nonlinearities. At layer l, the GCNN takes as input a collection of F signal features $\{\mathbf{x}_{l-1}^g\}_{g=1}^F$ from the previous layer, processes them in parallel with a bank of F^2 graph convolutional filters $\{\mathbf{H}_l^{fg}\}_{f=1}^F$, and passes these outputs through a nonlinearity to obtain the propagation rule

$$\mathbf{x}_l^f = \sigma\Big(\sum_{g=1}^{F}\mathbf{H}_l^{fg}(\mathbf{S})\mathbf{x}_{l-1}^g\Big) = \sigma\Big(\sum_{g=1}^{F}\sum_{k=0}^{K}h_{lk}^{fg}\mathbf{S}^k\mathbf{x}_{l-1}^g\Big) \tag{2}$$

for $f = 1, \ldots, F$. The F outputs of layer l, $\{\mathbf{x}_l^f\}_f$, are inputs to the subsequent layer $l+1$, and this process repeats itself until the last layer, $l = L$, is reached. If we consider for simplicity only one input feature $\mathbf{x}_0 := \mathbf{x} \in \mathbb{R}^N$ and one output feature $\mathbf{x}_L := \mathbf{x}_L^1 \in \mathbb{R}^N$, this GCNN can be written succinctly as the map $\mathbf{x}_L = \mathbf{\Phi}(\mathbf{S}; \mathbf{x}; \mathcal{H})$. This notation emphasizes the dependence of the parametrization on the GSO $\mathbf{S}$ and on the filter coefficients $\mathcal{H} = \{\mathbf{h}_l^{fg}\}_{fgl}$ for all layers l and filter pairs f, g. Graph convolutional neural networks exhibit several desirable properties. Namely, they are local and distributed information processing architectures, making them perfectly suited for distributed learning [20, 21]. They are also permutation equivariant [22, 23] and stable to changes in the underlying graph support [22]. Finally, they have isomorphic properties [9, 24, 25] and are found to be more discriminable than graph filters [26].

2.1. The State-Space Model of Graph Convolutions

A discrete linear system with inputs $\mathbf{u}_k \in \mathbb{R}^N$ and outputs $\mathbf{y}_k \in \mathbb{R}^N$ can be expressed through its *state-space representation*

$$\begin{aligned}\mathbf{w}_k &= \mathbf{A}\mathbf{w}_{k-1} + \mathbf{B}\mathbf{u}_k \\ \mathbf{y}_k &= \mathbf{C}\mathbf{w}_k + \mathbf{D}\mathbf{u}_k\end{aligned} \qquad k = 1, \ldots, K \tag{3}$$

where $\mathbf{w}_k \in \mathbb{R}^N$ is the system state and $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D} \in \mathbb{R}^{N\times N}$ are the state-to-state, input-to-state, state-to-output, and input-to-output transition matrices, respectively. Comparing the recursive implementation of (1) with (3), we see that the convolutional module of the GCNN layer can be represented as a discrete linear system where the steps k correspond to graph shifts. Explicitly, the filter output $\mathbf{y} = \mathbf{H}(\mathbf{S})\mathbf{x}$ can be formulated as

$$\begin{aligned}\mathbf{w}_k &= \mathbf{S}\mathbf{w}_{k-1} \\ \mathbf{y}_k &= h_k\mathbf{w}_k\end{aligned} \qquad k = 1, \ldots, K \tag{4a}$$

$$\mathbf{y} = \sum_{k=0}^{K}\mathbf{y}_k \tag{4b}$$

where $\mathbf{u}_k = \mathbf{0}$, $\mathbf{A} = \mathbf{S}$ and $\mathbf{C} = h_k\mathbf{I}$. The state is initialized as $\mathbf{w}_0 = \mathbf{x}$, and the instantaneous output as $\mathbf{y}_0 = h_0\mathbf{w}_0$. The overall filter output $\mathbf{y}$ is calculated as the sum of the $K+1$ instantaneous outputs $\{\mathbf{y}_k\}_k$. Equation (4) makes for an interesting parallel between linear systems and graph convolutions. At the same time, it shows that the linear components of the layers of a GCNN are rather simple dynamical systems. While this is not necessarily a disadvantage, it reveals the opportunity of increasing the expressive power of GCNNs by modifying the linear system in (4).

