MIMO Graph Filters for Convolutional Neural Networks

Gama, Fernando; Marques, Antonio G.; Ribeiro, Alejandro; Leus, Geert

DOI
10.1109/SPAWC.2018.8445934

Publication date
2018

Document Version
Final published version

Published in
2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications, SPAWC 2018

MIMO Graph Filters for Convolutional Neural Networks

Fernando Gama, Antonio G. Marques, Alejandro Ribeiro, and Geert Leus

Abstract—**Superior performance and ease of implementation have fostered the adoption of Convolutional Neural Networks (CNNs) for a wide array of inference and reconstruction tasks. CNNs implement three basic blocks: convolution, pooling and pointwise nonlinearity. Since the two first operations are well-defined only on regular-structured data such as audio or images, application of CNNs to contemporary datasets where the information is defined in irregular domains is challenging. This paper investigates CNNs architectures to operate on signals whose support can be modeled using a graph. Architectures that replace the regular convolution with a so-called linear shift-invariant graph filter have been recently proposed. This paper goes one step further and, under the framework of multiple-input multiple-output (MIMO) graph filters, imposes additional structure on the adopted graph filters, to obtain three new (more parsimonious) architectures. The proposed architectures result in a lower number of model parameters, reducing the computational complexity, facilitating the training, and mitigating the risk of overfitting. Simulations show that the proposed simpler architectures achieve similar performance as more complex models.**

Index Terms—**Convolutional neural networks, network data, graph signal processing, MIMO.**

I. Introduction

Convolutional Neural Networks (CNNs) have emerged as the information processing architecture of choice in a wide range of fields as diverse as pattern recognition, computer vision and medicine, for solving problems involving inference and reconstruction tasks [1]–[3]. CNNs have demonstrated remarkable performance, as well as ease of implementation and low online computational complexity [4], [5]. CNNs take the input data and process it through several layers, each of which performs three simple operations on the output of the previous layer. These three operations are convolution, pointwise nonlinearity and pooling. The objective of such an architecture is to progressively extract useful information, from local features to more global aspects of the data. This is mainly achieved by the combination of convolution and pooling operations which sequentially combine data that is located further away. The nonlinearity dons the architecture with enough flexibility to represent a richer class of functions that may describe the problem.

One of the most outstanding characteristics of CNNs is that the filters used for convolution can be efficiently *learned* from training datasets by means of a backpropagation algorithm [6]. This implies that the CNN architecture is capable of *learning* which are the most useful features for the task at hand. Intimately related to the capability of successful training, is the fact that the filters used are small, thus containing few parameters, making training easier. While, convolution and pooling are well-defined only in regular domains such as time or space, contemporary data is increasingly being described on domains that exhibit more irregular behavior [7], with examples including marketing, social networks, or genetics [8]–[10]. With the objective of extending the remarkable performance of CNNs to broader data domains, extensions capable of processing network data have been developed [11]–[18], see [19] for a survey. In particular, the works of [11], [14] make use of the concept of graph filters (GFs) [20], [21] to extend the convolution operation to graph signals [22]. Leveraging the framework of multiple-input multiple-output (MIMO) GFs on existing results, this paper proposes three novel architectures for GF-based CNNs. The main idea is to replace the bank of parallel GFs with a more structured filtering block which reduces the degrees of freedom (number of parameters) on each layer. This new architecture facilitates the training, incurs reduced computational complexity, and can be beneficial to avoid overfitting.

Section II introduces notation and reviews existing GF-based CNNs under the framework of MIMO GFs. Section III describes the novel architectures. Section IV presents simulations showing the benefits of our schemes. Concluding remarks are provided in Section V.

II. CNNs on Graph Signals

Let $\mathbf{x} \in \mathcal{X}$ be the input data defined on some field $\mathcal{X}$ and let $\mathbf{y} \in \mathcal{Y}$ be the output data such that $\mathbf{y} = f(\mathbf{x})$ for some (unknown) function $f : \mathcal{X} \to \mathcal{Y}$. The general objective in machine learning is to estimate or *learn* the function f [23].

