Quantization in Graph Convolutional Neural Networks

Leila Ben Saad and Baltasar Beferull-Lozano
WISENET Center, Department of ICT, University of Agder, Grimstad, Norway

***Abstract*—By replacing classical convolutions with graph filters, graph convolutional neural networks (GNNs) have emerged as powerful tools to learn a nonlinear mapping for data defined over graphs and address a variety of tasks encountered in many applications. GNNs inherit the distributed implementation of graph filters, where local exchanges among neighbor nodes are performed. In such distributed setting, the quantization can play a fundamental role to save communication and energy resources prior to data transmission, in scenarios where nodes are resource constrained. In this paper, we propose a quantized GNN architecture based on distributed graph filters for signals defined on graphs and analyze how the quantization noise can affect its performance. We show also that the expected error due to quantization at the GNN output is upper-bounded and the use of a decreasing quantization stepsize leads to more accuracy. The performance of the proposed schemes is evaluated by numerical experiments for the application of source localization.**

***Index Terms*—Graph neural networks; Graph signal processing; Graph filters; Quantization.**

I. Introduction

Recently, graph convolutional neural networks (GNNs) [1–3] have emerged as a way to generalize and extend the convolutional neural networks (CNNs) to data supported on graphs by processing signals defined in irregular domains and replacing classical convolutions with graph filters [4–6]. GNNs offer the possibility to learn a nonlinear mapping for data defined over graphs that can be encountered in many applications, such as social networks, sensor networks and recommendation systems. GNNs are formed by layers of graph filters followed by a pointwise nonlinearity. By using Finite Impulse Response (FIR) graph filters [4–6], GNNs inherit their distributed implementation, which is very important to ensure the scalability and robustness to possible node failures. FIR graph filters can have different forms of implementation [5, 6]: node-invariant, node-variant and edge-variant. In such graph filters, each node can communicate its input signal through local exchanges with neighbors in a finite number of iterations. However, when implemented over distributed networks with constrained node resources, the graph filters in GNNs have to deal with the limited energy, processing, and communication capabilities. The quantization is recognized as an effective approach to save communication and energy resources prior to data transmission. Although the quantization has been well studied in neural networks [7, 8] and CNNs [9–11], very few works [12, 13] have explored the quantization problem in graph neural networks. In [12], a method for training a quantized GNN is proposed, while in [13] a GNN quantization algorithm and a fine-tuning scheme are proposed to solve the GNN memory problem. To the best of our knowledge, there is no work investigating the quantization for GNNs via distributed graph filters for signals supported on graphs. In this paper, we propose a quantized GNN architecture built on distributed graph filters and analyze how the quantization errors accumulated over the different layers can affect its final output. Considering the different forms of FIR graph filters implementation, we show how the expected error due to quantization at the GNN output is upper-bounded and the use of a decreasing quantization stepsize leads to more accuracy as compared to a fixed quantization stepsize. Numerical experiments have been conducted to corroborate our findings.

The remainder of this paper is organized as follows. Section II presents the background material. Section III introduces the quantization in GNNs for signals supported on graphs and Section IV analyzes its impacts. Section V presents the numerical results. Section VI concludes the paper.

Notation: Vectors (respectively matrices) are denoted by bold lowercase (uppercase) letters. We denote by $\|\mathbf{v}\|$ the l_2-norm of vector $\mathbf{v}$ whereas by $\|\mathbf{M}\|$ the spectral norm of matrix $\mathbf{M}$. We indicate by $\text{tr}(\cdot)$, $\text{diag}(\cdot)$ and $\mathbf{\Sigma_x}$ the trace operator, the diagonal matrix and the covariance matrix of a random process $\mathbf{x}$, respectively.

II. Background

Consider an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{1, \ldots, N\}$ is the set of N vertices and $\mathcal{E}$ is the set of edges such that if there is a link from node j to node i, then $(j, i) \in \mathcal{E}$. The structure of $\mathcal{G}$ is generally represented by the graph-shift operator $\mathbf{S}$, which is an $N \times N$ matrix such that its entries are non-zero only if $i = j$ or if $(j, i) \in \mathcal{E}$. Common choices of the graph-shift operator are the graph adjacency matrix $\mathbf{A}$, the graph Laplacian $\mathbf{L}$ or their normalized versions. We define on the vertices of $\mathcal{G}$ a graph signal as a mapping $\mathbf{x} : \mathcal{V} \to \mathbb{R}$, which can be represented as a vector $\mathbf{x} = [x_1, ..., x_N]^\top \in \mathbb{R}^N$, where the i-th entry represents the signal value at node i.

FIR graph filters. A linear graph filter (GF) [4] $\mathbf{H}(\mathbf{S}) : \mathbb{R}^N \to \mathbb{R}^N$ is a function of the shift operator $\mathbf{S}$, and where a graph signal $\mathbf{x}$ is taken as an input and another graph signal $\mathbf{y} = \mathbf{H}(\mathbf{S})\mathbf{x}$ is produced as an output. One common form for $\mathbf{H}(\mathbf{S})$ is the so-called node-invariant GF, which is a polynomial in $\mathbf{S}$ with output:

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

where $h_0, \ldots, h_K$ are the scalar filter coefficients.

To approximate a broader class of operations, the node-variant GF [6] has been proposed and has as output:

$$\mathbf{y} = \mathbf{H}(\mathbf{S})\mathbf{x} = \sum_{k=0}^{K} \text{diag}(\mathbf{h}^{(k)})\, \mathbf{S}^k\, \mathbf{x} \tag{2}$$

where each node uses different filter coefficients collected in an $N \times 1$ vector $\mathbf{h}^{(k)} = [h_1^{(l)}, ..., h_N^{(k)}]^\top$.

This work was supported by the PETROMAKS Smart-Rig grant 244205/E30, the TOPPFORSK WISECART grant 250910/F20 and the IKTPLUSS INDURB grant 270730/O70 from the Research Council of Norway.

To enable the use of a different set of weights in each shift, the edge-variant GF [5] has been proposed. The latter is an extended version of node-variant GF and has as output:

$$\mathbf{y} = \mathbf{H}(\mathbf{S})\mathbf{x} = \sum_{k=1}^{K} \big(\prod_{\kappa=1}^{k} \boldsymbol{\Psi}_{\kappa} \big)\, \mathbf{x} \tag{3}$$

where $\boldsymbol{\Psi}_{\kappa}$ is $N \times N$ edge weighting matrix and $\boldsymbol{\Psi}_1 \cdots \boldsymbol{\Psi}_K$ are a collection of K matrices sharing the sparsity pattern of $\mathbf{I}_N + \mathbf{S}$.

