AGGREGATION GRAPH NEURAL NETWORKS

Fernando Gama[†]*, Antonio G. Marques*[*]*, Alejandro Ribeiro*[†]*, and Geert Leus*[‡]

† Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, USA
∗ Department of Signal Theory and Communications, King Juan Carlos University, Madrid, Spain
‡ Department of Microelectronics, Delft University of Technology, Delft, The Netherlands

ABSTRACT

Graph neural networks (GNNs) regularize classical neural networks by exploiting the underlying irregular structure supporting graph data, extending its application to broader data domains. The aggregation GNN presented here is a novel GNN that exploits the fact that the data collected at a single node by means of successive local exchanges with neighbors exhibits a regular structure. Thus, regular convolution and regular pooling yield an appropriately regularized GNN. To address some scalability issues that arise when collecting all the information at a single node, we propose a multi-node aggregation GNN that constructs regional features that are later aggregated into more global features and so on. We show superior performance in a source localization problem on synthetic graphs and on the authorship attribution problem.

Index Terms— graph neural networks, convolutional neural networks, network data, graph signal processing

1. INTRODUCTION

Convolutional neural networks (CNNs) have become the de facto standard for solving a myriad of classification and regression tasks involving images and time signals, as evidenced in their ubiquitous application in a wide array of fields, such as pattern recognition, computer vision and medicine [1, 2].

The resounding success of CNNs might be rooted in the fact that they regularize classical neural networks (i.e. multilayer perceptrons) by exploiting the underlying regular structure of the data. More precisely, by replacing general linear transforms by convolutions with banks of filters, they address the statistical issue of the curse of dimensionality, the optimization issue of needing a large dataset, and the computational issue of having to compute large matrix multiplication results [3].

Data collected from networks usually deviates from the regular structure that is exhibited in images or time signals, since elements in each datapoint are related by arbitrary pairwise relationships described by an underlying graph support [4–6]. In the hope of extending the remarkable performance of CNNs to new data domains, graph neural networks (GNNs) have emerged [7]. Most popular solutions entail the replacement of convolutions by banks of linear shift-invariant graph filters, as carried out by [8–11] and reducing the size of the graph by either graph coarsening [9] or zero-padding [10]. Alternative regularizations using node-variant graph filters [12] or MIMO graph filters [13] are also possible. Graph neural networks have also enjoyed great popularity in the problem of semi-supervised learning, with several architectures that include the use of graph filters [14], receptive fields [15] and attention networks [16]. In what follows, we focus on the problem of either classification or regression on graph signals (the dataset constitutes signals, all of them supported on the same graph, as is the case of weather measurements [17] or recommendation systems [18, 19]).

The objective of this paper is to introduce a novel architecture called *aggregation GNN*. This architecture exploits the fact that all the relevant information can be collected at a single node by means of local exchanges with neighbors (Sec. 3). This aggregated signal exhibits a regular structure that takes into account the underlying graph support. Thus, applying regular convolution does indeed linearly relate neighboring values of the signal, and using regular pooling lends itself naturally to a multi-resolution analysis in terms of summarizing information from further away neighborhoods. In order to address some scalability issues that arise in large networks, we introduce multi-node aggregation GNNs (Sec. 4) where we distribute the computation of regional features across several nodes, and then aggregate the resulting summaries by means of zero-padding. We show in two experiments involving a source localization on a synthetic graph and an authorship attribution problem, that the presented architecture outperforms GNNs using graph filters and graph coarsening (Sec. 5). An introduction to GNNs can be found in Sec. 2 and conclusions in Sec. 6.

