Convexified Graph Neural Networks for Distributed Control in Robotic Swarms

Saar Cohen , Noa Agmon
Department of Computer Science, Bar-Ilan University, Israel
saar30@gmail.com, agmon@cs.biu.ac.il

Abstract

A network of robots can be viewed as a signal graph, describing the underlying network topology with naturally distributed architectures, whose nodes are assigned to data values associated with each robot. Graph neural networks (GNNs) learn representations from signal graphs, thus making them well-suited candidates for learning distributed controllers. Oftentimes, existing GNN architectures assume ideal scenarios, while ignoring the possibility that this distributed graph may change along time due to link failures or topology variations, which can be found in dynamic settings. A mismatch between the graphs on which GNNs were trained and the ones on which they are tested is thus formed. Utilizing online learning, GNNs can be retrained at testing time, overcoming this issue. However, most online algorithms are centralized and work on convex problems (which GNNs scarcely lead to). This paper introduces novel architectures which solve the convexity restriction and can be easily updated in a distributed, online manner. Finally, we provide experiments, showing how these models can be applied to optimizing formation control in a swarm of flocking robots.

1 Introduction

Nowadays, data networks are everywhere to be found due to the arising networking revolution. Networks model entities in a wide variety of domains, such as communication networks, biology, and sociology. Unique to swarm robotics, data exhibits irregular and complex structures, provoking existing linear methods. Modeling data networks as graphs, whereas the data is viewed as a signal on top of it, *Graph Signal Processing* (GSP) [Ortega *et al.*, 2018] expands the concept of *graph convolutions*, which account for the underlying graph structure during the processing of data. Built upon the GSP's goal, *Graph Neural Networks (GNNs)* constitute nonlinear representation maps with capability of exploiting those underlying structures [Gama *et al.*, 2020]. As in most neural networks, GNNs are then trained by solving a non-convex Empirical Risk Minimization (ERM) problem, which is known to be NP-hard [Blum and Rivest, 1992].

Considering the problem of controlling a robotic swarm in dynamic settings with different initial positions and velocities, as well as topology changes, yields a difference in structures between training and testing due to the dynamic nature of the system. GNNs have been proven to have the fundamental properties of permutation equivariance and stability to changes in the topology [Gama *et al.*, 2020].

For improving the performance, *Online Learning* constitutes an appropriate solution, providing us with adaptive optimization algorithms suitable for continuously time-varying topologies [Dabbagh and Bannan-Ritland, 2005]. Yet, it requires the convexity of both the domain and the loss function. However, ERM on a nonlinear neural network model is generally non-convex, as on GNNs. Thus, this paper introduces *convexified* GNN architectures, which are well-suited for both online algorithms and real-world applications, in which data can be naturally represented as graphs (e.g., recommendation systems [Gao *et al.*, 2020]), while maintaining convergence guarantees. Following [Zhang *et al.*, 2017], we first illustrate a convex relaxation when regarding linear activation functions. For the nonlinear case, we then propose a relaxation to a suitably chosen *Reproducing Kernel Hilbert Space* (RKHS), reducing the problem to the linear case. Finally, we demonstrate in rigorous experiments the superiority of the convexified GNN solutions over existing solutions for formation control in a swarm of robots in terms of faster convergence rate, up to an expected generalization error.

2 Related Work

[Li *et al.*, 2019] concern with the problem of collision-free navigation in multi-robot systems, where the robots possess limited sensing and no global reference frame for localization. They compose a Convolutional Neural Network (CNN), which extracts adequate features from local observations, and a GNN that communicates these features among the robots. [Sartoretti *et al.*, 2019] assume no explicit communication. Yet, both researches are restricted to zero communication time-delay. Accordingly, [Tolstaya *et al.*, 2020] offer the use of Aggregation GNNs [Gama *et al.*, 2018a] (A-GNNs) to operate on time-varying graph processes. Both papers utilize *Imitation Learning*, and rely on the availability of an optimal solution (i.e., expert data). Furthermore, they each run one GNN per time instant, posing a high computational expense, which thus requires shallow GNNs and short filters.

Online learning is vastly explored in the context of neural networks [Li *et al.*, 2004; Hong *et al.*, 2015]. [Gao *et al.*, 2020] tackle the convexity requirement of online algorithms by proposing the *Wide and Deep GNN* (WD-GNN), consisting of two components: a *wide component* (which is a bank of graph filters), and a *deep component* (which is a GNN). They solely retrain the linear *wide* component, whereas the deep one remains fixed after the training phase. Yet, their solution is limited to the extent of linear transforms and does not involve online retraining of the GNN. As opposed to their work, we retrain the GNN part as well.

[Nikolentzos *et al.*, 2018]'s approach to the graph classification problem is similar to our own: they extract patches from each input graph via *community detection*, embed them with graph kernels and then feed them through a 1D CNN, to which pooling is applied. A fully-connected layer with a softmax completes the architecture. Community detection algorithms, which partition a network into clusters of densely connected nodes, are known to be computationally intractable and require a large storage space [Blondel *et al.*, 2008]. We thus leverage the graph signal processing (GSP) concepts for learning from graph signals (via GNNs) [Ortega *et al.*, 2018], rather than communities. Additionally, as opposed to their one-layered model, ours are multi-layered.

Notations. For brevity, $[n]$ denotes the discrete set $\{1, 2, \ldots, n\}$ for any $n \in \mathbb{N}_{>0}$. For a rectangular matrix A, let $||A||_*$ be its nuclear norm, and $||A||_2$ be its spectral norm (i.e., maximal singular value). Let $\mathcal{L}^2(\mathbb{N})$ denote the set of countable dimensional vectors v such that $\sum_{i=1}^{\infty} v_i < \infty$.

