Towards Finite-Time Consensus with Graph Convolutional Neural Networks

Iancu, A.; Isufi, E.

Publication date
2020

Document Version
Final published version

Published in
28th European Signal Processing Conference (EUSIPCO 2020)

Towards Finite-Time Consensus with Graph Convolutional Neural Networks

Bianca Iancu and Elvin Isufi
Intelligent Systems Department, Delft University of Technology, Delft, The Netherlands
E-mails: a.iancu-1@student.tudelft.nl; e.isufi-1@tudelft.nl

***Abstract*—This work proposes a learning framework for distributed finite-time consensus with graph convolutional neural networks (GCNNs). Consensus is a central problem in distributed and adaptive optimisation, signal processing, and control. We leverage the link between finite-time consensus and graph filters, and between graph filters and GCNNs to study the potential of a readily distributed architecture for reaching consensus. We have found GCNNs outperform classical graph filters for distributed consensus and generalize better to unseen topologies such as distributed networks affected by link losses.**

***Index Terms*—Finite-time consensus, graph convolutions, graph signal processing, graph neural networks.**

I. Introduction

Distributed average consensus is a fundamental problem in signal processing, sensor networks, and multi-agent control [1]–[7]. A first approach to reach consensus is through distributed iterative solvers, such as randomized gossip [8] or methods of multipliers [9]. These methods reach consensus at steady-state and their convergence rate is dominated by the network topology. A more recent direction considers reaching consensus within a finite number of iterations and frames this problem as a graph filtering operation [10].

The first work to formalize finite-time consensus through graph filters is [11]. This work uses the finite impulse response (FIR) graph filters and designs the filter coefficients by relying on the graph spectrum. Conditions on when the latter is feasible are further analyzed in [12], [13]. A main limitation of these theoretical contributions is that the filter coefficients depend on the specific eigenvalues of the graph Laplacian matrix. The cost of computing the eigendecomposition limits also their applicability to graphs of small dimensions. The designed filters suffer also from numerical issues due to the finite-precision of the eigenvalues. Besides the theoretical insights, the practical benefit of these works is to approximate better consensus in a finite number of iterations compared with the other distributed solvers. The fastest converging filter is the edge varying graph filter [14], which differently from FIRs exploits also nodes' locality and sparsity to enhance the degrees of freedom. However, the edge varying graph filter requires a fixed labeling in both design and implementation phase and the graph structure to be fixed; both assumptions that might be infeasible in practical distributed settings or when the topology changes slightly (e.g. nodes and links that fail).

In this paper, we address distributed finite-time consensus as a learning problem on graphs. We employ a distributed graph convolutional neural network (GCNN) to learn the consensus function in a data-driven fashion. GCNNs can be thought of as extending to graphs conventional CNNs, where the spatial convolutional filters are substituted by graph convolutional filters [15], [16]. By having the FIR graph filter as their integral part, GCNNs link directly to finite-time consensus if the activation functions leave unaffected the distributed implementation. The coupling *graph filter-activation function* also facilitates the transferability of GCNNs to graphs that deviate slightly from the ones they were trained on [17]. The main research question we address is *how do GCNNs behave for distributed finite-time consensus.* Our preliminary results show the potential of the GCNNs to outperform FIRs for reaching consensus. The improved performance is sensitive to the activation function and to the graph topology. Parametric activation functions should be employed when the GCNN with non-parametric ones (e.g., ReLU) has limited discriminatory power or when the communication complexity is limited. Also, better connected graphs yield a better performance. Finally, we observed GCNNs are more robust than FIRs in reaching consensus over graphs affected by link losses.

This paper is organized as follows. Section II recalls some background material about graph signal processing and distributed consensus with graph filters. Section III details the architecture and nonlinearities under study. The numerical experiments are reported in Section IV, while the paper conclusions in Section V.

II. Background

We start with some basic concepts from graph signal processing and then we continue with graph filters and their link to distributed consensus.

