RADEL: Resilient and Adaptive Distributed Edge Learning in Dynamic Environments

Qiyuan Wang

Submitted in fulfilment of the requirements for the Degree of Doctor of Philosophy

School of Computing Science
College of Science and Engineering
University of Glasgow

Aug 2025

Abstract

The rapid evolution of edge computing has fundamentally transformed distributed machine learning by enabling intelligence at the network periphery, where data originates. This paradigm shift has facilitated real-time analytics and responsive services across decentralized environments. However, deploying effective machine learning systems at the edge introduces substantial challenges: unpredictable node failures compromise service continuity; evolving data distributions trigger concept drift that degrades model performance; irrelevant data inclusion reduces prediction accuracy; client mobility undermines traditional learning assumptions; and communication inefficiencies limit scalability in resource-constrained settings.

This thesis presents RADEL, a comprehensive framework for Resilient and Adaptive Distributed Edge Learning that systematically addresses these challenges through five interconnected contributions. First, we introduce a novel resilience mechanism that maintains service continuity during node failures by enabling surrogate nodes to effectively serve prediction requests of failing counterparts. Our approach leverages statistical signatures of neighboring nodes' data to build enhanced local models through various information extraction strategies, demonstrating significant improvements over traditional replication-based methods while requiring minimal inter-node data transfer.

Second, we develop maintenance strategies to preserve model resilience under concept drift, a critical challenge in dynamic edge environments where data distributions evolve over time. By analyzing how different types of drift affect enhanced models, we propose efficient maintenance mechanisms that achieve optimal trade-offs between data transmission volume and adaptation effectiveness, maintaining performance across heterogeneous data sources while minimizing communication overhead.

Third, we propose a query-driven data-centric learning approach that progressively discovers relevant data regions from query patterns to optimize predictive performance. This mechanism integrates optimal stopping theory to determine when to conclude model refinement and incorporates adaptive update policies that respond to changes in both data and query distributions. Our experimental results demonstrate up to 63% improvement in predictive accuracy compared to traditional approaches that utilize all available data.

Fourth, we present DA-DPFL, a dynamic aggregation framework for decentralized personalized federated learning that addresses data heterogeneity while reducing communication and computational costs. By enabling clients to reuse previously trained models within the same communication round and employing a sparse-to-sparser pruning strategy based on model compressibility, DA-DPFL achieves superior test accuracy while reducing energy consumption by up to 5× compared to state-of-the-art approaches.

Finally, we introduce MOBILE, a mobility-aware framework that optimizes client selection and bandwidth allocation in federated learning environments with mobile clients. By formulating the problem as a regularized Mixed-Integer Quadratic Programming optimization and incorporating both historical and current mobility patterns, MOBILE increases successful client participation rates from 32% to 89% while reducing wasted bandwidth by 43% compared to mobility-agnostic approaches.

Through comprehensive theoretical analysis and extensive empirical evaluation on diverse real-world datasets, we demonstrate that RADEL significantly improves learning efficiency, reliability, and adaptability in dynamic edge environments. Our research establishes a robust foundation for deploying resilient machine learning services at the edge, advancing the state-of-the-art in distributed intelligence for next-generation computing systems. The frameworks and algorithms developed in this thesis provide practical solutions for the deployment of intelligent services in smart cities, industrial IoT, autonomous systems, and other edge-centric applications where reliability and adaptability are paramount.

Contents

List of Tables

List of Figures

Acknowledgements

Looking back on this challenging but fulfilling PhD journey, it feels like training a complex neural network: every connection, layer, and weight has been shaped by the people who contributed their time, wisdom, and unwavering support. To all of you: thank you for being the backbone of this intellectual architecture.

First and foremost, my deepest gratitude goes to my supervisor, Dr. Christos Anagnostopoulos. Your mentorship has been the optimizer that guided this research toward convergence. Thank you for challenging my assumptions while nurturing curiosity, for balancing rigour with empathy, and for showing me how to turn "what if?" into "here's how." Your trust and belief, even when my own confidence wavered, kept the gradients of progress steady.

To my colleagues and collaborators: you are the distributed nodes in this edge-learning adventure. Special thanks to Dr. Qianyu Long. Your brilliance, late night brainstorming sessions, and willingness to debug both code and ideas made this journey less solitary. Thank you for being my intellectual neighbor in this networked pursuit of knowledge.

My friends, you are the regularization term that prevented overfitting to academic stress. Thank you for dragging me out of work mode, for hikes, dinners, and laughter, for reminding me that life exists beyond loss functions and edge nodes. Special thanks to Dr. Xuri Ge, Weiyun Wang, Xinyu Li, Ke Xiao and many other friends for their valuable support along this journey.

Finally, to my family, you are the training data that shaped my earliest layers. My mother, father and sister, your love is the constant in my loss function. Thank you for weathering my absences and my complaints about the pressure and anxiety I felt. Your support means the world to me.

To the universe of open-source developers, dataset curators, and coffee beans: though unnamed here, your contributions fueled this work.

As I deploy this thesis into the world, I am reminded that no learning system, whether human or machine, can thrive in isolation. Thank you all for being my ensemble model.

Declaration

With the exception of chapters 1 and 2, which contain introductory material, and Chapter 6, which contains the collaborative work the author has done with Dr. Qianyu Long (the author's contributions to the work presented in that chapter are detailed at the beginning of Chapter 6), all work in this thesis was carried out by the author unless otherwise explicitly stated.

Chapter 1

Introduction

Edge Computing has emerged as a transformative computing paradigm that brings computation closer to data sources, enabling real-time decision-making and context-aware services. Concurrently, advances in Artificial Intelligence (AI) have created new opportunities for intelligent applications at the network edge. This thesis explores novel frameworks for resilient and adaptive learning in distributed edge computing environments, addressing key challenges related to node failures, concept drift, query-driven learning, data relevance, and mobility awareness.

1.1 Motivation

1.1.1 The Evolution of Edge Computing and Edge Intelligence

The advancement in communication technologies has paved the way for significant progress in Artificial Intelligence (AI). This surge is primarily due to the colossal volume of data generated by mobile and Internet of Things (IoT) devices that can now be effectively transmitted and accumulated. One major offshoot of these developments is Edge Computing (EC), a novel computing paradigm that shifts the computation closer to the source of data generation—right at the network's edge, near sensors and end-users (Ren et al., 2018). This approach ensures optimal use of resources and enables services that require real-time decision-making, delay-sensitive responses, and are context-aware.

As AI solutions often require substantial data processing capabilities and appreciate minimal latency for real-time services, the integration of AI and EC is emerging swiftly. This amalgamation can primarily be categorized into two segments: 'AI for edge', which aspires to enhance EC using AI techniques, and 'AI on edge', which perceives EC as a utility for the efficient deployment of AI systems (Deng et al., 2020a). A plethora of 'AI on Edge' applications have already shown success, offering a wide range of predictive services such as anomaly detection, regression, classification, and clustering. Notably, by harnessing 'AI on Edge', considerable costs associated with data transmission to the Cloud can be saved (Shi et al., 2016).

The synergy between AI and EC has indeed heralded a new era of technology. By pushing AI capabilities to the edge, we're not only decentralizing intelligence but also ensuring that decision-making is faster, more efficient, and less reliant on distant data centers. However, as with all nascent integrations, some challenges need to be addressed, especially when delving deeper into predictive analytics within the EC framework.

1.1.2 Challenges in Distributed Edge Learning

Predictive analytics, relying heavily on Machine Learning (ML) models, have found significant applications in domains such as smart cities and sustainable agriculture. In these contexts, ML models transition from raw data processing to advanced tasks like regression and classification (Jan et al., 2021). However, when applied in the Edge Computing context, several challenges emerge:

Computing Node Failures and Service Continuity

Conventional ML systems designed for 'AI on Edge' applications typically have each edge node (or *node* for short) accessing its local data and containing ML models trained solely with this data. Despite nodes potentially operating under similar conditions and gathering data from like environments, the unique statistical characteristics of their data render their models non-interchangeable. In realistic EC scenarios, where nodes might fail due to connectivity issues or security breaches, the current system design means that functioning nodes can't readily take over the predictive tasks of their failing counterparts (Wang et al., 2019b).

Traditional approaches to tackle node failures, such as replications, backups, and check-pointing (Borg et al., 1983; Cully et al., 2008; Sherry et al., 2015), are often resource-intensive and may not be suitable for Edge Computing contexts. This necessitates novel approaches that can enhance the adaptability of models on neighboring nodes, enabling them to efficiently handle 'unfamiliar' data from failing nodes.

Data Concept Drift and Model Maintenance

Distributed ML systems operate in dynamic environments, thus requiring their models to be updated according to the novel trends embedded in the new data they encounter. Concept drift, also interchangeably called non-stationary data distributions, is a cause of deteriorating predictive model performances (Webb et al., 2016). As stationary models were built upon prior knowledge (like Machine Learning models), they are expected to not handle the unforeseen changes that happen later on well.

The standard ways of handling concept drift include detecting them, analysing and adapting the models to the new concept (which could include forgetting mechanisms that make the models forget the old information) and estimating the loss (Gama et al., 2014b). However, in contexts where enhanced models operate on multiple nodes, the concept drifts in one node may only partially impact the models' performance. Furthermore, any modifications made to the models may not affect their performance on all the entailed nodes equally. Hence, it is a challenge to approach concept drifts related to enhanced models than typical setups.

Data Relevance and Query-Driven Learning

Building predictive models using entire datasets often yields suboptimal performance because different data regions impact model accuracy unequally. Including irrelevant data hinders predictive capacity, yet identifying relevant data tailored to analytics queries remains challenging. Existing approaches relying on historical queries are computationally expensive and struggle to adapt to changing data and query distributions.

The growing scale and complexity of data create substantial challenges for predictive modeling and analytics. Machine Learning systems must efficiently allocate their resources to handle analytics task requests, which we refer to as *queries*, that require access to *specific* data subsets for effective model building and maintenance (Kraska et al., 2013; Savva et al., 2020a; Verbraeken et al., 2020a).

While distributed learning paradigms (Verbraeken et al., 2020b) including federated learning (McMahan et al., 2017; Yang et al., 2023) can provide satisfactory performance in homogeneous environments, they face limitations in dynamic settings where data distributions and query access patterns vary significantly. In such environments, global models trained over all available data are unaware of the diversity in query access patterns, serving as a 'blanket' over different data and query distributions.

Client Mobility and Resource Allocation

Federated Learning (FL) (McMahan et al., 2017) preserves data privacy by allowing distributed clients to train models locally, sharing only model updates with a central server for aggregation. However, assuming stationary clients is often unrealistic in Mobile Computing (MC), where clients exhibit significant mobility behavior due to irregular connectivity, fluctuating network conditions, and varying computational resources. Client mobility challenges FL assumptions: (i) clients may drop out during training, (ii) connections become unstable, and (iii) resources vary unpredictably. In

MC, intermittent connectivity complicates the synchronization of model update (Lim et al., 2020), affecting the stable client-server relationship conventional in FL.

Clients' sporadic availability due to mobility challenges assumptions of consistent participation and synchronous model updates (Nishio & Yonetani, 2019). Effective resource management and client selection are therefore essential. While previous works have addressed related aspects, few acknowledge that selected clients may fail during learning due to unstable connections or mobility patterns. Moreover, none directly addresses current mobility and mobility patterns of clients along with optimal client selection per round.

Communication Efficiency in Decentralized Learning

Decentralized Federated Learning (DFL) has become popular due to its robustness and avoidance of centralized coordination. In this paradigm, clients actively engage in training by exchanging models with their networked neighbors. However, DFL introduces increased costs in terms of training and communication. Existing methods focus on minimizing communication, often overlooking training efficiency and data heterogeneity (Dai et al., 2022; Sun et al., 2022; Yau & Wai, 2023; Zhao et al., 2022).

While FL frameworks aim at communication and training efficiency, Personalized FL (PFL) addresses statistical heterogeneity, and DFL supports privacy without relying on central servers. However, challenges remain: FL reduces communication cost at the expense of increased training cost associated with gradient/model compression. PFL overlooks communication or training efficiency, while DFL struggles with non-i.i.d. data. This indicates the need for an integrated approach balancing these aspects across different learning paradigms.

1.2 Thesis Statement

This thesis presents novel frameworks and methodologies for resilient and adaptive learning in distributed edge computing environments. It addresses key challenges related to node failures, concept drift, query-driven learning, data relevance, client mobility, and communication efficiency. The central hypothesis is that by developing specialized techniques for each of these challenges, we can create more robust, efficient, and effective distributed learning systems that can operate in dynamic and uncertain environments.

Specifically, this thesis aims to:

1. Enhance the adaptability of models on neighboring nodes to efficiently handle data from failing nodes in edge computing environments.

2. Develop mechanisms for maintaining model performance in the presence of concept drifts in distributed systems.
3. Create data-centric learning approaches for discovering relevant data from query patterns and refining models accordingly.
4. Design frameworks for efficient and personalized federated learning in decentralized environments.
5. Optimize client selection and resource allocation in mobile computing environments based on mobility patterns.

1.3 Contributions

This thesis makes several significant contributions to the field of distributed edge computing:

1.3.1 Resilient Edge-based Predictive Analytics

We introduce a comprehensive framework to enhance the adaptability of models on neighboring nodes, enabling them to efficiently handle 'unfamiliar' data from failing nodes. Our main contributions in this area include:

1. A comprehensive framework to identify the optimal surrogate node during node failures.
2. Innovative strategies for effective information extraction, bolstering model generalizability.
3. A guidance mechanism facilitating node invocation and load balancing during failures.
4. Exhaustive experimental results that underscore the advantages of our approach against traditional methods.

1.3.2 Maintenance of Model Resilience under Concept Drift

We investigate the interactions between enhanced models and concept drifts in distributed edge learning environments. Our contributions include:

1. Methods to maintain the performance of enhanced models in a distributed ML system that operates on multiple data sources in the case of partial concept drifts.
2. Strategies that yield different trade-offs between the amount of data needed to be transmitted and the performance of the maintained models.
3. Comprehensive experimental evaluation of resilience maintenance mechanisms over synthetic and real data.

1.3.3 Query-Driven Predictive Analytics

We propose a novel query-driven data-relevance approach for predictive analytics that progressively discovers relevant data from query patterns to train specialized models. Our technical contributions include:

1. A novel Query-Driven Learning method that identifies relevant data and progressively refines predictive models, discovering regions of interest related to query preferences.
2. A mechanism for determining the optimal learning and refining horizon based on Optimal Stopping Theory.
3. Two mechanisms for maintaining model accuracy when data and query distributions drift, enabling identification of new relevant regions.

1.3.4 Dynamic Aggregation & Decentralized Personalized Federated Learning

We propose a novel Dynamic Aggregation Decentralized Personalized Federated Learning (DA-DPFL) framework that reduces communication and training costs, expedites convergence, and overcomes data heterogeneity. Our contributions include:

1. An innovative alignment of a dynamic aggregation framework that allows clients to reuse previous models for local training within the same communication round.
2. A further pruning strategy that effectively accommodates and extends existing sparse training techniques in Decentralized Federated Learning.
3. Comprehensive experiments showcasing that DA-DPFL achieves comparative or superior model performance across various tasks and DNN architectures.

1.3.5 Mobility and Outage-Based Intelligent Federated Learning

We introduce the Mobility & Outage-Based Intelligent Federated Learning (MOBILE) framework that exploits FL within the uncertainty landscape due to mobility, stochastic client selection, and optimized bandwidth allocation per round. Our contributions include:

1. A framework for jointly optimizing client selection and resource allocation with minimal overhead in mobile computing-based FL environments.
2. Formulation of the objective as a regularized Mixed-Integer Quadratic Programming (MIQP) problem under the principle of means-variance optimization on clients' mobility patterns.

3. An adaptive approach to address the cold-start challenge in mobility-aware FL.
4. Comprehensive experimental evaluation against FL baselines on i.i.d. and non-i.i.d data and real-world mobility data.

1.4 Thesis Outline

The remainder of this thesis is organized as follows:

Chapter 2: Background
Provides a comprehensive overview of the core concepts, key techniques, and methods for efficiency in distributed machine learning. It covers distributed statistical learning, distributed deep learning, compression techniques, and knowledge reuse in machine learning.

Chapter 3: Resilient Edge Predictive Analytics by Enhancing Local Models
Introduces a framework for enhancing model adaptability to handle data from failing nodes in edge computing environments.

Chapter 4: Maintenance of Model Resilience in Distributed Edge Learning Environments
Presents strategies for maintaining model performance in the presence of concept drifts in distributed systems.

Chapter 5: Query-Driven Predictive Analytics via Discovered Relevant Data
Proposes a data-centric learning approach for discovering relevant data from query patterns and refining models accordingly.

Chapter 6: Dynamic Aggregation & Decentralized Personalized Federated Learning
Describes a novel framework for efficient and personalized federated learning in decentralized environments.

Chapter 7: MOBILE: Mobility and Outage-Based Intelligent Federated Learning
Presents a framework for optimizing client selection and resource allocation in mobile computing environments.

Chapter 8: Conclusion and Future Work
Summarizes the contributions of the thesis, discusses the implications of the research, and outlines directions for future work.

Chapter 2

Background

2.1 Introduction

This chapter provides the necessary background for understanding the research presented in this thesis. It begins by introducing the core concepts and definitions related to distributed machine learning in edge environments, followed by a discussion of key techniques in distributed statistical learning and distributed deep learning. The chapter then explores methods for efficiency in distributed systems, focusing on compression techniques, knowledge reuse, and adaptive approaches for handling mobility and concept drift. Finally, it establishes the notation and definitions used throughout the thesis.

2.2 Fundamental Concepts

2.2.1 Definitions and Scope

Edge Computing

Edge Computing (EC) is a novel computing paradigm that shifts the computation closer to the source of data generation, right at the edge of the network, near sensors and end-users (Ren et al., 2018; Satyanarayanan, 2017). This approach ensures optimal use of resources and enables services that require real-time decision making, delay-sensitive responses, and are context-aware. The foundational concepts of edge computing can be traced back to cloudlet architectures (Satyanarayanan et al., 2009) and fog computing (Bonomi et al., 2012). The booming developments of the Internet of Things (IoT), in specific, the advances in intelligent sensors, low-energy wireless communication and sensor network technologies, make it possible for a large number of computing devices to be networked through the IoT infrastructure (Li & Liu, 2012; Welbourne et al., 2009).

IoT devices produce an unprecedented amount of data, raising the demand for pushing computation to the edge of the network to save the cost of storing and transferring the data to the Cloud back end and fully utilize the computational power. This gives birth to EC and makes it possible for many applications (e.g., services provided in smart cities) of which the definition and concept are still emerging and have not been reached a consensus by diverse stakeholders (Yin et al., 2015).

EC differs from traditional cloud computing in several key aspects. While cloud computing centralizes data processing and storage in remote data centers, edge computing distributes these functions to the edge of the network, closer to where data is generated. This architectural difference leads to reduced latency, bandwidth usage, and potentially enhanced privacy and security since sensitive data can be processed locally without transmission to the cloud (Ren et al., 2018). Additionally, edge computing offers greater resilience to network outages, as edge nodes can continue to function even when disconnected from the central infrastructure.

Edge Intelligence

As AI solutions often require substantial data processing capabilities and appreciate minimal latency for real-time services, the integration of AI and EC is emerging rapidly. This amalgamation can primarily be categorized into two segments: 'AI for edge', which aspires to enhance EC using AI techniques, and 'AI on edge', which perceives EC as a utility for the efficient deployment of AI systems (Deng et al., 2020a, 2020b). Recent advances in edge intelligence include secure inference systems for neural networks at the edge (Shen et al., 2022) and novel architectures for edge networking services (Brown et al., 2024).

A plethora of 'AI on Edge' applications have already shown success, offering a wide range of predictive services such as anomaly detection, regression, classification, and clustering. Notably, by harnessing 'AI on Edge', considerable costs associated with data transmission to the Cloud can be saved (Shi et al., 2016). These environments are heavily dependent on all kinds of predictive analytics (Brauneis & Goodman, 2018) like public transportation analysis (Audu et al., 2019), human spatial activity pattern prediction (Song et al., 2010) and city subway station planning (Bram & McKay, 2005).

Edge Intelligence introduces unique challenges not present in traditional AI deployments. These include resource constraints (limited processing power, memory, and energy), heterogeneity in hardware and software platforms, and intermittent connectivity. Solutions must adapt to these constraints while maintaining acceptable performance levels. Furthermore, the distributed nature of edge computing environments necessitates novel approaches to model training, deployment, and maintenance, particularly in addressing issues like data heterogeneity, concept drift, and model resilience across multiple nodes with varying capabilities (Deng et al., 2020a).

Distributed Machine Learning

Distributed Machine Learning (DML) refers to the process of training and deploying machine learning models across multiple computing devices. In the context of edge computing, DML involves model learning and inference across networking nodes over distributed data. This approach offers several advantages, including improved scalability, reduced latency, and enhanced privacy.

In a DML system, each node in the network typically maintains a local dataset and trains models based on this local data. These local models can then be combined or aggregated to form a global model that captures the patterns and insights from all nodes. However, DML systems face challenges such as heterogeneity in data distributions, communication overhead, and node failures.

DML systems generally fall into two broad categories based on their architecture: centralized and decentralized. In centralized DML, a central server coordinates the learning process, aggregating updates from distributed nodes and disseminating the updated global model. The parameter server architecture represents a foundational approach to centralized DML (Li et al., 2014). In contrast, decentralized DML eliminates the central coordinator, allowing nodes to communicate directly with their neighbors to exchange model updates. This decentralized approach offers enhanced fault tolerance, as the system can continue to function even if some nodes fail, and improved privacy, as sensitive data remains localized (Verbraeken et al., 2020a). Unified architectures have been proposed to handle heterogeneous GPU/CPU clusters in distributed training (Jiang et al., 2020).

Federated Learning

Federated Learning (FL) is a specific approach to DML where models are trained across multiple decentralized edge devices or servers holding local data samples, without exchanging them (McMahan et al., 2017). FL preserves data privacy by allowing distributed clients to train models locally, sharing only model updates with a central server for aggregation.