3 GNNs - Background

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ be the underlying graph structure, where: $\mathcal{V} = \{1, ..., n\}$ denotes the vertex set, $\mathcal{E}$ denotes the edge set and $\mathcal{W} : \mathcal{E} \to \mathbb{R}$ denotes the edge weight function, where a missing edge (i, j) is depicted by $\mathcal{W}(i, j) = 0$. The *neighborhood* of node $i \in \mathcal{V}$ is denoted by $N_i := \{j \in \mathcal{V} | (j, i) \in \mathcal{E}\}$. Following [Ortega *et al.*, 2018], we leverage the graph signal processing (GSP) concepts for deriving the foundation of learning from graph signals. Thus, the data $\mathcal{X} \in \mathbb{R}^{n \times F}$ is modeled as a *graph signal* where $[\mathcal{X}]_i = x_i$ constitutes an F-dimensional *feature vector*, assigning a *state* for node $i \in \mathcal{V}$. In the context of swarm robotics, this state is typically described by the agent's position, velocity or acceleration. For the sake of relating the the graph signal $\mathcal{X}$ with the underlying structure of $\mathcal{G}$, we define a *graph shift operator* (GSO) by $\mathcal{S} \in \mathbb{R}^{n \times n}$, which correlates to the graph's sparsity and satisfies $[\mathcal{S}]_{ij} = s_{ij} = 0$ if $(j, i) \notin \mathcal{E}$ for $j \neq i$. Commonly utilized GSOs include the **adjacency** matrix [Sandryhaila and Moura, 2013; Sandryhaila and Moura, 2014], which corresponds to either directed or undirected graphs, the **Laplacian** matrix [Shuman *et al.*, 2013], which solely applies to *undirected* graphs, and their normalized counterparts [Defferrard *et al.*, 2016; Gama *et al.*, 2018b]. Hence, we clarify that the consideration of either a directed or an undirected graph is with close proximity to the choice of the GSO. The graph signal $\mathcal{X}$ can be *shifted* over the nodes by viewing $\mathcal{S}$ as a linear operator, yielding that the output at node i for the f^{th} feature becomes:

$$[\mathcal{S}\mathcal{X}]_{if} = \sum_{j=1}^{n} [\mathcal{S}]_{ij} [\mathcal{X}]_{jf} = \sum_{j \in N_i} s_{ij} x_{jf} \tag{1}$$

The second equality thereby emphasizes the sparsity nature (or, more accurately, locality) of $\mathcal{S}$. We shall observe that, whereas $\mathcal{S}\mathcal{X}$ corresponds to the local information exchange between a given node and its direct neighbors, applying this linear operator recursively yields information from nodes in further hops along the graph, i.e., the *k-shifted signal* $\mathcal{S}^k\mathcal{X}$ constitutes the aggregation of information from nodes which are k-hops away. Hence, given a set of *filter taps* $H := \{H_k \in \mathbb{R}^{F \times G}\}_{k=0}^{K}$, a *graph (convolutional) filter* [Ortega *et al.*, 2018] with G output features can be defined as a linear combination of data located at consecutive hops:

$$\Psi(\mathcal{X}; \mathcal{S}, H) := \sum_{k=0}^{K} \mathcal{S}^k \mathcal{X} H_k \tag{2}$$

where the output is another graph signal of order $n \times G$, over the same graph. Feeding a graph filter into a pointwise nonlinear activation function yields a **Graph Neural Network** (GNN) [Gama *et al.*, 2020]. It is defined as a nonlinear mapping between graph signals $\Phi : \mathbb{R}^{n \times F} \to \mathbb{R}^{n \times G}$ given by:

$$\Phi(\mathcal{X}; \mathcal{S}, H) = \mathcal{X}_L; \quad \mathcal{X}_\ell := \sigma(\Psi(\mathcal{X}_{\ell-1}; \mathcal{S}, H_\ell)) \tag{3}$$

where $\ell \in [L]$. $\sigma : \mathbb{R} \to \mathbb{R}$ is a pointwise non-linear activation function (which, for brevity, denotes its entrywise application in (3)), and $H := \{H_{\ell k} = (h_{\ell k}^{fg}) \in \mathbb{R}^{F_{\ell-1} \times F_\ell} | k \in [K] \cup \{0\}, \ell \in [L]\}$ is a set of filter taps. The graph signal $\mathcal{X}_\ell \in \mathbb{R}^{n \times F_\ell}$ is referred to as a *graph perceptron* [Gama *et al.*, 2020], occupying the ℓ^{th} layer of a *multi-layer graph perceptron* generated by the cascade of all L graph signals above. The input is $\mathcal{X}_0 = \mathcal{X}$, yielding $F_0 = F$, where the output has $F_L = G$ features. Each input feature $x_{\ell-1}^g$ ($g \in [F_{\ell-1}]$) at layer ℓ is processed in parallel by F_ℓ graph filters to output the following features. Given some positive constants R_ℓ, which can be viewed as communication power constraints, the *non-convex* function class of GNNs is provided by:

$$\mathcal{F}_{gnn} := \{\Phi \text{ as in (3)} : \max_{0 \leq k \leq K} h_{\ell k}^{fg} \leq R_\ell \; \forall f, g\} \tag{4}$$

3.1 Empirical Risk Minimization (ERM)

We train the GNN (3) by solving the following empirical risk minimization (ERM) problem for some loss function $J : \mathbb{R}^{n \times G} \to \mathbb{R}$ and a training set $T := \{\mathcal{X}^{(j)}\}_{j=1}^{m}$:

$$\Phi_{gnn}^* \in \operatorname*{arg\,min}_{\Phi \in \mathcal{F}_{gnn}} \frac{1}{m} \sum_{j=1}^{m} J(\Phi(\mathcal{X}^{(j)}; \mathcal{S}, H)) \tag{5}$$

The number of parameters H to be trained is determined by the number of: layers L, filter taps K and features F_ℓ in the ℓ^{th} layer. Abusing the notation slightly, we regard (5) as capable of coping with supervised problems, due to the possible extension of J to operate on the optimal action as well.

4 GNNs Convexification

Since $\mathcal{F}_{gnn}$ (4) forms a non-convex set and due to the non-linearity of the activation function, the ERM problem on GNNs is non-convex, yielding that: (1) their training process is NP-hard, and (2) they are unsuitable for online learning. Hence, we aim at overcoming this issue by *convexifying* both the ERM's predictions domain and loss function. Following the method put forth by [Zhang *et al.*, 2017] for Convolutional Neural Networks (CNNs), we develop the class of **Convexified GNNs (Cx-GNNs)**. As a first step, in Section 4.1 we illustrate a convex relaxation of a low-rank constraint based on the nuclear norm when regarding linear activation functions. In the nonlinear case (Section 4.2), we propose relaxation to a suitably chosen *Reproducing Kernel Hilbert Space* (RKHS), where the resulting relaxed filters satisfy a low-rank constraint, reducing the problem to the linear case.