2. GRAPH NEURAL NETWORKS

Let $\mathcal{T} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{|\mathcal{T}|}$ be a training set comprised of $|\mathcal{T}|$ examples. We typically consider $\mathbf{x} \in \mathcal{X}$ to be the input data and $\mathbf{y} \in \mathcal{Y}$ to be some desired target representation that is useful for the task at hand. The objective is to learn how to obtain an adequate representation $\hat{\mathbf{y}} \in \mathcal{Y}$ for some input $\mathbf{x} \notin \mathcal{T}$. In order to achieve this, we propose a model $f : \mathcal{X} \to \mathcal{Y}$ such that $\hat{\mathbf{y}} = f(\mathbf{x})$ and that minimizes some loss function $\mathcal{L}(\hat{\mathbf{y}}_i, \mathbf{y}_i)$ over the training set $\hat{\mathbf{y}}_i = f(\mathbf{x}_i)$, $\mathbf{y}_i \in \mathcal{T}$. It is expected that minimizing $\mathcal{L}$ would lead to a good model f that generalizes well to unseen data $\mathbf{x} \notin \mathcal{T}$.

In the case of neural networks, the proposed model f is a lay-

Supported by USA NSF CCF 1717120, ARO W911NF1710438, ISTC-WAS and Intel AI DevCloud; and Spanish MINECO TEC2013-41604-R and TEC2016-75361-R.

ered information processing architecture $f = f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)}$ consisting of a succession of linear transforms composed with pointwise nonlinearities

$$\mathbf{x}^{(\ell)} = f^{(\ell)}(\mathbf{x}^{(\ell-1)}) = \sigma^{(\ell)}(\mathbf{A}^{(\ell)}\mathbf{x}^{(\ell-1)}) \, , \; \ell = 1, \ldots, L \quad (1)$$

where $\mathbf{x}^{(\ell)} \in \mathcal{X}^{(\ell)}$, $\mathbf{A}^{(\ell)} : \mathcal{X}^{(\ell-1)} \to \mathcal{X}^{(\ell)}$ is a linear transform and $\sigma^{(\ell)} : \mathcal{X}^{(\ell)} \to \mathcal{X}^{(\ell)}$ is a pointwise nonlinearity. Typically, we have $\mathbf{x}^{(0)} = \mathbf{x}$ and $\mathbf{x}^{(L)} = \hat{\mathbf{y}}$. Once this model is set (i.e. the number of layers L and the dimensions of $\mathbf{x}^{(\ell)}$ are fixed, and the nonlinearity $\sigma^{(\ell)}$ has been chosen), the optimal linear transformations can be obtained by minimizing the loss function over the training set, $\min_{\mathbf{A}^{(\ell)}} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{T}} \mathcal{L}(\hat{\mathbf{y}}_i, \mathbf{y}_i)$.

We observe that the number of parameters to learn in the linear transform $\mathbf{A}^{(\ell)}$ depends on both the size of the input and the size of the output. This causes major hurdles in the application of neural networks with arbitrary linear transforms to large datasets, such as the curse of dimensionality, the need for larger and richer datasets, and a large computational cost. All these challenges can be overcome by regularizing the linear transform $\mathbf{A}^{(\ell)}$ to take into account some structure present in the data.

In this paper, we focus on graph data, and therefore, we seek for a regularization of the linear transform that takes into account its graph nature, with the objective of making neural networks efficient and effective in these domains. To be more precise, let us start by defining $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ to be a graph with a set of nodes $\mathcal{V} = \{1, \ldots, N\}$, a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and a weight function $\mathcal{W} : \mathcal{E} \to \mathbb{R}$ assigning weights to the edges. We assume $\mathcal{W}(i,j) = w_{ij} \geq 0$ for all $(i,j) \in \mathcal{E}$. We describe the data $\mathbf{x}$ as a graph signal $\mathbf{x} \in \mathbb{R}^N$ where each element $[\mathbf{x}]_n$ is the value assigned to node $n \in \mathcal{V}$. To describe the interaction between the data $\mathbf{x}$ and the underlying graph support $\mathcal{G}$, we introduce the concept of graph shift operator (GSO) $\mathbf{S} \in \mathbb{R}^{N \times N}$. The GSO is any matrix that respects the sparsity of the graph, that is, any matrix $\mathbf{S}$ that satisfies that $[\mathbf{S}]_{ij}$ can be nonzero if and only if $(j,i) \in \mathcal{E}$ or $i = j$. This forces $\mathbf{x}' = \mathbf{S}\mathbf{x}$ to be a local operation, where each element $[\mathbf{x}']_n$ can be computed by interacting only with the 1-hop neighborhood of n. Examples of GSOs found in the literature include the adjacency matrix [4, 5], the graph Laplacian [6], the transition matrix of a Markov chain [20], and several normalized versions of these graph operators [9, 14].