A neural network is an information processing architecture that aims at constructing an estimator $\hat{f}$ that consists of a concatenation of layers, each of which applies three simple operations on the output of the layer before, namely a linear transform, a pointwise nonlinearity and a pooling operator. Formally, the estimator $\hat{f}$ can be written as $\hat{f} = f_L \circ f_{L-1} \circ \cdots \circ f_1$ where f_ℓ denotes the operations to be applied at layer $\ell = 1, \ldots, L$ [24]. Denote by $\mathbf{x}_\ell \in \mathcal{X}_\ell$ the N_ℓ-dimensional output of layer ℓ defined over field $\mathcal{X}_\ell$, by $\mathbf{A}_\ell : \mathcal{X}_{\ell-1} \to \mathcal{X}'_\ell$ a linear transform between fields $\mathcal{X}_{\ell-1}$ and $\mathcal{X}'_\ell$, by $\sigma_\ell : \mathcal{X}'_\ell \to \mathcal{X}'_\ell$ a pointwise nonlinearity, and by $\mathcal{P}_\ell : \mathcal{X}'_\ell \to \mathcal{X}_\ell$ the pooling operator. Then, each layer can be described as $\mathbf{x}_\ell =$

Work supported by USA NSF CCF-1717120 and ARO W911NF1710438, Spanish MINECO grants No. TEC2013-41604-R and TEC2016-75361-R. F. Gama and A. Ribeiro are with the Dept. of Electrical and Systems Eng., Univ. of Pennsylvania., A. G. Marques is with the Dept. of Signal Theory and Comms., King Juan Carlos Univ., G. Leus is with the Dept. of Microelectronics, Delft Univ. of Technology. Emails: {fgama,aribeiro}@seas.upenn.edu, antonio.garcia.marques@urjc.es, and g.j.t.leus@tudelft.nl .

$f_\ell(\mathbf{x}_{\ell-1}) = \mathcal{P}_\ell\{\sigma_\ell(\mathbf{A}_\ell \mathbf{x}_{\ell-1})\}$, $\ell = 1, \ldots, L$ with $\mathcal{X}_0 \equiv \mathcal{X}$ the input data field and $\mathcal{X}_L \equiv \mathcal{Y}$ the output data field.

In particular, a CNN assumes that the linear operator $\mathbf{A}_\ell$ is comprised of a collection of F_ℓ filters of small support K_ℓ, $\mathbf{A}_\ell = \{\mathbf{h}_{\ell,1}, \ldots, \mathbf{h}_{\ell,F_\ell}\}$. Then, the application of the linear operator yields a collection of signals $\{\mathbf{h}_{\ell,1} * \mathbf{x}_{\ell-1}, \ldots, \mathbf{h}_{\ell,F_\ell} * \mathbf{x}_{\ell-1}\}$, in which each element $i = 1, ..., N_{\ell-1}$,

$$[\mathbf{h}_{\ell,f} * \mathbf{x}_{\ell-1}]_i = \sum_{k=0}^{K_\ell - 1} [\mathbf{h}_{\ell,f}]_k [\mathbf{x}_{\ell-1}]_{i-k} \tag{1}$$

is considered a feature, $f = 1, \ldots, F_\ell$, and where we assumed that $[\mathbf{x}_{\ell-1}]_{i-k} = 0$ for $i \leq k$. Using filters with a small support has a twofold goal. First, the convolution operation linearly relates *nearby* values (consecutive time instants, or neighboring pixels) and consolidates them in a feature value that aggregates this local information. Second, such filters only have a few parameters and, therefore, are easy to be *learned* from data. We also note that the pooling operation serves the function of changing the *resolution* of data, so that on each layer, the *nearby* values that are related by the convolution operator are actually located further away. In this way, the convolution and pooling operations act in tandem to guarantee that the CNN aggregates information at different levels, from local to global.

The operation of convolution, in particular, depends upon the existence of a notion of neighborhood. Such a notion also exists in domains like manifolds and graphs, and thus, the convolution can be extended to operate on signals defined on these irregular domains. In particular, for signals defined on graphs, let us start by considering the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, where $\mathcal{V}$ is the set of N nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, and $\mathcal{W} : \mathcal{E} \to \mathbb{R}$ is the function that assigns weights to the edges. The neighborhood of node $i \in \mathcal{V}$ is then defined as the set of nodes $\mathcal{N}_i = \{j \in \mathcal{V} : (j,i) \in \mathcal{E}\}$. With these notations in place, a graph signal is defined as a map which associates a real value to each element of $\mathcal{V}$. This graph signal can be conveniently represented as the vector $\mathbf{x} \in \mathbb{R}^N$, where the i-th element $[\mathbf{x}]_i = x_i$ corresponding to the value of the signal at node i. In order to relate the values of the graph signal at any node with those at its neighborhood, we can make use of a matrix description of the graph. More precisely, let $\mathbf{S} \in \mathbb{R}^{N \times N}$ be a *graph shift operator* (GSO) which is a matrix whose (i,j)-th element can be nonzero if and only if $i = j$ or if $(j,i) \in \mathcal{E}$ [20]. Note then, that $\mathbf{S}\mathbf{x}$ is a linear combination of the values of the signal with that of its neighbors. More precisely, we have that, for each $i \in \mathcal{V}$