Quantization. The quantization operation encodes data (before communication) in a certain number of bits to reduce the amount of data to be transmitted. However, in practice, the quantization introduces always quantization errors that affect the graph filtering output. At the k-th graph filter shift, the quantized message is given by $\tilde{\mathbf{x}}^{(k)} = Q(\mathbf{x}^{(k)}) = \mathbf{x}^{(k)} + \boldsymbol{\epsilon}^{(k)}$, where $\boldsymbol{\epsilon}^{(k)}$ is the quantization error and $\mathbf{x}^{(k)}$ is the unquantized input signal. In order to control the quantization noise and ensure the uniform random variable assumption and uncorrelation for the quantization errors, we consider substractive dithering quantization [14, 15]. The latter consists of adding a pseudo-random generator, called dither, to the input signal prior to quantization at the transmitter node, and then subtracting it at the receiver node after transmission. This technique will guarantee that the quantization error $[\boldsymbol{\epsilon}^{(k)}]_j$ at each node j at iteration k, is uniformly distributed over $[-\Delta_k/2\ , \Delta_k/2]$ with a zero mean and variance $(\sigma^{(k)})^2 = \Delta_k^2/12$. Here, Δ_k denotes the quantization stepsize at iteration k. At the filter initialization, the unquantized input signal is $\mathbf{x}^{(0)} = \mathbf{x}$. After quantization, it becomes $\tilde{\mathbf{x}}^{(0)} = \mathbf{x}^{(0)} + \boldsymbol{\epsilon}^{(0)}$. This quantized signal is then exchanged with the neighbor nodes and the resulting quantized shifted signal has the expression $\mathbf{x}^{(1)} = \mathbf{S}\tilde{\mathbf{x}}^{(0)} = \mathbf{S}(\mathbf{x}^{(0)} + \boldsymbol{\epsilon}^{(0)})$. The signal $\mathbf{x}^{(1)}$ is further quantized into $\tilde{\mathbf{x}}^{(1)}$ and subsequently transmitted to the neighbor nodes. The process is repeated k times, where the output of the shifted signal for a general iteration k is $\mathbf{x}^{(k)} = \mathbf{S}\tilde{\mathbf{x}}^{(k-1)} = \mathbf{S}(\mathbf{x}^{(k-1)} + \boldsymbol{\epsilon}^{(k-1)})$.

Graph Convolutional neural networks. GNNs [1–3] are composed of a cascade of L layers. Each layer performs a graph convolution followed by a pointwise nonlinearity (e.g, ReLu). Graph convolutions can be exactly modeled by graph filters in the node-invariant [cf. (1)], node-variant [cf. (2)] and edge-variant [cf. (3)] forms, depending on the architecture. Moreover, the descriptive power of GNNs can be increased by incorporating multiple parallel features per layer. These features are the result of processing multiple input features with a parallel bank of graph filters. At layer ℓ, the GNN has as input $F_{\ell-1}$ features $\{\mathbf{x}_{\ell-1}^{g}\}_{g=1}^{F_{\ell-1}}$ from layer $(\ell-1)$ and returns F_ℓ output features $\{\mathbf{x}_{\ell}^{f}\}_{f=1}^{F_\ell}$. Each input feature $\mathbf{x}_{\ell-1}^{g}$ for $g = 1 \cdots F_{\ell-1}$ is processed in parallel by F_ℓ graph filters $\{\mathbf{H}_{\ell}^{fg}(\mathbf{S})\}_f$ of the form:

$$\mathbf{H}_{\ell}^{fg}(\mathbf{S}) = \begin{cases} \sum_{k=0}^{K} h_{k\ell}^{fg}\mathbf{S}^k & \text{for node-invariant} \\ \sum_{k=0}^{K} \operatorname{diag}(\mathbf{h}_{k\ell}^{fg})\mathbf{S}^k & \text{for node-variant} \\ \sum_{k=1}^{K} \big(\prod_{\kappa=1}^{k} \boldsymbol{\Psi}_{\kappa\ell}^{fg}\big) & \text{for edge-variant} \end{cases} \tag{4}$$

The filter outputs are aggregated over the input index g to generate the fth convolved feature:

$$\mathbf{z}_{\ell}^{f} = \sum_{g=1}^{F_{\ell-1}} \mathbf{H}_{\ell}^{fg}(\mathbf{S})\mathbf{x}_{\ell-1}^{g} \ \text{ for } f = 1 \cdots F_\ell \tag{5}$$

Then, the convolved feature $\mathbf{z}_{\ell}^{f}$ goes through an activation function $\sigma(\cdot)$ to lead to the fth convolutional layer output:

$$\mathbf{x}_{\ell}^{f} = \sigma(\mathbf{z}_{\ell}^{f}) = \sigma\left(\sum_{g=1}^{F_{\ell-1}} \mathbf{H}_{\ell}^{fg}(\mathbf{S})\mathbf{x}_{\ell-1}^{g}\right) \ \text{ for } f = 1 \cdots F_\ell \tag{6}$$

At layer $\ell = 1$, the input feature is the graph signal $\mathbf{x}_0^1 = \mathbf{x}$. At the last layer L, we assume without loss of generality that the number of output features is $F_L = 1$. This last and single output feature $\mathbf{x}_L^1 = \mathbf{x}_L$ represents the output of the GNN.

The GNN output is a function of the input signal $\mathbf{x}$ and the collection of filter banks $\mathbf{H}_{l}^{fg}$ [cf. (4)]. By defining the filter tensor $\mathcal{H}(\mathbf{S}) = \{\mathbf{H}_{l}^{fg}(\mathbf{S})\}_{lfg}$ that regroups the filters of all layers, we can consider the GNN as a map $\boldsymbol{\Phi}(\cdot)$ that has as inputs the graph signal $\mathbf{x}$ and the filter tensor $\mathcal{H}(\mathbf{S})$ and as output:

$$\boldsymbol{\Phi}\big(\mathbf{x}; \mathcal{H}(\mathbf{S})\big) = \mathbf{x}_L \tag{7}$$

Consider a training data set of input-output pairs $\mathcal{T} = (\mathbf{x}, \mathbf{u})$. The filter parameters are trained to minimize over the training set $\mathcal{T}$ a cost function (e.g, mean squared error and L1 loss function). This cost function can assess the difference between the GNN output $\mathbf{x}_L$ and the true value $\mathbf{u}$ averaged over the examples $(\mathbf{x}, \mathbf{u}) \in \mathcal{T}$.

