FLAGS Framework and Decentralized Federated Learning under Device Volatility

by

Ahnaf Hannan Lodhi

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Doctor of Philosophy

in

Computer Science and Engineering

KOÇ ÜNİVERSİTESİ

5 September, 2023

FLAGS Framework and Decentralized Federated Learning under Device Volatility

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a doctoral dissertation by

Ahnaf Hannan Lodhi

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____
Prof. Öznur Özkasap (Advisor)

_____
Asst. Prof. Barış Akgün (Co-Advisor)

_____
Prof. Deniz Yuret

_____
Prof. Berk Canberk

_____
Prof. Sinem Çöleri

_____
Assoc. Prof. İlker Demirkol

Date: _____

Dedicated to the people of Pakistan!

May their hopes overcome the odds stacked against them.

ABSTRACT

FLAGS Framework and Decentralized Federated Learning under Device Volatility

Ahnaf Hannan Lodhi

Doctor of Philosophy in Computer Science and Engineering

5 September, 2023

Federated Learning (FL) has become a key choice for distributed machine learning. Initially focused on centralized aggregation, recent works in FL have emphasized greater decentralization supported by standardization of serverless interaction in the next-generation communication networks. However, the diversity of devices, data distributions, and communication settings, compounded by dynamic operating conditions, result in multiple challenges for Decentralized FL (DFL). There have been various approaches to DFL, from utilizing intermediate edge servers to fully device-to-device approaches. In decentralized settings, communication cost and learning performance are usually assessed together and certain trade-offs are made based on scenarios. However, there is a lack of existing work on comparing DFL approaches in an apples-to-apples manner in a multitude of scenarios and operating conditions. To bridge this gap between methods and their comparative analysis, we design and develop the **F**ederated **L**earning **A**lgorithms **S**imulation (FLAGS) Framework. One important challenge we noticed that most DFL methods struggle with is the extreme fluctuations in device availability, especially for purely decentralized approaches. This **device volatility** leads to poor learning performance. To address this issue, we investigate the effects of neighborhood selection, memory and multi-hop information passing on DFL performance. We introduce a fully decentralized FL approach that can operate under realistic and highly volatile device participation settings.

The key contributions of this thesis are as follows: (i) development of a lightweight FL framework for benchmarking a large plethora of methods, (ii) analysis and comparison of multiple FL methods with this framework under multiple operating con-

ditions, (iii) empirical analysis of various node selection strategies under heavy device volatility, and (iv) utilizing memory and relayed communication to enhance device-to-device FL by developing a novel algorithm that can operate under realistic operating conditions and heavy device volatility.

Federated Learning supports a wide variety of node interactions and autonomous operations across the network edge. With the aim to encompass this multi-faceted heterogeneity, the FLAGS framework was proposed and developed as a lightweight FL implementation and testing platform. FLAGS framework allows for a wide range of device behaviors and cooperation mechanisms, enabling rapid testing of multiple FL algorithms. FLAGS's built-in features allow it to subject existing and novel FL algorithms to a wide range of data distributions, simulating the nodes with multiple neural networks as well as participation conditions ranging from homogeneous to highly volatile.

Different network tiers and communication mechanisms enable various FL algorithms to be configured by employing various combinations of the aforementioned factors. In order to consolidate this very extensive FL landscape and offer an objective analysis of the major FL algorithms, comprehensive cross-evaluations for a wide range of operating conditions have also been conducted. Starting with the three foundational FL algorithms, including Hierarchical FL (HFL), Decentralized FL (DFL), and Gossip FL (GFL), this work evaluates six derived algorithms ranging from fully centralized to fully decentralized. The experiments indicate that fully decentralized FL algorithms achieve comparable accuracy under multiple operating conditions, including asynchronous aggregation and the presence of stragglers. Furthermore, DFL can also operate in noisy environments and with a comparably higher local update rate. However, the impact of extremely skewed data distributions on DFL is much more adverse than on centralized variants.

The analysis of the cross-evaluation indicates that DFL performance is considerably impacted by node participation. This part of the thesis focuses on improving DFL performance under realistic and volatile device behavior. Node selection in various forms has been experimented with to improve both communication efficiency and convergence rate. We experimented with multiple node selection mechanisms and

also proposed and evaluated a time-varying parameterized node selection method for DFL employing validation accuracy and its per-round change. The mentioned criteria are evaluated using both hard and stochastic/soft selection on sparse networks. The results indicate that the bias associated with node selection adversely impacts performance as training progresses, and a uniform random selection is preferable under extremely limited participation conditions.

Continuing with volatile conditions, we investigate and propose mechanisms to improve DFL operating on sparse graphs in the presence of stragglers and non-participating nodes. We first propose two algorithms: Memory-Assisted DFL (MA_DFL) and Augmented-Graph Assisted DFL (AG_DFL). These algorithms employ memory and selective relaying to improve DFL performance. Both algorithms outperform the baseline DFL and gossip interaction for volatile node participation. Then, we propose a hybrid of these two algorithms, Memory and Augmented-Graph Assisted DFL (MAG_DFL), that employs memory and graph augmentation to improve the performance of DFL under highly volatile devices and extreme data conditions.

The research conducted in this thesis evaluates the multi-faceted challenges to the DFL operation in volatile conditions and proposes mechanisms to improve its performance. Our work indicates that DFL holds the potential to assist learning operations distributed across the edge network. It may be used to augment the FL in the presence of costly upstream communication or limited connectivity. However, node density has a major impact on DFL, and sparse networks, along with volatile device behavior and non-IID distributions, tend to reduce its convergence rate. The enhanced neighborhood interaction and intelligent use of local information has the potential to improve DFL performance under such adverse conditions based on the presented results. The analysis, algorithms and results presented in this thesis pave the way for additional developments and more practical applications of DFL in the next-generation communication networks.

ÖZETÇE

FLAGS Platformu ve Cihaz Dalgalanması Durumunda Merkeziyetsiz Federe Öğrenme
Ahnaf Hannan Lodhi
Bilgisayar Bilimi ve Mühendisliği, Doktora
5 Eylül, 2023

Federe Öğrenme (FL), dağıtık makine öğrenimi için önemli bir seçenek haline gelmiştir. Başlangıçta merkezi birleştirme üzerine odaklanan FL'deki son çalışmalar, yeni nesil iletişim ağlarında sunucusuz etkileşimin standartlaştırılmasını destekleyen merkezi olmayan yaklaşımlara vurgu yapmıştır. Ancak cihazların çeşitliliği, veri dağılımları ve iletişim ayarları, dinamik işletme koşulları tarafından karmaşık hale getirilmiş ve bu durum Merkeziyetsiz Federe Öğrenmenin (DFL) birden fazla zorlukla karşılaşmasına neden olmuştur. DFL konusunda, sınır sunucularını kullanmaktan tamamen cihazdan cihaza yaklaşımlara kadar farklı yöntemler bulunmaktadır. Merkezi olmayan durumda iletişim maliyeti ve öğrenme performansı genellikle bir arada değerlendirilir ve senaryolara dayalı belirli bir denge kurulur. Ancak, DFL yaklaşımlarını çok sayıda senaryo ve çalışma koşulu altında eşit durumda karşılaştırmak için literatürde eksiklik bulunmaktadır. Bu yöntemler ile karşılaştırılabilir analiz arasındaki boşluğu kapatmak amacıyla, Federe Öğrenme Algoritmaları Simülasyon (FLAGS) Platformunu tasarlayıp geliştirdik. Birçok DFL yönteminin özellikle tamamen merkeziyetsiz yaklaşımlar için cihaz erişilebilirliğindeki aşırı dalgalanmalar önemli bir zorluk olarak öne çıkmaktadır. Bu cihaz dalgalanması, zayıf öğrenme başarımına yol açar. Bu sorun kapsamında, çevre seçiminin, belleğin ve çoklu atlama bilgisi iletimin DFL başarımı üzerindeki etkilerini araştırdık. Gerçekçi ve yüksek derecede dalgalı cihaz katılım koşulları altında çalışabilen tamamen merkeziyetsiz bir FL yaklaşımı sunuyoruz.

Bu tezin temel katkıları şu şekilde özetlenebilir: (i) birçok yöntemi karşılaştırmak için hafif bir FL simülasyon platformu geliştirilmesi, (ii) bu platformda bir dizi işletme koşulunda birçok FL yönteminin analizi ve karşılaştırılması, (iii) yoğun ci-

haz dalgalanması altında çeşitli düğüm seçimi stratejilerinin deneysel analizi ve (iv) bellek ve iletimli iletişimi kullanarak FL başarımını geliştirmek için gerçekçi işletme koşulları ve yoğun cihaz dalgalanması altında çalışabilen yeni bir algoritma geliştirilmesi.

Federe Öğrenme, ağ kenarında geniş bir düğüm etkileşimi ve otonom işlemleri destekler. Bu çok yönlü heterojenliği kapsamak amacıyla FLAGS simülatörü, hafif bir FL uygulama ve test platformu olarak geliştirilmiştir. FLAGS platformu, geniş kapsamlı cihaz davranışları ve işbirliği mekanizmaları için olanak tanır ve böylelikle çeşitli FL algoritmalarının hızlı bir şekilde test edilmesine olanak verir. FLAGS özellikleri, mevcut ve yeni FL algoritmalarının geniş bir veri dağılımında test edilmesine olanak tanır, düğümleri birden çok sinir ağıyla ve homojen katılımdan yüksek derecede dalgalı katılıma kadar çeşitli cihaz katılım koşullarında simüle edebilir.

Farklı ağ katmanları ve iletişim mekanizmaları, çeşitli FL algoritmalarının yukarıda bahsedilen faktörlerin birleşimlerini kullanarak yapılandırılmasına olanak tanır. Bu çok kapsamlı FL yaklaşımlarını bir araya getirmek ve temel FL algoritmalarının nesnel bir analizini sunmak amacıyla bir dizi kapsamlı analiz gerçekleştirilmiştir. Hiyerarşik FL (HFL), Merkeziyetsiz FL (DFL) ve Salgın FL (GFL) gibi üç temel FL algoritması ile başlayarak, bu çalışma tamamen merkezi olanlardan tamamen merkezi olmayanlara kadar uzanan altı algoritmanın analizi yapılmışır. Deneyler, özellikle asenkron birleştirme ve geciken düğümlerin varlığı gibi bir dizi çalışma koşulu altında tamamen merkeziyetsiz FL algoritmalarının karşılaştırılabilir doğruluk elde ettiğini göstermektedir. Ayrıca, DFL gürültülü ortamlarda da çalışabilir ve daha yüksek yerel güncelleme oranına sahip olabilir. Bununla birlikte, aşırı eğri veri dağılımlarının DFL üzerindeki etkisi, merkezi yaklaşımlara kıyasla daha olumsuzdur.

Çapraz değerlendirmenin analizi, DFL başarımının düğüm katılımından oldukça etkilendiğini göstermektedir. Tezin bu bölümü, gerçekçi ve dalgalı cihaz davranışı altında DFL başarımını iyileştirmeye odaklanmaktadır. İletişim verimliliğini ve yakınsama hızını iyileştirmek için çeşitli düğüm seçimi mekanizmalarıyla deneyler yapılmıştır. Farklı düğüm seçim mekanizmaları ile deneyler gerçekleştirilerek, DFL için doğruluğu ve bunun tur başına değişimini kullanan zamanla değişen parametreli bir düğüm seçim yöntemi önerilmiştir. Bu ölçütler, seyrek ağlarda hem sert

hem de stokastik/yumuşak seçim kullanılarak değerlendirilmiştir. Sonuçlar, düğüm seçimi ile ilişkilendirilen önyargının, eğitim ilerledikçe başarımı olumsuz etkilediğini ve aşırı sınırlı katılım koşulları altında rastgele bir seçimin tercih edilebilir olduğunu göstermektedir.

Geciken düğümler ve katılmayan düğümlerin bulunduğu seyrek çizgeler üzerinde çalışan DFL başarımını iyileştirmek için mekanizmalar geliştirdik. Öncelikle, Bellek Destekli DFL (MA_DFL) ve Artırılmış Çizge Destekli DFL (AG_DFL) olmak üzere iki algoritma önerdik. Bu algoritmalar belleği ve seçici iletimi kullanarak DFL başarımını iyileştirmektedir. Her iki algoritma da dalgalı düğüm katılımı için temel DFL ve salgın etkileşimi yaklaşımlarından daha iyi başarım göstermektedir. Ardından, bu iki algoritmanın bir melezini önerdik: Bellek ve Artırılmış Çizge Destekli DFL (MAG_DFL), yüksek derecede dalgalı cihazlar ve aşırı veri koşulları altındaki DFL başarımını iyileştirmek için belleği ve çizge artırımını kullanan bir algoritmadır.

Bu tezde yürütülen araştırmalar, dalgalı koşullarda DFL işleyişine yönelik çok yönlü zorlukları değerlendirir ve başarımını iyileştirmek için mekanizmalar önerir. Çalışmamız, DFL'nin sınır ağı boyunca dağıtılan öğrenme işlemlerine yardımcı olma potansiyeline işaret etmektedir. Maliyetli yukarı yönlü iletişim veya sınırlı bağlantı durumlarında FL'yi iyileştirmel için kullanılabilir. Ancak, düğüm yoğunluğu DFL üzerinde önemli bir etkiye sahiptir ve seyrek ağlar, dalgalı cihaz davranışı ve homojen olmayan veri dağılımları, yakınsama hızını azaltma eğilimindedir. Geliştirilmiş komşuluk etkileşimi ve yerel bilginin akıllıca kullanımı, böyle olumsuz koşullar altında DFL başarımını artırma potansiyeline sahiptir. Bu tezde sunulan analizler, algoritmalar ve sonuçlar, DFL'nin gelecek nesil iletişim ağlarında daha fazla gelişmenin ve pratik uygulamanın yolunu açmaktadır.

ACKNOWLEDGMENTS

Before everything else, I bow my head in gratitude to the ALMIGHTY for having granted me this opportunity and bestowing success upon me in this endeavor.

I wish to express my profound gratitude to my advisors Prof. Öznur Özkasap and Asst. Prof. Barış Akgün for their invaluable guidance and immense support throughout this journey. Their patience and understanding have been essential in taking this journey to fruition. I am also grateful to the entire committee, Prof. Deniz Yuret, Prof. Berk Canberk, Prof. Sinem Çöleri and Assoc. Prof. İlker Demirkol for their consideration and support in evaluating and finalizing this thesis.

I would also like to extend my thanks to the administrative team, particularly Ms. Bahar Hısım and Ms. Emine Büyükdurmuş, at the Graduate School of Science and Engineering (GSSE) and the KUIS AI Center for their untiring support during my time here. I am also indebted to the support extended by Koç University, the KUIS AI Center and TÜBITAK (2247-A Project 121C338) for their extensive support throughout this time. Also, all the individuals who have made life easier for me and countless other students like me, yet whose names I might have missed or never got the courage to ask, I would like to thank them for their efforts.

I would like to express my special gratitude to my parents. Theirs has been the most profound impact on my life. They have been the inspiration that motivated me to undertake the rigors of PhD and their words have pushed me through some trying times during this time. The encouragement extended by my brothers, Dr. Aemen Lodhi and Arqam Lodhi, and their constant belief in my abilities has been

invaluable. A special word of thanks for Vaqas, Usman, Taimoor and Waqas for their badgering and bickering which lightened up the mood and re-energized me to continue under dreary phases.

Finally, Maryam, Dawood and Haytham, I consider my equal partners in this entire process. They have gone through all the stages alongside me, sweated with me in times of stress, rejoiced with me in moments of success and waited patiently till that magical moment of the declaration of success. They renewed my sense of purpose at every turn and fueled my ambition to see things through. To them, I owe a world of thanks and more for remaining steadfast with me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

Chapter 1

INTRODUCTION

The last decade has witnessed a remarkable transcendence of data, interconnectedness and information-centric services in all spheres of the global domain. An unprecedented growth in data-generating devices and sensors coupled with the availability of powerful computational resources has further led to the ascendancy of machine learning. It has been estimated that close to 29 billion devices will be connected to the Internet by 2023 [1] with the data traffic expected to reach 131 Exabytes (EB) by the end of 2024 [2]. Furthermore, the requirements for 6G aiming for data rates of approximately 1Tbps per user [3] have further reinforced a growing realization that the traditional centralized/cloud systems would find it increasingly difficult to manage the accompanying computation requirements, particularly in a lifelong learning scenario.

The true potential of Artificial Intelligence (AI) is unlocked in a connected / networked domain where intelligent services can be extended simultaneously to a large scale of users. For instance, temperature assessment using thermal imaging was deployed as the first line of detection of the novel Coronavirus–caused respiratory disease (COVID-19) [4] at major transit points. However, deployment of isolated or centralized processing clusters over a longer duration proved inefficient due to resource wastage and prohibitive costs, with the major disadvantage being the distributed nature of data itself. Distributed learning on the other hand, allows devices to not only collaborate resources but also the learned parameters across devices, enabling more effective information diffusion. Collaborative healthcare, smart cities and management, energy systems, disaster coordination and response, cyber security, and threat identification are some common applications of distributed learning.

Developing a framework that supports distributed operation and access to remote data is an essential requirement for rapidly transforming applications.

The paradigm shift in the nature of networked services and the transformation of the connected devices coupled with the distributed nature of data requires a decentralized approach for extracting maximum learning benefits. To avoid overwhelming the network and data servers, the computational load must be moved at or closer to the network edge. Edge Computing [5] offers a potentially powerful solution to this problem. The edge computing framework aims to leverage distributed computing concepts to alleviate the computational load from the network core, benefiting from processing power available close to the network edge. The computation power of the elements closer to the edge network offers a powerful alternative to centralized computing, albeit in a distributed manner. Successful exploitation has the potential to shift the elements of cloud services close to the data sources, ensuring better data security and reduced load on the network backbone.

Edge Computing

Edge Computing is the emerging domain being developed to address the limitations of cloud computing. This paradigm aims at shifting the computation load towards the outer edges of the network utilizing the devices at the data origin and their geographical proximity. The motivation behind the development of this domain is two-fold: It aims to reduce the computational and communication load from the network core while enabling dynamic resource allocation for applications in cyber-physical systems such as industrial IoT, smart buildings and grids, autonomous transportation, and remote healthcare [6].

Conventional networks can be characterized by reduced computational power in the elements at the fringes of the network. However, the addition of devices at the outer levels has also significantly outpaced the increase of processing power at the network core [1]. Edge Computing has emerged to support this dramatic increase in resource requirements by leveraging the untapped potential away from the enterprise data centers. Processing power is obtained by a collaborative operation between

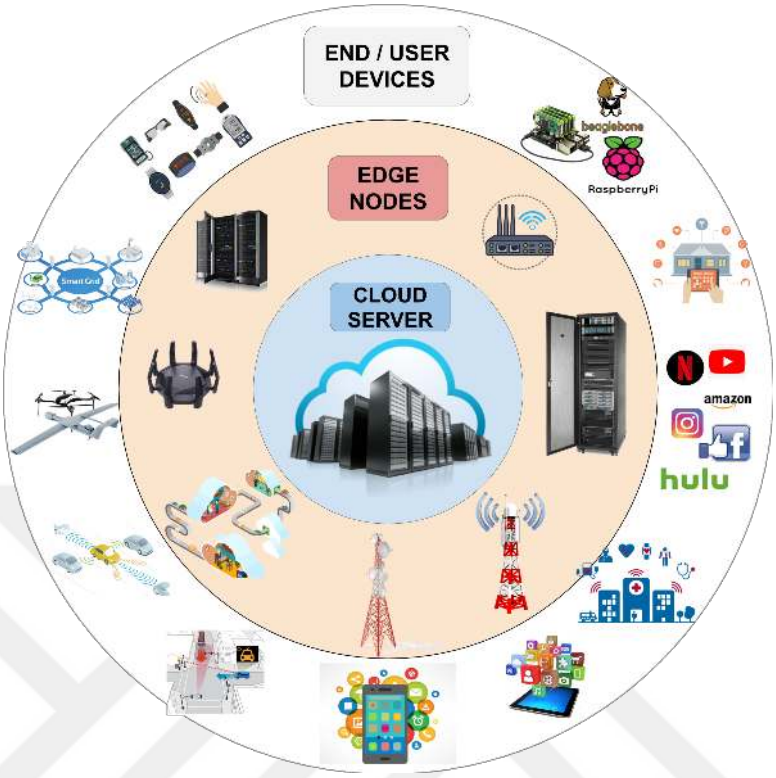


Figure 1.1: Levels of Network Hierarchy: *a) End / User device:* Devices/sensors at the origin of data, *b) Edge Nodes:* Intermediate network devices connecting end devices to the network core, *c) Cloud Servers:* Hub of computation and decision-making in existing networks. The two outer levels are jointly referred to as *"Edge Levels"* and are the focus of Edge Computing.

various entities at the network edge, including the user devices, mobile-based stations and gateways and access points.

Network Hierarchy

To formalize various elements, a conventional network can be categorized into three hierarchical levels for Edge Computing as depicted in Fig-1.1:

1. **End or user devices** are the elements responsible for data generation, including mobile and smart devices, IoT sensors, smart applications, autonomous cars and drones etc. These devices, lying at the outermost level, are generally characterized by the lowest individual computing power.

2. **Edge server or nodes** comprise of mobile base stations, gateways, access points (APs), micro datacenters, etc. established in proximity to the data sources having relatively comparatively higher computational and storage capacity. *Combined with the end devices, these two levels jointly form the 'Edge Level'.*

3. **Cloud or data server**, the innermost level in the network hierarchy, possesses the maximal computation and storage ability, currently supporting most of off-device computation. However, the remoteness of the Cloud level incurs a considerable communication overhead and potential privacy and latency issues.

1.1 Distributed Learning

With the proliferation of Edge Computing and the ubiquity of smart devices, the focus on Distributed Learning has rapidly accelerated. Distributed learning collaboratively leverages the computational resources and data available at geographically disparate locations to train intelligent agents. While distributed optimization has been widely researched since the 1980's, the area has once again garnered significant attention by its application in deploying Artificial Intelligence (AI) applications using the Edge Computing paradigm. Traditionally, Machine Learning and inference-generation operations were centralized at the cloud level. They required huge amounts of data to be orchestrated for learning a generalizable model. Additionally, a slew of supporting services are then needed to extend the benefits to the multitude of smart devices. However, the evolving network edge, increased device capabilities, and emerging communication networks have enabled the principles of distributed learning to maximally offload learning and inference computations to the 'Edge Level' (Fig-1.1). In general, distributed learning requires a highly multidisciplinary approach using knowledge and efficient practices from fields including but not limited to AI, Computer Architecture, Embedded and Distributed Systems, etc., to achieve desired performance levels.

Operational Constraints Adopting distributed computing brings forth its own set of challenges for optimal operation, including data sharing, security, latency, etc. Additionally, the typical learning scenarios have so far utilized centralized frameworks unifying both data and computing resources at one single entity in the form of servers or clusters. Distributed Learning, on the contrary, aims to exploit the significantly untapped potential of the billions of elements at the network edge. The current research to achieve this goal is primarily driven by the following factors [7], [8]:

1. ***Cost:*** Any form of decentralized computation results in major costs related to communication, energy, processing, and memory.

    - *Communication Costs:* Remote computations require data to be exchanged between various distributed elements, introducing not only the cost of data communication but the associated overhead costs in already congested networks. The services at the end devices, whether provided by mobile applications over cellular networks or IoT networks, are increasingly resorting to providing improved user experiences as well as information. The demand for immersive Quality of Experience (QoE) using Augmented Reality / Virtual Reality (AR/VR) alone is expected to result in an 8-fold increase in data traffic [9]. Furthermore, while emerging networks offer higher speeds, legacy networks would face increasingly difficult prospects for supporting such services. Incorporating distributed learning in such an environment would thus be associated with its communication costs.

    - *Energy:* A considerable majority of nodes at the edge level, whether the user devices or the edge nodes, often operate with a limited energy budget. Unlike a cloud server where energy constraints are considered for economical operation, mobile devices at the user end cease operation if they exceed their energy constraints. Thus, all aspects of the learning process, from training to inference, must respect these energy limitations.

- *Processing and Memory:* Deep Learning models require considerable processing and memory resources to extract and learn the deep representations of the data. In addition to a dearth of processing power at the edge level, managing sufficient resources for deep learning models to run in the presence of numerous competing services is a major challenge. In general, deeper networks more often outperform shallow networks, and thus higher performance requires larger storage requirements.

2. ***Latency:*** Time-critical applications require real-time or near-real-time inferences. For example, translation of conversations from one language to another, object segmentation in photos or analysis, fusion and logical inference of data carried out by the sensors of autonomous vehicles, especially self-driving cars, are some applications where the delay between the input and inference can result in seriously compromised performance. Additionally, offboarding data for remote computation imposed additional time costs due to communication to deeper levels of the network. The impact of proximity is investigated in [10] using the face detection task running on Amazon Web Services as a reference. The results indicate a 50% task completion rate between 200-600ms depending on the server location.

3. ***Scalability:*** With the proliferation of devices at the Edge level, sharing data for centralized and distributed computing becomes increasingly difficult due to communication and processing bottlenecks. Decentralized computing must be able to seamlessly cater to the billions of devices that contribute data or computing resources without degrading the device performance and congesting the network.

4. ***Privacy and Security:*** Malicious adversarial actions against intelligent systems can encompass both the data and the learning framework [11], [12], [13]. AI in a distributed setting exposes the learning architecture to a certain degree if the learning parameters are being shared. These could then be used

for adversarial actions against the learners leading to degraded performance and crippling critical automated systems. Furthermore, maintaining data integrity and prohibiting the exploitation of the associated metadata are challenges faced whenever data is shared over the network. This challenge and the privacy risk to the data make Distributed Learning operation [14] [15] more difficult.

5. ***Communication Reliability:*** Conventional Deep Neural Networks (DNNs) do not cater to reliability issues. However, in a distributed setting, reliability guarantees need to be established to obtain correct loss in addition to the fact that the state of the network affects the performance of the edge computing scenarios [16]. There exists an algorithmic challenge for implementing distributed learning in the presence of network (e.g. packet error and dropout, congestion) and client (e.g., offline, busy, adversarial environment) reliability issues both for training and inference processes. The emergence of 5G, Ultra-Reliable-Low Latency (URLLC) networks offers promising avenues [17]. However, the learning frameworks themselves need to be adapted to utilize the benefits of these modern communication technologies fully.

6. ***Inference Transparency:*** Critical learning applications for domains such as health, finance, and security, among others, require interpretability [18] "as an important element of engendering trust in the inference system". Interpretability, however, is yet to have a precise definition in terms of machine learning, though [19, 20] have presented a general framework for gauging the underlying dynamics of learned decision-making. An elaborate survey of the existing works is presented in [21] , categorizing them under Model transparency and Model functionality for Deep Learning. A distributed setting, however, offers a unique scenario where the model transparency not only causes additional constraints on the model and operating costs, it constitutes an entirely different challenge due to the environment and nature of distributed models themselves. The first reported case of the hazards of opaque learning systems and their

impact on real-life scenarios was recently reported as the wrongful arrest by Detroit Police due to a faulty match by the facial recognition system. It is imperative, thus, that transparency is integrated into the distributed learning process.

7. ***Device Diversity in Edge:*** The edge levels house many nodes with distinct storage and processing capabilities. The devices farther away from data origins are more capable both in terms of memory and computational power, with central servers being the most resourceful devices. Furthermore, the networks employ multiple communication links and protocols, which results in a highly diverse environment. These device and network characteristics exacerbate the difficulty of developing a collaborative structure for learning. Any distributed learning system must, therefore, possess inherent tolerance for this heterogeneity. Furthermore, the designed frameworks should consider the nature (e.g., memory, processing, energy, static or mobile, data availability) of the platforms they are intended for. Platforms such as mobiles and autonomous vehicles have relatively better processing power than IoT or Smart Home sensor networks. A DNN designed for IoT will not be optimal for a mobile setting and vice versa. It is, therefore, imperative that the learning systems consider the characteristics of their intended platforms as a tunable parameter.

1.2 Federated Learning

Machine Learning and its applications have increased multi-fold due to the abundance of both computing resources and data. The issues associated with data privacy, sharing, and orchestration have accelerated research into distributed learning algorithms, with the key goal being to move learning as close to data origin as possible. The distinction between various distributed learning techniques lies in how devices interact with each other to achieve consensus. Federated Learning (FL) [22,23] is one such learning framework that enables disjoint devices to learn collaboratively without sharing data. Since its inception by Google researchers, Federated Learning

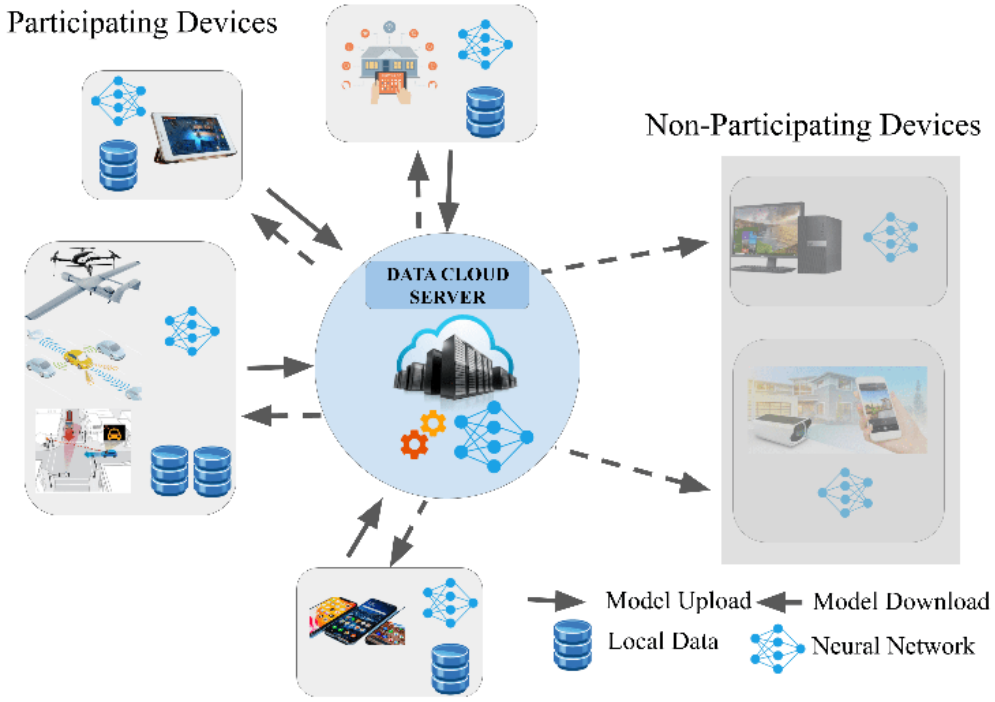


Figure 1.2: Centralized Federated Learning

(FL), proposed as FedAvg, has been one of the foremost avenues for distributed learning. Gboard-Google keyboard and Siri-Apple smart assistant are real-world applications that have benefited tremendously from FL [24]. Federated Learning (FL) has been at the forefront of enabling distributed learning for various applications while advocating stronger privacy across wider collaboration. FL allows nodes to update/train their models on their private data and share the updated models with a central server, which aggregates them to form a global model. The global model is then returned to the participating nodes to continue training (Fig-1.2). This allows FL to support participation at scale and devices to maintain data privacy by requiring them to share only their trained models. FL also benefits from the individual computing resources available at devices present throughout the Edge Network [25–27]. However, the collaboration in FL also brings forth challenges in the form of device and data heterogeneity, significant communication overhead, and server bottlenecks [24].