$$[\mathbf{S}\mathbf{x}]_i = \sum_{j=1}^{N} [\mathbf{S}]_{ij} x_j = \sum_{j \in \mathcal{N}_i \cup \{i\}} [\mathbf{S}]_{ij} x_j \tag{2}$$

where the second equality follows because $[\mathbf{S}]_{ij} = 0$ if $j \notin \mathcal{N}_i \cup \{i\}$. The operation in (2) is the basic element to extend the notion of convolution (filtering) to signals defined on graphs; see, e.g., [21]. First, observe that while $\mathbf{S}\mathbf{x}$ collects information from the one-hop neighborhood of each node, $\mathbf{S}^k\mathbf{x}$ collects information up to the k-hop neighborhood of each node. Denote by $\mathbf{h} = [h_0, \ldots, h_{K-1}]^\mathsf{T} \in \mathbb{R}^K$ a collection of K filter taps. Then, we can linearly combine neighboring values up to the $(K-1)$-hop neighborhood by [cf. (1)]

$$[\mathbf{h} * \mathbf{x}]_i = \sum_{k=0}^{K-1} h_k [\mathbf{S}^k \mathbf{x}]_i. \tag{3}$$

Upon defining matrix $\mathbf{H} := \sum_{k=0}^{K-1} h_k \mathbf{S}^k \in \mathbb{R}^{N \times N}$, (3) can be equivalently written as

$$\mathbf{h} * \mathbf{x} = \mathbf{H}\mathbf{x} = \sum_{k=0}^{K-1} h_k \mathbf{S}^k \mathbf{x}. \tag{4}$$

with $\mathbf{H}$ being known as a linear shift-invariant (LSI) GF [21].

Since LSI-GFs are regarded as the generalization of convolutions to operate on graph signals, the operator in (3) can be used to extend CNNs to operate on graph signals [11]. More specifically, assume that in each layer ℓ of the CNN, output $\mathbf{x}_\ell$ consists of F_ℓ features, each of which is considered a graph signal $\mathbf{x}_\ell^{(f)} \in \mathbb{R}^{N_\ell \times 1}$ defined on an N_ℓ-node graph described by GSO $\mathbf{S}_\ell \in \mathbb{R}^{N_\ell \times N_\ell}$, $f = 1, \ldots, F_\ell$. Then, all F_ℓ features can be concatenated on vector $\mathbf{x}_\ell = [(\mathbf{x}_\ell^{(1)})^\mathsf{T}, \ldots, (\mathbf{x}_\ell^{(F_\ell)})^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{F_\ell N_\ell \times 1}$. Assume that the linear transform $\mathbf{A}_\ell$ constructs F_ℓ features out of the existing $F_{\ell-1}$ ones. Then, $\mathbf{A}_\ell$ can be regarded as a MIMO GF since it takes $F_{\ell-1}$ input signals and outputs F_ℓ graph signals. By denoting as $\otimes$ the Kronecker matrix product, the output of the convolution operation on graph signals can be compactly written as a MIMO GF as follows

$$\mathbf{A}_\ell \mathbf{x}_{\ell-1} = \sum_{k=0}^{K_\ell - 1} \left(\mathbf{H}_{\ell,k} \otimes \mathbf{S}_\ell^k \right) \mathbf{x}_{\ell-1} \tag{5}$$

