Towards Decentralized Distributed Learning for Dynamic Edge Networks

Towards Decentralized Distributed Learning for Dynamic Edge Networks

by

Ryan Jones

Under the Direction of Committee Chair Yue Wang, Ph.D.

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

in the College of Arts and Sciences

Georgia State University

2025

ABSTRACT

As Machine Learning (ML) becomes ever more prevalent across disciplinary boundaries and throughout society's innovations, technological requirements and advancements pull the storage of data and responsibility of computation towards the edge. Federated Learning (FL) began a new wave of algorithms designed for distributed learning. Research in Machine Learning is now progressing even further from distributed to decentralized distributed machine learning. This requires additional considerations such as the limited computational power and communication resources which characterize systems utilizing wireless networks. Current decentralized learning algorithms are not in compliance with these strenuous limitations. We introduce a new algorithm, Peer-to-Peer Critical-Infrastructure-Free Distributed Swarm Learning (PC-DSL), which leverages the characteristics of edge and wireless networks to optimize fully decentralized distributed learning. PC-DSL reduces the maximum number of communications of parameter weight vectors to 1 per agent per step while retaining an 88% testing average on MNIST with 300 training points at 50 agents.

INDEX WORDS: Machine learning, decentralized learning, internet of things, edge devices, peer-to-peer

Towards Decentralized Distributed Learning for Dynamic Edge Networks

by

Ryan Jones

Committee Chair: Dr. Yue Wang

Committee: Dr. Anu G. Bourgeois

Dr. Jingyu Liu

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

August 2025

DEDICATION

I first want to thank my wonderful family for supporting me all along my academic journey. My mother always led by example while challenging me to strive further whether it be in my moments of success or setback. To my father, a leader of people, who shows that greatness is achieved through the balance between knowledge and the unknowable and that one's organizing principles can be a resolute comfort in times of success and failure. To my younger sister, the first and foremost of the future for whom we carry the torch ever more into the unknown illuminating our collective understandings for a brighter tomorrow. Finally, to all those other unnamed persons who have motivated me, supported me, and inspired me to seek into the unknown in hopes of discovering further truths.

ACKNOWLEDGMENTS

Thank you to my thesis committee. To Dr. Yue Wang for introducing me to distributed learning algorithms. To Dr. Anu for creating an environment for me to discover a wide variety of information regarding computer networking and for creating an environment conducive to academic excellence. To Dr. Liu who introduced me to many key aspects of digital image processing. I would also like to extend an extra thank you to Center for Human Artificial Intelligence Interaction as well as all the faculty and staff at Georgia State University and Rhodes College who made this work possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
Introduction

Recent developments in wireless communication technologies and computer networking have led to a marked increase in the prevalence of Internet of Things (IoT) systems that employ edge-distributed communication paradigms. Illustrative examples of these systems include wireless sensor networks, autonomous vehicles, micro-satellite arrays, and drone-enabled disaster recovery systems. Due to their inherent hardware constraints, limited computational capacities, extensive multiplicity of agents, prevalence of device heterogeneity, and lack of security feature guarantees, edge systems present distinct challenges and considerations compared to traditional centralized network architectures Jin et al. (2022). Simultaneously, the swift advancements in computing capabilities and the enhanced accessibility of large-scale datasets have positioned machine learning as a fundamental pillar of innovation across a myriad of disciplines in both the academic sphere and industrial sectors. In this context, the integration of machine learning methodologies for implementation in edge-distributed systems presents a promising avenue for leveraging distributed learning to tackle the distinct tasks necessitated by edge networks.

A fundamental requirement of distributed learning is the ability of a learning agent to achieve convergence on a set of weights for a shared model while operating within the constraints imposed by agent data parallelism. A seminal advancement in the solution to this problem is federated learning (FL) Wen et al. (2023), in which distributed agents independently train their local instances of a global model on their respective subsets of data during specified rounds. Following this, the updated model weights are subsequently aggregated by a central parameter server, which amalgamates them to facilitate the update of the global model for the subsequent training iteration. This methodology encounters limitations when implemented within certain edge-distributed communication paradigms. Primarily, numerous such edge paradigms lack the requisite infrastructure to support a centralized server for the orchestration of model weight aggregation. Moreover, the communication overhead of FL escalates significantly with greater model complexity, an increased number of agents, and/or the expansion of network scales characterized by link sparsity.

Facing the limitations mentioned above in edge scenarios, some recent research efforts have designed distributed learning algorithms for peer-to-peer communications. For example, FL has been customized to adopt a gossip protocol to calculate and distribute model weights in a peer-to-peer network Naik et al. (2023). However, some issues remain with this approach, including slow agent synchronization and degraded learning performance due to network dynamics. Another approach that has been introduced to FL is to use blockchain infrastructure to securely decentralize the learning process Shayan et al. (2018). Although this algorithm is applicable to peer-to-peer distributed learning, strictly organized responsibilities and high resource requirements make such a method not feasible for deployment in dynamic self-organized systems at the edge.

To fill this gap, this paper aims to develop an adaptable distributed learning algorithm for edge-distributed learning that is tunable to the requirements of diverse applications in the absence of a centralized server. We propose a new decentralized distributed learning algorithm called peer-to-peer critical-infrastructure-free distributed swarm learning (PC-DSL), which pushes the computation and communication to the edge devices of networks while considering their hardware constraints. Based on previous work on distributed swarm learning (DSL)Wang et al. (2024); Fan et al. (2024), we leverage a holistic integration between stochastic gradient descent (SGD) and particle swarm optimization (PSO) for distributed learning at the edge, with advantages such as communication efficiency, Byzantine robustness, trainability on non-i.i.d. data, and fast convergence. Our PC-DSL inherits these advantages, while also benefiting from many of the common advantages of previous edge-distributed adaptations of FL discussed in the related literature. Beyond this, PC-DSL further enhances the collaboration gains of multi-agent learning systems, which allows for a boost in adaptation to system dynamics and an enhancement in system robustness to link failures.

1.1 Objectives

This publication advances the field of decentralized distributed learning by making the following contributions: an evaluation of the suitability of existing decentralized distributed learning algorithms for application in systems characterized by the presence of edge devices, or those bearing the limitations inherent to edge devices; the introduction of a novel decentralized distributed learning algorithm specifically tailored for the constraints associated with edge deployment, as well as its variants and their potential use cases; and an analysis of the attributes of this proposed algorithm, alongside the implications of these findings.