A. Graph signal processing

Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertex set $\mathcal{V} = \{v_1, v_2, ..., v_N\}$ of cardinality $|\mathcal{V}| = N$ and edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ of cardinality $|\mathcal{E}| = M$. An edge is a tuple $e_{ij} = (v_i, v_j)$ connecting nodes v_i and v_j. The neighborhood of node v_i is the set of nodes connected to v_i, i.e., $\mathcal{N}_i = \{v_j | (v_i, v_j) \in \mathcal{E}\}$. Associated with $\mathcal{G}$ is an $N \times N$ matrix $\mathbf{S}$, named the graph shift operator (GSO) matrix, whose sparsity pattern matches the graph structure. The entry (i, j) of $\mathbf{S}$ is $[\mathbf{S}]_{i,j} = s_{i,j} \neq 0$, if $i = j$ or $(i, j) \in \mathcal{E}$. Commonly used GSOs are the adjacency matrix $\mathbf{A}$, the graph Laplacian matrix $\mathbf{L}$ (for undirected graphs), or their normalized and translated forms.

On the vertices of $\mathcal{G}$, we define a graph signal $\mathbf{x} = [x_1, x_2, ..., x_N]^\top \in \mathbb{R}^N$, whose i-th component x_i is the signal value of node v_i. The GSO can be used to represent the signal in the graph spectral domain. For this, consider the eigendecomposition $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$ with eigenvectors $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_N]$ and eigenvalues $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_N)$. The graph Fourier transform (GFT) of $\mathbf{x}$ is defined as $\hat{\mathbf{x}} = \mathbf{U}^{-1}\mathbf{x}$, where $\hat{x}_i$ quantifies how much eigenvector $\mathbf{u}_i$ contributes to the variation of signal $\mathbf{x}$ over the graph [10], [18]. As we shall see in the sequel, this Fourier decomposition plays a role in approaching consensus from a spectral perspective. For completeness, the inverse GFT is $\mathbf{x} = \mathbf{U}\hat{\mathbf{x}}$ and the eigenvalues $\mathbf{\Lambda}$ are referred to as the graph frequencies.

B. Consensus as graph signal filtering

Consider the signal $\mathbf{x}$ and the consensus version $\bar{\mathbf{x}} = \bar{x}\mathbf{1}$, with $\bar{x}$ being the mean of $\mathbf{x}$ and $\mathbf{1}$ the vector of all ones. We can think of $\bar{\mathbf{x}}$ as a signal whose GFT coefficients $\hat{\bar{\mathbf{x}}}$ are such that the combined eigenvectors yield the DC component. For the GSO being the graph Laplacian $\mathbf{S} = \mathbf{L}$, this is straightforward since eigenvector $\mathbf{u}_1$ associated to the smallest eigenvalue $\lambda_1 = 0$ is constant, i.e., $\mathbf{u}_1 = 1/\sqrt{N}\mathbf{1}$. Only the first coefficient $\hat{\bar{x}}_1$ is necessary to represent the consensus signal, while all other coefficients can be null, $\hat{\bar{x}}_2 = \ldots = \hat{\bar{x}}_N = 0$. For $\mathbf{S}$ being the adjacency matrix or any other graph representation matrix that does not have a constant eigenvector, vector $\hat{\bar{\mathbf{x}}}$ will have more than one entry (if not all) non-zero to represent the constant signal.

