Learning by Transference: Training Graph Neural Networks on Growing Graphs

Juan Cerviño ©, Graduate Student Member, IEEE, Luana Ruiz ©, and Alejandro Ribeiro ©, Member, IEEE

Abstract—Graph neural networks (GNNs) use graph convolutions to exploit network invariances and learn meaningful feature representations from network data. However, on large-scale graphs convolutions incur in high computational cost, leading to scalability limitations. Leveraging the graphon—the limit object of a graph—in this paper we consider the problem of learning a graphon neural network (WNN)—the limit object of a GNN—by training GNNs on graphs sampled from the graphon. Under smoothness conditions, we show that: (i) the expected distance between the learning steps on the GNN and on the WNN decreases asymptotically with the size of the graph, and (ii) when training on a sequence of growing graphs, gradient descent follows the learning direction of the WNN. Inspired by these results, we propose a novel algorithm to learn GNNs on large-scale graphs that, starting from a moderate number of nodes, successively increases the size of the graph during training. This algorithm is further benchmarked on a recommendation system and a decentralized control problem, where it retains comparable performance to its large-scale counterpart at a reduced computational cost.

Index Terms—Graph neural networks, graphons, machine learning.

I. INTRODUCTION

Graph neural networks (GNNs) are deep convolutional architectures formed by a succession of layers where each layer composes a graph convolution and a pointwise non-linearity [1], [2]. Tailored to network data, GNNs have been used in a variety of applications such as recommendation systems [3], [4], [5], [6], [7] and Markov chains [8], [9], [10], and fields such as biology [11], [12], [13], [14] and robotics [15], [16], [17]. Their success in these fields and applications provides ample empirical evidence of the ability of GNNs to generalize to unseen data. More recently, their successful performance has also been justified by theoretical works showing that GNNs are invariant to relabelings [18], [19], stable to graph perturbations [20], transferable across graphs [21], and expressive [22].

One of the most important features of a GNN is that, since the linear operation is a graph convolution, its number of parameters does not depend on the number of nodes of the graph. In theory, this means that GNNs can be trained on graphs of any size. But if the graph has a large number of nodes, in practice training the GNN is costly because computing graph convolutions involves large matrix operations. While this issue could be mitigated by transferability [23], i.e., by training the GNN on a smaller graph to execute it on the large graph, this approach does not give any guarantees on the distance between the optimal solutions on the small and on the large graph. Several recent works have proposed to exploit the transferability property of the GNN [21], [23], [24]. In general, these works bound the output difference between two GNNs supported on graphs with different numbers of nodes and the same parameters. Intuitively, transferability works focus on showing that if the graphs belong to a family of graphs modeled by a graphon [25], when the number of nodes increases the transferability error decreases. While transferability is a desired property, it does not immediately imply that the optimal parameters of a GNN on a small graph are optimal, or sufficiently close to optimal, on the large graph. This subtle albeit meaningful observation is what motivates our work.

In this paper, we propose a learning technique for training a GNN on a large graph by progressively increasing the size of the network, and we provide the theoretical guarantees under which the GNN converges to its limit architecture. We consider the limit problem of learning an “optimal” neural network for a graphon, which is both a graph limit and a random graph model [25], [26], [27]. We postulate that, because sequences of graphs sampled from the graphon converge to it, the so-called graphon neural network [21] can be learned by sampling graphs of growing size and training a GNN on these graphs (Algorithm 1). We prove that this is true in two steps. In Theorem 1, we bound the expected distance between the gradient descent steps on the GNN and on the graphon neural network by a term that decreases asymptotically with the size of the graph. A consequence of this bias bound is that it allows us to quantify the trade-off between a more accurate gradient and one that could be obtained with less computational power. We then use this theorem to prove our main result in Theorem 2, which is stated in simplified form below.

Theorem (Graphon neural network learning, informal): Let \( W \) be a graphon and let \( \{G_n\} \) be a sequence of growing graphs sampled from \( W \). Consider the graphon neural network \( \Phi(W) \)
and assume that it is learned by training the GNN $\Phi(G_n)$ with loss function $\ell(y_n, \Phi(G_n))$ on the sequence $\{G_n\}$. Over a finite number of training steps, we obtain

$$\|\nabla \ell(Y, \Phi(W))\| \leq \epsilon$$

with probability 1.

The most important implication of this result is that the learning iterates computed in the sequence of growing graphs follow the direction of the graphon gradient up to a small ball, which provides theoretical support to the proposed training methodology. To further validate our algorithm, we apply it in a numerical experiment where we consider the problem of flocking, and a movie recommendation system. In the flocking case, we train GNNs to learn the actions agents need to take to flock, i.e., to fly with the same velocities while avoiding collisions [28]. In the movie recommendation system case, we train a GNN to predict the rating a user will assign to a given movie based on the known ratings by the user. In both cases, we compare the results obtained when progressively increasing the number of agents during training and when training is done directly on the large target graph.

### A. Related Work

GNNs are neural information processing architectures for network data that follow from seminal works in deep learning applied to graph theory [29], [30], [31], [32]. They have been successfully used in a wide variety of statistical learning problems [33], [34], where their good performance is generally attributed to the fact that they exploit invariances present in network data [35], [36], [37]. More recently, a number of works show that GNNs can be transferred across graphs of different sizes [21], [38], [39]. Specifically, [21] leverages graphons to define families of graphs within which GNNs can be transferred with an error that decreases asymptotically with the size of the graph. The papers by [38] and [39] offer similar results by considering the graph limit to be a generic topological space and a random graph model respectively. In this paper, we use an extension of the transferability bound derived in [21], [23] to propose a novel learning algorithm for GNNs.

### II. Preliminary Definitions

A graph is represented as a triplet $G_n = (\mathcal{V}, \mathcal{E}, W)$, where $\mathcal{V}$, $|\mathcal{V}| = n$, is the set of nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges and $W : \mathcal{E} \rightarrow \mathbb{R}$ is a map assigning weights to each edge. Alternatively, we can also represent the graph $G_n$ by its graph shift operator (GSO) $S_n \in \mathbb{R}^{n \times n}$, a square matrix that respects the sparsity pattern of $G_n$. Examples of GSOs include the adjacency matrix $A$, the graph Laplacian $L = \text{diag}(A1) - A$ and their normalized counterparts [36]. In this paper we consider undirected graphs $G_n$, so that $S_n$ is always symmetric, and fix $S_n = A/n$. In this paper, $\| \cdot \|$ corresponds to the $L^2$ norm, $| \cdot |$ corresponds to the absolute value, and $\| \cdot \|$ corresponds to the 2-norm.

#### A. Graph Convolutions

Graph data is represented in the form of graph signals. A graph signal $x_n = [x_1, \ldots, x_n]^T \in \mathbb{R}^n$ is a vector whose $i$-th component corresponds to the information present at the $i$-th node of graph $G_n$. A basic data aggregation operation can be defined by applying the GSO $S_n$ to graph signals $x_n$ as

$$z_n = S_n x_n.$$  \hfill (1)

The value of signal $x_n$ at node $i$, $x_i$, is a weighted average of the information in the 1-hop neighborhood of $i$, $z_i = \sum_{j \in N_i} [S_n]_{ij} x_j$, where $N_i = \{ j \mid [S_n]_{ij} \neq 0 \}$. Information coming from further neighborhoods (e.g., the $k$-hop neighborhood) can be aggregated by successive applications of the GSO (e.g., $S_n^k x_n$).

Using this notion of shift [40], we can define graph convolutions. Explicitly, the graph convolutional filter with coefficients $h = [h_0, \ldots, h_{K-1}]$ is defined as a polynomial on the GSO $S_n$,

$$y_n = h * S_n x_n = \sum_{k=0}^{K-1} h_k S_n^k x_n  \hfill (2)$$

where $x_n, y_n \in \mathbb{R}^n$ are graph signals and $* S_n$ denotes the convolution operation with GSO $S_n$.

Since the adjacency matrix of an undirected graph is always symmetric, the GSO admits an eigendecomposition $S_n = \mathbf{V} \boldsymbol{\Lambda} \mathbf{V}^H$. The columns of $\mathbf{V}$ are the GSO eigenvectors and the diagonal elements of $\boldsymbol{\Lambda}$ are its eigenvalues, which take values between $-1$ and $1$ and are ordered as $-1 \leq \lambda_{-1} \leq \lambda_{-2} \leq \ldots \leq 0 \leq \ldots \leq \lambda_2 \leq \lambda_1 \leq 1$. Since the eigenvectors of $S_n$ form an orthonormal basis of $\mathbb{R}^n$, we can project (2) onto this basis to obtain the spectral representation of the graph convolution,

$$h(\lambda) = \sum_{k=0}^{K-1} h_k \lambda^k. \hfill (3)$$

Importantly, note that (3) only depends on the $h_k$ and on the eigenvalues of the GSO. Hence, by the Cayley-Hamilton theorem, convolutional filters may be used to represent any graph filter with spectral representation $h(\lambda) = f(\lambda)$ where $f$ is analytic [41].