Federated Learning was initially conceived as a two-tiered learning framework alternating between clients/nodes performing the training and the cloud servers aggregating the local models to yield a global model. In order to increase the efficiency, the envisaged FL was further spread over the Edge Network to benefit not only from the Edge devices' capabilities but also address the challenges inherent in centralization. This evolution has yielded various FL algorithms suitable to a variety of scenarios, including vehicular networks, dense IoT networks, and cross-silo operations between relatively large data centers. These algorithms employ three different types of communication links associated with different communication costs: (a) Device-to-Device (D2D) (b) Device-to-Edge (D2E) (c) Edge-to-Cloud (E2C) (see Fig.1.3 for depicting these links).

D2D links are limited to the device level, whereas D2E links connect the nodes with the edge servers. Finally, E2C links connect the edge servers to the core network. While D2D links and D2E links operate independently, E2C links require the latter for information to be transmitted from the devices to the edge server for onward transmission to the core network. All subsequent communication analyses in this work use this fact to establish the effectiveness of a particular form of communication in FL operations.

Decentralized Federated Learning (DFL) utilizes D2D links to share the learned models between the nodes. Each node updates its models based on the local data, followed by sharing the same with other devices through D2D communication. The communication may be limited to 1-hop neighborhood or across multi-hop devices. The proposed methods under various settings show multiple advantages over Centralized FL (FedAvg). Furthermore, current and future devices come equipped with multiple communication links. This fact negates the limitation of operating a single FL algorithm at all times.

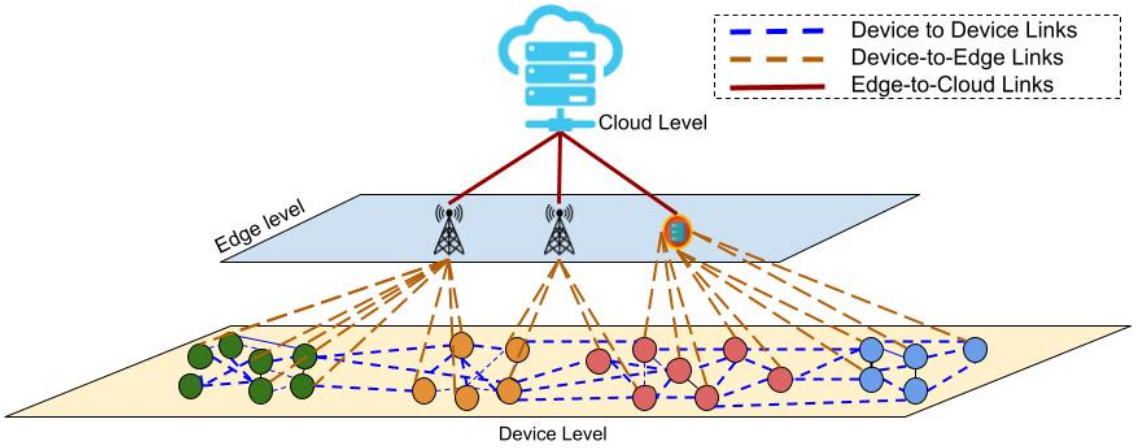


Figure 1.3: Network topology including Edge network levels and communication links.

1.3 Decentralized Federated Learning

One thrust of the potential solutions to the challenges faced by FL has been directed toward addressing the limitations posed by the central server. In [28], the authors propose a multi-center FL environment to maintain performance under conditions of non-**IID** (Independent Identically Distributed) data. On the contrary, to vanilla FL, the authors suggest aggregating local models at local cluster centers representing similar data distributions and models. Gossip-Federated Learning in [29] suggests another P2P aggregation formulation outperforming regular FL under conditions of IID data. Another blockchain-based solution for FL is proposed in [30] that relies on the availability of a few honest nodes working as a 'Consensus-Committee' to validate the shared updates. For autonomous vehicles, [31] devise a Blockchain-based FL scenario for on-vehicle Machine Learning (oVML). The proposed mechanism relies on a blockchain-based solution to address network connectivity and motivating devices with rewards proportional to their contributions. Finally, [32] presents a potential solution for a more practicable implementation of FL by devising a mechanism *'q-FFL'* to improve the accuracy of the worst performing nodes in a centralized FL setting.

Furthermore, emerging networks are fully primed to benefit from a more decentralized form of communication. Device-to-Device (D2D) communication [33] is a departure from the conventional forms of centrally coordinated communication frameworks. D2D setups allow nodes to communicate with their local neighbors, resulting in the creation of a massive decentralized ad-hoc mesh network while also allowing for the ascertainment of global network parameters [34]. Given the benefits such truly distributed frameworks may offer, a more recent take on developing FL-based solutions envisions D2D scenarios for learning. The participants in such networks operate *autonomously* with *local* information.

Decentralized Federated Learning (DFL) attempts to allay the communication and server-related challenges by allowing the devices to interact directly within their neighborhoods without external/server coordination [35]. Greater emphasis has been laid on serverless interactions in emerging and next-generation communication networks such as 5G and 6G [36]. This allows for many potential learning applications in a fully decentralized manner. DFL, operating on arbitrary network topologies, allows nodes to collaborate within their neighborhoods to reach a consensus model using Device-to-Device (D2D) interactions [33]. Enabling direct communication among devices restricts upstream communication, making learning more communication efficient. Additionally, it removes the server-related constraints both in terms of orchestration and a single point of failure. Furthermore, since the model exchange is limited to the node neighborhood, DFL promises greater FL scalability by enabling a much greater proportion of devices to undertake the learning process independently.

In Decentralized Federated Learning (DFL), each node shares performs a local stochastic gradient update on its model using its private data. It then shares its updated model within its 1-hop neighborhood through D2D interaction. The shared updates are used in weighted aggregated locally and successively propagated along

the network [37]. The information diffusion across the network is slower as the shared updates are down-weighted during each communication round as the models propagate through the network. While fully and densely connected topologies improve convergence, real-world configurations follow sparse configurations e.g. due to the presence of stragglers, implying reduced neighborhood sizes and reduced convergence rates [38]. The autonomous operation of nodes also impacts the convergence rate negatively since they may choose to skip training/communication rounds altogether. A significant proportion of nodes depicting such uncertain behaviors leads to a volatile participation environment [39], resulting in an unstable availability of neighboring nodes for model exchange. As explained in the upcoming section, the main focus of this thesis remains first the characterization of DFL under extreme conditions and subsequently to propose DFL algorithms to improve its convergence.

1.4 Contributions

Fully decentralized interactions derive their significance from minimizing their reliance on central orchestration. DFL also benefits from these aspects and enables learning without a global aggregation mechanism. It affords the benefits of lowering upstream communication costs, reducing reliance on central coordination as well as the ability to cope with the induced absence of global servers. While emerging networks like 5G and 6G introduce features to enable serverless communication, scenarios such as emergency situations, unplanned communication/power outages as well as disaster situations may render edge and global servers with significantly reduced capacity to handle communication and in certain cases, may even cause them to be completely unreachable [40–42]. Such cases may also be extended to unmanned aerial vehicle (UAV) setups where medium to high altitude systems may face restrictive communication scenarios and have to rely on device-to-device form of communication [43, 44]. Additionally, the node usage and characteristics themselves govern the participation preference of node in a given federation round. Such scenarios may arise due to performance and application requirements and mobility considerations [45]. The nodes in a vehicular network may receive little incidence

time with a given edge server or may travel farther away from edge servers. Similarly, disaster affected areas may be faced with a significant decrease in the participation by the local nodes. One of the main contributions of this work remains conducting the analysis and proposing solutions for DFL under volatile conditions. The majority of works on DFL assume 100% participation under decentralized scenarios thereby allowing a significant exchange of information between the nodes. However, taking realistic node behavior into account, it becomes important to consider probabilistic behaviors depending upon heterogeneous environmental and device characteristics.

The main contribution of this thesis is the assessments of DFL in extremely heterogeneous conditions and proposing mechanisms to improve its operation in volatile environments.

Design and Development of Federated Learning Algorithms Simulation (FLAGS) Framework

Benchmarking FL takes on greater significance as the envisioned operating environment is highly diverse and different from centralized or silo-based distributed learning. To achieve nuanced evaluation and ensure *extensibility*, this work designs and develops the Federated Learning Algorithm Simulation **FLAGS** framework. It allows each network entity, whether a node or a server, to emulate a realistic behavior separately. By implementing a range of functionalities associated with each entity, this framework allows diverse network interactions, enabling multiple FL algorithms to be simulated easily. Each device has various associated functions to enable multiple operating conditions and device behaviors. During the experiments, the FL algorithms are subjected to some of the key challenges facing general FL research. The empirical analysis of the performance of these FL algorithms is conducted by varying the operating parameters, such as device participation, communication noise, and data distribution.

FLAGS is a lightweight FL prototyping framework allowing for an accurate assessment of multiple FL algorithms for highly realistic network topologies. Inherent

in FLAGS are different levels of device participation (controlled by parameters p_k for participation in central aggregation and d_k for controlling neighborhood aggregation), device selection mechanism for nodes and servers, as well as generating multiple data distributions including IID, non-IID, and extremely-skewed non-IID data partitions. It also supports synchronous and deadline-based asynchronous operation to replicate heterogeneous device behavior [46]. where node training progresses at different paces. The highlight of FLAGS lies in its capability to configure and subject multiple FL algorithms to a realistic multi-tiered environment, which allows users to propose the best environment-FL fit. The framework's modular architecture affords ease of feature addition and expansion across all major building blocks. The details of the FLAGS framework have been provided in Chapter-4, including the architecture, capabilities, and run-time settings.

Comparative Assessment of Federated Learning Algorithms

Federated Learning is envisioned to operate in a heterogeneous environment with no prior assumptions. The challenge then arises regarding the performance of various FL algorithms and their feasibility under various conditions. Our work in [47] aims to contrast the performance of the major FL algorithms under some of the most prevalent challenges to distributed learning at the network edge, i.e., non-IID data, noisy communication, volatile nodes, and asynchronous aggregation. The main goal is to pave the way for a multi-FL system where devices may select the most suitable FL algorithm depending on the operating characteristics. The comparative characterization of FL algorithms conducted in this work is aimed at bridging this gap. The analysis conducted here paves the way for establishing the economy of operation for these algorithms for various operating scenarios in a particularly heterogeneous Network Edge.

1. A detailed comparative evaluation of the fundamental Federated Learning algorithms, namely (a) *Hierarchical FL (HFL)* (b) *Device-to-Device FL (D2DFL)* (c) *Gossip FL (GFL)* (d) *Centralized FL (FedAvg)* has been conducted. To expand the scope of the evaluation, this work also considers other FL algorithms

employing multiple aggregation strategies jointly. These include: (a) *Hierarchical Device-to-Device FL (HD2DFL)* (b) *Hierarchical Gossip FL (HGFL)* (c) *Clustered Device-to-Device FL (CD2DFL)* (d) *Inter-Cluster FL (iCFL)* (e) *Inter-Cluster Device-to-Device FL (iCD2DFL)*

Their performance is tested under ideal and noisy Device-Device (D2D), Device-Edge (D2E), and Device/Edge-Cloud (D2C/E2C) communication links as well as non-IID data distributions and various device participation levels. The results indicate that even in the presence of non-IID data, decentralized FL performs with a marginal loss in performance compared to centralized variants despite operating entirely in the absence of any global information. Extremely skewed data distributions, however, greatly impact decentralized FL, drastically reducing convergence rate and accuracy.

2. We also conduct a detailed study of these algorithms under various levels of device participation (such as active, inactive, and stragglers) has been conducted. Decentralized FL shows a marginal loss in performance when compared with centralized FL in the presence of degraded participation. A decrease in device participation and extremely skewed data distributions have a confounding effect on all the algorithms, more so on the decentralized ones.

3. An analysis using modified Few-Shot Learning, with considerably more local updates before aggregation, has also been conducted for the FL algorithms under consideration. Decentralized algorithms show a greater loss in accuracy than centralized ones. However, clustered operation helps mitigate the absence of a global server while restricting all communications to the device level.

The details of considered algorithms, network topology, and results have been discussed in detail in Ch-5

Impact of Node Selection on Decentralized Federated Learning in Volatile Settings

Evidence suggests that the convergence rate for DFL is more adversely impacted due to heterogeneous conditions. One avenue to improve convergence while operating under non-uniform conditions has been node selection. Node selection relies on identifying and using only beneficial nodes in the aggregation process. The major focus of client selection has remained in conventional FL settings. Therefore, in [48], we evaluate the implications of node selection for DFL under more challenging operating conditions. The summary of the same work presented in Ch-6 is as follows:

1. Our work evaluates performance-based node selection for DFL under sparse conditions using hard (top-k selection) and soft (probabilistic assignment) filtering. Results indicate that performance metrics may be insufficient to capture the overall model diversity in fully decentralized scenarios

2. A time-varying parameterized node selection method is proposed based on validation accuracy and its corresponding change from preceding rounds. The proposed criterion outperforms hard selection under extreme operating conditions.

3. The results indicate that operating fully locally, random stochastic selection has a better chance of capturing model diversity, resulting in better performance under spare and extreme data conditions.

Assisted Decentralized Federated Learning

One reason for slow information diffusion in DFL, particularly across sparse networks, is that the shared updates are gradually down-weighted during each communication round, especially in real-world sparse network configurations where neighborhood sizes are smaller. Additionally, the autonomous behavior of nodes negatively impacts convergence rates since some nodes may choose to skip training or communication rounds. When a significant number of nodes demonstrate such

uncertain behaviors, it creates a volatile participation environment, leading to an unstable availability of neighboring nodes for model exchange. Both graph sparsity and participation volatility significantly affect the convergence rate in DFL. Furthermore, they exacerbate the impact of data heterogeneity, causing the models to become increasingly biased towards the data distributions of the proportion of available neighboring nodes in a sparse neighborhood.

For such a scenario, we utilize local information to accelerate the emergent behavior and propose three algorithms and provide the details in Ch-7

1. We outline an asynchronous DFL setting for sparse network topology under volatile participation conditions. The presented setting works with a fraction of the nodes participating in the DFL aggregation phase while hosting highly skewed and non-IID data distributions.

2. We propose a Memory-Assisted Decentralized Federated Learning (MA_DFL) algorithm and assess its performance in highly volatile scenarios. Each node retains only the models received in the previous round and uses them as respective approximations of the non-participating nodes. The results indicate that employing memory with asynchronous training and low participation scenarios improves the performance and convergence of DFL.

3. We propose an Augmented Graph-assisted Decentralized Federated Learning (AG_DFL) algorithm. Under limited participation conditions, our work is based on augmenting the graph to add virtual edges. The proposed mechanism allows nodes to relay additional models (virtual edge) while sharing their own. Evaluation under highly skewed data and volatile participation scenarios shows that the said algorithm improves DFL algorithm under extreme and non-IID distributions.

4. We finally propose a hybrid algorithm employing both memory and relay operations: Memory and Augmented Graph-Assisted Decentralized Federated

Learning (MAG_DFL) algorithm. Subjecting to the same set of conditions of limited participation conditions and asynchronous operations, our proposed algorithm outperforms both the baseline DFL and individually proposed algorithms.

In the rest of this document, we provide details of the Related Work in Ch-2 and the preliminaries in Ch-3. The details of the developed FLAGS framework and our detailed analysis of FL algorithms are provided in chapters 4 and 5. Our work on node selection for DFL under volatile settings is presented in Ch-6. We present the details of the three proposed algorithms, Memory-Assisted, Augmented-Graph Assisted, and the hybrid Memory and Augmented Graph-Assisted DFL in Ch-7. We present a concluding summary, potential future avenues, and extensions in Ch-8.

Chapter 2

RELATED WORK

Edge Intelligence tries to leverage the resources available mostly at the edge level of the network. However, this domain has a heterogeneous nature both in terms of the device and the communication protocols that link these devices. Federated learning, proposed in [22], enables these entities (nodes) to learn collaboratively while being coordinated by a central server without ever exchanging raw data. This mechanism decentralizes the learning process by enabling nodes to start from a global model and train on the locally available data. Federated Learning ensures the users' privacy since no data is communicated between the nodes during the training process. Subsequently, multiple nodes are then solicited by the central server to share their parameters, which are then aggregated by the server itself (Fig 1.2). The updated model is then broadcast to the nodes, repeating the process until convergence. The inference process takes place on the client itself, and thus, data is never transmitted over the network, which ensures user and client privacy. A comprehensive survey in [24] classifies the challenges for federated learning, dividing the issues as either "Algorithmic" or "Practical" in nature. Furthermore, the FL process is envisioned for a highly heterogeneous environment. The search for optimal application of FL has resulted in various FL algorithms designed to benefit from edge network topology uniquely. The impact of some of the most prevalent among these must first be characterized to adapt FL accordingly. To this end, some of the representative works that have focused on the FL algorithms and simulation frameworks have been covered in this section.

2.1 Federated Learning Algorithms

Most of the current research in Federated Learning leans heavily toward centralization. Communication and model optimization are the two key parameters that are used to gauge the effectiveness of an FL algorithm. However, research indicates that careful selection of design parameters may yield competitive results for decentralized FL as well. The authors in [37] describe a fully decentralized FL algorithm with IID data distributed across users interacting according to a directed graph. In [49], the authors present a gossip-aggregation framework for FL. The research reports more favorable results from gossip learning-based FL with uniform data distribution across the nodes. More recently, [50] suggests the application of the Decentralized Stochastic Gradient Descent (*DSGD*) [51] for D2D belief aggregation in the presence of wireless impairments. A Consensus-based Federated Averaging (CFA) for dense IoT networks is proposed in [52]. Their analysis suggests that serverless cooperation between devices may also yield results comparable to FedAvg. Extending their work, the authors in [35] evaluate Consensus-based Federated Averaging (CFA) and CFA-Gradient Exchange (CFA-GE) for dense IoT networks with D2D interaction. Their results support the original hypothesis indicating that although slow to converge, CFA and CFA-GE achieve performance similar to FedAvg with communication restricted only to device level. In [53], the authors try to tackle the problem of data heterogeneity in a decentralized fit learning setting. The authors propose a peer-to-peer model exchange method with model fusion using Mutual Knowledge Transfer.

Other research has also proposed hybrid FL algorithms employing hierarchical aggregation in conjunction with D2D interaction. In [54], the authors perform divergence-based client grouping in a Hierarchical Federated Learning (HFL) scenario. The latency analysis of a hierarchical FL system operating in a heterogeneous cellular network is provided in [55], where the local aggregation is conducted at Mobile Base Stations. The impact of HFL on training time and energy consumption is investigated in [56]. The same work also indicates that a trade-off between latency

and computation may be achieved, resulting in better performance than centralized FL. A two-time scale Hybrid FL model is proposed in [57], which compliments device-device communication with hierarchical server-based aggregations. This work introduces a control algorithm scheduling global aggregations, local interactions, and learning rate to achieve a convergence rate of $\mathcal{O}(1/t)$. The work in [58] uses gossip interaction between the devices before allowing them to upload their models to the hierarchical servers. The resultant FL algorithm provides near-optimal results even in the presence of reduced communication frequency and volume. Device clustering in the same network location is suggested in [59], which uses the corresponding mobile edge nodes for local aggregation operations. Their suggested method, in conjunction with a cosine-similarity-based device filtering, attains higher convergence speeds using less number of local updates. Then [60] experiment with FL using clustered devices where the Cluster Heads, selected from the devices, are awarded reputation scores by the members and may communicate with their one-hop neighbors. The results indicate that the proposed configuration improves network efficiency.

The work in [61] compares the communication efficiency of split learning and FL. This research evaluates the learning techniques for increasing the client population as well as increasing the size of the global dataset. The results indicate that split learning favors the former, whereas FL better serves the latter. In [62], the authors compare FedAvg, Federated Stochastic Variance Reduced Gradient (FSVRG), and CO-OP FL algorithms to investigate the impact of non-IID distribution of various FL optimization techniques. However, the comparison only considers centralized FL aggregation for the mentioned optimization schemes. The results indicate that FedAvg fares better than the other two techniques for non-IID data distribution. A detailed empirical analysis of federated and gossip learning conducted in [63] shows that gossip performs competitively with federated learning.

2.2 The non-IID Challenge

The presence of non-IID data remains the most pressing challenge for FL. One of the foremost works on challenges faced by FL is [24, 64] that also suggests that

non-IID data results in degraded performance for CFL schemes such as Federated Averaging (FedAvg) [22] and Deep Gradient Compression [65]. The ubiquity of this challenge remains the central focus of [66], which tries to assess the impact of non-IID data on FL. The research indicates that the presence of non-IID data increases divergence across all layers of the local DNN, exacerbating as the distribution skew becomes extreme, i.e., the presence of samples from a limited number of classes. The authors offer a potential solution to this problem by sharing a representative dataset among all participant devices. Some of the ways the non-IID nature of data manifests across devices have been covered in [67]. These include uneven feature and sample distribution and concept shift which entails a mismatch between labels and features at the device level. The proposed solution uses clustering client inputs based on the similarity between local and global models. The concept of client drift is presented in [68] and addresses the phenomenon in which nodes reach the local optima away from the global optimum, aggravated by the presence of non-IID data. The adverse impact of non-IID data on the performance of FedAvg is investigated in [69]. This work indicates that the presence of non-IID across devices reduces convergence rate, providing theoretical arguments for learning rate decay to reduce client drift. The research in [70] suggests that random selection of participants in FL scenarios leads to degraded performance, particularly when data at the nodes follow a non-IID distribution. The authors propose a statistical utility measure to ascertain the benefit of choosing a particular node in an aggregation round.

2.3 Existing Frameworks

Benchmarking FL is more significant as the envisioned operating environment is highly diverse and different from centralized or silo-based distributed learning. It is, therefore, necessary that an effective benchmark be capable of allowing heterogeneity in device behavior, data distribution, and connectivity. Additionally, the benchmark should be able to support different cooperation schemes that form the basis of different FL algorithms.

Various benchmarking frameworks have taken different approaches to satisfy the

aforementioned objectives. These have been aimed at achieving rapid prototyping [71], higher scalability [72, 73], and realistic data heterogeneity [74, 75]. Among these options, FedML [76] additionally offers topology management by allowing the configuration of multiple FL algorithms. However, it is pertinent to note that these frameworks are designed to simulate either FedAvg or a single FL algorithm for a given environment. This indicates that greater work is involved in simulating various FL algorithms to capture more accurate performance details.

This work presents the performance of key FL algorithms distinguished in their ability to utilize various network levels (Fig. 1.1). In order to achieve nuanced evaluation and ensure *extensibility*, this work presents the **FLAGS** framework. The framework allows each network entity, whether a node or a server, to emulate a realistic behavior separately. By implementing a range of functionalities associated with each entity, this framework allows diverse network interactions, enabling multiple FL algorithms to be simulated easily. Each device has various associated functions to enable multiple operating conditions and device behaviors. During the experiments, the FL algorithms are subjected to some of the key challenges facing general FL research. The empirical analysis of the performance of these FL algorithms is conducted by varying the operating parameters, such as device participation, communication noise and data distribution.

Federated Learning (FL) and Decentralized Federated Learning (DFL) are envisioned to operate in a highly heterogeneous environment. While FL research has seen remarkable interest, challenges such as data orchestration and privacy, communication latency, and server-related issues remain major impediments. Based on the principles of widely researched distributed optimization, DFL further democratizes learning by allowing devices to learn and exchange models among themselves without the need for a coordinating entity.

2.4 Decentralized Federated Learning

In DFL, nodes communicate with their peers in their respective neighborhoods without server intervention. This allows DFL added communication efficiency and better

privacy preservation. A DFL framework with a Bayesian parameter space is introduced in [37]. The work carried out in [35] extends the application of DFL to an industrial scenario with a large-scale Internet of Things (IOT) sensor network. They explore both model exchange in Consensus-based Federated Averaging (CFA) and CFA with gradient exchange (CFAGE). A gossip learning approach is evaluated for uniform bandwidth assignment in [63]. The authors show that for the given scenario, gossip learning performs comparably to FL and contend that the overall sentiment towards DFL should be encouraging. An Over-The-Air (OTA) DFL setup is explored in [50]. The authors suggest mechanisms to assist DFL in coping with wireless impediments and depict the efficacy of DFL in random network topologies. In [77], the authors provide an extensive survey of DFL research and some of the major characteristics involved in its characterization. They detail the major network topologies, key performance indicators, and various federation architectures. Their work also highlights research into DFL depending upon gossip, peer-to-peer and device-to-device communication. In [78], the authors suggest a DFL framework with unreliable communication. Their work focuses on using User Datagram Protocol (UDP) for model transmission and shows that the convergence rate at par with DFL may still be achieved.

Asynchronous DFL From a synchronization perspective, the FL landscape is divided into synchronous and asynchronous FL. Compared to the synchronous setting, the asynchronous environment allows the nodes to conduct training according to their own capabilities and available resources. Furthermore, the nodes are permitted to conduct aggregation without waiting for all the nodes in the neighborhoods. In [79], the authors propose an asynchronous DFL algorithm that allows utilizing the faster nodes for aggregation without waiting for stragglers. The work in [80] presents a fault diagnostic scenario enabling photo voltaic stations to learn fault features from their neighbors asynchronously. The work proposes a hybrid Federated learning model with both the presence of a server and decentralized node

interactions. the nodes are allowed to communicate among themselves as well as share their models with the global server whenever it is possible for them. The aggregation, both at the nodes and the server, takes place once a defined proportion threshold of received models is achieved. Finally, the work in [81] proposes an asynchronous DFL algorithm communicating over wireless network Edge. The authors suggest that the design algorithm is robust to the impediments of a heterogeneous wireless network and can operate in a time-varying communication topology in the presence of stragglers.

2.5 Topology Optimization

A major avenue of DFL research has adopted topology-configuring schemes to improve the convergence rate and improve communication efficiency in DFL. In [82], the authors propose the optimization of the network topology and model compression under heterogeneous communication and conditions and non-IID data distribution. They propose consensus distance as a metric and propose an algorithm CoCo, that dynamically constructs a network topology to determine compression ratios for communication efficient operation. Then [83] provides theoretical convergence guarantees for gossip Heritage learning. They also designed a bandwidth-aware gossip-matrix generation algorithm. They also propose a model specification method that allows each node to communicate a highly specified model with one peer in a gossip round. The work in [84] devises a way to assign mixing weights to the neighboring nodes for synchronous DFL. They deploy an encoder of model weights and an attention module to compute the weights from the pairwise similarity between the representations. To specify the topology, they remove the top K neighbors with the smallest mixing weights, thereby reducing communication costs. The authors of [85] propose a sparse training technique with each node keeping only a fixed number of parameters active throughout the training by using a personalized mask. The averaging is conducted using only the intersection weights from the received models. The model sparsification process reduces the communication cost and computation load. The experiments indicate that a fully connected topology shows higher con-

vergence rates than sparse ones.

2.6 Device Volatility

Device participation in a realistic FL setting is subject to multiple factors, including activity status, resource constraints, or the dynamic nature of the node itself. The overall learning environment becomes highly volatile when such node behavior is expected from a significant proportion of the nodes. The research in [86] explores the volatility aspects of FL training in an Intelligent Transportation Setting (ITS) where all the nodes are inherently dynamic and constantly join and leave the system. The proposed solution employs Road Side Units (RSUs) as intermediate parameter servers and intelligent resource allocation to undertake FL. An example of a reward-oriented participation behavior for FL is explored in [87]. The process is modeled as a Minority Game (MG) in which the nodes decide whether to participate or not based on a reward-resource trade-off. A coalition-based MG is proposed to incentivize participation and reduce *defectors*. The work in [88] tackles volatility in an FL setting by proposing a deadline-based node selection mechanism. The node selection process is further refined by Exploration and Exploitation-based client selection (E3CS). The best-case scenario is shown for the regret measure of the suggested solution equaling that of an even-random selection. The same conclusion was drawn empirically in our earlier work in [48], which explores node selection under volatile DFL settings. The results indicated that in high volatility scenarios, a random selection of nodes ultimately leads to better convergence rates even under extreme data distributions. The work in [89] explores the impact of volatility in FL settings. The authors propose a Cumulative Effective Participation Data (CEPD) measure as the optimization objective. The work categorizes volatility into set, statistical, and training volatility contingent on nodes joining or leaving the system, dynamic data characteristics, and participation. The server uses a multi-arm bandit approach to learn the effective participation data and attempts to maximize the cumulative measure.

2.7 Memory and Relay Mechanisms in FL

One of the major factors in enabling fully decentralized behavior is the ability to act as relays for neighboring nodes. The work [90] proposes a RelaySum mechanism for Decentralized Federated Learning for spanning trees. The authors consider each node to receive and send 2 models per iteration in a spanning tree topology, although their analysis extends to other graph topologies as well. Another work in [91] performs a relaying operation in a centralized FL environment where the node aggregates models from its neighborhood before forwarding them to the parameter server. The authors suggest that the impact of poor connectivity may be mitigated by nodes in the locality having better participation and connectivity options. The authors in [92] focus on the over-the-air (OTA) implementation of FL in wireless environments. The solution is based on a two-phase aggregation process where the first stage involves the nodes and the clients sending their updates to the relay and the Access Point (AP). In the next stage, both the relays and nodes send the updates to AP, following which the aggregation error is established based on this relay-assisted FL.