III. Graph convolutional neural networks with quantization

In this section, we propose a quantized GNN architecture based on distributed graph filters for signals supported on graphs. Consider at layer $\ell = 1$, the input feature of GNN is a graph signal $\boldsymbol{\varkappa}_0^1 = \mathbf{x}$. At any layer $\ell > 1$, the GNN has as input $F_{\ell-1}$ features $\{\boldsymbol{\varkappa}_{\ell-1}^{g}\}_{g=1}^{F_{\ell-1}}$ from layer $(\ell-1)$ and returns F_ℓ output features $\{\boldsymbol{\varkappa}_{\ell}^{f}\}_{f=1}^{F_\ell}$. Each input feature $\boldsymbol{\varkappa}_{\ell-1}^{g}$ for $g = 1 \cdots F_{\ell-1}$ is quantized and processed in parallel by F_ℓ graph filters.

For GNN based on a node-invariant graph filter, consider the kth shifted input feature $\boldsymbol{\varkappa}_{\ell-1}^{g(k)} = \mathbf{S}^k\boldsymbol{\varkappa}_{\ell-1}^{g}$ exchanged with the neighbors at layer $\ell-1$. The quantized form of the latter is $\tilde{\boldsymbol{\varkappa}}_{\ell-1}^{g(k)} = Q(\boldsymbol{\varkappa}_{\ell-1}^{g(k)}) = \boldsymbol{\varkappa}_{\ell-1}^{g(k)} + \boldsymbol{\epsilon}_{\ell-1}^{g(k)}$. Initially at layer $\ell-1$, we have $\boldsymbol{\varkappa}_{\ell-1}^{g(0)} = \boldsymbol{\varkappa}_{\ell-1}^{g}$ and its quantized form is $\tilde{\boldsymbol{\varkappa}}_{\ell-1}^{g(0)} = \boldsymbol{\varkappa}_{\ell-1}^{g(0)} + \boldsymbol{\epsilon}_{\ell-1}^{g(0)}$. This quantized signal is exchanged with neighbors leading to the quantized shifted signal $\boldsymbol{\varkappa}_{\ell-1}^{g(1)} = \mathbf{S}\tilde{\boldsymbol{\varkappa}}_{\ell-1}^{g(0)} = \mathbf{S}(\boldsymbol{\varkappa}_{\ell-1}^{g(0)} + \boldsymbol{\epsilon}_{\ell-1}^{g(0)})$. The signal $\boldsymbol{\varkappa}_{\ell-1}^{g(1)}$ is further quantized into $\tilde{\boldsymbol{\varkappa}}_{\ell-1}^{g(1)}$ and subsequently transmitted to the neighbor nodes. This procedure is repeated K times. The output of the shifted graph signal with quantization at iteration k is as follows:

$$\begin{aligned} \boldsymbol{\varkappa}_{\ell-1}^{g(1)} &= \mathbf{S}\tilde{\boldsymbol{\varkappa}}_{\ell-1}^{g(0)} = \mathbf{S}(\boldsymbol{\varkappa}_{\ell-1}^{g(0)} + \boldsymbol{\epsilon}_{\ell-1}^{g(0)}) = \mathbf{S}\boldsymbol{\varkappa}_{\ell-1}^{g(0)} + \mathbf{S}\boldsymbol{\epsilon}_{\ell-1}^{g(0)} \\ \boldsymbol{\varkappa}_{\ell-1}^{g(2)} &= \mathbf{S}\tilde{\boldsymbol{\varkappa}}_{\ell-1}^{g(1)} = \mathbf{S}(\boldsymbol{\varkappa}_{\ell-1}^{g(1)} + \boldsymbol{\epsilon}_{\ell-1}^{g(1)}) = \mathbf{S}^2\boldsymbol{\varkappa}_{\ell-1}^{g(0)} + \mathbf{S}^2\boldsymbol{\epsilon}_{\ell-1}^{g(0)} + \mathbf{S}\boldsymbol{\epsilon}_{\ell-1}^{g(1)} \\ &\ \vdots \\ \boldsymbol{\varkappa}_{\ell-1}^{g(k)} &= \ \mathbf{S}^k\ \boldsymbol{\varkappa}_{\ell-1}^{g(0)} + \sum_{\kappa=0}^{k-1} \mathbf{S}^{k-\kappa}\ \boldsymbol{\epsilon}_{\ell-1}^{g(\kappa)}, \ \ k \geq 1. \end{aligned} \tag{8}$$

From (8), the fth node-invariant graph filter output with quantization at layer ℓ is:

$$\begin{aligned} \tilde{\mathbf{y}}_{\ell}^{fg} &= h_{0\ell}^{fg}\boldsymbol{\varkappa}_{\ell-1}^{g} + h_{1\ell}^{fg}(\mathbf{S}\boldsymbol{\varkappa}_{\ell-1}^{g} + \mathbf{S}\boldsymbol{\epsilon}_{\ell-1}^{g(0)}) \\ &\quad + h_{2\ell}^{fg}(\mathbf{S}^2\boldsymbol{\varkappa}_{\ell-1}^{g} + \mathbf{S}^2\boldsymbol{\epsilon}_{\ell-1}^{g(0)} + \mathbf{S}\boldsymbol{\epsilon}_{\ell-1}^{g(1)}) + \cdots \end{aligned} \tag{9}$$

$$
\begin{aligned}
&+ h_{k\ell}^{fg}\big(\mathbf{S}^{K}\boldsymbol{\varkappa}_{\ell-1}^{g} + \mathbf{S}^{K}\boldsymbol{\epsilon}_{\ell-1}^{g(0)} + \mathbf{S}^{K-1}\boldsymbol{\epsilon}_{\ell-1}^{g(1)} + \cdots \\
&+ \mathbf{S}^{2}\boldsymbol{\epsilon}_{\ell-1}^{g(K-2)} + \mathbf{S}\boldsymbol{\epsilon}_{\ell-1}^{g(K-1)}\big) \\
&= \sum_{k=0}^{K} h_{k\ell}^{fg}\mathbf{S}^{k}\boldsymbol{\varkappa}_{\ell-1}^{g} + \sum_{k=1}^{K} h_{kl}^{fg}\sum_{\kappa=0}^{k-1}\mathbf{S}^{k-\kappa}\boldsymbol{\epsilon}_{\ell-1}^{g(\kappa)} = \sum_{k=0}^{K} h_{k\ell}^{fg}\mathbf{S}^{k}\boldsymbol{\varkappa}_{\ell-1}^{g} + \boldsymbol{\xi}_{\ell-1}^{fg}
\end{aligned}
$$

where $\boldsymbol{\xi}_{\ell-1}^{fg}$ denotes the accumulated shifted quantization errors.