In this context, graph neural networks can be defined as layered information processing architectures, where each layer $f^{(\ell)}$ is as follows [cf. (1)]

$$\mathbf{x}^{(\ell)} = \sigma^{(\ell)}(\mathbf{A}^{(\ell)}(\mathbf{S})\mathbf{x}^{(\ell-1)}) \quad (2)$$

thus regularizing $\mathbf{A}^{(\ell)}$ to explicitly be a function of $\mathbf{S}$. This can be readily achieved by the use of graph filters [8, 9], see [10] for details. In what follows, we introduce aggregation GNNs, where we construct an alternative lossless representation of the data $\mathbf{x}$ such that applying regular CNNs results in a GNN as in (2).

In regular CNNs, the nonlinear operation $\sigma^{(\ell)}$ is expanded to include a *pooling* operation, which acts as a summarizing function of information in a hierarchical fashion. This operation heavily depends on the structure of the data as well, so that, technically, $\sigma^{(\ell)}(\cdot) = \sigma^{(\ell)}(\cdot; \mathbf{S})$. In selection GNNs, this can be achieved by means of graph coarsening [9] or zero-padding [10].

3. AGGREGATION GNN

Let us start by defining matrix $\mathbf{X} \in \mathbb{R}^{N \times N}$ comprised of all shifted versions of the signal

$$\mathbf{X} = [\mathbf{x}, \mathbf{S}\mathbf{x}, \ldots, \mathbf{S}^{N-1}\mathbf{x}] = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N] \quad (3)$$

where each column of $\mathbf{X}$ is given by $\mathbf{y}_n = \mathbf{S}^{n-1}\mathbf{x} = \mathbf{y}_n(\mathbf{S})$, for $n = 1, \ldots, N$. We note that $\mathbf{y}_n(\mathbf{S})$ is a function of $\mathbf{S}$. Matrix $\mathbf{X}$ is a redundant representation of data $\mathbf{x}$ since, for any connected graph, any row of $\mathbf{X}$ is sufficient to recover $\mathbf{x}$ as each row contains N linear combinations of $\mathbf{x}$ [21]. We thus note that any such row has successfully incorporated the graph structure included in the powers of the graph shift operator $\mathbf{S}$.

In particular, we focus on a single node in the network $p \in \mathcal{V}$ and observe the aggregated signal $\mathbf{z} = \mathbf{z}(p, \mathbf{S}) \in \mathbb{R}^N$ at this node:

$$\begin{aligned} \mathbf{z} = \mathbf{z}(p, \mathbf{S}) &= \Big[[\mathbf{x}]_p, [\mathbf{S}\mathbf{x}]_p, \ldots, [\mathbf{S}^{N-1}\mathbf{x}]_p \Big]^{\mathsf{T}} \\ &= \Big[[\mathbf{y}_1(\mathbf{S})]_p, [\mathbf{y}_2(\mathbf{S})]_p, \ldots, [\mathbf{y}_N(\mathbf{S})]_p \Big]^{\mathsf{T}}. \end{aligned} \quad (4)$$

First, we note that the representation $\mathbf{z}$ contains information about the underlying structure included in $\mathbf{S}$. Second, is clear that we can recover $\mathbf{x}$ from $\mathbf{z}$ (as long as the graph is connected), and therefore it is a lossless representation. Third, we note that it can be obtained in an entirely local fashion, by successive exchanges with neighboring nodes. Finally, and most importantly for what follows, we observe that $\mathbf{z}$ exhibits a regular structure akin to that of discrete-time signals. More precisely, contiguous elements in vector $\mathbf{z}$ represent nearby neighborhoods in the graph.