4.1 Linear Activation Functions

In this section, we employ the nuclear norm in the simple case of the linear activation function $\sigma(w) = w$ (yielding $x_\ell^f = u_\ell^f$ due to (3)), which provides us with a convex function class, as opposed to (4). Indeed, consider the eigenvector decomposition of the shift operator $\mathcal{S} = V\Lambda V^T$, with orthogonal eigenvectors matrix $V := [v_i]_{i=1}^n$ and distinct eigenvalues matrix $\Lambda := diag[\lambda_i]_{i=1}^n$, in ascending order. For each filter ($1 \leq f \leq F_\ell, 1 \leq g \leq F_{\ell-1}$):

$$H_\ell^{fg}(\mathcal{S}) = H_\ell^{fg}(V\Lambda V^T) = \sum_{k=0}^{K} h_{\ell k}^{fg} V\Lambda^k V^T = V H_\ell^{fg}(\Lambda) V^T \quad (6)$$

where $H_\ell^{fg}(\Lambda) = diag[h_\ell^{fg}(\lambda_i)]$ for $h_\ell^{fg} : \mathbb{R} \to \mathbb{R}$ given by $h_\ell^{fg}(\lambda) = \sum_{k=0}^{K} h_{\ell k}^{fg} \lambda^k$, which satisfies (from (4)):

$$h_\ell^{fg}(\lambda) \leq R_\ell \sum_{k=0}^{K} \lambda^k =: R_\ell s_K(\lambda) \quad (7)$$

Note that $\{h_\ell^{fg}(\lambda_i) v_i\}_{i=1}^n$ would be less than n if and only if λ_i is a root of the polynomial $h_\ell^{fg}(\lambda)$, i.e., $rk(H_\ell^{fg}(\Lambda)) \leq n$. From (3) and (6), x_ℓ^f can be written as follows:

$$x_\ell^f = \sum_{g=0}^{F_{\ell-1}} V H_\ell^{fg}(\Lambda) V^T x_{\ell-1}^g = tr(\mathcal{X}_{\ell-1} \tilde{H}_\ell^f) \quad (8)$$

where, in the last step, we have defined the $F_{\ell-1} \times F_{\ell-1}$ block matrix $\tilde{H}_\ell^f := diag[V H_\ell^{fg}(\Lambda) V^T]_{g=0}^{F_{\ell-1}}$. Hence, x_ℓ^f linearly depends on $\tilde{H}_\ell^f$, whose rank satisfies ($rk(H_\ell^{fg}(\Lambda)) \leq n$):

$$rk(\tilde{H}_\ell^f) = \sum_{g=0}^{F_{\ell-1}} rk(H_\ell^{fg}(\Lambda)) \leq n F_{\ell-1} \quad (9)$$

Let $\tilde{H}_\ell := [\tilde{H}_\ell^f]_{f=1}^{F_\ell}$. From (9), we clearly infer:

$$rk(\tilde{H}_\ell) \leq n F_\ell F_{\ell-1} \quad (10)$$

whereas $\mathcal{X}_\ell$ can be written as:

$$\mathcal{X}_\ell = [tr(\mathcal{X}_{\ell-1} \tilde{H}_\ell^f)]_{f=1}^{F_\ell} =: \Phi^{\tilde{H}_\ell}(\mathcal{X}_{\ell-1}) \quad (11)$$

As in [Zhang *et al.*, 2017], the nuclear norm of $\tilde{H}_\ell$ is a convex relaxation of its rank constraint. In the supplementary material, we provide a rigorous proof, yielding:

$$||\tilde{H}_\ell||_* \leq R_\ell F_\ell F_{\ell-1} \sqrt{tr\Big(\Big(\sum_{k=0}^{K} \mathcal{S}^k\Big)^2\Big)} =: \mathcal{B}_\ell \quad (12)$$

Hence, defining the function class for Cx-GNN:

$$\mathcal{F}_{cx\text{-}gnn} := \{\Phi^{\tilde{H}_\ell} \text{ as in (11)} : ||\tilde{H}_\ell||_* \leq \mathcal{B}_\ell\} \quad (13)$$

it is guaranteed that $\mathcal{F}_{gnn} \subseteq \mathcal{F}_{cx\text{-}gnn}$. Consequently, solving the ERM (5) over $\mathcal{F}_{cx\text{-}gnn}$ instead of $\mathcal{F}_{gnn}$ yields a convex optimization problem over a wealthier class of functions.

4.2 Nonlinear Activation Functions

Following [Zhang *et al.*, 2017], for certain nonlinear activation functions σ and suitably chosen kernel functions $\mathcal{K}$, we prove that the class of GNNs can be relaxed to a *Reproducing Kernel Hilbert Space* (RKHS), reducing the problem to the linear activation case. **Full proofs are omitted due to space constraints, and can be found in the supplementary material (https://u.cs.biu.ac.il/~agmon/IJCAI21Sup.pdf)**. Furthermore, readers should refer to [Schölkopf *et al.*, 2001] for a brief overview on RKHS.

Let $h_\ell^{fg} : \mathbb{R} \to \mathbb{R}$ be given by $h_\ell^{fg}(\lambda) = \sum_{k=0}^{K} h_{\ell k}^{fg} \lambda^k$. Let $\mathcal{K} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be a positive semidefinite kernel function. We are able to show that ($w_\ell^f(i) := \sum_{g=0}^{F_{\ell-1}} h_\ell^{fg}(\lambda_i) e_i$):

$$u_\ell^f = (\langle y_\ell^f, w_\ell^f(i) \rangle)_{i=1}^n; \quad y_\ell^f := \sum_{g=0}^{F_{\ell-1}} x_{\ell-1}^g \quad (14)$$