Moreover, we can see that if $\mathbf{S}$ has eigenvalues greater than one in magnitude, the instantaneous state $\mathbf{w}_k$ explodes. This implies that the instantaneous outputs $\mathbf{y}_k$ will see little of the input signal for larger k. On the one hand, this will force the network coefficients $\mathbf{h}$ to learn convolutional representations of the input, and, on the other, to mitigate the explosive states for larger order states. Likewise, a similar trade-off is present if $\mathbf{S}$ has eigenvalues with magnitude less than one. In that case, we have to face with vanishing states, meaning that higher-order shifts from multi-hop neighbors will play a small role in the final output. While normalization would prevent large powers of the graph shift operator from exploding, it does not stop them from vanishing when the graph signal is aligned with eigenvectors associated with eigenvalues of magnitude less than one (i.e., any eigenvalue that is not the largest in absolute value). These trade-offs implicitly limit the degrees of freedom of the GCNN. Consequently, the model can only partially capture the coupling between the signal and the topology. This translates into limited expressive power. In the sequel, our goal is to generalize the convolutional state-space model (4) to architectures closer to a full-flagged non-linear state-space representation that still captures the coupling between the signal and the underlying topology.

3. NONLINEAR STATE-SPACE EXTENSIONS OF GCNNS

In this paper, we work towards extending the graph convolutional neural network layer to a propagation rule that, within a layer, can be represented as the N-state discrete *nonlinear* system

$$\begin{aligned}\mathbf{w}_k &= \sigma_w\big(\mathbf{A}\mathbf{w}_{k-1} + \mathbf{B}\mathbf{x}_k\big) \\ \mathbf{y}_k &= \sigma_y\big(\mathbf{C}\mathbf{w}_k + \mathbf{D}\mathbf{x}_k\big)\end{aligned}. \tag{5}$$

Contrasting (5) with the state space GCNN model (4), we note three key differences.

First, system (4) is linear in all of its components. The GCNN only applies the nonlinearity to the filter output $\mathbf{y}$, but not to the shifted signals $\mathbf{w}_k$ nor to the instantaneous outputs $\mathbf{y}_k$. Thus, graph convolutions limit nodal feature aggregations to the linear space.

Second, in (4) both the state $\mathbf{w}_k$ and the instantaneous output $\mathbf{y}_k$ are disconnected from the input $\mathbf{x}$. In fact, the input is only considered when initializing the state as $\mathbf{w}_0 = \mathbf{x}$. Therefore, its contribution to high-order shifts $\mathbf{w}_k$ and instantaneous outputs $\mathbf{y}_k$ is small and affected by the shift operator spectra. Additionally, nodes only learn weights to scale the influence of the values of the shifted signals $\mathbf{S}^k\mathbf{x}$ in their immediate neighborhood but leave any direct relationship with the input signal components of their k-hop neighbors unexploited.

Third, while in (4) the state $\mathbf{w}_{k-1}$ is diffused through the graph to produce the next state $\mathbf{w}_k$, there is no parametric relationship between state updates; i.e., $\mathbf{w}_{k-1}$ and $\mathbf{w}_k$. In turn, this leads to the aforementioned instabilities related to the state-transition matrix $\mathbf{S}$. Making $\mathbf{w}_k$ a graph parametric update of $\mathbf{w}_{k-1}$ improves our control over the stability of the state-transition matrix as a whole.

In the GNN architectures we develop next, the nodal aggregation schemes emulate a nonlinear state-space model [cf. (5)] that accounts for the graph structure in a similar fashion to graph convolutions [cf. (4)]. Approaching GNNs from this state-space perspective allows changing the family of propagation rules, which are generalized from the linear form in (2) to nonlinear node updates. As we will illustrate with the numerical experiments in Section 4, these modifications significantly improve GNN performance in a variety of application scenarios.