It is indeed this regular structure of vector $\mathbf{z}$ that can be exploited to readily process the information in the graph signal $\mathbf{x}$. Let $\mathbf{h} = [h_0, \ldots, h_{K-1}]^{\mathsf{T}} \in \mathbb{R}^K$ be a filter of size K. Then, the regular convolution between $\mathbf{h}$ and $\mathbf{z}$ is defined as

$$[\mathbf{u}]_n = [(\mathbf{h} * \mathbf{z})]_n = \sum_{k=0}^{K-1} [\mathbf{h}]_k [\mathbf{z}]_{n-k} \quad (5)$$

where we deal with border effects by zero-padding $\mathbf{z}$ and keeping $[\mathbf{u}]_n$ for $n = 1, \ldots, N$. Now, when we replace $\mathbf{z} = \mathbf{z}(p, \mathbf{S})$ by its definition (4), the regular convolution (5) becomes

$$[\mathbf{u}]_n = \sum_{k=0}^{K-1} h_k [\mathbf{z}(p, \mathbf{S})]_{n-k} = \sum_{k=0}^{K-1} h_k [\mathbf{S}^{n-k-1}\mathbf{x}]_p. \quad (6)$$

Thus, $[\mathbf{u}]_n$ is the result of a linear operation that relates neighboring values of the graph signal $\mathbf{x}$. In this way, we have effectively regularized the linear operation by taking into account the underlying graph structure of the data, $\mathbf{u} = \mathbf{A}(\mathbf{S})\mathbf{x}$ [cf. (2)].

Each element of $\mathbf{u}$ contains information of K successive neighborhoods of the signal around p. The application of a pointwise nonlinearity $\sigma(\cdot)$ on $\mathbf{u}$ is straightforward on each element $[\mathbf{u}]_n$. At the pooling stage, we apply regular pooling and downsampling, reducing the size of $\mathbf{u}$ from N to $N' \leq N$. We note that, since contiguous elements of $\mathbf{u}$ correspond to related neighborhoods, regular pooling is satisfactorily summarizing

neighboring information, thus the output of the pooling stage can be written as $\mathbf{z}' = \sigma(\mathbf{u}; \mathbf{S})$.

Typically, we want to obtain F features that describe different aspects of the aggregated input signal $\mathbf{z}$. In order to do this, we use a bank of filters, and compute the convolution with each of them [cf. (5)]. Also, we reduce the dimensionality of the vectors by means of pooling so that, typically, after convolution and pooling, we obtain F vectors $\mathbf{z}'_f$ of size N' each, $f = 1, \ldots, F$. We also observe that successive application of these operations leads to a multi-resolution analysis of the information contained in the graph signal, by pooling further away neighborhoods.

The general description at layer $\ell \in \{1, \ldots, L\}$ of the aggregation GNN at node $p \in \mathcal{V}$ is as follows. We add a superscript (ℓ) to denote values at layer ℓ and a subscript f or g to keep track of the feature (the signal) being processed. At any given layer ℓ, we have, at selected node $p \in \mathcal{V}$, $F^{(\ell-1)}$ input features $\mathbf{z}_g^{(\ell-1)} \in \mathbb{R}^{N^{(\ell-1)}}$, $g = 1, \ldots, F^{(\ell-1)}$, and we want to obtain $F^{(\ell)}$ features as the output of layer ℓ. Towards this, we filter each input feature $\mathbf{z}_g^{(\ell-1)}$ with $F^{(\ell)}$ (different) filters of length $K^{(\ell)}$, whose filter taps are $\mathbf{h}_{fg}^{(\ell)}$ for $f = 1, \ldots, F^{(\ell)}$ [cf. (5)]. Then, we sum all the $F^{(\ell-1)}$ outputs obtained for each of the $F^{(\ell)}$ filters (output of the tensor-convolution operation)

$$\mathbf{u}_f^{(\ell)} = \sum_{g=1}^{F^{(\ell-1)}} (\mathbf{h}_{fg}^{(\ell)} * \mathbf{z}_g^{(\ell)}) \tag{7}$$