### B. Graph Neural Networks

Graph neural networks are layered architectures where each layer consists of a graph convolution followed by a pointwise nonlinearity $\rho$. Layer $l$ can output multiple features $x_{nl}^j, 1 \leq f \leq F_l$, which we stack in a matrix $X_l = [x_{nl}^1, \ldots, x_{nl}^j] \in \mathbb{R}^{n \times F_l}$. Each column of the feature matrix is the value of the graph signal corresponding to feature $f$. To map the $F_{l-1}$ features coming from layer $l-1$ into $F_l$ features, $F_{l-1} \times F_l$ convolutions need to be computed, one per input-output feature pair. Stacking their weights in $K$ matrices $H_{lk} \in \mathbb{R}^{F_{l-1} \times F_l}$, we may write the $l$-th layer of the GNN as

$$X_l = \rho \left( \sum_{k=0}^{K-1} S_n^k X_{l-1} H_{lk} \right). \hfill (4)$$
The GNN input (i.e., the input of layer $l = 1$) is given by $X_0 = X \in \mathbb{R}^{n \times F_0}$ and, in an $L$-layer GNN, the operation in (4) is cascaded $L$ times to obtain the GNN output $Y = X_L$. In this paper, we assume $F_0 = F_L = 1$ so that $Y = y_n$ and $X = x_n$. A more concise representation of this GNN can be obtained by grouping all learnable parameters $H_{lk}$ in a tensor $\mathcal{H} = \{H_{lk}\}_{l,k}$ and defining the map $y_n = \Phi(x_n; \mathcal{H}, S_n)$. Due to the polynomial nature of the graph convolution, the dimensions of the learnable parameter tensor $\mathcal{H}$ are independent from the size of the graph ($K$ is typically much smaller than $n$). Ergo, a GNN trained on a graph $G_n$ can be deployed on a network $G_m$ with $m \neq n$.

C. Graphons

A graphon is a bounded, symmetric, and measurable function $W : [0, 1]^2 \rightarrow [0, 1]$ which has two theoretical interpretations: it is both a graph limit and a generative model for graphs. In the first interpretation, sequences of dense graphs converge to a graphon in the sense that the densities of adjacency-preserving graph motifs converge to the densities of these same motifs on the graphon [25]. A graphical example of the convergence of dense graphs to a graphon is depicted in Figure 1. In the second interpretation, graphs can be generated from a graphon by sampling points $u_i, u_j$ from the unit interval and either assigning weight $W(u_i, u_j)$ to edges $(i, j)$, or sampling edges $(i, j)$ with probability $W(u_i, u_j)$. In this paper, we focus on graphs $G_n$ where the points $u_i$ are defined as $u_i = (i - 1)/n$ for $1 \leq i \leq n$ and where the adjacency matrix $S_n$ is sampled from $W$ as

$$[S_n]_{ij} \sim \text{Bernoulli}(W_{ij}), \quad [S_n]_{ij} = W(u_i, u_j)$$  (5)

We refer to the graph $S_n$ as a stochastic graph, and to the graph $X_n$ from which its edges are sampled as the $n$-node template graph associated with the graphon $W$. Sequences of stochastic graphs generated in this way can be shown to converge to $W$ with probability one [25] [Chapter 11].

In practice, the two theoretical interpretations of a graphon allow thinking of it as an identifying object for a family of graphs of different sizes that are structurally similar. Hence, given a network we can use its family’s identifying graphon as a continuous proxy for the graph. This is beneficial because it is typically easier to operate in continuous domains than in discrete domains, even more so if the network is large. In the following, we will leverage these ideas to consider graphon data and graphon neural networks as proxies for graph data and GNNs supported on graphs of arbitrary size.

D. Graphon Convolutions

A graphon signal is defined as a function $X \in L^2([0, 1])$. Analogously to graph signals, graphon signals can be diffused by application of a linear integral operator given by

$$T_W X(v) = \int_0^1 W(u, v) X(u) du.$$  (6)

which we call the graphon shift operator (WSO). Since $W$ is bounded and symmetric, $T_W$ Hilbert-Schmidt and self-adjoint [42]. As such, $W$ can be written as $W(u, v) = \sum_{k \in \mathbb{Z}_0} \lambda_k \varphi_k(u) \varphi_k(v)$ where $\lambda_k$ are the graphon eigenvalues and $\varphi_k$ the graphon eigenfunctions. The eigenvalues have magnitude at most one and are ordered as $-1 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq 0 \leq \ldots \lambda_2 \leq \lambda_1 \leq 1$. The eigenfunctions form an orthonormal basis of $L^2([0, 1])$.

The notion of a graphon shift allows defining graphon convolutions and specifically, we define the graphon convolutional filter with coefficients $[h_0, \ldots, h_K]$ as

$$Y = T_h X = h \ast_W X = \sum_{k=0}^{K-1} h_k(T^{(k)}_W X)(v)$$

with

$$(T^{(k)}_W X)(v) = \int_0^1 W(u, v) (T^{(k-1)}_W X)(u) du$$  (7)

where $X, Y \in L^2([0, 1])$ are graphon signals, $\ast_W$ denotes the convolution operation with graphon $W$ and $T^{(0)}_W = I$ is the identity [43]. Projecting the filter (2) onto the eigenbasis $\{\varphi_k\}_{k \in \mathbb{Z}_0}$, we see that the graphon convolution admits a spectral representation given by

$$h(\lambda) = \sum_{k=0}^{K-1} h_k \lambda^k.$$  (8)

Like its graph counterpart, the spectral representation of the graphon convolution only depends on the coefficients $h$ and on the eigenvalues of the graphon.

E. Graphon Neural Networks

Graphon neural networks (WNNs) are the extensions of GNNs to graphon data. In the WNN, each layer consists of a
bank of graphon convolutions (7) followed by a nonlinearity \( \rho \). Assuming that layer \( l \) maps \( F_{l-1} \) features into \( F_l \) features, the parameters of the \( F_{l-1} \times F_l \) convolutions (7) can be stacked into \( K \) matrices \( \{ H_{lk} \} \in \mathbb{R}^{F_{l-1} \times F_l} \). Then, we can write the \( f \)th feature of the \( f \)th layer as

\[
X^f_l = \rho \left( \sum_{g=1}^{F_{l-1}} \sum_{k=1}^{K} (T_{W}^{(k)} X_{f-1}^g) | H_{lk} |_{gf} \right)
\]

for \( 1 \leq f \leq F_l \).

For an \( L \)-layer WNN, \( X^0_l \) is given by the input data \( X^0 \) for \( 1 \leq g \leq F_0 \), (9) is repeated for \( 1 \leq \ell \leq L \), and the WNN output is given by \( Y^f = X^f_L \). In this paper, we fix \( F_L = F_0 = 1 \) so that \( X_L = Y \) and \( X_0 = X \). Like the GNN, the WNN can be represented more succinctly as a map \( Y = \Phi(X; \mathcal{H}, \mathbf{W}) \). The tensor \( \mathcal{H} = \{ H_{lk} \} \) groups the filter coefficients at all layers and for all features of the WNN.

From the representation of the GNN and the WNN as maps \( \Phi(x_n; \mathcal{H}, S_n) \) and \( \Phi(x; \mathcal{H}, \mathbf{W}) \), we see that these architectures can share the same filter coefficients \( \mathcal{H} \). This observation is important because, since graphs can be sampled from the form \( \mathcal{H} \), \( \mathcal{H} \) is unknown, we cannot derive a WNN parametrized for \( \mathcal{H} \) is unknown. Given a GNN \( \Phi(x_n; \mathcal{H}, S_n) \), the WNN induced by this GNN is defined as

\[
y_n = \Phi(x_n; \mathcal{H}, S_n) \] and \( \mathcal{H} \) is unknown.

Conversely, a WNN can be induced by a GNN. Given a GNN \( y_n = \Phi(x_n; \mathcal{H}, S_n) \), the WNN induced by this GNN is defined as

\[
y_n = \Phi(x_n; \mathcal{H}, \mathbf{W}) \] and \( \mathcal{H} \) is unknown.

In other words, the WNN can be interpreted as a generative model for GNNs \( \Phi(x_n; \mathcal{H}, S_n) \).

As we show in the next subsection, the gradient descent iterates on the graph (15) are good approximations of the gradient descent steps on the graphon (14).

### III. GRAPHON EMPIRICAL LEARNING

In this paper, our goal is to solve statistical learning problems where the data is supported on very large graphs. We restrict the optimization domain to the function class of graphon neural networks (cf. (9)) both to avoid trivial solutions and to be able to generalize to unseen data while leveraging invariances of the graph. Explicitly, we aim to solve an optimization problem of the form

\[
\min_{\mathcal{H}} E_{p(Y,X)}[\ell(Y, \Phi(X; \mathcal{H}, \mathbf{W}))]
\]

where \( p(Y, X) \) is an unknown joint data distribution, \( \ell \) a non-negative loss function and \( \Phi(X; \mathcal{H}, \mathbf{W}) \) a WNN parametrized by the graphon \( \mathbf{W} \) and a set of learnable weights \( \mathcal{H} \). In this paper, we consider non-negative loss functions \( \ell : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^+ \), such that \( \ell(x, y) = 0 \) if and only if \( x = y \). Because the joint probability distribution \( p(Y, X) \) is unknown, we cannot derive a closed-form solution of (12), but this problem can be approximated by the empirical risk minimization (ERM) problem over graphons.