3.1. RSNs: Recursive Shift Networks

In the so-called *recursive shift networks* (RSNs), we enhance the capacity of the filter (4) by making both the state $\mathbf{w}_k$ and the instantaneous output $\mathbf{y}_k$ nonlinear on the state $\mathbf{w}_{k-1}$ and input $\mathbf{x}$. Explicitly, the nonlinear state-space model of a RSN has the form

$$\begin{aligned}\mathbf{w}_k &= \sigma_w\Big(h_{kww}\mathbf{S}\mathbf{w}_{k-1} + h_{kwx}\mathbf{x}\Big) \\ \mathbf{y}_k &= \sigma_y\Big(h_{kyw}\mathbf{w}_k + h_{kyx}\mathbf{x}\Big) \qquad k = 1, \ldots, K\end{aligned} \tag{6a}$$

$$\mathbf{y} = \sigma_y\Big(\sum_{k=0}^{K}\mathbf{y}_k\Big) \tag{6b}$$

where h_{kww}, h_{kwx} are scalar weights encoding the dependency of state $\mathbf{w}_k$ on state $\mathbf{w}_{k-1}$ and the input $\mathbf{x}$, respectively, while h_{kyw}, h_{kyx} are scalar weights encoding the dependency of the instantaneous output $\mathbf{y}_k$ on the state $\mathbf{w}_k$ and the input $\mathbf{x}$, respectively. On the one hand, (6) retains the simplicity and efficiency of the convolutional graph filter (4); on the other, it improves the expressive power of the graph convolution by including nonlinearities. These additional parameters as well as the nonlinearities endow the RSN with minimal degrees of freedom that are enough to control the explosion/vanishing of the state $\mathbf{w}_k$ with a better trade-off.

Despite looking similar to the conventional recurrent neural network (RNN) propagation rule, RSNs and RNNs are very different. RNNs have $N \times N$ parameter matrices, whereas in (6) the parameters of the nonlinear graph filters are independent of the graph dimensions. The nonlinear graph filters we consider share parameters across nodes—not shifts. This property is inherited from the graph convolutional filter [cf. (4)], in which the parameters h_k are distinct for different $\mathbf{y}_k$. These differences notwithstanding, we leverage the analogy with RNNs to consider gating mechanisms in Section 3.2.

3.2. LSSMs: Long Short Shift Memories

In both (4) and (6), the filter order K controls the information locality in the vertex and spectral domains. In the vertex domain, the order implies that state $[\mathbf{w}_K]_i$ at node i receives information from nodes up to K hops away; i.e., it defines a "local window" around the nodes. In the spectral domain, it controls the sharpness of the filter frequency response [10, 11]. When the information at a particular layer is localized around a few graph frequencies (eigenvalues of $\mathbf{S}$), higher filter orders are needed; i.e., the filter order imposes a "local window" around the graph frequencies. This is in agreement with the uncertainty principle of signal localization [27–29], which states that low values of K correspond to localized windows in the vertex domain, but not in the spectral domain (and vice-versa).

Increasing the filter order is thus necessary to capture more information in the vertex domain while retaining localized responses in the spectral domain. However, large K usually leads to numerical instabilities associated with large powers $\mathbf{S}^K$ and, depending on the eigenvalues of $\mathbf{S}$, we also have to cope with vanishing or exploding gradients. These challenges are similar to those encountered in RNNs. There, they are typically addressed by gating mechanisms that introduce an additional set of parameters to control how the information propagates in different state updates [30]. Here, we will use gates *within* the GNN layer to capture long-range dependencies over the graph because of the high order of (6).

In analogy with long-short term memories (LSTMs), we call our architecture *long-short shift memory* (LSSM). LSSMs comprise learnable gating parameters taking values in the interval $[0, 1]$. These parameters control the information passed to state $\mathbf{w}_k$ and instantaneous output $\mathbf{y}_k$ in (6). An LSSM filter comprises:

- Updating the $N \times 1$ internal memory variable $\tilde{\mathbf{c}}[k]$ as

$$\tilde{\mathbf{c}}_k = \tanh\big(h_{kcw}\mathbf{S}\mathbf{w}_{k-1} + h_{kcx}\mathbf{x}\big) \tag{7}$$

  to track the state update.

- Updating the $N \times 1$ forget gate $\boldsymbol{\gamma}_{\mathrm{f}}[k]$, update gate $\boldsymbol{\gamma}_{\mathrm{u}}[k]$, and state gate $\boldsymbol{\gamma}_{\mathrm{w}}[k]$ respectively as

$$\boldsymbol{\gamma}_{\mathrm{fk}} = \mathrm{sigmoid}\big(h_{kfw}\mathbf{S}\mathbf{w}_{k-1} + h_{kfx}\mathbf{x}\big) \tag{8a}$$

$$\boldsymbol{\gamma}_{\mathrm{u}k} = \mathrm{sigmoid}\big(h_{kuw}\mathbf{S}\mathbf{w}_{k-1} + h_{kux}\mathbf{x}\big) \tag{8b}$$

$$\boldsymbol{\gamma}_{\mathrm{w}k} = \mathrm{sigmoid}\big(h_{kww}\mathbf{S}\mathbf{w}_{k-1} + h_{kwx}\mathbf{x}\big) \tag{8c}$$

  which are internal variables that track the system evolution with their own set of parameters. The sigmoid nonlinearity ensures that the values are in the interval $[0, 1]$.