Similarly, we can easily show that the fth node-variant graph filter output and the fth edge-variant graph filter output with quantization at layer ℓ are respectively given by:

$$
\tilde{\mathbf{y}}_{\ell}^{fg} = \sum_{k=0}^{K}\operatorname{diag}(\mathbf{h}_{k\ell}^{fg})\mathbf{S}^{k}\boldsymbol{\varkappa}_{\ell-1}^{g} + \underbrace{\sum_{k=1}^{K}\operatorname{diag}(\mathbf{h}_{kl}^{fg})\sum_{\kappa=0}^{k-1}\mathbf{S}^{k-\kappa}\boldsymbol{\epsilon}_{\ell-1}^{g(\kappa)}}_{\boldsymbol{\xi}_{\ell-1}^{fg}} \tag{10}
$$

$$
\tilde{\mathbf{y}}_{\ell}^{fg} = \sum_{k=1}^{K}\Big(\prod_{\kappa=1}^{k}\boldsymbol{\Psi}_{\kappa\ell}^{fg}\Big)\boldsymbol{\varkappa}_{\ell-1}^{g} + \underbrace{\sum_{k=1}^{K}\sum_{\kappa=0}^{k-1}\Big(\prod_{\tau=\kappa+1}^{k}\boldsymbol{\Psi}_{\tau\ell}^{fg}\Big)\boldsymbol{\epsilon}_{\ell-1}^{g(\kappa)}}_{\boldsymbol{\xi}_{\ell-1}^{fg}} \tag{11}
$$

More specifically, the node-invariant graph filter outputs are aggregated over the input index g to generate the fth convolved feature:

$$
\tilde{\mathbf{z}}_{\ell}^{f} = \sum_{g=1}^{F_{\ell-1}}\left(\sum_{k=0}^{K} h_{k\ell}^{fg}\mathbf{S}^{k}\boldsymbol{\varkappa}_{\ell-1}^{g} + \sum_{k=1}^{K} h_{kl}^{fg}\sum_{\kappa=0}^{k-1}\mathbf{S}^{k-\kappa}\boldsymbol{\epsilon}_{\ell-1}^{g(\kappa)}\right) \quad \text{for } f = 1\cdots F_{\ell} \tag{12}
$$

The node-variant graph filter outputs are aggregated over the input index g to generate the fth convolved feature:

$$
\tilde{\mathbf{z}}_{\ell}^{f} = \sum_{g=1}^{F_{\ell-1}}\left(\sum_{k=0}^{K}\operatorname{diag}(\mathbf{h}_{k\ell}^{fg})\mathbf{S}^{k}\boldsymbol{\varkappa}_{\ell-1}^{g} + \sum_{k=1}^{K}\operatorname{diag}(\mathbf{h}_{kl}^{fg})\sum_{\kappa=0}^{k-1}\mathbf{S}^{k-\kappa}\boldsymbol{\epsilon}_{\ell-1}^{g(\kappa)}\right) \quad \text{for } f = 1\cdots F_{\ell} \tag{13}
$$

The edge-variant graph filter outputs are aggregated over the input index g to generate the fth convolved feature:

$$
\tilde{\mathbf{z}}_{l}^{f} = \sum_{g=1}^{F_{\ell-1}}\left(\sum_{k=1}^{K}\Big(\prod_{\kappa=1}^{k}\boldsymbol{\Psi}_{\kappa\ell}^{fg}\Big)\boldsymbol{\varkappa}_{\ell-1}^{g} + \sum_{k=1}^{K}\sum_{\kappa=0}^{k-1}\Big(\prod_{\tau=\kappa+1}^{k}\boldsymbol{\Psi}_{\tau\ell}^{fg}\Big)\boldsymbol{\epsilon}_{\ell-1}^{g(\kappa)}\right) \quad \text{for } f = 1\cdots F_{\ell} \tag{14}
$$

Independently of which form of graph filter is adopted, the filter outputs are aggregated over the input index g to generate the fth convolved feature:

$$
\tilde{\mathbf{z}}_{\ell}^{f} = \sum_{g=1}^{F_{\ell-1}}\big(\mathbf{H}_{\ell}^{fg}(\mathbf{S})\boldsymbol{\varkappa}_{\ell-1}^{g} + \boldsymbol{\xi}_{\ell-1}^{fg}\big) \quad \text{for } f = 1\cdots F_{\ell} \tag{15}
$$

The quantized convolved feature $\tilde{\mathbf{z}}_{l}^{f}$ goes through an activation function $\sigma(\cdot)$ to obtain the fth convolutional layer output:

$$
\boldsymbol{\varkappa}_{\ell}^{f} = \sigma(\tilde{\mathbf{z}}_{\ell}^{f}) = \sigma\left(\sum_{g=1}^{F_{\ell-1}}(\mathbf{H}_{\ell}^{fg}(\mathbf{S})\boldsymbol{\varkappa}_{\ell-1}^{g} + \boldsymbol{\xi}_{\ell-1}^{fg})\right) \quad \text{for } f = 1\cdots F_{\ell} \tag{16}
$$

The output feature of the last convolutional layer is $\boldsymbol{\varkappa}_{L}$, implying that the GNN output is $\boldsymbol{\varkappa}_{L}$. The latter depends on the input signal $\mathbf{x}$ and the sequence of filter banks with quantization. By defining the filter tensor with quantization $\tilde{\mathcal{H}}(\mathbf{S})$ that regroups the filters with quantization of all layers, we can consider the GNN as a map $\tilde{\boldsymbol{\Phi}}(\cdot)$ that has as inputs the graph signal $\mathbf{x}$ and the filter tensor $\tilde{\mathcal{H}}(\mathbf{S})$ and as output:

$$
\tilde{\boldsymbol{\Phi}}\big(\mathbf{x};\tilde{\mathcal{H}}(\mathbf{S})\big) = \boldsymbol{\varkappa}_{L} \tag{17}
$$

Consider a training data set of input-output pairs $\mathcal{T} = (\mathbf{x},\mathbf{v})$. The filter parameters are trained to minimize a cost function over the training set $\mathcal{T}$. This cost function assesses the difference between the GNN output $\boldsymbol{\varkappa}_{L}$ and the true value $\mathbf{v}$ averaged over the examples $(\mathbf{x},\mathbf{v}) \in \mathcal{T}$.