#### A. Graphon Empirical Risk Minimization

Suppose that we have access to samples of the distribution \( \mathcal{D} = \{(X^j, Y^j) \sim p(X, Y), j = 1, \ldots, |\mathcal{D}|\} \). Provided that these samples are obtained independently and that \(|\mathcal{D}| \) is sufficiently large, the statistical loss in (12) can be approximated by the empirical sum over \( \mathcal{D} \),

\[
\min_{\mathcal{H}} \left[ \sum_{j=1}^{|\mathcal{D}|} \ell(Y^j, \Phi(X^j; \mathcal{H}, \mathbf{W})) \right]
\]

giving way to the ERM problem [44], [45], [46], [47]. To solve this problem using local information, we can use gradient descent [48]. The \( k \)th gradient descent iteration is given by

\[
\mathcal{H}_{k+1} = \mathcal{H}_k - \eta_k \frac{1}{|\mathcal{D}|} \sum_{j=1}^{|\mathcal{D}|} \nabla_{\mathcal{H}} \ell(Y^j, \Phi(X^j; \mathcal{H}_k, \mathbf{W}))
\]

(14)

where \( \eta_k \in (0, 1) \) is the step size at iteration \( k \). In practice, the gradients on the right hand side of (14) cannot be computed because the graphon \( \mathbf{W} \) is an unknown limit object. However, we can exploit the fact that \( \mathbf{W} \) is a random graph model to approximate the gradients \( \nabla_{\mathcal{H}} \ell(Y^j, \Phi(X^j; \mathcal{H}_k, \mathbf{W})) \) by sampling stochastic graphs \( \mathbf{G}_n \) with GSO \( S_n \) [cf. (5)] and calculating \( \nabla_{\mathcal{H}} \ell(y^j_n, \Phi(x^j_n; \mathcal{H}_k, S_n)) \), where \( y^j_n \) and \( x^j_n \) are respectively graph signals and graph labels as in (10). In this case, the graphon empirical learning step in (14) becomes

\[
\mathcal{H}_{k+1} = \mathcal{H}_k - \eta_k \frac{1}{|\mathcal{D}|} \sum_{j=1}^{|\mathcal{D}|} \nabla_{\mathcal{H}} \ell(y^j_n, \Phi(x^j_n; \mathcal{H}_k, S_n))
\]

(15)

As we show in the next subsection, the gradient descent iterates on the graph (15) are good approximations of the gradient descent steps on the graphon (14).

#### B. Gradient Approximation

In order to derive an upper bound for the expected error made when using the graph gradients (15) to approximate the graphon gradients (14), we introduce a Lipschitz smoothness definition, as well as two definitions related to the spectra of the graphon \( \mathbf{W} \) and the graphon \( \mathbf{W}_n \) induced by \( S_n \) (cf. equation 11).

**Definition 1** (Lipschitz continuous functions): A function \( f(u_1, u_2, \ldots, u_d) \) is \( \lambda \)-Lipschitz on the variables \( u_1, \ldots, u_d \) if it satisfies \( \|f(v_1, v_2, \ldots, v_d) - f(u_1, u_2, \ldots, u_d)\| \leq \lambda \sum_{i=1}^d \|v_i - u_i\| \). If \( A = 1 \), we say that this function is normalized Lipschitz.

**Definition 2** (c-band cardinality of \( \mathbf{W} \)): The c-band cardinality, denoted \( B_{\mathbf{W}}^c \), is the number of eigenvalues whose absolute
value is larger than \( c \),

\[
B^c_W = \#\{\lambda_i : |\lambda_i| \leq c\}
\]

**Definition 3** (\( c \)-eigenvalue margin of \( W - W_n \)): The \( c \)-eigenvalue margin of \( W \) and \( W_n \) is the minimum distance between eigenvalues \( \lambda_i(Tw) \) and \( \lambda_j(Tw_n) \), \( i \neq j \).

\[
\delta^c_{W W_n} = \min_{i, j \neq i} \{|\lambda_i(Tw) - \lambda_j(Tw_n)| : |\lambda_i(Tw_n)| \geq c\}
\]

We also impose the following Lipschitz continuity assumptions (AS1–AS4), and an assumption on the size of the graph \( S_n \) (AS5).

**Assumption 1**: The graphon \( W \) and the graphon signals \( X \) and \( Y \) are normalized Lipschitz.

**Assumption 2**: The convolutional filters \( h \) are normalized Lipschitz and non-amplifying, i.e., \( |h(\lambda)| < 1 \).

**Assumption 3**: The activation functions and their gradients are normalized Lipschitz, and \( \rho(0) = 0 \).

**Assumption 4**: The loss function \( \ell : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^+ \) and its gradient are normalized Lipschitz, and \( \ell(x, x) = 0 \).

**Assumption 5**: For a fixed value of \( \lambda \) and \( x \), it holds

\[
n - \log(2n/\xi)/d_W > 2/d_W \quad \text{where} \quad d_W = \max_{e} \int_0^1 W(u, v)du.
\]

Under AS1–AS5, the expected norm of the difference between the graphon and graph gradients in (14) and (15) is upper bounded by Theorem 1. The proof is deferred to the Appendix A.

**Theorem 1**: Consider the ERM problem in (13) and let \( \Phi(X; \mathcal{H}, W) \) be an \( L \)-layer WNN with \( F_0 = F_L = 1 \), and \( F_l = F \) for \( 1 \leq l \leq L - 1 \). Let \( e \in [0, 1] \) be the cardinality band of the convolution filters (cf. Definition 2) and assume that the graphon convolutions in all layers of this WNN have \( K \) filter taps (cf. (7)). Let \( \Phi(x_n; \mathcal{H}, S_n) \) be a GNN sampled from \( \Phi(X; \mathcal{H}, W) \) as in (10). Under assumptions AS1–AS5, it holds

\[
E[||\nabla_{\mathcal{H}} \ell(Y, \Phi(X; \mathcal{H}, W)) - \nabla_{\mathcal{H}} \ell(Y_n, \Phi(X_n; \mathcal{H}, W_n))||] \\
\leq \gamma c + O\left(\sqrt{\frac{\log(n^3/\delta^2)}{n}}\right)
\]

where \( Y_n \) is the graphon signal induced by \( [y_{n_i}]_i = Y(u_i) \) (cf. (11)), and \( \gamma \) is a constant that depends on the number of layers \( L \), features \( F \), and filter taps \( K \) of the GNN (cf. Definition 4 in the Appendix A).

Theorem 1 upper bounds the distance between the learning step on the graph and on the graphon for a given input-output pair \( (X, Y) \in \mathcal{D} \) as a function of the size of the graph. Note that this bound is controlled by two terms. The first term decreases with \( n \) and is an approximation bound related to the approximation of \( W \) by \( S_n \). The second term is a constant nontransferable bound controlled by \( c \) and related to a threshold that we impose on the convolutional filter \( h \). Since graphons \( W \) have an infinite spectrum that accumulates around zero, we can only show convergence of all of the spectral components of \( Y_n \) in the limit [21]. Given that \( n \) is finite, we can only upper bound distances between spectral components associated with eigenvalues larger than some fixed threshold \( c \in (0, 1] \). The perturbations associated with the other spectral components are controlled by bounding the variation of the filter response below \( c \).

**Algorithm 1**: Learning by Transference.

1: Initialize parameters \( \mathcal{H}_0 \), set initial number of nodes \( n(0) \) and sample graph \( S_n(0) \) from graphon \( W \)
2: repeat for epochs \( e = 0, 1, \ldots \)
3: for \( k = 1, \ldots, |D| \) do
4: Obtain sample \( (Y, X) \sim D \)
5: Construct graph signal \( y_{n(e)}, x_{n(e)} \) (cf. (10))
6: Take learning step:
\[
\mathcal{H}_{k+1} = \mathcal{H}_k - \eta_k \nabla_\mathcal{H} \ell(y_{n(e)}, \Phi(x_{n(e)}; \mathcal{H}_k, S_{n(e)}))
\]
7: end for
8: Increase number of nodes \( n(e + 1) = n(e) + \Delta(e) \)
9: Sample \( S_n \) Bernoulli from graphon \( W \) (cf. (5))
10: until convergence

The bound in Theorem 1 is important because it allows us to quantify the error incurred by taking gradient steps not on the graphon, but on the graph data. This is instrumental for learning a meaningful solution of the graphon ERM problem (13). Although we cannot calculate gradients of the function that we want to learn (cf. (14)), Theorem 1 shows that we can expect to follow the direction of the gradient on the WNN by measuring the ratio between the norm of the gradient of the loss on the GNN and the difference between the two gradients. Note however that we are only able to decrease this approximation bound down to the value of the nontransferable bound \( \gamma c \).

**Remark 1** (Feasibility of assumptions): AS1–AS4 are normalized Lipschitz smoothness conditions which can be relaxed by making the Lipschitz constant greater than one. AS3 holds for most activation functions used in practice, e.g., the ReLU and the sigmoid. AS4 can be achieved by normalization and holds for most loss functions in a closed set (e.g., the hinge or mean square losses). AS5 is necessary to guarantee a \( O(\sqrt{\log n/n}) \) rate of convergence of \( S_n \) to \( W \) [49].