- Updating the $N \times 1$ global memory variable $\mathbf{c}_k$ and state $\mathbf{w}_k$

$$\mathbf{c}_k = \boldsymbol{\gamma}_{\mathrm{f}k} \circ \mathbf{c}_{k-1} + \boldsymbol{\gamma}_{\mathrm{u}k} \circ \tilde{\mathbf{c}}_k \tag{9a}$$

$$\mathbf{w}_k = \boldsymbol{\gamma}_{\mathrm{w}k} \circ \tanh(\mathbf{c}_k) \tag{9b}$$

  where " $\circ$ " denotes the element-wise Hadamard product. The forget gate $\boldsymbol{\gamma}_{\mathrm{f}k}$ and update gate $\boldsymbol{\gamma}_{\mathrm{u}k}$ control which entries of the former global memory $\mathbf{c}_{k-1}$ to propagate and which entries to update through the internal memory $\tilde{\mathbf{c}}_k$ [cf. (7)]. The global memory variable is then used to update the state $\mathbf{w}_k$, whose value is in turn controlled by the state update gate $\boldsymbol{\gamma}_{\mathrm{w}k}$.

- Setting the instantaneous output $\mathbf{y}_k$ to

$$\mathbf{y}_k = \sigma_y\bigg(h_{kyw}\mathbf{w}_k + h_{kyx}\mathbf{x}\bigg). \tag{10}$$

- Setting the overall LSSM output to

$$\mathbf{y} = \sigma_y\big(\sum_{k=0}^{K}\mathbf{y}_k\big). \tag{11}$$

In summary, the LSSM filter is defined by steps (7)–(11). Note that this *nonlinear graph filter* updates the state $\mathbf{w}_k$ as a nonlinear, shifted version of the former state while prioritizing information coming from certain nodes and, thus, only learning state updates on nodes that are relevant for the task at hand. The update is controlled by the gating mechanisms [cf. (8)], which are graph-based state-space models themselves. The additional parameters further increase the LSSM degrees of freedom compared with RSNs, allowing to control the state updates and also to learn where a higher vertex-spectra locality is needed. Substituting $\mathbf{H}_l^{fg}(\mathbf{S})$ for the LSSM filter in (2) leads to an LSSM-GNN layer update rule. Because of gating, the LSSM-GNN can be parametrized with higher values of K. This *longer memory* over the graph endows the LSSM-GNN with a better accuracy-robustness trade-off than the GCNN and the RSN.

Remark 1. Regarding the number of parameters of the proposed state-space extensions, they indeed have more parameters per layer than the GCNN, but the difference is not expressive — the RSN has $4KF^2$ [cf. equation (6)] and the LSSM $10(K+1)F^2$ parameters per layer [cf. equations (7)–(11)], versus KF^2 for the GCNN.

4. NUMERICAL EXPERIMENTS

In the following, we describe the scenarios and respective experimental setups used to corroborate the proposed solutions. The baseline setups are those of [15], which compares the GCNN with different state-of-the-art approaches. The models we evaluate are: (i) the conventional GCNN with linear filters [cf. (4)]; (ii) the RSN [cf. (6)]; (iii) the LSSM [cf. (7)-(11)]. All models have ReLU nonlinearities between layers and are trained using the ADAM optimizer with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ [31].

Source localization. The goal of this experiment is to identify the source community of a signal diffused over the graph given a snapshot of the signal at an arbitrary time step. The graph is an undirected stochastic block model (SBM) with $N = 50$ nodes divided into $C = 5$ blocks, each representing a community $\{c_1, \ldots, c_5\}$. The intra-community probability is $p = 0.8$ and the inter-community probability is $q = 0.2$. The source signal $\mathbf{x}[0]$ is a Kronecker delta centered at one source node and diffused at time $t \in [0, 50]$ as $\mathbf{x}[t] = \mathbf{S}^t\mathbf{x}[0]$, where $\mathbf{S}$ is the graph adjacency matrix normalized by the maximum eigenvalue. The training set comprises $10,240$ tuples of the form $(\mathbf{x}[i], c_i)$ for a random t and $i \in \{1, 2, 3, 4, 5\}$. The validation and the test sets are both composed of $2,560$ tuples. The models are trained with batch sizes of 100 samples for 40 epochs and a learning rate 10^{-3}, which is tuned for the GNN but not for the proposed models. The performances are averaged over ten different graph realizations and ten data splits, for a total of 100 realizations.