where $\mathbf{H}_{\ell,k} \in \mathbb{R}^{F_\ell \times F_{\ell-1}}$ contains the filter taps corresponding to the $F_{\ell-1}F_\ell$ LSI-GFs employed. More precisely, by denoting as $[\mathbf{H}_{\ell,k}]_{f,f'} = h_{k,f,f'}^{(\ell)}$, the filter taps of the (f,f') filter can be written as $\mathbf{h}_{f,f'}^{(\ell)} = [h_{0,f,f'}^{(\ell)}, \ldots, h_{K_{\ell-1},f,f'}^{(\ell)}]^\mathsf{T} \in \mathbb{R}^{K_\ell \times 1}$, $f = 1, \ldots, F_\ell$, $f' = 1, \ldots, F_{\ell-1}$. Construction (5) builds F_ℓ different LSI-GFs for each of the $F_{\ell-1}$ features contained in the previous layer, totaling $F_{\ell-1}F_\ell$ LSI-GFs. The total number of *trainable* parameters is thus $F_{\ell-1}F_\ell K_\ell$. While written differently, the MIMO GF in (5) represents the per-layer architecture proposed in [14].

Finally, with respect to the pooling operation, the use of multiscale hierarchical clustering to reduce the size of the graph in each layer has been employed, yielding $N_\ell \leq N_{\ell-1}$ and $\mathbf{S}_\ell$ the corresponding GSO of each layer [11], [14]. Also, due to the computational and performance issues of clustering, alternative approaches which do not rely on pooling exist [16]. In this work, we focus on the convolutional operation of CNNs on graph signals, letting the user determine the preferred choice of pooling scheme.

III. CNNs based on structured MIMO GFs

This paper proposes three new architectures for CNNs on graph signals, obtained by imposing a certain parsimonious representation on the MIMO GF matrices $\{\mathbf{H}_{\ell,1}, \ldots, \mathbf{H}_{\ell,K_\ell}\}$. The resulting architectures yield a considerably lower number of trainable parameters, reducing the complexity of the network, as well as avoiding certain pitfalls such as overfitting or the curse of dimensionality [25]. For simplicity, from now on, we focus on some specific layer ℓ, hence dropping the subscript on all notations. We denote as $\mathbf{x} = \mathbf{x}_{\ell-1}$ the input, with $\mathbf{x} = [\mathbf{x}_1^\mathsf{T}, \ldots, \mathbf{x}_Q^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{QN \times 1}$ where $\mathbf{x}_q \in \mathbb{R}^N$ are the $F_{\ell-1} = Q$ input features, $q = 1, \ldots, Q$. We denote as $\mathbf{y} = \mathbf{x}_\ell$ the output features, with $\mathbf{y} = [\mathbf{y}_1^\mathsf{T}, \ldots, \mathbf{y}_P^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{PN \times 1}$ where $\mathbf{y}_p \in \mathbb{R}^{N \times 1}$ are the $F_\ell = P$ output features, $p = 1, \ldots, P$. The length of the filters is $K_\ell = K$ and the matrix of filter taps $\mathbf{H}_{\ell,k} = \mathbf{H}_k \in \mathbb{R}^{P \times Q}$ has elements $[\mathbf{H}_k]_{p,q} = h_{k,p,q}$, $k = 0, \ldots, K-1$, $p = 1, \ldots, P$, $q = 1, \ldots, Q$. Each of the (p,q) filters is represented by a vector of filter taps $\mathbf{h}_{p,q} = [h_{0,p,q}, \ldots, h_{K-1,p,q}]^\mathsf{T} \in \mathbb{R}^{K \times 1}$. Equation (5) becomes

$$\mathbf{y} = \sum_{k=0}^{K-1} \left(\mathbf{H}_k \otimes \mathbf{S}^k\right) \mathbf{x} \tag{6}$$

and each new feature is computed as

$$\mathbf{y}_p = \sum_{q=1}^{Q} \sum_{k=0}^{K-1} h_{k,p,q} \mathbf{S}^k \mathbf{x}_q \ , \ p = 1, \ldots, P. \tag{7}$$

The design variables are the collection of matrices $\{\mathbf{H}_0, \ldots, \mathbf{H}_{K-1}\}$ containing the PQ filters, and thus totaling PQK parameters.

A. Aggregating the input features

First, we propose to aggregate all input features so as to reduce the number of filters. Instead of designing P different filters for each one of the Q input features, we first aggregate the Q input features into one graph signal, and proceed to design P different filters to be applied to this graph signal.