FL can be categorized into Centralized FL (CFL) and Decentralized FL (DFL) based on the clients' communication methods during training. CFL, exemplified by FedAvg (McMahan et al., 2017), involves a central server coordinating client model aggregation. In contrast, DFL (Yuan et al., 2023) offers privacy enhancements and risk mitigation by enabling direct, dynamic, non-hierarchical client interactions within various network topologies such as line/bus, ring, star, or mesh.

The communication pattern in FL introduces additional considerations not present in traditional distributed learning. Clients may have unreliable connectivity, limited bandwidth, or resource constraints affecting their participation. Moreover, the non-IID (non-independent and identically distributed) nature of data across clients poses challenges for model convergence and generalization. These challenges have spurred research in areas such as client selection strategies, communication-efficient training, and techniques to address statistical heterogeneity (McMahan et al., 2017; Verbraeken et al., 2020b).

The theoretical foundations of federated learning were established through federated optimization (Konečný et al., 2015), which formalized the distributed optimization problem beyond traditional datacenter settings.

Personalized Federated Learning

Personalized Federated Learning (PFL) emerges to allow a local (personalized) model per client rather than a global one shared among clients. Although PFL remains in its early development stages, a plethora of works (Dai et al., 2022; Huang et al., 2022, 2023; Li et al., 2021a; Li et al., 2021b; Wang et al., 2023; Zhang et al., 2021) shows its efficiency in data heterogeneity.

In the context of PFL, the problem seeks to find the models $\omega_k, \forall k \in \mathcal{K}$, that minimize:

$$\min_{\{\omega_k\},k\in[1,K]} f(\{\omega_k\}_{k=1}^{K}) = \frac{1}{K}\sum_{k=1}^{K} F_k(\omega_k), \tag{2.1}$$

where $F_k(\omega_k) = \mathbb{E}[\mathcal{L}(\omega_k;(\mathbf{x},y))|(\mathbf{x},y)\in\mathcal{D}_k]$ with expected loss function $\mathcal{L}(.;.)$ between actual and predicted output given local data $\mathcal{D}_k$ (Dai et al., 2022).

PFL approaches can be broadly categorized into several paradigms: (1) Model mixture approaches, which combine global and local models with client-specific weights (Li et al., 2021b); (2) Feature-based approaches, which personalize specific layers or components of the model while sharing others (Li et al., 2021a); (3) Meta-learning approaches, which aim to learn a model initialization that can quickly adapt to new tasks or clients (Fallah et al., 2020b); and (4) Clustering-based approaches, which group clients with similar data distributions to build specialized models for each cluster (Ghosh et al., 2022). These approaches offer different trade-offs between personalization, communication efficiency, and computational requirements, with the optimal choice depending on the specific application context and constraints.

Concept Drift

Concept drift refers to the change in the statistical properties of the target variable that the model is trying to predict (Webb et al., 2016). This phenomenon is common in real-world environments where data distributions evolve over time. The domains of learning under concept drift could be divided into three sections: concept drift detection, concept drift understanding, and concept drift adaption (Lu et al., 2018).

Since concept drift is known to harm model performance, one typical way to carry out concept drift detection is by monitoring the model's performance in specific windows. This gives birth to many error-based detection methods like Drift Detection Method (DDM) (Gama et al., 2004) and ADaptive WINdowing (ADWIN) (Bifet & Gavalda, 2007).

Recent work has extended concept drift adaptation to federated learning settings (Chen et al., 2024; Jothimurugesan et al., 2023; Panchal et al., 2023). A comprehensive survey of concept drift adaptation methods can be found in (Gama et al., 2014a).

Concept drift can be categorized into several types based on its nature and impact. Virtual drift occurs when the distribution of input features changes while the relationship between features and target remains the same. Actual drift involves changes in the conditional probability of the target given the features. Total drift encompasses changes in both the feature distribution and the feature-target relationship. Additionally, concept drift can be sudden (abrupt change), gradual (slow transition), incremental (small successive changes), or recurring (patterns that reappear periodically) (Gama et al., 2004; Lu et al., 2018). Each type of drift requires different detection and adaptation strategies for effective model maintenance.

Query-Driven Learning

Query-Driven Learning (QDL) encompasses discovering query access patterns over data to leverage query-driven analytics (Huang et al., 2018; Savva et al., 2020a, 2020b). QDL specifically focuses on extracting knowledge from queries. QDL can be adopted to exploit end-users' queries for discovering data regions of interest, which can then be used to train, refine, and progressively improve model performance.

In the QDL paradigm, several approaches discover sub-regions of interest within larger datasets. These methods benefit large-scale data visualization and exploratory analytics (Bethel et al., 2006; Meyer et al., 2008; Rübel et al., 2012), natural language processing (Fang et al., 2021), and database optimization (Chen et al., 2002).

The core insight behind QDL is that not all data points contribute equally to model performance for specific query patterns. By analyzing query patterns, QDL methods can identify data regions that are frequently accessed or most relevant to the queries of interest. This enables more efficient use of computational resources and potentially improves model accuracy by focusing training on the most relevant data. Furthermore, QDL facilitates adaptive model refinement as query patterns evolve over time, allowing the system to maintain optimal performance in dynamic environments (Savva et al., 2020a, 2020b).

Data-Centric Learning

Data-Centric Learning (DCL) represents a paradigm shift that focuses on training models with quality data rather than constantly modifying model structures as in traditional model-centric approaches (Jarrahi et al., 2022). The quality of data significantly impacts model performance (Subramonyam et al., 2021) since data are more versatile (Miranda, 2021), while models suffer from concept drifts and limited applicability (Sambasivan et al., 2021).

DCL emphasizes collecting and identifying data relevant to training models (Zha et al., 2023), which helps alleviate model overfitting (Cao et al., 2022; Cocarascu et al., 2020) and improves model maintenance (Anik & Bunt, 2021). Building models on identified relevant data results in more accurate and tailored models.

The data-centric approach offers several advantages over the traditional model-centric paradigm. First, it acknowledges that data quality issues such as noise, imbalance, and incompleteness often have a more significant impact on model performance than model architecture refinements. Second, it promotes better understanding and documentation of data characteristics, leading to more transparent and interpretable models. Third, it facilitates more systematic approaches to model development and maintenance, as improvements in data quality can benefit multiple models and applications simultaneously (Jarrahi et al., 2022; Zha et al., 2023). This shift towards data-centricity is particularly relevant in distributed environments where data heterogeneity and quality variances across nodes can significantly impact the overall system performance.

Mobility-Aware Computing

Mobility-aware computing refers to systems that are designed to consider the movement patterns and location changes of computing devices or users. In the context of distributed learning at the edge, mobility introduces additional challenges such as intermittent connectivity, varying network conditions, and unpredictable resource availability (Lim et al., 2020). These challenges can significantly impact the stability and performance of distributed learning processes.

Traditional distributed learning approaches often assume stationary or semi-stationary clients, which is unrealistic in many mobile computing scenarios. Mobile clients may experience frequent handovers between access points, changes in signal strength, and periods of disconnection. These mobility-induced challenges necessitate specialized techniques for client selection, resource allocation, and communication optimization to maintain effective learning (El Mokadem et al., 2023).

Mobility patterns can be characterized through various metrics and models, including spatial-temporal trajectories, dwell times, transition probabilities between locations, and point-of-interest preferences. These patterns contain valuable information that can be exploited to predict future connectivity status, optimize resource allocation, and improve the reliability of distributed learning processes in mobile environments (Yabe et al., 2024).

2.2.2 Fundamentals of DML

In this section, we present a general mathematical formulation for distributed machine learning that will be used throughout the thesis.

Node and Data Representation

Consider a data management system with n nodes indexed in $\mathcal{N} = \{1, \ldots, n\}$. Each node $i \in \mathcal{N}$ maintains a local dataset and serves analytics queries that require access to its data. The local dataset of node i is defined as $\mathcal{D}_i = \mathcal{X}_i \times \mathcal{Y}_i$, where the domain $\mathcal{X}_i \subseteq \mathcal{X} \subset \mathbb{R}^d$ refers to d-dimensional input vectors $\mathbf{x} = [x_1, \ldots, x_d]^\top \in \mathcal{X}_i$ drawn from an input probability $\mathcal{P}_{\mathcal{X}}(\mathbf{x})$, and the domain $\mathcal{Y}_i \subset \mathcal{Y} \subset \mathbb{R}$ refers to output values drawn from an output probability $\mathcal{P}_{\mathcal{Y}}(y)$, where $y \in \mathcal{Y}_i$ (Savva et al., 2020a).

In real-world distributed systems, the datasets across different nodes may exhibit heterogeneity in various dimensions. Statistical heterogeneity refers to differences in the underlying distributions of data, including variations in feature distributions (covariate shift), label distributions (label shift), or conditional probabilities (concept shift). Quantity heterogeneity pertains to differences in the amount of data available at each node, with some nodes potentially having significantly more or fewer samples than others. Quality heterogeneity encompasses variations in data completeness, noise levels, and label accuracy across nodes. These forms of heterogeneity present challenges for distributed learning algorithms and necessitate specialized approaches for effective model training and inference (Verbraeken et al., 2020b).

Model and Prediction

Each node trains its local model $f_i(\mathbf{x}; \mathbf{w}_i)$ over local data $\mathcal{D}_i$, where $\mathbf{w}_i \in \mathcal{W}$ are the model parameters from parameter space $\mathcal{W}$. Given an input $\mathbf{x} \in \mathcal{X}$, the local model predicts the output $\hat{y} = f_i(\mathbf{x})$ with prediction error (loss) $\ell(\mathbf{w}_i, (\mathbf{x}, y))$, where y is the actual output.

A query to model f_i of node $i \in \mathcal{N}$ is represented by an input $\mathbf{q} \in \mathcal{Q} \subseteq \mathcal{X}$ drawn by a query probability distribution $\mathcal{P}_Q$. The query output is the predicted output $\hat{y}(\mathbf{q}) = f_i(\mathbf{q})$ (Savva et al., 2020b).

The choice of model architecture and loss function depends on the specific task and data characteristics. Common model architectures include linear models, decision trees, support vector machines, and neural networks, each with different computational requirements and expressive capabilities. For regression tasks, common loss functions include mean squared error (MSE) or mean absolute error (MAE), while classification tasks typically use cross-entropy loss or hinge loss. The loss function quantifies the discrepancy between the model's predictions and the ground truth labels, guiding the optimization process during model training (Verbraeken et al., 2020a).

Network Topology in DFL

In a distributed federated learning system with K clients indexed by $\mathcal{K} = \{1, 2, \ldots, K\}$, the clients are networked given a topology represented by a graph $\mathcal{G}(\mathcal{K}, \mathbf{V})$, where the adjacency matrix $\mathbf{V} = [v_{i,j}] \in \mathbb{R}^{K \times K}$ (Sun et al., 2022) defines the neighborhood $\mathcal{G}_k$ of client $k \in \mathcal{K}$, i.e., subset of clients that directly communicate with client k, $\mathcal{G}_k = \{i \in \mathcal{K} : v_{i,k} > 0\}$.

An entry $v_{i,k} = 0$ indicates no communication from client i to client k, i.e., $i \notin \mathcal{N}_k$. The topology can be static or dynamic. In the case of dynamic communication among clients, entries in $\mathbf{V}^t$ depend on (discrete) time instance $t \in \mathbb{T} = \{1, 2, \ldots\}$, resulting in a time-varying and non-symmetric network topology via $\mathbf{V}^t = [v^t_{i,j}] \in \mathbb{R}^{K \times K}$ accommodating temporal neighborhood $\mathcal{G}^t_k$ for k-th client (Dai et al., 2022).

Network topology plays a crucial role in determining the convergence properties, communication efficiency, and fault tolerance of distributed learning systems. Common topologies include:

- *Star topology*: Clients communicate only with a central server, forming a typical centralized FL architecture.
- *Ring topology*: Each client communicates only with two neighboring clients, forming a circular chain.
- *Mesh topology*: Clients communicate with multiple neighbors based on a predefined connectivity pattern.
- *Fully connected topology*: Each client communicates with all other clients in the network.

Each topology offers different trade-offs between communication overhead, convergence speed, and resilience to node failures. Additionally, network characteristics such as bandwidth, latency, and reliability influence the design and performance of distributed learning algorithms (Yuan et al., 2023).

2.3 Key Techniques

2.3.1 Distributed Statistical Learning

Distributed statistical learning involves the application of statistical methods in distributed environments. The key challenge in distributed statistical learning is to develop algorithms that can efficiently process data distributed across multiple nodes while maintaining statistical properties such as consistency, efficiency, and robustness.

Resilient Model Design

Resilience in distributed systems is defined as the 'ability of a system to provide an acceptable level of service in the presence of challenges' (Sterbenz et al., 2010). In the context of edge computing, resilience is crucial when facing challenges such as external attacks or internal failures (Beutel et al., 2009; Delic, 2016; Khan et al., 2010; Shao et al., 2015).

Traditional approaches to achieve resilience include methods like replications and backups (Borg et al., 1983; Cully et al., 2008) and checkpointing (Harchol et al., 2020; Mudassar et al., 2022; Sherry et al., 2015; Siavvas & Gelenbe, 2019). However, these approaches are often resource-intensive and may not be suitable for edge computing environments.

A more novel approach involves developing generalized models on nodes, equipping them with the capability to assist their neighboring nodes during instances of failure. This approach draws inspiration from adversarial training (Bai et al., 2021; Shafahi et al., 2020; Wong et al., 2020) and domain adaptation (Daumé III, 2007; Farahani et al., 2021), aiming to create models capable of processing requests from both local and failing nodes.

Model Maintenance under Concept Drift

As models operate in dynamic environments, they require mechanisms to adapt to concept drifts. The standard approaches include error-based detection methods like DDM (Gama et al., 2004) and ADWIN (Bifet & Gavalda, 2007), and adaptation techniques that update the models to the new concept.

In the context of enhanced models operating on multiple nodes, the challenge is to approach concept drifts in a way that considers the partial impact on the models' performance. The goal is to develop strategies that yield different trade-offs between the amount of data needed to be transmitted and the performance of the maintained models.

Model maintenance strategies can be broadly categorized into several approaches:

- *Model retraining*: Periodically retraining the model from scratch using current data to capture the latest patterns and distributions (Gama et al., 2014a).
- *Incremental learning*: Continuously updating the model using new data while preserving knowledge learned from historical data (Domingos & Hulten, 2000).
- *Ensemble methods*: Maintaining multiple models trained on different time windows and combining their predictions to handle various drift scenarios (Gomes et al., 2017).
- *Transfer learning*: Adapting pre-trained models to new distributions by fine-tuning specific components while preserving general knowledge (Pan & Yang, 2009).

In distributed environments, these maintenance strategies must consider additional factors such as communication overhead, resource constraints, and coordination across nodes. Furthermore, the partial impact of concept drift on enhanced models that operate across multiple nodes necessitates specialized approaches that can maintain performance across different data distributions while minimizing data transfer requirements (Gama et al., 2014b).

Query-Driven Relevant Data Discovery

Query-driven relevant data discovery involves learning from query patterns to identify data regions that contribute positively to model performance. This approach integrates Data-Centric Learning (DCL) (Chai et al., 2022) and Query-Driven Learning (QDL) (Settles, 2009) principles.

The process begins by defining relevance in terms of the degree to which data contributes positively to the performance of the model for specific query patterns. Relevant data regions are then identified based on query probability distributions.

Mechanisms for detecting changes in data and query distributions are also essential to maintain model accuracy when distributions drift (Gama et al., 2014a). These mechanisms enable the identification of new relevant regions as query patterns evolve over time.

The query-driven approach offers several advantages over traditional data selection methods:

- *Efficiency*: By focusing on data regions that are most frequently queried or most relevant to query patterns (Agrawal et al., 2000), the approach reduces the computational resources required for model training and inference.

- *Adaptability*: As query patterns evolve, the system can dynamically adjust its understanding of relevant data regions, ensuring continued optimal performance.
- *Tailored performance*: Models trained on relevant data regions provide more accurate responses to queries following observed access patterns, improving user experience.
- *Resource optimization*: In resource-constrained environments like edge computing, selectively using only relevant data allows for more efficient use of limited storage and processing capabilities.

Implementing query-driven relevant data discovery typically involves progressive learning mechanisms that refine the understanding of relevance based on incoming queries, optimal stopping criteria to determine when to conclude the refinement process, and adaptation mechanisms that respond to changes in data and query distributions (Savva et al., 2020a, 2020b).

2.3.2 Distributed Deep Learning

Distributed deep learning extends the principles of distributed statistical learning to deep neural networks, addressing challenges related to model size, computational requirements, and data distribution.

2.3.3 Federated Learning Approaches

Federated Learning (FL) allows distributed clients to train models locally, sharing only model updates with a central server for aggregation. The seminal FedAvg algorithm (McMahan et al., 2017) is outlined in Algorithm 1.

Beyond FedAvg, several key federated learning algorithms have emerged to address specific challenges in distributed deep learning:

- *FedProx* (Li et al., 2020): Extends FedAvg with a proximal term to improve convergence stability under data heterogeneity.
- *SCAFFOLD* (Karimireddy et al., 2021): Introduces control variates to correct for client drift in heterogeneous settings.
- *FedNova* (Wang et al., 2020c): Normalizes local updates to account for varying numbers of local iterations across clients.
- *FedOpt* (Reddi et al., 2021): Applies server-side optimization algorithms to improve convergence and performance.
- *FedBN* (Li et al., 2021c): Keeps batch normalization statistics local to address feature distribution shifts across clients.

Algorithm 1 Federated Averaging (FedAvg)

```
1: Input: $K$ clients with local datasets $\{\mathcal{D}_k\}_{k=1}^K$, number of rounds $T$, number of local
   epochs $E$, learning rate $\eta$
2: Output: Global model $w^T$
3: Initialize global model $w^0$
4: for $t = 0, 1, \ldots, T-1$ do
5:     Server selects a subset $\mathcal{S}_t \subset \{1, 2, \ldots, K\}$ of clients
6:     for each client $k \in \mathcal{S}_t$ in parallel do
7:         Set $w_k^t \leftarrow w^t$
8:         for local epoch $e = 1, 2, \ldots, E$ do
9:             for batch $b$ of local data $\mathcal{D}_k$ do
10:                $w_k^t \leftarrow w_k^t - \eta \nabla \ell(w_k^t; b)$
11:            end for
12:        end for
13:        Client $k$ sends $w_k^t$ back to server
14:    end for
15:    Server aggregates: $w^{t+1} \leftarrow \frac{1}{|\mathcal{S}_t|} \sum_{k \in \mathcal{S}_t} w_k^t$
16: end for
17: return $w^T$
```

These approaches represent different ways of balancing the trade-offs inherent in federated learning, including convergence speed, model performance, communication efficiency, and robustness to data heterogeneity. The choice of algorithm depends on the specific requirements and constraints of the application context.

Personalized Federated Learning

Personalized Federated Learning (PFL) addresses statistical heterogeneity by allowing each client to maintain a personalized model tailored to its local data distribution. Several approaches have been proposed for PFL, including:

- FedMask (Li et al., 2021a) and FedSpa (Huang et al., 2022), which use personalized masks to customize sparse local models.
- Ditto (Li et al., 2021b), which offers a fair personalization framework through global-regularized multitask FL.
- FOMO (Zhang et al., 2021), which focuses on first-order optimization for personalized learning.
- FedABC (Wang et al., 2023), which employs a 'one-vs-all' strategy and binary classification loss for class imbalance.
- FedSLR (Huang et al., 2023), which integrates low-rank global knowledge for efficient downloading during communication.

In the context of pruning-based PFL, the goal is to find a global model ω and individual masks $\mathbf{m}_k, \forall k \in \mathcal{K}$ that minimize:

$$\min_{\omega, \{\mathbf{m}_k\}, k \in [1,K]} f(\{\omega_k\}_{k=1}^{K}) = \frac{1}{K} \sum_{k=1}^{K} F_k(\omega \odot \mathbf{m}_k), \tag{2.2}$$

where $F_k(\omega \odot \mathbf{m}_k) = \mathbb{E}[\mathcal{L}(\omega \odot \mathbf{m}_k; (\mathbf{x}, y)) | (\mathbf{x}, y) \in \mathcal{D}_k]$ (Dai et al., 2022).

The personalization spectrum in federated learning can be viewed as a continuum, ranging from fully personalized models (trained exclusively on local data with no collaboration) to fully global models (trained collaboratively with no personalization). Between these extremes lie various approaches that balance personalization and collaboration to different degrees:

- *Local fine-tuning*: Starting with a global model and adapting it to local data through additional training (Wang et al., 2020b).
- *Interpolation*: Combining global and local models through weighted averaging (Deng et al., 2020c).
- *Meta-learning*: Learning a meta-model that can quickly adapt to individual clients with minimal fine-tuning (Fallah et al., 2020a).
- *Multi-task learning*: Treating each client as a separate task while sharing representations across clients (Smith et al., 2017).
- *Modular approaches*: Personalizing specific components or layers of the model while sharing others (Collins et al., 2021).

Each approach offers different trade-offs between model performance, communication efficiency, privacy, and computational requirements. The optimal approach depends on factors such as the degree of data heterogeneity, privacy requirements, and resource constraints (Li et al., 2021b; Zhang et al., 2021).

Decentralized Federated Learning

Decentralized Federated Learning (DFL) represents a paradigm shift from centralized federated learning by eliminating the central server bottleneck and enabling direct peer-to-peer collaboration among clients. This architecture fundamentally alters how model updates are aggregated and shared, requiring novel approaches to coordination and convergence.

The absence of a central aggregator in DFL necessitates distributed consensus mechanisms where clients must coordinate model updates through direct communication with their neighbors. This decentralized approach offers compelling advantages: enhanced privacy through elimination of a central data collection point, improved system robustness against single points of failure, and reduced communication bottlenecks that can limit scalability in centralized systems. However, these benefits come at the cost of increased complexity in ensuring convergence guarantees, managing heterogeneous client capabilities, and optimizing communication patterns across dynamic network topologies.

Communication protocols form the backbone of DFL systems, with three primary paradigms emerging in the literature. *Gossip protocols* enable clients to randomly select neighbors for model exchange, creating a natural diffusion process that gradually propagates information throughout the network (Hegedűs et al., 2019; Tang et al., 2022b). *Consensus protocols* take a more structured approach, where clients iteratively average their models with immediate neighbors until approximate consensus is achieved across the entire network. *Dynamic aggregation* protocols allow clients to leverage trained models from neighbors within the same communication round, potentially accelerating convergence by reusing computation rather than starting fresh iterations.