to yield $F^{(\ell)}$ features $\mathbf{u}_f^{(\ell)}$, $f = 1, \ldots, F^{(\ell)}$. Then, a pointwise nonlinearity followed by regular pooling, are applied, summarizing neighboring information into features $\mathbf{z}_f^{(\ell)} = \sigma(\mathbf{u}_f^{(\ell)}; \mathbf{S})$ of dimension $N^{(\ell)} \leq N^{(\ell-1)}$. We set $\mathbf{z}_1^{(0)} = \mathbf{z}(p, \mathbf{S})$ since $F^{(0)} = 1$. We also note that node $p \in \mathcal{V}$ can learn appropriate filter taps $\mathbf{h}_{fg}^{(\ell)}$ by optimizing a given loss function over the training set.

While aggregation GNNs are easy to implement, entirely local and can be computed at a single node p, they suffer from scalability issues when applied to large networks. We need to aggregate data up to the $(N-1)$-hop neighborhood in order to obtain a lossless representation $\mathbf{z}(p, \mathbf{S})$ of $\mathbf{x}$; and the communication cost of these neighborhood exchanges for large N might be prohibitive. Also, numerical instabilities are likely to arise from large powers of $\mathbf{S}$ that are required in large networks. Nevertheless, we might still be able to process partial information from a subgraph of the network around the selected node p.

4. MULTI-NODE AGGREGATION GNN

In order to address the scalability issues of (single-node) aggregation GNNs described in Sec. 3 we introduce multi-node aggregation GNNs next. Let $\mathcal{P} \subseteq \mathcal{V}$ be a subset of $P = |\mathcal{P}|$ nodes, and let $Q \geq 0$ represent a given number of neighbor exchanges. Consider a submatrix of $\mathbf{X}$ in (3) containing P rows corresponding to nodes $p \in \mathcal{P}$ and Q consecutive columns. Using a slight abuse of notation, each row $\mathbf{z}(p)^{\mathsf{T}}$ of this new submatrix can be described as [cf. (4)]

$$\mathbf{z}(p) = \mathbf{z}(p, Q, \mathbf{S}) = \Big[[\mathbf{x}]_p, [\mathbf{S}\mathbf{x}]_p, \ldots, [\mathbf{S}^{Q-1}\mathbf{x}]_p\Big]^{\mathsf{T}} \tag{8}$$

for $p \in \mathcal{P} \subseteq \mathcal{V}$ and $Q \leq N$. This vector $\mathbf{z}(p) \in \mathbb{R}^Q$ exhibits a regular structure that takes into account the underlying graph support, much like (4). Therefore, regular convolution, pointwise nonlinearities, and regular pooling can be readily (and repeatedly) applied at each node to obtain $F^{(L)}$ descriptive features of the information gathered at each node. More precisely, $\mathbf{z}_f^{(L)}(p)$ is a $N^{(L)}$-dimensional vector representing feature f for $f = 1, \ldots, F^{(L)}$, obtained at node $p \in \mathcal{V}$ as follows

$$\mathbf{z}_f^{(\ell)}(p) = \sigma^{(\ell)}\left(\sum_{g=1}^{F^{(\ell-1)}} \Big(\mathbf{h}_{fg}^{(\ell)}(p) * \mathbf{z}_g^{(\ell)}(p)\Big); \mathbf{S}\right) \tag{9}$$

for $\ell = 1, \ldots, L$, and where $\mathbf{z}_1^{(0)}(p) = \mathbf{z}(p, Q, \mathbf{S})$ [cf. (8)].

At this point, we have successfully managed to obtain $F^{(L)}$ descriptive features of the $(Q-1)$-hop neighborhood surrounding each of the $p \in \mathcal{P}$ selected nodes. In order to keep processing the data and aggregate information at larger neighborhoods, we need to share these features. This can be achieved by considering a signal $\mathbf{x}'_f(m)$ containing $[\mathbf{z}_f^{(L)}(p)]_m$ the m-th component of feature f at each of the $p \in \mathcal{P}$ nodes, and zero in all the non-selected nodes, for each of the $m = 1, \ldots, N^{(L)}$ components. Consequently, the zero-padded signal $\mathbf{x}'_f(m)$ can be written as