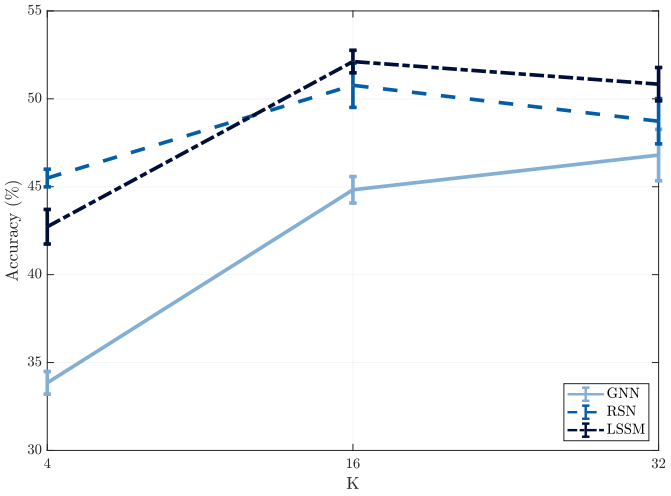

Fig. 1. Source localization accuracy for $K = 4$, 16 and 32. Error bars have been scaled by 0.5.

Table 1: Source localization accuracy for multiple layers (%).

	$L = 2$	$L = 4$
GCNN	49 ± 5	40 ± 17
RSN	62 ± 5	53 ± 12
LSSM	$\mathbf{62 \pm 3}$	$\mathbf{60 \pm 16}$

We vary the filter order K in the set $\{4, 16, 32\}$ to compare the accuracy-robustness trade-off of the GCNN, RSN, and LSSM in the source localization scenario. All architectures have $L = 1$ layer and $F = 4$ features. The nonlinearities σ_w and σ_y are the ReLU in (6); and σ_y is the ReLU in (10) and (11). At the output of each architecture, a readout layer maps the output signal to a one-hot vector of dimension C, which is then fed to a softmax.

The average test accuracies are shown in Figure 1. Both the RSN and the LSSM outperform the GCNN by a significant margin for all values of K. While the RSN achieves the best accuracy when K is fixed at $K = 4$, the LSSM exhibits a better performance for larger values of K, which indicates its better robustness-accuracy trade-off for high-order filters. This is further validated by the fact that they present the smallest standard deviation for $K = 16$ and $K = 32$.

In a second experiment, we aimed at verifying if increasing the number of layers of the GCNN (which results in adding more nonlinearities to the architecture) would yield similar gains in performance to those observed for the RSN and LSSM in Figure 1. We set $K = 4$ and $F = 4$ and train all architectures for $L = 2$ and $L = 4$. Average results are shown in Table 1 for 5 graph realizations and 5 dataset realizations each. Once again, we observe that the LSSM achieves the best accuracy and that both state-space extensions outperform the GCNN.

Authorship attribution. In authorship attribution, the learning task is to decide whether a $1,000$-word text excerpt has been authored by a specific author or by any of the other 20 contemporary authors in the author pool, given their word adjacency network (WAN) [32]. WANs are author-specific directed graphs whose nodes are function words without semantic meaning (e.g., prepositions, pronouns, conjunctions) and whose directed edges capture the transition probabilities between pairs of function words. An example of WAN is shown in Figure 2 and the graph signal is the word frequency count.