We can think of consensus as a graph filter that takes an heterogeneous graph signal $\mathbf{x}$ and filters it to return the constant mean signal $\bar{\mathbf{x}} = \bar{x}\mathbf{1}$ over the nodes [11]. A graph filter matrix $\mathbf{H}(\mathbf{S})$ w.r.t. the GSO $\mathbf{S}$ is defined as the polynomial matrix of order K

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

that takes as input a graph signal $\mathbf{x}$ to return the output signal $\mathbf{y} = \mathbf{H}(\mathbf{S})\mathbf{x}$. Vector $\mathbf{h} = [h_0, \ldots, h_K]^\top$ contains the $K+1$ filter coefficients. Exploiting the GFT, we can write the input-output graph filtering relation as $\hat{\mathbf{y}} = \mathbf{H}(\mathbf{\Lambda})\hat{\mathbf{x}}$, where the diagonal matrix $\mathbf{H}(\mathbf{\Lambda}) = \sum_{k=0}^{K} h_k \mathbf{\Lambda}^k$ contains the filter frequency response on the main diagonal. Reaching consensus with graph filters of the form in (1) accounts for learning the filter parameters $\mathbf{h}$ such that the signal is low-pass filtered to pass only the DC component.

Another advantage of (1) is its readily distributed implementation. In building the output $\mathbf{y}$, we need to compute the terms $\mathbf{S}\mathbf{x}, \mathbf{S}^2\mathbf{x}, \ldots, \mathbf{S}^K\mathbf{x}$. By exploiting the recursion $\mathbf{S}^k\mathbf{x} = \mathbf{S}(\mathbf{S}^{k-1}\mathbf{x}) = \mathbf{S}\mathbf{x}^{(k-1)}$, node i can compute the shifted signal $\mathbf{x}^{(k)}$ by exchanging previous shift information $\mathbf{x}^{(k-1)}$ with its direct one-hop neighbors $\mathcal{N}_i$, since the shift operator is local. This recursive implementation allows for a distributed communication and computational cost of order $\mathcal{O}(MK)$ [12].

The main benefit of (1) is that, under appropriate conditions on the spectrum of $\mathbf{S}$ [13], coefficients $\mathbf{h}$ can be designed to achieve exact finite-time consensus in at most $K = N$ iterations [11], [19]. However, their applicability is limited to simple (cyclic or star) graphs, since these approaches require high numerical precision of the eigenvalues. An approach to tackle the numerical issues is to consider a different graph filter in (1), such as ARMA [20], node varying [12], or edge varying [21]. Of particular interest is the so-called edge varying graph filter [21], which substitutes scalars h_k with $N \times N$ coefficient matrices $\mathbf{H}_k$ in which entry (i, j) is the coefficient applied to edge e_{ij}. In this case, finite-time consensus can be approximated with higher accuracy compared with (1), but the graph structure and its labeling should be fixed. The latter is also practically non-transferable to a slightly different graph, such as a graph affected by link losses.

Employing instead a GCNN with filters of the form in (1) does not require the GSO eigendecomposition, a fixed labeling, and it is better transferable to unseen graphs than the linear graph filter [17].

III. Methods

In this section, we first detail the GCNN architecture and the activation functions under study. Then, we discuss two properties of the GCNN, namely, the permutation equivariance and transference to unseen graphs and their suitability to distributed consensus.

Architecture. We consider a GCNN composed of L graph convolutional layers followed by a *per node* fully connected layer –Figure 1. Each graph convolutional layer comprises a bank of graph filters [cf. (1)] and a nonlinearity. At layer l, the GCNN takes F input features $\{\mathbf{x}_{l-1}^g\}_{g=1}^F$ and produces other F output features $\{\mathbf{x}_l^f\}_{f=1}^F$. Each input feature $\mathbf{x}_{l-1}^g$ is treated as graph signals and processed by a parallel bank of F graph filters $\{\mathbf{H}_l^{fg}\}_f$ of the form (1). The filter outputs are then aggregated over the input index g to yield the f-th intermediate feature

$$\mathbf{z}_l^f = \sum_{g=1}^{F} \mathbf{H}_l^{fg}(\mathbf{S})\mathbf{x}_{l-1}^g = \sum_{g=1}^{F}\sum_{k=0}^{K} h_{kl}^{fg}\mathbf{S}^k\mathbf{x}_{l-1}^g, \text{ for } f \in \{1, \ldots, F\}. \tag{2}$$