$$[\mathbf{x}'_f(m)]_n = \begin{cases} [\mathbf{z}_f^{(L)}(n)]_m & \text{if } n \in \mathcal{P} \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Since signal $\mathbf{x}'_f(m) = \mathbf{x}'_f$ is appropriately defined on the graph $\mathbf{S}$ for every $m = 1, \ldots, N^{(L)}$, we can readily share feature f by means of Q' local exchanges [cf. (8)]

$$\mathbf{z}'_f(p) = \mathbf{z}'_f(p, Q', \mathbf{S}) = \Big[[\mathbf{x}'_f]_p, [\mathbf{S}\mathbf{x}'_f]_p, \ldots, [\mathbf{S}^{Q'-1}\mathbf{x}'_f]_p\Big]^{\mathsf{T}} \tag{11}$$

for $p \in \mathcal{P}'$. Typically, we have $Q' \geq Q$ and $\mathcal{P}' \subseteq \mathcal{P}$ in order to aggregate further away features at a smaller subset of nodes.

We term *outer layers* the local exchanges happening between nodes [cf. (8) and (11)], and *inner layers* the regular convolution, nonlinearity and pooling operations applied within each node [cf. (9)]. The general description of the multi-node aggregation GNN is as follows. Let us start with outer layer r, by having $F^{(r-1)}$ signal features $\mathbf{x}_g^{(r-1)}$ defined at $\mathcal{P}^{(r-1)}$ nodes and zero-padded at nodes $\mathcal{V} \backslash \mathcal{P}^{(r-1)}$, $g = 1, \ldots, F^{(r-1)}$ [cf. (10)]. We then select a subset $\mathcal{P}^{(r)} \subseteq \mathcal{P}^{(r-1)}$ of nodes and carry out $Q^{(r)}$ local exchanges to obtain $\mathbf{z}_f(p, Q^{(r)}, \mathbf{S})$ for $p \in \mathcal{P}^{(r)}$ [cf. (11)]. Then, on the aggregated signal $\mathbf{z}_f(p, Q^{(r)}, \mathbf{S})$, we perform, at each of these nodes $p \in \mathcal{P}^{(r)}$, a number of $L^{(r)}$ inner layers [cf. (9) with $\ell = 1, \ldots, L^{(r)}$] in order to obtain $F^{(r)}$ features at each of the $\mathcal{P}^{(r)}$ nodes. We repeat this procedure for $r = 1, \ldots, R$.

The multi-node aggregation GNN addresses the scalability issues of the single-node by acting as a decentralized method for constructing regional features. The computational cost at each outer layer r is $O(\sum_{p=1}^{P^{(r)}} \sum_{\ell=1}^{L^{(r)}} N^{(\ell-1)} K^{(\ell)} F^{(\ell-1)} F^{(\ell)})$ and the number of parameters to learn is $O(\sum_{p=1}^{P^{(r)}} \sum_{\ell=1}^{L^{(r)}} K^{(\ell)} F^{(\ell)} F^{(\ell-1)})$, independent of the size of the graph.

5. NUMERICAL EXPERIMENTS

In this section, we test the proposed single-node and multi-node aggregation GNNs on the problems of source localization on a synthetic SBM network [22], and on authorship attribution [23]. We select the nodes following three different strategies: those with the highest degree, or experimentally designed sampling (EDS) scores [24], or spectral proxies (SP) [25]. In all architectures, we consider max-pooling summarizing functions and ReLU activation functions for the corresponding GNN layers; and the last layer is a fully-connected readout layer, followed by a softmax, to perform classification.

Unless otherwise specified, all GNNs were trained using the ADAM optimizer [26] with learning rate 0.001 and forgetting factors $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The training phase is carried out for 40 epochs with batches of 100 training samples. The loss function considered in all cases is the cross-entropy loss between one-hot target vectors and the output from the last layer of each architecture.