Hence, regarding certain choices of kernels and a sufficiently smooth σ, the function $\tau_\ell^f : z \mapsto \sigma((\langle z, w_\ell^f(i) \rangle)_{i=1}^n)$ is contained within the RKHS induced by the kernel function $\mathcal{K}$. From the *representer theorem*, provided a sufficiently smooth σ, and in regard with a certain choice of kernel $\mathcal{K}$, there exists a feature mapping $\varphi : \mathbb{R}^n \to \mathcal{L}^2(\mathbb{N})$ for which $\mathcal{K}(z, z') = \langle \varphi(z), \varphi(z') \rangle$. From (14) ($i \in [n]$):

$$\sigma(\langle z, w_\ell^f(i) \rangle) \equiv \langle \varphi(z), \varphi(w_\ell^f(i)) \rangle =: \langle \varphi(z), \bar{w}_\ell^f(i) \rangle \quad (15)$$

where $\bar{w}_\ell^f(i) \in \mathcal{L}^2(\mathbb{N})$ is a countable-dimensional vector, due to the fact that φ itself is a countable sequence of functions. Regarding the supplementary material, we have that $||\bar{w}_\ell^f(i)||_2 \leq C_\sigma(||w_\ell^f(i)||_2)$, provided a monotonically increasing C_σ which depends on $\mathcal{K}$. Consequently, φ may be utilized as the vectorized representation of the output features u_ℓ^f, where $\bar{w}_\ell^f(i)$ constitutes the linear graph filter taps, thus reducing the problem to training a GNN with the identity activation function. That is, each graph perceptron is now parametrized by the countable-dimensional vector $\bar{w}_\ell^f(i)$.

As our next step, we shall provide a reduction of the original ERM problem to a finite-dimensional one. Output on the training data $T := \{\mathcal{X}^{(j)}\}_{j=1}^m$ should be solely considered when solving the ERM problem, i.e., the output of $\langle \varphi(y_\ell^f(j)), \bar{w}_\ell^f(i) \rangle$ (where $y_\ell^f(j)$ is as given in (14) for $\mathcal{X}_0 = \mathcal{X}^{(j)}$). Without loss of generality, for the sake of solving the

ERM, we assume that $\bar{w}_\ell^f(i) \in span\{\varphi(y_\ell^{f'}(j))|(j,f') \in [m]\times[F_\ell]\}$. We can thus reparameterize it by:

$$\bar{w}_\ell^f(i) = \sum_{(j,f')\in[m]\times[F_\ell]} \alpha_{\ell,j}^{f'}(i)\varphi(y_\ell^{f'}(j)) \quad (16)$$

Let $\alpha_\ell^f(i) := [\alpha_{\ell,j}^{f'}(i)]_{(j,f')\in[m]\times[F_\ell]}$. Hence, for estimating $\bar{w}_\ell^f(i)$, it suffices to estimate the vector $\alpha_\ell^f(i)$. By definition, the relation $(\alpha_\ell^f(i))^T K_\ell^f \alpha_\ell^f(i) = ||\bar{w}_\ell^f(i)||_2^2$ holds, where $K_\ell^f \in \mathbb{R}^{mF_\ell\times mF_\ell}$ denotes the symmetric kernel matrix, whose rows and columns are indexed by the example-state index $(j,f') \in [m]\times[F_\ell]$. The entry at row (j',f') and column (j'',f'') equals to $\mathcal{K}(y_\ell^{f'}(j'), y_\ell^{f''}(j''))$. For a suitable approximation of K_ℓ^f such that $K_\ell^f \approx QQ^T$ for $Q \in \mathbb{R}^{mF_\ell\times p}$, let Q^+ be the pseudo-inverse of Q. Let $v_\ell(z) \in \mathbb{R}^{mF_\ell}$ be a vector whose $(j,f)^{th}$ coordinate equals to $\mathcal{K}(z, y_\ell^f(j))$. Denoting $\mathcal{A}_\ell := [\mathcal{A}_\ell^f]_{f=1}^{F_\ell}$, the **Cx-GNN** is given as follows (in regard with (3) and (14)):

$$\tilde{\mathcal{X}}_\ell = [tr(\mathcal{A}_\ell^f \tilde{y}_\ell^f)]_{f=1}^{F_\ell} =: \Phi^{\mathcal{A}_\ell}(\mathcal{X}_{\ell-1}) \quad (17)$$

$$\tilde{y}_\ell^f := Q^+ v_\ell(y_\ell^f); \tilde{w}_\ell^f(i) := Q^T\alpha_\ell^f(i); \mathcal{A}_\ell^f := [\tilde{w}_\ell^f(i)]_{i=1}^n \quad (18)$$

Remark 1. *Note that:* $\tilde{y}_\ell^f(j) = Q^+ v_\ell(y_\ell^f(j)) = [Q]_{j,f}$.

The Cx-GNN can also be depicted by:

$$\tau_\ell^f(z) := (\langle Q^+ v_\ell(z), Q^T\alpha_\ell^f(i)\rangle)_{i=1}^n \quad (19)$$

It can thus be proven that:

$$||\mathcal{A}_\ell||_* \leq \sum_{i=0}^{n} F_\ell C_\sigma(R_\ell F_{\ell-1} s_K(\lambda_i)) =: \tilde{\mathcal{B}}_\ell \quad (20)$$

4.3 Learning Multi-Layer Cx-GNNs

Alg.1 is an abbreviated version for learning a multi-layer Cx-GNN. At each layer, given regularization parameters $\{\mathcal{R}_\ell\}_{\ell=1}^L$, solving the following optimization problem is required (from (17)):

$$\hat{\mathcal{A}}_\ell \in \underset{||\mathcal{A}_\ell||_*\leq\mathcal{R}_\ell}{\arg\min}\ \tilde{J}(\mathcal{A}_\ell); \quad \tilde{J}(\mathcal{A}_\ell) := \frac{1}{m}\sum_{j=1}^{m} J(\tilde{\mathcal{X}}_\ell^{(j)}) \quad (21)$$

The naive approach is solving via *projected gradient descent*: At iteration t, for a step size $\eta^t > 0$, we compute:

$$\mathcal{A}_\ell^{t+1} = \Pi_{\mathcal{R}_\ell}(\mathcal{A}_\ell^t - \eta^t \nabla_{\mathcal{A}} \tilde{J}(\mathcal{A}_\ell^t)) \quad (22)$$

where $\Pi_{\mathcal{R}_\ell}$ denotes the Euclidean projection onto the nuclear norm ball with radius $\mathcal{R}_\ell$. As in [Zhang *et al.*, 2017], this can be done by projecting the vector of singular values onto the ℓ_1-ball, and then reconstructing the matrix using the projected singular values. Afterwards, the ℓ^{th} layer output shall be retrieved: they propose to compute the singular value decomposition $\hat{\mathcal{A}}_\ell = U_\ell\Sigma_\ell W_\ell^T$, and then define $\tilde{U}_\ell$ to be the first F_ℓ columns of U_ℓ. The resulting output, whose $(f,i)^{th}$ element equals to $\langle Q^+ v_\ell(z), Q^T\alpha_\ell^f(i)\rangle$, is given by:

$$\tilde{\mathcal{H}}_\ell(z) := \tilde{U}_\ell^T \tilde{\mathcal{Y}}_\ell(z); \quad \tilde{\mathcal{Y}}_\ell(z) := Q^+ v_\ell(z) \quad (23)$$

