SpreadGNN: Decentralized Multi-Task Federated Learning for Graph Neural Networks on Molecular Data

Chaoyang He*, Emir Ceyani*, Keshav Balasubramanian*
Murali Annavaram, Salman Avestimehr
University of Southern California, Los Angeles, CA 90089
{chaoyang.he,ceyani,keshavba,annavara,avestime}@usc.edu

Abstract

Graph Neural Networks (GNNs) are the first choice methods for graph machine learning problems thanks to their ability to learn state-of-the-art level representations from graph-structured data. However, centralizing a massive amount of real-world graph data for GNN training is prohibitive due to user-side privacy concerns, regulation restrictions, and commercial competition. Federated Learning is the de-facto standard for collaborative training of machine learning models over many distributed edge devices without the need for centralization. Nevertheless, training graph neural networks in a federated setting is vaguely defined and brings statistical and systems challenges. This work proposes SpreadGNN, a novel multi-task federated training framework capable of operating in the presence of partial labels and the absence of a central server for GNNs over molecular graphs. We provide convergence guarantees and empirically demonstrate the efficacy of our framework on a variety of non-I.I.D. distributed graph-level molecular property prediction datasets with partial labels. Our results show that SpreadGNN outperforms GNN models trained over a central server-dependent federated learning system, even in constrained topologies.

Introduction

Graph Neural Networks (GNNs) (Hamilton, Ying, and Leskovec 2017) are expressive models that can distill structural knowledge into highly representative embeddings. While graphs are the representation of choice in domains such as social networks (Bian et al. 2020), knowledge graphs for recommendation systems (Chen et al. 2020), in this work we focus on molecular graphs that are the core of drug discovery, molecular property prediction (Gilmer et al. 2017; Kearnes et al. 2016) and virtual screening (Zheng, Fan, and Mu 2019). Molecular graphs differ from their more well-known counterparts such as social network graphs. First, each molecule is a graph representation of the basic atoms and bonds that constitute the molecule and hence the size of the graph is small. Second, even though each graph may be small numerous molecules are being developed continuously for varied use cases. Hence, what they lack in size they make up for it in structural heterogeneity. Third, molecules can be labeled along multiple orthogonal dimensions. Since each graph has multiple labels the learning itself can be characterized as multi-task learning. For instance, whether a molecule has potentially harmful interactions with a diabetics drug, or whether that molecule can turn toxic under certain conditions are distinct labels. Molecular property analysis and labeling require wet-lab experiments, which are time-consuming and resource-costly. As a consequence, many entities may only have partially labeled molecules even if they know the graph structure. Finally, molecules are coveted inventions and hence entities often possess a proprietary graph representation that cannot be shared with other institutions for competitive and regulatory reasons. However, training collectively over a private set of molecular graphs can have immense societal benefits such as accelerated drug discovery.

Federated Learning (FL) is a distributed learning paradigm that addresses this data isolation problem via collaborative training. In this paradigm, training is an act of collaboration between multiple clients (such as research institutions) without requiring centralized local data while providing a certain degree of user-level privacy (McMahan et al. 2017; Kairouz et al. 2019). However, there are still challenges and shortcomings to training GNNs in a federated setting. This setting (Figure 1) is the typical case in molecular graphs since each owner may have different molecules and even when they have the same molecular graph each owner may have an incomplete set of labels for each molecule. The left half of Figure 1 shows a simpler case where all clients can communicate through a central server. However, in practice, the presence of a central server is not feasible when multiple competing entities may want to collaboratively learn. The challenges are further compounded by the lack of a central server as shown in the right half of the Figure 1. Thus, it remains an open problem to design a realistic federated learning framework for molecular GNNs, in which clients only have partial labels and one in which there is no reliance on a central server. This is the problem we seek to address in this work.

We propose a multi-task federated learning framework called SpreadGNN that operates in the presence of multiple, but partial labels for each client and the absence of a central server as shown in Figure 1. First, we present a multi-task learning (MTL) formulation to learn from partial labels. Second, in our MTL formulation, we utilize decentralized periodic averaging stochastic gradient descent to solve the

*Equal contribution (alphabetical order).

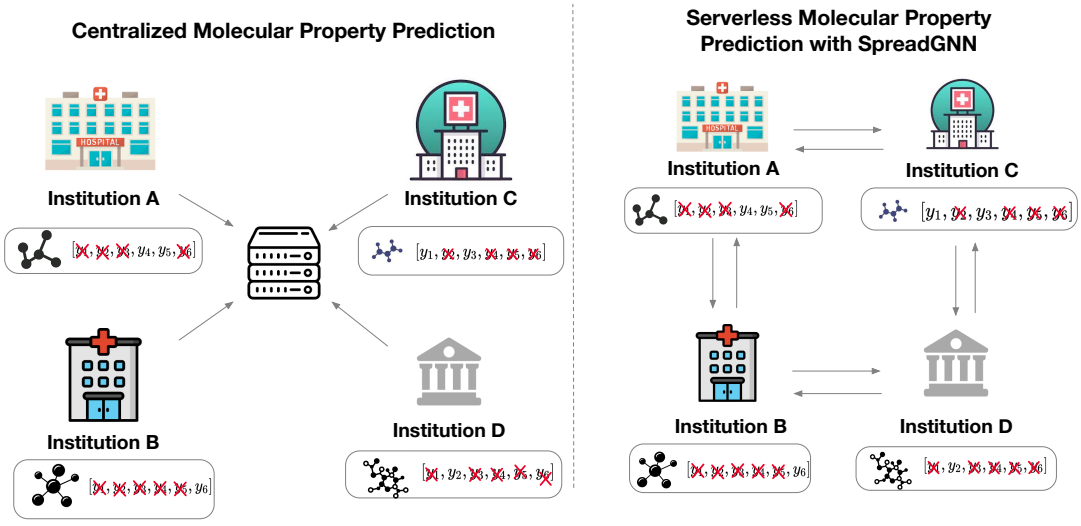

Figure 1: Serverless Multi-task Federated Learning for Graph Neural Networks.

serverless MTL optimization problem and provide a theoretical guarantee on the convergence properties, which further verifies the rationality of our design.

We evaluate SpreadGNN on graph-level molecular property prediction and regression tasks. We synthesize non-I.I.D. and partially labeled datasets by using curated data from the MoleculeNet (Wu et al. 2018) machine learning benchmark. With extensive experiments and analysis, we find that SpreadGNN can achieve even better performance than `FedAvg` (McMahan et al. 2016b), not only when all clients can communicate with each other, but also when clients are constrained to communicate with a subset of other clients. We plan on publishing the source code of SpreadGNN as well as related datasets for future exploration.