IV. Quantization effects on Graph Convolutional Neural Networks

In this section, we analyze the quantization effects on the GNN output. We are interested in analyzing the norm of the expected error due to quantization at the GNN output given by:

$$
\begin{aligned}
\big\|\mathbb{E}\big[\boldsymbol{\Phi}(\mathbf{x};\mathcal{H}(\mathbf{S})) - \tilde{\boldsymbol{\Phi}}(\mathbf{x};\tilde{\mathcal{H}}(\mathbf{S}))\big]\big\| &= \|\mathbb{E}\big[\mathbf{x}_{L} - \boldsymbol{\varkappa}_{L}\big]\| \\
&= \big\|\mathbb{E}\big[\sigma(\mathbf{z}_{L}^{f}) - \sigma(\tilde{\mathbf{z}}_{L}^{f})\big]\big\|
\end{aligned} \tag{18}
$$

The following Theorem provides an upper-bound on the norm of the expected error due to the quantization at the GNN output.

Theorem 1. Consider a GNN comprising of a cascade of L layers applying graph filters that can be in the form of node-invariant, node-variant or edge-variant, respectively, such that $|h_{kl}^{fg}| \leq \beta_{\text{inv}}$, $\|\operatorname{diag}(\mathbf{h}_{k\ell}^{fg})\| \leq \beta_{\text{nv}}$ and $\|(\boldsymbol{\Psi}_{\kappa\ell}^{fg})\| \leq \beta_{\text{ev}}$. Consider also a graph-shift operator $\mathbf{S}$ such that $\|\mathbf{S}\| \leq \gamma$. Assume also a pointwise nonlinearity $\sigma(\cdot)$ that is normalized Lipschitz such that there exists a constant C_{σ} so that $|\sigma(b) - \sigma(a)| \leq C_{\sigma}|b - a|$. Consider also that the input signal features are quantized with a uniform quantizer but with a dynamic stepsize Δ_{k} over each k iteration. Then, the norm of the expected error due to quantization at the GNN output is upper-bounded by:

$$
\begin{aligned}
&\big\|\mathbb{E}\big[\boldsymbol{\Phi}(\mathbf{x};\mathcal{H}(\mathbf{S})) - \tilde{\boldsymbol{\Phi}}(\mathbf{x};\tilde{\mathcal{H}}(\mathbf{S}))\big]\big\| \\
&\leq \alpha\left(C_{\sigma}^{L-1}\Big(\prod_{\ell=1}^{L-1}F_{\ell}\Big)\mu^{L-1} + \sum_{i=1}^{L-1}C_{\sigma}^{L-i}\Big(\prod_{\ell=i}^{L-1}F_{\ell}\Big)\,\mu^{L-i-1}\right)
\end{aligned} \tag{19}
$$

where α is the upper-bound of $\mathbb{E}\big[\|\boldsymbol{\xi}_{\ell-1}^{fg}\|\big]$, given by:

$$
\alpha = \begin{cases}
\alpha_{\text{inv}} = \beta_{\text{inv}}\sqrt{\frac{N}{12}}\sum_{k=1}^{K}\sum_{\kappa=0}^{k-1}\gamma^{k-\kappa}\Delta_{\kappa} & \text{for node-invariant} \\
\alpha_{\text{nv}} = \beta_{\text{nv}}\sqrt{\frac{N}{12}}\sum_{k=1}^{K}\sum_{\kappa=0}^{k-1}\gamma^{k-\kappa}\Delta_{\kappa} & \text{for node-variant} \\
\alpha_{\text{ev}} = \sqrt{\frac{N}{12}}\sum_{k=1}^{K}\sum_{\kappa=0}^{k-1}\beta_{\text{ev}}^{k-\kappa}\Delta_{k} & \text{for edge-variant}
\end{cases} \tag{20}
$$

and μ is the upper-bound of $\|\mathbf{H}_{\ell}^{fg}(\mathbf{S})\|$, given by:

$$
\mu = \begin{cases}
\mu_{\text{inv}} = \sum_{k=0}^{K}\beta_{\text{inv}}\gamma^{k} & \text{for node-invariant} \\
\mu_{\text{nv}} = \sum_{k=0}^{K}\beta_{\text{nv}}\gamma^{k} & \text{for node-variant} \\
\mu_{\text{ev}} = \sum_{k=1}^{K}\beta_{\text{ev}}^{k} & \text{for edge-variant}
\end{cases} \tag{21}
$$

Proof: See Appendix I

Theorem 1 shows that the expected error due to quantization at the GNN output is affected by the filter coefficients, the shift operator and the number of layers and input features. In addition, the quantization stepsize Δ_k has a significant impact on the expected error due to quantization through the constant α. To reduce the quantization errors accumulated through the layers of filter banks, we can choose a decreasing quantization stepsize at each iteration k so that α becomes smaller in (19). The following Corollary presents this result.

Corollary 1. Consider the same settings as Theorem 1 with input signal features quantized with a uniform quantizer, where the quantization stepsize Δ_k is decreasing at each iteration k such that (for node-invariant or node-variant) $\Delta_k = \gamma^k \Delta_0$ if $0 < \gamma < 1$ and $\Delta_k = \gamma^{-k}\Delta_0$ if $\gamma > 1$, and such that (for edge-variant) $\Delta_k = \beta_{\text{ev}}^k \Delta_0$ if $0 < \beta_{\text{ev}} < 1$ and $\Delta_k = \beta_{\text{ev}}^{-k}\Delta_0$ if $\beta_{\text{ev}} > 1$. The norm of the expected error due to quantization at the GNN output is upper-bounded by (19) with smaller α given by:

$$\alpha = \begin{cases} \alpha_{\text{inv}} = \Delta_0 \beta_{\text{inv}} \eta_1 \sqrt{\frac{N}{12}} & \text{for node-invariant} \\ \alpha_{\text{nv}} = \Delta_0 \beta_{\text{nv}} \eta_1 \sqrt{\frac{N}{12}} & \text{for node-variant} \\ \alpha_{\text{ev}} = \Delta_0 \eta_2 \sqrt{\frac{N}{12}} & \text{for edge-variant} \end{cases} \tag{22}$$

for $\gamma < 1$ and $\beta_{\text{ev}} < 1$ and where $\eta_1 = \frac{\gamma\left(1-(K+1)\gamma^K + K\gamma^{K+1}\right)}{(1-\gamma)^2}$ and $\eta_2 = \frac{\beta_{\text{ev}}\left(1-(K+1){\beta_{\text{ev}}}^K + K{\beta_{\text{ev}}}^{K+1}\right)}{(1-\beta_{\text{ev}})^2}$.