Algorithm 1 Learning Multi-Layer Cx-GNNs

Input: Samples $\{\mathcal{X}^{(j)}\}_{j=1}^m$, kernel $\mathcal{K}$, $\{\mathcal{R}_\ell\}_{\ell=1}^L$, number of graph filters $\{F_\ell\}_{\ell=1}^L$, $\tilde{\mathcal{H}}_1(z) = z$

1: **for** $2 \leq \ell \leq L$ **do**
2: Taking $\{\tilde{\mathcal{H}}_{\ell-1}(\mathcal{X}^{(j)})\}_{j=1}^m$ as training examples, construct a kernel matrix $K_\ell^f \in \mathbb{R}^{mF_\ell\times mF_\ell}$, such that the entry at row (j',f') and column (j'',f'') equals to $\mathcal{K}(y_\ell^{f'}(j'), y_\ell^{f''}(j''))$. Compute a suitable approximation of K_ℓ^f such that $K_\ell^f \approx QQ^T$ for $Q \in \mathbb{R}^{mF_\ell\times p}$.
3: Compute $\tilde{\mathcal{Y}}_\ell^{(j)} := [\tilde{y}_\ell^f(j)]_{f=1}^{F_\ell}$ as in Remark 1 for $\mathcal{X}^{(j)}$.
4: Solve (21) so as to obtain a matrix $\hat{\mathcal{A}}_\ell$.
5: Compute the singular value decomposition $\hat{\mathcal{A}}_\ell = U_\ell\Sigma_\ell W_\ell^T$, and define $\tilde{U}_\ell$ as the first F_ℓ columns of U_ℓ.
6: Compute $\tilde{\mathcal{P}}_\ell(z) := [tr(\hat{\mathcal{A}}_\ell^f \tilde{\mathcal{Y}}_\ell(z))]_{f=1}^{F_\ell}$ and $\tilde{\mathcal{H}}_\ell(z)$ (23).
7: **end for**

Output: Predictor $\tilde{\mathcal{P}}_L$, output $\tilde{\mathcal{H}}_L$ and parameters $\hat{\mathcal{A}}_L$.

The time complexity is affected by p. For instance, by *Nyström method* [Williams and Seeger, 2001], an approximation of K_ℓ^f is achieved by randomly sampling $p \ll mF_\ell$ rows/columns from the original K_ℓ^f in $\mathcal{O}(p^2 mF_\ell)$ time.

4.4 Convergence Analysis

As in Theorem 1 of [Zhang *et al.*, 2016], we prove the following theorem, from which we infer that the role R_ℓ plays is twofold: it ensures that $\mathcal{F}_{gnn} \subseteq \mathcal{F}_{cx\text{-}gnn}$ and $\mathcal{F}_{cx\text{-}gnn}$ is a wealthier class of functions, but not "too big". We denote $y_\ell^f(z) := \sum_{g=0}^{F_{\ell-1}} z^g$. We use the random kernel matrix $K(\mathcal{X}_{\ell-1}) \in \mathbb{R}^{F_\ell} \times \mathbb{R}^{F_\ell}$, whose $(f,f')^{th}$ entry is given by $\mathcal{K}(y_\ell^f, y_\ell^{f'})$, where y_ℓ^f is as in (3) and the ℓ^{th} layer input $\mathcal{X}_{\ell-1} \in \mathbb{R}^{n\times F_{\ell-1}}$ is drawn randomly.

Theorem 1. *Let $J(\cdot)$ be L-Lipchitz continuous and $\mathcal{K}$ be either the inverse polynomial kernel or the Gaussian kernel. For any valid activation function σ, in regard with the radius $\tilde{\mathcal{B}}_\ell$ as in* (20)*, the expected generalization error satisfies:*

$$\mathbb{E}_{\mathcal{X}}[J(\tilde{\mathcal{P}}_\ell(\mathcal{X}))] \leq \inf_{\Phi\in\mathcal{F}_{gnn}} \mathbb{E}_{\mathcal{X}}[J(\Phi(\mathcal{X};\mathcal{S},H))] +$$

$$+ \frac{cL\tilde{\mathcal{B}}_\ell\sqrt{\log(mF_\ell)\mathbb{E}[||K(\mathcal{X}_{\ell-1})||_2]}}{\sqrt{m}} \quad (24)$$

at layer ℓ and for a universal constant $c > 0$.

Proof. (*Sketch*) Let $\tilde{\mathcal{B}}_{\ell i}^f := C_\sigma((R_\ell^{fg} F_{\ell-1} s_K(\lambda_i))^2)$. While regarding the ℓ^{th} layer, the proof comprises of two parts: First, a wealthier function class shall be considered, which contains the class of GNNs. It is defined as follows:

$$\mathcal{F}_{cx\text{-}gnn} := \{z \mapsto [(\langle y_\ell^f(z), w_\ell^f(i)\rangle)_{i=1}^{n^*}]_{f=1}^{F_\ell} : n^* < \infty$$

$$\text{and } ||w_\ell^f(i)||_{\mathcal{H}} \leq \tilde{\mathcal{B}}_{\ell i}^f\ \forall i, f\} \quad (25)$$

where $||\cdot||_{\mathcal{H}}$ is the associated RKHS norm. We prove that the predictor $\tilde{\mathcal{P}}_\ell$, as computed by Alg.1, solves the ERM problem for $\mathcal{F}_{cx\text{-}gnn}$. Afterwards, we provide an upper bound on its *Rademacher complexity*. After proving $\mathcal{F}_{gnn} \subseteq \mathcal{F}_{cx\text{-}gnn}$, we infer that the generalization loss of $\tilde{\mathcal{P}}_\ell$ is bounded by that of GNNs. Combined with the theory of Rademacher complexity [Bartlett and Mendelson, 2002], this loss converges to the least possible error of $\mathcal{F}_{cx\text{-}gnn}$, establishing (24). □

5 Extensions of Cx-GNNs

In this section, we extend the model proposed in Section 4 to other existing architectures found in the literature.