The intermediate feature $\mathbf{z}_l^f$ is another graph signal whose i-th entry $[\mathbf{z}_l^f]_i$ is associated with node v_i. The latter is subsequently passed through an activation function $\sigma(\cdot)$ to yield the f-th output of the l-th convolutional layer

$$\mathbf{x}_l^f = \sigma(\mathbf{z}_l^f), \quad \text{for } f \in \{1, \ldots, F\}. \tag{3}$$

Layer l is characterized by the F^2 coefficient vectors $\mathbf{h}_l^{fg} = [h_{0l}^{fg}, \ldots, h_{Kl}^{fg}]^\top$ of filters $\mathbf{H}_l^{fg}(\mathbf{S})$ in (2). Remark the number of input and output features do not need to be the same, but we assume so to ease notation.

The input feature of layer $l = 1$ is the graph signal $\mathbf{x}_0 := \mathbf{x}$ for which we want to reach consensus. The output features of layer L, $\mathbf{x}_L^1, \ldots, \mathbf{x}_L^F$, represent the final convolutional features. The latter can also be seen as a collection of F graph signals, where on node v_i we have the $F \times 1$ feature vector $\boldsymbol{\chi}_{Li} =$

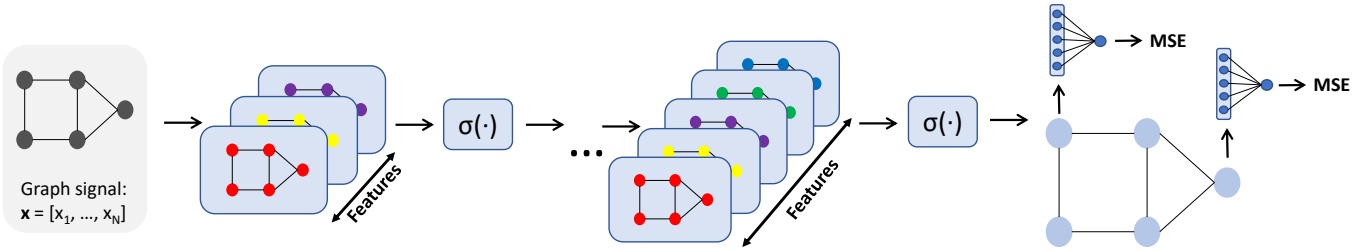


Fig. 1: Distributed GCNN architecture for finite-time consensus. The input is a graph signal $\mathbf{x}$, which is filtered by a filter bank of F FIR graph filters [cf. (1)] and then passed through and activation function $\sigma(\cdot)$. This forms a graph convolutional layer, which is cascaded L times. The final convolutional features are concatenated per node and passed to a *per-node* fully connected layer to compute the final output. This output is used during training the minimze the mean squared error (MSE).

$[x_{Li}^1, \ldots, x_{Li}^F]^\top$. Each node locally combines the features $\boldsymbol{\chi}_{Li}$ with a one-layer perceptron[1] to build the final scalar output

$$y_i = \mathbf{h}_{\text{FC}}^\top \boldsymbol{\chi}_{Li} \tag{4}$$

where $\mathbf{h}_{\text{FC}} = [h_1, \ldots, h_F]^\top$ is the $F \times 1$ vector of parameters in the local fully connected layer. Vector $\mathbf{h}_{\text{FC}}$ is shared among all nodes to keep the number of trainable parameters independent from the graph dimensions.

Activation functions. If the activation functions in the convolutional layers were local, the GCNN would be readily distributable. In fact, all filters $\{\mathbf{H}_l^{fg}(\mathbf{S})\}_{fgl}$ are distributable, as discussed in Section II-B. The last fully connected layer leaves unaffected the distributed implementation since it is local over the nodes. In this work, we study the effect of three activation functions for distributed consensus: the pointwise ReLU, the pointwise kernel [22], and the local max [23].