The selection of communication protocols involves critical trade-offs between convergence speed, communication overhead, and resilience to network topology changes. These considerations have driven the development of several sophisticated DFL algorithms, each addressing specific aspects of the decentralized learning challenge.

Early work in DFL focused on adapting centralized algorithms to decentralized settings. DFedAvgM (Sun et al., 2022) extends the foundational FedAvg algorithm to decentralized contexts by incorporating momentum-based SGD, addressing the challenge of maintaining convergence without global coordination. Building on this foundation, BEER (Zhao et al., 2022) enhances convergence properties through a combination of communication compression techniques and gradient tracking mechanisms that help maintain consistency across the network despite the absence of a central coordinator.

Recent advances have addressed more nuanced challenges in DFL. GossipFL (Tang et al., 2022b) introduces bandwidth-aware communication by constructing gossip matrices that account for available communication resources, enabling efficient peer-to-peer exchanges using sparsified gradients. The optimization landscape has been further refined through DFedSAM (Shi et al., 2023), which incorporates Sharpness-Aware Minimization techniques to improve generalization in decentralized settings. DisPFL (Dai et al., 2022) tackles the dual challenge of communication efficiency and model performance by employing RigL-inspired pruning strategies that simultaneously reduce communication costs and generalization error through decentralized sparse training.

The communication efficiency challenge in DFL has motivated several innovative approaches. Wait-free model communication protocols (Bornstein et al., 2022) eliminate synchronization overhead by allowing asynchronous updates, while sketching techniques (Rothchild et al., 2020) provide mathematically principled methods for gradient compression. Topology-aware approaches (Ma et al., 2024) optimize communication patterns by considering the underlying network structure, leading to more efficient information propagation.

The choice of DFL approach ultimately depends on the specific requirements of the deployment scenario, including network connectivity patterns, client computational heterogeneity, privacy constraints, and application-specific performance requirements (Dai et al., 2022; Sun et al., 2022; Yuan et al., 2023). As the field continues to mature, the integration of these various techniques promises to unlock the full potential of decentralized collaborative learning.

Mobility-Aware Federated Learning

Mobility-Aware Federated Learning addresses the challenges introduced by client mobility in FL systems, particularly in mobile computing environments (Lim et al., 2020). Traditional FL approaches often assume stationary clients with stable connections, which is unrealistic in many mobile scenarios where clients move freely, experiencing varying network conditions and potential disconnections.

The primary challenges in mobile FL environments include (Chen et al., 2020):

- *Intermittent connectivity*: Clients may temporarily disconnect due to movement or signal issues.
- *Varying network conditions*: Signal strength, bandwidth, and latency fluctuate as clients move.
- *Resource variability*: Available computational and energy resources may change based on location and usage patterns.
- *Participation uncertainty*: Selected clients may fail to complete training tasks due to mobility-related issues, often referred to as 'stragglers' (Wang et al., 2019c).

To address these challenges, mobility-aware approaches incorporate client mobility patterns into the FL process, particularly for client selection and resource allocation. By considering historical mobility data, current proximity to access points, and mobility correlations among clients, these approaches aim to maximize successful participation while minimizing wasted resources.

The client selection process in mobility-aware FL typically involves analyzing factors such as current distance from access points, historical mobility patterns, and predicted future locations. This information helps identify clients likely to maintain stable connections throughout the training round. Similarly, bandwidth allocation strategies consider clients' mobility characteristics to optimize resource utilization and enhance system efficiency (El Mokadem et al., 2023; Lim et al., 2020).

Recent work has developed mobility-aware coordination algorithms (Macedo et al., 2023) and self-supervised learning approaches for vehicular networks (Gu et al., 2024).

2.4 Methods For Efficiency

2.4.1 Compression Techniques

Compression techniques play a crucial role in reducing communication overhead and storage requirements in distributed learning systems. These techniques can be broadly categorized into gradient compression, model pruning, and quantization.

Gradient Compression

Gradient compression techniques aim to reduce the size of gradient updates transmitted between clients and servers in distributed learning systems. Three notable approaches are:

- LAQ (Sun et al., 2019), which performs gradient quantization to reduce communication costs.
- DGC (Lin et al., 2018), which uses deep gradient compression to significantly reduce the amount of data transmitted during model updates.
- PowerSGD (Vogels et al., 2019), which employs practical low-rank gradient compression for distributed optimization.

Gradient compression operates on the principle that not all gradient information is equally important for model convergence. Many gradient elements can be approximated, quantized, or even omitted without significantly affecting the learning process. Common gradient compression strategies can be categorized into:

- *Sparsification*: Transmitting only the gradient elements with the largest magnitudes and zeroing out the rest (Lin et al., 2018).
- *Quantization*: Reducing the precision of gradient values, e.g., from 32-bit to 8-bit representation (Alistarh et al., 2017; Sun et al., 2019).
- *Low-rank approximation*: Representing gradient matrices using low-rank factorizations (Vogels et al., 2019).
- *Error feedback*: Accumulating the compression error and adding it to future gradients to ensure no information is permanently lost (Seide et al., 2014).

These techniques can reduce communication overhead by 10-100× while maintaining comparable model accuracy when properly implemented. However, they may introduce challenges such as increased convergence time, additional computational overhead for compression/decompression, and potential accuracy degradation if applied too aggressively (Lin et al., 2018; Sun et al., 2019).

Model Pruning

Model pruning involves removing unnecessary weights or connections from a neural network, resulting in a sparse model with reduced storage and computational requirements. Several pruning-based approaches have been proposed for federated learning:

- PruneFL (Jiang et al., 2023), which achieves sparsity in model pruning while maintaining model performance.
- FedDST (Bibikar et al., 2022), which combines model pruning with federated learning to reduce model size and communication overhead.
- FedDIP (Long et al., 2023), which achieves sparsity in model pruning through a dynamic pruning approach.
- pFedGate (Chen et al., 2023), which addresses the challenges by adaptively learning sparse local models with a trainable gating layer, enhancing model capacity and efficiency.
- FedPM (Isik et al., 2023) and FedMask (Li et al., 2021a), which focus on efficient model communication using probability masks.

The individual mask $\mathbf{m}_k$ in pruning-based approaches denotes a pruning operator specific to client k. Given mask $\mathbf{m}_k$, the sparsity $s_k \in [0, 1]$ of $\mathbf{m}_k$ indicates the proportion of zero model weights among all weights (Dai et al., 2022).

Model pruning techniques can be categorized based on several dimensions (Blalock et al., 2020):

- *Pruning granularity*: Weight-level pruning removes individual weights, while structured pruning removes entire structures like filters or channels.
- *Pruning schedule*: One-shot pruning removes weights in a single step, while iterative pruning gradually increases sparsity over multiple steps.
- *Pruning criteria*: Magnitude-based pruning removes weights with small absolute values, while gradient-based pruning considers the impact on the loss function.
- *Pruning timing*: Early-bird pruning identifies and removes redundant weights early in training, while lottery ticket approaches find sparse subnetworks that can be trained from scratch.

Foundational work in neural network pruning includes the lottery ticket hypothesis (Frankle & Carbin, 2019), which demonstrates the existence of sparse, trainable subnetworks. Subsequent work has rethought the value of network pruning (Liu et al., 2019), showing that pruned architectures can be trained from scratch to achieve comparable accuracy.

In federated learning contexts, pruning approaches must balance communication efficiency with model performance across heterogeneous data distributions. Personalized pruning approaches like FedMask allow each client to maintain a customized sparse model tailored to its local data, potentially improving both performance and efficiency simultaneously (Li et al., 2021a; Long et al., 2023).

Quantization

Quantization techniques reduce the precision of model parameters, typically converting floating-point values to lower-precision representations such as 8-bit integers. This results in smaller model sizes and faster computation, albeit with a potential loss in accuracy.

Quantization can be applied to both model parameters and gradients in distributed learning systems, offering a trade-off between communication efficiency and model performance.

Different quantization approaches vary in their precision-performance trade-offs (Jacob et al., 2018; Krishnamoorthi, 2018):

- *Post-training quantization*: Applied after training completion, requiring no retraining but potentially causing larger accuracy drops.
- *Quantization-aware training*: Incorporates quantization effects during training, producing models that are more robust to precision reduction.
- *Mixed-precision quantization*: Applies different precision levels to different model components based on their sensitivity to quantization.
- *Dynamic quantization*: Adapts quantization ranges during inference based on activation statistics.

In federated learning, quantization can significantly reduce communication costs, but must be applied carefully to avoid exacerbating the challenges of statistical heterogeneity. Client-specific quantization approaches that account for local data characteristics may offer advantages over one-size-fits-all solutions, particularly in highly heterogeneous environments (Bibikar et al., 2022).

Recent advances in quantization include SmoothQuant for post-training quantization of large language models (Xiao et al., 2023) and AWQ for activation-aware weight quantization (Lin et al., 2024).

2.4.2 Knowledge Reuse and Transfer

Knowledge reuse and transfer methods enable leveraging existing models or knowledge to improve the efficiency and effectiveness of learning in new contexts or for new tasks.

Transfer Learning and Model Adaptation

Transfer learning involves leveraging knowledge gained from one task or domain to improve learning in another related task or domain. In distributed environments, transfer learning can be particularly valuable for addressing data heterogeneity and resource constraints.

Common transfer learning approaches include (Zhuang et al., 2020):

- *Fine-tuning*: Starting with a pre-trained model and adapting it to a new task through additional training (Girshick et al., 2014).
- *Feature extraction*: Using representations learned by a pre-trained model as features for a new task (Donahue et al., 2014).
- *Domain adaptation*: Modifying models to perform well on target domains with different distributions than the source domain (Daumé III, 2007).
- *Multi-task learning*: Training a model to perform multiple related tasks simultaneously, leveraging shared representations (Caruana, 1997).

In federated learning, transfer learning can be applied to initialize client models with knowledge from related tasks, reducing the number of communication rounds needed for convergence. It can also facilitate personalization by adapting a global model to local data distributions more efficiently (Daumé III, 2007; Farahani et al., 2021).

Knowledge distillation, pioneered by (Hinton et al., 2015), has been adapted for federated learning to enable communication-efficient training (Wu et al., 2022). Group knowledge transfer techniques have been developed specifically for federated learning of large CNNs at the edge (He et al., 2020).

Dynamic Aggregation Techniques

Dynamic aggregation techniques optimize the process of combining model updates in distributed learning systems. Rather than using fixed aggregation schemes, these approaches adapt the aggregation process based on various factors such as data characteristics, model performance, and system conditions.

Key dynamic aggregation strategies include (Kairouz et al., 2021):

- *Weighted aggregation*: Assigning different weights to client updates based on factors such as data quantity, quality, or model performance (McMahan et al., 2017).
- *Selective aggregation*: Including only a subset of client updates in the aggregation process, based on criteria such as update quality or client reliability (Nishio & Yonetani, 2019).
- *Adaptive timing*: Dynamically adjusting the frequency and timing of aggregation rounds based on convergence metrics or system conditions (Xie et al., 2019).

- *Hierarchical aggregation*: Organizing clients into hierarchical structures and performing aggregation at multiple levels to improve scalability (Abad et al., 2020).

Dynamic aggregation is particularly valuable in decentralized environments, where it can help coordinate learning across diverse network topologies and client capabilities. By intelligently scheduling and combining model updates, these techniques can improve convergence speed, model performance, and resource efficiency (Dai et al., 2022; Yuan et al., 2023).

2.4.3 Adaptive Learning Approaches

Adaptive learning approaches dynamically adjust learning strategies based on changing conditions, enabling more efficient and effective learning in dynamic environments.

Resource-Aware Learning

Resource-aware learning approaches consider the computational, communication, and energy constraints of participating devices when designing and executing distributed learning processes. These approaches aim to optimize resource utilization while maintaining acceptable learning performance.

Key considerations in resource-aware learning include (Lim et al., 2020):

- *Computational efficiency*: Adapting model complexity and training procedures based on available computational resources (Howard et al., 2017).
- *Communication efficiency*: Optimizing the frequency, volume, and timing of data exchanges to minimize bandwidth usage (Lin et al., 2018; Vogels et al., 2019).
- *Energy efficiency*: Managing energy consumption to extend device battery life and reduce operational costs (Yang et al., 2020).
- *Storage efficiency*: Minimizing model and data storage requirements to accommodate devices with limited memory (Han et al., 2015).

Resource-aware approaches are particularly important in edge computing environments, where devices may have heterogeneous capabilities and face significant resource constraints. By adapting learning processes to match available resources, these approaches enable more inclusive and sustainable distributed learning systems (Shi et al., 2016; Verbraeken et al., 2020a).

Adaptive Client Selection

Adaptive client selection approaches dynamically decide which clients should participate in each round of distributed learning based on various factors such as data characteristics, resource availability, and expected contribution to model improvement.

Selection criteria may include (Kairouz et al., 2021):

- *Data quality and quantity*: Prioritizing clients with larger or more diverse datasets (Li et al., 2019).
- *Resource availability*: Selecting clients with sufficient computational and communication resources (Nishio & Yonetani, 2019).
- *Reliability*: Favoring clients with consistent participation history (Blanchard et al., 2017).
- *Expected contribution*: Estimating the potential impact of each client's update on model performance (Ning et al., 2024).

Recent advances in client selection include PyramidFL for fine-grained selection (Li et al., 2022), FedGCS for gradient-based optimization of client selection (Ning et al., 2024), and resource-constrained approaches for edge computing systems (Wang et al., 2019c). Lightweight, event-driven platforms have been developed to handle dynamic client participation (Qi et al., 2024).

Adaptive client selection can improve learning efficiency by focusing resources on the most promising clients, enhance model quality by prioritizing valuable data sources, and increase system reliability by avoiding clients likely to fail during training. These benefits are particularly significant in mobile or resource-constrained environments where client availability and capability may vary considerably (El Mokadem et al., 2023; Lim et al., 2020).

2.5 General Notation and Definitions

To ensure clarity and consistency throughout the thesis, this section establishes the notation and definitions that will be used in subsequent chapters.

Table 2.1: General Notation: System and Data Components

Symbol	**Description**
System Components	
n	Number of nodes in the system (Chapter 3 and 4)
$\mathcal{N}$	Set of all nodes (Chapter 3 and 4)
N_i	Node i (Chapter 3 and 4)
i, j, k	Node/client indices
t	Time/round index, $t \in \{1, 2, \ldots, T\}$
T	Total number of rounds/time horizon
Data and Features	
d	Input data dimensionality
$\mathbf{x}$	Input feature vector, $\mathbf{x} \in \mathbb{R}^d$
y	Output/target variable
$(\mathbf{x}, y)$	Input-output pair
$\mathcal{D}$	Dataset (general)
$\mathcal{D}_i$	Local dataset of node/client i
$\mathcal{X}$	Input/feature space
$\mathcal{Y}$	Output/label space
Neighborhoods and Selection	
$\mathcal{N}_i$	Neighborhood of node i
$\mathcal{G}_k$	Neighborhood of client k (graph notation)
M	Size of neighborhood/number of neighbors
$\mathcal{S}_t$	Selected subset at time t
a_k^t	Binary selection indicator
$\mathbf{a}_t$	Selection vector

Table 2.2: General Notation: Models and Mathematical Operations

Symbol	**Description**
Models and Parameters	
f	Model/function (general)
f_i	Local model of node/client i
$\mathbf{w}$	Model parameters/weights
$\boldsymbol{\omega}$	Model parameters (alternative notation)
$\mathcal{L}$	Loss function
ℓ	Loss function (alternative notation)
η	Learning rate
Mathematical Operators	
$\mathbb{E}[\cdot]$	Expectation operator
$\lvert \cdot \rvert$	Cardinality of a set
$\lVert \cdot \rVert$	Norm operator
$\odot$	Hadamard product
$\top$	Transpose
Performance Metrics	
RMSE	Root Mean Square Error
ϵ	Error/threshold (general)

2.6 Chapter-Specific Notations and Definitions

2.6.1 Chapter 3: Resilient Edge Predictive Analytics

Table 2.3: Chapter 3 Notation: Enhanced Models and Strategies

Symbol	**Description**
Enhanced Models and Data	
$\tilde{f}_i^s$	Enhanced model on node N_i using strategy s
$\bar{D}_i$	Enhanced dataset (mixture of local and external data)
$\Gamma(D_i)$	Selected subset/sample from dataset D_i
α	Sample mixing rate, $\alpha \in (0, 1)$
Strategies	
$\mathcal{S}$	Set of all strategies
s	A specific strategy, $s \in \mathcal{S}$
GS	Global Sampling strategy
NCG	Nearest Centroid Guided strategy
CG	Centroid Guided strategy
WG	Weighted Guided strategy
MD	Model-driven Data strategy
Clustering and Statistics	
K	Number of clusters
$\mathbf{w}_{jk}$	The k-th centroid from dataset D_j
D_{jk}	The k-th cluster of dataset D_j
L_{jk}	Number of data points in cluster D_{jk}
p_{jk}	Probability of sampling from k-th cluster
U_i	Average vectors from top-m centroids
$\boldsymbol{\mu}_i$	Mean vector of dataset D_i
$\boldsymbol{\sigma}_i$	Standard deviation vector
$\hat{\sigma}_j^2$	Unbiased regression variance estimate

Table 2.4: Chapter 3 Notation: Graph Representation and Distance Metrics

Symbol	**Description**
Graph Representation	
$\mathcal{G}(\mathcal{V}, \mathcal{E})$	Directed graph for node substitution
$e_{ij}^{\epsilon,s}$	Edge from N_i to N_j with RMSE ϵ using strategy s
$\delta_H(\cdot,\cdot)$	Hausdorff distance
θ	Closeness threshold for adjacency

2.6.2 Chapter 4: Maintenance of Model Resilience

Table 2.5: Chapter 4 Notation: Concept Drift and Model Maintenance

Symbol	Description
Concept Drift	
$\mathbf{x}', \mathbf{y}'$	Input and output after concept drift
D_i'	Drifted data from node N_i
$D_i^{v'}$	Virtual drifted data
$D_i^{a'}$	Actual drifted data
$D_i^{t'}$	Total drifted data
$P(\cdot)$	Probability distribution
Maintained Models	
$\bar{f}_i$	Enhanced model (strategy unspecified)
$\bar{f}_i'$	Maintained/retrained enhanced model
$\bar{D}_i^s$	Enhanced dataset using strategy s
$\Gamma^s(D_j)$	Sample from D_j using strategy s
Enhanced Centroid Guided (ECG) Strategy	
ECG	Enhanced Centroid Guided strategy
λ	Intensity parameter for ECG
$\hat{\mathcal{X}}_j, \hat{\mathcal{Y}}_j$	Fabricated input/output data
$\bar{\boldsymbol{\sigma}}_j$	Standard Error of Mean (SEM)
cov_i	Covariance matrix for data generation
$\mathbf{c}_i$	Mean/center for data generation
Performance Metrics	
ϵ_t^v	Validation error at round t
$e^*(\mathbf{x})$	Prediction error for model f_T^*
$e_0(\mathbf{x})$	Prediction error for initial model

2.6.3 Chapter 5: Query-Driven Predictive Analytics

Table 2.6: Chapter 5 Notation: Queries and Data Relevance

Symbol	**Description**
Queries and Regions	
$\mathbf{q}$	Query input vector, $\mathbf{q} \in \mathcal{Q}$
$\mathcal{Q}$	Query domain
$\mathcal{P}_Q$	Query probability distribution
$\mathcal{Q}_t$	Set of queries up to time t
$\mathbf{q}_{t,k}$	The k-th query in round t
N_t	Number of queries in round t
Relevant Data	
$\mathcal{D}_R$	Relevant data subset
$\mathcal{D}_I$	Irrelevant data, $\mathcal{D}_I = \mathcal{D} \setminus \mathcal{D}_R$
$\mathcal{D}_{t,k}$	Relevant region for k-th query in round t
$\mathcal{D}_t^*$	Best relevant data up to round t
$\mathcal{X}', \mathcal{Y}'$	Relevant input and output regions
Models	
f_0	Initial model (cold start)
f_t^*	Best model up to round t
f_C	Data relevance discriminator
f_B	Model B (comparison approach)
f^*	Ground truth model

Table 2.7: Chapter 5 Notation: Optimal Stopping and Performance

Symbol	**Description**
Optimal Stopping	
S_T	Cumulative confidence at round T
c	Learning cost per round
ξ	Optimal rate of confidence
q	Loss exponent factor
μ	Expected loss at round 1
k^*	Optimal stopping time
Update Mechanisms	
$\mathcal{I}_k$	Indicator for data updates
$\mathcal{I}'_k$	Indicator for query updates
β	Data-change risk factor
β'	Query-change risk factor
$\mathcal{R}_k$	Reward for data updates
$\mathcal{R}'_k$	Reward for query updates
Performance Metrics	
R_t	Stability metric
R^*_t	Relevance ratio
$\bar{R}^*_t$	Balanced relevance (Tversky's index)
ψ	Percentage of output range

2.6.4 Chapter 6: Dynamic Aggregation in DPFL

Table 2.8: Chapter 6 Notation: Network Topology and Dynamic Aggregation

Symbol	**Description**
Network Topology	
K	Number of clients in the system
$\mathcal{K}$	Set of all clients
$\mathbf{V}$	Adjacency matrix, $\mathbf{V} = [v_{i,j}]$
$\mathbf{V}^t$	Time-varying adjacency matrix
$v_{i,j}$	Connection from client i to j
$\mathcal{G}_k^t$	Neighborhood at time t
$\mathcal{G}_{k+}^t$	Extended neighborhood including k
Masks and Sparsity	
$\mathbf{m}_k$	Binary mask for client k
$\mathbf{m}_k^t$	Mask at round t
s_k	Sparsity level of client k
s^*	Target sparsity level
Dynamic Aggregation	
$\mathcal{N}_k^t$	Reuse index set
π_k^t	Random bijection mapping
$\mathcal{N}_{k(a)}^t$	Prior client subset
$\mathcal{N}_{k(b)}^t$	Posterior client subset
$\mathcal{N}_{k(a)}^{(*)t}$	Truncated prior subset
N	Maximum clients to wait for

Table 2.9: Chapter 6 Notation: Pruning, PQI Parameters, and Cost Analysis

Symbol	**Description**
Pruning	
$\Delta_0^t(k)$	Detection score for client k
δ_{pr}	Pruning threshold
$v_t(k)$	Vote of client k
δ_v	Voting threshold
t^*	First pruning time
$\mathcal{T}$	Set of pruning times
I_τ	Pruning frequency interval
b, c	Pruning parameters
PQI Parameters	
p, q	Norm indices, $0 < p \leq 1 < q$
γ	Scaling factor
η_c	Compression hyperparameter
$I(\cdot)$	PQ Index function
Cost Analysis	
C_{total}	Total cost
C_{energy}	Energy cost
C_{comm}	Communication cost
θ	Time-energy trade-off

2.6.5 Chapter 7: MOBILE Framework

Table 2.10: Chapter 7 Notation: System Components and Communication

Symbol	**Description**
System Components	
K	Number of clients in the system
$\mathcal{K}$	Set of all clients
System Components	
BS	Base Station (server)
RAN	Radio Access Network
w_t	Global model at round t
$w_k^{(t+1)}$	Local model after round t
Communication	
B	Total available bandwidth
b_k^t	Bandwidth fraction for client k
$b_{\min}$	Minimum bandwidth fraction
$b_k^{\prime,t}$	Adjusted bandwidth
$R_{k,t}$	Upload communication rate
$p_{k,t}$	Transmission power
$h_{k,t}$	Channel power gain
$\mathbb{P}$	Noise power
θ	Path loss exponent
G_k	Size of model/gradients
C_t	Communication cost at round t
Time and Selection	
$\mathcal{T}$	Upload deadline
$\tau_{k,t}$	Upload duration
$\mathcal{N}_t'$	Initially selected clients
ψ	Throughput usage
ψ'	Proportion of wasted throughput
ψ_F'	Throughput wasted by failed clients

Table 2.11: Chapter 7 Notation: Mobility and Optimization

Symbol	**Description**
Mobility	
$d_{k,t}$	Distance from BS at time t
$d_{\max,t}$	Maximum distance at round t
δ	BS coverage radius
r_k^t	Return value for client k
$\mathbf{r}_t$	Return vector
Q_t	Covariance matrix of returns
$\mathcal{W}_t$	Sliding window of returns
W	Window size
W^*	Cold-start threshold
Optimization	
ρ	Tunable parameter (Problem 2)
λ	Regularization for covariance
γ	Regularization for selection
MIQP	Mixed-Integer Quadratic Programming

2.7 Conclusions

This chapter has provided a comprehensive overview of the core concepts, key techniques, and methods for efficiency in distributed machine learning that form the foundation for the research presented in this thesis. We have introduced edge computing and edge intelligence, discussed distributed machine learning and federated learning, and explored concept drift, query-driven learning, and mobility-aware computing. We have also presented key techniques in distributed statistical learning and distributed deep learning, as well as methods for efficiency through compression techniques, knowledge reuse, and adaptive learning approaches.