The work in [93] proposes MIFA, a memory-augmented Federated Learning framework that avoids excessive delay by employing a deadline and memory-based aggregation mechanism. Each node maintains the previous global model and the updated model and sends the difference to the server to avoid memory implications at the server. In [94], the authors suggest FedVARP, a variance reduction method for FL tackling partial client participation. The server maintains in its memory all received updates, and for the nodes not participating in a particular round, it approximates their models by the ones stored in its memory. The memory utilization at the server side is catered for by clustering similar updates as a single model.

Chapter 3

PRELIMINARIES

As with any distributed learning process, it is important to understand the underlying network topology connecting the various entities. Federated Learning offers an efficient means of distributed learning at the Edge Network. This section first establishes a System Model for the interaction between network entities such as nodes and servers connected by various communication links. The mathematical framework behind the FedAvg algorithm is also presented, which is extended in subsequent sections to describe major FL algorithms.

3.1 System Model

Network Topology The system model is defined using a set of nodes, edge servers, and a global cloud server. A stochastic formulation is used to establish the interaction between the nodes at the device layer as well as between the device and hierarchical servers at the edge layer. A set of $\boldsymbol{N}$ static nodes are randomly distributed in $\boldsymbol{C}$ geo-proximal clusters ($\mathcal{C}$) representing natural grouping. Each cluster may be associated with its own Cluster Head randomly selected from the cluster. Cluster Head selection and Cluster membership are active areas of research. However, the behavior in the presence of one has been replicated. One node from each cluster $\mathcal{C}_i$ is designated randomly as a Cluster Head. This work intends to evaluate the performance under Clustered FL operation and thus assumes automatic membership of proximal nodes. Each node i assumes full knowledge of its neighborhood $n_i \in \boldsymbol{N}$ such that $n_i = \{j \in \boldsymbol{N} | \{i\} : (i,j) \in \mathcal{E}\}$. The graph $\mathcal{G}$ is also defined with its adjacency matrix $\mathcal{W} \in \{0,1\}^{\boldsymbol{N} \times \boldsymbol{N}}$ that describes the edge information of all the nodes in the graph. While $\mathcal{W}$ represents the adjacency information of a static graph, a dynamic behavior by the nodes can be encompassed by sampling $\mathcal{W}^t \sim \mathcal{W}$. In the

setting considered for this work, both $\mathcal{W}$ and $\mathcal{W}$ are row-stochastic matrices which is shown in Ch 7. It is a realistic assumption that $\mathcal{W}$ may be obtained by the local interaction of the nodes with a fractional overhead. However, the behavior in the presence of one has been replicated. One node from each cluster $\mathcal{C}_i$ is designated randomly as a Cluster Head. This work intends to evaluate the performance under Clustered FL operation and thus assumes automatic membership of proximal nodes.

A random graph topology is assumed for the purpose of this work, with each node i able to communicate within its one-hop neighborhood n_i. The network topology forms an undirected graph $G = (\boldsymbol{N}, \mathcal{E})$ where $\boldsymbol{N} = \{1, 2, \ldots\ldots N\}$ is the set of nodes and $\mathcal{E}$ is the set of edges. The sparsity constraint on the graph structure stipulates that $\mathcal{E} << \mathcal{E}_{max}$ where $\mathcal{E}_{max} = \frac{\boldsymbol{N}(\boldsymbol{N}-1)}{2}$. The sparsity condition ensures that the graph density $\zeta << 1$. However, it is assumed that the graph remains *reachable*, implying that there are no isolated nodes. The devices are assumed to operate stochastically in pursuing local updates and participation in aggregation rounds.

Each node i is assumed to have a degree n_i and possesses knowledge about its neighborhood. However, it is pertinent to note that a device's neighborhood is not restricted to its cluster, and devices can be connected with those in other clusters. The link density at the device level is controlled by a set of two parameters (γ, υ). The probability of a link between geographically proximal devices lying within a cluster $\mathcal{C}_i$ is:

$$\underset{m,n \in \mathcal{C}_i}{P(\xi_{mn})} = \{\gamma \mid 0 < \gamma < 1\}$$

Similarly, the probability of a device-level link between two nodes belonging to distinct clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ is:

$$\underset{p \in \mathcal{C}_i, q \in \mathcal{C}_j}{P(\xi_{pq})} = \{\upsilon \mid 0 < \upsilon \leq \gamma\}$$

The parameters υ and γ are used to alter the network topology from a random place-

Param	Description
$\boldsymbol{\theta^t}$	Global model parameters at time t
$\boldsymbol{\theta_k^t}$	Model parameters of node k at time t
$\boldsymbol{\theta^*}$	Optimal parameters of the global model
$\boldsymbol{\mathcal{L}}$	Global loss function
$\nabla\mathcal{L}$	Gradient of the loss
$\boldsymbol{x}, \boldsymbol{y}$	Input feature vector, reference output
α_k	Learning rate at node k
η_k	Aggregation weight associated with node k
$\mathcal{D}$	Global dataset partitioned among the nodes
$\mathcal{D}_k$	Dataset associated with node k
$\boldsymbol{N}$	Number of nodes in the network
l_k	Local loss function associated with node k
$\xi_{l,m}$	Edge link between nodes l and m
$\mathcal{C}_i$	Cluster Head (CH) of the cluster i
$\boldsymbol{C}$	Number of geo-proximal clusters in the network
d_k	Neighbor cooperation probability of node k
p_k	Edge server cooperation probability of node k

Table 3.1: List of parameters used in the system model and the algorithmic description.

ment to a highly clustered setting depending upon the target environment. For these parameters, a setting of ($\gamma \to 1,\ \upsilon << \gamma$) indicates that the proximal groups are densely connected, with almost every device sharing a link with the other within the group. On the other hand, only a few edges exist between devices between different groups, indicating a reduced probability of the device's ability to communicate directly with devices that are farther off. In contrast, ($\gamma \to 1, \upsilon \to 1$) indicates a densely connected device layer where each device is connected to a large number of other devices. Finally, ($\gamma \to 1, \upsilon = 0$) indicates that devices only maintain connections within a proximal group without any inter-group connectivity.

The framework can simulate both ideal and noisy links with zero-mean Gaussian noise ($\mathcal{N}(0, \sigma)$). Each device has a probability d_k of participating in a neighborhood aggregation and p_k for an edge or cloud server-based aggregation. These probabilities reflect the device status (active/inactive) as well as stragglers that may choose not

to participate in certain aggregation rounds. Furthermore, the nodes are assumed to be fully capable of undertaking local learning operations, including adjusting hyperparameters.

Node Volatility and Asynchronous Participation The current environment setting realizes volatile participation from the nodes through the use of the parameter p_{ij}^t that controls the ability of the node i to communicate with its neighbor $j \in s_i^t \subset n_i$ during the round t. We assume that node i exchanges its model with $j \in s_i^t$ nodes in a round t if $p_{ij}^t = 1$, with no prior assumption on the distribution of p_{ij}. The overall setting maintains considerable volatility in a given round t, assuming that $0 < |s_i| << |n_i|$. This implies fractional participation from the neighborhood of node i in a given aggregation round. The entire process is considered under asynchronous FL settings, during which devices can control local update rounds e_i and participate in aggregation. The communication rounds are indexed by t.

3.2 Federated Learning Methodology

The global objective in a Federated Learning setting is a weighted sum of the local loss functions:

$$\boldsymbol{\mathcal{L}}(\boldsymbol{\theta^t}) = \sum_{k=1}^{\boldsymbol{N}} \eta_k \mathcal{L}_k(\boldsymbol{\theta_k^t}) \tag{3.1}$$

where $\boldsymbol{\mathcal{L}}(\boldsymbol{\theta}^t)$ represents the global loss function and $\mathcal{L}_k(\boldsymbol{\theta_k})$ and η_k are the local loss function and weight associated with node k at time t. The FL objective is to minimize the weighted sum of the individual losses of $\boldsymbol{N}$ devices

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^M} \boldsymbol{\mathcal{L}}(\boldsymbol{\theta^t}) = \min_{\boldsymbol{\theta}} \sum_{k=1}^{\boldsymbol{N}} \eta_k \mathcal{L}_k(\boldsymbol{\theta_k^t}) \tag{3.2}$$

Each $\mathcal{L}_k(\boldsymbol{\theta_k^t})$ at time t is associated with an M-dimensional model parameterized by $\boldsymbol{\theta_k^t} \in \mathbb{R}^M$ and the weight η_k for node k. The initial parameters of the model, $\boldsymbol{\theta}^t$, are shared with the devices through a server. The local loss at a device is calculated

through the training as follows

$$\mathcal{L}_k(\boldsymbol{\theta_k^t}) = \frac{1}{|\mathcal{B}_k|} \sum_{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathcal{B}_k} l(\boldsymbol{x_i}, \boldsymbol{y_i}; \boldsymbol{\theta}_k^t) \tag{3.3}$$

where $l(\mathbf{x}_i, \mathbf{y}_i; \boldsymbol{\theta_k^t})$ represents the loss of the machine learning task (e.g cross-entropy, mean squared error) calculated for the local model for the current model parameters. This is calculated for each point $(\boldsymbol{x_i}, \boldsymbol{y_i})$ in the minibatch $\mathcal{B}_k$ sampled from the local dataset $\mathcal{D}_k$. Using the principles of Distributed Stochastic Gradient Descent (DSGD), each device with the learning rate α_{kr} for a round r, updates its model as:

$$\boldsymbol{\theta}_k^{t+\tau} = \boldsymbol{\theta}_k^t - \alpha_k^t \nabla \mathcal{L}_k^t(x_i, y_i; \boldsymbol{\theta}_k^t) \tag{3.4}$$

where $\nabla \mathcal{L}_k^t(x_i, y_i; \boldsymbol{\theta}_k^t)$ are the gradients calculated by node k during the local update round t and α_k^t is the learning rate associated with kth node at time t. The updated model parameters from Eq. 3.4 or the gradients calculated above are then shared with the server.

This process is conducted for e_i update rounds before the aggregation phase. For aggregation, each node, with a probability $p_{ij} \in \{0, 1\}$, shares its locally learned parameters $\theta_i^{t+\tau}$ with $s_i \in n_i$ set of nodes. The number of nodes in s changes with $p_{ij}^t \ \forall \ j \in s_i$ such that $s_i = \{\phi \mid p_{ij} = 0 \ \forall \ i \in \mathcal{N} \ \& \ j \in n_i\}$ and $s_i = \{n_i \mid p_{ij} = 1 \ \forall \ i \in \ \mathcal{N} \ \& \ j \in n_i\}$. Upon receiving models from $\pi_i : \{\pi_i | \pi_i \in n_i\}$ set of nodes from its neighborhood, each node i performs local aggregation using the received models $\theta_r^{t+\tau} \in {\pi_i}^t$ along with its own model $\theta_i^{t+\tau}$ with their corresponding weights to generate θ_i^{t+1}:

$$\theta_i^{t+1} = \eta_i \theta_i^{t+\tau} + \sum_{r \in \pi_i} \eta_r {\theta_r}_i^{t+\tau} \tag{3.5}$$

where η_i and η_r denote the corresponding weights assigned to the contribution of the node i's own and r received models, respectively, towards the aggregation at

node i and

$$\eta_i + \sum_{r \in \pi_i} \eta_r = 1$$

where η_i and η_r denote the corresponding weights assigned to the contribution of the node i's own and r received models, respectively, towards the aggregation at node i and

$$\eta_i + \sum_{r \in \pi_i} \eta_r = 1$$

The complete process is repeated over multiple communication rounds until convergence is achieved.

As evident from Eq. 3.2-3.5, the entire Federated Learning process bypasses the need for the devices to share data at any stage, offering a strong privacy advantage over centralized forms of ML. It instead relies on communicating model parameters to the local server. The server is also required to share the aggregated model with the connected nodes for the subsequent rounds, adding to the overall communication volume needed to execute FL successfully. Such servers are typically located at the network cloud level, offering multiple services. Thus, frequent communication to and from these servers becomes an extremely expensive operation in addition to straining the latency requirements of various edge services in addition to subjecting the shared models to additional adversarial risks.Table 3.1 presents the list of parameters used in this work.

Furthermore, the entire process is considered under asynchronous FL settings, during which devices can control local update rounds e_i and participate in aggregation. The value of e_i depends on multiple factors, including device activity status, computation capacity and size of the local dataset. While an asynchronous setting introduces additional parameters that may be significant in ascertaining the overall performance not in the purview of current work, it remains closer to a realistic operation. It offers greater autonomy to the participating devices.

Chapter 4

FEDERATED LEARNING ALGORITHMS SIMULATION (FLAGS) FRAMEWORK

Federated Learning (FL) provides an effective mechanism for distributed learning. However, it is expected to operate in a highly diverse setting with distinct behaviors from the participating nodes as well as dynamic network conditions. The FL performance, therefore, is subject to change due to the highly transitory nature of the overall system. An efficient simulation framework must be flexible to allow a range of participant behaviors, interactions, and environmental characteristics. In this chapter, we present the Federated Learning Algorithms Simulation (FLAGS) framework that we propose as a lightweight FL implementation and testing platform. FLAGS framework allows for a wide range of device behaviors and cooperative mechanisms, enabling rapid testing of multiple FL algorithms.

4.1 Introduction

Federated Learning (FL) [22] has become a widely researched domain for distributed learning. It enables the individual nodes to learn from their local data and share the models with other participating entities. The shared models are fused together to generate a consensus model, which forms the starting point for the next learning round until convergence is reached. FL is envisioned to operate in a highly heterogeneous setting [24], allowing the participating nodes to cooperate toward building a consensus model. The major sources of heterogeneity are the environment, device behavior, and data distributions. A reliable and accurate evaluation of FL algorithms that caters to multiple operating characteristics should, therefore, be an important consideration.

With the widespread application of distributed learning, benchmarking FL has assumed even greater significance. The FL algorithms are envisioned for a highly heterogeneous setting with limited prior information. Accurate assessment of their performance requires that the simulation framework provide repeatable and uniform conditions, support easily configurable FL algorithms, possess the flexibility to implement new ones and cater to multiple types of communication links. An effective benchmarking platform should be versatile and capable of incorporating diverse device behaviors, environments, network topologies, and data distributions, as well as supporting various cooperative mechanisms. To enable rapid prototyping, accurate assessment, and experiment with multiple cooperative topologies, we present FLAGS[1].: the Federated Learning Algorithms Simulation framework. FLAGS is a lightweight FL prototyping framework allowing for an accurate assessment of multiple FL algorithms for highly realistic network topologies [95]. Inherent in FLAGS are different levels of device participation (controlled by parameters p_k for participation in central aggregation and d_k for controlling neighborhood aggregation), device selection mechanism for nodes and servers as well as generating multiple data distributions including IID, non-IID and extremely-skewed non-IID data partitions. It also supports synchronous and deadline-based asynchronous operation to replicate heterogeneous device behavior where device training progresses at different paces. The highlight of FLAGS lies in its capability to configure and subject multiple FL algorithms to a realistic multi-tiered environment, which allows users to propose the best environment-FL fit.

The distinguishing features of the FLAGS framework include configuring entities across the Edge Network environment, simulating multiple FL algorithms through a single framework, heterogeneous client participation including stochastic and volatile behaviors, enabling asynchronous node operation and formulating sparse, dense, and clustered network topologies.

[1]https://github.com/ahnaflodhi/FLAGS-v2

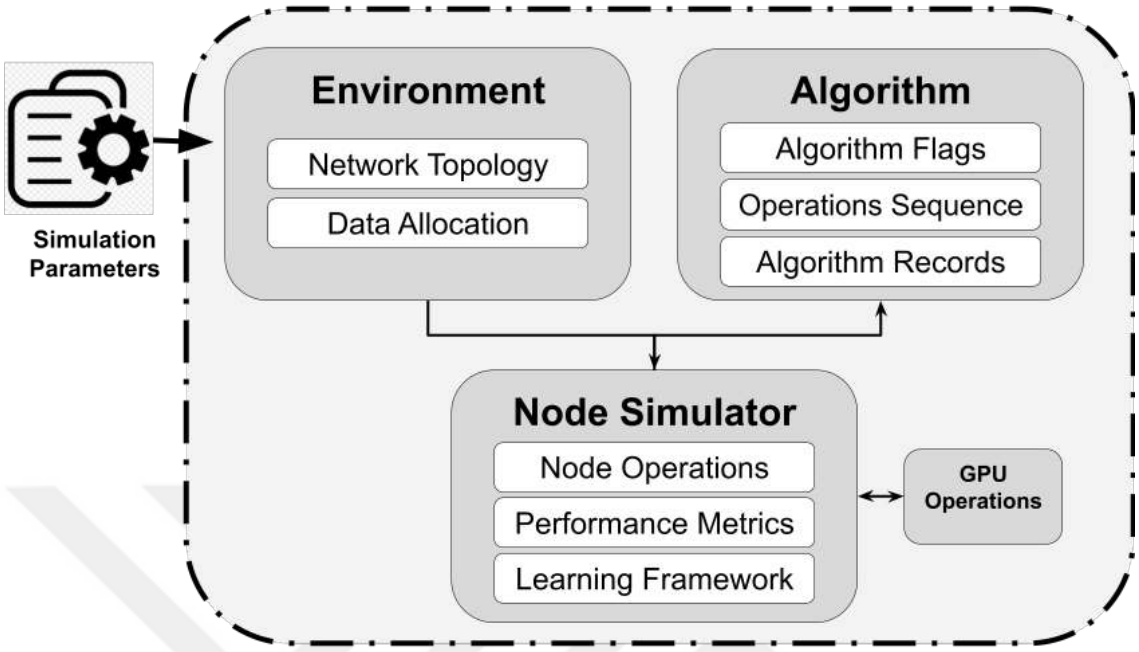


Figure 4.1: FLAGS Framework Software Architecture: Three main modules and their interactions.

4.2 Software Architecture

Existing FL benchmarking frameworks have adopted different mechanisms for implementing and testing node behavior in FL algorithms. However, most of such frameworks have been designed with a single FL algorithm in perspective [47]. FLAGS framework has been developed with the capability to configure and test multiple FL algorithms easily [96]. It allows incorporating various conditions during testing to assess the response under a diverse set of operating circumstances. The overall architecture is highly modular and enables users to easily augment existing functionality by adding new features. The FLAGS framework consists of three main modules, namely environment, algorithms, and node operations, implemented as independent classes as shown in Fig.4.1 and described next.

4.2.1 Environment

The *Environment module* allows the users to configure the target system model for FL operations. The generated environment mimics the edge network [5], generating

the device, hierarchical and cloud layers. The network topology is controlled by specifying the number of nodes $\mathcal{N}$, the number of clusters $\mathcal{C}$, the number of hierarchical servers $\mathcal{S}$ and a global cloud server $\mathcal{S}_g$. The underlying connectivity may be controlled to be dense or sparse by using the parameters γ and ν where $0 \leq \gamma, \nu, \leq 1$ control each node's connectivity within and outside of its assigned cluster, respectively. Higher values of γ and ν result in dense graphs with a value of 1 resulting in a complete graph. At the initialization, each cluster is assigned a ClusterHead (CH) C_i; however, the role may be re-assigned subsequently [97]. The hierarchical/edge servers each are assigned multiple clusters $\mathcal{C} > 1$. The edge servers are, in turn, connected to a global server, a role assigned to the last indexed server in the server array. The graph generated by this module precludes the presence of any isolated node, and *networkx* library can access its information.

The current topology simulates random graph. However, more topologies may be incorporated in the environment module using the options from the corresponding libraries. However, these changes remain insulated from other modules and do not affect their operations which enables multiple options with FLAGS framework.

4.2.2 Algorithms

The *Algorithm module* generates and maintains a record of the FL algorithms. It also dictates the sequence of operations, including local updates, validation, aggregation, and algorithm testing. Each algorithm is configured by setting a set of six flags. These flags dictate the interaction between various entities and the exact nature of the interaction as well. This interaction determines the level of communication between different network entities spread across various layers of the edge network. The flags set assigns two flags to each layer of the edge network, controlling the nature and specifics of the interaction. The following are part of the flags set: a) Device Op b) Device-Mode c) Cluster-Op d) Cluster-Mode e) Edge-Op f) Edge-Mode The *'Op'* flags specify whether model exchange across entities belonging to a particular hierarchy is permissible. *'Device-Op'* specifies Device-to-Device (D2D) interaction for device layer whereas *'Cluster-Op'* indicates interactions between ClusterHeads.

Algorithm / Interaction	D2D Op	D2D Mode	Cluster Op	Cluster Mode	Edge Op	Edge Mode
FedAvg	False	False	False	False	False	False
HFL	False	False	False	False	**True**	**Default**
DFL	**True**	**Default**	False	False	False	False
Gossip	**True**	**Random**	False	False	False	False
HDFL	**True**	**Default**	False	False	**True**	**Default**
HGFL	**True**	**Random**	False	False	**True**	**Default**
CFL	False	False	**True**	**Default**	False	False
iCFL	False	False	**True**	**Random**	False	False
iCDFL	**True**	**Default**	**True**	Default	False	False
DFL_Sel	**True**	**Model**	False	False	False	False
MA_DFL	**True**	**Memory**	False	False	False	False
AG_DFL	**True**	**Relay**	False	False	False	False
MAG_DFL	**True**	**Mem_AG**	False	False	False	False

Table 4.1: Flag settings for various Algorithms in FLAGS framework.

Finally, *'Edge-Op'* controls whether the devices can communicate with Edge servers for intermediate aggregation or not. The *'Mode'* flags indicate which particular mode of interaction is conducted. The *'Device-Mode'* set to *'model-selective'* will perform the node selection process at the device level based on *'cosine similarity'* or *'L2 Norm'*. The same flag set to *'stale-global'* allows using a stale global model as a reference to compute the similarity for target models. A number of algorithms may be configured using different configurations of the flags. Additionally, any number of new features may be added and used across different FL algorithms by specifying appropriate flag settings. A number of configured algorithms using the FLAGS are presented in Table-4.1.

4.2.3 Node Operations

In order to allow autonomy of operation to the participating devices, the *Node Simulator module* imparts the relevant functionality to all the devices configured for a particular FL algorithm. The topological information accessible to each node includes its local neighborhood, the assigned cluster, and the hierarchical server. Each node's primary functions include training, testing, and exchanging models with the designated devices while maintaining the record of the relevant performance metrics. Each node concurrently maintains three models, which include the updated model, the post-aggregation model, and the model from the previous round. Each node is assigned unique training and test datasets based on the overarching data

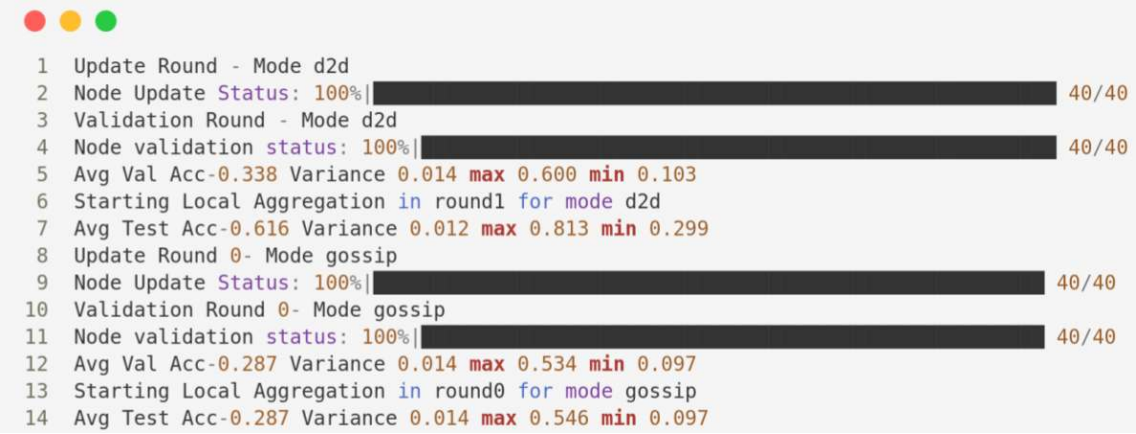


Figure 4.2: Default argument state: Each of these may be adjusted as input arguments to the `Main-Fed.py` file

distribution, i.e., IID, non-IID, skewed data distribution (by default, the test data is always IID distributed). Each node may alter its learning rate independently of others and can exhibit synchronous or asynchronous behavior. The nodes support several aggregation functions that may be invoked according to the algorithm. A separate *Servers* class governs the behavior of the global and hierarchical servers. Although implemented as a separate class, the servers, when invoked, only perform intermediate or global aggregation and pass the models back to the nodes.

4.2.4 Additional Modules

The three main modules are assisted by many other modules implemented either as separate classes or individual functions. The framework currently supports three different datasets, namely *MNIST, FashionMNIST* and *CIFAR-10*. The `Net` class implements the neural network layers and the associated operations. Testing and aggregation operations have been implemented individually so that they may be generally accessible. The supported models include shallow (2 Convolutional layers, 2 fully connected layers) and deep models (VGG-11), which may be used when initializing the execution. The framework supports various data distributions using the Dirichlet parameter ($\alpha > 0$). Higher values of α result in IID data distributions across the nodes, whereas fractional values make the data distribution increasingly

```
    parser = argparse.ArgumentParser()
    parser.add_argument('-b', type = int, default = 8, help = 'Batch size for the dataset')
    parser.add_argument('-t', type = int, default = 8, help = 'Batch size for the test dataset')
    parser.add_argument('-d', type = str, default = 'mnist', help='Name of the dataset : mnist, cifar10 or fashion')
    parser.add_argument('-n', type = int, default = 40, help='Number of nodes')
    parser.add_argument('-c', type = int, default = 9, help='Number of clusters')
    parser.add_argument('-ser', type = int, default = 3, help='Number of servers')
    parser.add_argument('-emin', type = int, default = 1, help='Min Number of epochs')
    parser.add_argument('-emax', type = int, default = 4, help='Max Number of epochs')
    parser.add_argument('-sim', type = float, default = 0.9996, help='Cosine Similarity Threshold')
    parser.add_argument('-r', type = int, default = 30, help='Number of federation rounds')
    parser.add_argument('-D', type = int, default = 5, help='Delay induced for Stale Global Models')
    parser.add_argument('-o', type = float, default = 0.75, help='Overlap factor in cluser boundaries')
    parser.add_argument('-s', type = int, default = 0, help = ' Skew for Exteme Non-IID distribution. Min: 1 - Max: Nu
m of classes') # Positive skew overrides alpha value
    parser.add_argument('-a', type = float, default = 0.1, help = 'Alpha parameter (+ve only) for Dirichlet based non-
IID distribution')
    parser.add_argument('-prop', type = float, default = 0.9, help = 'Proportion of nodes chosen for server aggregatio
n : 0.0-1.0')
    parser.add_argument('-aggprop', type = float, default = 0.9, help = 'Aggregation-Proportion: Proportion of nodes i
n neighborhood for device-level aggregation: 0.0-1.0')
    parser.add_argument('-model', type = str, default = 'shallow', help = 'Base model type to run the experiments')
    parser.add_argument('-clint', type=float, default = 0.15, help='Probability of Edges within a cluster')
    parser.add_argument('-clext', type=float, default = 0.2, help='Probability of Edges with nodes external to Cluster
of current membership')
```

Figure 4.3: Simulation Output

non-IID. A parameter *'s'* enables extremely skewed distributions, with each node getting samples from only 's' classes from the dataset, where the samples themselves are randomly distributed. It is pertinent to note here that employment of α and s is mutually exclusive, and $s > 0$ overrides any value of α. Several utilities are provided that assist in intermediate and final data logging and generating the performance plots.

Framework Extensibility It is pertinent to mention here that the framework implementation is built around possible extensibility. There a number of learning frameworks with various levels of capability, functionality and strengths. In order to ensure that FLAGS framework remains extensible, the modules have been developed in such a way as to permit users to extend FLAGS using a learning framework their choice without affecting the overall functionality. The modules Environment and Algorithms module have been developed using public domain Python libraries. On the other hand, the learning framework and dataset inclusion depends upon the choice of the learning framework which in this case happens to be Pytorch. However, users could easilt incorporate the same functionality using any other learning framework without having to adjust the core functions provided by the Algorithms and Node operations. This allows FLAGS framework to potentially be modified and

extended using different learning frameworks likes TensorFlow, Keras etc.

Additionally, the datasets employed by the FLAGS framework also correspond to the learning framework being used. Therefore, new datasets may be introduced or modified according to the learning framework being employed without requiring additional changes to the FLAGS framework.