As a first step, we consider **Aggregation GNNs (A-GNNs)**, which apply to a signal with temporal structure, incorporating the topology of the graph [Gama *et al.*, 2018a]. Denoting the ℓ^{th} *aggregation sequence* at node i by:

$$\mathcal{Z}_i^{(\ell)} := [[\mathcal{Y}_k^{(\ell)}]_i]_{k=0}^K; \quad \mathcal{Y}_k^{(\ell)} := \mathcal{S}^k \mathcal{X}_\ell \tag{26}$$

the ℓ^{th} layer output can be thus expressed as:

$$\zeta(\mathcal{X}; \mathcal{S}, H) := \mathcal{Z}_i^{(L)}; \quad \mathcal{Z}_i^{(\ell)} = \sigma(\mathcal{Z}_i^{(\ell-1)} H_\ell) \tag{27}$$

As in Section 4, leveraging the concept of convexified CNNs [Zhang *et al.*, 2017], the class of **Convexified A-GNNs (CA-GNNs)** can be readily formed. Indeed, for a linear activation function, regarding (27), we infer that $\mathcal{Z}_i^{(\ell)}$ linearly depends on H_ℓ, satisfying: $rk(H_\ell) \leq k \min(F_{\ell-1}, F_\ell)$. As in Section 4.1, we prove that the nuclear norm of H_ℓ is bounded. Noting $||H_\ell||_* = \sum_{f=1}^{F_\ell} ||H_\ell^f||_*$, those nuclear norms can be bounded as follows:

$$||H_\ell^f||_* \leq \sum_{g=0}^{F_{\ell-1}} ||H_\ell^{fg}||_* := \sum_{g=0}^{F_{\ell-1}} tr(\sqrt{(H_\ell^{fg})^T H_\ell^{fg}}) =$$

$$= \sum_{g=0}^{F_{\ell-1}} \sqrt{\sum_{k=1}^{K} (h_{\ell k}^{fg})^2} \leq R_\ell F_{\ell-1} \sqrt{K} \tag{28}$$

where the last step stems from (4). Consequently:

$$||H_\ell||_* \leq R_\ell F_\ell F_{\ell-1} \sqrt{K} =: \mathcal{C}_\ell \tag{29}$$

Hence, defining the function class for CA-GNN:

$$\mathcal{F}_{ca\text{-}gnn} := \{\zeta \text{ as in (27)} : ||H_\ell||_* \leq \mathcal{C}_\ell \ \forall \ell\} \tag{30}$$

it is guaranteed that $\mathcal{F}_{gnn} \subseteq \mathcal{F}_{ca\text{-}gnn}$. Consequently, solving the ERM (5) over $\mathcal{F}_{ca\text{-}gnn}$ instead of $\mathcal{F}_{gnn}$ yields a convex optimization problem over a wealthier class of functions.

As in Subsection 4.2, for certain nonlinear activations and suitably chosen kernel functions $\mathcal{K} : \mathbb{R}^K \times \mathbb{R}^K \to \mathbb{R}$, we are able to show that $z \mapsto \sigma(zH_\ell)$ is contained in the RKHS induced by $\mathcal{K}$. Alg.1 can then be applied.

Now, we regard a variant of A-GNN, referred to as **Time-Delayed A-GNN**. At time t, we denote the *time-delayed aggregation sequence* [Tolstaya *et al.*, 2020] of i by:

$$\mathcal{Z}_{it} := [[\mathcal{Y}_{kt}]_i]_{k=0}^K; \quad \mathcal{Y}_{kt} := \mathcal{S}_{t,k} \mathcal{X}_{t-k} \tag{31}$$

where $\mathcal{S}_{t,k} = \prod_{\tau=t-k+1}^{t} \mathcal{S}_\tau$ for $t-k \geq 0$, and $\mathcal{S}_{t,k} \equiv 0$ otherwise. $\mathcal{S}_{t,k}$ constitutes the aggregation of information from nodes which are k-hops away at time $t-k$. The ℓ^{th} layer output of a **Time-Delayed A-GNN (TA-GNN)** is given by:

$$u_i^t := \mathcal{Z}_{it}^{(L)}; \ \mathcal{Z}_{it}^{(\ell)} = \sigma(\mathcal{Z}_{it}^{(\ell-1)} H_\ell), \ \mathcal{Z}_{it}^{(0)} := \mathcal{Z}_{it} \tag{32}$$

As for CA-GNNs, the class of **Convexified Time-Delayed A-GNNs (CTA-GNNs)** can be equivalently formed.

Finally, following [Gao *et al.*, 2020], in this section we propose the **Half-Convex GNN** (HC-GNN) model. For Φ as in (3), a convex GNN ζ, and combination weights $\alpha_{gnn}, \alpha_{cx\text{-}gnn}$, it is given as follows:

$$\Upsilon(\mathcal{X}; \mathcal{S}, H) = \alpha_{gnn} \Phi(\mathcal{X}; \mathcal{S}, H) + \alpha_{cx\text{-}gnn} \zeta(\mathcal{X}; \mathcal{S}, H) \tag{33}$$

An **Aggregation HC-GNN** (AHC-GNN) is an HC-GNN which comprises of an A-GNN and a CA-GNN. A **Time-Delayed AHC-GNN** (TAHC-GNN) is defined similarly.