This strategy amounts to designing filter taps $\mathbf{h}_{p,1} = [h_{0,p,1}, \ldots, h_{K-1,p,1}]^\mathsf{T} \in \mathbb{R}^{K \times 1}$ for $p = 1, \ldots, P$. Then, matrix $\mathbf{H}_k$ in (6) becomes a replication of the first column

$$\mathbf{H}_k = \begin{bmatrix} h_{k,1,1} & h_{k,1,1} & \cdots & h_{k,1,1} \\ h_{k,2,1} & h_{k,2,1} & \cdots & h_{k,2,1} \\ \vdots & \vdots & \ddots & \vdots \\ h_{k,P,1} & h_{k,P,1} & \cdots & h_{k,P,1} \end{bmatrix}. \tag{8}$$

Each output feature (7) is thus computed as

$$\mathbf{y}_p = \sum_{q=1}^{Q} \sum_{k=1}^{K} h_{k,p,1} \mathbf{S}^k \mathbf{x}_q = \sum_{k=1}^{K} h_{k,p,1} \mathbf{S}^k \left(\sum_{q=1}^{Q} \mathbf{x}_q\right) \tag{9}$$

for $p = 1, \ldots, P$.

Observe that the structure imposed on (8) leads to only P different LSI-GFs and therefore the number of trainable parameters has been reduced to PK. The effect of this filter, as observed from (9) is to first aggregate all the input features into a single graph signal $\sum_{q=1}^{Q} \mathbf{x}_q$ and then applying P different filters to it, yielding the P different output features.

B. Consolidating output features

The second proposed architecture consists of designing one filter for each input feature and then consolidating all the filtered input features into a single output feature.

In order to do this, we need to design Q LSI-GFs described by filter taps $\mathbf{h}_{1,q} = [h_{0,1,q}, \ldots, h_{K-1,1,q}]^\mathsf{T} \in \mathbb{R}^{K-1}$ specific to each input feature $q = 1, \ldots, Q$. Matrix $\mathbf{H}_k$ in (6) can thus be written as a replication of the first row

$$\mathbf{H}_k = \begin{bmatrix} h_{k,1,1} & h_{k,1,2} & \cdots & h_{k,1,Q} \\ h_{k,1,1} & h_{k,1,2} & \cdots & h_{k,1,Q} \\ \vdots & \vdots & \ddots & \vdots \\ h_{k,1,1} & h_{k,1,2} & \cdots & h_{k,1,Q} \end{bmatrix}. \tag{10}$$

This leads to each output feature (7) being calculated as

$$\mathbf{y}_p = \sum_{q=1}^{Q} \left(\sum_{k=0}^{K-1} h_{k,1,q} \mathbf{S}^k\right) \mathbf{x}_q = \sum_{q=1}^{Q} \mathbf{G}_q \mathbf{x}_q \tag{11}$$

for $p = 1, \ldots, P$.

Note that all P output features $\mathbf{y}_p$ are actually the same, since (11) does not depend on p. Thus, the filter taps given by (10) actually yield a single output feature, which is the graph signal given by (11). This could be particularly useful for reducing operational complexity of subsequent layers, since graph signals can be handled easily. Observe that the imposed structure (10) containing Q distinct filters yields QK trainable parameters.

C. Convolution of features

As a third approach to reducing the number of parameters involved in (6), we consider a set of $P+Q-1$ filters to be applied sequentially and progressively to the input features, yielding output features that resemble convolutions of the input features.

Let $\mathbf{h}_{p-q+1} = [h_{0,p-q+1}, \ldots, h_{K-1,p-q+1}]^\mathsf{T} \in \mathbb{R}^{K-1}$ be a set of filter taps, $p = 1, \ldots, P$, $q = 1, \ldots, Q$. Then, for this third proposed strategy, matrix $\mathbf{H}_k$ in (6) becomes

$$\mathbf{H}_k = \begin{bmatrix} h_{k,1} & h_{k,0} & \cdots & h_{k,2-Q} \\ h_{k,2} & h_{k,1} & \cdots & h_{k,3-Q} \\ \vdots & \vdots & \ddots & \vdots \\ h_{k,P} & h_{k,P-1} & \cdots & h_{k,P+1-Q} \end{bmatrix}. \tag{12}$$