IV. TRAINING ON GROWING GRAPHS

An insightful observation that can be made from Theorem 1 is that, since the discretization error depends on the number of nodes on the graph, it is possible to reduce its effect by starting from a small number of nodes and iteratively increasing the graph size at regular intervals during the training process. To do so, we denote \( n(e) \) as the number of nodes at epoch \( e \), and we introduce the iteration,

\[
\mathcal{H}_{k+1} = \mathcal{H}_k - \eta_k \nabla_\mathcal{H} \ell(y_{n(e)}, \Phi(x_{n(e)}; \mathcal{H}_k, S_{n(e)})).
\]

Given that graph convolutions are implemented as a series of matrix-vector multiplications (cf. (2)) the number of computations required at epoch \( e \) is proportional to the dimensions of the GSO \( S_n(e) \). Alternatively, if the GNN is trained on a fixed graph of size \( N \) nodes this cost is in the order of \( O(N^2) \) (cf. (15)). From this analysis, it is then ready to see that Algorithm 1 needs less computations than training on the full graph. Formally, in a large-scale graph with \( N \) nodes, training the GNN for \( E \) epochs requires \( EO(N^2) \) calculations. Algorithm 1 only requires \( \sum_{e=0}^{E-1} O(n(e)^2) \) computations. Since typically \( n(e)^2 \ll N^2 \), in practice we can expect Algorithm 1
to be less computationally expensive than training directly on the large-scale graph.

In Algorithm 1, after every epoch $e$, we increase the number of nodes by $\Delta(e)$. In practice, the number of nodes added per epoch $\Delta(e)$ can be a fixed quantity, or can vary between epochs.

### A. Algorithm Convergence

A bound on the difference between the gradients does not, however, imply that we can follow the gradient on the graph blindly. In other words, the fact that gradients are close does not imply that the direction that the iterates follow in the graphon and the graph is the same. In the following subsection, we will use Theorem 1 to obtain the conditions under which the gradient steps taken on the graph follow the learning direction we would expect to follow on the graphon, so that Algorithm 1 converges to a neighborhood of the optimal solution on the graphon. To prove that Algorithm 1 converges, we need the following Lipschitz assumption on the neural network and its gradients.

**Assumption 6:** The graphon neural network $\Phi(X; H, W)$ is $A_{\Phi}$-Lipschitz, and its gradient $\nabla_H \Phi(X; H, W)$ is $A_{\Phi}$-Lipschitz, with respect to the parameters $H$ [cf. Definition 1].

We also need the following lemma that we prove in Appendix B.

**Lemma 1:** Consider the ERM problem in (13) and let $\Phi(X; H, W)$ be an $L$-layer WNN with $F_0 = F_L = 1$, and $F_l = F$ for $1 \leq l \leq L - 1$. Let $c \in (0, 1]$, $\epsilon > 0$, step size $\eta = A_{\Phi}c^{-1}$, with $A_{\Phi} = A_{\Phi} + A_{\Phi} F^2 L / \sqrt{K}$ and assume that the graphon convolutions in all layers of this WNN have $K$ filter taps [cf. (7)]. Let $\Phi(x_{n(k)}; H, S_{n(k)})$ be a GNN sampled from $\Phi(X; H, W)$ as in (10). Consider the iterates generated by (15).

Under Assumptions AS1–AS6, if at step $k$ of epoch $e$ the number of nodes $n(e)$ verifies

$$
\mathbb{E}[\|\nabla_H \ell(Y, \Phi(X; H, W)) - \nabla_H \ell(Y_{n(e)}, \Phi(X_{n(e)}; H, W_{n(e)}))\|] 
\leq \|\nabla_H \ell(Y, \Phi(X; H, W))\|
$$

(18)

then the iterates generated by graph learning step (15) satisfies

$$
\mathbb{E}[\ell(Y, \Phi(X; H_{k+1}, W))] \leq \ell(Y, \Phi(X; H_k, W)).
$$

(19)

Lemma 1 states that if the norm of the difference between the gradients on the graph and the graphon is bounded by the norm of the gradient on the graphon, then the loss on the graphon can be decreased by taking steps on the graph. In conjunction with Theorem 1, Lemma 1 thus gives an upper bound for the number of nodes that needed to follow the learning direction on the graphon, implying that it is always possible to construct a decreasing sequence of loss values by taking steps on the sampled graphs.

The specific condition on the number of nodes is determined by the norm of the gradient of the loss on the graphon. The intuition behind this is that, as the GNN improves its performance, the value of the loss decreases, and so the magnitude of its gradient decreases as well. Alternatively, we can think of the rate of decrease of the graphon gradient as a measure of the difficulty of the learning problem, to which Lemma 1 allows us to set the number of nodes accordingly.

Our main result is stated in Theorem 2, which shows that Algorithm 1 converges if the rate at which the graph grows satisfies the condition in (20).

**Theorem 2:** Consider the ERM problem in (13) and let $\Phi(X; H, W)$ be an $L$-layer WNN with $F_0 = F_L = 1$, and $F_l = F$ for $1 \leq l \leq L - 1$. Let $c \in (0, 1]$, $\epsilon > 0$, step size $\eta = A_{\Phi}c^{-1}$, with $A_{\Phi} = A_{\Phi} + A_{\Phi} F^2 L / \sqrt{K}$ and assume that the graphon convolutions in all layers of this WNN have $K$ filter taps [cf. (7)]. Let $\Phi(x_n; H, S_n)$ be a GNN sampled from $\Phi(X; H, W)$ as in (10). Consider the iterates generated by (15), under Assumptions AS1–AS6, if at each step $k$ of epoch $e$ the number of nodes $n(e)$ verifies

$$
\mathbb{E}[\|\nabla_H \ell(Y, \Phi(X; H, W)) - \nabla_H \ell(Y_{n(e)}, \Phi(X_{n(e)}; H, W_{n(e)}))\| + \epsilon] < \epsilon
$$

(20)

then Algorithm 1 converges to an $\epsilon$-neighborhood of the solution of the Graphon Learning problem (13) in at most $k^* = O(1/\epsilon^2)$ iterations, where $\gamma$ is a constant that depends on the number of layers $L$, features $F$, and filter taps $K$ of the GNN [cf. Definition 4 in the Appendix A].

**Proof:** For every $\epsilon > 0$, we define the stopping time $k^*$ as

$$
k^* := \min_{k \geq 0} \left\{ \mathbb{E}_D[\|\nabla_H \ell(Y, \Phi(X; H, W))\|] \leq \gamma \epsilon \right\}.
$$

(21)

Given the final iterates at $k = k^*$ and the initial values at $k = 0$, we can express the expected difference between the loss $\ell$ as the summation over the difference of iterates,

$$
\mathbb{E}[\ell(Y, \Phi(X; H_{k^*}, W)) - \ell(Y, \Phi(X; H, W))] = \mathbb{E} \left[ \sum_{k=1}^{k^*} \ell(Y, \Phi(X; H_{k-1}, W)) - \ell(Y, \Phi(X; H, W)) \right].
$$

Taking the expected value with respect to the final iterate $k = k^*$, we get

$$
\mathbb{E}[\ell(Y, \Phi(X; H_{k^*}, W)) - \ell(Y, \Phi(X; H_{k^*}, W))] = \mathbb{E} \left[ \sum_{k=1}^{k^*} \ell(Y, \Phi(X; H_{k-1}, W)) - \ell(Y, \Phi(X; H, W)) \right].
$$

(22)

Lemma 1 applied to any $k \leq k^*$ verifies

$$
\mathbb{E}[\ell(Y, \Phi(X; H_{k-1}, W)) - \ell(Y, \Phi(X; H, W))] \geq \eta \gamma \epsilon^2.
$$

Coming back to (22), we get

$$
\mathbb{E}[\ell(Y, \Phi(X; H_{k^*}, W)) - \ell(Y, \Phi(X; H_{k^*}, W))] \geq \eta \gamma \epsilon^2 \sum_{t=0}^{\infty} tP(k^* = t) = \eta \gamma \epsilon^2 \mathbb{E}[k^*].
$$
Since the loss function $\ell$ is non-negative,
\[
\mathbb{E}\left[\ell(Y, \Phi(X; H_0, W))\right] \geq \mathbb{E}[k^*],
\]
from which we conclude that $k^* = O(1/\epsilon^2)$. □

The condition in (20) implies that the rate of decrease in the norm of the gradient of the loss on the GNN needs to be slower than roughly $n^{-1/2}$. If the norm of the gradient does not vary between iterations, the number of nodes does not need to be increased. We also see that the convergence of the gradient on the graphon given by Theorem 1 is equal to the bias term in (16) plus a constant $\epsilon$. We can keep increasing the number of nodes—and thus decreasing the bias—until the norm of the GNN gradient is smaller than the non-transferable constant value. Once this point is attained, there is no gain in decreasing the approximation bound any further [50]. Recall that the constant term can be made as small as desired by tuning $\epsilon$, at the cost of decreasing the approximation bound. Further, note that assuming smoothness of the GNN is a mild assumption [51], [52], [53], [54], [55], [56]. A characterization of the Lipschitz constant is out of the scope of this work.