SpreadGNN Framework

Federated GNNs for Graph-Level Learning

We seek to learn *graph-level* representations in a federated learning setting over decentralized graph datasets which cannot be centralized for training due to privacy and regulation restrictions. For instance, compounds in molecular trials (Rong et al. 2020) may not be shared across entities because of intellectual property or regulatory concerns.

Under this setting, we assume that there are K clients in the FL network, and the k^{th} client has its own dataset $\mathcal{D}^{(k)} := \left\{\left(G_i^{(k)}, \mathbf{y}_i^{(k)}\right)\right\}_{i=1}^{N^{(k)}}$, where $G_i^{(k)} = (\mathcal{V}_i^{(k)}, \mathcal{E}_i^{(k)})$ is the i^{th} graph sample in $\mathcal{D}^{(k)}$ with node & edge feature sets $\boldsymbol{X}^{(k)} = \left\{\boldsymbol{x}_m^{(k)}\right\}_{m \in \mathcal{V}_i^{(k)}}$ and $\boldsymbol{Z}^{(k)} = \left\{\boldsymbol{e}_{m,n}^{(k)}\right\}_{m,n \in \mathcal{V}_i^{(k)}}$, $\mathbf{y_i}^{(k)}$ is the corresponding label of $G_i^{(k)}$, $N^{(k)}$ is the sample number in dataset $\mathcal{D}^{(k)}$, and $N = \sum_{k=1}^{K} N^{(k)}$.

Each client owns a GNN with a readout, to learn graph-level representations. We call this model of a GNN followed by a readout function, a *graph classifier*. Multiple clients are interested in collaborating to improve their GNN models without necessarily revealing their graph datasets. In this work, we build our theory upon the Message Passing Neural Network (MPNN) framework (Gilmer et al. 2017) as most spatial GNN models (Kipf and Welling 2016; Veličković et al. 2018; Hamilton, Ying, and Leskovec 2017) can be unified into this framework. The forward pass of an MPNN has two phases: a message-passing phase (Eq (1)) and an update phase (Eq (2)). For each client, we define the graph classifier with an L-layer GNN followed by a readout function as:

$$\boldsymbol{m}_i^{(k,\ell+1)} = \text{AGG}\left(\left\{\boldsymbol{M}_\theta^{(k,\ell+1)}\left(\boldsymbol{h}_i^{(k,\ell)}, \boldsymbol{h}_j^{(k,\ell)}, \boldsymbol{e}_{i,j}^{(k)}\right) | j \in \mathcal{N}_i\right\}\right), \ell = 0, \ldots, L-1 \quad (1)$$

$$\boldsymbol{h}_i^{(k,\ell+1)} = \boldsymbol{U}_\Psi^{(k,\ell+1)}\left(\boldsymbol{h}_i^{(k,\ell)}, \boldsymbol{m}_i^{(k,\ell+1)}\right), \ell = 0, \ldots, L-1 \quad (2)$$

$$\hat{\mathbf{y}}_i^{(k)} = \boldsymbol{R}_{\boldsymbol{\Phi}_{\text{pool}}, \boldsymbol{\Phi}_{\text{task}}}\left(\left\{\boldsymbol{h}_j^{(k,L)} | j \in \mathcal{V}_i^{(k)}\right\}\right) \quad (3)$$

where $\boldsymbol{h}_i^{(k,0)} = \boldsymbol{x}_i^{(k)}$ is the k^{th} client's node features, ℓ is the layer index, `AGG` is the aggregation function (e.g., in the GCN model (Kipf and Welling 2016), the aggregation function is a simple `SUM` operation), and $\mathcal{N}_i$ is the neighborhood set of node i. In Eq. (1), $\boldsymbol{M}_{\theta^{(k,\ell+1)}}(\cdot)$ is the message generation function which takes the hidden state of current node $\boldsymbol{h}_i$, the hidden state of the neighbor node $\boldsymbol{h}_j$ and the edge features $\boldsymbol{e}_{i,j}$ as inputs to gather and transform neighbors' messages. In other words, $\boldsymbol{M}_\theta$ combines a vertex's hidden state with the edge and vertex data from its neighbors to generate a new message. $\boldsymbol{U}_\Psi^{(k,\ell+1)}(\cdot)$ is the state update function that updates the model using the aggregated feature $\boldsymbol{m}_i^{(k,\ell+1)}$ as in Eq. (2). After propagating through L GNN layers, the final module of the graph classifier is a readout function $\boldsymbol{R}_{\boldsymbol{\Phi}_{\text{pool}}, \boldsymbol{\Phi}_{\text{task}}}(\cdot)$ which allows clients to predict a label for the graph, given node embeddings that are learned from Eq.(2). In general the readout is composed of two neural networks: the pooling function parameterized by $\boldsymbol{\Phi}_{\text{pool}}$; and a classifier

parameterized by $\mathbf{\Phi}_{\text{task}}$. The role of the pooling function is to learn a single graph embedding given node embedding from Eq (2). The classifier then uses the graph level embedding to predict a label for the graph.

To formulate GNN-based FL, using the model definition above, we define $\boldsymbol{W} = \{\theta, \mathbf{\Psi}, \mathbf{\Phi}_{\text{pool}}, \mathbf{\Phi}_{\text{task}}\}$ as the overall learnable weights. Note that $\boldsymbol{W}$ is independent of graph structure as both the GNN and Readout parameters make no assumptions about the input graph. Thus, one can learn $\boldsymbol{W}$ using a FL based approach. The the overall FL task can be formulated as a distributed optimization problem as:

$$\min_{\boldsymbol{W}} F(\boldsymbol{W}) \stackrel{\text{def}}{=} \min_{\boldsymbol{W}} \sum_{k=1}^{K} \frac{N^{(k)}}{N} \cdot f^{(k)}(\boldsymbol{W}) \tag{4}$$

where $f^{(k)}(\boldsymbol{W}) = \frac{1}{N^{(k)}} \sum_{i=1}^{N^{(k)}} \mathcal{L}(\hat{\mathbf{y}}_i^{(k)}, \mathbf{y}_i^{(k)})$ is the k^{th} client's local objective function that measures the local empirical risk over the heterogeneous graph dataset $\mathcal{D}^{(k)}$. $\mathcal{L}$ is the loss function of the global graph classifier.

With such a formulation, it might seem like an optimization problem tailored for a `FedAvg` based optimizers (McMahan et al. 2016b). Unfortunately, in molecular graph settings this is not the case for the following reasons. (a) In our setting, clients belong to a decentralized, serverless topology. There is no central server that can average model parameters from the clients. (b) Clients in our setting possess incomplete labels, hence the dimensions of $\mathbf{\Phi_{task}}$ can be different on different clients in decentralized topologies. For instance, a client may only have partial toxicity labels for a molecule, while another client may have a molecule's interaction properties with another drug compound. Even with such incomplete information from each client, our learning task is interested in classifying each molecule across multiple label categories.