The output features are then obtained using (12) in (7) yielding

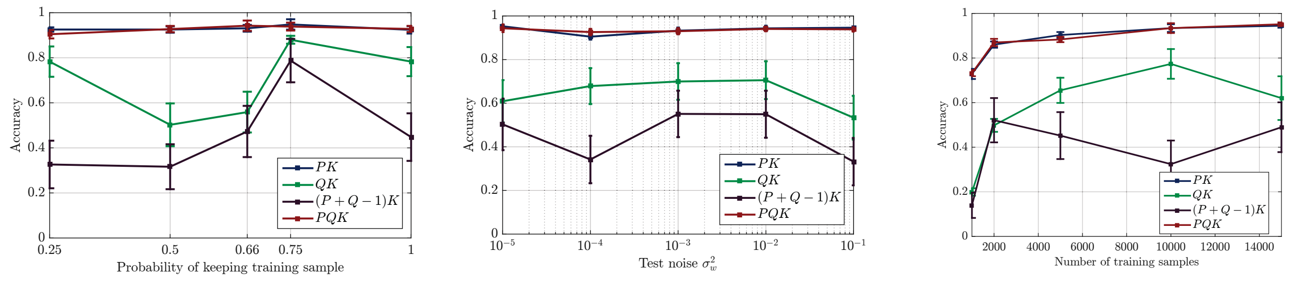
$$\mathbf{y}_p = \sum_{q=1}^{Q} \sum_{k=0}^{K-1} h_{k,p+1-q} \mathbf{S}^k \mathbf{x}_q = \sum_{k=0}^{K-1} \mathbf{S}^k \sum_{q=1}^{Q} h_{k,p+1-q} \mathbf{x}_q \tag{13}$$

for $p = 1, \ldots, P$.

From (13) we observe that each output feature $\mathbf{y}_p$ can be thought of as the convolution of the input feature vector $\mathbf{x}$ with the collection of filter taps given by $\{h_{k,p}, \ldots, h_{k,p+1-Q}\}$. Note also that consecutive output features weigh input features similarly. In a way, (13) acts as a smoother of input features. Matrix $\mathbf{H}_k$ in (12) is a Toeplitz matrix and thus has $P+Q-1$ elements, so that the total number of trainable parameters is $(P+Q-1)K$.

IV. Numerical tests

Here we compare the performance of the three architectures proposed in Sections III-A, III-B and III-C, which involve, respectively, PK, QK and $(P+Q-1)K$ parameters, with that of [14], which involves PQK parameters [cf. (5)]. In the first testcase we consider a synthetic dataset of a source localization problem in which different diffused graph signals



(a) (b) (c)

Figure 1: Accuracy in the source localization problem. Results were averaged across 10 different graph realizations. For clarity of figures, error bars represent $1/4$ of the estimated variance. (a) As a function of the probability of keeping a sample during the training phase, i.e. $1-\text{prob}_{\text{dropout}}$. (b) As a function of the noise in the test set. (c) As a function of the number of training samples. Overall, we observe that the full architecture (5) yields a performance similar to that proposed in Section III-A.

Architecture	Parameters	Accuracy
PQK	$10,400$	93.9%
PK	480	94.8%
QK	165	88.0%
$(P+Q-1)K$	635	78.8%

Table I: Source localization results for $N = 16$ nodes.

Architecture	Parameters	Accuracy
PQK	$10,400$	61.32%
PK	480	62.48%
QK	165	58.22%
$(P+Q-1)K$	635	64.06%

Table II: Results for classification on `20NEWS` dataset on a `word2vec` graph embedding of $N = 5,000$ nodes.

are processed to determine the single node that originated them. In the second testcase we use the `20NEWS` dataset and a `word2vec` embedding underlying graph to classify articles in one out of 20 different categories [26]. For both problems, we evaluate an architecture with 2 convolutional layers, the first one generating $F_1 = 32$ features, and the second one outputting $F_2 = 64$ features. GFs are of length $K_1 = K_2 = K = 5$. No pooling is employed, so that $N_1 = N_2 = N$, the number of nodes in the graph, and $\mathbf{S}_1 = \mathbf{S}_2 = \mathbf{S}$ is the GSO specific to each testcase. We denote as PQK the architecture in [14], and as PK, QK and $(P+Q-1)K$ the ones developed in Sections III-A, III-B and III-C, respectively. The number of parameters in the convolutional layers are 10400 for PQK, 480 for PK, 165 for QK and 635 for $(P+Q-1)K$. The selected nonlinearity is a ReLU applied at each layer and all architectures include a readout layer. For the training stage in both problems, an ADAM optimizer with learning rate 0.005 was employed [27], for 20 epochs and batch size of 100.