V. NUMERICAL RESULTS

A. Decentralized Control

In this section we consider the problem of coordinating a set of $n$ agents initially flying at random to avoid collisions, and to fly at the same velocity. Also known as flocking, at each time $t$ agent $i$ knows its own position $r_i(t) \in \mathbb{R}^2$, and speed $v_i(t) \in \mathbb{R}^2$, and reciprocally exchanges it with its neighboring nodes if a communication link exists between them. Links are governed by physical proximity between agents forming a time varying graph $G_n = (V, E)$. A communication link exists if the distance between two agents $i, j$ satisfies
\[
r_{ij}(t) = |r_i(t) - r_j(t)| \leq R = 2m,
\]
where $m$ stands for meters. We assume that at each time $t$ the controller sets an acceleration $u_i \in [-10, 10]^2$, and that it remains constant for a time interval $T_s = 20$ ms. The system dynamics are governed by
\[
\begin{align*}
    r_i(t + 1) &= u_i(t)T_s^2/2 + v_i(t)T_s + r_i(t) \\
    v_i(t + 1) &= u_i(t)T_s + v_i(t).
\end{align*}
\]
To avoid the swarm of robots to reach a null common velocity, we initialize the velocities at random $v(t) = [v_1(t), \ldots, v_n(t)]$, by uniformly sampling a common bias velocity $v_{BIAS} \sim \mathcal{U}[-3, 3]$, and then adding independent uniform noise $\mathcal{U}[-3, 3]$ to each agent. The initial deployment is randomly selected in a circle, always verifying that the minimum distance between two agents is larger than $0.1m$. On the one hand, agents pursue a common average velocity $\bar{v} = (1/n)\sum_{i=1}^{n} v_i(t)$, thus minimizing the velocity variation of the team. On the other hand, agents are required to avoid collisions. We can thus define the velocity variation of the team
\[
\sigma_{v(t)} = \sum_{i=1}^{n} ||v_i(t) - \bar{v}(t)||^2,
\]
and the collision avoidance potential
\[
CA_{ij} = \begin{cases} 
\frac{1}{||r_i - r_j||^2} - \log(||r_i - r_j||^2) & \text{if } ||r_i - r_j|| \leq R_{CA} \\
\frac{1}{R_{CA}^2} - \log(R_{CA}^2) & \text{otherwise}
\end{cases}
\]
with $R_{CA} = 1m$. A centralized controller can be obtain by $u_i(t)^* = -n(v_i - \bar{v}) + \sum_{j=1}^{n} \nabla_{r_j} CA(r_i, r_j)$ [57].

We run the system for $T = 2s$, and used 400 samples for training, 20 for validation, and 20 for the test set. For the optimizer, we used 20 samples for the batch size, and ADAM algorithm [58] with learning rate 0.0005, $\beta_1 = 0.9$, $\beta_2 = 0.999$, without learning rate decay. We used a one layer Graph Neural Networks with $F = 64$ hidden units and $K = 3$ filter taps, and used the hyperbolic tangent as non-linearity $\rho$. We run 10 independent realizations of each experiment.

Exploiting the fact that neural networks are universal approximators [59], [60], Imputation learning can be utilized as a framework to train neural networks from information provided by an expert [61], [62]. Formally, we have a set of pairs $T = \{x_m, u_m^\star\}, m = 1, \ldots, M$, and during training we minimize the mean square error between the optimal centralized controller, and the output of our GNN $||u_m - \Phi(x_m; H, S)||^2$. Denoting $N_i(t)$ the neighborhood of agent $i$ at time $t$, the state of the agents $x(t) = [x(1), \ldots, x(t)]$, $x_i(t) \in \mathbb{R}^6$, is given by
\[
x_i(t) = \sum_{j: j \in N_i(t)} \left[ v_i(t) - v_j(t), \frac{r_{ij}(t)}{||r_{ij}(t)||^2}, \frac{r_{ij}(t)}{||r_{ij}(t)||^2} \right].
\]
Note that state $x_i(t)$, gets transmitted between agents if a communication link exists between them. In Fig. 2 we can see the empirical manifestation of the claims we put forward. First and foremost, we are able to learn a GNN that achieves a comparable performance while taking steps on smaller graphs. As seen in Fig. 2, GNNs trained with $n_0 = \{10, 20\}$ agents in the first epoch and adding 10 agents per epoch (green line) are able to achieve a similar performance when reaching 100 agents than the one they would have achieved by training with 100 agents the same number of epochs. Besides, if we add less nodes per epoch, we are able to achieve a similar performance that we would have achieved by training on the large network for 30 epochs.

In Fig. 3, we can see that the time per epoch increases as the number of nodes increases. This is related to the fact that as the number of nodes increases, the cost of the computation also increases. In Table I we show the total time of training per method. As can be seen, all the combinations of learning by transference are done faster than taking both 30 and 9 epochs with 100 nodes. Moreover, our method obtains a comparable performance to the GNN trained for 30 epochs on 100 nodes. That is to say, learning by transference obtains a performance comparable to the one obtained with 30 epochs at 100 nodes, in less time that it takes to compute 9 epochs at 100 nodes.
Fig. 2. Velocity variation and standard deviation of the flocking problem for the whole trajectory in the testing set relative to the centralized controller.

Fig. 3. Average time per epoch as a function of the number of nodes in the GNN in the flocking problem.

Fig. 4. RMSE and standard deviation of a GNN predicting the rate a user would assign to a movie based on the known ratings by the user as a function of the number of epochs. The GNN is tested in a graph with 2100 nodes, the Normal train GNN was trained with 2000 nodes, and the other two GNNs were trained with Algorithm 1 starting at 1000 nodes.

We split the dataset with 90% for the training set, and 10% for the testing set, and we run 20 independent random partitions. For the optimizer, we used 5 samples for the batch size, and ADAM algorithm [58], with learning rate 0.005, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and without learning rate decay. For the loss, we used the smooth $L_1$ loss. For the GNN, we used ReLU as non-linearity, we considered $F = 32$ features, $K = 5$ filter taps, and $L = 2$ layers.

For the experiment, we trained a GNN with a constant number of nodes $N = 2000$, and our Algorithm 1 with initial nodes $n_0 = 1000$, and added $\{25, 50\}$ nodes per epoch.

In Fig. 4, we are able to see the empirical manifestation of the benefit of Algorithm 1. Namely, regardless of the number of nodes added per epoch, in all two cases, using relative RMSE as a figure of merit, we achieve a comparable performance to that of a GNNs trained on 2000 nodes for a larger number of epochs. This reinforces the fact that training on a growing number of graphs can attain similar performance, while requiring a lighter computational cost.

**TABLE I**

| Method                              | Time (s) |
|-------------------------------------|----------|
| Normal train 9 epochs               | 2907     |
| Normal train 30 epochs              | 9690     |
| Starting with 10 nodes and adding 2 per epoch | 523     |
| Starting with 10 nodes and adding 5 per epoch | 2393    |
| Starting with 10 nodes and adding 10 per epoch | 1235    |
| Starting with 20 nodes and adding 2 per epoch | 478     |
| Starting with 20 nodes and adding 5 per epoch | 2376    |
| Starting with 20 nodes and adding 10 per epoch | 1228    |

**B. Recommendation System**

We consider a social recommendation problem given by movie ratings, for which we use the MovieLens 1M dataset [63]. The dataset contains 1,000,209 integer ratings between 1 and 5, that were collected between $U = 6,040$ users and $M = 3,900$ movies. We consider the problem of predicting the rating that different users would give to the movie in node 50. To exploit the social connectivity of the problem, we build the movie similarity graph, by computing the pairwise correlations between the different movies in the training set [64].
In Fig. 5 we show the average time per epoch as a function of the number of nodes in the GNN in the recommendation system problem.

**TABLE II**

| Method                  | Time (s) |
|-------------------------|----------|
| Normal Train            | 1484     |
| 50 nodes added per epoch| 819      |
| 25 nodes added per epoch| 823      |

VI. CONCLUSION

We have introduced a learning procedure for GNNs that progressively grows the size of the graph while training. Our algorithm requires less computational cost—as the number of nodes in the graph convolution is smaller—than training on the full graph without compromising performance. Leveraging transferability results, we bounded the expected difference between the gradient on the GNN, and the gradient on the WNN. Utilizing this result, we provided the theoretical guarantees that our Algorithm converges to a neighborhood of a first order stationary point of the WNN in finite time. We benchmarked our algorithm on a decentralized control problem and a recommendation system, achieving comparable performance to the one achieved by a GNN trained on the full graph.

APPENDIX A

**Proof of Theorem 1**

**Proposition 1:** Let $X \in L_2([0, 1])$ be a normalized Lipschitz graphon signal, and let $X_n$ be the graphon signal induced by the graph signal $x_n$ obtained from $X$ on the template graph $\mathbb{G}_n$, i.e., $[x_n]_i = X((i-1)/n)$ for $1 \leq i \leq n$. It holds that

$$\|X - X_n\| \leq \frac{1}{n}. \quad (27)$$

**Proof:** Let $I_i = [(i-1)/n, i/n)$ for $1 \leq i \leq n - 1$ and $I_n = [(n-1)/n, 1]$. Since the graphon is normalized Lipschitz, for any $u \in I_i$, $1 \leq i \leq n$, we have

$$\|X(u) - X_n(u)\| \leq \max \left( \left| u - \frac{i-1}{n} \right|, \left| \frac{i}{n} - u \right| \right) \leq \frac{1}{n}.$$

We can then write

$$\|X - X_n\|^2 = \int_0^1 |X(u) - X_n(u)|^2 du$$

$$\leq \int_0^1 \left( \frac{1}{n} \right)^2 du = \left( \frac{1}{n} \right)^2,$$

which completes the proof.