1.2 Thesis Outline

The structure of the subsequent sections of this work is delineated as follows. Chapter 2 conducts a review of existing literature pertinent to decentralized distributed learning, encompassing critical details regarding the requirements of edge devices, distributed learning algorithms, and established decentralization methods. Chapter 3 delineates specific constraints, requirements, and evaluation metrics pertinent to decentralized distributed learning. Chapter 4 presents a comprehensive proposal of the PC-DSL algorithm. Chapter 5 comprises a series of experiments investigating the characteristics of decentralized distributed learning, aligned with the paradigm established by PC-DSL. Chapter 6 scrutinizes the implications of these results, proposes future research directions, and concludes with a summary of the publication's content and findings.

CHAPTER 2

Related Works and Literature Review

2.1 Federated Learning(FL)

In 2016, a team from Google Research published a paper in the 20th Proceedings of the International Conference on Artificial Intelligence and Statistics, titled ”Communication-Efficient Learning of Deep Networks from Decentralized Data.” This paper introduced Federated Learning, Federated Optimization, and two algorithms: FedSGD and FedAVG. This publication laid the groundwork for the majority of modern distributed learning algorithms. In this paper, the FedAVG algorithm outperformed FedSGD. A system utilizing FedAvg distributes a model to all workers every round. The worker then completes a gradient descent step and sends its weight vector to a parameter server where the next round's model is generated by averaging all the weight vectors from the worker nodes McMahan et al. (2023).

Federated Learning possesses several favorable characteristics for edge distributed systems such as privacy preservation, quick convergence, and well-studied performance on non-i.i.d. data McMahan et al. (2023) Li et al. (2020). However, the communication overhead of FL is expensive as the model is required to be transmitted at every communication round from every agent to the parameter server, and then the parameter server must distribute the model for use in the next round of gradient descent. In response to this, several optimizations have been introduced, which limit the communication frequency of local agents who contribute to local model updates or limit the amount of contribution they send Konečný et al. (2017) Mao et al. (2022) Chen et al. (2021).

Federated learning has been the progenator of many decentralized distributed learning algorithms. Perhaps the closest of these to the original is Clustered Hierarchical Distributed Federated Learning. This algorithm utilizes a structured system of clusters where each cluster distributes responsibilities among the agent nodes. The model is divided into multiple sections, each section being computed by a distinct subset of agents Gou et al. (2022). Additional decentralizations of Federated Learning will be discussed in section 2.3.

2.2 Gossip Learning

Gossip learning was among the earliest algorithms developed for decentralized distributed learning. In this framework, agents exchange update information with adjacent agents, who are subsequently responsible for assimilating these updates into their own revised weight vectors. Hegedundefineds et al. (2019). Despite the presence of similarities, Gossip Learning should not be misconstrued as a decentralized extension of Federated Learning. The communication protocol employed by Gossip Learning for inter-agent exchange, as well as its aggregation techniques, are methodologically akin to those utilized within the framework of PC-DSL. To mitigate communication overhead, agents can transmit only subsets of their parameter weights.

2.3 Machine Learning Attack Resilience

There are 4 major classifications for attack vectors to which machine learning at the edge is susceptible: poisoning attacks, injection attacks, adversarial example attacks, and Byzantine attacks Jin et al. (2022). Many traditional systems for distributed learning employ measures at the central parameter server to identify and prevent the possibility that non-trusted agents or data sources are exploiting any such attack vectors. In decentralized distributed learning, alternative methods of privacy and security assurance are required which do not depend on a centralized server. A common means to ensure trustworthy collaboration in decentralized compute processes is the utilization of a block chain. Göndör et al. (2022) Yuan et al. (2021). This trend continues when looking at emerging decentralized distributed learning algorithms. Biscotti is a combination of multiple existing techniques. It employs a blockchain system with different members taking on separate roles for the training round: noiser, verifier, aggregator. For identification of attacks, it uses multi-KRUM. This algorithm relies on stake for selection with external rewards Shayan et al. (2018). Another decentralized distributed learning algorithm, Brave, consists of a five-stage system. In the first stage, each agent forms a commitment, which it broadcasts using Byzantine Fault-Tolerant (BFT) broadcast. The second stage involves a privacy-preserving comparison of the models. The result is then used to select the contributors. Subsequently, the models are aggregated, producing the next round Xu et al. (2024). While both methods enhance security, they concurrently exacerbate the computational and communication overhead associated with the algorithm.

2.4 Distributed Swarm Learning (DSL)

Distributed Swarm Learning (DSL) represents a distributed learning algorithm that integrates the principles of Federated Learning (FL) and Particle Swarm Optimization (PSO). The principal advantages delineated in the initial proposal for DSL, Fan et al. (2022), include expedient convergence, reduced communication overheads, resilience against local optimum traps, relaxed device restrictions, and enhanced robustness to node and link failures. For the objectives of this publication, these benefits can be systematically divided into two discrete categories. The first category comprises general learning advantages, which encompass benefits observable even within fully centralized learning frameworks. The convergence time of an algorithm is a crucial factor in nearly all learning scenarios, as it relates to both the temporal and computational efficiency of the algorithm. The remaining advantages are more specifically associated with distributed learning. These attributes become increasingly advantageous in an edge-distributed system with elevated constraints on computation and communication.

In the framework of Distributed Swarm Learning (DSL), instead of the parameter server overseeing the aggregation and dissemination of each worker's model weight vector, the server is responsible for computing and distributing a global optimal weight vector. This vector is subsequently utilized by each client in their forthcoming DSL update. The integration of Particle Swarm Optimization with Federated Learning is encapsulated within the subsequent equation.

$$\mathbf{w}_{i,t+1} \leftarrow \mathbf{w}_{i,t} + c_0\mathbf{v}_i + c_1(\mathbf{w}^p_{i,t} - \mathbf{w}_{i,t}) + c_2(\mathbf{w}^g_{i,t} - \mathbf{w}_{i,t}) - \alpha\Delta F_i(\mathfrak{D})$$

This update constitutes a weighted summation of the current weight vector and several additional terms. The initial term represents the velocity, scaled by a constant factor c_0. The subsequent term embodies the difference vector, calculated between the client's prior best weight vector and the present weight vector, also multiplied by a constant c_1. The third term is the difference vector between the global best weight vector and the current weight vector, similarly scaled by a constant c_2. Lastly, to aid in guiding the exploration, a gradient descent term is incorporated. Fan et al. (2022)In the context of DSL, there exists a configuration that employs a complete randomized scaling of the additional weights vector within the interval of zero to one. This mechanism resembles a synthesis of the two aggregation methodologies employed in Gossip Learning, resulting in the new vector being located within the hypercube delineated by the two terms.