$$\alpha = \begin{cases} \alpha_{\text{inv}} = \Delta_0 \beta_{\text{inv}} \varsigma_1 \sqrt{\frac{N}{12}} & \text{for node-invariant} \\ \alpha_{\text{nv}} = \Delta_0 \beta_{\text{nv}} \varsigma_1 \sqrt{\frac{N}{12}} & \text{for node-variant} \\ \alpha_{\text{ev}} = \Delta_0 \varsigma_2 \sqrt{\frac{N}{12}} & \text{for edge-variant} \end{cases} \tag{23}$$

for $\gamma > 1$ and $\beta_{\text{ev}} > 1$ and where $\varsigma_1 = \frac{1}{(1-\gamma^{-2})}\big(\frac{1-\gamma^{K+1}}{1-\gamma} - \frac{1-\gamma^{-K-1}}{1-\gamma^{-1}}\big)$ and $\varsigma_2 = \frac{1}{(1-{\beta_{\text{ev}}}^{-2})}\big(\frac{1-{\beta_{\text{ev}}}^{K+1}}{1-\beta_{\text{ev}}} - \frac{1-{\beta_{\text{ev}}}^{-K-1}}{1-{\beta_{\text{ev}}}^{-1}}\big)$.
Proof: By replacing in (20) Δ_κ with $\gamma^\kappa \Delta_0$ for both node-invariant and node-variant and with $\beta_{\text{ev}}^\kappa \Delta_0$ for edge-variant, we obtain readily finite summations than can be written as (22). By following a similar approach and replacing in (20) Δ_κ with $\gamma^{-\kappa}\Delta_0$ and $\beta_{\text{ev}}^{-\kappa}\Delta_0$, we can obtain (23). □

V. Numerical Experiments

In this section, numerical experiments are conducted to validate our theoretical findings. The application of interest is source localization, where a diffusion process over a connected and undirected graph of $N = 50$ nodes splitted into 5 communities $\{c_1, ..., c_5\}$ is considered. As in [16], the graph considered is a stochastic block model graph, which has respectively as intra- and inter-edge probabilities 0.8 and 0.2. The objective is to determine which community is the source that originated the diffusion process. For that, we observe different realizations at different time instants. The initial source signal $\mathbf{x}(0)$, which is a Kronecker delta centered at the source node, is diffused at time τ such that $\mathbf{x}(\tau) = \mathbf{S}^\tau \mathbf{x}(0)$, where $\mathbf{S} = \mathbf{A}/\lambda_{\max}(\mathbf{A})$ is the normalized graph adjacency matrix. The generated training data set $(\mathbf{x}(\tau), c_i)$ is comprising of 10000 samples, picked uniformly at random τ and $i \in \{1 \cdots 5\}$.

TABLE I: GNN average source localization accuracy. The value inside $(\cdot)$ is the standard deviation. The nonlinearity function used is ReLu. The graph filter order is $K = 5$. The maximum size of the messages exchanged is 64 bytes. The maximum number of bits that can be used for quantization over the iterations is 25 bits. The initial given quantization stepsize is $\Delta_0 = 0.015$.

GNN architecture	**1 Layer**	**2 Layers**	**4 Layers**
Node-invariant	64.88%(±1.02)	79.42%(±1.34)	79.00%(±0.29)
Node-invariant Δ	63.75%(±1.08)	72.50%(±0.78)	77.50%(±0.82)
Node-invariant Δ_k	64.50%(±1.42)	75.38%(±1.18)	77.58%(±0.72)
Node-variant	66.42%(±1.37)	79.88%(±1.03)	79.17%(±0.38)
Node-variant Δ	65.21%(±0.91)	76.00%(±1.08)	77.71%(±0.80)
Node-variant Δ_k	65.67%(±1.45)	77.21%(±2.02)	78.00%(±0.52)
Edge-variant	78.92%(±0.91)	79.92%(±1.52)	79.90%(±2.76)
Edge-variant Δ	77.46%(±0.88)	77.54%(±0.89)	77.50%(±3.38)
Edge-variant Δ_k	77.75%(±0.83)	77.64%(±1.92)	78.12%(±3.77)

Table I compares the performance of 9 different GNN architectures composed of $L = \{1, 2, 4\}$ layers of distributed graph filters. For each form of graph filter used in the GNN architecture (i.e, node-invariant, node-variant, edge-variant), we analyze the average source localization accuracy for three different cases: $i)$ without quantization, $ii)$ with a fixed quantization stepsize $\Delta = \Delta_0$, $iii)$ with a decreasing quantization stepsize Δ_k as suggested in Corollary 1. These results are tested and validated with 200 new samples and averaged over ten different graph realizations and ten different data. Among the three forms of distributed graph filters adopted, with or without quantization, GNN with edge-variant filters achieves the best accuracy, followed by GNN with node-variant filters. The results show also that the GNN accuracy in case of quantization is sligthly lower than the case without quantization, but the overall performance still remains relatively good, hence justifying the use of quantization in GNNs with resource constrained nodes. The results highlight also the benefits of using a decreasing quantization stepsize in GNN as compared to a fixed quantization stepsize, where better accuracy is achieved.

VI. Conclusion

In this work, we propose a quantized GNN architecture based on distributed graph filters for signals defined on graphs and analyze the effects of quantization on its final output. Independently of which form of distributed graph filters is used, we demonstrate that the expected error due to quantization at the GNN output can be upper-bounded. We show also the importance of adopting a decreasing quantization stepsize to reduce the accumulated quantization errors, improving thus the GNN performance. Numerical results show that better accuracy for the application of source localization is obtained with a decreasing quantization stepsize as compared to a fixed quantization stepsize.