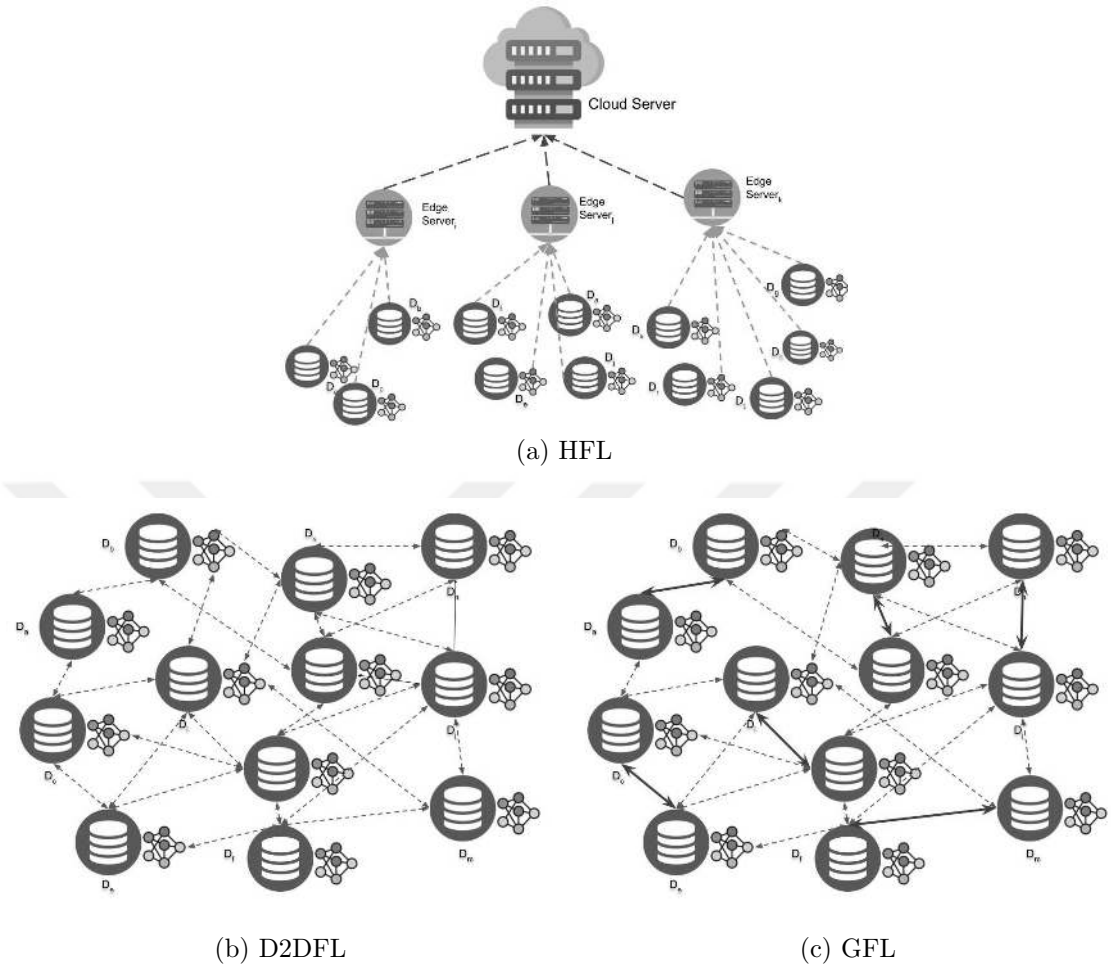
Chapter 5

COMPARATIVE ASSESSMENT OF FEDERATED LEARNING ALGORITHMS

5.1 Introduction

Existing research in Federated Learning (FL) shows a strong proclivity towards centralized orchestration. Access to updates from a large number of devices, a global model, and considerable computational and storage resources have traditionally made a strong case for Centralized FL (FedAvg). However, there exist a number of challenges [24] to such an operation, including adversarial attacks, malicious participation, privacy preservation, heterogeneous devices and participation, non-Independently and Identically Distributed (non-IID) data, and communication issues. Simultaneously, next-generation networks [36] envision considerable serverless interaction between the devices. Advances, such as application-specific data rates, Network Function Virtualization (NFV), and network slicing, have enabled researchers to propose various FL algorithms based on the permissible device interactions (see Fig.1.3 for the depiction of these links). The proposed methods show multiple advantages over Centralized FL (FedAvg) under various settings. Furthermore, current and future devices come equipped with multiple communication links. This fact negates the limitation of operating a single FL algorithm at all times. The challenge then arises regarding the selection of an FL algorithm to employ under various conditions. The comparative characterization of FL algorithms conducted in this work is aimed at bridging this gap. The analysis conducted here paves the way for establishing the economy of operation for these algorithms for various operating scenarios in a particularly heterogeneous Network Edge.

The remainder of this chapter is organized as follows: An overview of the system



(a) HFL

(b) D2DFL

(c) GFL

Figure 5.1: Main Federated Learning Algorithms. Dashed lines represent links between various entities, whereas only solid lines in (c) depict active established-pair links in GFL.

model and FL is presented in Section-3.1. Details of the FL algorithms have been provided in Section-5.2. This is followed by a description of the experiments and performance analysis in Section-5.3 and 5.4. The report concludes with a summary of the important findings in Section-5.5.

5.2 Federated Learning Algorithms

Federated Learning was initially conceived as a two-tiered learning framework alternating between clients/nodes performing the training and the cloud servers aggregating the local models to yield a global model. In order to increase the efficiency, the

envisaged FL was further spread over the Edge Network to benefit not only from the Edge devices' capabilities but also address the challenges inherent in centralization. This evolution has yielded various FL algorithms suitable to a variety of scenarios, including vehicular networks, dense IoT networks, and cross-silo operations between relatively large data centers. These algorithms employ three different types of communication links as shown in Fig. 1.3 associated with different communication costs: (a) Device-to-Device (D2D) (b) Device-to-Edge (D2E) (c) Edge-to-Cloud (E2C)

D2D links are limited to the device level, whereas D2E links connect the nodes with the edge servers. Finally, E2C links connect the edge servers to the core network. While D2D links and D2E links operate independently, E2C links require the latter for information to be transmitted from the devices to the edge server for onward transmission to the core network. All subsequent communication analyses in this work use this fact to establish the effectiveness of a particular form of communication in FL operations.

Algorithm 1: Hierarchical Federated Learning

Input: Initial Model parameters $\boldsymbol{\theta}_0$, $\mathcal{S}$ hierarchical servers with device association

Output: Global model parameters $\boldsymbol{\theta}_{t\to\infty}$

Initialization by global server: $\boldsymbol{\theta}_0$ received by each hierarchical server S_i and broadcast among the connected nodes

for *Nodes* $k = 1 \ldots N$ ***(in parallel)*** **do**

  Perform Local update for e epochs

  $\boldsymbol{\theta}_{\boldsymbol{k}}^{\boldsymbol{t+\tau}} = \boldsymbol{\theta}_k^t - \alpha_k^t \nabla \mathcal{L}_k(x_m, y_m; \boldsymbol{\theta}_k^t)$

  TX: With a probability p_k, send the $\boldsymbol{\theta}_k^{t+\tau}$ to the hierarchical server for aggregation

end

for *Servers* $\mathcal{S}_i \; \forall \; i = 1 \ldots\ldots \mathcal{K}$ ***(in parallel)*** **do**

  AGG: Aggregate the models received from the n_i nodes

  $\boldsymbol{\theta}_{S_i}^{t+T} = \boldsymbol{\theta}_{S_i}^t + \sum_{j \in n_i} w_j \boldsymbol{\theta}_j^{t+\tau}$

end

if *round % f == 0* **then**

  for *Servers* $S_i \; \forall \; i = 1 \ldots\ldots \mathcal{K}$ ***(in parallel)*** **do**

    Upload models to the global server

  end

  Global Aggregation

  $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \sum_{j \in \mathcal{K}} \eta_j \boldsymbol{\theta}_j^{t+T}$

  Send model back to Edge Servers S_i

  Edge Servers S_i broadcast model parameters $\boldsymbol{\theta}^{t+1}$ to all nodes connected respectively

end

A key aspect in major FL literature is the absence of cross-configuration comparison and evaluation of FL algorithms. Such an analysis promises to identify the optimal employment of these FL algorithms based on the operating environment and the nature of the problem being addressed. This section first provides a description of these algorithms, followed by details of the experiments in the subsequent sections.

5.2.1 Hierarchical Federated Learning

Hierarchical Federated Learning (HFL) introduces intermediate model aggregation closer to the data origin using an edge server [56]. This FL algorithm introduces $\mathcal{F}$ aggregation layers between the nodes and the global cloud server, $\mathcal{F}$ increasing toward the cloud layer. Cellular base stations or Mobile Edge Computing (MEC)

Servers are envisioned to realize this role [8] for their respective connected devices. Each network level, as depicted in Fig. 5.1(a), aggregates the models received from the previous layer and passes them to the next layer in the hierarchy. This scheme offers a reduction in the overall upstream communication as well as progressively offloading the computational load at the edge network.

Each hierarchical layer $\mathcal{F}$ houses $\mathcal{K}_{\mathcal{F}}$ edge servers. Each server at layer $\mathcal{F} = 1$ aggregates models from the devices linked to it respectively. The pseudocode for HFL has been laid out in Alg. 1. The initial model parameters, i.e., $\boldsymbol{\theta}_0$ are initialized by the global server $\mathcal{S}_g$ and shared with the nodes. The nodes, in turn update the model over a mini-batch of size $\mathcal{M}$ to yield $\boldsymbol{\theta}_k^{t+1}$. After completing e local updates, each node shares its updated parameters with the hierarchical server S_i with a probability p_k. The servers aggregate the received models, which are then shared with the servers in the next layer/global server after f aggregation rounds. At this stage, the global server aggregates the input models from the $\mathcal{K}$ hierarchical servers and generates the global model $\boldsymbol{\theta}^{t+\tau}$, which are then shared through the local servers with the associated nodes.

Algorithm 2: Device-to-Device Federated Learning

Input: Model parameters θ_k for each node, degree n_k
Output: D2D Aggregated Models
for *Nodes* $k = 1 \ldots N$ ***in parallel*** **do**
  Update local model for e epochs
  $\boldsymbol{\theta_k^{t+\tau}} = \boldsymbol{\theta}_k^t - \alpha_k^t \nabla \mathcal{L}_k(x_m, y_m; \boldsymbol{\theta}_k^t)$
  Send model to 1-hop neighbors n_k with probability d_k
  Receive models from n_k 1-hop neighbors
  Perform weighted aggregation for the received models
  $\boldsymbol{\theta}_k^{t+1} = \boldsymbol{\theta}_k^{t+\tau} + \sum_{j \in n_k} \eta_j \boldsymbol{\theta}_j^{t+\tau}$
end

5.2.2 Device-to-Device Federated Learning

With Device-to-Device (D2D) communication [33], FL can be operated without the requirement of a central aggregator [35]. In a purely serverless FL framework, the nodes communicate with their immediate neighbors and resort to local aggregation Fig. 5.1(b). At a given time t, the nodes share their locally learned parameters

$\boldsymbol{\theta}_k^t$ with their neighborhood n_k. The recipients, in turn, perform aggregation of the received models combining the incoming parameters $\boldsymbol{\theta}_k^t$ with corresponding weights $\{\eta_j \forall j \in n_k\}$ to generate $\boldsymbol{\theta}_k^{t+1}$: The entire operation of D2DFL is depicted in Alg. 2.

Algorithm 3: Gossip Federated Learning

Input: Model parameters $\boldsymbol{\theta}_k$, optimization parameters
Output: Gossip-aggregated model
for *Nodes* $k = 1 \ldots N$ ***in parallel*** **do**
  $\boldsymbol{\theta}_k^{t+\tau} \longleftarrow$ Model-Update($\boldsymbol{\theta}_k^(t)$)
  $\boldsymbol{\theta}_k^{t+1} \longleftarrow$ *Gossip Exchange* ($\theta_k^{t+\tau}$)
end
Function `Model-Update(`$\boldsymbol{\theta}_l^t$`):`
  for *batch* b *of size* $\mathcal{M} \in \mathcal{D}_l$ **do**
    $\xi_b = (x_b, y_b)$ is the data sample pairs in b
    $\boldsymbol{\theta_l^{t+\tau}} = \boldsymbol{\theta}_l^t - \alpha^t \nabla \mathcal{L}_l(\xi_b; \boldsymbol{\theta}_l^t)$
  end
  return $\boldsymbol{\theta}_l^{t+\tau}$
Function `Gossip Aggregate(`$\boldsymbol{\theta}_l^t$`):`
  Perform aggregation handshake with a random node
  Exchange models for aggregation between the pair
  Aggregate received model
  $\boldsymbol{\theta}_l^{t+T} = \boldsymbol{\theta}_l^{t+\tau} + \eta_j \boldsymbol{\theta}_j^{t+\tau}$
  return $\boldsymbol{\theta}_i^{t+T}$

5.2.3 Gossip Federated Learning

Gossip communication is another form of decentralized communication wherein the nodes communicate with randomly selected peer nodes. Gossip Federated Learning (GFL) was among the early fully decentralized FL frameworks to have been proposed [98]. For a network with $\boldsymbol{N}$ nodes, the nodes form random pairs with other nodes from the network forming a gossip pair ($i,\ j\ \forall\ \ i,\ j \in\ \boldsymbol{N}$) as shown in Fig. 5.1(c). Once the connection has been established, the pair exchange their locally updated models $\boldsymbol{\theta}_i^t$, $\boldsymbol{\theta}_j^t$ and in turn aggregate with the received model parameters to generate $\boldsymbol{\theta}_i^{t+1}$ and $\boldsymbol{\theta}_j^{t+1}$. The process results in considerably curtailed communication costs. The details of the operation have been provided in Alg. 3.

5.2.4 Hierarchical Device-to-Device Federated Learning

Hierarchical D2DFL employs aggregation at the device levels, edge servers, and cloud servers. Thus the D2D, D2E and E2C links all are employed to enable such operation [57]. The proposed mechanism allows for devices to perform local updates on their model parameters $\boldsymbol{\theta}_k^t$ and share them with their n_k neighbors, leading to device level aggregation. Simultaneously, the devices also share the model parameters with the connected servers $\mathcal{S}_i$. The servers in turn aggregate the received models and share along the hierarchy $\mathcal{F}$ eventually leading up to the global server $\mathcal{S}_g$ as indicated in Fig. 5.1(d). The global model $\boldsymbol{\theta}^{t+\tau}$ gets disseminated back to the nodes through the respective servers and the process is continued till convergence. The details of the entire process has been outlined in Alg. 4. While greater cooperation is envisioned with this algorithm, the learning requires communication at both device and upstream levels.

5.2.5 Hierarchical Gossip Federated Learning

Hierarchical FL, as described in Alg.1, allows the device layer to interact with Edge Servers to aggregate local models, resulting in sub-global models. These models are then passed onto the global server for generating the global model $\boldsymbol{\theta}^t$. HD2DFL builds on this configuration by allowing device layer entities to interact among themselves before sharing the local aggregated models with the higher edge layers. The performance as shall be seen in Sec-5.4 indicates better performance albeit at the communication volume, which is almost the joint sum of HFL and D2DFL. This observation led the researchers in [58] to introduce gossip steps at the device level instead of the full neighborhood aggregation as indicated in Alg 5.

Algorithm 4: Hierarchical-D2D Federated Learning

Input: Initial Model parameters $\boldsymbol{\theta}_0$, Neighborhood information $\mathcal{G}_k = \{\boldsymbol{N}, \xi_k\}$, Hierarchical servers K with device association

Output: Global model parameters $\boldsymbol{\theta}_{t\to\infty}$

Initialization by global server: $\boldsymbol{\theta}_0$ received by each hierarchical server S_i and broadcast among the connected nodes n_i

for *Node* $k = 1 \ldots N$ ***in parallel*** **do**

  $\boldsymbol{\theta}_k^{t+\tau} \longleftarrow Model\ Update(\boldsymbol{\theta}_{\boldsymbol{k}}^{\boldsymbol{t}})$

  $\boldsymbol{\theta}_k^{t+\delta} \longleftarrow\ D2D\ Aggregate(\boldsymbol{\theta}_{\boldsymbol{k}}^{\boldsymbol{t+\tau}}, n_k)$;

end

if *aggregation round* $\%\ f == 0$ **then**

  for *Servers* $S_i\ \forall\ i = 1 \ldots\ldots H$ ***(in parallel*** **do**

    Sample q from the associated nodes and request parameters

    Aggregate received parameters: $\boldsymbol{\theta}_{S_i}^{t+\tau} = \boldsymbol{\theta}_{S_i}^{t} + \sum_{j\in q} w_j \boldsymbol{\theta}_j^t$

  end

  for *Servers* $S_i\ \forall\ i = 1 \ldots\ldots H$ ***in parallel*** **do**

    Upload model $\boldsymbol{\theta}_{S_i}^t$ to the global server

  end

  Global Aggregation : $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \sum_{j\in S_i} \eta_j \boldsymbol{\theta}_j^t$

  Send model back to nodes through respective Edge Servers S_i

end

Function `Model-Update(`$\boldsymbol{\theta}_l^t$`):`

  for *batch* b *of size* $\mathcal{M} \in \mathcal{D}_l$ **do**

    $\xi_b = (x_b, y_b)$ is the data sample pairs in b

    $\boldsymbol{\theta}_{\boldsymbol{l}}^{\boldsymbol{t+\tau}} = \boldsymbol{\theta}_l^t - \alpha_l^t \nabla \mathcal{L}_l(\xi_b; \boldsymbol{\theta}_l^t)$

  end

  return $\boldsymbol{\theta}_l^{t+\tau}$

Function `D2D-Aggregate(`$\boldsymbol{\theta}_k^t,\ n_l$`):`

  Exchange model with 1-hop neighbors n_l with probability d_l

  Perform weighted aggregation for the received models

  $\boldsymbol{\theta}_l^{t+\delta} = \boldsymbol{\theta}_l^{t+\tau} + \sum_{j\in n_l} \eta_j \boldsymbol{\theta}_j^{t+\tau}$

  return $\boldsymbol{\theta}_l^{t+\delta}$

The nodes communicate at the device level using random pairings, resulting in gossip communication. These random node pairs exchange local models during this stage to perform local aggregation yielding $\boldsymbol{\theta}_k^{t+\tau}$ for $k = 1 \ldots N$ network nodes. Subsequently, these gossip-aggregated models are shared with the edge servers for hierarchical aggregation and subsequently cloud aggregation, respectively.

Algorithm 5: Hierarchical-Gossip Federated Learning

Input: Initial Model parameters $\boldsymbol{\theta}_0$, Neighborhood information $\mathcal{G}_k = \{\boldsymbol{N}, \xi_k\}$, Hierarchical servers K with device association

Output: Global model parameters $\boldsymbol{\theta}_{t \to \infty}$

Initialization by global server: $\boldsymbol{\theta}_0$ received by each hierarchical server S_i and broadcast among the connected nodes n_i

for *Node* $k = 1 \ldots N$ ***in parallel*** **do**

  $\boldsymbol{\theta}_k^{t+\tau} \longleftarrow$ *Model Update* $(\boldsymbol{\theta}_k^t)$

  $\boldsymbol{\theta}_k^{t+\delta} \longleftarrow$ *Gossip Exchange* $(\boldsymbol{\theta}_k^{t+\tau})$

end

if *aggregation round* $\%\ f == 0$ **then**

  for *Servers* $S_i \ \forall\ i = 1 \ldots\ldots H$ *(**in parallel*** **do**

    Sample q from the associated nodes

    for $j \in$ *Sampled Nodes* **do**

      Share model parameters $\boldsymbol{\theta}_j^{t+\delta}$ with the server

    end

    Aggregate the received parameters

    $\boldsymbol{\theta}_{S_i}^{t+T} = \boldsymbol{\theta}_{S_i}^{t+\delta} + \sum_{j \in q} w_j \boldsymbol{\theta}_j^{t+\delta}$

  end

  for *Servers* $S_i \ \forall\ i = 1 \ldots\ldots H$ *(**in parallel*** **do**

    Upload model $\boldsymbol{\theta}_{S_i}^{t+T}$ to the global server

  end

  Global Aggregation : $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \sum_{j \in S_i} \eta_j \boldsymbol{\theta}_j^{t+T}$

  Send model back to the nodes through respective Edge Servers S_i

end

Function `Model-Update(`$\boldsymbol{\theta}_l^t$`):`

  for *batch* b *of size* $\mathcal{M} \in \mathcal{D}_l$ **do**

    $\xi_b = (x_b, y_b)$ is the data sample pairs in b

    $\boldsymbol{\theta_l^{l+\tau}} = \boldsymbol{\theta}_l^t - \alpha_l^t \nabla \mathcal{L}_l(\xi_b; \boldsymbol{\theta}_l^t)$

  end

  return $\boldsymbol{\theta}_l^{t+\tau}$

Function `Gossip Aggregate(`$\boldsymbol{\theta}_l^t$`):`

  Perform aggregation handshake with a random node

  Exchange models for aggregation between the pair

  Aggregate received model : $\boldsymbol{\theta}_l^{t+\zeta} = \boldsymbol{\theta}_l^{t+\tau} + \eta_j \boldsymbol{\theta}_j^{t+\tau}$

  return $\boldsymbol{\theta}_l^{t+\zeta}$

5.2.6 Clustered Federated Learning

Clustered FL (CFL) builds on fully decentralized Federated Learning while aiming to reduce communication volume by clustering operations. The nodes in proximal locations are grouped into Clusters formed around a Cluster Head (CH). The devices become a Cluster-Member (CM) by associating with a Cluster Head $\mathcal{C}_i$, itself a device. During the training phase, the device performs local updates of their own models i.e $\boldsymbol{\theta}_k^t$. At the aggregation stage, each CM i shares its local model with the CH i.e $\mathcal{C}_i$, with a probability d_k. Subsequently, all the received models are aggregated to generate the sub-global model $\boldsymbol{\theta}_{\mathcal{C}_i}^t$. The aggregated model is shared with the respective CMs of i^{th} cluster. The process continues until an acceptable threshold is reached. The details of the entire operation are presented in Alg. 6.

Algorithm 6: Clustered Federated Learning

Input: Model parameters θ_k for each node, degree n_k
Output: D2D Aggregated Models
Devices $i = 1.....\boldsymbol{C}$ chosen as Cluster Head (CH) of cluster $\mathcal{C}_i$
for *free nodes* $k = 1.....\boldsymbol{N - C}$ **do**
  Join $\mathcal{C}_i \; \forall \; i \in \boldsymbol{C}$ as a Cluster-Member (CM)
end
for *Cluster Head* $\mathcal{C}_i \forall \; i = 1 \ldots C$ *in parallel* **do**
  Send model to CM $\{p \; : \; p \in \mathcal{C}_i\}$
  for *Nodes* $k \; = \; 1 \; \ldots N$ ***in parallel*** **do**
    $\boldsymbol{\theta}_k^{t+\tau} \longleftarrow$ Model-Update $(\boldsymbol{\theta}_k^t)$
  end
  Receive models from CM $\{p \; : \; p \in \mathcal{C}_i\}$ with probability d_p
  Perform weighted aggregation of received models
  $\boldsymbol{\theta}_{\mathcal{C}_i}^{t+1} = \boldsymbol{\theta}_{\mathcal{C}_i}^{t+\tau} + \sum\limits_{j \in C_i} \eta_j \boldsymbol{\theta}_j^{t+\tau}$
end
Function `Model-Update(`$\boldsymbol{\theta}_l^t$`):`
  for *batch* b *of size* $\mathcal{M} \in \mathcal{D}_l$ **do**
    $\xi_b = (x_b, y_b)$ is the data sample pairs in b
    $\boldsymbol{\theta_l^{t+\tau}} = \boldsymbol{\theta}_l^t - \alpha_l^t \nabla \mathcal{L}_l(\xi_b; \boldsymbol{\theta}_l^t)$
  end
  return $\boldsymbol{\theta}_l^{t+\tau}$

5.2.7 Clustered Device-to-Device Federated Learning (CD2DFL)

Clustered D2DFL (CD2DFL) builds on fully decentralized Federated Learning while aiming to reduce communication volume by clustering operations. The devices join a cluster by associating with a Cluster Head $(CH)_i$, itself a device. The Cluster-Members (CMs) in this case, continue local updates and D2D aggregating models over their respective neighborhoods. After a number of local aggregation rounds, each CH orchestrates a cluster-level aggregation wherein the CMs share their respective models with the CH. Once aggregated, the CMs receive the aggregated model $\boldsymbol{\theta}_{\mathcal{C}_i}^{t+1}$ from the CH and continue with the D2DFL. Alg. 7 outlines the overall procedure for CD2DFL.

Algorithm 7: Clustered Device-to-Device Federated Learning

Input: Model parameters θ_k for each node, degree n_k
Output: D2D Aggregated Models
Devices $i = 1.....\boldsymbol{C}$ chosen as Cluster Head (CH) of cluster $\mathcal{C}_i$
for *free nodes* $k = 1.....\boldsymbol{N - C}$ **do**
  Join $\mathcal{C}_i \ \forall \ i \in \boldsymbol{C}$ as a Cluster-Member (CM)
end
for *Cluster Head* $\mathcal{C}_i \forall \ i = 1 \dots C$ *in parallel* **do**
  Send model to CM $\{p \ : \ p \in \mathcal{C}_i\}$
  for *Nodes* $k \ = \ 1 \ \dots N$ ***in parallel*** **do**
    $\boldsymbol{\theta}_k^{t+\tau} \longleftarrow$ Model-Update $(\boldsymbol{\theta}_k^t)$
    $\boldsymbol{\theta}_k^{t+T} \longleftarrow$ D2D-Aggregate $(\boldsymbol{\theta}_k^{t+\tau}, n_k)$
  end
  Receive models from CM $\{p \ : \ p \in \mathcal{C}_i\}$ with probability d_p
  Perform weighted aggregation
  $\boldsymbol{\theta}_{\mathcal{C}_i}^{t+1} = \boldsymbol{\theta}_{\mathcal{C}_i}^{t+T} + \sum_{j \in C_i} \eta_j \boldsymbol{\theta}_j^{t+T}$
end
Function `Model-Update(`$\boldsymbol{\theta}_l^t$`):`
  for *batch* b *of size* $\mathcal{M} \in \mathcal{D}_l$ **do**
    $\xi_b = (x_b, y_b)$ is the data sample pairs in b
    $\boldsymbol{\theta_l^{t+\tau}} = \boldsymbol{\theta}_l^t - \alpha_l^t \nabla \mathcal{L}_l(\xi_b; \boldsymbol{\theta}_l^t)$
  end
  return $\boldsymbol{\theta}_l^{t+\tau}$
Function `D2D-Aggregate(`$\boldsymbol{\theta}_l^t, \ n_l$`):`
  Exchange models with 1-hop neighbors n_l with probability d_l
  Perform weighted aggregation : $\boldsymbol{\theta}_l^{t+\delta} = \boldsymbol{\theta}_l^{t+\tau} + \sum_{j \in n_l} \eta_j \boldsymbol{\theta}_j^{t+\tau}$
  return $\boldsymbol{\theta}_l^{t+\delta}$

5.2.8 Inter-Cluster Device-to-Device Federated Learning

Clustered FL in the presence of D2D interactions significantly improves local cooperation. In this regard, the device's ability to interact within their neighborhood and subsequent consolidation at Cluster Head (CH) results in greater participation in the overall aggregation scheme. In order to further expand this functionality, this algorithm allows the Cluster Heads to communicate with the other CH's using a gossip mechanism. While the rest of the learning and aggregation process remains similar to CD2DFL, the aggregation at the cluster head level is followed by the CH exchanging the locally aggregated cluster model $\boldsymbol{\theta}_{\mathcal{C}_i}^t$ with other CH $\{\mathcal{C}_j | i \neq j\}$ using

randomized gossip. Each CH $\mathcal{C}_i$ is allowed a limited number of gossip steps before sharing the gossip-aggregated model $\boldsymbol{\theta}_{\mathcal{C}_i}^{t+\tau}$ with its respective CMs. As shown in Alg. 8, this allows both of communication to remain restricted to proximal locations while still enabling models learned in farther clusters to be acquired.

5.2.9 Centralized to Decentralized Spectrum

The FedAvg is on the centralized end of the FL algorithms spectrum, whereas GFL, D2DFL and CD2DFL are on the decentralized end. HFL, HD2DFL and HGFL avail a mix of centralized and decentralized operations. iCFL and iCD2DFL, despite being fully decentralized, mimic hierarchical behavior at the device level. The boundaries between centralized and decentralized operations FL algorithms thus remain fluid, allowing present and future algorithms to benefit from both types of interactions.

Algorithm 8: Inter-Cluster Device-to-Device Federated Learning (iCD2DFL)

Input: Model parameters θ_k for each node, degree n_k
Output: D2D Aggregated Models
Devices $i = 1.....\boldsymbol{C}$ chosen as Cluster Head (CH) of cluster $\mathcal{C}_i$
for *free nodes* $k = 1.....\boldsymbol{N - C}$ **do**
  Join $\mathcal{C}_i \ \forall \ i \in \boldsymbol{C}$ as a Cluster-Member (CM)
end
for *Cluster Head* $\mathcal{C}_i \forall \ i = 1 \dots C$ *in parallel* **do**
  Send model to CM $\{p \ : \ p \in \mathcal{C}_i\}$
  for *Nodes* $k \ = \ 1 \ \dots N$ ***in parallel*** **do**
    $\boldsymbol{\theta}_k^{t+\tau} \longleftarrow$ Model-Update $(\boldsymbol{\theta}_k^t)$
    $\boldsymbol{\theta}_k^{t+T} \longleftarrow$ D2D-Aggregate $(\boldsymbol{\theta}_k^{t+\tau}, n_k)$
    Share model with CH $\mathcal{C}_i$ with probability d_k
  end
  Aggregate models received from CM $\{p \ : \ p \in \mathcal{C}_i\}$
  $\boldsymbol{\theta}_{\mathcal{C}_i}^{t+\beta} = \boldsymbol{\theta}_{\mathcal{C}_i}^{t+T} + \sum_{j \in \mathcal{C}_i} \eta_j \boldsymbol{\theta}_j^{t+T}$
  $\boldsymbol{\theta}_{\mathcal{C}_i}^{t+1} \longleftarrow$ *Gossip Exchange* $(\boldsymbol{\theta}_{\mathcal{C}_i}^{t+1})$
end
Function `Model-Update(`$\boldsymbol{\theta}_s^t$`):`
  for *batch* b *of size* $\mathcal{M} \in \mathcal{D}_s$ **do**
    $\xi_b = (x_b, y_b)$ is the data sample pairs in b
    $\boldsymbol{\theta_s^{t+\tau}} = \boldsymbol{\theta}_s^t - \alpha_s^t \nabla \mathcal{L}_s(\xi_b; \boldsymbol{\theta}_s^t)$
  end
  return $\boldsymbol{\theta}_k^{t+\tau}$
Function `D2D Aggregate(`$\boldsymbol{\theta}_l^t, \ n_l$`):`
  Exchange models with 1-hop neighbors n_l with probability d_l
  Perform weighted aggregation : $\boldsymbol{\theta}_l^{t+\delta} = \boldsymbol{\theta}_l^{t+\tau} + \sum_{j \in n_l} \eta_j \boldsymbol{\theta}_j^{t+\tau}$
  return $\boldsymbol{\theta}_l^{t+\delta}$
Function `Gossip Aggregate(`$\boldsymbol{\theta}_{\mathcal{C}}^t$`):`
  for $g = 1 \dots\dots \zeta$ *gossip rounds* **do**
    Perform aggregation handshake with a randomly selected $\mathcal{C}_j$
    Exchange and aggregate models between the pair
    $\boldsymbol{\theta}_i^{t+1} = \boldsymbol{\theta}_i^{t+\tau} + \eta_j \boldsymbol{\theta}_j^{t+\tau}$
  end
  return $\boldsymbol{\theta}_i^{t+1}$

5.3 Experiments and Evaluation

A uniform evaluation strategy was devised to conduct an objective assessment of the various FL algorithms. The experiments performed in this work use the MNIST [99]

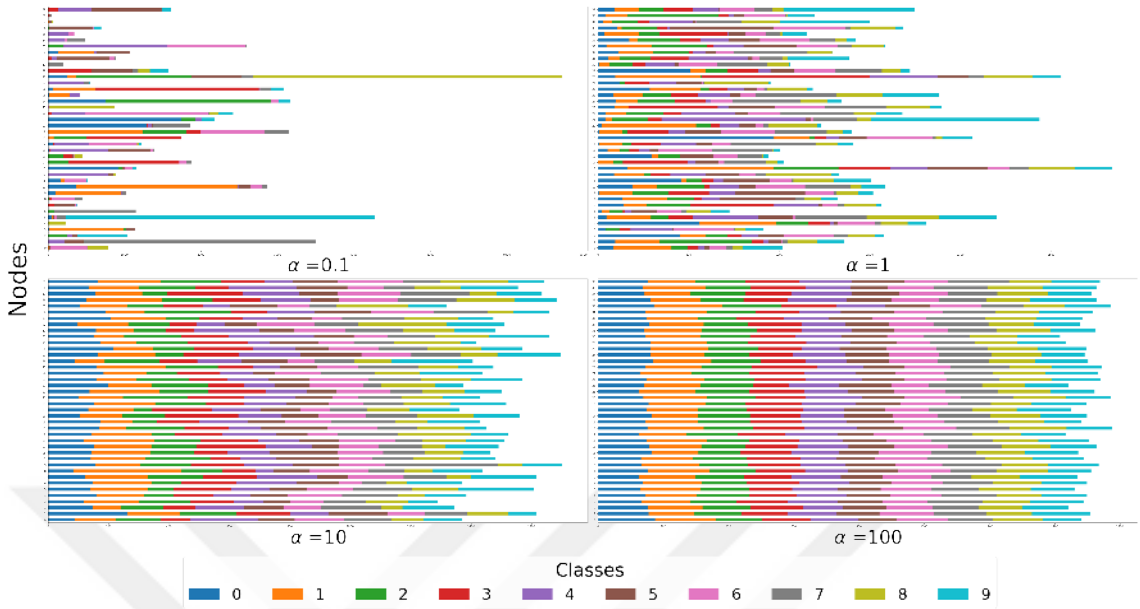


Figure 5.2: Synthetic data distributions for MNIST dataset across 40 nodes derived from Dirichlet Distribution for different α values.

and FashionMNIST datasets [100]. A non-IID data distribution remains of primary interest in this work. The experimentation included subjecting the FL algorithms to ideal and noisy communication, probabilities of participation, data distribution and aggregation frequency. Additionally, these algorithms were also subjected to Few-Shot Learning details of which have been shared subsequently.