CHAPTER 3

Design Specifications

3.1 System Requirements and Considerations for Edge Distributed Systems in both Generalized and Machine Learning Specific Cases

Due to the specific characteristics of edge devices and the networks that connect them, such as limited computational capability, massively multiple numbers of agents, prevalence of device heterogeneity, and lack of security feature guarantees, edge systems pose additional unique considerations compared to traditional architectures. These considerations must be addressed all while minimizing the impact on system performance Jin et al. (2022). No existing method for decentralized distributed learning meets all the considerations of edge devices, nor does the algorithm proposed in chapter 4. This suggests that the optimization of a specific requirement will inevitably necessitate a trade-off with another performance metric or adherence to an alternative constraint. For example, the quantization of weight vectors could effectively diminish the size of the communication model, albeit at the potential cost of increased performance degradation. The considerable quantity of stringent constraints intrinsic to particular applications on edge devices markedly limits the flexibility of any specific solution functions, while necessitating enhanced adaptability from any solution paradigm that endeavors to be adjustable to diverse situations.

3.2 Security and Privacy of Decentralized Distributed Learning

While the scope of this paper is confined to the application layer, particularly focusing on the machine learning training phase, it must be recognized that the performance, security, and privacy of the application are intrinsically reliant on the performance, security, and privacy across all other layers. We now re-examine the four categories of attacks. The significance of addressing these attacks in the design of a machine learning solution may vary depending on the specific characteristics of the network in which the application is intended to be deployed. Subsequently, the resilience against each type of attack necessitates varying levels of trade-off concerning the costs they may incur. For example, consider a network of satellites cooperating to train a collective model for forecasting wildfire spread. The logistical difficulty of deliberately starting wildfires to corrupt data designated for the workers is markedly different from the costs entailed by conducting a Byzantine attack. Therefore, the distribution of resources to address each of these threats should be adjusted to accommodate this variation.

3.3 Domain Specific Definitions

While existing methods, such as those expounded in chapter 2, traditionally conform to classifications of critical infrastructure-free or peer-to-peer, this paper aims to present refined and elevated definitions for these terms. We define Critical-Infrastructure-Free to be that throughout the network, there should be no subset of agents raised to criticality for the successful learning of a hypothesis within the bounds of approximation error at a probability below a threshold in comparison to the results of removing another set of nodes selected at random. An illustrative system would be a distributed approach to power grid management necessitating the highest feasible resistance to interference at any particular site. We define peer-to-peer to be that any communication between 2 parties is not mediated or reliant on a third party, and neither party has elevated status or responsibility beyond that of its own needs. In the case of a single-hop peer-to-peer, information is limited to only immediate neighbors, and any information that passes further is originated by the new agent as a representation of its own gleaned information. An exemplary system that may benefit from this elevated single hop constraint would be a health analytics application, wherein users have the capability to restrict the sharing of their models to only those entities they explicitly designate.

3.4 Requirements for the Decentralization of Distributed Swarm Learning

To delineate the fundamental components requisite for the decentralization of DSL, an initial step involves deconstructing the algorithm's mechanisms and communication requirements. In the context of DSL, each agent performs updates at every communication iteration by employing a weighted summation of vectors derived from both the global optimum w_t^G and the individual agent's optimum $w_{i,t}^P$ in conjunction with its current weight vector, in addition to terms related to velocity and gradient descent Hegedundefineds et al. (2019). This means that the decentralization of such an algorithm's primary concern is the synchronization of the global term; however, there are also secondary objectives to address. To reduce the communication cost of DSL, CB-DSL delays the transmission of the weights vector until after global best contributor selection, as the cost of communicating a parameter weights vector eclipses that of communicating a loss value Fan et al. (2022). To enhance the resilience of the algorithm against Byzantine attacks, CB-DSL employs the parameter server to verify the fair loss value of the weights vector before incorporating it into the generation of the global best parameter weight vector w_g Fan et al. (2022). This mechanism is essential for CB-DSL's defense against Byzantine attacks. Decentralized algorithms, due to the absence of a parameter server, are required to find alternative means of mitigating such attacks.

3.5 Evaluation and Comparison of Decentralized Distributed Learning Algorithms

In the assessment of distributed learning algorithms, the primary emphasis is on the comparison of optimization processes rather than testing accuracy McMahan et al. (2023). This can be seen in the proposal of Gossip Learning as although it is compared to federated learning the primary benefits are its decentralization while minimizing degradation of comparative performance Hegedundefineds et al. (2019). Therefore, our evaluation of these methods employed standardized resources to effectively isolate and discern the impact of the learning or optimization technique. To facilitate a comprehensive comparison of the generalization capabilities of these methods, our evaluations were conducted through multiple experimental iterations. In the context of decentralized comparison, although algorithms may be evaluated on identical tasks, such comparison does not accurately reflect their actual utility, as the constraints fulfilled by these algorithms manifest in varying contexts. With this in mind our comparisons are undertaken with an understanding of the wider costs and benefits provided by each solution.

The evaluation metrics for learning performance employed in this comparative analysis do not necessitate the use of application-specific measures, which would typically be required for a comprehensive assessment of the particular task, as previously discussed. Consequently, for the sake of simplicity, the evaluation of learning and optimization performance presented herein is restricted to accuracy just as was the evaluation of DSL Hegedundefineds et al. (2019). Furthermore, we conducted a comparative assessment of the various costs associated with the different methodologies and the specific constraints that each algorithm can satisfy or fails to fulfill.

3.6 Format of Existing Experimentation

For the specific applications to be tested in this thesis, we will use variations of a subset of the evaluations used in the existing related work, specifically, the proposals of Distributed Swarm Learning and Federated Learning. As such, we have included an overview of relevant experimentation in this section.

The experimentation for the proposal of DSL was carried out using the MNIST dataset for the purpose of handwritten digit classification. A convolution neural network (CNN) comprising 5 layers, as described in Table 5.1, was employed. The simulation incorporated 50 workers, each equipped with a dataset containing 500 samples, in addition to a global dataset comprising 2000 samples. This data was subsequently partitioned into testing and validation datasets at a rate of 0.4. The simulation comprised 10 rounds, with each round consisting of 3 epochs of Stochastic Gradient Descent (SGD). Each worker minimized the mean with respect to the loss functions. The experimental results demonstrated improved performance coupled with reduced communication costs compared to the existing methodologies Fan et al. (2022).