6 Distributed Online Convex Optimization

The convexity of our suggested models along with the bounded convergence time (as proven in Theorem 1) allows us to apply them as a *distributed* online algorithm, to tackle the problem of time-varying topologies and handling the computational expense of running one GNN per time instant. Let $\hat{\mathcal{A}}_L$ be the parameters provided by the training phase. At time t, each node $i \in \mathcal{V}$ has access to a local loss $J_i^t(\Phi^{\mathcal{A}_i}(\mathcal{X}_t))$, where $\mathcal{X}_t$ is the observed signal. Nodes should thus aim at minimizing the sum of local losses, s.t. $\mathcal{A}_i = \mathcal{A}_q \equiv \mathcal{A} \ \forall i, q : q \in \mathcal{N}_i$. (21) can thus be altered as:

$$\{\hat{\mathcal{A}}_i\}_{i=1}^n \in \underset{\{\mathcal{A}_i\}_{i=1}^n}{\arg\min} \sum_{i=1}^{n} J_i^t(\Phi^{\mathcal{A}_i}(\mathcal{X}_t)) \tag{34}$$

Due to the convexity of our proposed architectures, they can thus be retrained. Setting $\mathcal{A}_i^0 = \hat{\mathcal{A}}_L$, each node i updates its local parameters in respect with:

$$\mathcal{A}_i^{t+1} = \mathcal{A}_i^t - \eta^t \nabla_{\mathcal{A}} J_i^t(\Phi^{\mathcal{A}_i}(\mathcal{X}_t)) \tag{35}$$

Literally, each $\mathcal{A}_i^t$ is descended along the local gradient to approach the optimal solution of (34).

7 Flocking Model

Given n robots moving with random initial velocities sampled in the interval $[-v, v]^2$, at time t, each robot $i \in \mathcal{N}$ is characterized by $p_i^t, v_i^t, u_i^t \in \mathbb{R}^2$, which are its position, velocity and acceleration (respectively). We aim to derive controllers $\mathcal{U}_t := [u_i^t]_{i=1}^n$, for the robots to converge to a common velocity without collisions. [Tolstaya *et al.*, 2020] provide a *centralized* optimal controller, which requires access to all agent velocities and positions. In the *distributed* settings, at time t, let $N_i^t := \{q \in \mathcal{N} : ||p_i^t - p_q^t||_2 \leq R\}$ be the neighborhood of robot i with *visibility radius* R. A communication graph $\mathcal{G}_t$ is thus established in respect with Section 3, whereas the GSO $\mathcal{S}_t$ is the adjacency matrix.

As in [Tolstaya *et al.*, 2020], GNNs can be utilized for learning a distributed controller $\mathcal{U}_t = \Phi^{\mathcal{A}}(\mathcal{X}_t)$, with a graph signal $\mathcal{X}_t$, which linearly aggregates position and velocity information of neighboring robots. At testing time, they measure performance by the velocity variation among robots over the whole trajectory and also at the final time step (a detailed flocking model can be found in the supplementary material).

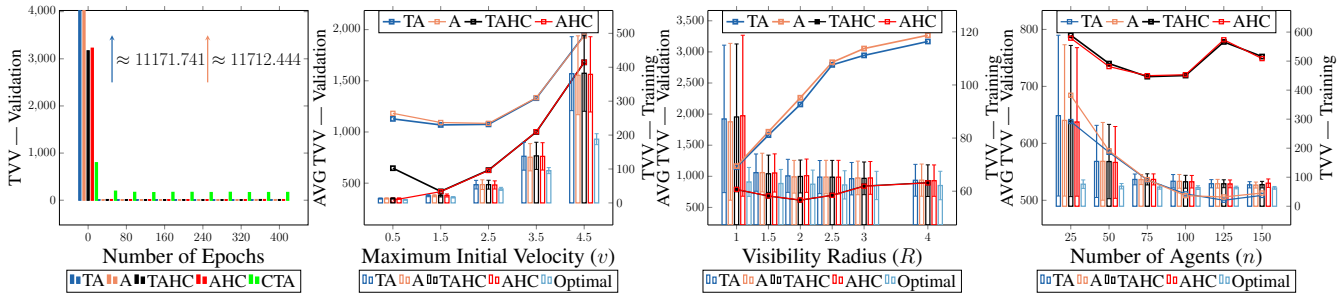

Figure 1: Fig.1a provides a comparison with respect to the **total velocity variation (TVV)** relative to the optimal controller over the validation set, along the training phase. Figs.1b-1d depict the change in the average TVV over the validation set and over the training set (bar charts), both relative to the optimal controller, with respect to varied values of: v (1b), R (1c), and n (1d).

8 Empirical Analysis on Flocking Swarms

In this section, our methods are evaluated through the task in which robots aim to align their velocities and regulate their spacing. We compare our models to architectures found in the literature, where the later are solely trained offline.

8.1 Experimental Setup

Evaluations [Cohen and Agmon, 2021] were conducted using a 12GB NVIDIA Tesla K80 GPU, implemented in PyTorch v1.7.0, accelerated with Cuda v10.1, and situated in the GymFlock [Tolstaya *et al.*, 2020] flocking environment. For training each type of GNN, the Dataset Aggregation (DAgger) algorithm is used, following the learner's policy instead of the expert's with probability $1-\beta$ when collecting training trajectories [Ross *et al.*, 2011], where β is decayed by a factor of 0.993 to a minimum of 0.5. The ADAM optimizer is used with learning rate $5 \cdot 10^{-4}$, decaying factors $0.9, 0.999$, and a MSE cost function. The dataset contains 400 trajectories for training, 40 for validation and 40 for testing, each of length 200 total. All GNNs consist of two hidden layers, with 32 neurons each, where K=3, σ= $\tanh$ for non-convex GNNs and σ= $\sin$ for convex GNNs. For our baseline scenario, we consider R=$1m$, v=$3m/s$, N=100. The robots' locations were initialized uniformly on a disc of radius $\sqrt{n}$ to normalize the density of agents for changing flock sizes. α_{gnn}=1, $\alpha_{cx\text{-}gnn}$=0.001 is chosen for all HC-GNNs.

8.2 Results

We first compare both A-GNNs and TA-GNNs to their Half-Convex counterparts and a CTA-GNN (Fig.1a). After a *single* epoch, our models exhibit the best performance, whereas the CTA-GNN outperforms them all. This behaviour is attributed to the fact that A-GNNs and TA-GNNs are trained by solving a non-convex ERM, which is an NP-hard process. However, during the upcoming epochs, the CTA-GNN is outperformed by all the other models, which can be explained by Theorem 1. We also note that the HC-GNNs achieve performance competitive with the non-convex GNNs. This stems from the magnified representation power attained when utilizing a linear combination of a GNN and its convex counterpart: the GNN ensures that robots approach a perfect velocity consensus, whereas the convexified GNN provides us with a faster convergence.