Non-IID Distribution The non-IID data partitions are generated using the Dirichlet Distribution [101] parametrized by its concentration parameter α. The value of α controls the degree of non-IID sampling spread across the clients with lower values, resulting in higher imbalance. Lower values of α result in more skewed datasets as shown in Fig. 5.2. A value of α between 1 and 10 results in a typically non-IID data distribution, whereas lower values of $\alpha < 1$ result in extreme non-IID distributions. On the other end, values of $\alpha > 100$ result in increasingly IID distributions. The evaluation in this work uses $\alpha = 0.1$ and $\alpha = 1.0$ for generating highly non-IID distributions.

Extremely Skewed non-IID Distribution This work also employs an extremely skewed 2-class and 3-class non-IID cases where each node is trained using data samples from two and three classes only, respectively. Each node is randomly assigned the determined number of classes and the individual class data is grouped into *shards* of 50 images each. The nodes are then assigned a randomly chosen number of shards sampled from their respective classes.

Neural Network and Development Environment The DNNs used for evaluations based on MNIST and Fashion MNIST datasets are comprised of two 2D Convolutional blocks followed by a Dropout and two linear layers. The modes are instantiated with the same weights while training is conducted using the SGD optimizer with a learning rate $\eta = 0.01$. The entire framework has been developed in Python, whereas the learning algorithms were implemented using PyTorch. The neural network trained for MNIST and Fashion MNIST contains approximately 2 million parameters Both neural networks use Rectified Linear Unit (ReLU) activations with all layers except the final layer, which uses Log-Softmax activation to generate the class probabilities.

Training Regime The training environment assumes a network of 40 nodes, all initiated with similar weights. Our framework allows for each node to control the number of local updates as well as the learning rate and other hyperparameters. However, for this work, only variation in number of training epochs has been conducted across the devices.

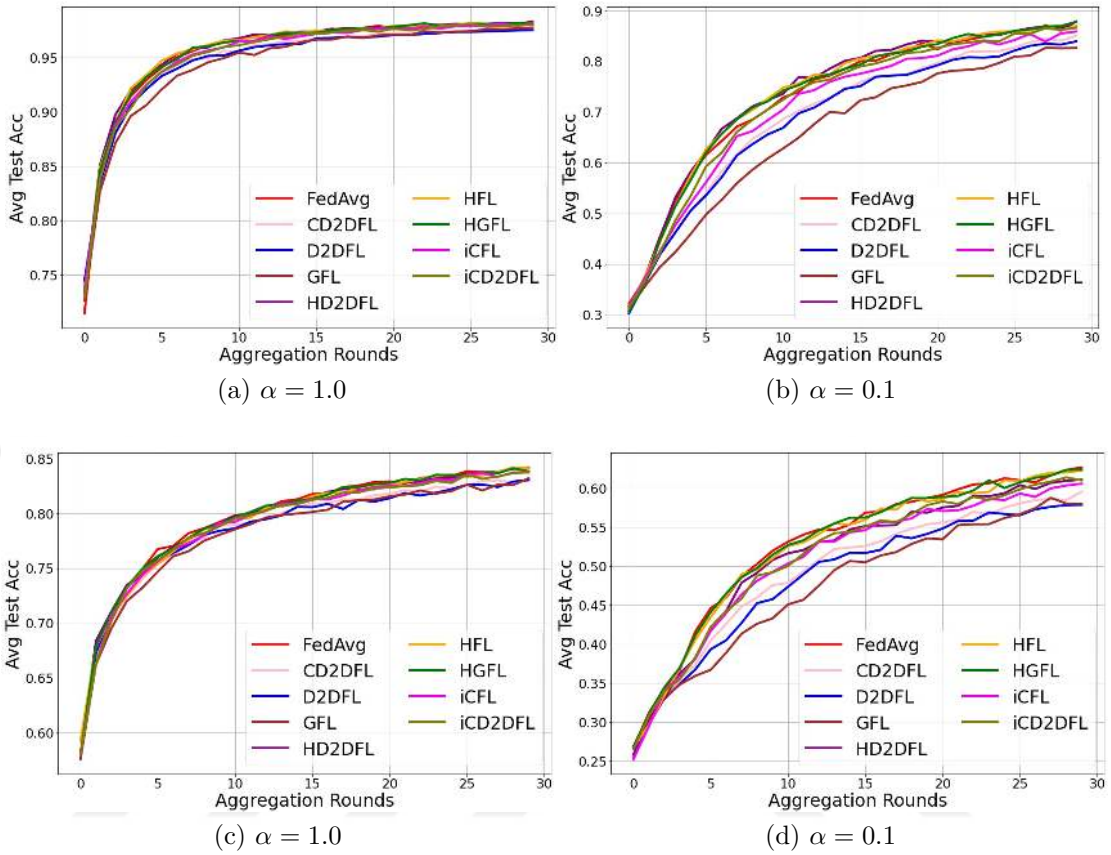
Operating Characteristics The network topology generated for the experiments consists of $\boldsymbol{N} = 40$ nodes that are divided into $\mathcal{C} = 7$ clusters. In order to ensure *reachability*, $\gamma = 0.95$ and $\upsilon = 0.1$ have been used. The former ensures that each device has a direct link to 95% of the devices from the same cluster. Additionally, 10% of each devices' neighbors lie outside the cluster. This ensures that the groups have sufficiently dense intra-device connections as well as having links with nodes outside the current group. This resulting network graph remains *reachable*.

The experiments conducted during the course of this work encompass two different participation scenarios: (a) Upstream participation with probability p_k and (b) Neighborhood participation with probability d_k The participation probabilities of $p_k = \{0.9, 0.6\}$ and $d_k = \{0.9, 0.6\}$ have been used during the course of this work. The two conditions have also been tested jointly to mimic scenarios of frequent stragglers and communication limitations.The channel conditions for both D2D and D2E communication have been subjected to zero-mean Gaussian noise $\mathcal{N}(\mu = 0, \sigma^2)$[1] with $\sigma^2 = 0.01, 0.0025, 0.0001$.

Few-Shot Federated Learning One-Shot FL [102] suggests training local models to completion before sharing with the global server. The resulting mechanism enables significant reduction in communication frequency. Extending the idea, this work subjects the FL algorithms to Few-Shot FL. The nodes perform significantly more local updates before sharing their models. The scheme works with significantly reduced aggregation rounds and communication at both device and edge level. However, it also results in higher client drift.

Asynchronous Operation FL envisions autonomy of operation at the device level. By extension, this implies that not all of the devices perform same number of update operations during the training phase. This may be caused by device being active (or inactive) or a straggler. This work assumes that the devices are required to work with a deadline after which they are required to share their respective models for aggregation. This deadline-based asynchronous aggregation behavior is replicated for the FL algorithms by allowing each device to undergo a range-limited random number of training epochs.

[1] $\boldsymbol{N}(0, \sigma^2)$ noise instead of more detailed fading models and frequency-selective channel impairments has been considered since the primary aim remains to gauge the impact of noisy updates being used for aggregation.



(a) $\alpha = 1.0$ (b) $\alpha = 0.1$

(c) $\alpha = 1.0$ (d) $\alpha = 0.1$

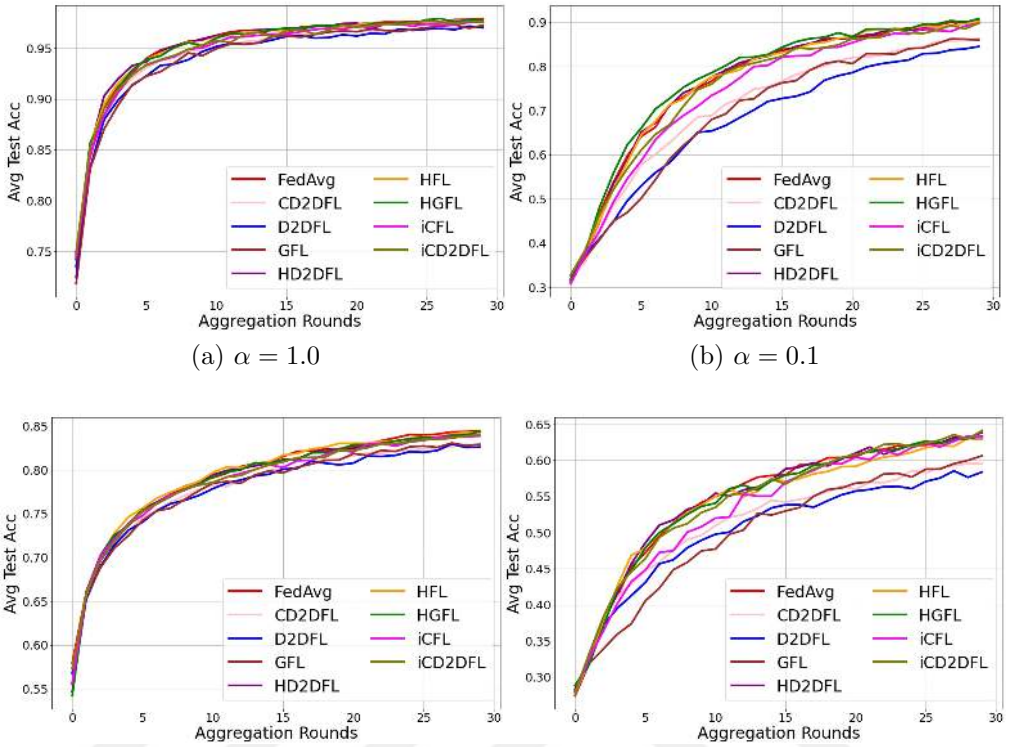
Figure 5.3: Average Test Accuracy for (a)-(b) MNIST and (c)-(d) FashionMNIST for maximal participation $p_k, d_k = 0.9$ with asynchronous communication for Non-IID distributions with Dirichlet parameter $\alpha = 10$ and $\alpha = 0.1$

5.4 Performance Analysis

The following section encapsulates the findings for each of the mentioned simulation conditions. The final test accuracies of each algorithm, averaged over three runs and over all the nodes have been reported in Figures 5.3-5.7. Since one of the aims of decentralized FL is to reduce the communication cost, we also present the number of messages for each link type for each algorithm in Fig. 5.8.

5.4.1 FL with Maximal Participation and Ideal Communication

The ideal scenario assumes noise-free communication and a 90% device participation probability i.e. $p_k, d_k > 0.9$. As expected, greater non-IID distribution across



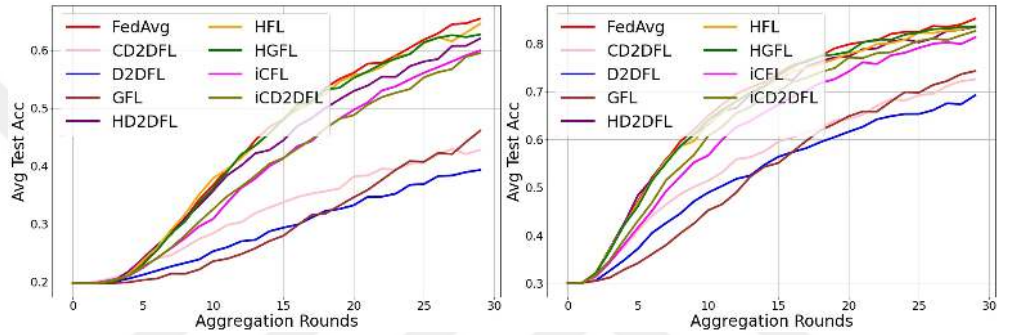
(a) $\alpha = 1.0$ (b) $\alpha = 0.1$

(c) $\alpha = 1.0$ (d) $\alpha = 0.1$

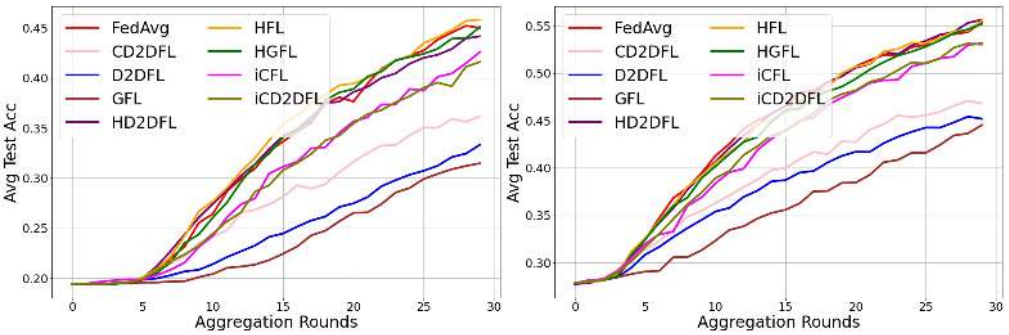
Figure 5.4: Average Test Accuracy for (a)-(b) MNIST and (c)-(d) FashionMNIST for limited participation $p_k, d_k = 0.6$ with asynchronous communication for Non-IID distributions with Dirichlet parameter $\alpha = 1.0$ and $\alpha = 0.1$

devices adversely affects the convergence rate and performance. The impact is reflected across all algorithms. However, results for FashionMNIST depict more loss in performance. This indicates a strong correlation between the performance degradation jointly due to a DNN's ability to learn and greater non-IID levels. The results in Fig. 5.3 indicate that decentralized FL performs on par with their more centralized counterparts. The results also show that as distributions become more non-IID, GFL, D2DFL and CD2DFL lag in learning performance when compared to the other algorithms. However, GFL also incurs the least communication volume among all the considered FL schemes since it only uses the least costly communication link as shown in Fig. 5.8.

The decentralized algorithms also indicate a slightly slower convergence rate ex-



(a) 2-class non-IID (b) 3-class non-IID



(c) 2-class non-IID (d) 3-class non-IID

Figure 5.5: Average Test Accuracy for (a)-(b) MNIST and (c)-(d) FashionMNIST for limited participation $p_k, d_k = 0.6$ with extremely skewed (2-class and 3-class) Non-IID distributions.

acerbated as the training progresses. However, clustered operations, particularly iCD2DFL and iCFL indicate better convergence and consistently perform better than D2DFL and GFL. This indicates that inter-cluster cooperation orchestrated by these algorithms helps in the learning performance and may help in arresting overtraining even in the absence of global information. This is particularly evident from iCFL where D2D interaction at the device level is absent.

5.4.2 FL with Limited Device Participation

The results in Fig. 5.4 present the results with reduced device participation. Less number of devices joining an aggregation round may be caused due to stragglers or busy devices. All the algorithms suffer from convergence issues as the device participation is decreased. This implies that data distribution and device participation have confounding effects on the learning performance of these algorithms. The difference in performance of the D2DFL, GFL and CD2DFL becomes more pronounced than the rest as the training proceeds. The performance suffers clearly from overtraining compounded by lower number of devices as well highly non-IID data. The convergence rate of these three algorithms slows more evidently than the others as the training proceeds. On the other hand, inter-cluster operations allow iCFL and iCD2DF to perform better than the others even with 60% device participation.

The results from Fig. 5.4 are reinforced by testing the FL algorithms under limited device participation and extremely skewed data distribution as shown in Fig. 5.5. When subjected to 2-class and 3-class non-IID data distributions, the localized operation in D2DFL, GFL and CD2DFL suffers more than the others. Lesser number of classes result in almost 20% performance difference between these algorithms and the rest as indicated in Fig. 5.5(a) and (c). The performance of iCFL and iCD2DFL, however, remains remarkably robust even in the presence of these extremely adverse conditions. Both algorithms, supported by inter-CH operation, perform within 5% of the centralized and hierarchical algorithms. The convergence rate of these algorithms retains its trajectory, while other decentralized algorithms show greater divergence as the training proceeds. It may, therefore, be inferred

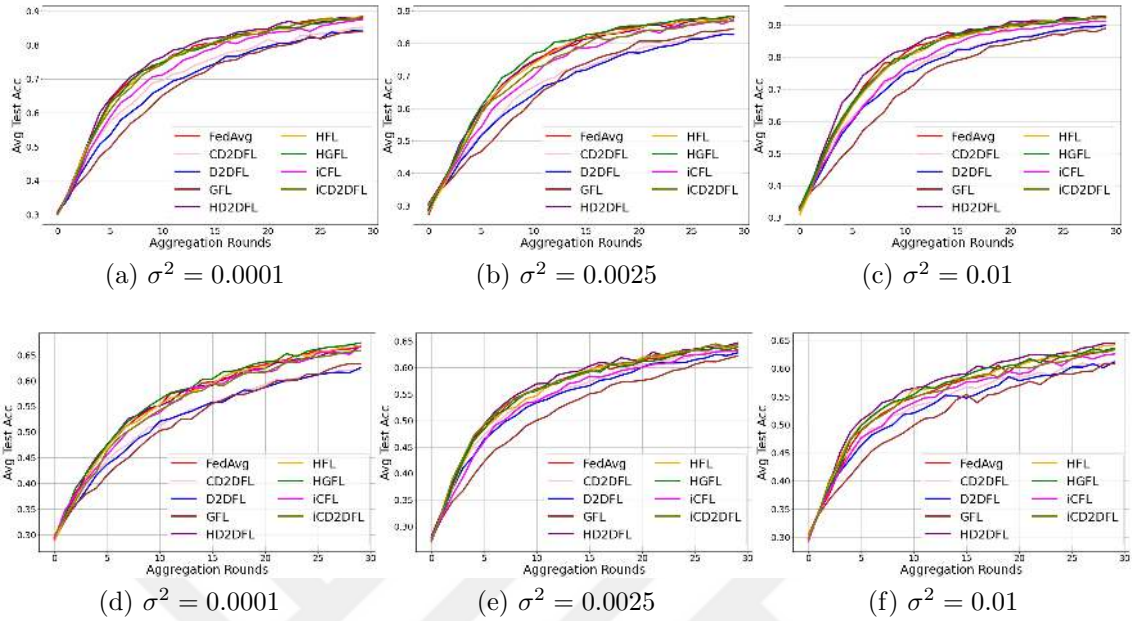
that gossip operations by the CHs partially offset the degradation caused by the lesser device participation and extremely non-IID data and acts akin to DropOut operations in a DNN. Table. 5.1 provides an accuracy-based performance overview of all the algorithms both with maximal and limited participation conditions.

Algorithm	$p_k = 0.6, d_k = 0.6$		$p_k = 0.9, d_k = 0.9$	
	$\alpha = 1.0$	$\alpha = 0.1$	$\alpha = 1.0$	$\alpha = 0.1$
FedAvg	0.9794	0.8736	0.9786	0.8810
D2DFL	0.9758	0.8454	0.9752	0.8849
HFL	0.9805*	0.9022*	0.9808	0.9257*
HD2DFL	0.9798*	0.8980	0.9831*	0.9189
GFL	0.8744	0.3824	0.9753	0.8398
HGFL	0.9801*	0.9015*	0.9800	0.9241
iCFL	0.9762	0.8972	0.9791	0.9142
CD2DFL	0.9694	0.8651	0.9730	0.8855
iCD2DFL	0.9803*	0.8983	0.9800	0.9162

Table 5.1: Accuracy comparison of FL algorithms for two sets of p_k, d_k values and different values of the Dirichlet parameter α (lower values indicate increased non-IID distribution). Highest accuracies within $\Delta = 0.001$ have been marked with asterisk (*)

5.4.3 FL with Noisy Communication

The largest application of Federated Learning is envisioned in wireless spectrum. However, wireless media is subject, among other challenges, to a significant presence of noise. With FL requiring to exchange hundreds of thousands of parameters, presence of noise may cause the convergence to slow down considerably. The FL algorithms during the current research were subjected to the presence of Gaussian Noise $\mathcal{N}(0, \sigma^2)$ both at the device communication level and during D2E and E2C communication. From Fig. 5.6 it can be observed that the addition of noise at the aggregation stage mildly slows down convergence. HGFL progressively shows



(a) $\sigma^2 = 0.0001$ (b) $\sigma^2 = 0.0025$ (c) $\sigma^2 = 0.01$

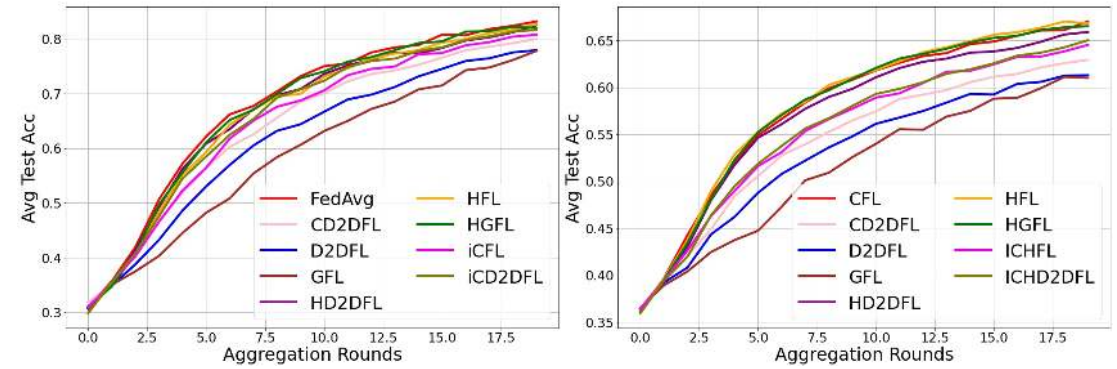
(d) $\sigma^2 = 0.0001$ (e) $\sigma^2 = 0.0025$ (f) $\sigma^2 = 0.01$

Figure 5.6: Average Test Accuracy for (a)-(c) MNIST and (d)-(f) FashionMNIST for $\boldsymbol{N}(0, \sigma^2)$ noisy communication with $p_k, d_k = 0.9$ and Dirichlet parameter $\alpha = 0.1$.

a reduction in performance as σ^2 is increased. However, clustering operations in iCFL and iCD2DFL allow them to closely follow the performance by HD2DFL and HFL. In contrast, GFL shows reduced convergence as compared to D2DFL. However, as training progresses, D2DFL and CD2DFL both also indicate a plateauing convergence rate.

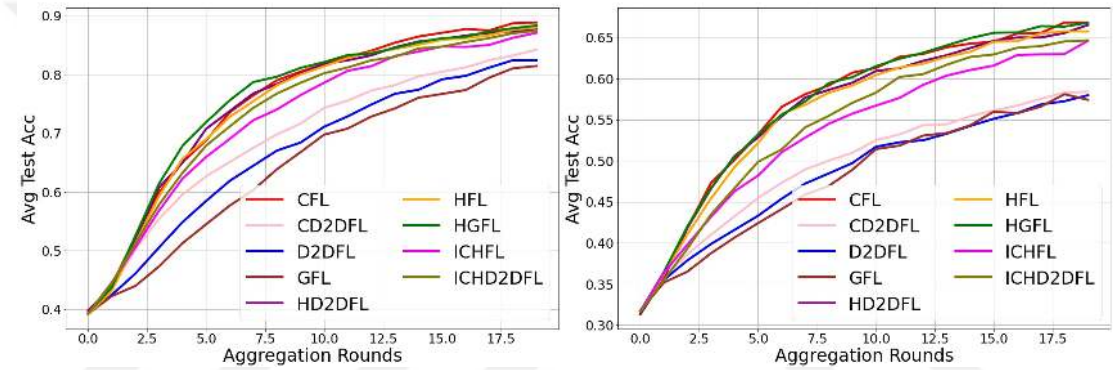
5.4.4 Few Shot Learning

Few-Shot FL offers one way of reducing communication cost when applied properly. The devices undergo multiple local updates before sharing their models for aggregation. Fig. 5.7 shows the result of Few-Shot learning applied under maximal and limited device participation scenario with $p_k, d_k = 0.9$ in Fig. 5.7(a)-(b) and $p_k, d_k = 0.6$ in Fig. 5.7 (c)-(d). The additional difference in performance can be attributed to greater non-IIDness in figures (c) and (d). The impact of client drift on all algorithms causing the learning to plateau earlier than the regular learning regime is evident when its results are compared with the ones shown in Fig. 5.3-5.4.



(a) $\alpha = 0.1$ MNIST

(b) $\alpha = 0.1$ FashionMNIST



(c) $\alpha = 0.1$ MNSIT

(d) $\alpha = 0.1$ FashionMNIST

Figure 5.7: Average Test Accuracy for Few-Shot Learning for MNIST and Fashion-MNIST with (a)-(b) $p_k, d_k = 0.9$ and (c)-(d) $p_k, d_k = 0.6$ with $r = 20$ aggregation rounds and epochs in range $[15, 20]$ with asynchronous communication for Non-IID distributions with Dirichlet parameter $\alpha = 0.1$

Overall, the decentralized variants are impacted worse than the centralized and hierarchical algorithms. Additionally, the performance of D2DFL, GFL and CD2DFL lags further than the rest as training progresses. The performance of iCFL and CD2DFL while showing a relatively bigger drop than when subjected to a normal training scheme, still achieves an accuracy within 5% of the centralized algorithms. The overall performance suggests scenarios where Few-Shot learning may be considered practicable when communication costs become prohibitive, especially or shared updates are infrequent.

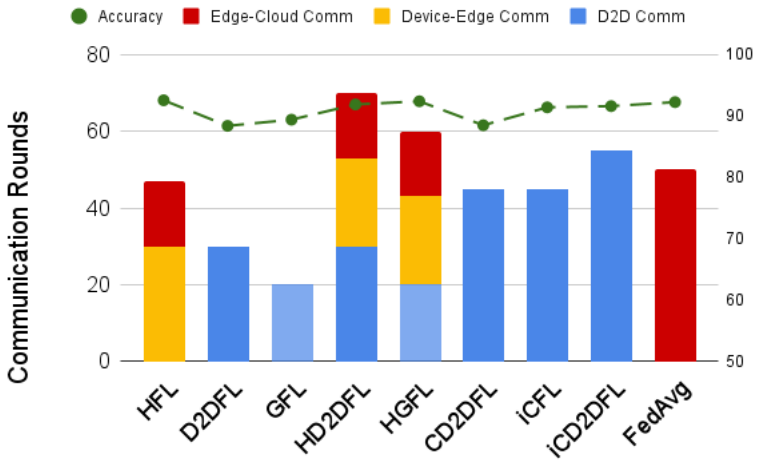


Figure 5.8: Average FL Algorithm Accuracy and per-Node Communication Volume for asynchronous aggregation for 30 rounds with the MNIST dataset.

5.4.5 Communication Cost and Volume

When dealing with different FL algorithms, it is essential that the difference between links employed during the operation is kept in purview. The work in [103] suggest that D2D links like LTE-Direct, WiFi Direct or DSRC are more energy efficient than Device-Edge cellular links like LTE. Furthermore, the same research also indicates that the mentioned D2D links offer reduced latency when compared with cellular communication. E2C links must be be preceded by D2E links for any communication meant for the cloud. It may therefore be inferred that the energy cost and latency of uploading to the cloud may well be more than both D2D and D2E communications. It is with this understanding that Fig.5.8 considers D2D links as least costly and E2C links as most expensive both in terms of energy and latency.

Fig-5.8 shows the communication volume required by different FL algorithms. This is calculated by identifying the number of aggregation rounds required by each of the FL algorithms and are categorized according to the target level (device, edge or cloud). The centralized variants employ more expensive cellular communication for D2E communication. The additional load on network backbone in form of E2C links is bound to significantly increase the cost for FL albeit its improved performance.

Fig.5.8 shows the communication volume in terms of communication rounds for various FL algorithms with the accuracy achieved by the respective algorithms under non-IID data settings. The highest performing algorithms use the most expensive forms of communication. On the contrary, decentralized FL variants indicate slower convergence but also incur less communication cost.

Comparably, decentralized scheme with inter-cluster aggregations show appreciable performance, falling slightly short of the accuracy of HGFL and HD2DFL.

5.5 Conclusion

This work presents a comprehensive comparison of various FL algorithms that have so far been viewed only from the perspective of Centralized FL. The FLAGS framework developed for this purpose enables configuring multiple FL architectures expeditiously. The contrasting highlight of this work is a detailed comparison of major FL algorithms under some of the most dominant challenges in this domain. frameworks. The analysis conducted here indicates that decentralized FL performs comparatively well despite no or minimal upstream communication. Even though noisy communication and irregular participation negatively impact decentralized FL performance, such modes are still capable of achieving acceptable performance thresholds. However, extremely skewed data distributions degrade fully decentralized FL considerably more than centralized ones. The results indicate that FL modes may be used interchangeably depending on the network conditions and the communication costs. As part of future work, we also intend to investigate the performance of the FL algorithms for feature-skewed non-IID distributions. Furthermore, based on the results presented here, the next milestone would be to formulate an adaptive mechanism based on the operating and environment characteristics where the nodes may be suggested to follow a particular FL algorithm for efficient operation.