The study on Federated Learning, which compares FedSGD and FedAVG algorithms, involved training a two-layer fully connected neural network, a five-layer Convolutional Neural Network comprised of two convolutional layers, two fully connected layers, and one linear layer, as well as a Long Short-Term Memory neural network. This analysis was conducted using the CIFAR-10 dataset, the MNIST dataset, and a dataset derived from the works of William Shakespeare. For the MNIST dataset, data was allocated to 100 clients, each receiving 600 data instances. In instances of non-independent and identically distributed (non i.i.d.) data, partitioning was performed based on labels, with data being evenly distributed across two partitions for each worker.

CHAPTER 4
Proposed Algorithm

In this section, we propose a new algorithm – Peer-to-Peer Critical-Infrastructure-Free Distributed Swarm Learning (PC-DSL) – as a decentralization of the Distributed Swarm Learning algorithm with the incorporation of additional mechanisms from other decentralized distributed learning algorithms, primarily Gossip Learning.

We propose Peer-to-Peer Critical-Infrastructure-Free Distributed Swarm Learning (PC-DSL) as illustrated in Algorithm 1. Like DSL, PC-DSL takes place with epochs of SGD with synchronization steps highly reminiscent of PSO. In order to facilitate decentralization, it introduces a lightweight adaptive methodology for computing what is designated as the global term by DSL, as delineated in Algorithm 1. Beyond this, PC-DSL supports configuration for localization of the universality of the global model, delayed communication of models, and alternative means of model loss evaluation.

4.1 Global Synchronization

To compute the global term used during the PSO update step of DSL, PC-DSL agents maintain local pseudo-global tables. We will refer to these tables as local pseudo-global tables and the model resulting from the aforementioned computation as the local pseudo-global weight vector. This naming convention represents that, while the weight vector is utilized for the global term in the PSO update step, this vector is not global as it is calculated locally and may have different values at different workers at the same time. When required for its own PSO step, that agent is responsible for calculating the global term based on the variant

Algorithm 1 PC-DSL

Let G be a network of K agents denoted as a_i
Let each agent a_i have data D_i, weights w_i, previous best weights w_i^p, and local pseudo-global table t_i
Let $L(w, x)$ be the loss value of weights w with respect to data x
function SGD Step(a_i)
  $w_i \leftarrow w_i - \alpha_i \nabla L(w_i, x_{i,train})$
  $\alpha_i \leftarrow \alpha_i * \gamma$
  if $L(w_i, x_{FC}) < L(w_i^p, x_{FC})$ **then**
    $w_i^p \leftarrow w_i$
    Update Table(a_i,w_i)
  end if
end function
function PSO Step(a_i)
  $w_i^g \leftarrow$ Get Global(a_i)
  $v_i \leftarrow c_0 v_i + c_1(w_i^p - w_i) + c_2(w_i^g - w_i)$
  $w_i \leftarrow w_i + v_i$
  if $L(w_i, x_{FC}) < L(w_i^p, x_{FC})$ **then**
    $w_i^p \leftarrow w_i$
    Update Table(t_i,w_i)
  end if
end function
function Update Table(a_i,w_j)
  if t_i contains some w from a_j & $L(w_j) < t_i[j]$ **then**
    $t_i[j] \leftarrow w_j$
    if TTL$>$ 0 **then**
      Broadcast w_j
    end if
  end if
  if t_i does not contain some w from a_j & $L(w_j) < min_x(L(t_i[x]))$ **then**
    $t_i[j] \leftarrow w_j$
    if TTL$>$ 0 **then**
      Broadcast w_j
    end if
  end if
  if t_i does not contain some w from a_j & $|t_i| < S$ **then**
    $t_i[j] \leftarrow w_j$
    if TTL$>$ 0 **then**
      Broadcast w_j
    end if
  end if
end function

of DSL employed by the system. Table contributors are enforced to contain only a singular model from a given source agent. The configuration of the table length can be adjusted to accommodate additional models beyond the stipulations of multi-contributor PC-DSL. Beyond the advantage of flexibility regarding DSL configuration of multiple selectors, the adaptive length of the local pseudo-global table provides additional support for attack resistance and further exploration expansion. This adaptive capacity of the local pseudo-global table's length offers further support for features that will be discussed subsequently. The trade-off of large tables includes additional resource costs for computation, communication, and storage. Upon updating a table, the agent transmits the model to its neighbors, which in turn handle the received weights accordingly.

4.2 Privacy and Model Sharing

An extra variable is incorporated into our implementation, facilitating variable scope and thereby constraining the dissemination of a model across the network implemented as a Time To Live (TTL). By constraining the permissible number of propagation hops to a single level, this implementation ensures a wholly one hop peer-to-peer architecture wherein weight vectors are distributed exclusively to agents that are immediate neighbors of the originating node. The sparsity of the network increases the impact of limited scope due to agents in sparse networks having fewer immediate neighbors. This feature protects user security by mitigating the number of peers which receive the model to those closer to the initial node, subsequently reducing the likelihood of attacks which attempt to retrieve knowledge of an agent's model. We expect practical deployments to enact one of two extreme cases: either an unlimited communication scope or a scope of a single hop.

4.3 Communication Efficiency

To address the requirements of communication Efficiency, We proposed several adaptive solutions. The first is to delay any weight vector communication until the vector is requested by a neighboring node. This increases the communication efficiency of the algorithm at the cost of time; however, the cost in time is negligible compared to the delay caused in a centralized system as the distances traveled and time for the propagation of the loss value and the model request are many times shorter as they only must propogate a single link. The second is to utilize the optimization introduced by Gossip Learning to transmit only partial updates Hegedundefineds et al. (2019). This update is only a partial step further than the current randomized scaling of full random DSL.

4.4 Data Privacy and Global Data Elimination

We propose alternative methods to evaluate the fair comparison loss values for local pseudo-global candidate selection $L(w, x_{FC})$. We hypothesize that the selection of possible global best parameter weight vectors on the grounds of approximate generalization error is sufficient for the selection of a globally best model weights vector.

In the case of i.i.d. data, due to both methods being approximations of generalization error to the true distribution, local validation data may be used in place of global data for the calculation of fair loss values $L(w, x_{FC}) = L(w, x_i)$. The probability trade-off of this method is a decreased likelihood that the majority of agents will be calculating approximate generalization error against a non-representative sample, while increasing the risk that at least one member will have a non-representative sample.