VII. Appendix I

Proof of Theorem 1: Using Jensen inequality of the l_2-norm in (18), we have:

$$\big\|\mathbb{E}\big[\sigma(\mathbf{z}_L^f) - \sigma(\tilde{\mathbf{z}}_L^f)\big]\big\| \leq \mathbb{E}\big[\big\|\sigma(\mathbf{z}_L^f) - \sigma(\tilde{\mathbf{z}}_L^f)\big\|\big] \tag{24}$$

Next, we assume that the nonlinear operation $\sigma(\cdot)$ is applied individually to each entry of $\mathbf{z}_L^f$ or $\tilde{\mathbf{z}}_L^f$ and the nonlinearity $\sigma(\cdot)$ is normalized Lipschitz so that $|\sigma(b)-\sigma(a)| \le C_\sigma |b-a|$ (which is true for ReLu). Thus, we can write:

$$\begin{aligned}
&\big\|\sigma(\mathbf{z}_L^f) - \sigma(\tilde{\mathbf{z}}_L^f)\big\| \le C_\sigma \big\|\mathbf{z}_L^f - \tilde{\mathbf{z}}_L^f\big\| \qquad (25)\\
&\le C_\sigma \Big\| \sum_{g=1}^{F_{L-1}} \mathbf{H}_L^{fg}(\mathbf{S})\mathbf{x}_{L-1}^g - \sum_{g=1}^{F_{L-1}} \big(\mathbf{H}_L^{fg}(\mathbf{S})\boldsymbol{\varkappa}_{L-1}^g + \boldsymbol{\xi}_{L-1}^{fg}\big)\Big\|\\
&\le C_\sigma \Big(\Big\| \sum_{g=1}^{F_{L-1}} \mathbf{H}_L^{fg}(\mathbf{S})(\mathbf{x}_{L-1}^g - \boldsymbol{\varkappa}_{L-1}^g)\Big\| + \Big\| \sum_{g=1}^{F_{L-1}} \boldsymbol{\xi}_{L-1}^{fg}\Big\|\Big)\\
&\le C_\sigma \sum_{g=1}^{F_{L-1}} \big\|\mathbf{H}_L^{fg}(\mathbf{S})\big\| \, \big\|\mathbf{x}_{L-1}^g - \boldsymbol{\varkappa}_{L-1}^g\big\| + C_\sigma \sum_{g=1}^{F_{L-1}} \big\|\boldsymbol{\xi}_{L-1}^{fg}\big\|
\end{aligned}$$

where we use the subadditivity properties of the l_2-norm i.e., $\|\mathbf{u} \pm \mathbf{v}\| \le \|\mathbf{u}\| + \|\mathbf{v}\|$, and the fact that $\|\mathbf{A}\mathbf{u}\| \le \|\mathbf{A}\|\|\mathbf{u}\|$.

By applying the expected value in (25), we can write for any layer ℓ:

$$\begin{aligned}
&\mathbb{E}\big[\big\|\sigma(\mathbf{z}_\ell^f) - \sigma(\tilde{\mathbf{z}}_\ell^f)\big\|\big] \qquad (26)\\
&\le C_\sigma \sum_{g=1}^{F_{\ell-1}} \big\|\mathbf{H}_\ell^{fg}(\mathbf{S})\big\| \mathbb{E}\big[\|\mathbf{x}_{\ell-1}^g - \boldsymbol{\varkappa}_{\ell-1}^g\|\big] + C_\sigma \sum_{g=1}^{F_{\ell-1}} \mathbb{E}\big[\|\boldsymbol{\xi}_{\ell-1}^{fg}\|\big]\\
&\le C_\sigma \sum_{g=1}^{F_{\ell-1}} \big\|\mathbf{H}_\ell^{fg}(\mathbf{S})\big\| \mathbb{E}\big[\|\sigma(\mathbf{z}_{\ell-1}^g) - \sigma(\tilde{\mathbf{z}}_{\ell-1}^g)\|\big] + C_\sigma \sum_{g=1}^{F_{\ell-1}} \mathbb{E}\big[\|\boldsymbol{\xi}_{\ell-1}^{fg}\|\big]
\end{aligned}$$

In addition, we have for the case of node-invariant filters:

$$\big\|\boldsymbol{\xi}_{\ell-1}^{fg}\big\| = \Big\| \sum_{k=1}^{K} h_{kl}^{fg} \sum_{\kappa=0}^{k-1} \mathbf{S}^{k-\kappa} \boldsymbol{\epsilon}_{\ell-1}^{g(\kappa)} \Big\| \le \sum_{k=1}^{K} \sum_{\kappa=0}^{k-1} |h_{kl}^{fg}| \|\mathbf{S}^{k-\kappa}\| \|\boldsymbol{\epsilon}_{\ell-1}^{g(\kappa)}\| \qquad (27)$$

By applying the expected value on (27), we have:

$$\mathbb{E}\big[\|\boldsymbol{\xi}_{\ell-1}^{fg}\|\big] \le \sum_{k=1}^{K} \sum_{\kappa=0}^{k-1} |h_{kl}^{fg}| \|\mathbf{S}^{k-\kappa}\| \mathbb{E}\big[\|\boldsymbol{\epsilon}_{\ell-1}^{g(\kappa)}\|\big] \qquad (28)$$

Similarly, for node-variant and edge-variant graph filters, we have respectively:

$$\mathbb{E}\big[\|\boldsymbol{\xi}_{\ell-1}^{fg}\|\big] \le \sum_{k=1}^{K} \sum_{\kappa=0}^{k-1} \|\mathrm{diag}(\mathbf{h}_{k\ell}^{fg})\| \|\mathbf{S}^{k-\kappa}\| \mathbb{E}\big[\|\boldsymbol{\epsilon}_{\ell-1}^{g(\kappa)}\|\big] \qquad (29)$$

$$\mathbb{E}\big[\|\boldsymbol{\xi}_{\ell-1}^{fg}\|\big] \le \sum_{k=1}^{K} \sum_{\kappa=0}^{k-1} \Big\|\Big(\prod_{\tau=\kappa+1}^{k} \boldsymbol{\Psi}_{\tau\ell}^{fg}\Big)\Big\| \mathbb{E}\big[\|\boldsymbol{\epsilon}_{\ell-1}^{g(\kappa)}\|\big] \qquad (30)$$