Chapter 6

IMPLICATIONS OF NODE SELECTION ON DFL UNDER VOLATILE CONDITIONS

6.1 Introduction

Decentralized Federated Learning (DFL) offers a fully distributed alternative to Federated Learning (FL). It enables devices to interact directly with their neighbors without needing external coordination. While DFL may suffer from slower convergence rates, it remains a feasible alternative and benefits from increased local interactions, particularly in clustered settings, as shown in [47].

Recent advances in DFL are based on Device-to-Device (D2D) or Gossip interactions that manage the learning process while restricting the communication between devices. A DFL setting for massive IOT setting is presented in [104]. Their results indicate convergence for DFL both in model and gradient exchange scenarios. The authors of [105] propose a DFL setting with compressed communication for enhanced communication efficiency. They provide a formal convergence analysis for compressed DFL for strongly convex objectives. The work in [106] proposes a DFL setting with unreliable communication using User Datagram Protocol (UDP). The suggested approach indicates that DFL remains robust and can match performance with vanilla DFL.

Research indicates that participation and data heterogeneity in FL lowers the convergence rates. One avenue to improve convergence while operating under non-uniform conditions has been node selection. Node selection relies on identifying and using only beneficial nodes in the aggregation process. The major focus of client

selection has remained in conventional FL settings. The node selection for FL formulated as the secretary selection problem is presented in [107]. Their proposed strategy for client identification employs test accuracy as the performance metric. The work in [108] uses a multi-agent Reinforcement Learning strategy for client selection. The associated state vector includes the probing loss, communication latency, and cost of communication to indicate the utility of a model as well as node accuracy and computation latency to cater to device diversity. Similarly, the authors in [109] propose a deadline-based client selection for a Federated learning setting. The suggested scheme maximizes the number of clients involved in a single aggregation round. To ensure quality updates, the solution aggregates all clients that have completed their tasks within the allocated computation and communication constraints. In [110], the authors model the client loss using a Gaussian Process (GP). Their client selection strategy is based on minimizing the posterior expectation of the GP. Each selection updates the posterior based on the change in loss brought by the previous selection. The research conducted in [111] proposes a Power of Choice client selection strategy based on their observation that biasing the client selection process towards higher local losses increases the convergence rate. The existing global model is sent to a set of sampled clients, and the clients returning the highest local loss are selected for participation in the next training round.

Due to its distributed nature, the convergence rate for DFL is more adversely impacted. Therefore, the research conducted here intends to evaluate the implications of node selection for DFL under more challenging operating conditions [48].

6.2 Node Selection in Federated Learning

The FL paradigm assumes significant heterogeneity among nodes. Device participation remains one of the major heterogeneity aspects since: 1. it must encompass realistic device characteristics 2. it ascertains the overall volume of communication 3. it determines the access to various distributions during the aggregation phase It has, therefore, been a major theme in FL research to ascertain node selection based

on their utility and its impact on convergence. Node selection in centrally orchestrated FL benefits from the presence of a global model and greater visibility into the node characteristics and performance over time. In DFL, on the other hand, the process rests on local information and interaction. While the aim is to restrict upstream communication to make DFL more communication-friendly, local interaction also means slower information diffusion across nodes. Furthermore, client drift due to multiple local steps in an asynchronous setting becomes a major factor. It, therefore, becomes more important to evaluate whether node selection based on local information may offer a feasible solution to accelerate DFL convergence rate.

Node Selection Frameworks: Node selection in FL can be attributed to one of the four main categories: 1. Similarity 2. Reputation 3. Deadline 4. Performance. Similarity-based selection depends upon a similarity measure between the client's neural network with a global reference model and uses nodes with higher similarity. The similarity may be calculated based on L2-Norm-based model divergence, cosine similarity, and a fraction of same-signed parameters. Reputation-based node selection assigns a trustworthiness factor to the participating nodes. This trust factor may be assigned based on maximal participation, training completion within the designated time, or contributing quality updates. Deadline-based selection entails using only those nodes that can share their updated models within a predetermined time frame. Performance-based selection relies instead on the results of the learning operation.

Feasibility of Node Selection in DFL: In a DFL setting, similarity-based systems are challenged by the absence of a meaningful reference. Similarly, reputation-based systems require some trusted intermediary to assign reputation scores and become impracticable in their absence. Reputation-based scoring also assumes an external entity associating trustworthiness scores with each node. Deadline and performance-based selection require no external intervention and may be utilized for DFL. However, the access to participating nodes in a DFL scenario is considerably

lower than in a centralized setting, which may adversely impact the performance. For the mentioned reasons, we focus on the performance-based selection for the default DFL setting in this work.

Hard vs. Soft Selection: Regardless of the specific criterion used for calculating the selection scores, two methods exist for filtering the most beneficial nodes: hard and soft. Hard selection involves selecting the top-k nodes based on their scores for participation in the aggregation round. The threshold to choose 'k' is determined by the overall fraction of neighborhood nodes allowed for aggregation. On the other hand, soft selection aims to promote greater client diversity by using a probabilistic selection process, with high-scoring nodes getting higher probabilities assigned. By guaranteeing a non-zero minimum probability, soft selection ensures that all nodes can be selected for aggregation.

6.3 Node Selection for DFL

The contribution of nodes participating in an FL process becomes skewed as data distribution, participation, and local updates become more heterogeneous. This work intends to identify whether local performance is a good indicator of node performance under DFL settings.

Selection Criteria For assessing the efficacy of node selection in DFL, we look at the following performance-based selection criteria: 1. Training Loss Score 2. Validation accuracy score 3. Joint parametric score

Training Loss Score: One of the commonly used metrics for node selection is the training loss [112] where the score is assigned as:

$$s_i^t = \sqrt{\frac{1}{|D_i|} \sum_{m \in D_i} l_i(k)^2} \tag{6.1}$$

where D_i is the local data, m is the minibatch and l_i is the local learning loss

function.

Validation Accuracy: Another metric used to rank the nodes according to their contributions is the validation accuracy [107] averaged over the number of local updates as:

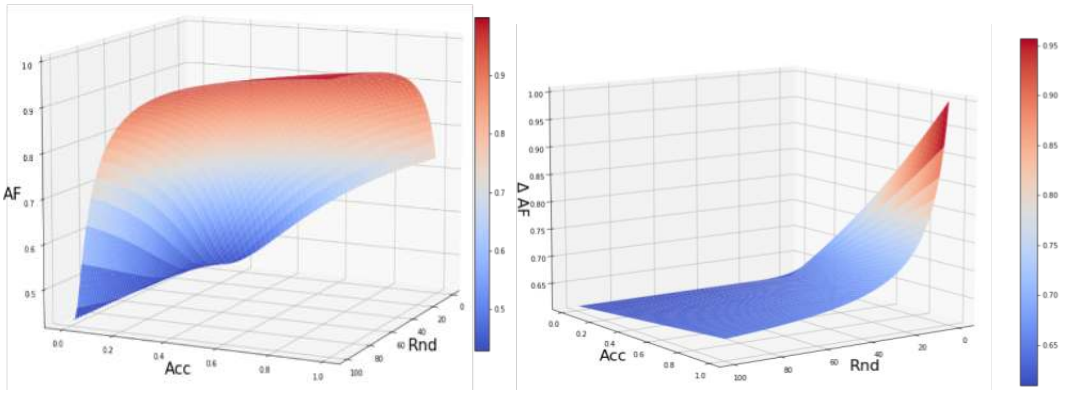
$$s_i^t = \frac{\%Accuracy}{e_i} \tag{6.2}$$

where % *accuracy* is calculated over a common validation dataset. It is worth mentioning that test accuracy has also been used to establish such a utility. However, this work considers using a common validation dataset as the realistic approach.

Parametric Score: The training loss and validation accuracy are indicators of various aspects of training. However, training loss may indicate biased results under heterogeneous and skewed data distributions. Accuracy calculated over a common dataset may offer a more insightful relative picture of the overall performance of a node. Furthermore, the node performance changes as training progresses, achieving a higher accuracy delta during the earlier training phase and the improvement in later rounds coming at a considerably slower pace. The parametric score proposed in this work uses this observation and suggests a time-varying scoring method based on both validation accuracy and its corresponding change. The score is designed to associate more weight with a change in accuracy during the earlier training phases. The parametric score for each node is assigned as:

$$s_i^t = \underbrace{\left(e^{\left(\frac{\alpha \Delta A_i^t}{1+\gamma t}\right)}\right)}_{\Delta \text{ AF}} \cdot \underbrace{\left(0.7 + e^{\left(\frac{-\beta}{(1+\gamma t)(A_i^t+\epsilon)}\right)}\right)}_{\text{AF}}$$

The expression has two components, the 'Accuracy Factor (AF)' and the Change in Accuracy Factor (ΔAF) where ΔA_i^t is the change in accuracy from the previous round, A_i^t is the validation accuracy for the current round, t represents the current communication round. The value of γ controls the shape of the weighing curve. We restrict the value of γ such that $0 \leq \gamma \leq 1$. The factors α and β are the



(a) Accuracy Factor (AF) (b) Δ Accuracy Factor (ΔAF)

Figure 6.1: Surface Plots for Accuracy Factor and Δ Accuracy Factor plotted against rounds and the accuracy and its change, respectively.

scaling factors for cA_i^t and ΔA_i^t., respectively. The ϵ in AF is an offset factor to avoid division by 0 and is chosen to be $0 < \epsilon << 1$. Both terms constituting the expression for s_i^t are normalized against their maximum possible values.

6.4 Experiments and Evaluation

6.4.1 Experimental Setup

The experiments performed in this work use FashionMNIST dataset with non-IID data partitions generated using the Dirichlet Distribution parameterized by its concentration parameter α. Lower values of α result in more imbalanced distributions across the node. This work also employs an extremely skewed 2-class and 3-class non-IID case where each node is trained using data samples from two and three classes, respectively. The neural networks used for evaluations comprise two 2D–Convolutional blocks followed by a Dropout and two linear layers instantiated using the same weights. The training uses the SGD optimizer with a learning rate $\eta = 0.01$. The evaluation uses $N = 60$ nodes operating asynchronously, each with several local updates ranging from $e_{min} = 1$ to $e_{max} = 4$ during each aggregation round. Each node has an associated participation probability ranging from 0.2 to 0.7 for replicating stragglers, incomplete training, and device activity. The diversity

created through both these aspects ensures an asynchronous FL setting, allowing the necessary autonomy to the nodes.

Criterion / Distribution	**$\alpha = 0.1$**			**s = 2**			**s = 3**		
	$p_i = 0.2$	$p_i = 0.3$	$p_i = 0.7$	$p_i = 0.2$	$p_i = 0.3$	$p_i = 0.7$	$p_i = 0.2$	$p_i = 0.3$	$p_i = 0.7$
PS-Exp	0.6	0.686	0.827	0.383	0.481	0.596	0.567	0.657	0.919
PS-Norm	0.549	0.699	0.823	0.305	0.427	0.599	0.538	0.611	0.918
PS-Skew	0.612	0.707	0.831	0.378	0.48	0.601	0.591	0.656	0.916
PS-Lin	0.597	0.696	**0.834**	0.341	0.465	0.603	0.555	0.659	0.922
HS-Acc	0.555	0.674	0.82	0.332	0.416	0.589	0.517	0.652	0.918
HS-Loss	0.493	0.676	0.825	0.333	0.4	0.596	0.491	0.638	0.918
Random	**0.752**	**0.776**	0.831	**0.564**	**0.577**	**0.627**	**0.691**	**0.779**	**0.963**

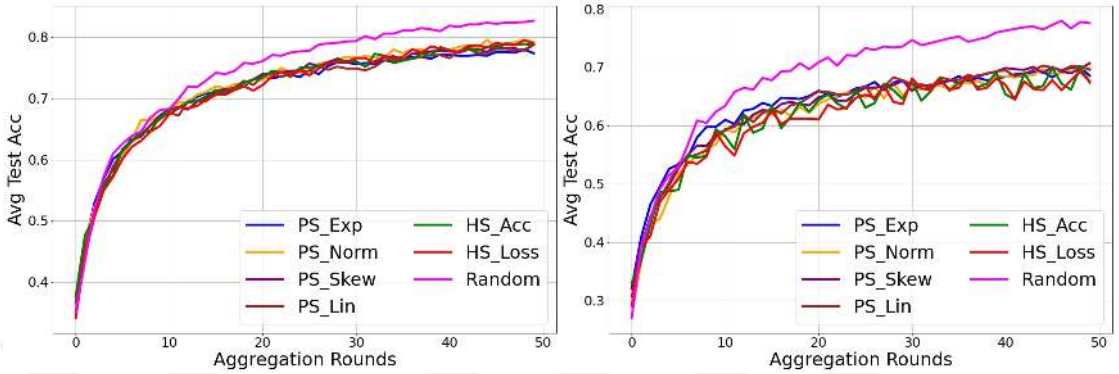
Table 6.1: Results for various selection criteria, data distributions, and participation probabilities (p_i) for FashionMNIST dataset

6.4.2 Selection Criteria

The node selection criteria employ both hard and soft selection for this work. Hard filtering uses the training loss (HS-Loss) and validation accuracy (HS-Acc) score. Node selection using the parametric score (PS) adopts a soft filtering approach. The probabilities in this work are assigned to the nodes as per the following distributions: 1. Exponential Distribution (PS-Exp) 2. Normal distribution (PS-Norm) 3. Skewed distribution (PS-Skew) 4. Linear distribution (PS-Lin) Furthermore, Random Selection (Random) extends stochastic participation by the nodes in all rounds. The participation threshold is $0.2 \leq p_i \leq 0.7$. This work assumes that for each round at t, a maximum of p_i proportion from the n_i one-hop neighbors and share their models within their respective neighborhoods.

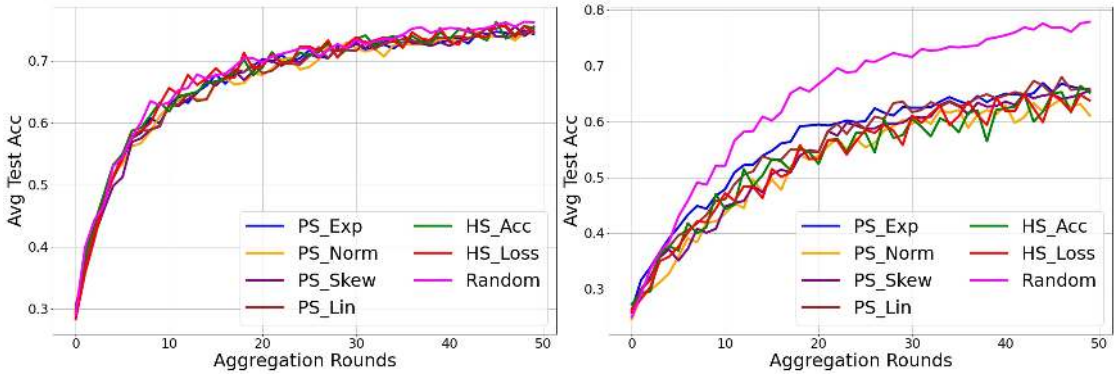
6.4.3 Results and Analysis

The results of node selection criteria in Section-6.3 are depicted in Fig-6.2 and 6.3 and Table-6.1. The results represent various participation levels from $p_i = \{0.7, 0.3, 0.2\}$. The participation levels are associated with three non-IID distributions, $\alpha = 0.1$ and $s = 2$ & 3 for two and 3-class skewed data distributions. It may be observed from the results that 'Random' selection tends to benefit the convergence process more than others, particularly under cases with lower partici-



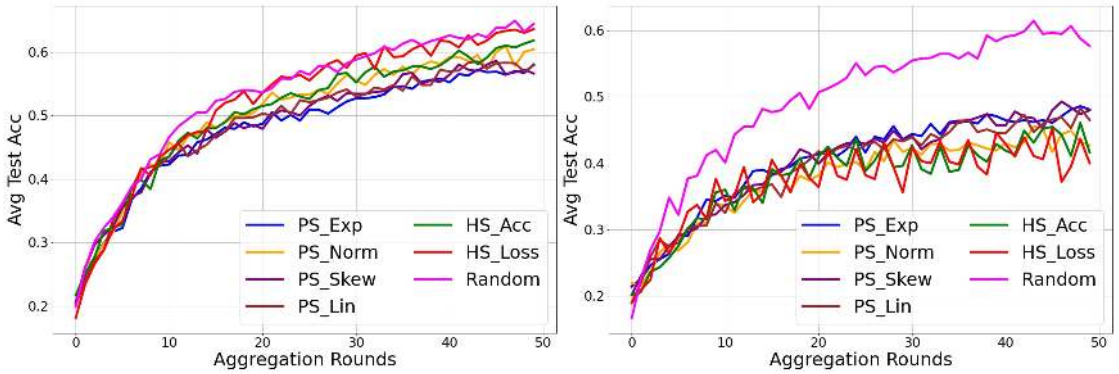
(a) $\alpha = 0.1$ and $p_i = 0.5$

(b) $\alpha = 0.1$ and $p_i = 0.3$



(c) $s = 3$ and $p_i = 0.5$

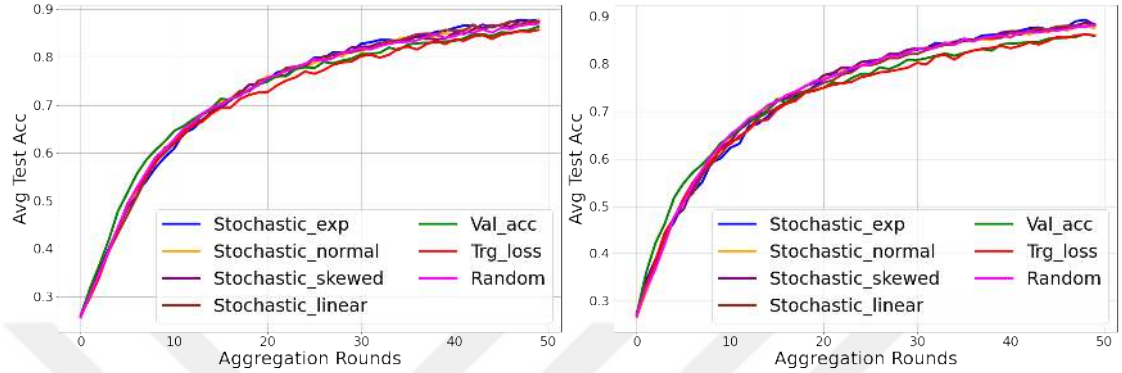
(d) $s = 3$ and $p_i = 0.3$



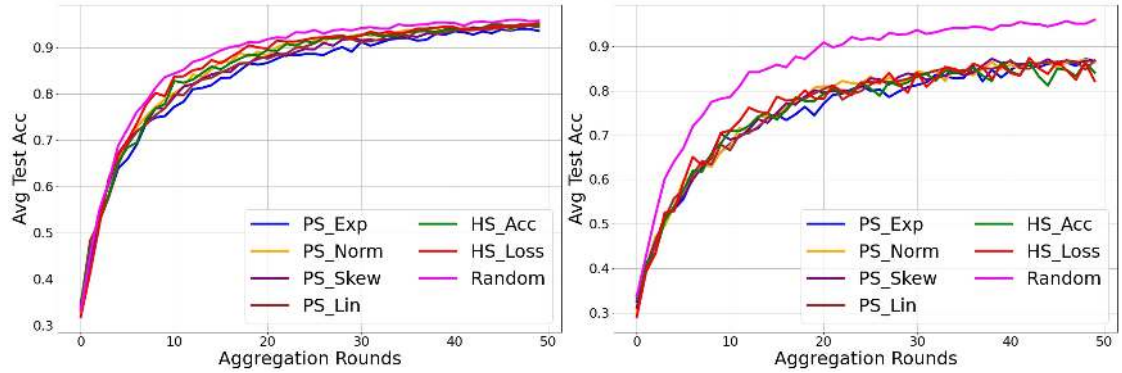
(e) $s = 2$ and $p_i = 0.5$

(f) $s = 2$ and $p_i = 0.3$

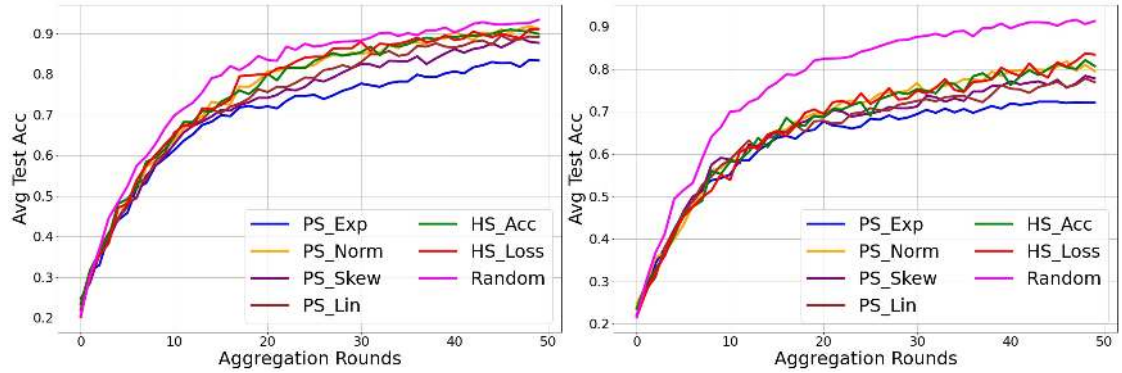
Figure 6.2: Average test accuracy for 50 Communication rounds for non-IID distributed FashionMNIST with $\alpha = 0.1$, $s = 3$ & $s = 2$ and participation threshold as $p_i = 0.5$ & 0.3



(a) $\alpha = 0.1$ and $p_i = 0.5$ (b) $\alpha = 0.1$ and $p_i = 0.3$



(c) $s = 3$ and $p_i = 0.5$ (d) $s = 3$ and $p_i = 0.3$



(e) $s = 2$ and $p_i = 0.5$ (f) $s = 2$ and $p_i = 0.3$

Figure 6.3: Average test accuracy for 50 Communication rounds non-IID data distributions of MNIST for $\alpha = 0.1$, $s = 3$, $s = 2$ and p_i 0.5 & 0.3

pation levels. As the participation levels improve, the convergence rate offered by other selection criteria approaches that of random participation. Furthermore, hard selection methods show higher variance levels than soft selection. This variance shows signs of increasing as participation levels drop. The results indicate that allowing maximal participation under conditions of low participation may indeed be more beneficial than filtering nodes. The performance of random selection points at achieving greater data diversity from among the nodes, especially under cases with increasingly heterogeneous data distribution. This diversity ensures better generalization among all other selection criteria considered in this work. Furthermore, results indicate that parametric scoring with exponential and skewed distribution (PS-Exp and PS-Skew) outperforms the other methods as the distribution and participation levels become more extreme.

6.5 Conclusion

Selection in FL settings has improved convergence and communication efficiency. This work evaluates the implications of using node selection in DFL settings, particularly regarding convergence. Assessments have been made with hard and soft/stochastic selection methods based on performance metrics such as training loss, validation accuracy, and a parameterized variant. Results indicate that the impact of selection bias is more pronounced in volatile environments with limited client participation. Hard selection methods are shown to be more prone to extreme data distributions and node participation. Random selection may be more suited to ensuring various distributions under extreme conditions.Further studies into information diffusion under extreme data distributions and volatile node participation may also help in understanding the progression trends of DFL.

Chapter 7

ASSISTING DECENTRALIZED FEDERATED LEARNING USING MEMORY AND AUGMENTED GRAPHS

7.1 Introduction

As a serverless alternative, the performance of Decentralized Federated Learning (DFL) is strongly impacted by the local connectivity among the nodes. In addition to the graph structure itself, node characteristics are also a major factor in determining the overall performance. Nodes' participation in the learning process, their ability to complete the learning tasks and share their models within its neighborhood are characteristics that tend to alter the underlying connectivity dynamically. The effect of these factors becomes even more significant when considering extreme behavior. In contrast to maximal participation, nodes may also depict extremely uncertain behaviors vis-a-vis participation in the learning process on account of various factors, including local resources, activity status, mobility, lack of incentives, etc. This leads to device volatility in which a significant fraction of devices are unable to participate in various stages of learning. The work presented in Ch-6 evaluates the impact of device volatility in DFL and its impact on the convergence rate. The results indicate that as the volatility increases, local information tends to become insufficient for ascertaining device performance, and thus, random neighbor selection remains the most viable option for node selection under volatile conditions. In this chapter, we investigate alternatives to boost DFL performance for volatile conditions that go beyond 1-hop neighbor node selection.

In Decentralized Federated Learning (DFL), each node shares performs a local stochastic gradient update on its model using its private data. It then shares its

updated model within its 1-hop neighborhood through D2D interaction. The shared updates are used in weighted aggregated locally and successively propagated along the network [37]. The information diffusion across the network is slower as the shared updates are down-weighted during each communication round as the models propagate through the network. While fully and densely connected topologies improve convergence, real-world configurations follow sparse configurations, implying reduced neighborhood sizes and lower convergence rates [38]. The autonomous operation of nodes also impacts the convergence rate negatively since they may choose to skip training/communication rounds altogether. A significant proportion of nodes depicting such uncertain behaviors leads to a volatile participation environment [39], resulting in an unstable availability of neighboring nodes for model exchange. Jointly, both graph sparsity and participation volatility significantly impact the convergence rate in DFL. Furthermore, they aggravate the impact of data heterogeneity as the models become increasingly skewed toward the data distributions of the proportion of available neighboring nodes in an already sparse neighborhood. The impact of sparse topology in DFL has been researched in [105, 113, 114]. However, the focus of volatility recent works has remained in centralized FL [39, 86, 87]. The works on DFL have so far assumed a consistent node behavior. Whereas FL can use global information to muster a larger proportion of available nodes, such is not feasible in DFL scenarios, resulting in time-varying sparse topologies. By extension, the solutions to these challenges must also be local in nature.

Our own work and existing literature have shown that volatile operating conditions and heterogeneous data distribution adversely impact learning performance and convergence of DFL. Relying only on local neighborhood information and interactions exacerbates these issues. We have also seen that curating local neighbors does not offer a significant advantage. This work focuses on DFL operation in a sparse topology under extremely volatile conditions coupled with highly heterogeneous data. As such, we investigate alternative methods to accelerate DFL convergence under harsh conditions by introducing ideas to assist learning, namely

a) limited memory and b) relay operations. With this framing, we first propose an algorithm by re-using models from the previous round as approximations for the absent nodes. We then design a second algorithm around graph augmentation by allowing nodes to relay models. Utilizing the similarities between these algorithms, we propose a joint scheme that allows nodes to utilize stored models both locally and for relay operations. We call this as Assisted DFL where the aforementioned mechanisms assist DFL in volatile conditions. The algorithms are referred to as Memory-Assisted DFL and Augmented Graph-Assisted DFL, whereas the hybrid is titled as Memory and Augmented Graph-Assisted DFL. The proposed algorithms outperform the baseline DFL as extreme scenarios challenge the overall learning operation. Furthermore, the proposed hybrid outperforms the individual algorithms and the baseline while operating locally.

7.2 Memory-Assisted DFL (MA_DFL)

The realistic behavior of nodes in an FL environment depicts heterogeneous participation and training [115]. This stems from several factors, including node activity and mobility, resource and communication heterogeneity, privacy concerns, and local data distributions. Participation volatility in such scenarios arises from a high probability of nodes dropping out from the update or communication phase. The overarching operating conditions subject the nodes to a highly volatile setting, assuming that only a fraction of nodes join a given aggregation round. This implies that participation across neighborhoods remains highly uncertain. In such a scenario, our solutions rely on the nodes' ability to retain the models from the previous round. Each node manages an array of models it had received in the previous round and uses them in conjunction with the models from the current round.

The impact of the absence of models from a significant proportion of neighborhoods may be offset by considering approximations of these models. The models shared by the respective nodes in the previous rounds may be considered as a potential representative of the current ones. However, under asynchronous operating

conditions where the nodes conduct varying levels of training, more recent model updates may be considered as the closest approximations to the current models. It is on the basis of this assumption that we introduce a memory array for the *Memory Assisted DFL*. Each node stores models from the current aggregation round in its own memory array. Due to limited neighborhood sizes and volatile participation, it can be seen that the array size remains limited. In the subsequent round, each node identifies the absent nodes and checks whether their models were shared during the previous round and, therefore, are part of the memory array. Wherever possible, each node utilizes these models as approximations and includes these in the aggregation process.

The sequence of operations in Memory Assisted - Decentralized Federated Learning (MA_DFL) has been provided in Alg-9. Each node i receives models from π_i^t set of nodes for a given round t, where $|\pi_i^t| << |n_i| \ \forall \ t$. The membership of π_i^t for each node $j \in n_i$ is controlled by its respective probability p_{ji} of sharing with a target node i in a Device-to-Device (D2D) operation. Each node in MA_DFL maintains an array $\mathcal{M}_i^{t-1}$ of models from $r \in \pi_i^{t-1}$ nodes it received in the previous $t-1$ aggregation round. At the beginning of the aggregation round t, the device i communicates its updated model $\theta_i^{t+\tau}$ to s_i^t nodes within its neighborhood such that $|s_i^t| << |n_i|$. Assuming D2D links, i communicates stochastically with each node in $j_i^t \in s_i^t$, determined by the parameter p_{ij}^t allowing devices to cooperate independently of others. This allows them the necessary autonomy over their operations as envisioned for a volatile FL setting. Similarly, every node i receives models $\theta_r^{t+\tau}$ from $\pi_i = \{r_i \mid r_i \in n_i\}$ set of nodes each with a probability p_{ri} such that $|\pi_i| << |n_i|$. Once the communication stage is completed, each node i performs weighted aggregation of its own model $\theta_i^{t+\tau}$ with the received models $\theta_r^{t+\tau}$ and the models $\nu^t = \{\theta_k^t \mid k \in (\pi_i^{t-1} \ \setminus \pi_i^t)\}$ implying using models stored $\mathcal{M}_i^{t-1}$ given they have not been received as part of the π_i^t. Finally, the node i updates its memory array $\mathcal{M}_i^t$ with models ${\theta_r}^t$ received from node set π_i^t.