A. Source localization.

Consider a connected Stochastic Block Model (SBM) graph with $N = 16$ nodes divided in 4 communities, with intra-community edge probability of 0.8 and intercommunity edge probability of 0.2. Let $\mathbf{W}$ denote its adjacency matrix. With $\boldsymbol{\delta}_c$ representing a graph signal taking the value 1 at node c and 0 elsewhere, the signal $\mathbf{x} = \mathbf{W}^t\boldsymbol{\delta}_c$ is a diffused version of the sparse input $\boldsymbol{\delta}_c$ for some unknown $0 \leq t \leq N-1$. The objective is to determine the source c that originated the signal $\mathbf{x}$ irrespective of time t. To that end, we create a set of N_{train} labeled training samples $\{(c', \mathbf{x}')\}$ where $\mathbf{x}' = \mathbf{W}^t\boldsymbol{\delta}_{c'}$ with both c' and t chosen at random. Then we create a test set with N_{test} samples in the same fashion, but we add i.i.d. zero-mean Gaussian noise $\mathbf{w}$ with variance σ_w^2, so that the signals to be classified are $\mathbf{W}^t\boldsymbol{\delta}_c + \mathbf{w}$. The goal is to use the training samples to design a CNN that determines the source (node) c that originated the diffused signal.

First, we run the source localization problem on 10 different realizations of randomly generated SBM graphs. For each graph, we train the four architectures using dropout with probability of keeping each training sample of 0.75. The total number of training samples is $10,000$. Once trained, the architectures are tested on a test set of 200 samples for each graph, contaminated with noise of variance $\sigma_w^2 = 10^{-1}$. Results are listed in Table I, where the accuracy is averaged over the 200 samples, over the 10 different graph realizations. We observe that the PK architecture proposed in Section III-A outperforms the full PQK architecture, with two orders of magnitude less parameters. Also, the QK and the $(P+Q-1)K$ architectures yields reasonable performances.

Additionally, we run tests changing the values of several of the simulations parameters. In Fig. 1a we observe the accuracy obtained when varying the probability of keeping training samples. It is noted that the PK architecture performs as well as the full PQK architecture. It is also observed that the other two architectures have significant variance, which implies that they depend heavily on the topology of the graph. The effect of noise σ_w^2 on the test samples can be observed in Fig. 1b. We observe that all four architectures are relatively robust to noise. The QK and $(P+Q-1)K$ architectures exhibit a dip in performance for the highest noise value simulated. Finally, in Fig. 1c we show the performance of all four architectures as a function of the number of training samples. We see that the PK and the full PQK architecture improve in performance as more training samples are considered, with a huge increase between $1,000$ and $2,000$ training samples. This same increase is observed for the remaining two architectures, although their performance behaves somewhat erratically afterwards. All in all, from these set of simulations, we observe that the PK

architecture performs as well as the full PQK architecture, but utilizing almost two orders of magnitude less parameters. We also observe that the QK and $(P+Q-1)K$ architectures have a high dependence on the topology of the network.

B. `20NEWS` dataset

Here we consider the classification of articles in the `20NEWS` dataset which consists of $18,846$ texts ($11,314$ of which are used for training and $7,532$ for testing) [26]. The graph signals are constructed as in [14]: each document x is represented using a normalized bag-of-words model and the underlying graph support is constructed using a 16-NN graph on the `word2vec` embedding [28] considering the $5,000$ most common words. The GSO adopted is the normalized Laplacian. No dropout is used in the training phase. Accuracy results are listed in Table II, demonstrating that the PK and $(P+Q-1)K$ architectures outperform the full PQK one, but requiring almost 100 times less parameters.

V. Conclusions

In this paper, we have studied the problem of extending CNNs to operate on graph signals. More precisely, we reframed existing architectures under the concept of MIMO GFs, and leveraged structured representations of such filters to reduce the number of trainable parameters involved. We proposed three new architectures, each of which arises from adopting a different parsimonious model on the MIMO GF matrices. All the resulting architectures yield a lower number of trainable parameters, reducing computational complexity, as well as helping in avoiding certain pitfalls of training like overfitting or the curse of dimensionality.