Source localization. Let $\mathcal{G}$ be a 120-node graph with 3 communities of 40 nodes each [22]. The probability of two nodes being connected within the same community is 0.8 and between different communities is 0.2. In the source localization problem, we observe a diffusion process after some unknown time t, that originated at some unknown node i, $\mathbf{x} = \mathbf{W}^t \boldsymbol{\delta}_i$, for $\mathbf{W}$ the adjacency matrix, and $\boldsymbol{\delta}_i$ a signal with all zeros and a 1 on node i. The objective is to determine which community the source node i belongs to. We generate a training sample of size $10,000$ by randomly selecting the origin i from a pool of $C = 3$ nodes (the highest-degree node of each community; recall that all nodes have, on average, the same degree) and randomly selecting the diffusion time $t < 25$, as well. Analogously, we generate a test set of 200 different samples. We repeat this experiment for 10 different random graphs, and for each graph we consider 10 realizations of the training and test set. The GSO is set to be $\mathbf{S} = \mathbf{W}/\lambda_{\max}(\mathbf{W})$ the adjacency matrix normalized by the largest eigenvalue to avoid numerical instabilities.

The single-node aggregation GNN processes the data through a $L = 2$ layer architecture, with regular convolutions of $K^{(1)} = 4$ and $K^{(2)} = 8$ filter taps, constructing $F^{(1)} = 16$ and $F^{(2)} = 32$ features, respectively. The downsampling on the pooling stage is by a factor of 2 on each case. In the multi-node aggregation GNN we have $R = 2$ outer layers where we select the $P^{(1)} = 30$ and $P^{(2)} = 10$ best nodes, respectively. We carry out $Q^{(1)} = Q^{(2)} = 5$ neighbor exchanges at each node. There are $L^{(1)} = L^{(2)} = 2$ inner layers, using filters of size 3 to compute 16 and 32 features, and downsampling by a factor of 2 on each regular pooling stage. The baseline graph coarsening architecture [9] consists of 2 layers with graph filters of size 5 that compute 32 features. After each layer, the graph is coarsened to be half of its size. Results shown in Table 1 (second column) indicate that the multi-node aggregation GNN with nodes selected by SP outperforms all other architectures.

Authorship attribution. Consider the problem of authorship attribution where the main task is to determine if a given text was written by a certain author. We construct author profiles by means of word adjacency networks (WANs). This WAN acts as the underlying graph support for the graph signal representing the word count (bag-of-words) of the target text of unknown authorship, see [23] for a detailed construction of WANs. In particular, we consider all works by Jane Austen. We construct a WAN with 188 nodes (functional words) using 617 of her texts. We build a training set consisting of these texts, together with 617 texts of other contemporary authors. For the test set, we consider 308 text excerpts, 154 being authored by Jane Austen and 154 written by others. The adopted GSO is the adjacency matrix of the corresponding WAN normalized so that each row adds up to 1, and then symmetrized.

Architecture	Source localization	Authorship
A-Degree	$92.7(\pm 3.5)\%$	$77.7(\pm 2.1)$
A-EDS	$87.2(\pm 6.1)\%$	$68.8(\pm 3.9)$
A-SP	$87.6(\pm 8.4)\%$	$74.6(\pm 3.5)$
MN-Degree	$89.7(\pm 6.4)\%$	$84.5(\pm 1.7)$
MN-EDS	$90.9(\pm 2.7)\%$	$83.2(\pm 2.7)$
MN-SP	$\mathbf{94.4}(\pm \mathbf{3.2})\%$	$\mathbf{88.4}(\pm \mathbf{1.5})$
C	$84.6(\pm 2.5)\%$	$75.6(\pm 3.4)$

Table 1: Accuracy on the test set for each of the problems considered. (A) single-node and (MN) multi-node aggregation GNNs, and (C) graph coarsening.