To combat these issues, we aim to propose a federated learning framework that can achieve model personalization and decentralized topology simultaneously. Particularly, we propose a novel decentralized multi-task learning framework to tackle the aforementioned issues. Next, we will first introduce a centralized FMTL framework for graph neural networks as preliminary. We then enhance this centralized FMTL to a serverless and decentralized scenario, which is named as SpreadGNN. We introduce how we address the multi-label inconsistent issue and the challenge of correlating model weights from different client numbers.

Federated Multi-Task Learning with GNNs

Under the regularized MTL paradigm (Evgeniou and Pontil 2004), we define the centralized federated graph MTL problem (FedGMTL) as follows:

$$\begin{aligned} \min_{\theta, \mathbf{\Psi}, \mathbf{\Phi}_{\text{pool}}, \mathbf{\Phi}_{\text{task}}, \mathbf{\Omega}} & \sum_{k=1}^{K} \frac{1}{N_k} \sum_{i=1}^{N_k} \mathcal{L}(\hat{\mathbf{y}}_i^{(k)}, \mathbf{y}_i^{(k)}) + \mathcal{R}(\boldsymbol{W}, \mathbf{\Omega}), \\ \text{s.t.} \quad & \mathbf{\Omega} \geq 0 \quad \text{and} \quad \text{Tr}(\mathbf{\Omega}) = 1. \end{aligned} \tag{5}$$

where

$$\mathcal{R}(\boldsymbol{W}, \mathbf{\Omega}) = \frac{1}{2}\lambda_1 \, \text{Tr}(\mathbf{\Phi}_{\text{task}} \mathbf{\Omega}^{-1} \mathbf{\Phi}_{\text{task}}^T) + \frac{1}{2} \sum_{\boldsymbol{\chi} \in \boldsymbol{W}} \lambda_{\boldsymbol{\chi}} ||\boldsymbol{\chi}||_F^2 \tag{6}$$

is the bi-convex regularizer introduced in (Zhang and Yeung 2012). The first term of the Eq. (5) models the summation of different empirical loss of each client, which is what Eq. (4) exactly tries to address. The second term serves as a task-relationship regularizer with $\mathbf{\Omega} \in \mathbb{R}^{\mathcal{S} \times \mathcal{S}}$ being the covariance matrix for $\mathcal{S}$ different tasks constraining the task weights $\mathbf{\Phi}_{\text{task}} = [\mathbf{\Phi}_{\text{task},1}, \ldots, \mathbf{\Phi}_{\text{task},\mathcal{S}}] \in \mathbb{R}^{d \times \mathcal{S}}$ through matrix trace $\text{Tr}(\mathbf{\Phi}_{\text{task}} \mathbf{\Omega}^{-1} \mathbf{\Phi}_{\text{task}}^T)$. Recall that each client in our setting may only have a partial set of tasks in the labels of its training dataset, but still needs to make predictions for tasks it does not have in its labels during test time. This regularizer helps clients relate its own tasks to tasks in other clients. Intuitively, it determines how closely two tasks i and j are related. The closer $\mathbf{\Phi}_{\text{task},i}$ and $\mathbf{\Phi}_{\text{task},j}$ are, the larger $\mathbf{\Omega}_{i,j}$ will be. If $\mathbf{\Omega}$ is an identity matrix, then each node is independent to each other. But as our results show, there is often a strong correlation between different molecular properties. This compels us to use a federated learning model.

Figure 2-a depicts the FedGMTL framework where clients' graph classifier weights are using an FL server. While the above formulation enhances the FMTL with a constrained regularizer that can be used for GNN learning, we still need to solve the final challenge, which is to remove the reliance on a central server to perform the computations in Eq. (5). Therefore, we propose a Decentralized Graph Multi-Task Learning framework, SpreadGNN, to extend the FedGMTL framework to the decentralized case, which is shown in the Figure 2-c. Note that each client's learning task remains the same but the aggregation process differs in SpreadGNN.

SpreadGNN: Serverless Federated MTL for GNNs

In a serverless setting, in which all clients are not necessarily connected to all other clients through a central server also makes it impossible to maintain one single task covariance matrix $\mathbf{\Omega}$. Thus, the naive formulation in Equation 5 becomes obsolete in a serverless case. To combat against this issue, we propose using distinct covariance matrices $\mathbf{\Omega}_k$ for each client that are efficiently updated using the exchange mechanism. We formalize this idea as follows: Consider one particular client m having task weights $\mathbf{\Phi}_{\text{task},m} \in \mathcal{R}^{d \times \mathcal{S}_m}$ where $\mathcal{S}_m$ is the number of tasks that client m has. Then, each client localizes the optimization procedure in Equation 5 with respect to his/her neighbors.

In the decentralized setting, we emphasize that clients can collectively learn an exhaustive set of tasks, even when clients may not have access to some of the classes in each label. That is, $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset \quad \forall i \neq j$ and $\cup_i \mathcal{S}_i = \mathcal{S}$. Let $\mathfrak{L} = \sum_{k=1}^{K} \frac{1}{N_k} \sum_{i=1}^{N_k} \mathcal{L}(\hat{y}_i^{(k)}(\boldsymbol{X}_i^k, \boldsymbol{Z}_i^k \, \boldsymbol{W}_k), \mathbf{y}_i^k)$. Then, the new non-convex objective function is defined as:

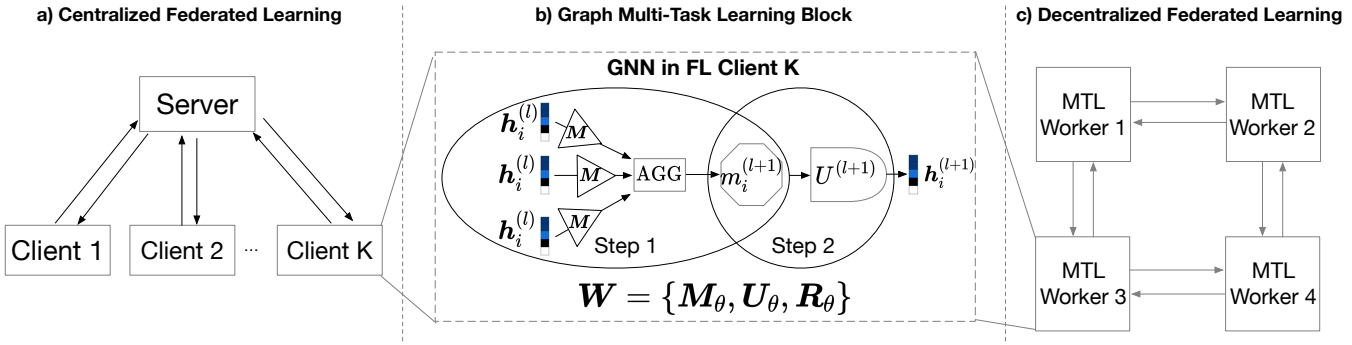

Figure 2: Federated Graph Multi-Task Learning Framework (FedGMTL).