For non-i.i.d. data, losses must be calculated at multiple agents to effectively approximate generalization error. The two methods for querying the losses are evaluating based on all neighbors of the agent $L(w, x_{FC}) = \sum_n^N L(w, x_n^v)$ where x_n^v is the validation data at neighbor n, or signing a promise and adding your loss value as it propagates through the network $L(w, x_{FC}) = \sum_p^P L(w, x_p^v)$ where x_p^v is the validation data at agent p along the path traveled by the weights in a scope greater than 1. The first method is favorable in dense networks or with limited scope, while the second is favorable in sparse networks with large communication scopes. In the first case, the querying node will know the loss values of all neighbors. In the second case, the only known value will be that of the direct neighbors, as well as the leaf nodes by their neighbors. Additionally, if the deployment case requires increased data privacy and security at the cost of resources, more advanced methods of querying may be employed, such as those found in both Biscotti and Brave which utilize Multi-KRUM Xu et al. (2024) Blanchard et al. (2017).

CHAPTER 5

Experimentation

5.1 Testing Framework

In the simulations, we began by comparing PC-DSL to DSL Fan et al. (2023). We used a standard Convolutional Neural Network (CNN) with five layers to classify the MNIST data set. The specifications of which can be seen in Table 5.1. Fifty simulated agents were placed in a three-dimensional space according to a normal distribution. Links between nodes were established between agents within a specified distance. The timing of the execution of events and model comparison was represented by simulated time with configurations for model transmission and propagation, as well as for other events such as the PSO and SGD steps at each agent. This configuration served as the standard framework for all experimental procedures unless otherwise specified.

5.2 Decentralized and Centralized Comparison

To establish a baseline comparison between the centralized and decentralized algorithms, we compared the simulated time taken for the synchronization of models each round in

Table 5.1 Layers of Convolutional Neural Network(CNN) used in DSL and PC-DSL experimentation

MNIST CNN
Conv2d(1, 6, 5)
Conv2d(6,16,5)
MaxPool2d(2,2)
Linear(512, 120)
Linear(120,84)
Linear(84,10)

a synchronously timed system with a non-limited communication scope and global dataset. The ML performance and convergence in these two systems are the same as in the centralized algorithm, since the best global models produced are the same. For the timing of the centralized method, we position the server according to the same normal distribution as the agents.

The PC-DSL algorithm synchronized on average faster each round than the centralized equivalent in all trials. Furthermore, as the sparsity of the graph increases, the advantage of the decentralized method also increases. The time for global best model weight synchronization in the centralized algorithm is the sum of the time required to gather all the losses of the model from each worker, notify the selected workers, gather the selected models and distribute the best calculated global model. In PC-DSL with multi-hop, the time required for synchronization can be calculated as the time for the best models to propagate across the network with each agent receiving loss values and subsequently requesting models as necessary. The variation in time is due to both the configuration of the graph and the location of the workers that have the best global models. In the case of single-hop PC-DSL time required for communication is even shorter as it is limited to the time for communication over a single link.

5.3 Effect of Scope Limitation

In order to investigate the impact of constraining the communication scope within a decentralized DSL algorithm, we extend our analysis beyond the conventional comparison of an

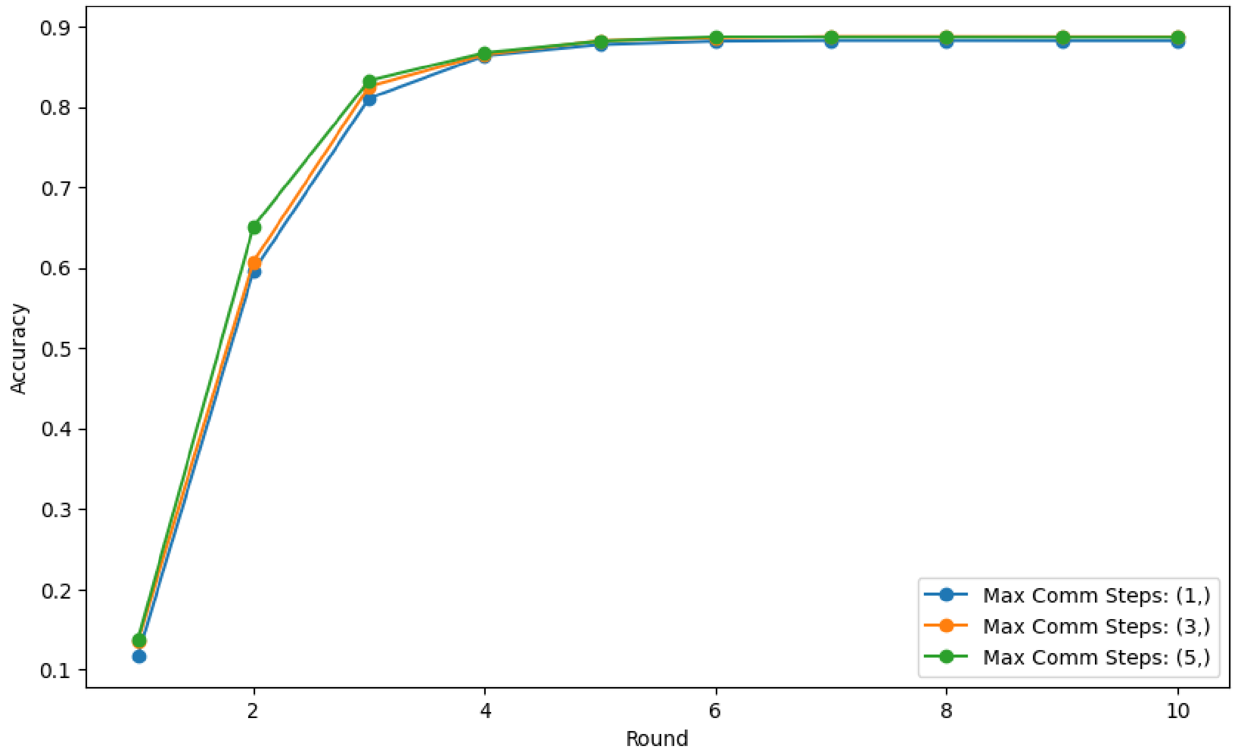

Figure 5.1 Average test accuracy across the network at each communication step with limited communication scope Waskom (2021) Hunter (2007)

unbounded versus completely restricted communication scope. We did this by modulating the initial Time-to-Live (TTL) set for each model being communicated across the network. For this experiment, requiring the graphs' construction to exhibit sparsity was critically significant, as the imposition of constraints on the communication scope yields no effect on a fully connected graph and only minimal influence on a dense graph in contrast to the influence on a sparse graph.

As illustrated in Figure 5.1, the imposed restriction in scope resulted in an insignificant degradation of the final model's performance. Nonetheless, a reduction in the maximum permissible number of hops was associated with a deceleration in convergence and an increase in variability. It is pertinent to note that, relative to numerous contemporary applications of machine learning, this scenario is comparatively simplistic. Therefore, further investigation is warranted to assess the scalability of this algorithm.