ReLU: The rectified linear unit is pointwise on each scalar entry x_i of the feature vector $\mathbf{x}$ and it is defined as

$$\sigma(\mathbf{x}) = \max(\mathbf{0}, \mathbf{x}). \tag{5}$$

Kernel: The pointwise kernel activation function considers a one-dimensional dictionary $\mathbf{d} = [d_1, \ldots, d_D]^\top$ of D atoms sampled uniformly around zero. Any scalar feature x_i of node v_i is combined with all elements of $\mathbf{d}$ to build the parametric nonlinear features

$$\sigma(x_i) = \sum_{j=1}^{D} h_j \kappa(x_i, d_j) \tag{6}$$

where $\mathbf{h}_\sigma = [h_1, \ldots, h_D]^\top$ is a $D \times 1$ vector of trainable parameters and $\kappa(x_i, d_j)$ is a one-dimensional kernel between feature value x_i and dictionary atom d_j. Following [22], we employ the Gaussian kernel function $\kappa(x_i, d_j) = \exp\big(-\gamma(x_i - d_j)^2\big)$, where γ is a tuneable parameter.

Max local: Differently from the above two, the max local activation function is not pointwise at node v_i but takes into account also feature values at neighboring nodes $\mathcal{N}_i$. Let $\mathbf{x}$ be an $N \times 1$ graph signal feature on which we want to apply the max local activation function. Then, the output of a local max operator $\max(\mathbf{S}, \mathbf{x})$ applied to signal $\mathbf{x}$ is another graph signal $\mathbf{z}$ whose i-th entry z_i is the maximum value in the neighboorhood, i.e., $z_i = [\max(\mathbf{S}, \mathbf{x})]_i = \max\big(\{x_j : v_j \in \mathcal{N}_i\}\big)$. The max local activation function for the feature signal $\mathbf{x}$ builds the parametric features

$$\sigma(\mathbf{x}) = h_0 \max(\mathbf{0}, \mathbf{x}) + h_1 \max(\mathbf{S}, \mathbf{x}). \tag{7}$$

with trainable parameters $\mathbf{h}_\sigma = [h_0, h_1]^\top$. The ReLU term nonlinearizes also the node features. In [23], the authors extended (7) to a neighborhood of order K. This choice, however, is not distributable and we shall not discuss it further.

The above activation functions leave unaffected the communication and computational costs of the GCNN, which remain governed by the cost of running all graph filters [cf.(1)]. For an architecture of F features per layer and L graph convolutional layers the cost is of order at most $\mathcal{O}(F^2LMK)$.

Parameter training. If the ReLU nonlinearity is used, the total number of parameters of the GCNN is $F^2(L-1)(K+1)+F(K+1)+F$. This divides as: $i)$ $F(K+1)$ parameters for the F filters in the first graph convolutional layer; $ii)$ $F^2(K+1)(L-1)$ for the parameters of the F^2 filters in the remaining $L-1$ graph convolutional layers; and $iii)$ F parameters in the final fully-connected layer. Instead, if the kernel or the max local activation functions are used, we should also consider the parameters in $\mathbf{h}_\sigma$. This adds DL or $2L$ parameters for the kernel or the max local activation function, respectively.

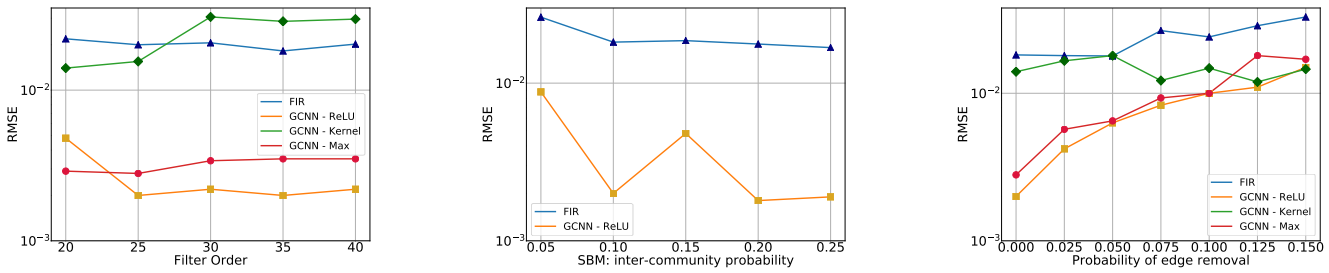
By grouping all parameters into set $\mathcal{H} = \{\mathbf{h}_l^{fg}; \mathbf{h}_{\sigma l}; \mathbf{h}_{\text{FC}}\}_{lfg}$, we can consider the GCNN as a mapping $\boldsymbol{\Phi}(\cdot)$ that takes as input a graph signal $\mathbf{x}$, a GSO $\mathbf{S}$, and a set of parameters $\mathcal{H}$ to produce the output