We now provide a comparison in regard with different flocking scenarios, which are displayed by different initial conditions. Figs.1b-1d depict the change in the average total velocity variation over the validation set and the total velocity variation over the training set (bar charts), with respect to varied values of: v (1b), R (1c), and n (1d). More accurately, in each scenario, we fix the parameters of the baseline scenario, and solely alter the examined one. We observe that the performance of all models over the training set is comparable, a fact which was also illustrated by Fig.1a. However, Figs.1b-1c illustrate that the HC-GNNs yield the best performance (on average), when regarding the validation set. This is due to the fast convergence rate during the first epoch demonstrated by Fig.1a. Yet, Fig.1d provides us with an *opposite* outcome: the non-convex GNNs perform better with respect to the validation set (on average). We note that this stems from the drawback embedded in convexified GNNs, which is proven by Theorem 1. Namely, since the expected generalization error is bounded by a term which depends on the number of robots n, we conclude that the higher the value of n, the higher the expected generalization error.

9 Conclusions and Future Work

In this paper we presented convexified GNN architectures, which address the NP-hardness of common non-convex ERM problems. The proposed convex relaxation is two-fold: the nuclear norm for the rank constraint of the parameter matrix, and the RKHS relaxation for handling non-linearity. Exploiting the resulting convexity, we employed a distributed online learning scheme, so as to learn distributed controls for controlling a robotic swarm in dynamic settings.

In many real-world applications, data can be naturally represented as graphs (e.g., recommendation systems [Gao *et al.*, 2020]), making our proposed models suitable in such contexts as well. Future work thus warrants exploring applicability of our models outside the realm of swarm robotics.

Acknowledgments

This research was funded in part by ISF grant 2306/18.

References

[Bartlett and Mendelson, 2002] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

[Blondel *et al.*, 2008] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[Blum and Rivest, 1992] Avrim L Blum and Ronald L Rivest. Training a 3-node neural network is np-complete. *Neural Networks*, 5(1):117–127, 1992.

[Cohen and Agmon, 2021] Saar Cohen and Noa Agmon. Code implementation. https://github.com/saarcohen30/convexified-gnn, 2021.

[Dabbagh and Bannan-Ritland, 2005] Nada Dabbagh and Brenda Bannan-Ritland. *Online learning: Concepts, strategies, and application*. Pearson/Merrill/Prentice Hall Upper Saddle River, NJ, 2005.

[Defferrard *et al.*, 2016] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[Gama *et al.*, 2018a] Fernando Gama, Antonio G Marques, Geert Leus, and Alejandro Ribeiro. Convolutional neural network architectures for signals supported on graphs. *IEEE Transactions on Signal Processing*, 67(4):1034–1049, 2018.

[Gama *et al.*, 2018b] Fernando Gama, Alejandro Ribeiro, and Joan Bruna. Diffusion scattering transforms on graphs. *arXiv preprint arXiv:1806.08829*, 2018.

[Gama *et al.*, 2020] Fernando Gama, Elvin Isufi, Geert Leus, and Alejandro Ribeiro. Graphs, convolutions, and neural networks. *arXiv preprint arXiv:2003.03777*, 2020.

[Gao *et al.*, 2020] Zhan Gao, Fernando Gama, and Alejandro Ribeiro. Wide and deep graph neural networks with distributed online learning. *arXiv preprint arXiv:2006.06376*, 2020.

[Hong *et al.*, 2015] Seunghoon Hong, Tackgeun You, Suha Kwak, and Bohyung Han. Online tracking by learning discriminative saliency map with convolutional neural network. In *International conference on machine learning*, pages 597–606, 2015.

[Li *et al.*, 2004] Yan Li, N Sundararajan, P Saratchandran, and Zhifeng Wang. Robust neuro-h/sub/spl infin//controller design for aircraft auto-landing. *IEEE Transactions on Aerospace and Electronic Systems*, 40(1):158–167, 2004.

[Li *et al.*, 2019] Qingbiao Li, Fernando Gama, Alejandro Ribeiro, and Amanda Prorok. Graph neural networks for decentralized multi-robot path planning. *arXiv preprint arXiv:1912.06095*, 2019.

[Nikolentzos *et al.*, 2018] Giannis Nikolentzos, Polykarpos Meladianos, Antoine Jean-Pierre Tixier, Konstantinos Skianis, and Michalis Vazirgiannis. Kernel graph convolutional neural networks. In *International Conference on Artificial Neural Networks*, pages 22–32. Springer, 2018.

[Ortega *et al.*, 2018] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

[Ross *et al.*, 2011] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[Sandryhaila and Moura, 2013] Aliaksei Sandryhaila and José MF Moura. Discrete signal processing on graphs. *IEEE transactions on signal processing*, 61(7):1644–1656, 2013.

[Sandryhaila and Moura, 2014] Aliaksei Sandryhaila and Jose MF Moura. Discrete signal processing on graphs: Frequency analysis. *IEEE Transactions on Signal Processing*, 62(12):3042–3054, 2014.

[Sartoretti *et al.*, 2019] Guillaume Sartoretti, Justin Kerr, Yunfei Shi, Glenn Wagner, TK Satish Kumar, Sven Koenig, and Howie Choset. Primal: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE Robotics and Automation Letters*, 4(3):2378–2385, 2019.

[Schölkopf *et al.*, 2001] Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *International conference on computational learning theory*, pages 416–426. Springer, 2001.

[Shuman *et al.*, 2013] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.

[Tolstaya *et al.*, 2020] Ekaterina Tolstaya, Fernando Gama, James Paulos, George Pappas, Vijay Kumar, and Alejandro Ribeiro. Learning decentralized controllers for robot swarms with graph neural networks. In *Conference on Robot Learning*, pages 671–682, 2020.

[Williams and Seeger, 2001] Christopher KI Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in neural information processing systems*, pages 682–688, 2001.

[Zhang *et al.*, 2016] Yuchen Zhang, Jason D Lee, and Michael I Jordan. l1-regularized neural networks are improperly learnable in polynomial time. In *International Conference on Machine Learning*, pages 993–1001, 2016.

[Zhang *et al.*, 2017] Yuchen Zhang, Percy Liang, and Martin J Wainwright. Convexified convolutional neural networks. In *International Conference on Machine Learning*, pages 4044–4053. PMLR, 2017.