**Proposition 2:** Let $W : [0, 1]^2 \rightarrow [0, 1]$ be a normalized Lipschitz graphon, and let $W_n := W_{G_n}$ be the graphon induced by the template graph $\mathbb{G}_n$ generated from $W$. It holds that

$$\|W - W_n\| \leq \frac{2}{n}. \quad (28)$$

**Proof:** Let $I_i = [(i-1)/n, i/n)$ for $1 \leq i \leq n - 1$ and $I_n = [(n-1)/n, 1]$. Since the graphon is Lipschitz, for any $u \in I_i$, $v \in I_j$, $1 \leq i, j \leq n$, we have

$$\|W(u, v) - W_n(u, v)\| \leq \max \left( \left| u - \frac{i-1}{n} \right|, \left| \frac{i}{n} - u \right| \right)$$

$$+ \max \left( \left| v - \frac{j-1}{n} \right|, \left| \frac{j}{n} - v \right| \right)$$

$$\leq \frac{1}{n} + \frac{1}{n} = \frac{2}{n}.$$

We can then write

$$\|W - W_n\|^2 = \int_0^1 |W(u, v) - W_n(u, v)|^2 du dv$$

$$\leq \int_0^1 \left( \frac{2}{n} \right)^2 du dv = \left( \frac{2}{n} \right)^2,$$

which concludes the proof.

**Lemma 2:** Consider the $L$-layer WNN given by $Y = \Phi(X; H, W)$, where $F_0 = F_L = 1$ and $F_l = F$ for $1 \leq l \leq L - 1$. Let $c \in (0, 1)$ and assume that the graphon convolutions in all layers of this WNN have $K$ filter taps [cf. (7)]. Under Assumptions 1 through 3, the norm of the gradient of the WNN with respect to its parameters $H = \{H_{lk}\}_{l,k}$ can be upper bounded by

$$\|\nabla_H \Phi(X; H, W)\| \leq F^2 L \sqrt{K}. \quad (29)$$

**Proof:** We will find an upper bound for any element $[H_{lk}[g][f]]$ of the tensor $H$. We start by the last layer of the WNN, applying the definition given in (9)

$$\|\nabla_{[H_{lk}[g][f]]} \Phi(X; H, W)\| = \left\| \nabla_{[H_{lk}[g][f]]} X_l^g \right\|$$

$$= \left\| \nabla_{[H_{lk}[g][f]]} \rho \left( \sum_{g=1}^{F_l-1} \sum_{k=1}^{K-1} (T^{(k)} X_{l-1}^g)[H_{lk}][g][f] \right) \right\|. $$
By Assumption 3, the nonlinearity $\rho$ is normalized Lipschitz, i.e., $\nabla \rho(u) \leq 1$ for all $u$. Thus, applying the chain rule for the derivative and the Cauchy-Schwartz inequality, the right hand side can be rewritten as

$$
\|\nabla_{[H_{ik,l}]_{g,f}} \Phi(X; H, W)\|
\leq \nabla \rho \left( \sum_{g=1}^{F_{i-1}} \sum_{k=1}^{K-1} (T_W^{(k)} X_{l-1}^g) [H_{lk}]_{g,f} \right) 
\times \left\| \nabla_{[H_{ik,l}]_{g,f}} \sum_{g=1}^{F_{i-1}} \sum_{k=1}^{K-1} (T_W^{(k)} X_{l-1}^g) [H_{lk}]_{g,f} \right\|
\leq \left\| \nabla_{[H_{ik,l}]_{g,f}} \sum_{g=1}^{F_{i-1}} \sum_{k=1}^{K-1} (T_W^{(k)} X_{l-1}^g) [H_{lk}]_{g,f} \right\|.
$$

Note that the a larger bound will occur if $l^1 < L - 1$, so by linearity of derivation and by the triangle inequality, we obtain

$$
\|\nabla_{[H_{ik,l}]_{g,f}} \Phi(X; H, W)\|
\leq \sum_{g=1}^{F_{i-1}} \left\| \nabla_{[H_{ik,l}]_{g,f}} X_{l-1}^g \right\|.
$$

By Assumption 2, the convolutional filters are non-amplifying, thus it holds that

$$
\|\nabla_{[H_{ik,l}]_{g,f}} \Phi(X; H, W)\|
\leq \sum_{g=1}^{F_{i-1}} \|H_{ik,l}\|_{g,f} X_{l-1}^g.
$$

Now note that as filters are non-amplifying, the maximum difference in the gradient will be attained at the first layer ($l = 1$) of the WNN. Also note that the derivative of a convolutional filter $T_H$ at coefficient $k^l = i$ is itself a convolutional filter with coefficients $h_i$. The values of $h_i$ are $h_{i,j} = 1$ if $j = i$ and 0 otherwise. Then, we have

$$
\|\nabla_{[H_{ik,l}]_{g,f}} \Phi(X; H, W)\|
\leq F^{L-1} \|H_{ik}\| \|X_0\|
\leq F^{L-1} \|X_0\|.
$$

To complete the proof note that tensor $H$ has $F^{L-1} K$ elements, and each individual gradient is upper bounded by (30), and $\|X\|$ is normalized by Assumption 1. □

Lemma 3: Let $\Phi(X; H, W)$ be a WNN with $F_0 = F_L = 1$, and $F_i = F$ for $1 \leq l \leq L - 1$. Let $c \in (0, 1)$, and assume that the graphon convolutions in all layers of this WNN have $K$ filter taps [cf. (7)]. Let $\Phi(x_n; H, S_n)$ be a GNN sampled from $\Phi(X; H, W)$ as in (10). Under Assumptions 1, 2, 3 and 5, with probability $1 - \xi$ it holds that

$$
\|\Phi(X; H, W) - \Phi(x_n; H, W_n)\|
\leq LF^{L-1} \left( 1 + \frac{\pi B^c \delta_{Wn}}{\delta_{Wn}} \right) \frac{2 \left( 1 + \sqrt{n \log (2n/\xi)} \right)}{n}
+ \frac{1}{n} + 4LF^{L-1} c.
$$

The fixed constants $B^c_W$ and $\delta_{Wn}$ are the $c$-band cardinality and the $c$-eigenvalue margin of $W$ and $W_n$, respectively [cf. Definitions 2, 3].

Proof: The proof can be found in [23, Theorem 3] where $A_h = 1$, $A_w = 1$, and $\|X\| = 1$ by Assumptions 1, 2; and $\alpha(n, \chi) = 1$ and $\beta(n, \chi) = \sqrt{n \log (2n/\xi)}$.

Lemma 4: Let $\Phi(X; H, W)$ be a WNN with $F_0 = F_L = 1$, and $F_i = F$ for $1 \leq l \leq L - 1$. Let $c \in (0, 1)$, and assume that the graphon convolutions in all layers of this WNN have $K$ filter taps [cf. (7)]. Let $\Phi(x_n; H, S_n)$ be a GNN sampled from $\Phi(X; H, W)$ as in (10). Under Assumptions 1, 2, 3, and 5, with probability $1 - \xi$ it holds that

$$
\|\nabla_{\Phi} \Phi(X; H, W) - \nabla_{\Phi} \Phi(x_n; H, W_n)\|
\leq \sqrt{KF^{L-1}} \left( 2F^{L-1} \frac{L}{n} + 8L^2 F^{2L-2} c \right)
+ 2L^2 F^{2L-2} \left( 1 + \frac{\pi B^c_{Wn}}{\delta_{Wn}} \right) \frac{2 \left( 1 + \sqrt{n \log (2n/\xi)} \right)}{n}.
$$

Proof: We will first show that the gradient with respect to any arbitrary element $[H_{ik,l}]_{g,f} \in \mathbb{R}$ of $H$ can be uniformly bounded. Note that the maximum is attained if $l^1 = 1$. Without loss of generality, assuming $l^1 > l - 1$, we can begin by using the output of the WNN in (9) to write

$$
\|\nabla_{[H_{ik,l}]_{g,f}} \Phi(X; H, W) - \nabla_{[H_{ik,l}]_{g,f}} \Phi(x_n; H, W_n)\|
= \|\nabla_{[H_{ik,l}]_{g,f}} X_{l^1}^f - \nabla_{[H_{ik,l}]_{g,f}} X_{n_{l^1}}^f\|
= \left\| \nabla_{[H_{ik,l}]_{g,f}} \rho \left( \sum_{g=1}^{F_{i-1}} \sum_{k=1}^{K-1} (T_W^{(k)} X_{l-1}^g) [H_{lk}]_{g,f} \right) \right\|
\leq \left\| \nabla_{[H_{ik,l}]_{g,f}} \rho \left( \sum_{g=1}^{F_{i-1}} \sum_{k=1}^{K-1} (T_W^{(k)} X_{l-1}^g) [H_{lk}]_{g,f} \right) \right\|.
$$