5.4 Effect of Asynchronous Learning

To empirically ascertain the impacts of asynchronous learning, in each worker, uniform variability was systematically introduced in the timing of the execution of the SGD and PSO step. This timing variability was significant enough to overlap the training rounds so that agent i could be executing a fifth PSO update step before agent j completes a fourth step. This was juxtaposed with the performance metrics observed when the temporal execution of steps was uniform across all agents within the network. The variability was introduced so that the average time was expected to be equivalent in both models. In both cases, the TTLs were configured for single-hop communication.

As can be seen in Figure 5.2, unlocking the set timing of training rounds allows information to propagate through the network earlier and, consequently, enhances the performance of model training. These results demonstrate potential in adeptly accommodating the timing variability inherent in edge systems. In addition, the enhanced ability of workers to accelerate convergence speed can lead to a reduction in operational costs for training in a wide variety of systems.

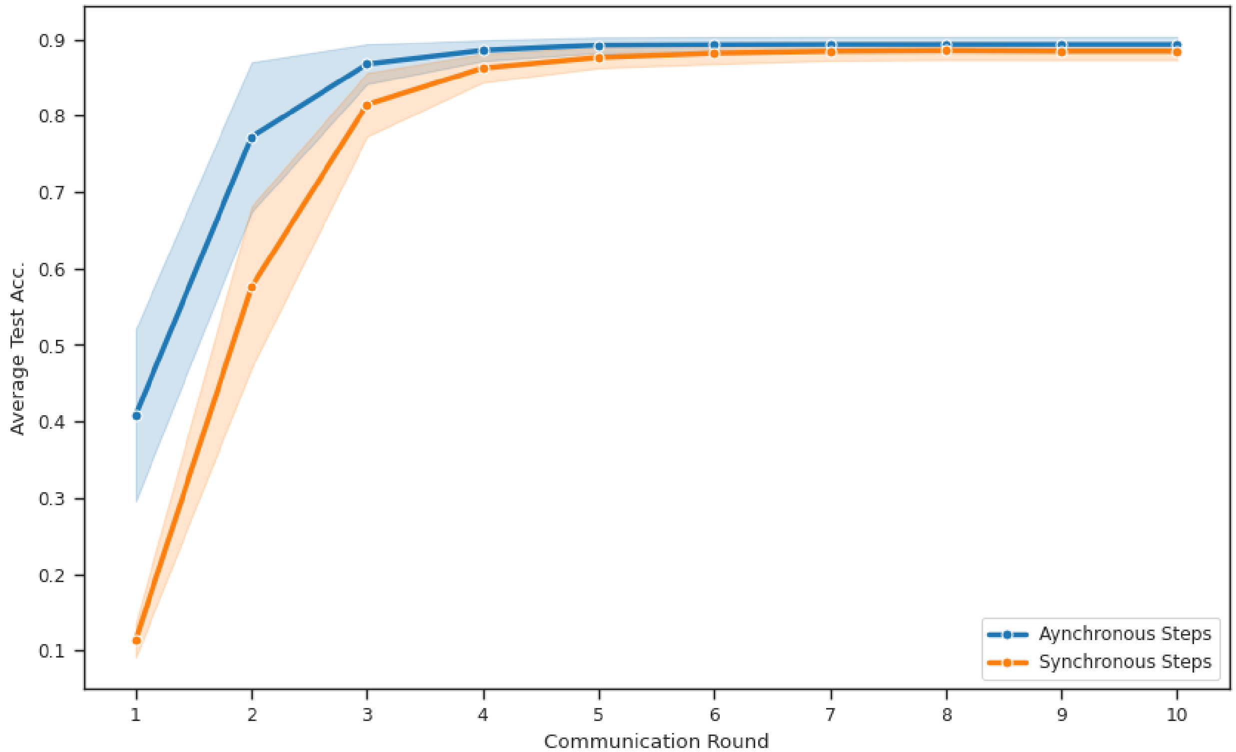

Figure 5.2 Comparison between the training curves for PC-DSL locked to synchronous time steps versus asynchronous with 95% confidence interval Waskom (2021) Hunter (2007)

5.5 Effect of the Removal of the Global Data Set

Within this work, we confine the scope of experimentation on the mechanisms of generalization error approximation for the purpose of contributor selection to the simplified scenario of assessing the fair loss function on local validation datasets, assuming i.i.d. data. Each agent received its share of the 2000 global data points, resulting in 540 total data points per agent. To balance this, we increased the cross-validation rate to 0.25, keeping the data allocated for training close to the initial value.

The loss value calculated at each step serves as an approximation of the generalization