The expanded sections on distributed deep learning, methods for efficiency, and mobility-aware federated learning provide a more balanced and comprehensive background for understanding the research presented in the subsequent chapters. The notation and definitions established in this chapter will be used consistently throughout the thesis to ensure clarity and precision in the presentation of our research.

In the following chapters, we will build upon these foundations to address the challenges of resilient edge predictive analytics, model maintenance under concept drift, query-driven predictive analytics, dynamic aggregation in decentralized personalized federated learning, and mobility-aware federated learning. Each chapter will draw on the concepts, techniques, and notation introduced here to present novel contributions to the field of distributed machine learning at the edge.

Chapter 3

Resilient Edge Predictive Analytics by Enhancing Local Models

3.1 Introduction

Predictive analytics applications deployed in edge computing environments often rely on machine learning (ML) models that are trained on locally collected data at each edge node. Despite nodes potentially operating under similar conditions, the unique statistical characteristics of their local data render their models non-interchangeable. This poses a significant challenge when nodes fail: the remaining nodes cannot effectively take over the predictive tasks of their failing counterparts without access to the failing nodes' data or models (Wang et al., 2019b).

Traditional approaches to address node failures, such as data replication (Borg et al., 1983; Cully et al., 2008) and checkpointing (Sherry et al., 2015), are often resource-intensive and ill-suited for resource-constrained edge environments. These approaches typically require substantial data transfers, contradicting the fundamental principle of edge computing to process data near its source.

In this chapter, we introduce a novel approach that enhances the adaptability of models on neighboring nodes, enabling them to efficiently handle 'unfamiliar' data from failing nodes. Our key insight is that by sharing and integrating statistical signature information during the training process, we can develop enhanced models on each node that can maintain predictive performance even during node failures. This approach provides resilience without requiring extensive data transfers to the cloud or massive replication of models across nodes.

The remainder of this chapter is organized as follows: Section 3.2 formulates the problem and introduces our rationale, Section 3.3 discusses related work and our technical contributions, Section 3.4 details our predictive model resilience strategies, Section 3.5 presents the experimental setup and performance evaluation, Section 3.6 discusses results and limitations, and Section 3.7 concludes the chapter.

3.2 Problem Formulation and Rationale

Consider an edge computing system with n distributed nodes: $\mathcal{N} = \{N_1, \ldots, N_n\}$. Node N_i has its own local data $D_i = \{(\mathbf{x}, y)_\ell\}_{\ell=1}^{L_i}$, with L_i input-output pairs $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$. The input $\mathbf{x} = [x_1, \ldots, x_d]^\top \in \mathbb{R}^d$ is a d-dimensional feature vector, which is assigned to output $y \in \mathcal{Y}$ used for regression (e.g., $\mathcal{Y} \subseteq \mathbb{R}$) or classification predictive tasks (e.g., $\mathcal{Y} \subseteq \{-1, 1\}$). Each node N_i trains a local model $f_i(\mathbf{x})$ on its local data D_i. The neighborhood of N_i, $\mathcal{N}_i \subseteq \mathcal{N} \setminus \{N_i\}$, is a subset of nodes which communicate directly with N_i.

Our rationale centers on training enhanced substitute models on nodes through strategies that can be used in case of failures. For each strategy $s \in \mathcal{S} = \{S_1, \ldots, S_{|\mathcal{S}|}\}$, certain training data and/or statistics on a node come from neighboring nodes; we refer to these externally received training data as *unfamiliar* data (or statistics). Strategy s results in an *enhanced* local model $\tilde{f}_i^s$ on node N_i, which is expected to be more generalizable than the local model f_i in terms of predictability, as it captures statistical features of unfamiliar data from neighboring nodes $N_j \in \mathcal{N}_i$.

When node N_j (with local model f_j) becomes unavailable and receives a prediction query input $\mathbf{x}$, we seek an alternative available node N_i (with enhanced models $\{\tilde{f}_i^s\}$), such that the prediction of the most appropriate model $\tilde{f}_i^{s^*}$ on node N_i for query $\mathbf{x}$ is as accurate as that of node N_j, i.e., $\tilde{f}_i^{s^*}(\mathbf{x}) \approx f_j(\mathbf{x})$. In this case, N_i invokes its enhanced model $\tilde{f}_i^{s^*}$ to serve prediction requests directed to node N_j for as long as N_j remains unavailable (Figure 3.1).

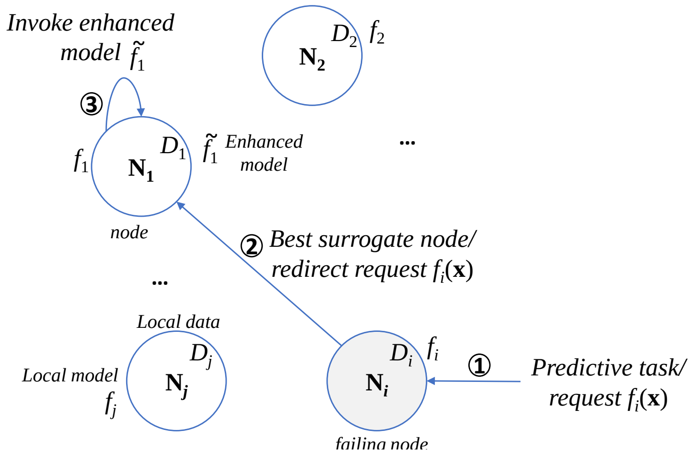

Invoke enhanced model $\tilde{f}_1$
③
f_1
D_1
$\mathbf{N_1}$
$\tilde{f}_1$ Enhanced model
node
D_2 f_2
$\mathbf{N_2}$
...
② Best surrogate node/ redirect request $f_i(\mathbf{x})$
...
Local data
Local model f_j
D_j
$\mathbf{N_j}$
D_i f_i
$\mathbf{N_i}$
failing node
① Predictive task/ request $f_i(\mathbf{x})$

Figure 3.1: Node N_i is failing, thus, (1) any incoming predictive task/request $f_1(\mathbf{x})$ is (2) redirected to the most appropriate surrogate operating node, e.g., N_1. Then, (3) N_1 invokes its enhanced model $\tilde{f}_1^s(\mathbf{x})$ to serve the request w.r.t. the best strategy.

Formally, our problem is to find the best mixture of enhanced models $\{\tilde{f}_i^s\}_{s\in\mathcal{S}}$ across all available nodes $N_i \in \mathcal{N} \setminus \{N_j\}$ and strategies $\mathcal{S}$ to achieve the same quality of predictions as that of the failing node N_j without engaging data transfer to the Cloud:

$$\mathcal{J}_j(\mathcal{S},\mathcal{N}) = \min_{(s,N_i)\in(\mathcal{S}\times\mathcal{N}), i\neq j} \mathbb{E}[(\tilde{f}_i^s(\mathbf{x}) - f_j(\mathbf{x}))^2]. \tag{3.1}$$

Given a strategy $s \in \mathcal{S}$, we obtain specific data and/or statistics from subsets of the datasets $\{D_j\}$, denoted as $\Gamma(\{D_j\})$, and include them in D_i. This results in an enhanced training dataset:

$$\bar{D}_i = D_i \cup \Gamma(\{D_j\}), j \in \mathcal{N}_i. \tag{3.2}$$

We then use $\bar{D}_i$ to train the enhanced model $\tilde{f}_i^s$ for strategy s. Different strategies yield different enhanced models in N_i based on how we select subsets of D_j or specific statistics from N_j.

From a statistical learning perspective, the local model $f_i(\mathbf{x})$ is trained to fit the local dataset D_i, attempting to estimate the conditional probability:

$$p(y|\mathbf{x} \in D_i) = \mathcal{F}_Y(\mathbf{b}_i^\top \mathbf{x}), \tag{3.3}$$

where $\mathcal{F}_Y$ is the cumulative density function of the output y. However, this model cannot effectively capture the structural uncertainty of $p(y|\mathbf{x} \notin D_i)$ for unfamiliar inputs $\mathbf{x} \notin D_i$ (Chatfield, 1995). Our enhanced model $\tilde{f}_i^s$ aims to learn the 'extended' input-output relationship across both familiar and unfamiliar data:

$$\begin{aligned} p(y|\mathbf{x} \in \bar{D}_i) &= \mathcal{F}_{Y,0}(\mathbf{b}_{i,0}^\top \mathbf{x}) I(\mathbf{x} \in D_i) \\ &+ \mathcal{F}_{Y,1}(\mathbf{b}_{i,1}^\top \mathbf{x}) I(\mathbf{x} \notin D_i), \end{aligned} \tag{3.4}$$

where $\mathcal{F}_{Y,0}$ and $\mathcal{F}_{Y,1}$ are the trained/adapted models fitting the node's local and unfamiliar data, respectively, and $I(\cdot) \in \{0, 1\}$ is the indicator function.

Remark 3.1. *The introduction of enhanced models offers significant advantages over model replication approaches for several reasons: (1) Replicating all models across nodes would be prohibitively costly in large-scale systems; (2) Model maintenance would become exponentially more difficult due to concept drifts and data changes; (3) Model replication disregards valuable inter-node relationships discovered through our approach; and (4) Non-parametric models that* *require access to training data during inference cannot be effectively replicated. Therefore, our approach of building enhanced models offers a more efficient and adaptable solution to the node failure problem in edge computing.*

3.3 Related Work and Contributions

3.3.1 Related Work

Resilience in distributed systems is defined as the "ability of a system to provide an acceptable level of service in the presence of challenges" (Sterbenz et al., 2010). In edge computing, resilience is crucial when facing challenges such as external attacks or internal failures (Beutel et al., 2009; Delic, 2016; Khan et al., 2010; Shao et al., 2015).

Traditional approaches to achieve resilience include methods like replications and backups (Borg et al., 1983; Cully et al., 2008) and checkpointing (Harchol et al., 2020; Mudassar et al., 2022; Sherry et al., 2015; Siavvas & Gelenbe, 2019). However, these approaches are often resource-intensive and may not be suitable for resource-constrained edge computing environments.

Our approach draws inspiration from several related fields. Adversarial training techniques (Bai et al., 2021; Shafahi et al., 2020; Wong et al., 2020) enhance model robustness by introducing challenging examples during training. Similarly, domain adaptation methods (Daumé III, 2007; Farahani et al., 2021) address the divergence between training and testing data distributions. These concepts inform our approach to creating models capable of processing requests from both local and failing nodes.

In the broader context of resilience in edge computing, Samanta et al. (Samanta et al., 2021) proposed an auction mechanism to balance user demands and task offloading to edge servers. Other approaches have addressed data stream processing failures (Takao et al., 2021) and real-time messaging architecture to counteract message losses (Wang et al., 2019a). Our framework distinguishes itself by focusing specifically on model adaptability and universality in edge computing environments.

3.3.2 Contributions

This chapter extends our initial findings in (Wang et al., 2022b) with several significant contributions:

1. A comprehensive framework to identify the optimal surrogate node during node failures, using statistical signatures to match failing nodes with the most suitable surrogates.
2. Innovative strategies for effective information extraction that bolster model generalizability without requiring complete data transfer between nodes.

3. A guidance mechanism that facilitates node invocation and load balancing during failures through a directed graph representation.
4. Extensive experimental evaluation across multiple real-world datasets that demonstrates our approach's advantages over traditional methods in maintaining predictive performance during node failures.

3.4 Predictive Model Resilience Strategies

The performance of enhanced models depends directly on the quality of the enhanced training dataset $\bar{D}_i$. To generate $\bar{D}_i$ in an adaptive manner, we propose a collection of strategies tailored to datasets with distinct characteristics. As shown in Figure 3.2, to acquire the enhanced dataset and build the enhanced model, we can take subsets from multiple neighboring nodes.

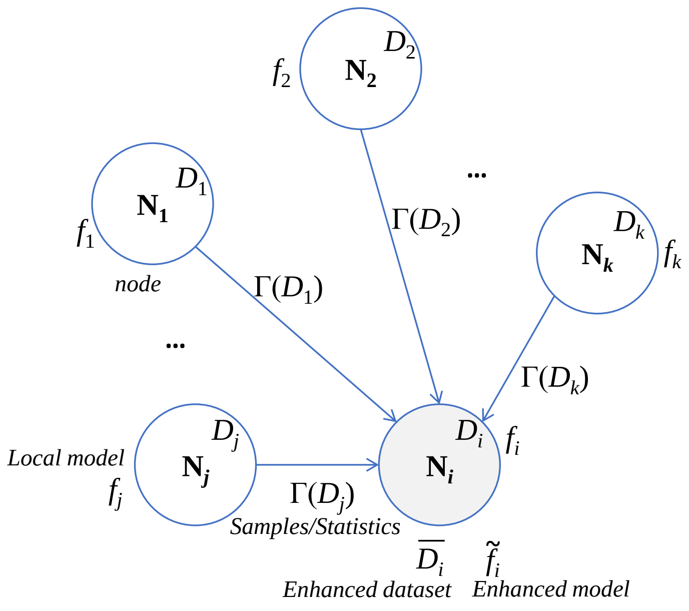


Figure 3.2: Node N_i builds its enhanced model $\tilde{f}_i^s$ based on the best strategy $s \in \mathcal{S}$ by receiving $\Gamma(D_j)$ data samples/statistics from its neighboring nodes $N_j \in \mathcal{N}_i$.

3.4.1 Global Sampling Strategy (GS)

GS is based on random sampling of node N_j's dataset, i.e., $\Gamma(D_j) \subset D_j$. Node N_i receives samples from neighbors' datasets and expands the local dataset as $\bar{D}_i = D_i \cup \{\Gamma(D_j)\}$, $\forall j, j \neq i$. The size of the sample $|\Gamma(D_j)|$ and sample mixing rate $\alpha = \frac{|\Gamma(D_j)|}{|D_j|} \in (0, 1)$ is controlled by N_i, which affects the generalizability of the enhanced model $\tilde{f}_i^S$.

3.4.2 Guided Sampling Strategies

While GS provides a relatively average summary of D_j (making it ideal for evenly distributed datasets), it does not account for the varying characteristics in the sampled data. To address this, we introduce guided sampling strategies that exploit data clustering to selectively capture representative data for training enhanced models.

Clustering Configuration: The clustering-based strategies employ the K-means algorithm to partition the data space. The number of clusters K is determined using the elbow method, which identifies the point where the rate of decrease in within-cluster sum of squares (WCSS) diminishes significantly. Specifically, we select K where adding another cluster provides minimal improvement in WCSS, balancing model complexity with data representation quality. In practice, we constrain $K \in [2, \min(10, \lfloor |D_j|/10 \rfloor)]$ to ensure each cluster contains sufficient samples for meaningful statistics while avoiding over-segmentation of smaller datasets.

Nearest Centroid Guided Strategy (NCG)

NCG quantizes only the input space $\mathcal{X} \subseteq \mathbb{R}^d$ of D_j. The centroids $\mathbf{w}_{jk} \in \mathcal{X}$ convey representative information to the enhanced model's input. For each centroid $\mathbf{w}_{jk}$, we select the m closest input-output pairs $(\mathbf{x}, y) \in D_{jk}$ from the k-th cluster, obtaining the sample:

$$\Gamma(D_{jk}) \quad = \quad \{(\mathbf{x}, y)_\ell \in D_{jk} : d_{(\ell)} = \|\mathbf{x} - \mathbf{w}_{jk}\|\}, \tag{3.5}$$

where $d_{(\ell)}$ is the ℓ-th order statistic of the Euclidean distance between input vector $\mathbf{x}$ and centroid $\mathbf{w}_{kj}$, for $\ell = 1, \ldots, m < L_{jk}$.

Centroid Guided Strategy (CG)

In CG, instead of selecting the nearest pairs to centroids, we select the centroids of clusters that partition the input-output space $\mathcal{X} \times \mathcal{Y}$ of D_j, i.e., centroids $\mathbf{w}_{jk} \in \mathbb{R}^{d+1}$ are samples:

$$\Gamma(D_j) = \cup_{k=1}^{K} \{\mathbf{w}_{jk}\}. \tag{3.6}$$

The CG strategy is particularly useful for applications with data privacy restrictions, as it avoids explicit data transfer among nodes.

Weighted Guided Strategy (WG)

The WG strategy addresses potential negative effects of anomalies in unfamiliar samples by assigning higher probabilities to selecting samples from larger clusters, which are less likely to contain anomalies. After clustering both input $\mathcal{X}$ and output $\mathcal{Y}$ of D_j, we define the probability p_{jk} to be proportional to the number of input-output pairs L_{jk} in cluster D_{jk}:

$$p_{jk} = \frac{L_{jk}}{\sum_{\kappa=1}^{K} L_{j\kappa}}. \tag{3.7}$$

Given a rate α of the data size $|D_j|$, we randomly select $\alpha \cdot p_{jk}$ samples from cluster D_{jk} along with centroid $\mathbf{w}_{jk}$:

$$\Gamma(D_j) = \cup_{k=1}^{K}\{\mathbf{w}_{jk} \cup \{(\mathbf{x}, y) \in D_{jk} : |D_{jk}| = \alpha \cdot p_{jk}\}\}. \tag{3.8}$$

3.4.3 Strategies based on Information Adjacency

The strategies introduced so far treat all neighboring nodes equally in terms of influence and statistical importance. However, this approach may not scale well with many neighboring nodes, and incorporating data from nodes with significantly different distributions might negatively impact the enhanced model's performance.

To address this, we introduce strategy variants based on *information adjacency* that leverage inherent or latent information embedded within the nodes' data. These adjacency variants function exactly like their counterparts except that, when generating the enhanced training dataset $\bar{D}_i$, only the D_j datasets from a pre-determined nodes adjacency list $N_j \in \mathcal{A}_i \subset \mathcal{N}_i$ are considered:

$$\bar{D}_i = D_i \cup \{\Gamma(D_j)\}, N_j \in \mathcal{A}_i \subset \mathcal{N}_i. \tag{3.9}$$

The adjacency list $\mathcal{A}_i$ is determined according to strategies tailored to the characteristics of the nodes' data (Figure 3.3). The adjacency variants include:

Adjacency Global Sampling Strategy (AGS)

In AGS, a neighboring node N_j belongs in $\mathcal{A}_i$ if the Euclidean distance between the mean vector $\boldsymbol{\mu}_j$ of the sample $\Gamma(D_j)$ and $\boldsymbol{\mu}_i$ of D_i is less than a *closeness* threshold θ, i.e., $N_j \in \mathcal{A}_i : \|\boldsymbol{\mu}_j - \boldsymbol{\mu}_i\|_2 \leq \theta$.

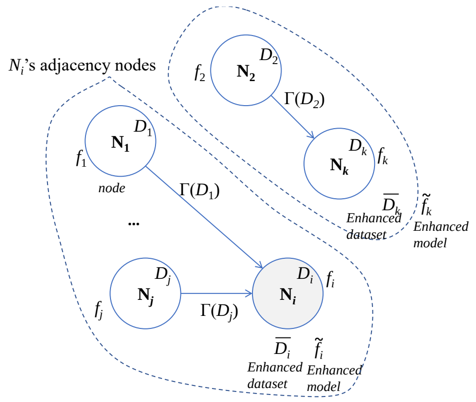


Figure 3.3: Two (groupings) adjacency lists of nodes $\mathcal{A}_i$ and $\mathcal{A}_k$ for node N_i and node N_k, respectively, determined according to data characteristics and dependencies.