$$\min_{\theta,\boldsymbol{\Psi},\boldsymbol{\Phi}_{\text{pool}},\boldsymbol{\Phi}_{\text{task}},\underline{\boldsymbol{\Omega}}} \mathfrak{L} + \sum_{k=1}^{K} \frac{1}{2}\lambda_1 \operatorname{Tr}(\boldsymbol{\Phi}_{\text{task}_{\mathcal{M}_k}} \boldsymbol{\Omega}_k^{-1} \boldsymbol{\Phi}_{\text{task}_{\mathcal{M}_k}}^T) + \frac{1}{2}\sum_{\boldsymbol{\chi}\in\boldsymbol{W}_k} \lambda_{\boldsymbol{\chi}} \|\boldsymbol{\chi}\|_F^2,$$
$$\text{s.t.} \quad \boldsymbol{\Omega}_k \geq 0 \qquad \text{and} \quad \operatorname{Tr}(\boldsymbol{\Omega}_k) = 1, \quad k = 1, 2, ..., K. \tag{7}$$

where $\boldsymbol{W}_k = \left\{\theta, \boldsymbol{\Psi}, \boldsymbol{\Phi}_{pool}, \boldsymbol{\Phi}_{\text{task}_{\mathcal{M}_k}}\right\}$ is the set of all learnable weights for client k, $\mathcal{M}_k = k \cup \mathcal{N}_k$ is the neighbor set for client k including itself. This gives rise to: $\boldsymbol{\Phi}_{\text{task}_{\mathcal{M}_k}} = [\boldsymbol{\Phi}_{\text{task},1} \| \boldsymbol{\Phi}_{\text{task},2} \| \dots \| \boldsymbol{\Phi}_{\text{task},|\mathcal{M}_k|}] \in \mathbb{R}^{d\times|\mathcal{S}_{\mathcal{M}_k}|}$ which is the task weight matrix for client k and its neighbors and $\|$ represents the row-wise concatenation operation. The matrix $\boldsymbol{\Omega}_k \in \mathbb{R}^{|\mathcal{S}_{\mathcal{M}_k}|\times|\mathcal{S}_{\mathcal{M}_k}|}$ represents the correlation amongst all the available tasks for the set $\mathcal{M}_k$.

To solve this non-convex problem, we apply the alternating optimization method presented in (Zhang and Yeung 2012), where $\boldsymbol{W}_k$ and $\boldsymbol{\Omega}_k$ are updated in an alternative fashion.

Optimizing $\boldsymbol{W}_k$: We define $\underline{\boldsymbol{\Omega}} = \{\boldsymbol{\Omega}_i\}_{i=1}^K$ to represent the set of correlation matrices for all clients. Fixing $\underline{\boldsymbol{\Omega}}$, we can use SGD to update $\boldsymbol{W}_k$ jointly. Then, our problem can then be reformulated as:

$$G(\boldsymbol{W}_k|\underline{\boldsymbol{\Omega}}) = \mathfrak{L} + \frac{1}{2}\lambda_1 \sum_{k=1}^{K} \operatorname{Tr}(\boldsymbol{\Phi}_{\text{task}_{\mathcal{M}_k}} \boldsymbol{\Omega}_k^{-1} \boldsymbol{\Phi}_{\text{task}_{\mathcal{M}_k}}) + \frac{1}{2}\sum_{\boldsymbol{\chi}\in\boldsymbol{W}} \lambda_{\boldsymbol{\chi}} \|\boldsymbol{\chi}\|_F^2. \tag{8}$$

where the summation in (8) is amongst all nodes connected to node k. Then the gradient formulations for each node are:

$$\nabla_{\boldsymbol{\Phi}_{\text{task}_{\mathcal{M}_k}}} G(\boldsymbol{W}_k|\underline{\boldsymbol{\Omega}}) = \nabla_{\boldsymbol{\Phi}_{\text{task}_{\mathcal{M}_k}}} \mathfrak{L} + \lambda_1 \sum_{i=1}^{|\mathcal{M}_k|} \frac{1}{N_i} \boldsymbol{\Phi}_{\text{task}_{\mathcal{M}_k}} \boldsymbol{\Omega}_i^{-1} + \lambda_{\boldsymbol{\chi}} \boldsymbol{\Phi}_{\text{task}_{\mathcal{M}_k}} \tag{9}$$

$$\nabla_{\boldsymbol{\chi}} G(\boldsymbol{W}_k|\underline{\boldsymbol{\Omega}}) = \nabla_{\boldsymbol{\chi}} \mathfrak{L} + \lambda_{\boldsymbol{\chi}} \boldsymbol{\chi}, \forall \boldsymbol{\chi} \in \boldsymbol{W}_k \backslash \{\boldsymbol{\Phi}_{\text{task}_{\mathcal{M}_k}}\} \tag{10}$$

Optimizing $\boldsymbol{\Omega_k} \in \underline{\boldsymbol{\Omega}}$: In (Zhang and Yeung 2012), an analytical solution for $\boldsymbol{\Omega}$ is equal to $\hat{\boldsymbol{\Omega}} = (\boldsymbol{\Phi}_{\text{task}}^T \boldsymbol{\Phi}_{\text{task}})^{\frac{1}{2}} / \operatorname{Tr}((\boldsymbol{\Phi}_{\text{task}}^T \boldsymbol{\Phi}_{\text{task}})^{\frac{1}{2}})$. However, this solution is only applicable for the centralized case. This is because missing central node forbids averaging parameters globally. So here we propose a novel way to update each $\boldsymbol{\Omega_k} \in \underline{\boldsymbol{\Omega}}$:

$$f_{align}(\boldsymbol{\Omega}_k^{(t+1)}) \leftarrow \eta \frac{1}{|\mathcal{M}_k|} \Big(\sum_{i=1}^{|\mathcal{M}_k|} \frac{1}{N_i} f_{align}(\boldsymbol{\Omega}_i^{(t)}) + f_{align}(\hat{\boldsymbol{\Omega}}_k)\Big) \tag{11}$$

where $\hat{\boldsymbol{\Omega}}_k$ is the analytical solution for $\boldsymbol{\Phi}_{\text{task}_{\mathcal{M}_k}}$ at node k. The first averaging term can incorporate the nearby nodes correlation into its own. It should be noticed that each $\boldsymbol{\Omega}_i$ may have a different dimension (different number of neighbors), so this averaging algorithm is based on the node-wised alignment in the global $\boldsymbol{\Omega}$. We refer readers to Appendix[1] for a full sketch of our algorithm.