By using $\|\mathbf{v}\| = \sqrt{\mathrm{tr}(\mathbf{v}\mathbf{v}^T)}$ and the Jensen inequality $\mathbb{E}[\sqrt{x}] \le \sqrt{\mathbb{E}[x]}$, the commutativity of trace with respect to the expectation, and assuming a uniform quantizer with a dynamic quantization stepsize Δ_τ, so that $\mathrm{tr}\big(\mathbb{E}\big[\boldsymbol{\epsilon}^{(\tau)}\boldsymbol{\epsilon}^{(\tau)^\top}\big]\big) = \mathrm{tr}\big(\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}^{(\tau)}} + \mathbb{E}[\boldsymbol{\epsilon}^{(\tau)}]\mathbb{E}[\boldsymbol{\epsilon}^{(\tau)}]^\top\big) = \mathrm{tr}\big(\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}^{(\tau)}}\big) = \mathrm{tr}\big((\sigma^{(\tau)})^2\,\mathbf{I}\big) = N(\sigma^{(\tau)})^2 = N\Delta_\tau^2/12$, we can write:

$$\begin{aligned}
\mathbb{E}\big[\|\boldsymbol{\epsilon}_{\ell-1}^{g(\kappa)}\|\big] &= \mathbb{E}\Big[\sqrt{\mathrm{tr}(\boldsymbol{\epsilon}_{\ell-1}^{g(\kappa)} \boldsymbol{\epsilon}_{\ell-1}^{g(\kappa)^T})}\Big] \qquad (31)\\
&\le \sqrt{\mathbb{E}\big[\mathrm{tr}(\boldsymbol{\epsilon}_{\ell-1}^{g(\kappa)} \boldsymbol{\epsilon}_{\ell-1}^{g(\kappa)^T})\big]} = \sqrt{N\Delta_\kappa^2/12}
\end{aligned}$$

If we assume for node-invariant graph filter that $|h_{kl}^{fg}| \le \beta_{\text{inv}}$ and $\|\mathbf{S}\| \le \gamma$ and we consider (31), $\mathbb{E}\big[\|\boldsymbol{\xi}_{\ell-1}^{fg}\|\big]$ in (28) can be upper-bounded by:

$$\begin{aligned}
\mathbb{E}\big[\|\boldsymbol{\xi}_{\ell-1}^{fg}\|\big] &\le \sqrt{N/12} \sum_{k=1}^{K} \sum_{\kappa=0}^{k-1} |h_{kl}^{fg}| \|\mathbf{S}^{k-\kappa}\| \Delta_\kappa \qquad (32)\\
&\le \sqrt{N/12} \sum_{k=1}^{K} \sum_{\kappa=0}^{k-1} \beta_{\text{inv}} \gamma^{k-\kappa} \Delta_\kappa
\end{aligned}$$

Similarly, for node-variant and edge-variant graph filters, if we assume that $\big\|\mathrm{diag}(\mathbf{h}_{k\ell}^{fg})\big\| \le \beta_{\text{nv}}$, $\big\|\big(\boldsymbol{\Psi}_{\kappa\ell}^{fg}\big)\big\| \le \beta_{\text{ev}}$ and $\|\mathbf{S}\| \le \gamma$, we can easily upper bound $\mathbb{E}\big[\big\|\boldsymbol{\xi}_{\ell-1}^{fg}\big\|\big]$ as presented in (20).

By using the upper-bounds α of $\mathbb{E}\big[\big\|\boldsymbol{\xi}_{\ell-1}^{fg}\big\|\big]$ [cf. (20)] and μ of $\big\|\mathbf{H}_\ell^{fg}(\mathbf{S})\big\|$ [cf. (21)], we can now write (26) as:

$$\begin{aligned}
&\mathbb{E}\big[\big\|\sigma(\mathbf{z}_\ell^f) - \sigma(\tilde{\mathbf{z}}_\ell^f)\big\|\big] \qquad (33)\\
&\le C_\sigma \sum_{g=1}^{F_{\ell-1}} \mu \, \mathbb{E}\big[\|\sigma(\mathbf{z}_{\ell-1}^g) - \sigma(\tilde{\mathbf{z}}_{\ell-1}^g)\|\big] + C_\sigma F_{\ell-1}\alpha
\end{aligned}$$

We observe in (33) that the upper-bound at layer ℓ depends on the bound at layer $\ell-1$ and the initial input features at layer 1 are $\mathbf{x}_0^1 = \boldsymbol{\varkappa}_0^1 = \mathbf{x}$. By solving (33) with the initial conditions $\mathbf{x}_0^1 = \boldsymbol{\varkappa}_0^1 = \mathbf{x}$ and considering that $\|\sigma(\mathbf{z}_1^f) - \sigma(\tilde{\mathbf{z}}_1^f)\| = \|\boldsymbol{\xi}_0^{fg}\|$ and $\mathbb{E}\big[\|\boldsymbol{\xi}_0^{fg}\|\big] \le \alpha$, we obtain finally the upper-bound (19), which concludes the proof. □

References

[1] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, "Spectral networks and locally connected networks on graphs," in *International Conference on Learning Representations*, 2014.

[2] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Int. Conf. on Neur. Inf. Process. Sys.*, 2016, pp. 3844–3852.

[3] F. Gama, E. Isufi, G. Leus, and A. Ribeiro, "Graphs, convolutions, and neural networks: From graph filters to graph neural networks," *IEEE Signal Processing Magazine*, vol. 37, no. 6, pp. 128–138, 2020.

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

[5] M. Coutino, E. Isufi, and G. Leus, "Advances in distributed graph filtering," *IEEE Trans. Sig. Process.*, vol. 67, no. 9, pp. 2320–2333, May 2019.

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

[7] Y. Guo, "A survey on methods and theories of quantized neural networks," *preprint arXiv:1808.04752*, Sep 2018.

[8] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.

[9] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *preprint arXiv:1510.00149*, 2016.

[10] D. D. Lin, S. S. Talathi, and V. S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *International Conference on Machine Learning*, 2016, pp. 2849–2858.

[11] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4820–4828.

[12] S. A. Tailor, J. Fernandez-Marques, and N. D. Lane, "Degree-Quant: Quantization-aware training for graph neural network," *preprint arXiv:2008.05000*, 2020.

[13] B. Feng, Y. Wang, X. Li, S. Yang, X. Peng, and Y. Ding, "SGQuant: Squeezing the last bit on graph neural networks with specialized quantization," *preprint arXiv:2007.05100*, 2020.

[14] L. Schuchman, "Dither signals and their effect on quantization noise," *IEEE Trans. Commun. Tech.*, vol. 12, no. 4, pp. 162–165, 1964.

[15] L. B. Saad, B. Beferull-Lozano, and E. Isufi, "Quantization analysis and robust design for distributed graph filters," *arXiv:2004.06692*, 2020.

[16] E. Isufi, F. Gama, and A. Ribeiro, "Generalizing graph convolutional neural networks with edge-variant recursions on graphs," in *Proc. European Sig. Process. Conf.*, 2019, pp. 1–5.