$$\boldsymbol{\Phi}(\mathbf{x}; \mathbf{S}; \mathcal{H}) := \tilde{\mathbf{y}}. \tag{8}$$

The output (8) is computed for a training set $\mathcal{T} = \{(\mathbf{x}_r, \mathbf{y}_r)\}$ of $|\mathcal{T}| = R$ pairs, where the input $\mathbf{x}_r$ is a graph signal and $\mathbf{y}_r$ is the vector containing the consensus signal $\bar{x}_r$ for all nodes; i.e., $\mathbf{y}_r = \bar{x}_r \mathbf{1}$. The goal of the GCNN is to learn the distributed averaging function from examples in $\mathcal{T}$ and extrapolating it to unseen graph signals $\mathbf{x} \notin \mathcal{T}$.

As a loss function, we considered the averaged mean squared error (MSE) between the GCNN output $\tilde{\mathbf{y}}_r$ and the

[1]Each node can also consider a local multi-layer perceptron to combine the features in $\boldsymbol{\chi}_{Li}$.



(a) (b) (c)

Fig. 2: Root mean square error (RMSE) of the GCNN and FIR graph filters for reaching finite-time consensus. (a) Comparison of different filter orders (iterations) and nonlinearities. (b) Performance of the FIR and GCNN with ReLU nonlinearity as a function of the graph connectivity. (c) Robustness of the different models as a function of link loss probability.

label $\mathbf{y}_r$; i.e.,

$$\mathcal{L} = \frac{1}{R} \sum_{r=1}^{R} \|\mathbf{y}_r - \tilde{\mathbf{y}}_r\|_2^2. \tag{9}$$

This loss is minimized w.r.t. parameters in $\mathcal{H}$ using standard backpropagation with stochastic gradient descent or any other preferred descent method.

Equivariance and transference. The coupling *graph filter-activation function* embodies the GCNN with two important properties, namely, permutation equivariance and transference to unseen graphs. Permutation equivariance implies that the processing of a graph signal with the GCNN is independent of node labeling. This is satisfactory for distributed consensus because we would like to train the GCNN on a graph $\mathcal{G}$ and deploy it on any permuted version of $\mathcal{G}$. Although permutation equivariance restricts the family of activation functions, pointwise nonlinearities and the max local nonlinearity are permutation equivariant [23].

The transference of the GCNN to unseen graphs is essential for distributed consensus since in practical scenarios communication links are prone to perturbations. Transference properties of the graph filters [cf. (1)] and of the GCNN [cf. (8)] are recently linked with their ability to be robust to perturbations [17], [24]. Next, we investigate this property for consensus and observe that GCNNs have a better transference to unseen graphs compared with the FIR filter (1).

IV. Numerical Results

We evaluate the impact of the three activation functions, ReLU (5), kernel (6), and localized max (7), for the GCNN architecture (8) and compare their performance with the FIR graph filter (1). Our goal is to highlight the benefits and limitations of the different activation functions as well as provide preliminary insights on the GCNN behavior when employed for distributed consensus. In specific, the research questions we aim to answer are:

RQ.1 What is the impact of the activation function and filter order on the GCNN?