Convergence Properties In this section, we present our convergence analysis for SpreadGNN optimization problem. Before any formalism, we first introduce a connection matrix $\mathbf{M} \in \mathbb{R}^{K\times K}$ to record the connections in the network. We assume that $\mathbf{M}$ satisfies the following:

1. $\mathbf{M}\mathbf{1}_K = \mathbf{1}_K$
2. $\mathbf{M}^T = \mathbf{M}$
3. $\max\{|\lambda_2(\mathbf{M})|, \dots, |\lambda_K(\mathbf{M})|\} < \lambda_1(\mathbf{M}) = 1$

where λ denotes an eigenvalue of $\mathbf{M}$. In our analysis, we assume that the following properties hold (Bottou, Curtis, and Nocedal 2018):

- The objective function $F(\cdot)$ is L-Lipschitz.
- $F(\cdot)$ is lower bounded by F_{inf} such that $F(\cdot) \geq F_{inf}$.
- The full gradient of the objective function $F(\cdot)$ is approximated by stochastic gradient g on a mini-batch ϵ_i with an unbiased estimate: $\mathbb{E}_{\epsilon_k}[g(\mathbf{x})] = \nabla F(\mathbf{x})$.
- The variance of stochastic gradient $g(\mathbf{x})$ given a mini-batch ϵ_k is upper bounded and $\text{Var}_{\epsilon_k}(g(\mathbf{x})) \leq \beta\|\nabla F(\mathbf{x})\|^2 + \sigma^2, \quad \exists\beta, \sigma^2 \geq 0, \quad \forall k.$

For fixed step size, our updates can be written as follows:

$$\mathbf{X}_{t+1} = (\mathbf{X}_t - \eta \mathbf{G}_t) \cdot \mathbf{M}_t, \tag{12}$$

where $\mathbf{X}_t = \left[\mathbf{x}_t^{(1)}, \dots, \mathbf{x}_t^{(K)}\right]$ is the matrix of our interest (e.g. each element of $\boldsymbol{W}_k^{(t)}$), $\mathbf{G}_t$ is the gradient and $\mathbf{M}_t$ is the

[1] Appendix is available at https://arxiv.org/abs/2106.02743

connection matrix at time t. When $t \mod \tau = 0, \mathbf{M}_t = \mathbf{M}$, otherwise, $\mathbf{M}_t = \mathbf{I}_K$. In equation (12), multiplying $\mathbf{I}_K/K$ on both sides, defining the averaged model $\mathbf{u}_t = \mathbf{X}_t \frac{\mathbf{I}_K}{K}$, we obtain the following update:

$$\begin{aligned} \mathbf{u}_{t+1} &= \mathbf{u}_t - \eta \mathbf{g}_t \\ &= \mathbf{u}_t - \eta \left[\frac{1}{K} \sum_{i=1}^{K} g(\mathbf{x}_t^{(i)}) \right] \end{aligned} \tag{13}$$

Next, we will present our analysis on the convergence of the above-averaged model $\mathbf{u}_t$. For non-convex optimization, previous works on SGD convergence analysis use the *average squared gradient norm* as an indicator of convergence to a stationary point [Bottou, Curtis, and Nocedal 2018].

Theorem 1 (Convergence of SpreadGNN) *If the learning rate η satisfies the following condition:*

$$\eta L + \frac{\eta^2 L^2 \tau^2}{1-\zeta} \left(\frac{2\zeta^2}{1+\zeta} + \frac{2\zeta}{1-\zeta} + \frac{\tau - 1}{\tau} \right) \leq 1 \tag{14}$$

where τ is the averaging period(one synchronization per τ local updates), and $\zeta = \max\{|\lambda_2(\mathbf{M})|, \ldots, |\lambda_m(\mathbf{M})|\}$, and all local models are initialized at a same point $\mathbf{x}_0$, then after T iterations the average squared gradient norm is bounded as

$$\begin{aligned} \mathbb{E}\left[\frac{1}{T} \sum_{t=1}^{T} \|\nabla F(\mathbf{u}_t)\|^2 \right] \leq \frac{2[F(\mathbf{x}_1) - F_{inf}]}{\eta T} + \frac{\eta L \sigma^2}{K} \\ + \eta^2 L^2 \sigma^2 \left(\frac{1+\zeta^2}{1-\zeta^2} \tau - 1 \right) \end{aligned} \tag{15}$$

Theorem 1 shows that SpreadGNN algorithm converges to the stationary point solution under certain conditions. (Wang and Joshi 2018) presents similar results in an unified framework, but it does not provide adequate theoretical analysis and empirical evaluation for federated learning. The proof of Theorem 1 is given in the Appendix[1].

Experiments

Setup

Implementation. All experiments are conducted on a single GPU server equipped with 2 Nvidia Geforce GTX 1080Ti GPUs and an AMD Ryzen 7 3700X 8-Core Processor. Our models are built on top of the FedML framework (He et al. 2020) and PyTorch (Paszke et al. 2019).

Multi-Label Dataset. We use molecular datasets from the MoleculeNet (Wu et al. 2018) machine learning benchmark in our evaluation. In particular, we evaluate our approach on molecular property prediction datasets described in Table 1. The label for each molecule graph is a vector in which each element denotes a property of the molecule. Properties are binary in the case of classification or continuous values in the case of regression. As such, each multi-label dataset can adequately evaluate our learning framework.

Non-I.I.D. Partition for Quantity and Label Skew. We introduce non-I.I.D.ness in two additive ways. The first is a non-I.I.D. split of the training data based on quantity. Here we use a Dirichlet distribution parameterized by α to split the training data between clients. Specifically, the number of training samples present in each client is non-I.I.D. The second source of non-I.I.D.ness is a label masking scheme designed to represent the scenario in which different clients may possess partial labels as shown in Figure 1. More specifically, we randomly mask out a subset of classes in each label on every client. In our experiments, the sets of unmasked classes across all clients are mutually exclusive and collectively exhaustive. This setting simulates a worst case scenario where no two clients share the same task. However our framework is just as applicable when there is label overlap. Such masking, introduces a class imbalance between the clients making the label distribution non-I.I.D. as well.

Baseline Algorithm. For a fair and reasonable comparison with our baseline, we utilize the same masking to simulate missing labels for each client. Afterwards, we train models trained with FedAvg with the same loss as we use for decentralized case.

Models. In order to demonstrate that our framework is agnostic to the choice of GNN model, we run experiments on GraphSAGE (Hamilton, Ying, and Leskovec 2017) and GAT (Veličković et al. 2018).