The single-node aggregation GNN model is as follows: $L = 3$ layers, $K^{(1)} = 6$ and $K^{(2)} = K^{(3)} = 4$ filter taps, $F^{(1)} = 32$, $F^{(2)} = 64$ and $F^{(3)} = 128$ features, and downsampling by 4 in the first layer and 2 in the next ones. In the multi-node aggregation GNN we have: $R = 2$ outer layers, $P^{(1)} = 30$ and $P^{(2)} = 10$ selected nodes, and $Q^{(1)} = Q^{(2)} = 5$ neighbor exchanges at each node. There are $L^{(1)} = L^{(2)} = 2$ inner layers, using filters of size 3 to compute 16 and 32 features, and downsampling by a factor of 2 on each regular pooling stage. The baseline graph coarsening architecture consists of 2 layers, graph filters of size 5 that compute 32 features, and graph coarsening by a factor of 2. Training is carried out for 80 epochs.

Table 1 (third column) shows that multi-node GNNs following any of the selection criteria outperforms the baseline, with SP being the best.

6. CONCLUSIONS

In this paper we proposed a novel GNN that acts on graph signals. Aggregation GNNs collect all the information at a single node by means of successive local exchanges with neighbors. The resulting aggregated signal exhibits a regular structure while incorporating information of the underlying graph structure as well. Given this regularity of the aggregated signal we can effectively apply regular convolution and regular pooling, obtaining linear combinations of neighboring values and multi-resolution analysis, respectively. To address some scalability issues we introduced multi-node aggregation GNNs where regional information is collected at a subset of nodes and then shared among these nodes to obtain descriptions of larger regions. Tests run on a synthetic source localization problem and on authorship attribution show better performance over the baseline.

7. REFERENCES

[1] 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.

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.

[8] 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.

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

[10] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro, "Convolutional neural network architectures for signals supported on graphs," *arXiv:1805.00165v1 [eess.SP]*, 1 May 2018.

[11] 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.

[12] F. Gama, G. Leus, A. G. Marques, and A. Ribeiro, "Convolutional neural networks via node-varying graph filters," in *2018 IEEE Data Sci. Workshop*, Lausanne, Switzerland, 4-6 June 2018, IEEE.

[13] F. Gama, A. G. Marques, A. Ribeiro, and G. Leus, "MIMO graph filters for convolutional networks," in *19th IEEE Int. Workshop Signal Process. Advances in Wireless Commun.*, Kalamata, Greece, 25-28 June 2018, IEEE.

[14] 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.

[15] 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.

[16] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *arXiv:1710.10903v3 [stat.ML]*, 4 Feb. 2018.

[17] D. Owerko, F. Gama, and A. Ribeiro, "Predicting power outages using graph neural networks," in *IEEE Global Conf. Signal and Inform. Process. 2018*, Anaheim, CA, 26-29 Nov. 2018, IEEE.

[18] W. Huang, A. G. Marques, and A. Ribeiro, "Rating prediction via graph signal processing," *IEEE Trans. Signal Process.*, vol. 66, no. 19, pp. 5066–5081, Oct. 2018.

[19] F. Monti, M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *31st Annu. Conf. Neural Inform. Process. Syst.*, Long Beach, CA, 4-9 Dec. 2017, NIPS Foundation.

[20] A. Heimowitz and Y. C. Eldar, "A unified view of diffusion maps and signal processing on graphs," in *2017 Int. Conf. Sampling Theory and Appl.*, Tallin, Estonia, 3-7 July 2017, IEEE.

[21] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Sampling of graph signals with successive local aggregations," *IEEE Trans. Signal Process.*, vol. 64, no. 7, pp. 1832–1843, Apr. 2016.

[22] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová, "Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications," *Physical Review E*, vol. 84, no. 6, pp. 066106, Dec. 2011.

[23] 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, Oct. 2015.

[24] R. Varma, S. Chen, and J. Kovačević, "Spectrum-blind signal recovery on graphs," in *2015 IEEE Int. Workshop Comput. Advances Multi-Sensor Adaptive Process.*, Cancún, México, 13-16 Dec. 2015, pp. 81–84, IEEE.

[25] A. Anis, A. Gadde, and A. Ortega, "Efficient sampling set selection for bandlimited graph signals using graph spectral proxies," *IEEE Trans. Signal Process.*, vol. 64, no. 14, pp. 3775–3789, July 2016.

[26] 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.