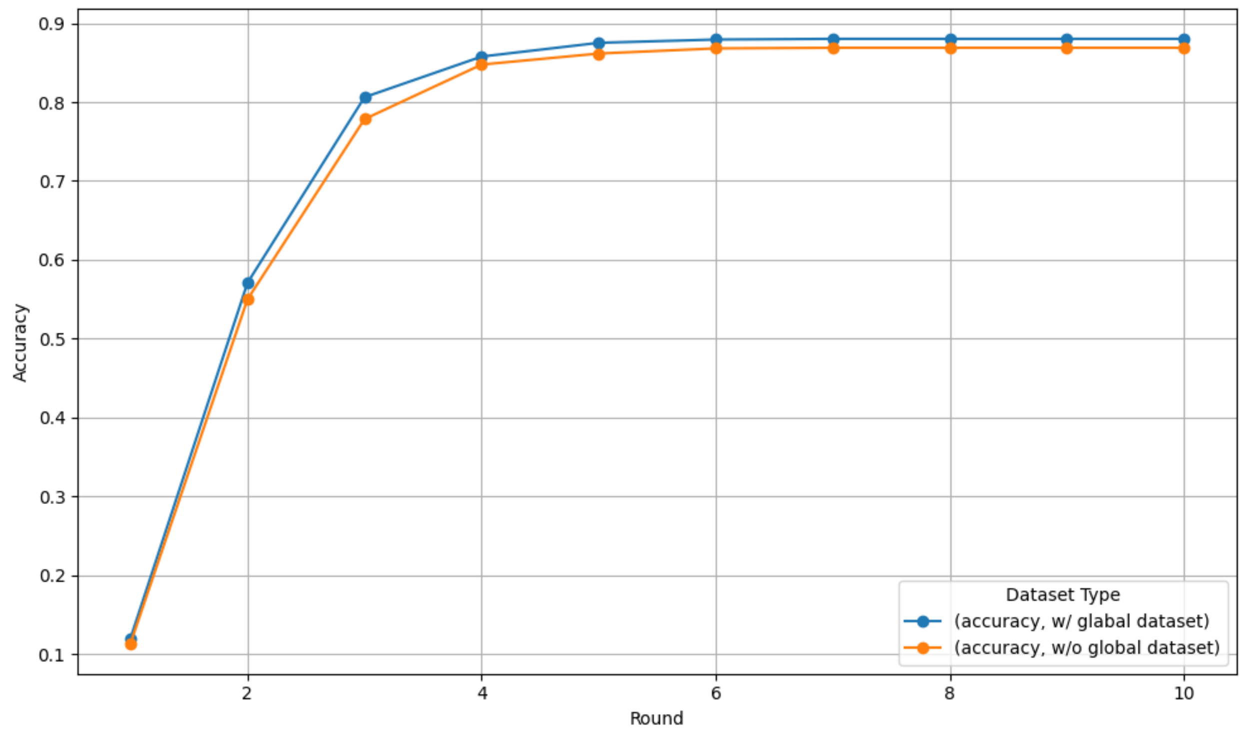

Figure 5.3 Comparison between the Testing accuracy over time with and without the presence of global data.

error of the given parameter weight vector. Note that the size of each validation set is only 135 data points compared to the global dataset's 2000 data points. This implementation is only practical in extremely low-budget systems where security is either not valued or not possible. As can be seen in Figure 5.3, the establishment of a global dataset maintains higher performance throughout the training rounds.

5.6 Effect of Dynamic Network Topology

In order to examine the impact of dynamism on the network's topology, the assumption that the network existed as a singular strongly connected component was discarded. Subsequent

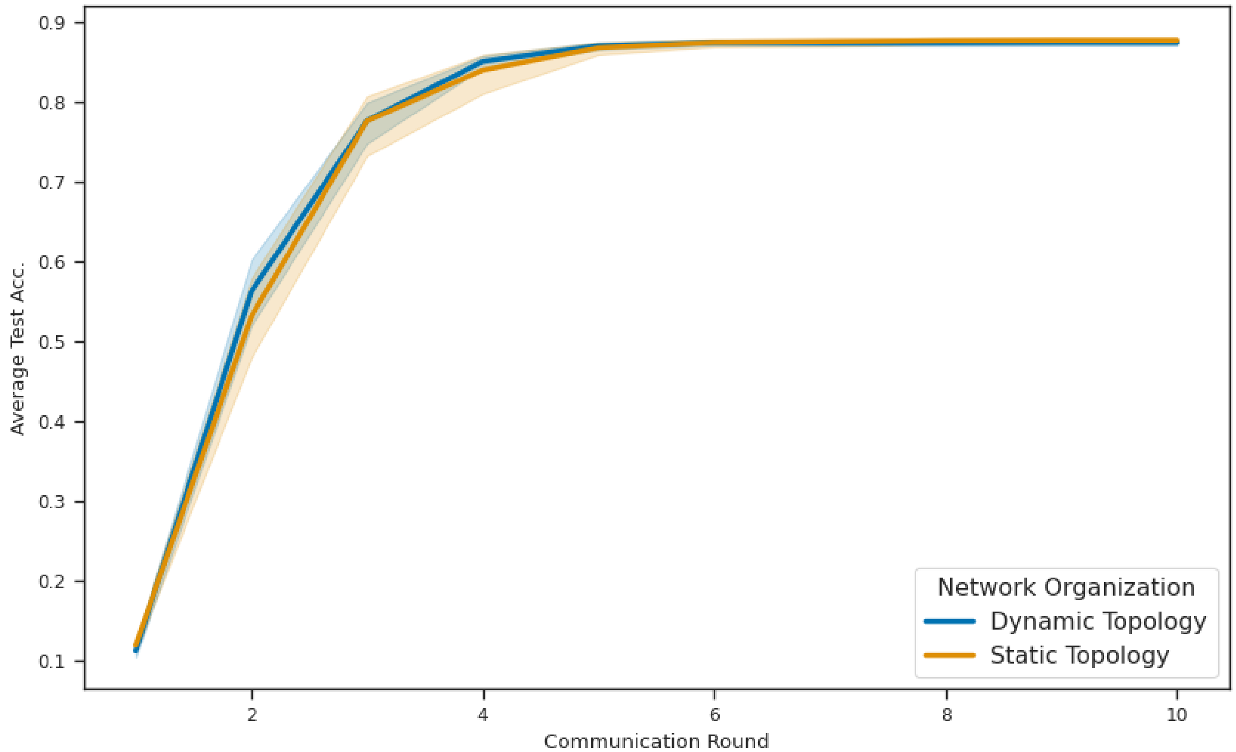

Figure 5.4 Comparison of average and 95% confidence interval between the testing accuracy over time in a static layout and with artificial churn.

to this modification, new events were introduced to randomly reposition agents. Upon the execution of these events, the simulation undertook the replacement and reformulation of links. The performance of this experimental setup was then compared with that of a similarly constructed experiment that did not incorporate simulated churn and dropout. In both scenarios, the communication scope was restricted to a maximum of one hop.

Despite a modest decline in the average final testing accuracy from 87.7% to 87.5%, the network's dynamism mitigated the variability in the training process that arose from reducing the communication scope. Regarding edge system considerations, it is imperative to acknowledge that in scenarios where churn is dynamically occurring, an agent that remains within range long enough to update both its own and its neighbors' tables before its DSL PSO update step will be perceived as being continuously present during that time frame.

CHAPTER 6
Summary and Discussion

6.1 Mechanical Relations to Related Works

This section delineates the precise relationships that PC-DSL shares with established algorithms. Initially, we illustrate the reduction of PC-DSL to Federated Learning (FL) and Particle Swarm Optimization (PSO). This demonstration serves to highlight the importance of the hyper-parameter weight constants and elucidate the connections between PC-DSL and other decentralized implementations, such as Gossip Learning and Hierarchical Federated Learning. In order to achieve equivalence with Federated Learning, DSL can be reduced by omitting the initial term for $w_{i,t}$, specifying $c_0 = v$, allowing $c_1 = 0$ and $c_2 = 1$, and equating the number of contributors, denoted S in 2.4, to the number of workers w. The simplification of PSO is more straightforward and can be achieved by adjusting the learning rate $\alpha = 0$. Likewise, to relate PC-DSL back to the decentralized versions of FL discussed in chapter 2, we may look at similar reductions. Moreover, as demonstrated in 5, by maximizing the constants to their extreme values with the sole active term being the c_2 constant, convergence can be compelled even within a restricted communication context. This scenario is feasible only in the presence of a global dataset. The least assured number of rounds required for synchronization corresponds to the greatest distance in communication steps between any two nodes. This observation serves as a reminder that the framework is adaptable, with privacy considerations being contingent upon specific configurations and implementations.