Applying the chain rule and using the triangle inequality, we get

$$
\|\nabla_{[H_{ik,l}]_{g,f}} X_{l^1}^f - \nabla_{[H_{ik,l}]_{g,f}} X_{n_{l^1}}^f\|
\leq \left\| \nabla \rho \left( \sum_{g=1}^{F_{i-1}} \sum_{k=1}^{K-1} (T_W^{(k)} X_{l-1}^g) [H_{lk}]_{g,f} \right) \right\|
\times \left\| \nabla_{[H_{ik,l}]_{g,f}} \rho \left( \sum_{g=1}^{F_{i-1}} \sum_{k=1}^{K-1} (T_W^{(k)} X_{l-1}^g) [H_{lk}]_{g,f} \right) \right\|
\times \left\| \nabla_{[H_{ik,l}]_{g,f}} \rho \left( \sum_{g=1}^{F_{i-1}} \sum_{k=1}^{K-1} (T_W^{(k)} X_{l-1}^g) [H_{lk}]_{g,f} \right) \right\|
+ \left\| \nabla \rho \left( \sum_{g=1}^{F_{i-1}} \sum_{k=1}^{K-1} (T_W^{(k)} X_{l-1}^g) [H_{lk}]_{g,f} \right) \right\|
+ \left\| \nabla \rho \left( \sum_{g=1}^{F_{i-1}} \sum_{k=1}^{K-1} (T_W^{(k)} X_{l-1}^g) [H_{lk}]_{g,f} \right) \right\|.
$$
Next, we use Cauchy-Schwartz inequality, Assumptions 3, 4, and Proposition 2 to bound the terms corresponding to the gradient of the nonlinearity $\rho$, the loss function $\ell$, and the WNN respectively. Explicitly,
\[
\|\nabla_{[h_{i,lkg}]} F_{i,l} - \nabla_{[h_{i,lkg}]} F_{i,L} \|_{X_{KL}}^2 \leq 2 \sum_{g=1}^{F_i-1} \sum_{k=1}^{K-1} (T_{W}^g X_{l-1}^g) \nabla_{[h_{lk}]}_{gf}^g \left( X_{l-1}^g - X_{n-1}^g \right).
\]

(36)

Applying triangle inequality to the second term, we get
\[
\|\nabla_{[h_{i,lkg}]} F_{i,l} - \nabla_{[h_{i,lkg}]} F_{i,L} \|_{X_{KL}}^2 \leq 2 \sum_{g=1}^{F_i-1} \sum_{k=1}^{K-1} (T_{W}^g X_{l-1}^g) \nabla_{[h_{lk}]}_{gf}^g \left( X_{l-1}^g - X_{n-1}^g \right).
\]

(35)

Now note that as we consider the case in which $l_i < l - 1$, using the Cauchy-Schwartz inequality we can use the same bound for the first and second terms of the right hand side of (35). Also note that, by Assumption 3, the filters are non-expansive, which allows us to write
\[
\|\nabla_{[h_{i,lkg}]} F_{i,l} - \nabla_{[h_{i,lkg}]} F_{i,L} \|_{X_{KL}}^2 \leq 2 \sum_{g=1}^{F_i-1} \sum_{k=1}^{K-1} (T_{W}^g X_{l-1}^g) \nabla_{[h_{lk}]}_{gf}^g \left( X_{l-1}^g - X_{n-1}^g \right).
\]

(34)
\( \Phi(X; \mathcal{H}, \mathbf{W}) \) as in (10). Under Assumptions 1–5, with probability \( 1 - \xi \) it holds that
\[
\begin{align*}
\|\nabla_{\mathcal{H}} \ell(Y_n, \Phi(X; \mathcal{H}, \mathbf{W})) - \nabla_{\mathcal{H}} \ell(Y_n, \Phi(X_n; \mathcal{H}, \mathbf{W}_n))\| & \leq \sqrt{K} F^{L-1} \left( \frac{4 F^{L-1} L}{n} + 12 L^2 F^{2L-2} \right. \\
& \quad + \left. 3 L^2 F^{2L-2} \left( 1 + \frac{\pi B_{\mathbf{W}_n}}{\delta_{\mathbf{W}_n}} \right) \frac{2 \left( 1 + \sqrt{n \log(2n/\xi)} \right)}{n} \right).
\end{align*}
\] (38)

Proof: In order to analyze the norm of the gradient with respect to the tensor \( \mathcal{H} \), we start by taking the derivative with respect to a single element of the tensor, \( [\mathbf{H}]_{i,k,l} \phi f t \). Using the chain rule to compute the gradient of the loss function \( \ell \), we get
\[
\begin{align*}
\|\nabla_{[\mathbf{H}]_{i,k,l} \phi f t} (\ell(Y, \Phi(X; \mathcal{H}, \mathbf{W})) - \ell(Y_n, \Phi(X_n; \mathcal{H}, \mathbf{W}_n)))\| \\
& \leq \|\nabla \ell(Y, \Phi(X; \mathcal{H}, \mathbf{W}))\| - \|\nabla \ell(Y, \Phi(X_n; \mathcal{H}, \mathbf{W}_n))\| \\
& \quad + \|\nabla \ell(Y_n, \Phi(X_n; \mathcal{H}, \mathbf{W}_n))\| \\
& \quad + \|\nabla \ell(Y, \Phi(X; \mathcal{H}, \mathbf{W}))\| \\
& \quad - \|\nabla \ell(Y_n, \Phi(X_n; \mathcal{H}, \mathbf{W}_n))\|.
\end{align*}
\]
and by Cauchy-Schwartz and the triangle inequality, it holds
\[
\begin{align*}
\|\nabla_{[\mathbf{H}]_{i,k,l} \phi f t} (\ell(Y, \Phi(X; \mathcal{H}, \mathbf{W})) - \ell(Y_n, \Phi(X_n; \mathcal{H}, \mathbf{W}_n)))\| \\
& \leq \|\nabla \ell(Y, \Phi(X; \mathcal{H}, \mathbf{W}))\| - \|\nabla \ell(Y, \Phi(X_n; \mathcal{H}, \mathbf{W}_n))\| \\
& \quad + \|\nabla \ell(Y_n, \Phi(X_n; \mathcal{H}, \mathbf{W}_n))\| \\
& \quad + \|\nabla \ell(Y, \Phi(X; \mathcal{H}, \mathbf{W}))\| \\
& \quad - \|\nabla \ell(Y_n, \Phi(X_n; \mathcal{H}, \mathbf{W}_n))\|.
\end{align*}
\]
By the triangle inequality and Assumption 4, it follows
\[
\begin{align*}
\|\nabla_{[\mathbf{H}]_{i,k,l} \phi f t} (\ell(Y, \Phi(X; \mathcal{H}, \mathbf{W})) - \ell(Y_n, \Phi(X_n; \mathcal{H}, \mathbf{W}_n)))\| \\
& \leq \|\nabla \ell(Y, \Phi(X; \mathcal{H}, \mathbf{W}))\| - \|\nabla \ell(Y, \Phi(X_n; \mathcal{H}, \mathbf{W}_n))\| \\
& \quad + \|\nabla \ell(Y_n, \Phi(X_n; \mathcal{H}, \mathbf{W}_n))\| \\
& \quad + \|\nabla \ell(Y, \Phi(X; \mathcal{H}, \mathbf{W}))\| \\
& \quad - \|\nabla \ell(Y_n, \Phi(X_n; \mathcal{H}, \mathbf{W}_n))\|.
\end{align*}
\]
Next, we can use Lemmas 2, 3, and 4, Proposition 1, and Assumption 1 to obtain
\[
\begin{align*}
\|\nabla_{[\mathbf{H}]_{i,k,l} \phi f t} (\ell(Y, \Phi(X; \mathcal{H}, \mathbf{W})) - \ell(Y_n, \Phi(X_n; \mathcal{H}, \mathbf{W}_n)))\| \\
& \leq \left( 3 L^2 F^{2L-2} \left( 1 + \frac{\pi B_{\mathbf{W}_n}}{\delta_{\mathbf{W}_n}} \right) \frac{2 \left( 1 + \sqrt{n \log(2n/\xi)} \right)}{n} \right) \\
& \quad + \frac{4 F^{L-1} L}{n} + 12 L^2 F^{2L-2} c.
\end{align*}
\] (39)

Noting that tensor \( \mathcal{H} \) has \( K F^{L-1} \) elements, and that each individual term can be bounded by (39), we arrive at the desired result.

Definition 4: We define the constant \( \gamma \) as
\[
\gamma = 12 \sqrt{K} F^{L-1} L^2 F^{2L-2} c
\]
where \( F \) is the number of features, \( L \) is the number of layers, and \( K \) is the number of filter taps of the GNN.