Algorithm 9: Memory Assisted-Decentralized Federated Learning

Input: Nodes $\boldsymbol{N}$ with model parameters θ_i^t, participation probability $p_i \; for \; i = 1 \ldots\ldots N$, aggregation rounds $t = 1 \ldots\ldots T$, memory array for round t $\mathcal{M}_i^t$

Output: Local consensus model $\boldsymbol{\theta}^*$

for *round* $t = 1, 2 \ldots\ldots$ **do**

  for *Nodes* $i = 1 \ldots N$ ***in parallel*** **do**

    for *minibatch* $\xi_m(x_m, y_m)$ *in local data* D_i **do**

      Gradient update: $\boldsymbol{\theta_i^{t+\tau}} = \boldsymbol{\theta}_i^t - \alpha_i^t \nabla \mathcal{L}_i(x_m, y_m; \boldsymbol{\theta}_i^t)$

    end

    Send θ_i^t to $j \in s_i^t \subset n_i$ neighboring nodes with probability p_{ij}^t

    Receive models $\theta_r^{t+\tau} \; \forall \; r \in \pi_i^t$ set of nodes with probabilities p_{ri}^t

    if $\mathcal{M}_i^{t-1} \neq \phi$ **then**

      $\nu_i^t = \{\theta_k^{t-1} \mid k \in (\pi_i^{t-1} \setminus \pi_i^t)\}$

      $\boldsymbol{\theta}_i^{t+1} = \eta_i \boldsymbol{\theta}_i^{t+\tau} + \sum_{r \in \pi_i} \eta_r \boldsymbol{\theta}_r^{t+\tau} + \sum_{k \in \nu_i^t} \eta_k \boldsymbol{\theta}_k^{t-1}$

    else

      $\boldsymbol{\theta}_i^{t+1} = \eta_i \boldsymbol{\theta}_i^{t+\tau} + \sum_{r \in \pi_i} \eta_j \boldsymbol{\theta}_r^{t+\tau}$

    end

  end

  Update $\mathcal{M}_i^t : \mathcal{M}_i^t = \{\theta_r^t | r \in \pi_i^t\}$

end

Full Memory-Assisted DFL (FMA_DFL) The size of the Memory array in MA_DFL has been restricted to retain models only from previous training rounds. The key aspect remains that of asynchronous training, where devices perform various levels of training and the participation remains uncertain. The most comparable work remains in [93] that employs the entire memory array as a difference of models at the node level while supporting asynchronous training. However, the difference is calculated using a global model aggregated at the parameter server, which allows the nodes to benefit from the global distribution. In DFL, the lack of global information and asynchronous operation may lead the nodes to diverse local minima resulting in a slowdown of the convergence rates. However, to justify this hypothesis, we present empirical results for Full Memory-Assisted DFL (FMA_DFL). The main difference between MA_DFL and FMA_DFL is the latter's ability to retain models from all training rounds. The models are updated once they are received; however, they are retained in the memory array until the node participates again.

7.3 Augmented Graph-Assisted DFL (AG_DFL)

Under the conditions of volatility where device participation remains uncertain and low, one of the key aspects remains to increase the participation of the nodes in the learning process, as evident from the results in [48, 88]. Incentivizing mechanisms and relaying operations are two major avenues through which node participation has been attempted to be increased. However, incentive solutions require third parties as purveyors of the rewards, which are currently assumed to be absent from a fully decentralized setting. Relaying models remain an important alternative in this regard. However, the key concern with relaying operations is the associated increase in the overall communication volume and communication resource heterogeneity among the participants. Anticipating these challenges, we propose an Augmented-Graph DFL (AG_DFL) scheme with selective relaying options.

The basic premise of decentralized operations assumes a knowledge of the local neighborhood. This allows the nodes to conduct 1-hop communication during DFL

when sharing their models. Extending this further, we can further use the standard assumption of prior knowledge of the Adjacency Matrix [83]. The same may be assumed in two stages:

1. Nodes establish local neighborhood information

2. Nodes exchange respective neighborhood information with their 1-hop neighbors

These two steps allow nodes to establish unified information on the adjacency matrix and employ it for relay operations. Using the adjacency information, each node i establishes a set of nodes $O_{ij} : O_{ij} = \{m | m \in (n_i \setminus n_j) \; \forall \; i \in \boldsymbol{N}, j \in n_i\}$ i.e., O_{ij} is a set of neighboring nodes of i not part of node $j \in n_i$. Each node establishes sets O_{ij} for each corresponding node j in its neighborhood. The setting follows volatile conditions where each node i gets access to π_i set of nodes from its neighborhood where $\pi_i << n_i$. With fractional participation, a relaying option allows participating nodes to relay one additional model during the current aggregation phase selectively. However, the sequence followed during the communication stage must remain intact as the nodes transmit the models first. Therefore, the models to be relayed must already be available at the transmitting nodes. To facilitate this operation, each node i maintains an array $\mathcal{M}_i^{t-1}$ of the models $\theta_r^{t-1} \; \forall \; r \in \pi_i^{t-1}$ it received in the round $t-1$. At the beginning of the round t, each node i first performs a local update on its model parameters to obtain $\theta_i^{t+\tau}$. Next, when sending its model to node j, node i identifies the set of nodes $C_{ij} = \{p \mid p \in (O_{ij} \cap \pi_i^{t-1})$ i.e C_{ij} is the set nodes from whom node i received models in the round $t-1$ while concurrently not being part of the neighborhood of node j i.e., n_j. When transmitting, the node i randomly selects a model $\theta_q^{t-1} : q \in C_{ij}$ & $\theta_q^{t-1} \in \mathcal{M}_i^{t-1}$. It the transmits the models $\theta_i^{t+\tau}$ and θ_q^{t-1} to the node j. Once the relaying process is completed, all the nodes perform the local averaging process based on the models $\theta_r^{t+\tau}$ received from $r \in \pi_i^t$ nodes to obtain θ_i^{t+1}. Subsequently, each node updates the stored array $\mathcal{M}^t$ with the models received in the current round from nodes in π^t.

The key factor behind choosing q randomly from C_{ij} is that the nodes in O_{ij} do not contribute to θ_j directly in any round. Sending models from q to j when for the underlying graph $\mathcal{G}$, $\mathcal{E}_{jq} \notin \mathcal{E}_{\mathcal{G}}$ is akin to adding a virtual edge between nodes j and q at time t, thereby augmenting the graph. Under conditions of participation volatility and sparse connectivity, such temporary edges allow greater participation and enable access to a wider distribution. The details of the operation of AG_DFL have been provided in Alg-10.

Algorithm 10: Augmented Graph-Assisted Decentralized Federated Learning

Input: Clients $\boldsymbol{N}$ with model parameters θ_i^t and participation probability $p_k \; for \; k = 1 \ldots\ldots N$, aggregation rounds T, Model Array $\mathcal{M}^t$, Adjacency Matrix $W_{\mathcal{G}}$

Output: Local consensus model $\boldsymbol{\theta}_i^*$

for *nodes* $i = 1 \ldots N$ ***in parallel*** **do**
  for *node* $j = 1 \ldots\ldots n_i$ *in neighborhood* **do**
    Establish $O_{ij} : O_{ij} = \{m \mid m \in n_i \; \wedge \; m \notin n_j\}$
  end
end
for *round* $t = 1, 2 \ldots\ldots$ **do**
  for *Nodes* $i \; = \; 1 \; \ldots N$ ***in parallel*** **do**
    Perform local gradient update:
    $\boldsymbol{\theta_i^{t+\tau}} = \boldsymbol{\theta}_i^t - \alpha_i^t \nabla \mathcal{L}_i(x_m, y_m; \boldsymbol{\theta}_i^t)$;
    if $\mathcal{M}_i^{t-1} \neq \phi$ **then**
      Identify $C_{ij} = \{p | p \in (O_{ij} \cap \pi_i^{t-1}) \}$;
      if $C_{ij} \neq \phi$ **then**
        Randomly choose a node q from C_{ij} ;
        Send $\theta_i^{t+\tau}$ and θ_q^{t-1} to $j \in n_i$ with probability p_{ij}^t
    else
      Send θ_i^t to $j \in n_i$ neighboring nodes with probability p_{ij}^t
    end
    Receive models $\theta_r^{t+\tau} \; \forall \; r \in \pi_i^t \subset n_i$ and relayed models $\rho_i^t = \{\theta_k^{t-1} \mid k \in C_{ri}\}$;
    if $\rho_i^t \neq \phi$ **then**
      $\boldsymbol{\theta}_i^{t+1} = \eta_i \boldsymbol{\theta}_i^{t+\tau} + \sum\limits_{r \in \pi_i} \eta_r \boldsymbol{\theta}_r^{t+\tau} + \sum\limits_{k \in \rho_i^t} \eta_k \boldsymbol{\theta}_k^{t-1}$
    else
      $\boldsymbol{\theta}_i^{t+1} = \eta_i \boldsymbol{\theta}_i^{t+\tau} + \sum\limits_{r \in \pi_i} \eta_r \boldsymbol{\theta}_r^{t+\tau}$
    end
  end
  Update $\mathcal{M}_i^t : \mathcal{M}_i^t = \{\theta_r^t \; | r \in \pi_i^t\}$
end

7.4 Memory and Augmented-Graph Assisted DFL (MAG_DFL)

The nodes in both MA_DFL and AG_DFL rely on storing models received in the previous round in a memory array $\mathcal{M}^{t-1}$. It is, therefore, a logical extension that both algorithms be merged into one that allows nodes to employ stored models jointly for local aggregation and relaying simultaneously. The two algorithms add fractional costs to both storage and computing. The memory array $\mathcal{M}^t$ contains models only from the previous round. Assuming each model has a size of d bytes, even if the array were to extend storing for all the rounds, the maximum size of the array, $h(\mathcal{M}^t)$, would be $h_{max} : h(\mathcal{M}^t_{max}) = d \times n_i$ bytes where $|n_i| << \boldsymbol{N}$. Under volatile conditions, when the nodes participating in the aggregation are $\pi_i << n_i$, this indicates $h(\mathcal{M}^t_{\pi i}) << h_{max}$. For the communication part in FL, each node transmits $b = d \times |n_i|$ bytes. However, when sending to $|s_i| << |n_i|$ nodes, the total communication volume for each node in a D2D setting is $b \times |s_i|$. In addition to the given relay mechanism that adds one additional model per each node in s_i, the total communication volume comes at $2b \times |s_i|$. However, since $|s_i| << |n_i|$, the communication volume, including relaying additional model, remains less than the overall volume for $|s_i| < \frac{|n_i|}{2}$. More so, until the neighborhood participation exceeds 50%, the communication volume for relay operation does not exceed the full participation scenario. Further reduction may be achieved if the relay operation is made stochastic.

Given the overall communication and storage cost, Alg-11 presents the *Memory and Augmented Graph-Assisted DFL (MAG_DFL)* algorithm, a hybrid based on MA_DFL and AG_DFL. MAG_DFL algorithm merges the unique steps of both Alg-9 and Alg-10. The key aspect is maintaining a local memory array $\mathcal{M}^t_i$ at each node i and computing the set O_{ij} for each node $j \in n_i$. Once the model update has been completed, each node i randomly selects the model $\theta^{t-1}_q \; for \; q \in C_{ij}$ where C_{ij} is formed by the intersection of π^{t-1}_i and O_{ij}. The models $\theta^{t+\tau}_i$ and θ^{t-1}_q are communicated to the s^t_i set of nodes. Once the models from nodes π_i from the 1-hop neighborhood and ρ_i relayed nodes have been received, the node performs

aggregation. The aggregation step sees the nodes' model and the ones received during the current round and two sets of models from the previous round, the ones stored in local memory and those relayed by its neighboring nodes.

Algorithm 11: Memory and Augmented Graph-Assisted Decentralized Federated Learning

Input: Clients $\boldsymbol{N}$ with model parameters θ_i^t and participation probability p_k *for* $k = 1 \ldots\ldots N$, aggregation rounds $\boldsymbol{R}$, Adjacency Matrix $\boldsymbol{W}$, Model Array $\mathcal{M}_i^t$

Output: The consensus model $\boldsymbol{\theta}^*$

for *nodes* $i = 1 \ldots N$ ***in parallel*** **do**
  for *node* $j = 1 \ldots\ldots n_i$ *in neighborhood* **do**
    Establish $O_{ij} : O_{ij} = \{m \mid m \in n_i \ \wedge \ m \notin n_j\}$
  end
end
for *round* $t = 1 \ldots\ldots$ **do**
  for *Nodes* $i \ = \ 1 \ \ldots N$ ***in parallel*** **do**
    Perform stochastic gradient update
    $\boldsymbol{\theta_i^{t+\tau}} = \boldsymbol{\theta}_i^t - \alpha_i^t \nabla \mathcal{L}_i(x_m, y_m; \boldsymbol{\theta}_i^t)$
    if $\mathcal{M}_i^{t-1} \neq \phi$ **then**
      Identify $C_{ij} = \{p | p \in (O_{ij} \cap \pi_i^{t-1})\ \}$
      if $C_{ij} \neq \phi$ **then**
        Randomly choose a node q from C_{ij}
        Send $\theta_i^{t+\tau}$ and θ_q^{t-1} to $j \in n_i$ with probability p_{ij}^t
      else
        Send θ_i^t to $j \in n_i$ neighboring nodes with probability p_{ij}^t
      end
    Receive models $\theta_r^{t+\tau}\ \forall\ r \in \pi_i^t \subset n_i$ and relayed models $\rho_i^t = \{\theta_k^{t-1} \mid k \in C_{ri}\}$
    Ascertain $\nu_i^t : \nu_i^t = \{\theta_k^t \mid k \in (\pi_i^{t-1} \ \setminus \pi_i^t)\}$
    if $\rho_i^t \neq \phi$ *and* $\nu_i^t \neq \phi$ **then**
      $\boldsymbol{\theta}_i^{t+1} = \eta_i \boldsymbol{\theta}_i^{t+\tau} + \sum\limits_{r \in \pi_i} \eta_r \boldsymbol{\theta}_r^{t+\tau} + \sum\limits_{k \in \rho_i^t} \eta_k \boldsymbol{\theta}_k^{t-1} + \sum\limits_{m \in \nu_i^t} \eta_m \boldsymbol{\theta}_m^{t-1}$
    else
      Given $\rho_i^t = \phi \rightarrow \boldsymbol{\theta}_i^{t+1} = \eta_i \boldsymbol{\theta}_i^{t+\tau} + \sum\limits_{r \in \pi_i} \eta_r \boldsymbol{\theta}_r^{t+\tau} + \sum\limits_{m \in \nu_i^t} \eta_m \boldsymbol{\theta}_m^{t-1}$
      Given $\nu_i^t = \phi \rightarrow \boldsymbol{\theta}_i^{t+1} = \eta_i \boldsymbol{\theta}_i^{t+\tau} + \sum\limits_{r \in \pi_i} \eta_r \boldsymbol{\theta}_r^{t+\tau} + \sum\limits_{k \in \rho_i^t} \eta_k \boldsymbol{\theta}_k^{t-1}$
    end
    else
      $\boldsymbol{\theta}_i^{t+1} = \eta_i \boldsymbol{\theta}_i^{t+\tau} + \sum\limits_{r \in \pi_i} \eta_j \boldsymbol{\theta}_r^{t+\tau}$
    end
  end
  Update $\mathcal{M}_i^t : \mathcal{M}_i^t = \{\theta_r^t \mid r \in \pi_i^t\}$
end

7.5 Convergence Guarantees

The local objective for each node is $\mathcal{L}_i$, and l_i is the loss function that depends on the nature of the problem e.g. cross entropy etc. Furthermore, the global data distribution is represented as $\mathcal{D}$ and the local distribution is ξ_i. We adopt the following assumptions for proving convergence in our case, which are fairly standard in works on decentralized learning [90, 93, 116]:

Assumption 1 (L-Smoothness) For each $i \in [n],\ \mathcal{L}_i(\theta_i, \xi_i) : \mathbb{R}^d$ is differentiable for $\xi_i \in supp(\mathcal{D})$ such that there exists a constant $L \geq 0$ such that for each $\theta_i,\ \theta_j \in \mathbb{R}^d$:

$$||\nabla \mathcal{L}_i(\theta_i, \xi_i) - \nabla \mathcal{L}_i(\theta_j, \xi_i)|| \leq L||\theta_i - \theta_j||$$

Assumption-2: Bounded Variance: There exists a constant $\overline{\sigma}$, such that for all $\theta \in \mathbb{R}^d,\ i \in [n]$,

$$E_\xi||\nabla \mathcal{L}_i(\theta_i, \xi_i) - \nabla l_i(\theta_i)||^2 \leq \overline{\sigma}^2$$

Assumption -3 : $l_1, l_2 \ldots\ldots l_N$ are all strongly convex for an θ_i and θ_j and a constant $\mu \geq 0$ such that:

$$l_i(\theta_i) \geq l_i(\theta_j) + \langle \nabla l_i(\theta_j),\ \theta_i - \theta_j \rangle + \frac{\mu}{2}||\theta_j - \theta_i||_2^2 \tag{7.1}$$

7.5.1 Properties of $\mathcal{W}$

In this part, we show that our formulation of the network topology and the underlying assumptions render the associated adjacency matrix $\mathcal{W}$ to possess the same properties as those described for the adjacency matrix in [90]. We use the Gershgorin Circle and Brouwer Fixed Point Theorem to determine the properties of $\mathcal{W}$, its dominant Eigenvalue and the associated left and right Eigen Vectors.

Definition A (Spectral Radius) Let $\lambda_1, \ldots\ldots \lambda_n$ be the eigenvalues of a matrix $\boldsymbol{A} \in \mathbb{C}^{n \times n}$. Then, its spectral radius can be defined as:

$$\zeta(\boldsymbol{A}) = max\{|\lambda_1|, \ldots\ldots |\lambda_n|\}$$

Definition B (Adjacency Matrix $\mathcal{W}$) Let the adjacency matrix for the network for $\boldsymbol{N}$ nodes be $\mathcal{W} \in \mathbb{R}^{N \times N}$. Then, sampling $\mathcal{W}^t \sim \mathcal{W} \in \mathbb{R}^{N \times N}$ may be used to indicate the presence of stragglers and volatile nodes. Assuming $\mathbb{P}$ is the set of active nodes at time t in the neighborhood of node i i.e. $|n_i|$ with $|\mathbb{P}| = k \leq |n_i|$ nodes active in the learning process, then the i-th row of $\mathcal{W}^t$ i.e $\mathcal{W}_i^t$ will have exactly k non-zero entries. Then $\mathcal{W}_i^t$ can then be defined as $\mathcal{W}_i^t = \frac{1}{k}\mathbb{1}_{\mathbb{P}} = \frac{1}{k}[p_1, p_2, \ldots\ldots p_N]$ where $\mathbb{1}_{\mathbb{P}} = [p_1, p_2, \ldots\ldots p_N]$ is an indicator vector such that $p_i = 1 \; \forall \; i \in \mathbb{P}$ and 0 otherwise. With this definition of the indicator vector $\mathbb{1}_{\mathbb{P}}$ and a uniform weighting mechanism, the entries ω_{ij} in row $\mathcal{W}_i^t$ can be described as:

$$\sum \mathcal{W}_i^t = \sum_{j=1}^{N} \omega_{ij} = \sum_{j=1}^{N} \frac{1}{k} \mathbb{1}_{\mathbb{P}}^T = 1$$

This shows that the sampled $\mathcal{W}^t$ is a row-stochastic matrix.

Lemma 1 The sampled adjacency matrix $\mathcal{W}^t$ satisfies the following:

1. 1 is the largest eigenvalue of $\mathcal{W}^t$ (both regular and absolute). In other words, the spectral radius of $\mathcal{W}^t$ is $\zeta(\mathcal{W}^t) = 1$

2. The right Eigen Vector of $\mathcal{W}^t$ associated with the Eigen-value 1 is $\mathbb{1} \in R^N$.

3. The left Eigen Vector of $\mathcal{W}^t$ associated with the Eigen-value 1, represented by ψ, is non-negative.

Proof From the Definition B, given that $\mathcal{W}^t$ is row-stochastic, then using $\mathbb{1}$ as vector of 1's with the dimension N,

$$\mathcal{W}^t\mathbb{1} = \lambda(\mathcal{W}^t)\mathbb{1} = \mathbb{1} \tag{7.2}$$

This shows that

1. $\mathbb{1}$ is a right Eigen Vector of $\mathcal{W}^t$
2. 1 is an Eigenvalue of $\mathcal{W}^t$

With the row-stochasticity property of $\mathcal{W}^t$ and using the Gershgorin Circle Theorem, the largest Eigenvalue, $\lambda_{max}(\mathcal{W}^t)$ of $\mathcal{W}^t$ satisfies the following:

$$\lambda(\mathcal{W}^t) \leq 1 \tag{7.3}$$

The Eq. 7.2 and Eq. 7.3 imply that 1 is the largest Eigen-value of $\mathcal{W}^t$, i.e., the spectral radius of $\mathcal{W}^t$ is 1, depicted as $\zeta(\mathcal{W}^t) = \lambda_{max}(\mathcal{W}^t) = 1$. Therefore, $\mathbb{1}$ is a right Eigen Vector of $\mathcal{W}^t$ corresponding to the largest Eigen-value of 1.

Given $\lambda_{max}(\mathcal{W}^t) = 1$, let the associated left Eigen Vector can be denoted as:

$$\psi = \begin{bmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_N \end{bmatrix} \tag{7.4}$$

then

$$\mathcal{W}^{t^T}\psi = \psi$$

From the Brouwer Fixed Point Theorem, for any stochastic matrix, there is a left Eigen vector which is a stationary probability vector. Hence, the left Eigen Vector ψ corresponding to the largest eigen-value is real and non-negative.

Let θ_i^t be the model parameters associated with node i at time t. Given $\boldsymbol{N}$ nodes, let $\mathbb{1}$ be a vector of 1's with dimensions $1 \times N$. Let $\boldsymbol{\theta}^t = [\theta_1^t, \theta_2^t \ldots\ldots \theta_N^t] \in \mathbb{R}^{d \times N}$. Then the average of all models at the time t is:

$$\begin{aligned}\overline{\theta}^t &= \frac{1}{N}\sum_{i=1}^{N}\theta_i^t \\ &= \boldsymbol{\theta}^t \frac{1}{N}\mathbb{1}^T\end{aligned}$$

Given the distributed learning problem, the distance between the consensus model and the local model at time t can be defined as:

$$d^t = E||\overline{\theta}^t - \theta_i^t||^2 \tag{7.5}$$

The final aggregation form for the hybrid Memory and Augmented Graph Assisted Algorithm is given as:

$$\boldsymbol{\theta}_i^{t+1} = \eta_i \boldsymbol{\theta}_i^{t+\tau} + \sum_{r \in \pi_i} \eta_r \boldsymbol{\theta}_r^{t+\tau} + \sum_{k \in \rho_i^t} \eta_k \boldsymbol{\theta}_k^{t-1} + \sum_{m \in \nu_i^t} \eta_m \boldsymbol{\theta}_m^{t-1} \tag{7.6}$$

The first two terms represent node i's own model as well as the received ones. The last two terms indicate the models employed from memory as well as those relayed by the participants. The adjacency matrix, $\mathcal{W}^t$, can be guaranteed to be always row-stochastic by the choice of the aggregation weights $\eta_i,\ \eta_r,\ \eta_k$ and η_m.

We rely on the analysis of [90] for our convergence guarantees after this point. In Eq. 7.6, the first two expressions represent the standard DFL process [104, 116]. The last two terms represent models from the previous time steps and have been shown to converge in [90, 93]. As the overall expression is a linear combination of the respective terms, it can be inferred that the main aggregation expression converges.

7.6 Results and Analysis

This work performs a combined analysis of the proposed algorithms using DFL as the baseline. All the algorithms are subjected to the same level of volatility, asynchronicity, and data distribution.

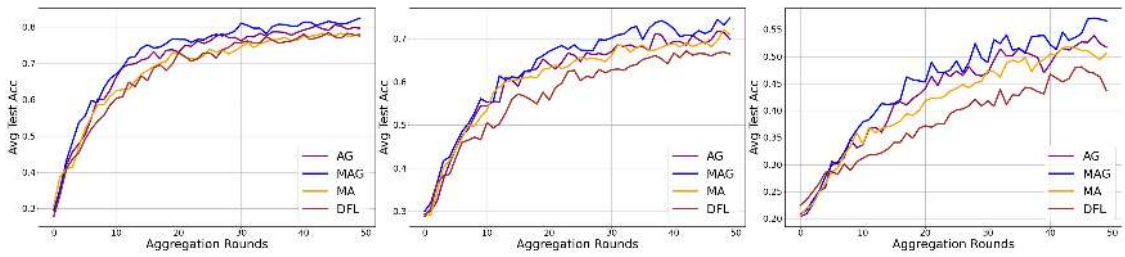
The results for the various scenarios of environment volatility with asynchronous participation have been depicted in Fig 7.1 - 7.5 and Tables 7.1 - 7.3. Results indicate both Memory assistance and Augmented-Graph assisted DFL to outperform DFL in scenarios of low participation and non-IID data. Additionally, the proposed hybrid algorithm Memory and Augmented-Graph assisted DFL outperforms all the rest in the given scenarios.

Participation proportion becomes a key factor, particularly in cases of heterogeneous data distribution. The figures Fig 7.1 - 7.5 indicate this recurring trend where the gap between the baseline and MAG_DFL increases for a given distribution as participation becomes more extreme. The said figures also indicate higher variance in the overall results as participation probability goes from $p_i = 0.4$ to $p_i = 0.05$. Furthermore, although MA_DFL and AG_DFL both employ models shared in the previous training round, the added diversity afforded by AG_DFL by sharing inaccessible models with the respective nodes gives it a better performance over MA_DFL. However, since the relayed models depend on whether the sending nodes themselves have received models or not and are not guaranteed to relay in every round, the comparable advantage indicates inconsistencies due to the stochasticity of operation.

The results in Tables 7.1 - 7.3 also indicate the results for FMA_DFL. The results are compiled for various data distributions. Each table presents the results for different asynchronous training levels for different participation levels, with entries indicating test accuracies averaged over 5 rounds. It is evident from the results that greater asynchronicity levels (Epochs 1-10) result in greater performance differences

between the said algorithms. With greater differences in training, the improvement obtained by MAG_DFL is over 10% with the lowest participation thresholds. As participation increases, this gap reduces as nodes access a greater variety of distribution. The results in Table 7.2 show the maximum performance gain between the baseline and MAG_DFL at 11%. With lesser data heterogeneity and asynchronous training, as shown in Table 7.1, MAG_DFL improvement falls to around $3-4\%$ for 30% or less participation.

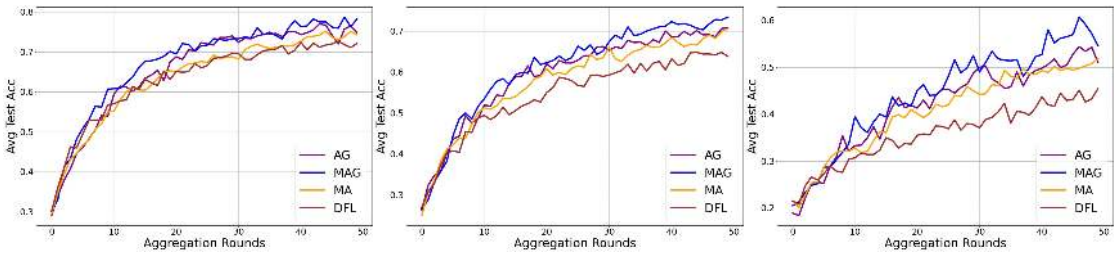
From Fig 7.1-7.5, it is evident that the gap reduces as the environment volatility reduces. Under cases of extremely low participation where only 5% of the neighborhood of any node participates, we see the proposed hybrid outperforms the baseline by 10% performance gain with the 2-class skewed distribution as evident in Fig 7.1-c. Within a given participation threshold, it may be observed that the performance for hybrid manages to outperform the individual algorithms as the data distribution becomes more extreme. This fact may be attributed to employing approximate models from the memory array from the previous round i.e. $\mathcal{M}_i^{t-1}$, and an increase in available distribution diversity in the form of relayed models from the neighborhood.