Network Topology. We first evaluate our framework in a complete topology in which all clients are connected to all other clients to measure the efficacy of our proposed regularizer. We then perform ablation studies on the number of neighbors of each client to stress our framework in the more constrained setting.

Hyperparameters. We use Adam (Kingma and Ba 2015) as the client optimizer in all of our experiments. For our Tox21, MUV, and QM8, we use an 8 client topology. For SIDER we use a 4 client topology. A more comprehensive hyperparameter list for network topology and models can be found in the Appendix[1].

Results

We use a central, server dependent `FedAvg` system as the baseline of comparison. More specifically, all clients are involved in the averaging of model parameters in every averaging round.

Our results summarized in Table 2 demonstrate that SpreadGNN (third column) outperforms a centralized federated learning system that uses FedAvg (first column) when all clients can communicate with all other clients. This shows that by using the combination of the task regularizer in equation 6 and decentralized optimization, we can eliminate the dependence on a central server and enable clients to learn more effectively in the presence of missing molecular properties in their respective labels. Additionally, the results also show that our framework is agnostic to the type of GNN model being used in the molecular property prediction task since both GraphSAGE and GAT benefit from our framework. Our framework also works in the case a trusted central server is available (second column). The presence of a trusted

[1]Appendix is available at https://arxiv.org/abs/2106.02743

[1]Appendix is available at https://arxiv.org/abs/2106.02743

Dataset	# Molecules	Avg # Nodes	Avg # Edges	# Tasks	Task Type	Evaluation Metric
SIDER	1427	33.64	35.36	27	Classification	ROC-AUC
Tox21	7831	18.51	25.94	12	Classification	ROC-AUC
MUV	93087	24.23	76.80	17	Classification	ROC-AUC
QM8	21786	7.77	23.95	12	Regression	MAE

Table 1: Dataset summary used in our experiments.

	GraphSAGE			GAT		
	FedAvg	**FedGMTL**	**SpreadGNN**	**FedAvg**	**FedGMTL**	**SpreadGNN**
SIDER	0.582	**0.629**	0.5873	0.5857	**0.61**	0.6034
Tox21	0.5548	**0.6664**	0.585	0.6035	**0.6594**	0.6056
MUV	0.6578	0.6856	**0.703**	0.709	0.6899	**0.713**
QM8	0.02982	0.03624	**0.02824**	0.0392	0.0488	**0.0315**

Table 2: Molecular property prediction results with complete topology: and communication period $\tau = 1$

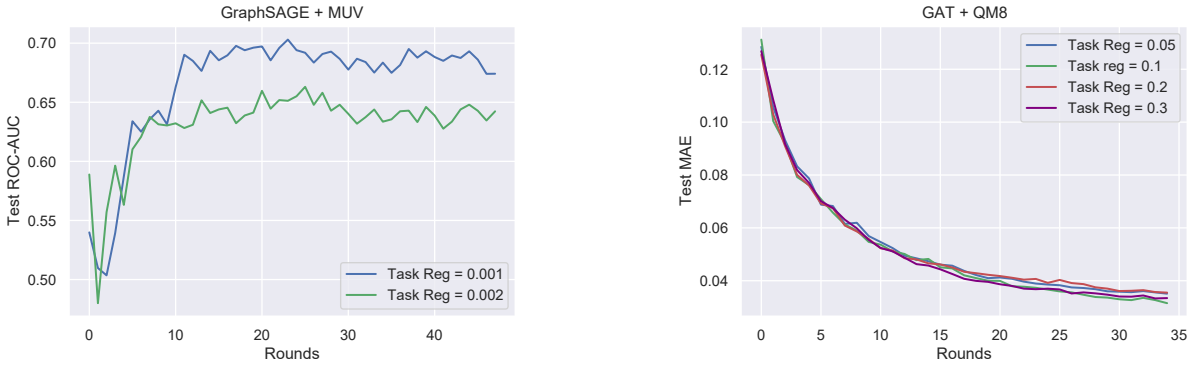

Figure 3: Effect of Task-Relationship Regularizer on Learning

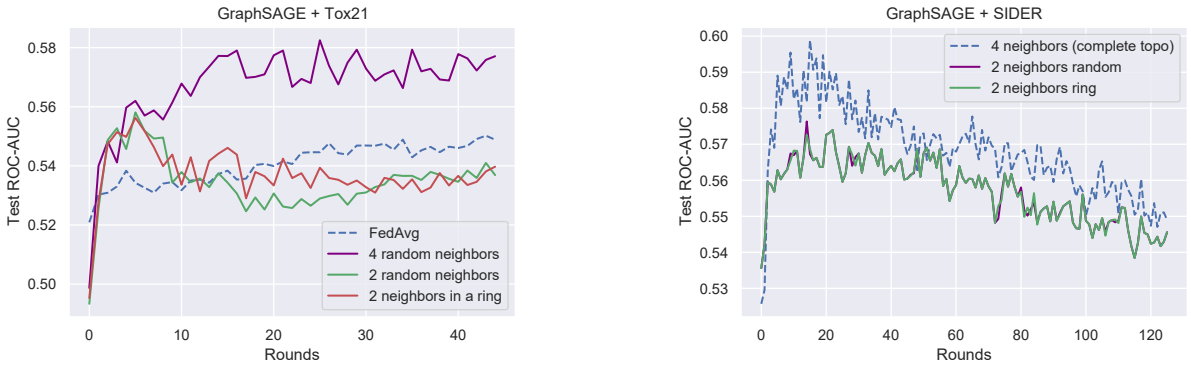

Figure 4: Effect of Topology on Learning for GraphSAGE Model.

central server improves the accuracy in a few scenarios. However, SpreadGNN provides competing performance in a more realistic setting where the central server is not feasible.

Sensitivity Analysis

Task Regularizer. Figure 3 illustrates the effect of λ_1 on both regression as well as classification tasks. Interestingly, regression is much more robust to variation in λ_1 while classification demands more careful tuning of λ_1 to achieve optimal model performance. This implies that the different properties in the regression task are more independent than the properties in the classification task.

Network Topology. The network topology dictates how many neighbors each client can communicate with in a com-

munication round. While Table 2 shows that SpreadGNN outperforms FedAvg in a complete topology, Figure 4 shows that our framework performs outperforms FedAvg even when clients are constrained to communicate with fewer neighbors. We can also see that it's not just $n_{neighbors}$ that matters, the topology in which clients are connected does too. When $n_{neighbors} = 2$, a ring topology outperforms a random topology, as a ring guarantees a path from any client to any other client. Thus, learning is shared indirectly between all clients. The same is not true in a random topology. Also, in Figure 4 illustrates the effect of varying topologies on SpreadGNN on the Sider dataset when using Graphsage as the GNN. The qualitative behavior is similar to Figure 4, in that when each client is connected to more neighbors, the local model of each client is more robust. However, when the total number of clients involved in the network is smaller, the effect of topology is understated and the total number of neighbors matters more. Recall that in an 8 client network, when each client was restricted to being connected to only 2 neighbors, random connections performed worse than a ring topology, meaning that the topology mattered as much as the mere number of neighbors. However, in the case of the 4 client network, there is a minimal difference between a 2 neighbor random configuration and a 2 neighbor ring configuration.