RQ.2 What is the impact of the graph connectivity when learning the GCNN consensus function?

RQ.3 How do different activation functions behave when the GCNN is deployed on different graphs?

Setup. We considered an undirected stochastic block model (SBM) graph of $N = 100$ nodes divided into $C = 5$ communities with intra- and inter-community probabilities $p = 0.8$ and $q = 0.1$, respectively. The graph signals are generated from a normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. We generated 2500 samples and split them into 80%, 10%, 10% training, validation and test sets, respectively. We averaged the performance across 10 different graph realizations and 10 different data splits for each graph. The GSO is the normalized adjacency matrix $\mathbf{S} = \mathbf{A}/\lambda_{\max}(\mathbf{A})$, where $\lambda_{\max}(\mathbf{A})$ is the maximum eigenvalue of the adjacency matrix[2]. The considered architecture is a two layer GCNN with $F = 32$ features per layer followed by a per-node fully connected layer. To train the parameters, we considered the ADAM optimizer with learning rate 0.001 and forgetting factors $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for 400 epochs and batch size of 100 samples. For the kernel activation function, we considered the same parameters as in [22].

Non-linearity and filter order. We analyzed the three activation functions in Section III and filter orders in the set $K \in \{20, 25, 30, 35, 40\}$. Since for consensus we want the filters to approximate a strongly low-pass transfer function, low filter orders (e.g., $K \in \{1, \ldots, 5\}$ as used for classification) significantly affect the performance. From Fig. 2a, we see the ReLU and the local max activation functions achieve a significantly lower root MSE compared with the kernel activation function but also with the FIR graph filter. The local max performs better than ReLU only for the lowest order $K = 20$, which goes in line with the classification results in [23]. When the filter order K increases, hence the degrees of freedom, adding a parametric nonlinearity is a disadvantage. In fact, the kernel activation function (6) has $D = 20$ extra

[2]We also experimented with the Laplacian as GSO but its performance was consistently worse compared with the normalized adjacency matrix.

parameters per layer and yields a worse performance compared with the local max which has two additional parameters. These observations suggest that parametric activation functions should be preferred when a GCNN architecture with non-parametric ones has a low discriminatory power or when the communication cost is limited.

Graph connectivity. To analyze the impact of the graph connectivity when learning the consensus function, we evaluated the inter-community edge formation probability in the interval $q \in [0.05, 0.25]$. In Fig. 2b, we compare directly the ReLU nonlinearity for $K = 25$ with the FIR graph filter since it was the best performing architecture. For both methods, we observe a lower RMSE when the communities are better connected. This finding is intuitively satisfying for distributed consensus, as the better connected the communities the easier nodes get information from further away neighbors.

Robustness. In this last experiment, we analyze the robustness of the different methods when transferred to graphs affected by link losses. For each method, we considered the best performing order. From the trained graph $\mathcal{G}$, we randomly removed edges with probabilities in the interval $[0.025, 0.15]$. Fig. 2 illustrates the performance averaged over 10 additional realizations. All GCNN models outperform the FIR. It is, however, remarkable that the kernel activation function is quite robust to link losses compared with the rest. We attribute this behavior to the increased degrees of freedom, which trade performance with robustness.

V. Conclusions and Future Work

We proposed a data-driven framework for addressing finite-time consensus with GCNNs. We exploited the link between consensus, graph convolutional filters, and GCNNs to propose a method that is readily distributable if the activation functions are properly chosen and the multilayer perceptron is applied per node. Our preliminary results suggest: i) parametric activation functions should be employed when the distributed graph filters embedded into a non-parametric nonlinearity have limited discriminatory power –the latter is often linked to communication complexity (i.e., filter order); ii) better connected graphs facilitate learning the consensus function –our rationale is the improved performance is because each node gets easier the information from all other nodes; iii) GCNNs generalize better to unseen graphs compared with FIR graph filters. These preliminary observations show the potential of the GCNNs for finite-time consensus rather than being conclusive. Three interesting research directions should be addressed in future work. First, theoretical research is needed to investigate the limits of GCNN for finite-time consensus and link them with the graph spectrum. Second, extensive results in different graphs are needed to validate our observations. Third, it is worth investigating an asynchronous implementation since the latter has often shown superior performance compared with the synchronous one. This work, nevertheless, shows GCNNs overcome by a margin FIR graph filters for finite-time consensus.