6.2 Future Work

A primary benefit of distributed learning, in addition to the communication cost required for data transmission, is additional data privacy. Although not directly sharing data, information can still be gleaned on the basis of the shared model. While PC-DSL limits the scope of communication for agents' parameter weight vectors, it increases privacy; however, all neighbors currently receive full copies of an agent's best model weights. Security is essential to the performance and reliability of any model produced. As such, we recommend extensive testing into the resistance of PC-DSL to various security targeting attacks.

The subsequent domain for future research concerns the algorithm's effectiveness when employed with non-independently and identically distributed (non-i.i.d.) data across diverse configurations. This effectiveness constitutes a core attribute of the foundational DSL algorithm. However, several principles inherent in PC-DSL, such as the limited scope and the absence of a global dataset, introduce additional complexities, behaviors, and opportunities for learning with data characterized by pronounced non-i.i.d. features. PC-DSL depends on the accurate evaluation of prospective global best weight vectors against a representative sample of equitable comparison data. This critical requirement constrains the ability to localize responsibility and communication in a non-i.i.d. context. Consequently, the presence of either a global dataset or alternative evaluation methods, as found in related literature, is essential. Further research in this area is crucial for PC-DSL to be a viable solution in edge distributed systems.

An intriguing potential direction for future research involves a thorough investigation into the optimization of exploration and exploitation mechanisms enabled by the frameworks of Distributed Swarm Learning (DSL). In addition, it necessitates an analysis of how these mechanisms are affected by the constraints imposed by single-hop peer-to-peer communications, which restrict the extent of algorithmic interaction. Current experimentation has predominantly focused on static, linearly varying, or random transformations of variables, such as the DSL scaling constants. Nonetheless, it is plausible that more effective approaches for managing the traversal of the exploration-to-exploitation spectrum may exist, such as employing the comprehension of the differential in quality between the current, previously best, and globally best weight vectors to refine the weighting of the PSO step aggregation.

The final suggestion for future research is the empirical investigation of applied PC-DSL. The capacity of fully simulated evaluations of this technology to replicate real-world scenarios is inherently constrained. Moreover, the preliminary analysis focused on a generalized instance of PC-DSL, whereas in practical applications, the requirements and criteria are intrinsically more specialized for various tasks. Therefore, empirical exploration of DSL performance within a particular application context would yield further substantial contributions to the scholarly discourse on the algorithm and associated subjects.

6.3 Summary

This manuscript presents an innovative decentralized distributed machine learning algorithm, developed by building upon DSL and integrating mechanisms from related approaches. The PC-DSL algorithm exhibits robustness against constraints associated with communication scope. Under asynchronous learning conditions, the PC-DSL algorithm demonstrates a notable advantage in learning speed over its synchronous counterpart, assuming parity in the average duration per learning step. Coupled with the reduced communication time required for synchronization and its ability to withstand limited scope, this asynchronicity further enhances the pace of training, adaptability to network variability, and communication efficacy.

REFERENCES

Blanchard, P., Mhamdi, E. M. E., Guerraoui, R., & Stainer, J. 2017, Byzantine-Tolerant Machine Learning

Chen, C., Xu, H., Wang, W., Li, B., Li, B., Chen, L., & Zhang, G. 2021, in 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS), 1–11

Fan, X., Wang, Y., Huo, Y., & Tian, Z. 2022, CB-DSL: Communication-efficient and Byzantine-robust Distributed Swarm Learning on Non-i.i.d. Data

Fan, X., Wang, Y., Huo, Y., & Tian, Z. 2023, in ICC 2023 - IEEE International Conference on Communications, 973–978

——. 2024, IEEE Transactions on Cognitive Communications and Networking, 10, 322

Gou, Y., Wang, R., Li, Z., Imran, M. A., & Zhang, L. 2022, in ICC 2022 - IEEE International Conference on Communications, 177–182

Göndör, S., Yildiz, H., Westerkamp, M., & Küpper, A. 2022, Blade: A Blockchain-supported Architecture for Decentralized Services

Hegedundefineds, I., Danner, G., & Jelasity, M. 2019, in Distributed Applications and Interoperable Systems: 19th IFIP WG 6.1 International Conference, DAIS 2019, Held as Part of the 14th International Federated Conference on Distributed Computing Techniques, DisCoTec 2019, Kongens Lyngby, Denmark, June 17–21, 2019, Proceedings (Berlin, Heidelberg: Springer-Verlag), 74–90

Hunter, J. D. 2007, Computing in Science & Engineering, 9, 90

Jin, X., Katsis, C., Sang, F., Sun, J., Kundu, A., & Kompella, R. 2022, Edge Security: Challenges and Issues

Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., & Bacon, D. 2017, Federated Learning: Strategies for Improving Communication Efficiency

Li, X., Huang, K., Yang, W., Wang, S., & Zhang, Z. 2020, On the Convergence of FedAvg on Non-IID Data

Mao, Y., Zhao, Z., Yan, G., Liu, Y., Lan, T., Song, L., & Ding, W. 2022, ACM Trans. Intell. Syst. Technol., 13

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. 2023, Communication-Efficient Learning of Deep Networks from Decentralized Data

Naik, D., Grace, P., Naik, N., Jenkins, P., Mishra, D., & Prajapat, S. 2023, in 2023 IEEE International Conference on ICT in Business Industry Government (ICTBIG), 1–8

Shayan, M., Fung, C., Yoon, C. J. M., & Beschastnikh, I. 2018, CoRR, abs/1811.09904

Wang, Y., Tian, Z., Fan, X., Cai, Z., Nowzari, C., & Zeng, K. 2024, IEEE Communications Magazine, 62, 160

Waskom, M. L. 2021, Journal of Open Source Software, 6, 3021

Wen, J., Zhang, Z., Lan, Y., Cui, Z., Cai, J., & Zhang, W. 2023, International Journal of Machine Learning and Cybernetics, 14, 513

Xu, Z., Jiang, F., Niu, L., Jia, J., & Poovendran, R. 2024, Brave: Byzantine-Resilient and Privacy-Preserving Peer-to-Peer Federated Learning

Yuan, L., He, Q., Tan, S., Li, B., Yu, J., Chen, F., Jin, H., & Yang, Y. 2021, in Proceedings of the Web Conference 2021, WWW ’21 (New York, NY, USA: Association for Computing Machinery), 2245–2257