Period. The communication period τ is another important hyperparameter in our framework. As we increase the communication period τ more, model performance decreases. However, selecting $\tau = 5$ can sometimes be better than averaging & exchanging each round. This indicates that tuning τ is important for while controlling the tradeoff between the performance and the running time. In general, our experiments suggest that a lower period is better, but this is not always the case. We include an ablation study on τ to support this claim in the Appendix[1].

Related Works

Molecular Representation Learning. (Rogers and Hahn 2010) encode the neighbors of atoms in the molecule into a fix-length vector to obtain vector space representations. To improve the expressive power of chemical fingerprints, (Duvenaud et al. 2015; Coley et al. 2017) use CNNs to learn rich molecule embeddings for downstream tasks like property prediction. (Kearnes et al. 2016; Schütt et al. 2017) explore the graph convolutional network to encode molecular graphs into neural fingerprints. To better capture the interactions among atoms, (Gilmer et al. 2017) proposes to use a message passing framework.

FL. Early examples of research into federated learning include (Konečný, McMahan, and Ramage 2015; McMahan et al. 2016a). To address both statistical and system challenges in FL, (Smith et al. 2017) proposes a multi-task learning framework for federated learning and its related optimization algorithms, which extends early works in distributed machine learning (Yang et al. 2013; Jaggi et al. 2014). The main limitation, however, is that strong duality is only guaranteed when the objective function is convex, which can not be generalized to the non-convex settings.(Jin et al. 2015; Mateos-Núñez, Cortés, and Cortes 2015; Wang, Kolar, and Srerbo 2016; Baytas et al. 2016; Liu, Pan, and Ho 2017) extends federated multi-task learning to the distributed multi-task learning setting, but not only this limitation remains same, but also nodes performing the same amount of work is prohibitive in FL.

Federated Graph Neural Networks. (Suzumura et al. 2019) and (Mei et al. 2019) use computed graph statistics for information exchange and aggregation to avoid node information leakage. (Jiang et al. 2020) utilizes cryptographic approaches to incorporate into GNN learning. (Wang et al. 2020) propose a hybrid of federated and meta learning to solve the semi-supervised graph node classification problem in decentralized social network datasets. (**?**) uses an edge-cloud partitioned GNN model for spatio-temporal traffic forecasting tasks over sensor networks. The previous works do not consider graph learning in a decentralized setting.

Stochastic Gradient Descent Optimization. In large-scale distributed machine learning problems, learning synchronized mini-batch SGD is a well-known method to address the communication bottleneck by increasing the computation-to communication ratio (Li et al. 2014). It is shown that FedAvg (Konečný et al. 2016) is a special case of local SGD which allow nodes to perform local updates and infrequent synchronization between them to communicate less while converging fast (Wang and Joshi 2018; Yu, Yang, and Zhu 2018; Lin, Stich, and Jaggi 2018). Decentralized SGD, another approach to reducing communication, was successfully applied to deep learning (Jiang et al. 2017; Lian et al. 2017). Asynchronous SGD is a potential method that can alleviate synchronization delays in distributed learning (Mitliagkas et al. 2016), but existing asynchronous SGD does not fit for federated learning because the staleness problem is particularly severe due to the reason of heterogeneity in the federated setting (Dai et al. 2018).

Conclusion

In this work, we propose SpreadGNN, to train federated graph neural networks in a decentralized manner. We motivate our framework through a realistic setting, in which clients involved in molecular machine learning research cannot share data with each other due to privacy regulations and competition. Moreover, we are aware that clients possess multiple, but partial labels. For the first time, experiments show that training federated graph neural networks does not require a centralized topology and that our framework can address the non-I.I.D.ness in dataset size and label distribution across clients. SpreadGNN can outperform a central server dependent baseline even when clients can only communicate with a few neighbors. To support our empirical results, we also provide a convergence analysis for our framework.

Acknowledgements

Authors acknowledge that this material is based upon work supported by Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001117C0053 and FA8750-19-2-1005, ARO award W911NF1810400, NSF grants CCF-1703575, CCF-1763673, and MLWINS-

[1]Appendix is available at https://arxiv.org/abs/2106.02743

2002874, ONR Award No. N00014-16-1-2189, and a gift from Intel/Avast/Borsetta via the PrivateAI institute, a gift from Cisco, and a gift from Qualcomm. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

References

Baytas, I. M.; Yan, M.; Jain, A. K.; and Zhou, J. 2016. Asynchronous multi-task learning. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, 11–20. IEEE.

Bian, T.; Xiao, X.; Xu, T.; Zhao, P.; Huang, W.; Rong, Y.; and Huang, J. 2020. Rumor detection on social media with bi-directional graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 549–556.

Bottou, L.; Curtis, F. E.; and Nocedal, J. 2018. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2): 223–311.

Chen, C.; Cui, J.; Liu, G.; Wu, J.; and Wang, L. 2020. Survey and Open Problems in Privacy Preserving Knowledge Graph: Merging, Query, Representation, Completion and Applications. *arXiv preprint arXiv:2011.10180*.

Coley, C. W.; Barzilay, R.; Green, W. H.; Jaakkola, T. S.; and Jensen, K. F. 2017. Convolutional embedding of attributed molecular graphs for physical property prediction. *Journal of chemical information and modeling*, 57(8): 1757–1772.

Dai, W.; Zhou, Y.; Dong, N.; Zhang, H.; and Xing, E. P. 2018. Toward Understanding the Impact of Staleness in Distributed Machine Learning. *arXiv:1810.03264 [cs, stat]*. ArXiv: 1810.03264.

Duvenaud, D.; Maclaurin, D.; Aguilera-Iparraguirre, J.; Gómez-Bombarelli, R.; Hirzel, T.; Aspuru-Guzik, A.; and Adams, R. P. 2015. Convolutional networks on graphs for learning molecular fingerprints. *arXiv preprint arXiv:1509.09292*.

Evgeniou, T.; and Pontil, M. 2004. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 109–117.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, 1263–1272. PMLR.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. *CoRR*, abs/1706.02216.

He, C.; Li, S.; So, J.; Zhang, M.; Wang, H.; Wang, X.; Vepakomma, P.; Singh, A.; Qiu, H.; Shen, L.; Zhao, P.; Kang, Y.; Liu, Y.; Raskar, R.; Yang, Q.; Annavaram, M.; and Avestimehr, S. 2020. FedML: A Research Library and Benchmark for Federated Machine Learning. *arXiv preprint arXiv:2007.13518*.