Adjacency Nearest Centroid Guided (ANCG) & Adjacency Centroid Guided (ACG) Strategies

For ANCG, we take the average vectors $\mathbf{u}_k$ of the top-m inputs from each centroid, obtaining sets U_j and U_i for nodes N_j and N_i, respectively. Node N_j belongs to list $\mathcal{A}_i$ if the Hausdorff distance between these sets is less than a threshold:

$$N_j \in \mathcal{A}_i : \delta_H(U_j, U_i) \leq \theta. \tag{3.10}$$

Similarly, for ACG, we use the Hausdorff distance between the sets of centroids:

$$N_j \in \mathcal{A}_i : \delta_H(\{\mathbf{w}_{jk}\}_{k=1}^{K}, \{\mathbf{w}_{ik}\}_{k=1}^{K}) \leq \theta. \tag{3.11}$$

Adjacency Weighted Guided Strategy (AWG)

AWG combines ACG and AGS approaches. A node N_j belongs to list $\mathcal{A}_i$ if:

$$N_j \in \mathcal{A}_i : \delta_H(\{\mathbf{w}_{jk}\}_{k=1}^{K}, \{\mathbf{w}_{ik}\}_{k=1}^{K}) \leq \theta \wedge \delta_H(U_j, U_i) \leq \theta. \tag{3.12}$$

3.4.4 Model-driven Data Strategy (MD)

The Model-driven Data (MD) strategy differs from the data-driven strategies above as nodes do not need to disseminate real data. Instead, node N_i generates training data points indirectly from its neighboring nodes' local models f_j, $N_j \in \mathcal{N}_i$.

Node N_j sends its local model parameters f_j along with statistics of the input domain

$\mathcal{X}_j$ (mean vector $\boldsymbol{\mu}_j$, standard deviation $\boldsymbol{\sigma}_j$, and unbiased regression variance estimate $\hat{\sigma}_j^2$) to node N_i. Then, N_i locally generates fabricated training pairs $(\hat{\mathbf{x}}, \hat{y})$ such that:

$$\Gamma(D_j) = \{(\hat{\mathbf{x}}, \hat{y}) : \hat{\mathbf{x}} \sim \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\sigma}_j^2), \hat{y} = f_j(\hat{\mathbf{x}}) + \epsilon_j\}. \tag{3.13}$$

where ϵ_j is a Gaussian random variable with expectation zero and variance $\hat{\sigma}_j^2$, and $\hat{\mathbf{x}}$ is randomly sampled from $\mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\sigma}_j^2)$.

The Adjacency Model-driven (AMD) strategy includes node N_j in the adjacency list $\mathcal{A}_i$ if:

$$N_j \in \mathcal{A}_i : \|\boldsymbol{\mu}_j - \boldsymbol{\mu}_i\|_2 \leq \theta \wedge |\boldsymbol{\sigma}_i - \boldsymbol{\sigma}_j| \leq \theta. \tag{3.14}$$

3.5 Experimental Evaluation

3.5.1 Datasets and Experimental Setup

We evaluated our framework using three datasets that represent different distributed computing scenarios:

1. **GNFUV** (Harth & Anagnostopoulos, 2018b): Contains temperature and humidity readings from sensors mounted on four Unmanned Surface Vehicles (USVs) monitoring the sea surface in a coastal area. Each node corresponds to a USV, with data exhibiting different distributions while bearing spatiotemporal correlation. We used temperature as the input variable to predict humidity.
2. **Accelerometer** (Sampaio et al., 2019): Contains accelerometer readings attached to a fan on x, y, and z axes, used to predict fan speed. We split the data into three parts based on different weight setups, assigning each part to a separate node.
3. **CCPP** (Tüfekci, 2014): Contains environmental sensor readings used to predict power plant output. We clustered the data using Gaussian Mixture Models and assigned the clusters to three nodes.

For predictive models, we adopted Support Vector Regression (SVR) across all nodes. We evaluated different strategies with various mixing rates $\alpha \in \{0.02, \ldots, 0.2\}$ and measured performance using Root Mean Square Error (RMSE).

3.5.2 Model Performance Assessment

We systematically assessed the performance of enhanced models trained with different parameters for each node. For each strategy and mixing rate, we built the corresponding enhanced datasets $\bar{D}_{i,s}$ and trained enhanced SVR models $\tilde{f}_i^s$.

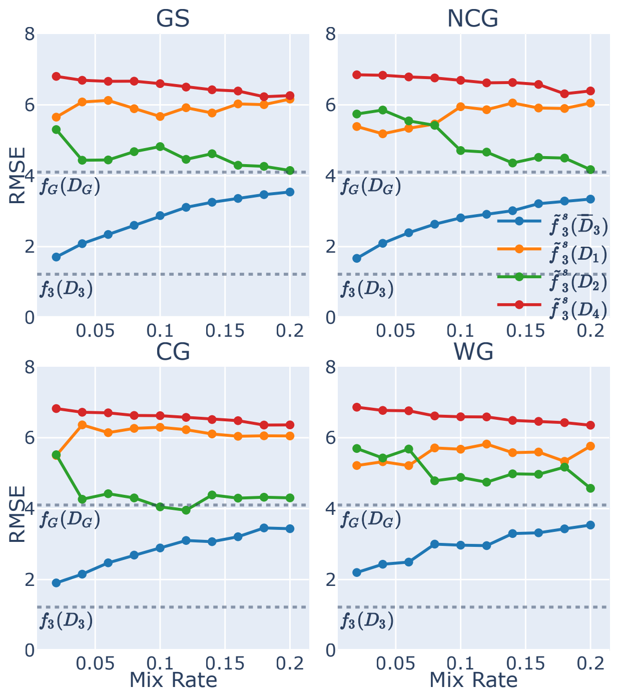


Figure 3.4: GNFUV: Sensitivity analysis of the α (mix rate) on the performance evaluation of the local model $f_3(D_3)$, Cloud model $f_G(D_G)$ and the enhanced models $\tilde{f}_3^s(D_i)$ in N_3 under the four strategies (GS,NCG,CG,WG).

For comparison, we evaluated (i) the Cloud model f_G trained on all nodes' data, and (ii) the local models f_i trained only on local data D_i. Figure 3.4 shows results for node N_3 in the GNFUV setting. The results indicate that N_3 is not a good substitute for nodes N_1 and N_4 (errors above baseline), but it is appropriate for node N_2 (errors below baseline for certain α values with NCG and CG strategies).

3.5.3 Resilience Policy Evaluation

Decision Making & Assignment Resilience Policy

Based on our model performance assessment, we constructed a directed graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ to guide node invocation during failures (Figure 3.5). Vertices represent edge nodes plus one Cloud vertex. A directed edge $e_{ij}^{\epsilon,s} \in \mathcal{E}$ from node N_i to node N_j indicates that if node N_j fails, node N_i could serve as a substitute with RMSE ϵ using strategy s.

To evaluate system performance with node failures, we considered three assignment resilience policies:

1. **Random Substitute Assignment** (zero guidance): Requests to failing nodes are redirected to randomly chosen available nodes, which use their local models.

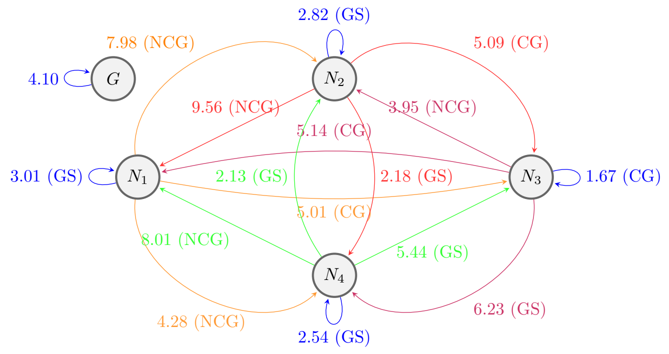


Figure 3.5: GNFUV: Directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ guiding the decision-making for the most appropriate substitute node and strategy upon node failures.

2. **Random Substitute Assignment with best Enhanced Model** (half guidance): Requests are redirected to randomly chosen available nodes, which use their best enhanced models.
3. **Guided Substitute Assignment** (full guidance): Requests are redirected to the most appropriate substitute nodes according to the directed graph, which use their best enhanced models.

Predictive Accuracy Results

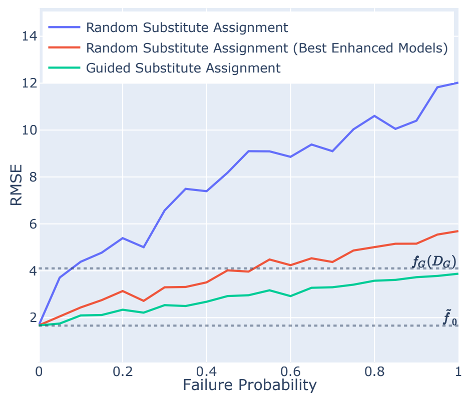


Figure 3.6: GNFUV: System performance against node failure probability p across different node assignment resilience policies.

Figure 3.6 shows the prediction accuracy results for different resilience policies against node failure probability p for the GNFUV dataset. The guided substitute assignment consistently outperforms the Cloud-based policy, even when the failure probability reaches 100%. When $p < 20\%$, the system performance is close to that of the best local enhanced models, indicating that our resilience method maintains performance equivalent to no-failure scenarios at low failure rates.

The half-guidance approach maintains higher predictive performance than the baseline until p reaches about 50%, demonstrating that enhanced models can provide better predictability than Cloud-based approaches even with significant failure rates. Comparing half-guidance to zero-guidance shows that enhanced models reduce RMSE by approximately half.

3.5.4 Load Impact Analysis

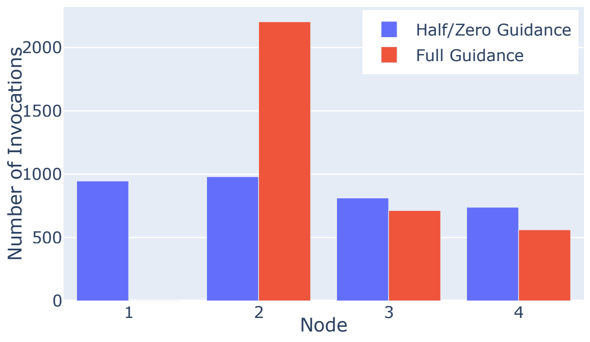


Figure 3.7: GNFUV: Extra load per node in the system with full guidance assignment policy and half/zero guidance policy.

We analyzed the impact of different assignment policies on the load distribution across nodes (Figure 3.7). With the full guidance policy, node loads are relatively balanced, although some imbalance can occur if multiple failing nodes share the same best substitute.

3.5.5 Extended Evaluation with Adjacency Strategies

We extended our evaluation to include the model-driven strategy (MD) and adjacency strategies across all three datasets (Figures 3.8, 3.9, and 3.10).

Based on these extended evaluations, we constructed improved directed graphs for each scenario (Figure 3.11) and evaluated system performance under varying node failure probabilities (Figure 3.12). The results confirm that our framework maintains resilience across different datasets and scenarios, with adjacency strategies providing particular benefits for the GNFUV scenario.

3.6 Discussion

3.6.1 Impact of Information Adjacency Strategies

The effectiveness of information adjacency strategies varies across datasets. In the GNFUV scenario, adjacency strategies yielded significant improvements, with 6 out of 7 improvements achieved by adjacency strategies. For Accelerometer and CCPP, adjacency strategies achieved only one best result each compared to global strategies.

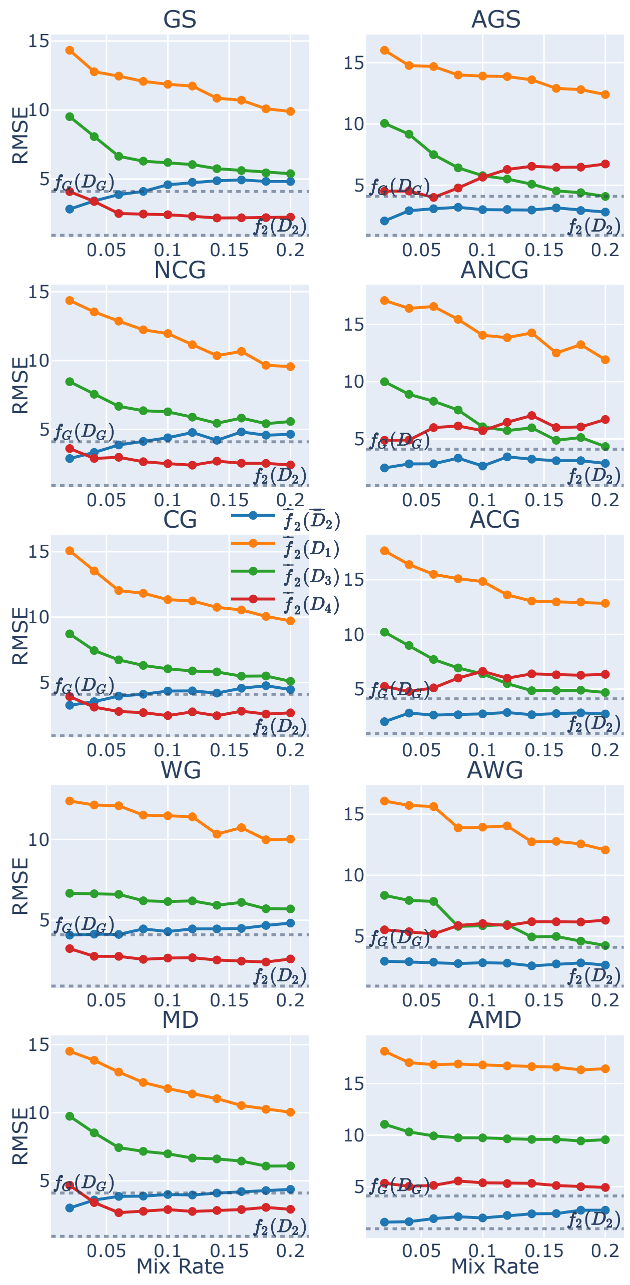


Figure 3.8: GNFUV: Sensitivity analysis on the performance evaluation of the local model $f_2(D_2)$, Cloud model $f_G(D_G)$ and the enhanced models $\bar{f}_2^s(D_i)$ in N_2 under global and adjacency strategies.

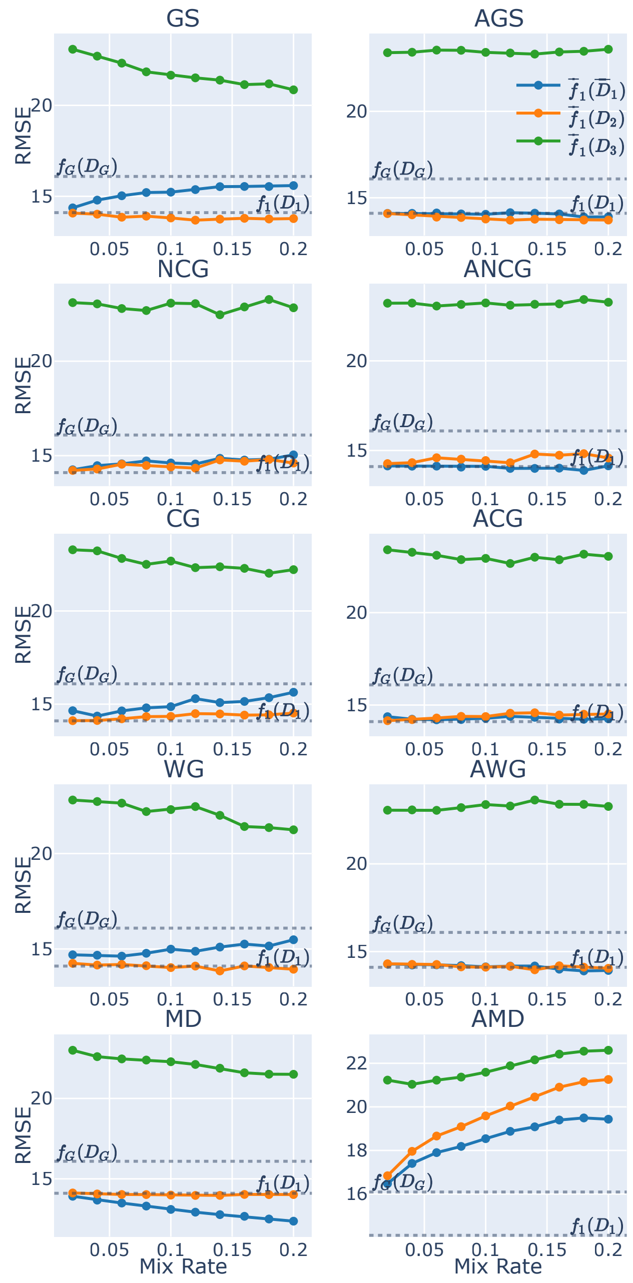


Figure 3.9: Accelerometer: Sensitivity analysis on the performance evaluation of the local model $f_1(D_1)$, Cloud model $f_G(D_G)$ and the enhanced models $\bar{f}_1^s(D_i)$ in N_1 under global and adjacency strategies.

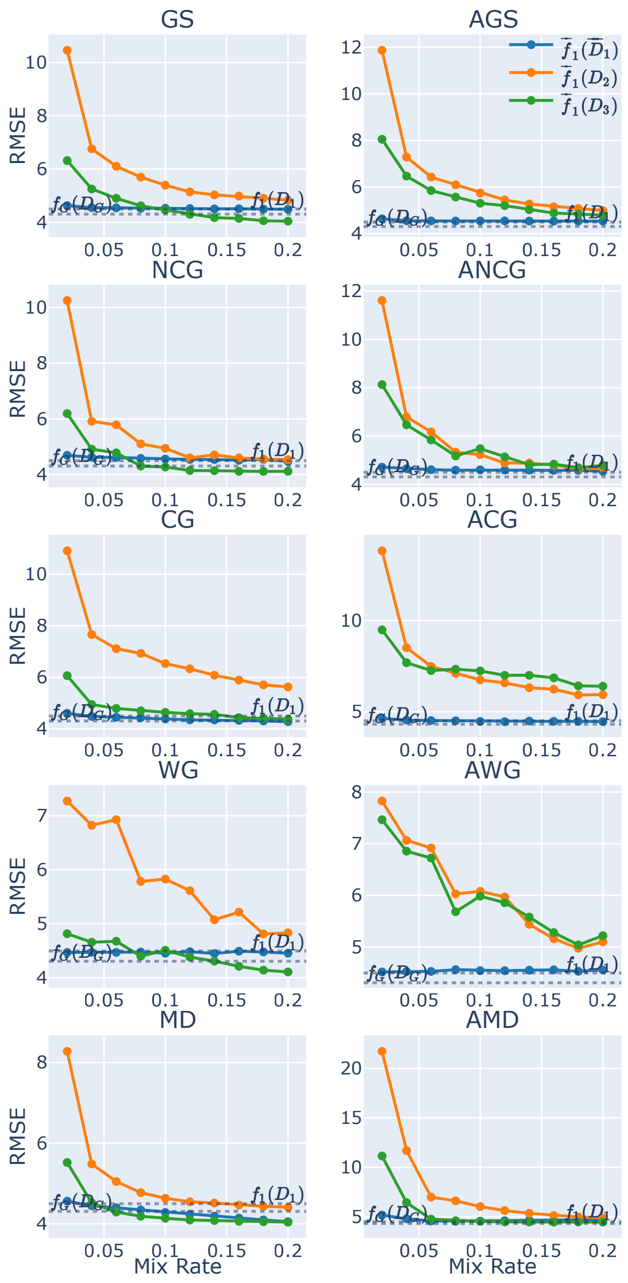
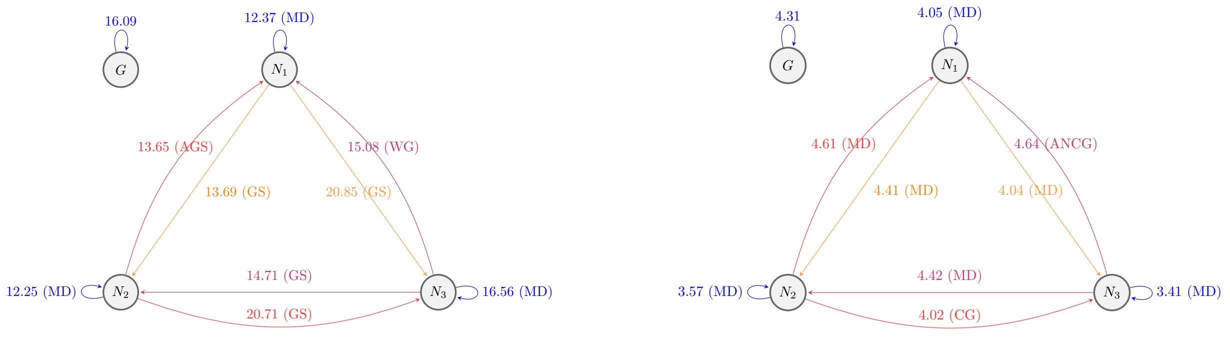
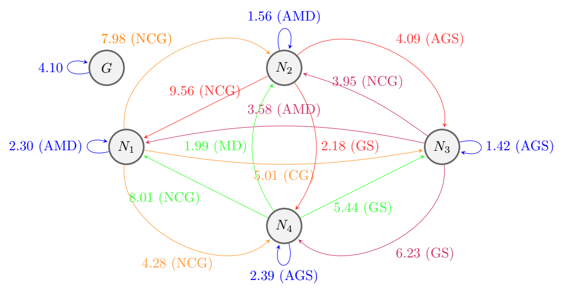


Figure 3.10: CCPP: Sensitivity analysis on the performance evaluation of the local model $f_1(D_1)$, Cloud model $f_G(D_G)$ and the enhanced models $\bar{f}_1^s(D_i)$ in N_1 under global and adjacency strategies.