(a) $\alpha = 0.1$ and $p_i = 0.05$ (b) $s = 3$ and $p_i = 0.05$ (c) $s = 2$ and $p_i = 0.05$

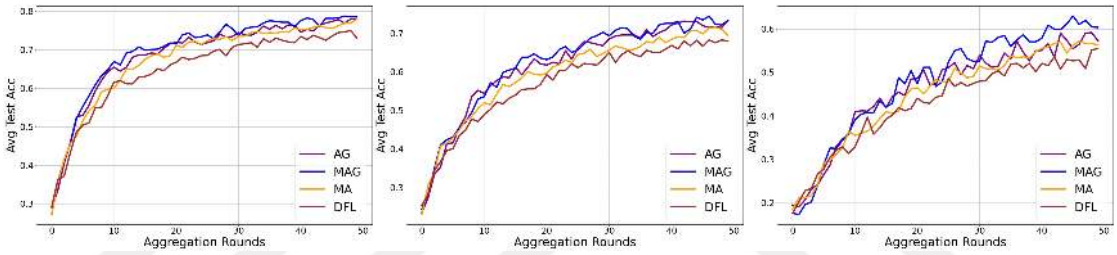
Figure 7.1: Average test accuracy for 50 Communication rounds for non-IID distributed FashionMNIST with $\alpha = 0.1$, $s = 3$ & $s = 2$ and participation threshold as $p_i = 0.05$

While the results indicate a performance improvement, the relay operation imposes additional communication costs. However, the communication overhead may



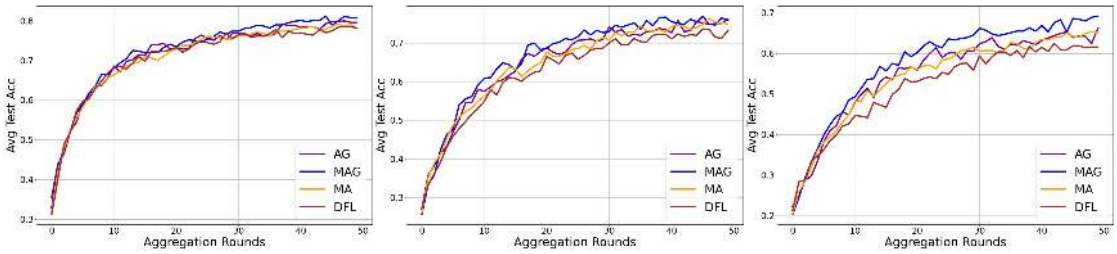
(a) $\alpha = 0.1$ and $p_i = 0.1$ (b) $s = 3$ and $p_i = 0.1$ (c) $s = 2$ and $p_i = 0.1$

Figure 7.2: Average test accuracy for 50 Communication rounds for non-IID distributed FashionMNIST with $\alpha = 0.1$, $s = 3$ & $s = 2$ and participation threshold as $p_i = 0.1$



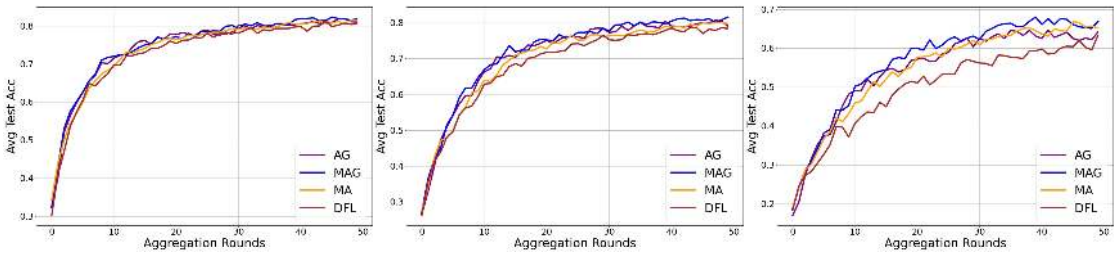
(a) $\alpha = 0.1$ and $p_i = 0.2$ (b) $s = 3$ and $p_i = 0.2$ (c) $s = 2$ and $p_i = 0.2$

Figure 7.3: Average test accuracy for 50 Communication rounds for non-IID distributed FashionMNIST with $\alpha = 0.1$, $s = 3$ & $s = 2$ and participation threshold as $p_i = 0.2$



(a) $\alpha = 0.1$ and $p_i = 0.3$ (b) $s = 3$ and $p_i = 0.3$ (c) $s = 2$ and $p_i = 0.3$

Figure 7.4: Average test accuracy for 50 Communication rounds for non-IID distributed FashionMNIST with $\alpha = 0.1$, $s = 3$ & $s = 2$ and participation threshold as $p_i = 0.3$



(a) $\alpha = 0.1$ and $p_i = 0.4$ (b) $s = 3$ and $p_i = 0.4$ (c) $s = 2$ and $p_i = 0.4$

Figure 7.5: Average test accuracy for 50 Communication rounds for non-IID distributed FashionMNIST with $\alpha = 0.1$, $s = 3$ & $s = 2$ and participation threshold as $p_i = 0.4$

Local updates	**Epochs 1-6**			**Epochs 1-3**		
Algorithm / Participation	**50%**	**30%**	**10%**	**50%**	**30%**	**10%**
DFL	**0.827**	0.794	0.734	**0.783**	**0.777**	0.704
MA_DFL	**0.827**	0.798	0.74	0.776	0.775	0.71
AG_DFL	0.806	0.784	0.748	0.755	0.765	0.717
MAG_DFL	0.821	**0.804**	**0.77**	0.762	**0.777**	**0.738**
FMA_DFL	0.815	0.765	0.691	0.77	0.757	0.655

Table 7.1: Results for various Algorithms, different asynchronous training levels and participation levels for $\alpha = 0.1$ skewed data distribution for FashionMNIST dataset.

be justified because the proposed algorithms are designed to operate in low participation scenarios where the nodes are infrequent in their aggregation rounds. Furthermore, the real operation may be subjected to the overall participation conditions to help control the communication costs.

Communication Volume With the Augmented Graph Assisted DFL (AG_DFL) and the inclusion of the same mechanism in the hybrid MAG_DFL algorithm, each node relays additional models within a given federation round. Given that, each node i sends models to its neighborhood n_i under default conditions, the total communication volume for each node becomes $O(n_i)$. With N total nodes in the network,

Local updates	**Epochs 1-10**			**Epochs 1-6**			**Epochs 1-3**		
Algorithm / Participation	**50%**	**30%**	**10%**	**50%**	**30%**	**10%**	**50%**	**30%**	**10%**
DFL	0.686	0.702	0.519	0.703	0.651	0.571	**0.669**	0.639	0.524
MA_DFL	0.709	.719	0.61	**0.719**	0.668	0.599	0.664	0.655	0.581
AG_DFL	0.691	0.712	0.628	0.687	0.668	0.645	0.64	0.646	0.592
MAG_DFL	0.707	**0.734**	**0.634**	0.691	**0.7**	**0.647**	0.66	**0.657**	**0.596**
FMA_DFL	**0.71**	0.688	0.506	0.702	0.652	0.491	0.657	0.62	0.498

Table 7.2: Results for various Algorithms, different asynchronous training levels and participation levels for S = 2 skewed data distribution for FashionMNIST dataset.

Local updates	**Epochs 1-10**			**Epochs 1-6**			**Epochs 1-3**		
Algorithm / Participation	**50%**	**30%**	**10%**	**50%**	**30%**	**10%**	**50%**	**30%**	**10%**
DFL	0.819	0.764	0.688	0.785	0.784	0.669	**0.773**	0.733	0.662
MA_DFL	**0.829**	0.778	0.72	0.79	0.797	0.693	0.77	0.753	0.691
AG_DFL	0.809	0.77	0.747	0.78	0.787	0.704	0.756	0.74	0.712
MAG_DFL	0.825	**0.781**	**0.77**	**0.793**	**0.803**	**0.719**	0.771	**0.759**	**0.717**
FMA_DFL	0.822	0.755	0.685	.792	0.778	0.635	0.763	0.734	0.621

Table 7.3: Results for various Algorithms, different asynchronous training levels, and participation levels for S = 3 skewed data distribution for FashionMNIST dataset.

the average degree of the nodes in the network becomes

$$\bar{n} = \frac{1}{N} \sum_{i=1}^{N} n_i$$

where n_i is the degree i.e. the number of neighbors, of each node i. Thus, the total communication volume of the entire network for the average degree $\bar{n}$ becomes $\sim O(\bar{n}N)$. For a complete network, $\bar{n} \approx N$, with each node connected to every other node of the network, the total communication volume for a complete graph becomes $\sim O(N^2)$.

However, with the conditions of sparsity and volatility chosen as part of our system model, for any given federation round,

$$1 \leq \bar{n} \ll N$$

This implies that the default communication volume for a sparse graph with the

above-defined condition becomes

$$O(\bar{n}N) \approx O(N)$$

The proposed graph augmentation requires each node to communicate two models whenever possible. Therefore, the total communication volume becomes:

$$O(2\bar{n}N) \approx O(N)$$

This communication volume persists for sparse and volatile conditions chosen as part of our research for both AG_DFL and MAG_DFL.

7.7 Conclusion

Decentralized Federated Learning offers a powerful alternative to distributed learning by removing the bottlenecks caused by central orchestration. However, a lack of global information and access to a much lesser fraction of devices leads to slower information diffusion and convergence rates. Node volatility, in addition to data heterogeneity, offers a critical challenge to the DFL performance. The algorithms proposed in this work, including Memory-Assisted DFL (MA_DFL), Augmented Graph-Assisted DFL (AG_DFL), and hybrid Memory and Augmented Graph-Assisted DFL (MAG_DFL), show promising results when subjected to highly volatile conditions and heterogeneous data distributions. The proposed algorithms keep the restrictions associated with node participation in perspective and require minimal computation overhead. With MA_DFL, the nodes utilize models stored in their respective memory arrays as approximations of the absent models. This allows higher data diversity and improves convergence in the presence of node volatility. In the Augmented Graph Assisted DFL, our proposed solution employs selective relaying in order to allow access to a greater fraction of models. With higher volatility, our solution relies on enabling the active nodes to relay models across neighborhoods. To avoid causing additional bias, the nodes only relay models from non-overlapping neighborhoods.

With the hybrid MAG_DFL algorithm, we propose a mechanism that jointly employs the solutions offered in MA_DFL and AG_DFL without additional overhead. The results indicate that the MAG_DFL outperforms not only the baseline but also its respective constituent algorithms under highly volatile conditions with extreme non-IID data distribution. The performance improvement becomes significant as the nodes are subjected to increasing volatile behaviors and skewed data distributions. The communication cost for relaying operations in Graph Augmented DFL doubles because of the relaying feature; however, the suggested mechanism is designed for operation in extremely low participation scenarios, which ensures that the overall communication remains at par when the entire set of devices is participating. These features make these algorithms extremely attractive for DFL operation in volatile conditions. An extension of this work would be quantifying the information diffusion rate under volatile participation conditions and measuring diversity under conditions of local interactions.

Chapter 8

CONCLUSION AND FUTURE WORK

In this research, we conduct an extensive analysis of the practical challenges faced in the application of Decentralized Federated Learning (DFL). Decentralized methods offer powerful alternatives to centralized learning schemes and are assuming greater significance in the next-generation communication networks. Despite the challenges of non-IID data, device and communication heterogeneity, we show that the DFL performance trade-off is well worth the decrease in communication cost. However, device behavior, particularly under volatile settings, impacts learning much more than centralized variants, and therefore, this research focuses on the performance of DFL under volatile participation settings.

8.1 Concluding Remarks

The development of the Federated Learning Algorithm Simulation (FLAGS) framework allowed us a platform to prototype and evaluate multiple FL algorithms rapidly. Being lightweight and extremely modular, it offers a powerful solution to undertake research into novel FL algorithms under different environmental conditions.

Building on the application of the FLAGS framework, our next step conducts a comprehensive comparison of various FL algorithms, expanding their evaluation beyond the Centralized FL context. The developed FLAGS framework allows the swift configuration of diverse FL architectures. Notably, the study highlights the performance of major FL algorithms in the face of significant challenges. Results show that decentralized FL performs well even with limited upstream communication. While noise and irregular participation affect decentralized FL, they still

achieve acceptable performance. However, highly skewed data distributions notably degrade fully decentralized FL more than centralized approaches. Findings suggest interchangeable use of FL modes based on network conditions and communication costs.

The utilization of selection strategies within Federated Learning (FL) has enhanced convergence and communication efficiency. Our work extends this research in the DFL domain and evaluates the impact of integrating node selection in volatile conditions. Through evaluations employing rigid and adaptable stochastic node selection techniques, utilizing metrics like training loss, validation accuracy, and a parameterized variant, findings indicate that selection bias exerts a more pronounced effect in dynamically unstable environments with limited client engagement. Notably, hard selection methods demonstrate heightened susceptibility to extreme data distributions and node involvement. Conversely, random selection demonstrates the potential for ensuring distribution diversity in extreme conditions.

Delving further into improving the performance of DFL under volatile device participation, we propose three different algorithms utilizing entirely local information. The Memory-Assisted DFL allows nodes to retain models from the previous rounds and use them as alternates to missing nodes. The next part of the research proposes graph augmentation to relay additional models between nodes. Nodes making use of neighborhood information send models from non-overlapping neighbors to other nodes. Finally, a hybrid algorithm fusing features of both these algorithms is proposed as Memory and Augmented Graph Assisted DFL. The hybrid outperforms not only the baseline but the individual algorithms as well without exceeding the costs of the respective individual algorithms. The proposed algorithms allow us to improve the performance of DFL under volatile conditions with extremely low participation thresholds.

8.2 Future Directions

Decentralized Federated Learning holds immense promise for practical applications in 5G and 6G networks. Simultaneously, however, the participating entities will become increasingly independent, and thus, volatile participation shall be encountered more frequently. In this regard, allowing devices to adopt a Multi-FL scheme in which they may adaptively select the FL algorithm to employ is an important avenue. With the devices getting equipped with multiple communication mechanisms, configuring a multi-FL scheme where devices are able to switch between different FL algorithms not only holds the potential for maximizing participation but also of more efficient communication of the allocated communication resources. Further extending this application, performing FL simultaneously over different links and across different edge network hierarchies may be considered as a potential future direction. Such a scenario would allow devices such as smartphones to simultaneously employ mobile networks, WiFi networks, and their serverless counterparts, short-range communication links such as Bluetooth etc. The important challenges arising from this case would be those of energy and communication budgeting as well as managing the asynchronicity arising by employing links supporting multiple data rates.

Future research avenues also include phase-wise selection strategies leveraging training phase trends and deeper exploration of information diffusion in contexts of extreme data distributions and node participation variability. Device participation characteristics can have a significant impact on decentralized learning performance. The information diffusion across the network may well slow down in highly volatile decentralized learning scenarios.

BIBLIOGRAPHY

[1] "Cisco annual internet report 2018–2023 white paper"."

[2] "Ericsson Mobility Report June 2019," p. 36, 2019.

[3] "Key drivers and research challenges for 6G ubiquitous wireless intelligence."

[4] D. S. W. Ting, L. Carin, V. Dzau, and T. Y. Wong, "Digital technology and covid-19," *Nature medicine*, vol. 26, no. 4, pp. 459–461, 2020.

[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[6] Y. Chen, Y. Zhang, S. Maharjan, M. Alam, and T. Wu, "Deep learning for secure mobile edge computing in cyber-physical transportation systems," *IEEE Network*, vol. 33, p. 36–41, Jul 2019.

[7] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proceedings of the IEEE*, vol. 107, p. 1655–1674, Aug 2019.

[8] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 22, no. 2, p. 869–904, 2020.

[9] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, p. 1657–1681, 2017.

[10] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[11] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, p. 14410–14430, 2018.

[12] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, p. 372–387, Mar 2016.

[13] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, p. 2805–2824, Sep 2019.

[14] Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv, "Edge computing security: State of the art and challenges," *Proceedings of the IEEE*, vol. 107, p. 1608–1631, Aug 2019.

[15] Y. Chen, Y. Zhang, S. Maharjan, M. Alam, and T. Wu, "Deep learning for secure mobile edge computing in cyber-physical transportation systems," *IEEE Network*, vol. 33, p. 36–41, Jul 2019.

[16] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," *IEEE Internet of Things Journal*, p. 1–1, 2020.

[17] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," *Proceedings of the IEEE*, vol. 107, p. 2204–2239, Nov 2019.

[18] Z. C. Lipton, "The mythos of model interpretability," *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[19] C. Molnar, *Interpretable Machine Learning*. 2019. `https://christophm.github.io/interpretable-ml-book/`.

[20] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[21] S. Chakraborty, R. Tomsett, R. Raghavendra, D. Harborne, M. Alzantot, F. Cerutti, M. Srivastava, A. Preece, S. Julier, R. M. Rao, and et al., "Interpretability of deep learning models: A survey of results," in *2017 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced nad Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, p. 1–6, IEEE, Aug 2017.

[22] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.

[23] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Advances in neural information processing systems*, vol. 30, 2017.

[24] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[25] T. Wang, Y. Liu, X. Zheng, H.-N. Dai, W. Jia, and M. Xie, "Edge-based communication optimization for distributed federated learning," *IEEE Transactions on Network Science and Engineering*, vol. 9, p. 2015–2024, Jul 2022.

[26] A. H. Lodhi, B. Akgün, and Ö. Özkasap, "State-of-the-art techniques in deep edge intelligence," *arXiv:2008.00824*, 2020.

[27] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence:

Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, p. 1738–1762, Aug 2019.

[28] M. Xie, G. Long, T. Shen, T. Zhou, X. Wang, and J. Jiang, "Multi-center federated learning," *arXiv:2005.01026 [cs, stat]*, May 2020. arXiv: 2005.01026.

[29] I. Hegedűs, G. Danner, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning," in *IFIP International Conference on Distributed Applications and Interoperable Systems*, pp. 74–90, Springer, 2019.

[30] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Network*, 2020.

[31] S. R. Pokhrel and J. Choi, "Federated learning with blockchain for autonomous vehicles: Analysis and design challenges," *IEEE Transactions on Communications*, vol. 68, no. 8, pp. 4734–4746, 2020.

[32] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair resource allocation in federated learning," *arXiv:1905.10497 [cs, stat]*, Feb 2020. arXiv: 1905.10497.

[33] M. N. Tehrani, M. Uysal, and H. Yanikomeroglu, "Device-to-device communication in 5g cellular networks: challenges, solutions, and future directions," *IEEE Communications Magazine*, vol. 52, no. 5, pp. 86–92, 2014.

[34] G. Soatti, M. Nicoli, S. Savazzi, and U. Spagnolini, "Consensus-based algorithms for distributed network-state estimation and localization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 2, pp. 430–444, 2016.

[35] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive iot networks," *IEEE Internet of Things Journal*, vol. 7, p. 4641–4654, May 2020.

[36] R. I. Ansari, C. Chrysostomou, S. A. Hassan, M. Guizani, S. Mumtaz, J. Rodriguez, and J. J. Rodrigues, "5g d2d networks: Techniques, challenges, and future prospects," *IEEE Systems Journal*, vol. 12, no. 4, pp. 3970–3984, 2017.

[37] A. Lalitha, S. Shekhar, T. Javidi, and F. Koushanfar, "Fully decentralized federated learning," in *Third workshop on Bayesian Deep Learning (NeurIPS)*, 2018.

[38] Y. Lu, Z. Yu, and N. Suri, "Privacy-preserving decentralized federated learning over time-varying communication graph," *ACM Trans. Priv. Secur.*, vol. 26, jun 2023.

[39] Y. Li, F. Li, L. Chen, L. Zhu, P. Zhou, and Y. Wang, "Power of redundancy: Surplus client scheduling for federated learning against user uncertainties," *IEEE Transactions on Mobile Computing*, p. 1–1, 2022.

[40] B. Ma, J. Wu, W. Liu, L. Chiaraviglio, and X. Ming, "Combating hard or soft disasters with privacy-preserving federated mobile buses-and-drones based networks," in *2020 IEEE 21st international conference on information reuse and integration for data science (IRI)*, pp. 31–36, IEEE, 2020.

[41] Z. Yuan, D. Lu, G. Zhang, and H. Liu, "Federated learning based path planning method for crowd evacuation," in *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 1161–1166, IEEE, 2022.

[42] V. Mittal, M. Jahanian, and K. Ramakrishnan, "Flare: Federated active learning assisted by naming for responding to emergencies," in *Proceedings of the 8th ACM Conference on Information-Centric Networking*, pp. 71–82, 2021.

[43] Y. Qu, H. Dai, Y. Zhuang, J. Chen, C. Dong, F. Wu, and S. Guo, "Decentralized federated learning for uav networks: Architecture, challenges, and opportunities," *IEEE Network*, vol. 35, no. 6, pp. 156–162, 2021.

[44] Y. Wang, Z. Su, N. Zhang, and A. Benslimane, "Learning in the air: Secure federated learning for uav-assisted crowdsensing," *IEEE Transactions on network science and engineering*, vol. 8, no. 2, pp. 1055–1069, 2020.

[45] H. Zhang and L. Hanzo, "Federated learning assisted multi-uav networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 14104–14109, 2020.

[46] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-iid data," in *2020 IEEE International Conference on Big Data (Big Data)*, pp. 15–24, IEEE, 2020.

[47] A. H. Lodhi, B. Akgün, and Ö. Özkasap, "Flags framework for comparative analysis of federated learning algorithms," *Internet of Things*, vol. 20, p. 100638, 2022.

[48] A. H. Lodhi, B. Akgün, and Özkasap, "Implications of node selection in decentralized federated learning," in *2023 31st Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2023.

[49] I. Hegedűs, G. Danner, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning," 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 19*, pp. 74–90, Springer, 2019.

[50] H. Xing, O. Simeone, and S. Bi, "Federated learning over wireless device-to-device networks: Algorithms and convergence analysis," *arXiv:2101.12704 [cs, eess, math]*, Jan 2021. arXiv: 2101.12704.

[51] S. S. Ram, A. Nedić, and V. V. Veeravalli, "Distributed stochastic subgradient

projection algorithms for convex optimization," *Journal of optimization theory and applications*, vol. 147, no. 3, pp. 516–545, 2010.

[52] S. Savazzi, M. Nicoli, V. Rampa, and S. Kianoush, "Federated learning with mutually cooperating devices: A consensus approach towards server-less model optimization," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, p. 3937–3941, May 2020.

[53] C. Li, G. Li, and P. K. Varshney, "Decentralized federated learning via mutual knowledge transfer," *IEEE Internet of Things Journal*, vol. 9, p. 1136–1147, Jan 2022.

[54] J. Wang, S. Wang, R.-R. Chen, and M. Ji, "Local averaging helps: Hierarchical federated learning and convergence analysis," *arXiv:2010.12998*, Mar 2021.

[55] M. S. H. Abad, E. Ozfatura, D. GUndUz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, p. 8866–8870, May 2020.

[56] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2020.

[57] F. P.-C. Lin, S. Hosseinalipour, S. S. Azam, C. G. Brinton, and N. Michelusi, "Semi-decentralized federated learning with cooperative d2d local model aggregations," *IEEE Journal on Selected Areas in Communications*, p. 1–1, 2021.

[58] A. Hashemi, A. Acharya, R. Das, H. Vikalo, S. Sanghavi, and I. Dhillon, "On the benefits of multiple gossip steps in communication-constrained decentralized federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 11, pp. 2727–2739, 2021.

[59] T. Wang, Y. Liu, X. Zheng, H.-N. Dai, W. Jia, and M. Xie, "Edge-based communication optimization for distributed federated learning," *IEEE Transactions on Network Science and Engineering*, 2021.

[60] J. S. Ng, W. Y. B. Lim, Z. Xiong, X. Cao, J. Jin, D. Niyato, C. Leung, and C. Miao, "Reputation-aware hedonic coalition formation for efficient serverless hierarchical federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 11, pp. 2675–2686, 2021.

[61] A. Singh, P. Vepakomma, O. Gupta, and R. Raskar, "Detailed comparison of communication efficiency of split learning and federated learning," *arXiv preprint arXiv:1909.09145*, 2019.

[62] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, and M. Jirstrand, "A performance evaluation of federated learning algorithms," in *Proceedings of the second workshop on distributed infrastructures for deep learning*, pp. 1–8, 2018.

[63] I. Hegedűs, G. Danner, and M. Jelasity, "Decentralized learning works: An empirical comparison of gossip learning and federated learning," *Journal of Parallel and Distributed Computing*, vol. 148, pp. 109–124, 2021.

[64] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The non-iid data quagmire of decentralized machine learning," in *International Conference on Machine Learning*, pp. 4387–4398, PMLR, 2020.

[65] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv preprint arXiv:1712.01887*, 2017.

[66] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-iid data: A survey," *arXiv preprint arXiv:2106.06843*, 2021.

[67] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," in *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9, IEEE, 2020.

[68] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International Conference on Machine Learning*, pp. 5132–5143, PMLR, 2020.

[69] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv:1907.02189*, Jun 2020.

[70] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient federated learning via guided participant selection," in *15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21)*, pp. 19–35, 2021.

[71] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, T. Parcollet, P. P. de Gusmão, and N. D. Lane, "Flower: A friendly federated learning research framework," *arXiv:2007.14390*, 2020.

[72] "TensorFlow Federated — tensorflow.org." `https://www.tensorflow.org/federated`.

[73] F. Lai, Y. Dai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Fedscale: Benchmarking model and system performance of federated learning," in *Proceedings of the First Workshop on Systems Challenges in Reliable and Secure Federated Learning*, pp. 1–3, 2021.

[74] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *arXiv:1812.01097*, 2018.

[75] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," *arXiv preprint arXiv:2102.02079*, 2021.

[76] C. He, S. Li, J. So, X. Zeng, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, *et al.*, "Fedml: A research library and benchmark for federated machine learning," *arXiv:2007.13518*, 2020.

[77] E. T. M. Beltrán, M. Q. Pérez, P. M. S. Sánchez, S. L. Bernal, G. Bovet, M. G. Pérez, G. M. Pérez, and A. H. Celdrán, "Decentralized federated learning: Fundamentals, state-of-the-art, frameworks, trends, and challenges," *arXiv preprint arXiv:2211.08413*, 2022.

[78] H. Ye, L. Liang, and G. Y. Li, "Decentralized federated learning with unreliable communications," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, p. 487–500, Apr 2022.

[79] P. Pinyoanuntapong, W. H. Huff, M. Lee, C. Chen, and P. Wang, "Toward scalable and robust aiot via decentralized federated learning," *IEEE Internet of Things Magazine*, vol. 5, no. 1, pp. 30–35, 2022.

[80] Q. Liu, B. Yang, Z. Wang, D. Zhu, X. Wang, K. Ma, and X. Guan, "Asynchronous decentralized federated learning for collaborative fault diagnosis of pv stations," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 3, pp. 1680–1696, 2022.

[81] E. Jeong, M. Zecchin, and M. Kountouris, "Asynchronous decentralized learning over unreliable wireless networks," in *ICC 2022-IEEE International Conference on Communications*, pp. 607–612, IEEE, 2022.

[82] L. Wang, Y. Xu, H. Xu, M. Chen, and L. Huang, "Accelerating decentralized federated learning in heterogeneous edge computing," *IEEE Transactions on Mobile Computing*, 2022.

[83] Z. Tang, S. Shi, B. Li, and X. Chu, "Gossipfl: A decentralized federated learning framework with sparsified and adaptive communication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 3, pp. 909–922, 2022.

[84] S. Li, T. Zhou, X. Tian, and D. Tao, "Learning to collaborate in decentralized learning of personalized models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9766–9775, 2022.

[85] R. Dai, L. Shen, F. He, X. Tian, and D. Tao, "Dispfl: Towards communication-efficient personalized federated learning via decentralized sparse training," in *International Conference on Machine Learning*, pp. 4587–4604, PMLR, 2022.

[86] D. M. Manias and A. Shami, "Making a case for federated learning in the internet of vehicles and intelligent transportation systems," *IEEE Network*, vol. 35, p. 88–94, May 2021.

[87] M. Hu, D. Wu, Y. Zhou, X. Chen, and M. Chen, "Incentive-aware autonomous client participation in federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, p. 2612–2627, Oct 2022.

[88] T. Huang, W. Lin, L. Shen, K. Li, and A. Y. Zomaya, "Stochastic client selection for federated learning with volatile clients," *IEEE Internet of Things Journal*, vol. 9, p. 20055–20070, Oct 2022.

[89] F. Shi, C. Hu, W. Lin, L. Fan, T. Huang, and W. Wu, "Vfedcs: Optimizing client selection for volatile federated learning," *IEEE Internet of Things Journal*, vol. 9, p. 24995–25010, Dec 2022.

[90] T. Vogels, L. He, A. Koloskova, S. P. Karimireddy, T. Lin, S. U. Stich, and M. Jaggi, "Relaysum for decentralized deep learning on heterogeneous data," *Advances in Neural Information Processing Systems*, vol. 34, pp. 28004–28015, 2021.

[91] M. Yemini, R. Saha, E. Ozfatura, D. Gündüz, and A. J. Goldsmith, "Semi-decentralized federated learning with collaborative relaying," in *2022 IEEE International Symposium on Information Theory (ISIT)*, pp. 1471–1476, IEEE, 2022.

[92] Z. Lin, H. Liu, and Y.-J. A. Zhang, "Relay-assisted cooperative federated learning," *IEEE Transactions on Wireless Communications*, vol. 21, no. 9, pp. 7148–7164, 2022.

[93] X. Gu, K. Huang, J. Zhang, and L. Huang, "Fast federated learning in the presence of arbitrary device unavailability," *arXiv*, Jun 2021. arXiv:2106.04159 [cs, math].

[94] D. Jhunjhunwala, P. Sharma, A. Nagarkatti, and G. Joshi, "Fedvarp: Tackling the variance due to partial client participation in federated learning," in *Uncertainty in Artificial Intelligence*, pp. 906–916, PMLR, 2022.

[95] A. H. Lodhi, B. Akgün, and Ö. Özkasap, "Flags simulation framework for federated learning algorithms," in *2023 14th International Conference on Network of the Future*, IEEE, 2023.

[96] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, "Federated learning for internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, 2021.

[97] Y. Gou, R. Wang, Z. Li, M. A. Imran, and L. Zhang, "Clustered hierarchical distributed federated learning," in *ICC 2022 - IEEE International Conference on Communications*, p. 177–182, May 2022.

[98] J. Daily, A. Vishnu, C. Siegel, T. Warfel, and V. Amatya, "Gossipgrad: Scalable deep learning using gossip communication based asynchronous gradient descent," *arXiv preprint arXiv:1803.05880*, 2018.

[99] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[100] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv:1708.07747*, 2017.

[101] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10713–10722, 2021.

[102] N. Guha, A. Talwalkar, and V. Smith, "One-shot federated learning," *arXiv:1902.11175*, 2019.

[103] M. Condoluci, L. Militano, A. Orsino, J. Alonso-Zarate, and G. Araniti, "Lte-direct vs. wifi-direct for machine-type communications over lte-a systems," in *2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 2298–2302, 2015.

[104] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive iot networks," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4641–4654, 2020.

[105] W. Liu, L. Chen, and W. Zhang, "Decentralized federated learning: Balancing communication and computing costs," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 8, 2022.

[106] H. Ye, L. Liang, and G. Y. Li, "Decentralized federated learning with unreliable communications," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 3, pp. 487–500, 2022.

[107] I. Mohammed, S. Tabatabai, A. Al-Fuqaha, F. E. Bouanani, J. Qadir, B. Qolomany, and M. Guizani, "Budgeted online selection of candidate iot clients to participate in federated learning," *IEEE Internet of Things Journal*, vol. 8, p. 5938–5952, Apr 2021.

[108] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, Jul 2020.

[109] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *2019 IEEE International Conference on Communications (ICC)*, p. 1–7, May 2019.

[110] M. Tang, X. Ning, Y. Wang, J. Sun, Y. Wang, H. Li, and Y. Chen, "Fedcor: Correlation-based active client selection strategy for heterogeneous federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10102–10111, 2022.

[111] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," *arXiv preprint arXiv:2010.01243*, 2020.

[112] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient federated learning via guided participant selection.," in *OSDI*, pp. 19–35, 2021.

[113] Y. Sun, J. Shao, Y. Mao, J. H. Wang, and J. Zhang, "Semi-decentralized federated edge learning with data and device heterogeneity," *IEEE Transactions on Network and Service Management*, 2023.

[114] B. Le Bars, A. Bellet, M. Tommasi, E. Lavoie, and A.-M. Kermarrec, "Refined convergence and topology learning for decentralized sgd with heterogeneous data," in *International Conference on Artificial Intelligence and Statistics*, pp. 1672–1702, PMLR, 2023.

[115] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE signal processing magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[116] A. Koloskova, S. U. Stich, and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," *arXiv preprint arXiv:1902.00340*, 2019.