Like in [15], we classify texts for: Jane Austen, Emily Bronte, and Edgar Alan Poe. The WANs of these authors have from $N = 190$ to $N = 210$ function word nodes. We consider a train-test split of $95\% - 5\%$ of the available texts per author and around 8.7% of the train samples are used for validation. This leads to around $1,000$ training samples and 100 validation and test samples. For each author, we

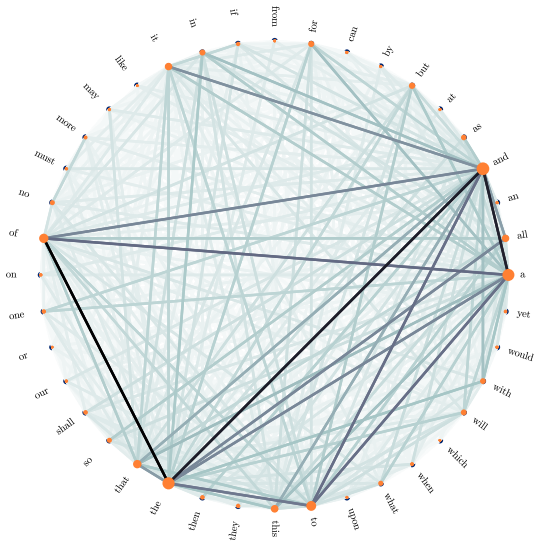

Fig. 2. Example of word adjacency network for the author Robert Louis Stevenson.

Table 2: Authorship attribution accuracy (%).

	Austen	Bronte	Poe
GCNN	64 ± 20	71 ± 16	64 ± 18
RSN	80 ± 19	68 ± 16	72 ± 19
LSSM	$\mathbf{87 \pm 14}$	$\mathbf{73 \pm 15}$	$\mathbf{83 \pm 14}$

extend the training, validation, and test sets by the same number of text samples taken at random from the author pool. All models are trained with batches of 100 samples for 25 epochs, and the learning rate is 5×10^{-3}. The loss function is the cross-entropy and we report average test accuracies for 30 data splits.

In this experiment, we fix the parameters to those of the 1-layer GCNN which achieved the best performance in the source localization experiment—$F = 4$ and $K = 32$—to make for a fair comparison. The results are presented in Table 2. We observe that the RSN outperforms the GCNN for all authors except Brönte, and that the LSSM exhibits the best performance by a large margin.

5. CONCLUSIONS

Implicitly, graph convolutional neural networks carry a state-space model in their convolutional update rule. In this paper, we make this relationship explicit and analyze its behavior from an internal state perspective. By noting that the internal state may explode or vanish depending on the spectrum of the shift operator, we argued that this places a burden on the GCNN parameters because they need to be learned to control such phenomena, leading to a stability-performance trade-off. We then built further links with discrete state-space models to develop a new family of graph neural networks, in which nodal aggregations are performed in a nonlinear and parametric manner. The latter leads to higher degrees of freedom to control the stability-performance trade-off and allows developing new solutions to improve the expressive power of GCNNs. We proposed two such solutions, namely, i) a recursive shift network that includes the input signal in every state update; ii) a long-short term shift memory that allows further increasing the filter order within a layer through the introduction of gating mechanisms akin to the gates in conventional LSTMs. Numerical results on source localization and authorship attribution validate both models. In the future, we plan on investigating the theoretical benefits of these nonlinear aggregation rules.

6. REFERENCES

[1] M. E. J. Newman, *Networks: An Introduction*, Oxford University Press, Oxford, UK, 2010.

[2] E. Bullmore and O. Sporns, "Complex brain networks: Graph theoretical analysis of structural and functional systems," *Nature Reviews Neuroscience*, vol. 10, pp. 186–198, March 2009.

[3] M. O. Jackson, *Social and Economic Networks*, Princeton University Press, Princeton, NJ, 2008.

[4] F. Gama, E. Isufi, G. Leus, and A. Ribeiro, "From graph filters to graph neural networks," *arXiv:2003.03777v3 [cs.LG]*, 8 Aug. 2020, accepted for publication in IEEE Signal Process. Mag.

[5] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and deep locally connected networks on graphs," in *2nd Int. Conf. Learning Representations*, Banff, AB, 14-16 Apr. 2014, pp. 1–14.

[6] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *30th Conf. Neural Inform. Process. Syst.*, Barcelona, Spain, 5-10 Dec. 2016, pp. 3844–3858, Neural Inform. Process. Foundation.

[7] J. Du, J. Shi, S. Kar, and J. M. F. Moura, "On graph convolution for graph CNNs," in *2018 IEEE Data Sci. Workshop*, Lausanne, Switzerland, 4-6 June 2018, pp. 239–243, IEEE.

[8] 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, 15 Feb. 2019.

[9] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *7th Int. Conf. Learning Representations*, New Orleans, LA, 6-9 May 2019, pp. 1–17.

[10] 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.