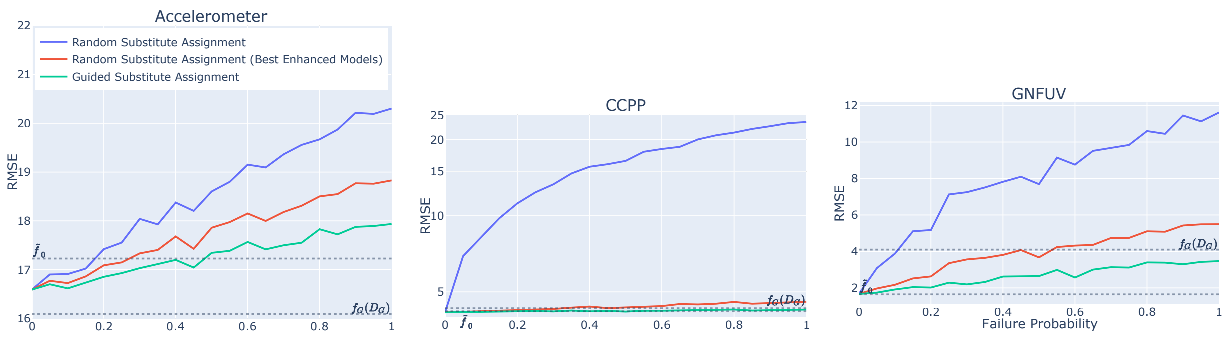


(a) Accelerometer **(b)** CCPP



(c) GNFUV

Figure 3.11: Directed graphs guiding the decision-making for all scenarios adopting the global and adjacency strategies.



(a) Accelerometer **(b)** CCPP **(c)** GNFUV

Figure 3.12: Framework fault-tolerant predictive performance against node failure probability across different node assignment resilience policies for all scenarios.

Notably, 87.5% of the best results achieved by adjacency strategies correspond to AGS and AMD, confirming our expectation that these strategies benefit more from the adjacency lists. Moreover, 67% of these best results are recursive (blue edges in Figure 3.11), indicating that adjacency strategies are particularly efficient at maintaining enhanced models' performance on their local data while still serving adequately as surrogate models for other nodes' data.

This suggests the possibility of equipping each node with just one enhanced model that can handle both local requests and requests from failing nodes with minimal sacrifice in predictive performance.

3.6.2 Model Maintenance Considerations

An important future extension of our framework is effective model maintenance in dynamically changing distributed ML systems. The graph-based guidance should remain largely valid even as data evolves, as it is built on fundamental inter-node data relationships.

When meaningful drifts are detected in a node's data, novel instances can be efficiently sent to nodes with enhanced models that require updating. This approach requires transferring only a few novel instances and retraining a limited number of models, making it well-suited for resource-constrained edge environments.

3.7 Conclusions

We have presented a comprehensive framework for resilient edge predictive analytics that enables operational nodes to effectively serve as surrogates for failing nodes. Our approach leverages statistical signatures of nodes' data to build enhanced models that can handle unfamiliar data, providing resilience without extensive data transfers to the cloud or massive model replication.

The directed graph representation provides an efficient mechanism for guiding request redirection upon node failures, while various sampling strategies offer flexibility to address different data characteristics. Experimental evaluations across multiple real-world datasets demonstrate that our framework maintains—and in some cases improves—predictive performance during node failures, outperforming traditional centralized approaches.

In ideal scenarios, our framework maintains superior predictability even with high failure probabilities, while offering flexibility for load balancing. The adjacency-based strategies further enhance performance by leveraging statistical similarities between nodes, particularly in spatiotemporally correlated data scenarios.

Future work will focus on extending the framework with: (1) intelligent node grouping and enhanced model allocation with multiple options for load balancing, (2) pattern detection and model maintenance mechanisms for concept drift adaptation, and (3) extensions to non-convex optimization tasks such as deep learning.

Chapter 4

Maintenance of Model Resilience in Distributed Edge Learning Environments

4.1 Introduction

As established in Chapter 1, distributed machine learning (DML) at the network edge enables model learning and inference across networked nodes over distributed data. Edge computing environments facilitate real-time predictive analytics services that are critical for intelligent applications such as traffic congestion prediction, smart city infrastructure monitoring, and autonomous systems. The quality of service and availability in these environments directly depend on each node's reliability, as they are highly susceptible to node failures.

In Chapter 3, we introduced a resilience framework that enables each node to identify optimal surrogate nodes and build *enhanced models* to serve requests on behalf of failing nodes with comparable performance. These *enhanced models* are trained primarily on the model's residing node's data, augmented with statistical signatures chosen through specially designed strategies from potential failing nodes. This approach enables the models to operate effectively on data from multiple nodes with minimal inter-node data transfer.

While our previous work demonstrated the enhanced models' capabilities to maintain system reliability even in failure-intensive environments, an important challenge remains: DML systems operate in dynamic environments where data distributions evolve over time, requiring models to adapt to emerging patterns. This chapter addresses the critical question of how our resilience framework behaves in the long term when concept drifts emerge.

Concept drift, also called non-stationary data distribution, is a well-known cause of deteriorating predictive model performance (Webb et al., 2016). Since stationary models are built upon prior knowledge (through ML training), they typically struggle to handle unforeseen distribution changes. Standard approaches to handling concept drift include detection, analysis, and adaptation, sometimes incorporating forgetting mechanisms to unlearn outdated information (Gama et al., 2014b).

However, maintaining enhanced models presents unique challenges compared to traditional setups. Since *enhanced models* operate on multiple nodes, concept drift in one node may only partially impact their performance. Furthermore, any modifications made to these models may affect their performance across different nodes unequally. This creates a more complex scenario than typical cases where drift affects a model holistically.

In this chapter, we investigate the interactions between *enhanced models* and concept drifts. To the best of our knowledge, this issue has not been addressed in previous work. We first analyze how enhanced models with different training strategies respond to various types of concept drift. We then propose strategies to extract signature information from drifted data to effectively retrain the models, evaluating their impact on performance across different data sources. Throughout this process, we comprehensively assess the trade-offs between inter-node data transfer volume and concept drift adaptation effectiveness.

The remainder of this chapter is organized as follows: Section 4.2 discusses related work and our contributions, Section 4.3 formalizes the problem, Section 4.4 demonstrates the challenges of partial concept drift through experimental evaluation, Section 4.5 introduces our model maintenance strategies, Section 4.6 presents performance evaluations and comparative assessments, and Section 4.7 concludes the chapter.

4.2 Related Work and Contributions

Research on learning under concept drift generally falls into three categories: concept drift detection, understanding, and adaptation (Lu et al., 2018).

4.2.1 Concept Drift Detection

Since concept drift typically harms model performance, a common detection approach involves monitoring model performance within specific time windows. This has led to error-based detection methods such as Drift Detection Method (DDM) (Gama et al., 2004) and ADaptive WINdowing (ADWIN) (Bifet & Gavalda, 2007). DDM maintains a dynamic window and monitors significant increases in online error rates, providing warnings when model rebuilding may be necessary or indicating drift when models should be replaced. ADWIN compares model error on two windows of adaptive sizes: one containing historical data and another containing recent data, with the difference in average errors determining if concept drift exists.

4.2.2 Concept Drift Understanding

Understanding concept drift involves evaluating its severity, which directly affects mitigation strategies. Minor drift might be addressed through incremental learning adjustments, while significant drift often requires complete model retraining (Lu et al., 2018). For example, Minku et al. characterized drift severity by measuring the percentage of the input space where target class assignments changed before and after the drift (Minku et al., 2009).

4.2.3 Concept Drift Adaptation

Adaptation involves updating predictive models during operation to respond to drift (Gama et al., 2014b). Existing approaches in this domain are often limited to specific models or tightly coupled with particular detection methods. For instance, the method proposed in (Bach & Maloof, 2008) employs two learners: a stable one that captures long-term information for prediction and a reactive one that learns recent information as a drift indicator. When the stable learner underperforms compared to the reactive one, it indicates concept drift and triggers retraining of the stable learner.

Considering drift severity, Wang et al. (Wang et al., 2022a) proposed an approach that automatically selects optimal retraining and tuning strategies, finding adaptive iterations to maintain Gradient Boosting Decision Tree (GBDT) models based on drift severity. However, this approach is specific to GBDT models. Similarly, CVFDT (Domingos & Hulten, 2000), an extension of the Very Fast Decision Tree (VFDT) classifier (Hulten et al., 2001), along with later extensions (Gama et al., 2003; Ge et al., 2021; Yang & Fong, 2012, 2015), can effectively adapt to new concepts by updating only parts of the model, but these approaches cannot be generally applied to the concept adaptation problem.

4.2.4 Contributions

Due to the lack of flexibility and generalizability in current approaches, none of the existing work addresses the maintenance challenges of enhanced models in DML systems. Our contributions in this chapter include:

- Methods to maintain the performance of enhanced models in a DML system that operates on multiple data sources in the context of partial concept drifts (where drift occurs in only one of the sources).
- Strategies that yield different trade-offs between the amount of data transmitted between nodes and the performance of the maintained models.
- A comprehensive investigation of how maintenance impacts model performance across different data sources.

- Extensive experimental evaluation of resilience maintenance mechanisms using both synthetic and real-world data.

4.3 Problem Definition

Building on the fundamentals established in Chapter 3, we consider a DML system providing predictive services in an edge computing environment with n nodes: $\mathcal{N} = \{N_1, \ldots, N_n\}$. While all nodes perform similar predictive tasks, each works independently on its local data $D_i = \{(\mathbf{x}, y)_\ell\}_{\ell=1}^{L_i}$, with L_i input-output pairs $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$. The input $\mathbf{x} = [x_1, \ldots, x_d]^\top \in \mathbb{R}^d$ is a d-dimensional feature vector assigned to output $y \in \mathcal{Y}$ for regression (e.g., $\mathcal{Y} \subseteq \mathbb{R}$) or classification tasks (e.g., $\mathcal{Y} \subseteq \{-1, 1\}$).

Each node N_i maintains a local model f_i trained exclusively on D_i. The neighborhood of N_i, denoted as $\mathcal{N}_i \subseteq \mathcal{N} \setminus \{N_i\}$, represents nodes that communicate directly with N_i. The collection of data from nodes in $\mathcal{N}_i$ is defined as $\mathcal{D}_i = \{D_j\}, \forall j, j \neq i$.

As introduced in Chapter 3, an *enhanced model* $\bar{f}_i^s$ on node N_i is trained using *enhanced data* $\bar{D}_i^s = D_i \cup \{\Gamma^s(D_j)\}, \forall j, j \neq i$, where $\Gamma^s(D_j)$ represents samples of representative information extracted from N_j using strategy s. For example, with a random sampling strategy, we sample random points from the original data.

For an enhanced model $\bar{f}_i^s$ operating on D_i and $\mathcal{D}_i$, the distribution of a neighboring node's data $D_j \in \mathcal{D}_i$ may change over time due to concept drift. Given an error threshold ε that the system can tolerate (i.e., $|f(\mathbf{x}) - \mathbf{y}| - |f(\mathbf{x}') - \mathbf{y}'| > \varepsilon$ indicates the model must be maintained to adapt to the new concept), the errors produced by all affected models when facing the same concept drift will likely differ. Therefore, not all models affected by the drift necessarily require maintenance.

Problem 1: *We seek to investigate how concept drifts of different degrees affect the performance of different kinds of models, particularly enhanced models.*

While the detection mechanisms and patterns of concept drift (sudden/abrupt, incremental, gradual, or recurring as identified in (Gama et al., 2014b)) are beyond our scope, the types of concept drift are crucial. In this chapter, we address three types of concept drift based on probability distributions. Given that $\mathbf{x}$, $\mathbf{y}$ are the original input and output and $\mathbf{x}'$, $\mathbf{y}'$ are the input and output when concept drift occurs, we define:

- **Virtual Drift:** Changes in the input distribution without affecting the decision boundary:
$$P(\mathbf{x}) \neq P(\mathbf{x}') \wedge P(\mathbf{y} \mid \mathbf{x}) = P(\mathbf{y}' \mid \mathbf{x}')$$
- **Actual Drift:** Changes in the conditional probability distribution (the actual concept or decision boundary):
$$P(\mathbf{y} \mid \mathbf{x}) \neq P(\mathbf{y}' \mid \mathbf{x}')$$

- **Total Drift:** Changes in both input distribution and the conditional probability distribution:

$$P(\mathbf{x}) \neq P(\mathbf{x}') \wedge P(\mathbf{y} \mid \mathbf{x}) \neq P(\mathbf{y}' \mid \mathbf{x}')$$

Problem 2: *Since enhanced models are built with data from multiple nodes and operate across them, we seek strategies to extract statistical information from new drifted data to maintain these models effectively and efficiently.*

Let an enhanced model $\bar{f}_i$ reside on node N_i and concept drift occur on node N_k. Consider a neighboring node N_j whose data D_j has not drifted, where $\{N_k, N_j\} \in \mathcal{N}_i$. The model $\bar{f}_i$ was built on node N_i to handle potential failures of N_k and/or N_j. After maintenance using the drifted data D_k' of node N_k, we obtain the maintained enhanced model $\bar{f}_i'$.

For notational convenience, we denote the expected loss of model f on dataset D as:

$$\mathbb{E}_D[\mathcal{L}(f)] \triangleq \mathbb{E}_{(\mathbf{x},y)\sim D}[\mathcal{L}(f(\mathbf{x}), y)]$$

where the expectation is taken over samples $(\mathbf{x}, y)$ drawn from dataset D.

We can assess the performance of $\bar{f}_i'$ through prediction errors (losses $\mathcal{L}$) on both drifted data D_k' ($\mathbb{E}_{D_k'}[\mathcal{L}(\bar{f}_i')]$) and non-drifted data D_j ($\mathbb{E}_{D_j}[\mathcal{L}(\bar{f}_i')]$). The objectives of the maintenance process are:

O1: Minimize $\mathbb{E}_{D_k'}[\mathcal{L}(\bar{f}_i')]$ (for node N_k).
O2: Minimize $\mathbb{E}_{D_j}[\mathcal{L}(\bar{f}_i')]$ (for node N_j).
O3: Reduce inter-node data transfer between nodes N_i and N_k during model maintenance.

4.4 Impact of Partial Concept Drift

4.4.1 Experimental Setup

To investigate concept drift effects in a controlled environment, we created an artificial dataset comprising three nodes $\{N_1, N_2, N_3\}$ with data generated from Gaussian distributions by manipulating means and covariance matrices (Figure 4.1).

In this setup, D_1 serves as the baseline for generating D_2 and D_3. Given the covariance matrix cov_i and mean $\mathbf{c}_i$ used to generate D_i, we derived cov_2, $\mathbf{c}_2$ and cov_3, $\mathbf{c}_3$ by adding 20-30% random noise to cov_1 and $\mathbf{c}_1$. We introduced concept drift only to D_1, generating three types of drifted data.

For virtual and actual drifted D_1 (denoted as $D_1^{v\prime}$ and $D_1^{a\prime}$), we used the same covariance matrix as cov_1. Following the definitions, virtual drift shifted centers along the x-axis, while actual drift shifted centers along both x- and y-axes. Total drifted

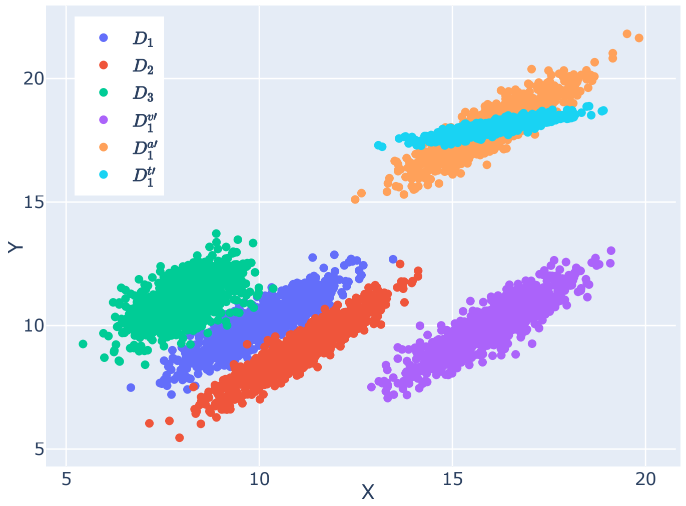


Figure 4.1: Distributions of the artificially generated data across three nodes and their drifted variants.

D_1 (denoted as $D_1^{t\prime}$) shared the same center as actual drifted D_1 but with a different covariance matrix. We shifted drifted data to minimize overlap with D_1, D_2, and D_3 along the x-axis to avoid scenarios where identical inputs map to different outputs.

4.4.2 Enhanced Model Construction

To test enhanced models' responses to concept drift, we focused on models residing on node N_2. These models allow us to understand performance on both a drifted node (N_1) and an unchanged node (N_3). We built enhanced models using two strategies from Chapter 3: Global Sampling (GS) and Centroid Guided (CG).

The GS strategy randomly samples data from neighboring nodes. For each node $N_j \in \mathcal{N}_i$, we acquire $\Gamma(D_j)$ by randomly sampling $\alpha|D_j|$ points, where α controls the sampling proportion. The enhanced dataset is then $\bar{D}_i = D_i \cup \Gamma(D_j), \forall j, j \neq i$.

The CG strategy uses centroids from vector quantization (clustering) of D_j instead of sampling real data. For each $N_j \in \mathcal{N}_i$, we quantize D_j into K clusters based on α and $|D_j|$ (where $K = \alpha|D_j|$). For each cluster, we derive a centroid $\mathbf{w}_{jk}$ representing the cluster center. $\Gamma(D_j)$ is then defined as:

$$\Gamma(D_j) = \cup_{k=1}^{K}\{\mathbf{w}_{jk}\}. \tag{4.1}$$

4.4.3 Effects of Concept Drift on Enhanced Model Performance

We simulated abrupt concept drift at node N_1 by concatenating D_1 with three types of drifted data ($D_1^{v\prime}$, $D_1^{a\prime}$, and $D_1^{t\prime}$) and applied $\bar{f}_2^{GS}$ and $\bar{f}_2^{CG}$ to them. We compared these results with the local model f_1 to assess differential impacts. To control for model choice effects, we trained each enhanced dataset with both Support Vector Regression (SVR) and Gradient Boosting Regression (GBR) models (local models were built with SVR by default).

Table 4.1: Performance of Different Models (RMSE)

Model	D_1	$D_1^{v}{}'$	$D_1^{a}{}'$	$D_1^{t}{}'$
f_1	0.47	1.29	8.06	7.93
$\bar{f}_2^{GS}(SVR)$	1.73	1.69	7.57	7.41
$\bar{f}_2^{CG}(SVR)$	1.69	1.68	7.57	7.41
$\bar{f}_2^{GS}(GBR)$	1.54	2.19	6.26	6.12
$\bar{f}_2^{CG}(GBR)$	1.50	2.17	6.29	6.15

$D_1^{v}{}'$ corresponds to virtual drifted D_1
$D_1^{a}{}'$ corresponds to actual drifted D_1
$D_1^{t}{}'$ corresponds to total drifted D_1

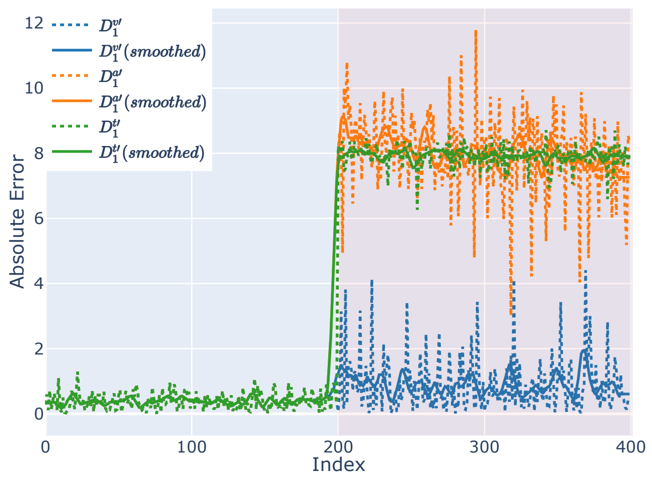


Figure 4.2: Performance of f_1 on D_1, $D_1^{v}{}'$, $D_1^{a}{}'$ and $D_1^{t}{}'$

Table 4.1 shows the RMSE for all models. Figures 4.2 through 4.6 visualize point-wise absolute errors. All models show similar patterns when facing the same type of concept drift (the light red regions on the right represent drifted data).

The impact of concept drift appears largely independent of the strategy used to build enhanced models (Figures 4.3 through 4.6), as GS and CG provided similar results. Compared to the local model f_1 (Figure 4.2), enhanced models—being more generalized—performed better on severe drift types ($D_1^{a}{}'$ and $D_1^{t}{}'$) while performing worse on $D_1^{v}{}'$.