We have applied the three proposed architectures to a synthetic problem on source localization, and compared its performance with the more complex, full MIMO GF model. We analyzed performance as a function of dropout probability in the training phase, noise in the test samples, and number of training samples. We noted that the proposed architecture that aggregates input features (Section III-A) has a performance similar to that of the full model, but involving two orders of magnitude less parameters. The other two architectures offer comparable performance for certain values of the analyzed parameters. Finally, we utilized the proposed architectures on the problem of classifying articles of the `20NEWS` dataset. In this case, we observed that two of the proposed parsimonious models outperform the full model.

References

[1] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1872–1886, Aug. 2013.

[2] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *2010 IEEE Int. Symp. Circuits and Syst.*, Paris, France, 30 May-2 June 2010, IEEE.

[3] H. Greenspan, B. van Ginneken, and R. M. Summers, "Deep learning in medical imaging: Overview and future promise of an exciting new technique," *IEEE Trans. Med. Imag.*, vol. 35, no. 5, pp. 1153–1159, May 2016.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 85–117, May 2015.

[5] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, and N. Seliya, "Deep learning applications and challenges in big data analytics," *J. Big Data*, vol. 2, no. 1, pp. 1–21, Dec. 2015.

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[7] D. Lazer et al., "Life in the network: The coming age of computational social science," *Science*, vol. 323, no. 5915, pp. 721–723, Feb. 2009.

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

[9] E. H. Davidson et al., "A genomic regulatory network for development," *Science*, vol. 295, no. 5560, pp. 1669–1678, Feb. 2002.

[10] J. Haupt, W. U. Bajwa, M. Rabbat, and R. Nowak, "Compressed sensing for networked data," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 92–101, March 2008.

[11] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and deep locally connected networks on graphs," *arXiv:1312.6203v3 [cs.LG]*, 21 May 2014.

[12] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv:1506.051631v1 [cs.LG]*, 16 June 2015.

[13] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *33rd Int. Conf. Mach. Learning*, New York, NY, 24-26 June 2016.

[14] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Neural Inform. Process. Syst. 2016*, Barcelona, Spain, 5-10 Dec. 2016, NIPS Foundation.

[15] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th Int. Conf. Learning Representations*, Toulon, France, 24-26 Apr. 2017, Assoc. Comput. Linguistics.

[16] F. Gama, G. J. T. Leus, A. G. Marques, and A. Ribeiro, "Convolutional neural networks via node-varying graph filters," *arXiv:1710.10355v1 [cs.LG]*, 27 Oct. 2017.

[17] B. Pasdeloup, V. Gripon, J.-C. Vialatte, and D. Pastor, "Convolutional neural networks on irregular domains through approximate translations on inferred graphs," *arXiv:1710.10035v1 [cs.DM]*, 27 Oct. 2017.

[18] J. Du, S. Zhang, G. Wu, J. M. F. Moura, and S. Kar, "Topology adaptive graph convolutional networks," *arXiv:1710.10370v2 [cs.LG]*, 2 Nov. 2017.

[19] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, July 2017.

[20] A. Sandyhaila and J. M. F. Moura, "Discrete signal processing on graphs: Frequency analysis," *IEEE Trans. Signal Process.*, vol. 62, no. 12, pp. 3042–3054, June 2014.

[21] S. Segarra, A. G. Marques, and A. Ribeiro, "Optimal graph-filter design and applications to distributed linear network operators," *IEEE Trans. Signal Process.*, vol. 65, no. 15, pp. 4117–4131, Aug. 2017.

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

[23] M. J. Kearns and U. V. Vazirani, *An Introduction to Computational Learning Theory*, The MIT Press, Cambridge, MA, 1994.

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, The Adaptive Computation and Machine Learning Series. The MIT Press, Cambridge, MA, 2016.

[25] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recognition 2017*, Honolulu, HI, 21-26 July 2017, IEEE Comput. Soc.

[26] T. Joachims, "Analysis of the Rocchio algorithm with TFIDF for text categorization," Computer Science Technical Report CMU-CS-96-118, Carnegie Mellon University, 1996.

[27] 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, Assoc. Comput. Linguistics.

[28] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st Int. Conf. Learning Representations*, Scottsdale, AZ, 2-4 May 2013, Assoc. Comput. Linguistics.