References

[1] M. H. DeGroot, "Reaching a consensus," *Journal of the American Statistical Association*, vol. 69, no. 345, pp. 118–121, 1974.

[2] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE transactions on automatic control*, vol. 31, no. 9, pp. 803–812, 1986.

[3] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on automatic control*, vol. 48, no. 6, pp. 988–1001, 2003.

[4] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[5] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[6] S. Sundaram and C. N. Hadjicostis, "Distributed function calculation and consensus using linear iterative strategies," *IEEE journal on selected areas in communications*, vol. 26, no. 4, pp. 650–660, 2008.

[7] S. Pequito, S. Kruzick, S. Kar, J. M. Moura, and A. P. Aguiar, "Optimal design of distributed sensor networks for field reconstruction," in *21st European Signal Processing Conference (EUSIPCO 2013)*. IEEE, 2013, pp. 1–5.

[8] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. SI, pp. 2508–2530, 2006.

[9] T. Sherson, R. Heusdens, and W. B. Kleijn, "On the distributed method of multipliers for separable convex optimization problems," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 3, pp. 495–510, 2019.

[10] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," vol. 30, no. 3, pp. 83–98, May 2013.

[11] A. Sandryhaila, S. Kar, and J. M. Moura, "Finite-time distributed consensus through graph filters," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1080–1084.

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

[13] M. Coutino, E. Isufi, T. Maehara, and G. Leus, "On the limits of finite-time distributed consensus through successive local linear operations," in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2018, pp. 993–997.

[14] M. Coutino, E. Isufi, and G. Leus, "Advances in distributed graph filtering," *IEEE Transactions on Signal Processing*, vol. 67, no. 9, pp. 2320–2333, 2019.

[15] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro, "Convolutional neural network architectures for signals supported on graphs," vol. 67, no. 4, pp. 1034–1049, Feb. 2019.

[16] E. Isufi, F. Gama, and A. Ribeiro, "Edgenets: Edge varying graph neural networks," *arXiv preprint arXiv:2001.07620*, 2020.

[17] F. Gama, J. Bruna, and A. Ribeiro, "Stability properties of graph neural networks," *arXiv:1905.04497v2 [cs.LG]*, 4 Sep. 2019. [Online]. Available: http://arxiv.org/abs/1905.04497

[18] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," vol. 61, no. 7, pp. 1644–1656, Apr. 2013.

[19] S. Sundaram and C. N. Hadjicostis, "Finite-time distributed consensus in graphs with time-invariant topologies," in *2007 American Control Conference*. IEEE, 2007, pp. 711–716.

[20] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Autoregressive moving average graph filtering," vol. 65, no. 2, pp. 274–288, Jan. 2017.

[21] M. Coutino, E. Isufi, and G. Leus, "Advances in distributed graph filtering," vol. 67, no. 9, pp. 2320–2333, May 2019.

[22] S. Scardapane, S. Van Vaerenbergh, D. Comminiello, and A. Uncini, "Improving graph convolutional networks with non-parametric activation functions," in *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE, 2018, pp. 872–876.

[23] L. Ruiz, F. Gama, A. G. Marques, and A. Ribeiro, "Invariance-preserving localized activation functions for graph neural networks," *arXiv preprint arXiv:1903.12575*, 2019.

[24] R. Levie, E. Isufi, and G. Kutyniok, "On the transferability of spectral graph filters," in *13th Int. Conf. Sampling Theory Applications*. Bordeaux, France: IEEE, 8-12 Jul. 2019, pp. 1–5.