Enhanced models did not perform well on D_1, indicating significant differences between D_1, D_2, and D_3 due to minimal overlap. From the enhanced models' perspective,

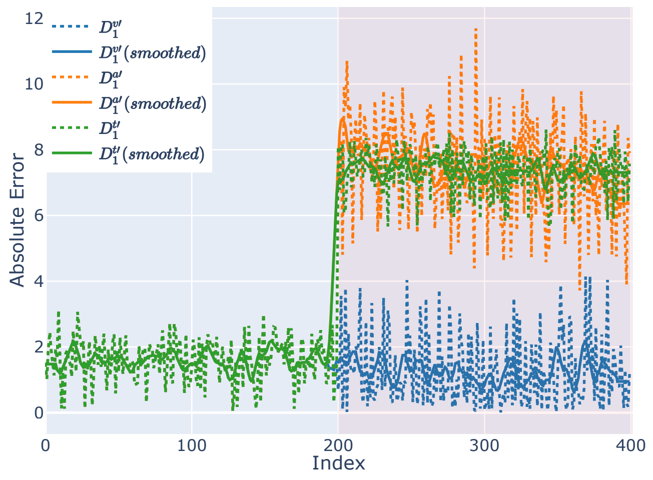


Figure 4.3: Performance of $\bar{f}_2^{GS}(SVR)$ on D_1, $D_1^{v}\prime$, $D_1^{a}\prime$ and $D_1^{t}\prime$

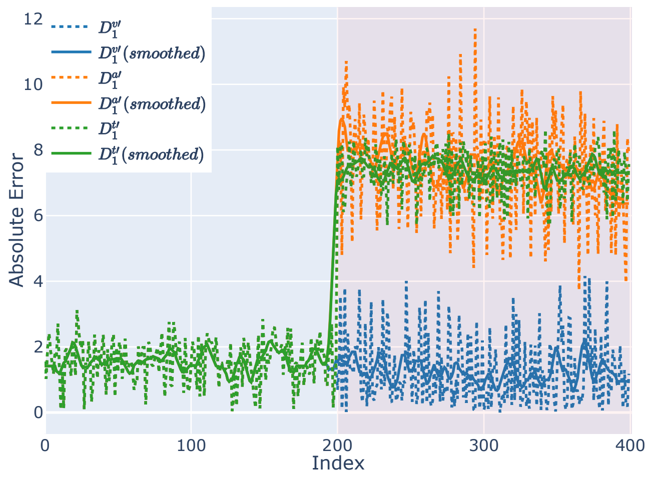


Figure 4.4: Performance of $\bar{f}_2^{CG}(SVR)$ on D_1, $D_1^{v}\prime$, $D_1^{a}\prime$ and $D_1^{t}\prime$

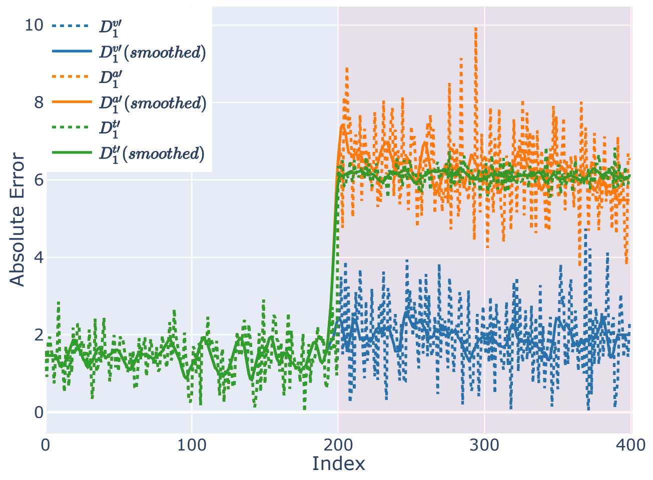


Figure 4.5: Performance of $\bar{f}_2^{GS}(GBR)$ on D_1, $D_1^{v}\prime$, $D_1^{a}\prime$ and $D_1^{t}\prime$

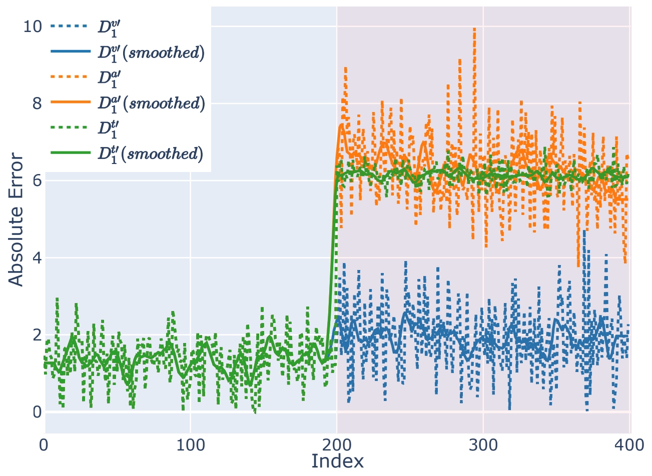


Figure 4.6: Performance of $\bar{f}_2^{CG}(GBR)$ on D_1, $D_1^{v\prime}$, $D_1^{a\prime}$ and $D_1^{t\prime}$

D_1 and $D_1^{v\prime}$ are similar, resulting in comparable performance on both. Additionally, GBR models (Figures 4.5 and 4.6) demonstrated better generalizability, performing noticeably better on $D_1^{a\prime}$ and $D_1^{t\prime}$ compared to SVR models (Figures 4.3 and 4.4).

These results suggest that if we use relative performance drops to trigger model retraining, for $D_1^{v\prime}$, retraining the local model f_1 doesn't necessarily mean enhanced models operating on D_1 need retraining as well. However, for $D_1^{a\prime}$ and $D_1^{t\prime}$, local model and enhanced model reactions align closely.

4.5 Model Maintenance Strategies

To adapt models to new concepts, we sample from drifted data and retrain them. This process mirrors the strategies used to extract signature information and initially train enhanced models, except we sample from drifted data D' instead of initial data D. Thus, GS and CG strategies from Chapter 3 can be directly applied to enhanced model maintenance.

We also introduce two new strategies—Mock Data (MD) and Enhanced Centroid Guided (ECG)—to investigate reducing inter-node data transmission while maintaining effective retraining.

4.5.1 Mock Data (MD) Strategy

The MD strategy avoids real data transmission by requiring only the local model and statistical information. Given input-output pairs from D_j denoted as $\mathcal{X}_j$ and $\mathcal{Y}_j$, we calculate:

1. Average of $\mathcal{X}_j$: $\boldsymbol{\mu}_j = \frac{\sum_{m=1}^{|D_j|} \mathbf{x}_m}{|D_j|} \in \mathbb{R}^d$

2. Standard deviation of $\mathcal{X}_j$: $\boldsymbol{\sigma_j} = \sqrt{\frac{1}{|D_j|}\sum_{m=1}^{|D_j|}(\mathbf{x}_m - \mu_j)^2} \in \mathbb{R}^d$
3. Standard Error of the Mean (SEM) on $\mathcal{Y}_j$:

$$\bar{\boldsymbol{\sigma}}_{\boldsymbol{j}} = \frac{\sqrt{\frac{1}{|D_j|}\sum_{m=1}^{|D_j|}(\mathbf{y}_m - \frac{\sum_{m=1}^{|D_j|}\mathbf{y}_m}{|D_j|})^2}}{\sqrt{|D_j|}} \in \mathbb{R}^d. \tag{4.2}$$

This statistical information, along with the local model f_j, is sent to N_i to build enhanced models. Node N_i samples $\alpha|D_j|$ vectors from Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\sigma}_j^2)$ to obtain fabricated training inputs $\hat{\mathcal{X}}_j$. Corresponding outputs are generated as $\hat{\mathcal{Y}}_j = f_j(\hat{\mathcal{X}}_j) + \boldsymbol{\epsilon_j}$, where $\boldsymbol{\epsilon_j}$ is random noise from $\mathcal{N}(0, \bar{\boldsymbol{\sigma}}_j^2)$. Thus:

$$\Gamma(D_j) = \{(\hat{\mathcal{X}}_j, \hat{\mathcal{Y}}_j) : \hat{\mathcal{X}}_j \sim \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\sigma}_j^2), \hat{\mathcal{Y}}_j = f_j(\hat{\mathcal{X}}_j) + \epsilon_j\} \tag{4.3}$$

With MD, we expect significantly reduced inter-node transmission with larger values of $\alpha|D_j|$. For smaller values, considering model size, MD may not reduce transmission volume.

4.5.2 Enhanced Centroid Guided (ECG) Strategy

ECG extends the CG strategy by introducing an *intensity* parameter λ to control both cluster count and a subsequent duplication process. We define the number of clusters as $K = \frac{\alpha|D_j|}{\lambda}$. Node N_j quantizes D_j into K clusters and sends the centroids $\{\mathbf{w}_{jk}\}$ to N_i. For each centroid $\mathbf{w}_{jk}$, N_i samples $\lambda - 1$ points $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ from $\mathcal{N}(\mathbf{w}_{jk}, \boldsymbol{\sigma}_j^2)$. The standard deviation $\boldsymbol{\sigma}_j$ should be small enough to ensure sampled points remain close to the original distribution of D_j. The final sample combines all centroids and sampled points:

$$\Gamma(D_j) = \cup_{k=1}^{K}\{\mathbf{w}_{jk} \cup \{(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \sim \mathcal{N}(\mathbf{w}_{jk}, \boldsymbol{\sigma}_j^2)\}\}. \tag{4.4}$$

4.6 Experimental Evaluation

We now evaluate how the proposed maintenance strategies affect model performance on new concepts and unchanged nodes' data, and how reducing inter-node data transmission influences this process.

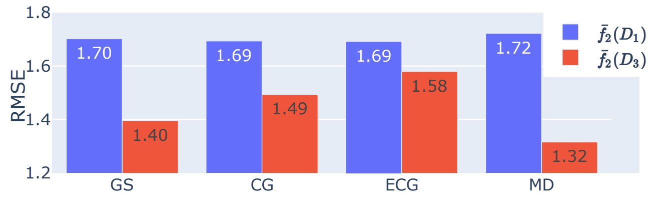


Figure 4.7: Performance of $\bar{f}_2^{GS}$, $\bar{f}_2^{CG}$, $\bar{f}_2^{ECG}$, $\bar{f}_2^{MD}$ on D_1 and D_3

4.6.1 Synthetic Dataset Evaluation

As shown in Figure 4.7, enhanced models built with different strategies performed similarly on D_1 but yielded noticeably different results on D_3. To control variables in our maintenance experiments, we (1) built all enhanced models using the GS strategy and (2) employed SVR for all models.

For each concept drift type at N_1, we extracted input-output pairs from drifted data using all four strategies (GS, CG, ECG, MD) and retrained $\bar{f}_2^{GS}$ to create four corresponding maintained models.

Figures 4.8-4.13 show enhanced model performance *before* and *after* maintenance with different types of drifted data. Though all maintained models started from an enhanced model built with GS, the "before maintenance" results include models built with different strategies to provide comprehensive comparison.

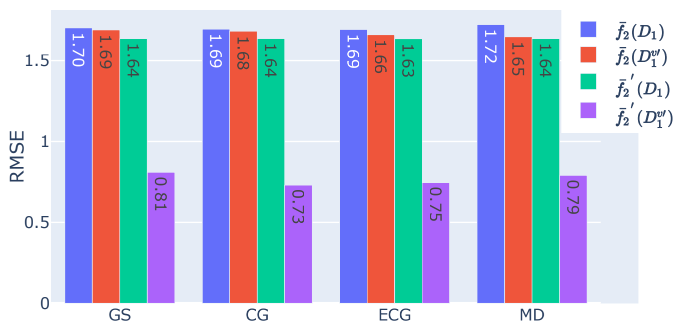


Figure 4.8: Performance of $\bar{f}_2$ and $\bar{f}_2'$ (retrained with $D_1^{v\prime}$) on D_1 and $D_1^{v\prime}$

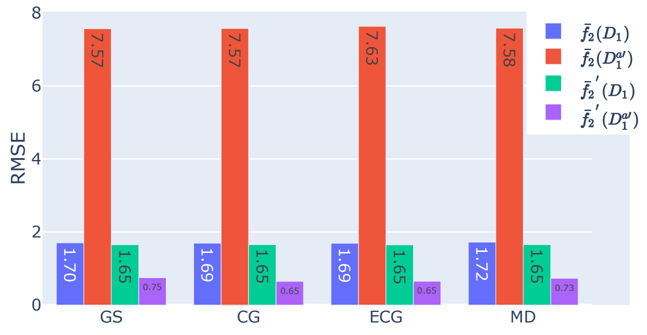


Figure 4.9: Performance of $\bar{f}_2$ and $\bar{f}_2'$ (retrained with $D_1^{a\prime}$) on D_1 and $D_1^{a\prime}$

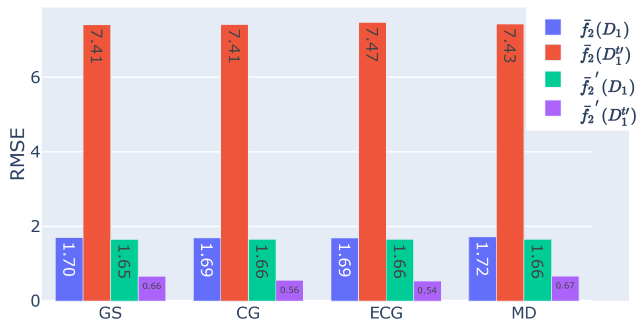


Figure 4.10: Performance of $\bar{f}_2$ and $\bar{f}_2'$ (retrained with $D_1^{t\prime}$) on D_1 and $D_1^{t\prime}$

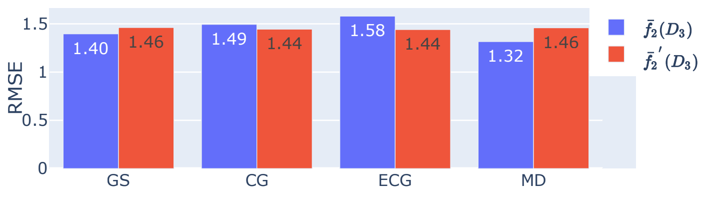


Figure 4.11: Performance of $\bar{f}_2$ and $\bar{f}_2'$ (retrained with $D_1^{v\prime}$) on D_3

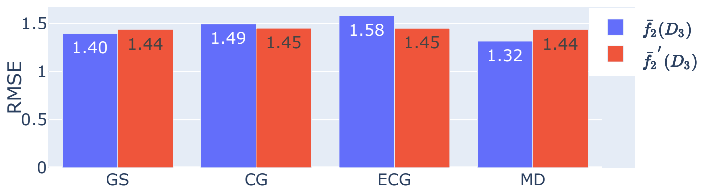


Figure 4.12: Performance of $\bar{f}_2$ and $\bar{f}_2'$ (retrained with $D_1^{a\prime}$) on D_3

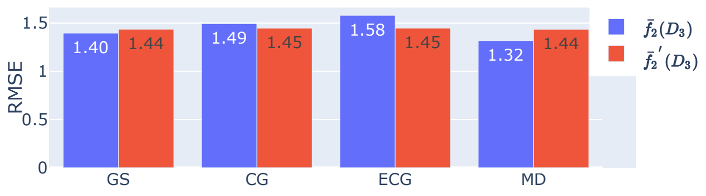


Figure 4.13: Performance of $\bar{f}_2$ and $\bar{f}_2'$ (retrained with $D_1^{t\prime}$) on D_3

In Figure 4.8, comparing blue bars (before maintenance) with red bars (performance on D_1 after maintenance), we observe that $\bar{f}_2$ performance barely degraded after encountering virtual drifted data ($D_1^{v\prime}$), which typically indicates no need for maintenance. However, the purple bars show that maintenance with $D_1^{v\prime}$ reduced error on $D_1^{v\prime}$ ($\mathbb{E}_{D_1^{v\prime}}[\mathcal{L}(\bar{f}_2^{\prime})]$) by nearly half while maintaining performance on D_1 ($\mathbb{E}_{D_1}[\mathcal{L}(\bar{f}_2^{\prime})]$, green bars) and D_3 ($\mathbb{E}_{D_3}[\mathcal{L}(\bar{f}_2^{\prime})]$, Figure 4.11). This represents a case where maintenance should occur but wouldn't be triggered by performance degradation in enhanced models. Here, the local model f_1 provides a more accurate indication of maintenance need.

Enhanced model performance before/after maintenance with actual and total drifted data shows similar patterns. As shown in Figures 4.9 and 4.10, before maintenance, all $\bar{f}_2$ models struggled with $D_1^{a\prime}$ and $D_1^{t\prime}$, producing errors several times higher than $\mathbb{E}_{D_1}[\mathcal{L}(\bar{f}_2)]$. After maintenance, errors on $D_1^{a\prime}$ and $D_1^{t\prime}$ were reduced to approximately one-third of $\mathbb{E}_{D_1}[\mathcal{L}(\bar{f}_2)]$, demonstrating the effectiveness of maintenance for both drift types. Furthermore, Figures 4.12 and 4.13 show that maintenance preserves model performance on D_3, indicating that $\bar{f}_2^{\prime}$ remains viable as a surrogate model for both N_1 and N_3 after maintenance.

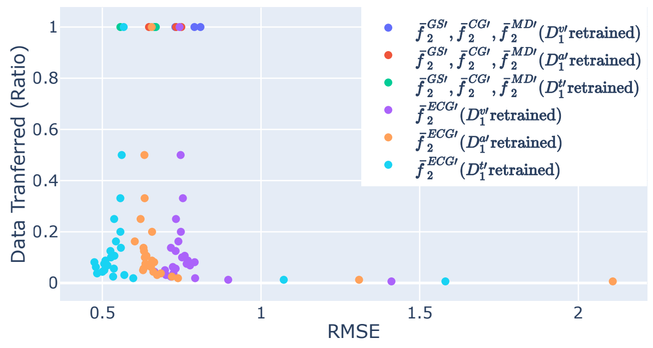


Figure 4.14: Performance of $\bar{f}_2^{\prime}$ on $D_1^{v\prime}$, $D_1^{a\prime}$ and $D_1^{t\prime}$ given different ratios of data transferred

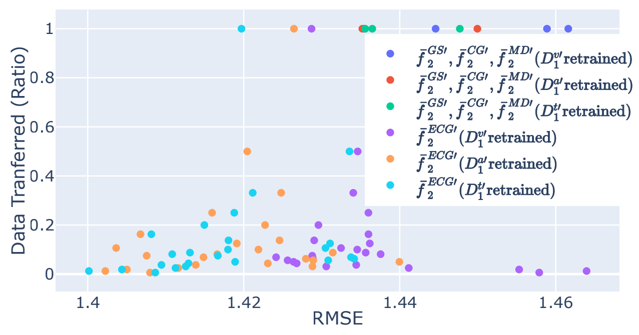


Figure 4.15: Performance of $\bar{f}_2^{\prime}$ on D_3 given different ratios of data transferred

To investigate how the ECG strategy reduces inter-node data exchanges, we experimented with different intensity values λ and compared with the other three strategies.

For MD, since model size is comparable to the data points being transferred, we consider it to transfer approximately the same amount of data as GS and CG. Results are shown in Figures 4.14 and 4.15, where the bottom left area indicates lower error with smaller data transfers.

As shown in Figure 4.14, for all drift types ($D_1^{v\prime}$, $D_1^{a\prime}$, and $D_1^{t\prime}$), the lowest errors were achieved by ECG while transferring only 5% to 20% of the data, demonstrating ECG's effectiveness in model maintenance with minimal data transfer. ECG also helped maintain performance on D_3, as shown in Figure 4.15, where all best results were achieved with ECG. Overall, maintained model performance on D_3 showed little sensitivity to the strategy used (maximum and minimum RMSE differed by only about 0.02).

4.6.2 Real Dataset Evaluation

We also evaluated our maintenance framework on a real multi-node dataset from (Harth & Anagnostopoulos, 2018a), which contains mobile sensor readings from four Unmanned Surface Vehicles (USVs) monitoring sea surfaces near Athens, Greece. Each USV represents an edge node. For our experiments, we used data from two USVs, denoted as D_1 and D_2. Using temperature as input ($x \in \mathbb{R}$), we predicted humidity as output ($y \in \mathbb{R}$).

Similar to our synthetic data experiments, we simulated concept drift only in D_1, covering the three drift types. We constructed enhanced models using all four maintenance strategies and evaluated their performance on drifted data. We then tested the ECG strategy with varying numbers of cluster centroids while maintaining a constant number of training samples to evaluate the trade-off between transferred data volume and performance.

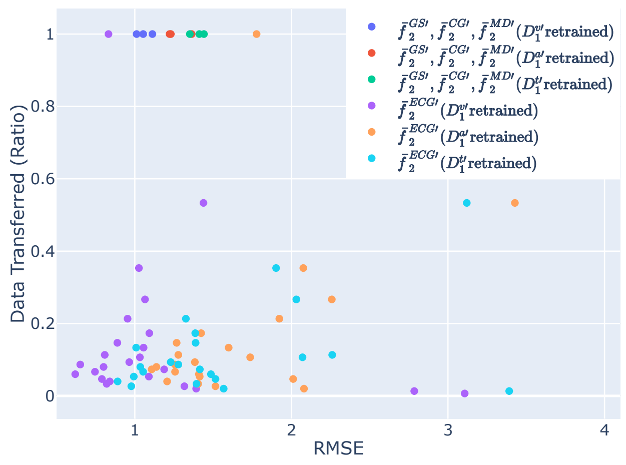


Figure 4.16: Performance of $\bar{f}_2^{\prime}$ on $D_1^{v\prime}$, $D_1^{a\prime}$ and $D_1^{t\prime}$ given different ratios of data transferred for the GNFUV dataset.

Figure 4.16 illustrates $\bar{f}_2$ enhanced model performance after maintenance across different drift types using various strategies. The results show that drift severity correlates with RMSE increase regardless of data transfer ratio. Importantly, the data compression ratio in ECG doesn't correlate proportionally with model performance. The enhanced model's performance deteriorates with both very small and very large numbers of cluster centroids. In the latter case, too many centroids with few surrounding points cause overfitting, while too few centroids leads to underfitting. The optimal trade-off occurs at around 0.1 data transmission ratio, which achieves the lowest RMSE for each drift type. This allows our framework to achieve similar performance to GS, CG, and MD strategies while transferring 10 times less data.

4.7 Conclusions and Future Work

In this chapter, we investigated the model maintenance problem for enhanced models confronting concept drift in distributed edge learning environments. Building on the resilience framework introduced in Chapter 3, we addressed the unique challenge posed by enhanced models that operate across multiple nodes' data sources, where concept drift may affect these sources unevenly.

Our experimental investigations revealed several key insights:

1. Enhanced models, due to their inherent generalizability, are less affected by simple concept drifts (virtual drift) and experience lower performance degradation from more complex drift types (actual and total drift) compared to local models.
2. These patterns hold consistently across different model types and construction strategies, enabling unified handling of concept drift through model retraining with drifted data.
3. After maintenance, enhanced models successfully adapt to new concepts while preserving performance on nodes where no drift occurred, maintaining their resilience capabilities.
4. The proposed Enhanced Centroid Guided (ECG) strategy achieves effective model maintenance while significantly reducing inter-node data transfer requirements, with optimal performance achieved when transferring only 10% of the data volume required by conventional approaches.

The findings demonstrate that our maintenance framework can effectively sustain the resilience properties of enhanced models even as data distributions evolve over time. This ensures continuous high-quality service in distributed edge learning environments despite both node failures and concept drifts.

Future work could focus on developing more sophisticated triggering mechanisms for model maintenance that account for the unique properties of enhanced models, combining both performance-based and distribution-based drift detection. Additionally, investigating incremental learning approaches for enhanced model maintenance could further reduce computational overhead and improve adaptation speed in highly dynamic environments.

Chapter 5

Query-Driven Predictive Analytics

5.1 Introduction

The growing scale and complexity of data create substantial challenges for predictive modeling and analytics. Machine Learning (ML) systems must efficiently allocate their resources to handle analytics task requests, which we refer to as *queries* throughout this chapter, that require access to *specific* data subsets for effective model building and maintenance (Kraska et al., 2013; Savva et al., 2020a; Verbraeken et al., 2020a).