[11] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1644–1656, 1 Apr. 2013.

[12] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Autoregressive moving average graph filtering," *IEEE Trans. Signal Process.*, vol. 65, no. 2, pp. 274–288, 15 Jan. 2017.

[13] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "CayleyNets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Trans. Signal Process.*, vol. 67, no. 1, pp. 97–109, 1 Jan. 2019.

[14] A. Wijesinghe and Q. Wang, "DFNets: Spectral cnns for graphs with feedback-looped filters," in *33rd Conf. Neural Inform. Process. Syst.*, Vancouver, BC, 8-14 Dec. 2019, pp. 6009–6020, Neural Inform. Process. Syst. Foundation.

[15] E. Isufi, F. Gama, and A. Ribeiro, "EdgeNets: Edge Varying Graph Neural Networks," *arXiv:2001.07620v2 [cs.LG]*, 12 March 2020.

[16] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *6th Int. Conf. Learning Representations*, Vancouver, BC, 30 Apr.-3 May 2018, pp. 1–12.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *31st Conf. Neural Inform. Process. Syst.*, Long Beack, CA, 4-9 Dec. 2017, pp. 5998–6008, Neural Inform. Process. Syst. Foundation.

[18] D. Simon, *Optimal State Estimation: Kalman, H_∞, and Nonlinear Approaches*, John Wiley & Sons, Hoboken, NJ, 2006.

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, Adaptive Comput. Mach. Learning ser. The MIT Press, Cambridge, MA, 2016.

[20] D. Owerko, F. Gama, and A. Ribeiro, "Optimal power flow using graph neural networks," in *45th IEEE Int. Conf. Acoust., Speech and Signal Process.*, Barcelona, Spain, 4-8 May 2020, pp. 5930–5934, IEEE.

[21] F. Gama, E. Tolstaya, and A. Ribeiro, "Graph neural networks for decentralized controllers," *arXiv:2003.10280v2 [cs.LG]*, 21 Oct. 2020.

[22] F. Gama, J. Bruna, and A. Ribeiro, "Stability properties of graph neural networks," *IEEE Trans. Signal Process.*, vol. 68, pp. 5680–5695, 25 Sep. 2020.

[23] D. Zou and G. Lerman, "Graph convolutional neural networks via scattering," *Appl. Comput. Harmonic Anal.*, vol. 49, no. 3, pp. 1046–1074, Nov. 2020.

[24] Z. Chen, S. Villar, L. Chen, and J. Bruna, "On the equivalence between graph isomorphism testing and function approximation with GNNs," in *33rd Conf. Neural Inform. Process. Syst.*, Vancouver, BC, 8-14 Dec. 2019, pp. 15894–15902, Neural Inform. Process. Syst. Foundation.

[25] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and Lehman go neural: Higher-order graph neural networks," in *33rd AAAI Conf. Artificial Intell.*, Honolulu, HI, 27 Jan.-1 Feb. 2019, vol. 33, pp. 4602–4609, Assoc. Advancement Artificial Intell.

[26] S. Pfrommer, F. Gama, and A. Ribeiro, "Discriminability of single-layer graph neural networks," *arXiv:2010.08847v2 [eess.SP]*, 21 Oct. 2020.

[27] A. Agaskar and Y. M. Lu, "A spectral graph uncertainty principle," *IEEE Trans. Inf. Theory*, vol. 59, no. 7, pp. 4338–4356, July 2013.

[28] M. Tsitsvero, S. Barbarossa, and P. Di Lorenzo, "Signals on graphs: Uncertainty principle and sampling," *IEEE Trans. Signal Process.*, vol. 64, no. 18, pp. 4845–4860, 15 Sep. 2016.

[29] O. Teke and P. P. Vaidyanathan, "Uncertainty principles and sparse eigenvectors of graphs," *IEEE Trans. Signal Process.*, vol. 65, no. 20, pp. 5406–5420, 15 Oct. 2017.

[30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[31] D. P. Kingma and J. L. Ba, "ADAM: A method for stochastic optimization," in *3rd Int. Conf. Learning Representations*, San Diego, CA, 7-9 May 2015, pp. 1–15.

[32] S. Segarra, M. Eisen, and A. Ribeiro, "Authorship attribution through function word adjacency networks," *IEEE Trans. Signal Process.*, vol. 63, no. 20, pp. 5464–5478, 15 Oct. 2015.