Jaggi, M.; Smith, V.; Takác, M.; Terhorst, J.; Krishnan, S.; Hofmann, T.; and Jordan, M. I. 2014. Communication-efficient distributed dual coordinate ascent. In *Advances in neural information processing systems*, 3068–3076.

Jiang, M.; Jung, T.; Karl, R.; and Zhao, T. 2020. Federated Dynamic GNN with Secure Aggregation. *arXiv preprint arXiv:2009.07351*.

Jiang, Z.; Balu, A.; Hegde, C.; and Sarkar, S. 2017. Collaborative deep learning in fixed topology networks. In *Advances in Neural Information Processing Systems*, 5904–5914.

Jin, X.; Luo, P.; Zhuang, F.; He, J.; and He, Q. 2015. Collaborating between local and global learning for distributed online multiple tasks. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 113–122. ACM.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. 2019. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*.

Kearnes, S.; McCloskey, K.; Berndl, M.; Pande, V.; and Riley, P. 2016. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8): 595–608.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Kipf, T. N.; and Welling, M. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *CoRR*, abs/1609.02907.

Konečný, J.; McMahan, B.; and Ramage, D. 2015. Federated Optimization:Distributed Optimization Beyond the Datacenter. *arXiv:1511.03575 [cs, math]*. ArXiv: 1511.03575.

Konečný, J.; McMahan, H. B.; Yu, F. X.; Richtárik, P.; Suresh, A. T.; and Bacon, D. 2016. Federated Learning: Strategies for Improving Communication Efficiency. *arXiv:1610.05492 [cs]*. ArXiv: 1610.05492.

Li, M.; Andersen, D. G.; Park, J. W.; Smola, A. J.; Ahmed, A.; Josifovski, V.; Long, J.; Shekita, E. J.; and Su, B.-Y. 2014. Scaling Distributed Machine Learning with the Parameter Server. In *OSDI*, volume 14, 583–598.

Lian, X.; Zhang, C.; Zhang, H.; Hsieh, C.-J.; Zhang, W.; and Liu, J. 2017. Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent. *arXiv:1705.09056 [cs, math, stat]*. ArXiv: 1705.09056.

Lin, T.; Stich, S. U.; and Jaggi, M. 2018. Don't Use Large Mini-Batches, Use Local SGD. *arXiv:1808.07217 [cs, stat]*. ArXiv: 1808.07217.

Liu, S.; Pan, S. J.; and Ho, Q. 2017. Distributed multi-task relationship learning. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 937–946. ACM.

Mateos-Núñez, D.; Cortés, J.; and Cortes, J. 2015. Distributed optimization for multi-task learning via nuclear-norm approximation. In *IFAC Workshop on Distributed Estimation and Control in Networked Systems*, volume 48, 64–69.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, 1273–1282. PMLR.

McMahan, H. B.; Moore, E.; Ramage, D.; Hampson, S.; and Arcas, B. A. y. 2016a. Communication-Efficient Learning of Deep Networks from Decentralized Data. *arXiv:1602.05629 [cs]*. ArXiv: 1602.05629.

McMahan, H. B.; Moore, E.; Ramage, D.; and y Arcas, B. A. 2016b. Federated Learning of Deep Networks using Model Averaging. *CoRR*, abs/1602.05629.

Mei, G.; Guo, Z.; Liu, S.; and Pan, L. 2019. Sgnn: A graph neural network based federated learning approach by hiding structure. In *2019 IEEE International Conference on Big Data (Big Data)*, 2560–2568. IEEE.

Mitliagkas, I.; Zhang, C.; Hadjis, S.; and Ré, C. 2016. Asynchrony begets momentum, with an application to deep learning. In *Communication, Control, and Computing (Allerton), 2016 54th Annual Allerton Conference on*, 997–1004. IEEE.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *CoRR*, abs/1912.01703.

Rogers, D.; and Hahn, M. 2010. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5): 742–754.

Rong, Y.; Bian, Y.; Xu, T.; Xie, W.; Wei, Y.; Huang, W.; and Huang, J. 2020. Self-Supervised Graph Transformer on Large-Scale Molecular Data. *Advances in Neural Information Processing Systems*, 33.

Schütt, K. T.; Arbabzadah, F.; Chmiela, S.; Müller, K. R.; and Tkatchenko, A. 2017. Quantum-chemical insights from deep tensor neural networks. *Nature communications*, 8(1): 1–8.

Smith, V.; Chiang, C.-K.; Sanjabi, M.; and Talwalkar, A. S. 2017. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, 4424–4434.

Suzumura, T.; Zhou, Y.; Baracaldo, N.; Ye, G.; Houck, K.; Kawahara, R.; Anwar, A.; Stavarache, L. L.; Watanabe, Y.; Loyola, P.; et al. 2019. Towards federated graph learning for collaborative financial crimes detection. *arXiv preprint arXiv:1909.12946*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. arXiv:1710.10903.

Wang, B.; Li, A.; Li, H.; and Chen, Y. 2020. GraphFL: A Federated Learning Framework for Semi-Supervised Node Classification on Graphs. *arXiv preprint arXiv:2012.04187*.

Wang, J.; and Joshi, G. 2018. Cooperative SGD: A unified Framework for the Design and Analysis of Communication-Efficient SGD Algorithms.

Wang, J.; Kolar, M.; and Srerbo, N. 2016. Distributed multi-task learning. In *Artificial Intelligence and Statistics*, 751–760.

Wu, Z.; Ramsundar, B.; Feinberg, E. N.; Gomes, J.; Geniesse, C.; Pappu, A. S.; Leswing, K.; and Pande, V. 2018. MoleculeNet: a benchmark for molecular machine learning. *Chemical science*, 9(2): 513–530.

Yang, T.; Zhu, S.; Jin, R.; and Lin, Y. 2013. Analysis of distributed stochastic dual coordinate ascent. *arXiv preprint arXiv:1312.1031*.

Yu, H.; Yang, S.; and Zhu, S. 2018. Parallel Restarted SGD with Faster Convergence and Less Communication: Demystifying Why Model Averaging Works for Deep Learning. *arXiv:1807.06629 [cs, math]*. ArXiv: 1807.06629.

Zhang, Y.; and Yeung, D.-Y. 2012. A convex formulation for learning task relationships in multi-task learning. *arXiv preprint arXiv:1203.3536*.

Zheng, L.; Fan, J.; and Mu, Y. 2019. Onionnet: a multiple-layer intermolecular-contact-based convolutional neural network for protein–ligand binding affinity prediction. *ACS omega*, 4(14): 15956–15965.