Proof of Theorem 1: To start, consider the event \( A_n \) such that
\[
A_n = \left\{ \|\nabla_{\mathcal{H}} \ell(Y, \Phi(X; \mathcal{H}, \mathbf{W})) - \ell(Y_n, \Phi(X_n; \mathcal{H}, \mathbf{W}_n))\| \\
\leq \sqrt{K} F^{L-1} \left( \frac{4 F^{L-1} L}{n} + 12 L^2 F^{2L-2} c \\
+ 3 L^2 F^{2L-2} \left( 1 + \frac{\pi B_{\mathbf{W}_n}}{\delta_{\mathbf{W}_n}} \right) \frac{2 \left( 1 + \sqrt{n \log(2n/\xi)} \right)}{n} \right) \right\}
\]
Taking the disjoint events \( A_n \) and \( A_n^c \), and denoting the indicator function \( 1(\cdot) \), we split the expectation as
\[
\mathbb{E}[\|\nabla_{\mathcal{H}} \ell(Y, \Phi(X; \mathcal{H}, \mathbf{W})) - \ell(Y_n, \Phi(X_n; \mathcal{H}, \mathbf{W}_n))\| \\
= \mathbb{E}[\|\nabla_{\mathcal{H}} \ell(Y, \Phi(X; \mathcal{H}, \mathbf{W})) \\
- \ell(Y_n, \Phi(X_n; \mathcal{H}, \mathbf{W}_n))\| 1(A_n)] \\
+ \mathbb{E}[\|\nabla_{\mathcal{H}} \ell(Y, \Phi(X; \mathcal{H}, \mathbf{W})) \\
- \ell(Y_n, \Phi(X_n; \mathcal{H}, \mathbf{W}_n))\| 1(A_n^c)]
\]
We can then bound the term corresponding to \( A_n^c \) using (41), by splitting the expectation as
\[
\mathbb{E}[\|\nabla_{\mathcal{H}} \ell(Y, \Phi(X; \mathcal{H}, \mathbf{W})) - \ell(Y_n, \Phi(X_n; \mathcal{H}, \mathbf{W}_n))\| \\
\leq \|\nabla_{\mathcal{H}} \ell(Y, \Phi(X; \mathcal{H}, \mathbf{W}))\| + \|\nabla_{\mathcal{H}} \ell(Y_n, \Phi(X_n; \mathcal{H}, \mathbf{W}_n))\| \\
\leq \|\nabla_{\mathcal{H}} \ell(Y, \Phi(X; \mathcal{H}, \mathbf{W}))\| \\
+ \|\nabla_{\mathcal{H}} \ell(Y_n, \Phi(X_n; \mathcal{H}, \mathbf{W}_n))\| \\
\leq 2 F^2 L \sqrt{K} + \|\nabla_{\mathcal{H}} \ell(Y_n, \Phi(X_n; \mathcal{H}, \mathbf{W}_n))\| \\
\leq 2 F^2 L \sqrt{K}.
\]
(42)
Going back to (41), we can substitute the bound obtained in (42), take \( P(A_n) = 1 - \xi \), and use Lemma 5 to get
\[
\mathbb{E}[\|\nabla_{\mathcal{H}} \ell(Y, \Phi(X; \mathcal{H}, \mathbf{W})) - \ell(Y_n, \Phi(X_n; \mathcal{H}, \mathbf{W}_n))\| \\
\leq \xi 2 F^2 L \sqrt{K} + (1 - \xi) \sqrt{K} F^{L-1} \left( \frac{4 F^{L-1} L}{n} + 12 L^2 F^{2L-2} c \\
+ 3 L^2 F^{2L-2} \left( 1 + \frac{\pi B_{\mathbf{W}_n}}{\delta_{\mathbf{W}_n}} \right) \frac{2 \left( 1 + \sqrt{n \log(2n/\xi)} \right)}{n} \right) \frac{2 \left( 1 + \sqrt{n \log(2n/\xi)} \right)}{n}.
\]
Setting \( \xi = \frac{1}{\sqrt{n}} \) completes the proof.
APPENDIX B
PROOF OF LEMMA 1

To simplify notation we have denoted $n(\epsilon)$ as $n$ because this quantity remains constant throughout the proof.

Lemma 6: Under Assumptions 4, 5, and 6, the gradient of the loss function $\ell$ with respect to the GNN parameters $H$ is $A_{\nabla \ell}$-Lipschitz, with $A_{\nabla \ell} = (A_{\nabla \Phi} + A_{\Phi} F^{2L} \sqrt{K})$. Explicitly,

$$\|\nabla H \ell (Y, \Phi(X; A, W)) - \nabla H \ell (Y, \Phi(X; B, W))\| \leq A_{\nabla \Phi} \|A - B\|$$

(43)

Proof: We start by applying the chain rule to obtain

$$\|\nabla H \ell (Y, \Phi(X; A, W)) - \nabla H \ell (Y, \Phi(X; B, W))\| = \|\nabla H \ell (Y, \Phi(X; A, W)) - \nabla H \ell (Y, \Phi(X; B, W))\|$$

followed by the triangle and Cauchy-Schwartz inequalities to get

$$\|\nabla H \ell (Y, \Phi(X; A, W)) - \nabla H \ell (Y, \Phi(X; B, W))\| \geq \|\nabla H \ell (Y, \Phi(X; A, W))\|$$

$$\geq \|\nabla H \ell (Y, \Phi(X; B, W))\|$$

$$\leq \|\nabla H \ell (Y, \Phi(X; A, W))\|$$

Finally, we use Assumptions 1, 4, 5, and 6, and Lemma 2 to write

$$\|\nabla H \ell (Y, \Phi(X; A, W)) - \nabla H \ell (Y, \Phi(X; B, W))\| \leq \|\nabla H \ell (Y, \Phi(X; A, W))\|$$

completing the proof. □

Proof of Lemma 1: To start, we do as in [65], i.e., we define a continuous function $g(\epsilon)$ at $\epsilon = 1$ takes the value of the loss function on the graphon data at iteration $k + 1$ and at $\epsilon = 0$, the value at iteration $k$. Explicitly,

$$g(\epsilon) = \ell(Y, \Phi(X; H_k + \epsilon \eta_k \nabla H \ell(Y_n, \Phi(X_n; H_k, W_n))))$$

Function $g(\epsilon)$ is evaluated on the graphon data $Y, X, W$, but the steps are controlled by the induced graphon data $Y_n, X_n, W_n$. Applying the chain rule, the derivative of $g(\epsilon)$ with respect to $\epsilon$ can be written as

$$\frac{\partial g(\epsilon)}{\partial \epsilon} = -\eta_k \nabla H \ell(Y_n, \Phi(X_n; H_k; W_n))$$

$$\times \nabla H \ell(Y, \Phi(X; H_k + \epsilon \eta_k \nabla H \ell(Y_n, \Phi(X_n; H_k, W_n))))$$

Between iterations $k + 1$ and $k$, the difference in the loss function $\ell$ can be written as the difference between $g(1)$ and $g(0)$,

$$g(1) - g(0) = \ell(Y, \Phi(X; H_{k+1}, W)) - \ell(Y, \Phi(X; H_k, W))$$

Integrating the derivative of $g(\epsilon)$ in $[0, 1]$, we get

$$\ell(Y, \Phi(X; H_{k+1}, W)) - \ell(Y, \Phi(X; H_k, W)) = \int_0^1 \frac{\partial g(\epsilon)}{\partial \epsilon} d\epsilon$$

$$= -\eta_k \int_0^1 \frac{\partial g(\epsilon)}{\partial \epsilon} d\epsilon$$

$$= -\eta_k \int_0^1 \nabla H \ell(Y_n, \Phi(X_n; H_k; W_n)) d\epsilon$$

Factoring out $\eta_k$, we get

$$\ell(Y, \Phi(X; H_{k+1}, W)) - \ell(Y, \Phi(X; H_k, W))$$

$$\leq -\eta_k \int_0^1 \left\|\nabla H \ell(Y_n, \Phi(X_n; H_k, W_n))\right\|^2 d\epsilon$$

$$+ 2\eta_k \int_0^1 \nabla H \ell(Y_n, \Phi(X_n; H_k, W_n))^T \nabla H \ell(Y, \Phi(X; H_k, W)) d\epsilon$$

$$+ \frac{\eta_k^2 A_{\nabla \ell}}{2} \left\|\nabla H \ell(Y_n, \Phi(X_n; H_k, W_n))\right\|^2.$$
\[ 2\langle A, B \rangle, \]

Hence,

\[ \ell(Y, \Phi(X; H_{k+1}); W)) - \ell(Y, \Phi(X; H_{k}; W)) \leq -\frac{\eta_k}{2} \left( \| \nabla \ell(Y, \Phi(X; H_{k}; W)) \|^2 \right) \]

\[ -\| \nabla \ell(Y, \Phi(X; H_{k}; W)) - \nabla \ell(Y, \Phi(X; H_{k}; W)) \|^2 \]

\[ + \frac{\eta_k^2 A_{\text{act}} - \eta_k}{2} \| \nabla \ell(Y, \Phi(X; H_{k}; W)) \|^2. \]

Considering the first term on the right hand side, we know that the norm of the expected difference between the gradients is bounded by Theorem 1. Given that norms are positive, the inequality still holds when the elements are squared (if \( a > b, a \in \mathbb{R}_+, b \in \mathbb{R}_+ \), then \( a^2 > b^2 \)). Considering the second term on the right hand side, we impose the condition that \( \eta_k < \frac{1}{A_{\text{act}}} \), which makes this term negative. Taking the expected value of all the nodes completes the proof.

\[ \square \]

APPENDIX C

DETAILS ON COMPUTATION TIMES

All experiments were done on a computer with 32gb of RAM, a CPU Intel Core i9-9900 K @3.60 GHz × 16, a GPU GeForce RTX 2080 Ti/PCle/SSE2, and Ubuntu 18.04.4 LTS.

REFERENCES