While distributed learning paradigms (Verbraeken et al., 2020b) including federated learning (McMahan et al., 2017; Yang et al., 2023) can provide satisfactory performance in homogeneous environments, they face limitations in dynamic settings where data distributions and query access patterns vary significantly. In such environments, global models trained over all available data are unaware of the diversity in query access patterns, serving as a 'blanket' over different data and query distributions. As demonstrated in (Toneva et al., 2019), not all the data owned by each node is relevant to queries. These distributed systems must contend with (i) different data distributions across and within nodes and (ii) different query access patterns issued by various queries (Anagnostopoulos, 2020).

This leads to models that are neither tailored to the patterns and trends of queries accessing *different* data spaces (regions), nor capable of identifying and exploiting *only the relevant* data requested by past, current, and future queries. Data samples favorable to model performance are considered relevant, while those detrimental to performance are irrelevant. A mechanism that learns from query patterns to identify and select *relevant data with respect to query patterns* for training tailored models becomes essential. When data are not representative of the query distribution, solutions that rely on *all* data lead to suboptimal performance that adversely affects model accuracy.

We address this challenge by introducing a progressive learning and refinement mechanism in a 'data-centric' fashion. Our approach leverages queries and current model predictions to select the most appropriate training data for model refinements. This adaptation enables a model to efficiently and accurately serve future queries, while ensuring irrelevant data do not participate in the refining process.

5.1.1 Rationale & Motivation

Our mechanism is based on two fundamental pillars: Data-Centric Learning (DCL) and Query-Driven Learning (QDL), which broadly involve *learning from queries* to discover relevant data, and in turn, *using these relevant data to train tailored* models for future analytics tasks.

Data-Centric Learning represents a paradigm shift that focuses on training models with *quality data* rather than constantly modifying model structures as in traditional model-centric approaches (Jarrahi et al., 2022). The quality of data significantly impacts model performance (Subramonyam et al., 2021) since data are more versatile (Miranda, 2021), while models suffer from concept drifts and limited applicability (Sambasivan et al., 2021). DCL emphasizes collecting and identifying data relevant to training models (Zha et al., 2023), which helps alleviate model overfitting (Cao et al., 2022; Cocarascu et al., 2020) and improves model maintenance (Anik & Bunt, 2021). In our context, irrelevant data refers to data regions rarely accessed by queries, indicating lower interest to end-users or applications. Building models on identified relevant data results in more accurate and tailored models.

Query-Driven Learning encompasses discovering query access patterns over data to leverage query-driven analytics (Huang et al., 2018; Savva et al., 2020a, 2020b). While rooted in Interactive Learning (IL) (Fails & Olsen, 2003; Teso & Kersting, 2019), QDL specifically focuses on extracting knowledge from queries. We adopt QDL to exploit end-users' queries for discovering data regions of interest, which can then be used to train, refine, and progressively improve model performance. This approach enables efficient data exploration (Anagnostopoulos & Triantafillou, 2017) and explanatory services (Anagnostopoulos & Triantafillou, 2015; Anagnostopoulos et al., 2018).

Motivation: Figure 5.1 illustrates the importance of detecting relevant data from queries issued to a node. Our experiment shows that a query patterns-unaware regression model trained on the entire dataset yields a Root Mean Square Error (RMSE) of 6.39 over test queries. In contrast, by progressively identifying relevant data regions and selectively using them for training, we obtain an RMSE of 1.55 over the same test queries. This demonstrates that including irrelevant data in modeling has detrimental effects on performance.

The challenge lies in developing mechanisms for a node to (1) learn a relevance prediction function that progressively assigns relevance probabilities to data samples, and (2) adapt to dynamic changes over time to handle both data and query pattern drifts.

The novelty of our approach stems from integrating DCL and QDL principles to discover relevant data from queries, select them for refining models, and adapt to dynamics.

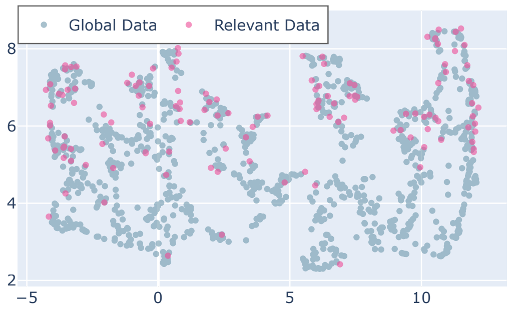


Figure 5.1: Whole, relevant and irrelevant data regions on a node. The relevant data have been progressively discovered by leveraging incoming queries.

5.2 Related Work

The concept of 'data relevance' spans multiple research domains. While subjective and application-specific, we identify several key approaches related to our work.

5.2.1 Data Relevance in Machine Learning

Mechanisms for selecting relevant data have been explored in traditional machine learning settings (Ghorbani & Zou, 2019; Nagalapatti et al., 2022; Toneva et al., 2019), operating under the assumption that all data are available in one place without external mechanisms to guide identification. For instance, (Nagalapatti et al., 2022) introduced a relevant data selector for denoising data in federated learning, while (Kurmanji & Triantafillou, 2023) addressed the maintenance of global models by detecting 'out-of-distribution' data.

Our approach differs fundamentally as our concept of relevance is defined *and* guided by learning from the query patterns. Rather than focusing on intrinsic data properties, we use query patterns to improve predictive models' performance in environments with variable data access patterns.

5.2.2 Query-Driven Approaches

In the QDL paradigm, several approaches discover sub-regions of interest within larger datasets. These methods benefit large-scale data visualization and exploratory analytics (Bethel et al., 2006; Meyer et al., 2008; Rübel et al., 2012), natural language processing (Fang et al., 2021), and database optimization (Chen et al., 2002).

For example, (Symeonides et al., 2019) introduced a query model to compose analytics queries, while (Ma & Triantafillou, 2022) demonstrated that different regression models excel in different data regions, supporting our motivation to identify relevant regions for predictive modeling tailored to query patterns.

These approaches utilize current query trends, but do not exploit the queries themselves to identify relevant data. In addition, they lack adaptation mechanisms for changes in the underlying data and query distributions, which is a crucial feature in dynamic environments that our approach provides.

5.2.3 Explanatory Analytics

Some approaches leverage historical queries and their results to predict future patterns. (Savva et al., 2020a) approximate the distribution of 'interesting data regions' using surrogate models trained on historical queries, while (Savva et al., 2020b) establish connections between query parameters and data regions through explanation functions. (Anagnostopoulos et al., 2018) demonstrated query-driven analytics in scalable predictive modeling.

The frameworks in (Savva et al., 2020b) and (Anagnostopoulos & Triantafillou, 2017) estimate the cardinality of data regions defined by queries based on patterns from previous queries. However, they differ from our approach as they don't identify data regions based on incoming queries or refine these data for selective model training.

5.3 Problem Fundamentals

5.3.1 Preliminaries

Before formalizing our problem, we introduce several key concepts that underpin our approach:

Interactive Learning (IL)

This paradigm enables systems to learn from interactions with users or environments (Fails & Olsen, 2003; Teso & Kersting, 2019). Unlike traditional learning approaches that rely solely on static training data, IL leverages ongoing feedback to continuously improve model performance. Our Query-Driven Learning (QDL) approach builds upon this foundation by specifically focusing on learning from analytics queries.

Data-Centric Learning (DCL)

While model-centric approaches focus on architecture improvements, DCL prioritizes data quality and relevance (Jarrahi et al., 2022). DCL recognizes that high-quality, task-relevant data often contribute more to model performance than architectural refinements, especially in specialized domains where data distributions vary significantly.

Optimal Stopping Theory (OST)

This mathematical framework addresses the problem of choosing the optimal time to perform an action (in our case, model updates) to maximize the expected reward (Albert & Goran, 2006). OST provides principled approaches for sequential decision making under uncertainty, which we apply to determine when to trigger model refinements in dynamic environments.

Relevance

In our context, relevance refers to the degree to which data contribute positively to the performance of the model for specific query patterns. Relevant data are samples that, when used for training, lead to improved prediction accuracy for queries following observed access patterns.

5.3.2 Problem Setting

Consider a data management system with n nodes indexed in $\mathcal{N} = \{1, \ldots, n\}$. Each node $i \in \mathcal{N}$ maintains a local dataset and serves analytics queries that require access to its data.

Definition 5.1 (Node i's local dataset)**.** *Consider a node $i \in \mathcal{N}$ with local data $\mathcal{D}_i = \mathcal{X}_i \times \mathcal{Y}_i$. The domain $\mathcal{X}_i \subseteq \mathcal{X} \subset \mathbb{R}^d$ refers to d-dimensional input vectors $\mathbf{x} = [x_1, \ldots, x_d]^\top \in \mathcal{X}_i$ drawn from an input probability $\mathcal{P}_\mathcal{X}(\mathbf{x})$. The domain $\mathcal{Y}_i \subset \mathcal{Y} \subset \mathbb{R}$ refers to output values drawn from an output probability $\mathcal{P}_\mathcal{Y}(y)$, where $y \in \mathcal{Y}_i$.*

Each node trains its local model $f_i(\mathbf{x}; \mathbf{w}_i)$ over local data $\mathcal{D}_i$, where $\mathbf{w}_i \in \mathcal{W}$ are the model parameters from parameter space $\mathcal{W}$. Given an input $\mathbf{x} \in \mathcal{X}$, the local model predicts the output $\hat{y} = f_i(\mathbf{x})$ with prediction error (loss) $\ell(\mathbf{w}_i, (\mathbf{x}, y))$, where y is the actual output. For example, $\ell(\mathbf{w}_i, (\mathbf{x}, y)) = |y - \hat{y}|$ or $(y - \hat{y})^2$ for regression models and $\ell(\mathbf{w}_i, (\mathbf{x}, y)) = \max\{0, (1 - y\hat{y})\}$ for binary classification models, with $y \in \{-1, +1\}$.

Definition 5.2 (Input & Output regions)**.** *The input region $\mathcal{X}_+$ is a subset of non-zero probability inputs $\mathbf{x} \in \mathcal{X}$ such that: $\mathcal{X}_+ = \{\mathbf{x} \in \mathcal{X} : \mathcal{P}_\mathcal{X}(\mathbf{x}) > 0\}$. Moreover, consider an input-output pair $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ drawn by the joint probability $\mathcal{P}_{\mathcal{X}\mathcal{Y}}(\mathbf{x}, y)$. The output region $\mathcal{Y}_+$ is defined as: $\mathcal{Y}_+ = \{y \in \mathcal{Y} : \mathbf{x} \in \mathcal{X}_+\}$.*

An input region refers to statistically observed data in the input domain. Each node equipped with model f_i handles the analytics tasks from users and applications. These tasks consist of predictive queries as defined below:

Definition 5.3 (Query)**.** *A query to model f_i of node $i \in \mathcal{N}$ is represented by an input $\mathbf{q} \in \mathcal{Q} \subseteq \mathcal{X}$ drawn by a query probability distribution $\mathcal{P}_\mathcal{Q}$. The query output is the predicted output $\hat{y}(\mathbf{q}) = f_i(\mathbf{q})$.*

For simplicity, we set $\hat{y} = \hat{y}(\mathbf{q}) = f_i(\mathbf{q})$ to indicate the output of f_i given query $\mathbf{q} \in \mathcal{Q}$.

In typical distributed learning systems, a central node (e.g., data server/Cloud) builds a global model f_G either by aggregating all nodes' local data $\mathcal{D}_G = \bigcup_{i \in \mathcal{N}} \mathcal{D}_i$ or through collaborative training methods like Federated Learning. All nodes ultimately obtain the same model copy: $f_1 = \ldots = f_n = f_G$.

While effective for general distributed learning, these approaches do not account for query access patterns specific to individual nodes. Such patterns vary both across nodes (not all nodes experience the same query distribution) and within nodes (different local data sub-regions experience different access frequencies). Queries $\mathbf{q} \sim \mathcal{P}_\mathcal{Q}$ received by a node require access to different data regions $\mathcal{X}_+ \sim \mathcal{P}_\mathcal{X}$, creating variability in the interestingness of the data.

Specifically, some data points $\mathbf{x}' \in \mathcal{X}_+$ become more popular than others $\mathbf{x} \in \mathcal{X}_+$ because queries prefer to access them. With a query probability distribution $\mathcal{P}_\mathcal{Q}$, we have $\mathcal{P}_\mathcal{Q}(\mathbf{x}') > \mathcal{P}_\mathcal{Q}(\mathbf{x})$ for data in region $\mathcal{X}_+$. This popularity pattern varies across data regions within each node and across different nodes.

This variation in query access patterns leads to the classification of data regions into two categories: relevant and irrelevant regions, defined as follows:

Definition 5.4 (Relevant Data regions)**.** *Given a query probability distribution $\mathcal{P}_\mathcal{Q}$ and Definition 5.2, a relevant data region is defined as $\mathcal{X}' \times \mathcal{Y}' = \{(\mathbf{x}, \mathbf{y}) \in \mathcal{X}_+ \times \mathcal{Y}_+ : \mathcal{P}_\mathcal{Q}(\mathbf{x}) > 0\}$.*

A node can identify its relevant data regions by incrementally learning query access patterns, allowing it to train/refine its model over only relevant regions. This results in a more accurate and query-adaptable local model. We formulate this as our first problem:

Problem 1 (Query-Driven Relevant Region Discovery)**.** *Given a series of queries and their outputs $\mathcal{Q}_t = \{(\mathbf{q}_1, \hat{y}_1), \ldots, (\mathbf{q}_t, \hat{y}_t)\}$ up to time $t \in \mathbb{T} = \{1, 2, \ldots, \}$, find all relevant regions of the data $\bigcup_{\tau=1}^{t} \left(\mathcal{X}'_\tau \times \mathcal{Y}'_\tau\right)$, with $\hat{y}_\tau = f_i(\mathbf{q}_\tau)$.*

In Problem 1, a node must identify the relevant data region(s) where it (i) observes queries $\mathbf{q} \sim \mathcal{P}_\mathcal{Q}$ from relevant input regions $\mathcal{X}'$ and (ii) refines its local model with relevant input-output training data from $\mathcal{X}' \times \mathcal{Y}'$.

For clarity, we now remove the node index i from the notation. We denote the identified relevant and irrelevant data from a node's local data $\mathcal{D}$ as $\mathcal{D}_R \subset \mathcal{D}$ and $\mathcal{D}_I \equiv \mathcal{D} \setminus \mathcal{D}_R$, respectively. This gives us the input-output domains $\mathcal{D}_R \subseteq \mathcal{X}' \times \mathcal{Y}'$ and $\mathcal{D}_I \subseteq (\mathcal{X} \times \mathcal{Y}) \setminus (\mathcal{X}' \times \mathcal{Y}')$.

Only $\mathcal{D}_R$ contributes positively to the node's model f tailored to queries $\mathcal{Q}_t$, while including irrelevant data $\mathcal{D}_I$ could negatively impact performance through unnecessary generalization. This leads to our second problem:

Problem 2 (Relevant Training Data Identification)**.** *Given a node's local data set $\mathcal{D}$ and issued queries $\mathcal{Q}_t$ up to time t, find the relevant training data $(\mathbf{x}, y) \in \mathcal{D}_R \subset \mathcal{D}$ up to time t to be used to incrementally train the node's model f. The predictive performance of the model f trained on $\mathcal{D}_R$ should remain at the same accuracy level for the subsequent received queries $\mathcal{Q}_\tau$, $\tau \geq t$.*

Finally, dynamic environments experience concept drifts in data distributions and changes in query patterns over time. When the data distribution $\mathcal{P}_{\mathcal{X}}$ changes in the relevant regions, the model can become inaccurate for queries targeting these regions. Similarly, changes in query distribution $\mathcal{P}_{\mathcal{Q}}$ may cause previously irrelevant regions to become relevant or vice versa. Our mechanism must therefore learn and adapt to these changes, leading to our third problem:

Problem 3 (Relevant Data & Query Updates)**.** *Consider a node's model tailored to queries over identified relevant regions. Upon changes in either data distribution $\mathcal{P}_{\mathcal{X}}$ or query distribution $\mathcal{P}_{\mathcal{Q}}$, the node should be equipped with mechanisms that can both identify newly formed relevant regions and adjust the node's model f to be tailored to the new query distribution.*

5.3.3 Key Challenges

The three problems outlined above present several significant challenges:

Relevance Identification Determining which data samples are relevant to specific query patterns without prior knowledge of these patterns requires sophisticated mechanisms that can learn from query behavior over time.

Exploration-Exploitation Balance The system must balance between exploring potentially relevant data regions and exploiting known relevant regions to maximize predictive performance.

Online Learning with Limited Feedback Unlike supervised learning with explicit labels, our approach must operate with limited feedback: only the queries themselves provide guidance on relevant regions.

Concept Drift Detection Detecting when data or query distributions have changed significantly enough to warrant model updates is challenging due to the noisy nature of real-world query patterns.

Dynamic Adaptation The system must adapt to the changes in the distribution without requiring complete retraining, which would be computationally inefficient in dynamic environments.

Our approach addresses these challenges through a novel data-centric learning framework that progressively refines models based on query patterns, as described in the following sections.

5.4 Data Relevance Framework

5.4.1 Query-Driven Identification of Relevant Data

Initially, a node has a local dataset $\mathcal{D}$. To identify the relevant data samples from $\mathcal{D}$, the node should be able to predict query outputs, which will be taken into account to extract relevant data. However, the node cannot predict outputs without first building and deploying an initial model. To *cold start* the process of using queries to find relevant data, the node begins training a local model on the entire dataset $\mathcal{D}$, which we denote as f_0. This initial model f_0, built at time $t = 0$, is unaware of any query distribution. The model f_0 will be progressively refined with incoming queries $\mathbf{q}_1, \dots, \mathbf{q}_t$, $t > 0$, each accessing different regions over the data space determined by $\mathcal{D}$.

At the first round $t = 1$, the node receives a sequence of N_t queries $\mathcal{Q}_t = \{\mathbf{q}_{t,1}, \dots, \mathbf{q}_{t,N_t}\}$ and proceeds with predictions $(\hat{y}_{t,1}, \dots, \hat{y}_{t,N_t})$ using its model from round $t-1$. For the k-th query in round t, $\mathbf{q}_{t,k} \in \mathcal{Q}_t$, we obtain:

$$\hat{y}_{t,k} = \begin{cases} f_0(\mathbf{q}_{t,k}), & \text{if } t = 1, \\ f_{t-1}(\mathbf{q}_{t,k}), & \text{if } t > 1. \end{cases} \tag{5.1}$$

At each step $k = 1, \dots, N_t$ of round t, when the node receives query $\mathbf{q}_{t,k}$, we instantiate an input-output data region based on the prediction $\hat{y}_{t,k}$. This region is determined by a *relevance parameter* $\lambda > 0$, which controls the impact of each query on the model refinement process. At each step k, we select data samples from $\mathcal{D}$ belonging to the k-th query to form a relevant region as:

$$\mathcal{D}_{t,k} = \{(\mathbf{x}, y) \in \mathcal{D} : |y - \hat{y}_{t,k}| \leq \lambda\}. \tag{5.2}$$

The subset $\mathcal{D}_{t,k} \subset \mathcal{X}'_{t,k} \times \mathcal{Y}'_{t,k}$ reflects the instantaneous relevance of the region *around* query $\mathbf{q}_{t,k}$. At the end of round t, after N_t queries have been received and their corresponding relevance subsets $\{\mathcal{D}_{t,k}\}_{k=1}^{N_t}$ have been determined, we obtain the training samples for round t as:

$$\mathcal{D}_t = \bigcup_{k=1}^{N_t} \mathcal{D}_{t,k}. \tag{5.3}$$

The dataset in (5.3) is used to train a model f_t on all relevant regions identified during round t. The training can be performed either in batch mode at the end of round t or incrementally after each k-th query over the subset $\mathcal{D}_{t,k}$ by adopting (regularized) Stochastic Gradient Descent (SGD). For incremental training, the model weight $\mathbf{w}_{t,k}$ is updated based on the training samples in $\mathcal{D}_{t,k}$:

$$\mathbf{w}_{t,k}^{(\kappa+1)} = (1 - \zeta\eta_\kappa)\mathbf{w}_{t,k}^{(\kappa)} + \eta_\kappa \nabla \ell(\mathbf{w}_{t,k}^{(\kappa)}), \tag{5.4}$$

where $\eta_\kappa \in (0, 1)$ is a learning rate with schedule $O(\frac{1}{\kappa})$, ζ is the regularization parameter, and $\ell(\mathbf{w}_{t,k}^{\kappa})$ is the prediction loss at iteration κ, with $\kappa = 1, \dots, |\mathcal{D}_{t,k}|$. Note that $|\mathcal{D}_{t,k}|$ represents the cardinality of samples in the relevant region $\mathcal{D}_{t,k}$.

Upon receiving the $(k + 1)$-th query, the model continues to incrementally train its parameters based on the relevant region $\mathcal{D}_{t,k+1}$ in a similar fashion as in (5.4), for $k = 1, \dots, N_t$. Finally, the model $f_t(\mathbf{x}; \mathbf{w}_t)$ with trained parameter $\mathbf{w}_t$ at the end of round t can be used for predictions in the next round $t + 1$.

5.4.2 Data-Centric Learning Mechanism

During the learning process, at each round t, we track the model f_t's predictive performance by estimating the generalization error $\epsilon_t = \mathbb{E}_{(\mathcal{X},\mathcal{Y})}\ell(\mathbf{w}_t; (\mathbf{x}, y)) \approx \frac{1}{m}\sum_{j=1}^{m}(y_j - f_t(\mathbf{x}_j))^2$. In addition, we maintain only the best-performing model up to round t, denoted as f_t^*, along with its relevant training data $\mathcal{D}_t^*$, i.e.,

$$f_t^* = \arg\min_{\tau=1,\dots,t} \epsilon_\tau. \tag{5.5}$$

If in round t the performance degradation of model f_t compared to model f_{t-1} of round $t - 1$ exceeds a threshold $\gamma \in [1, \infty)$, then, in round $t + 1$, model f_t is substituted with the best model f_t^* so far. This means that the predictions for queries in round $t + 1$ are obtained by model f_t^* instead of model f_t. Given the k-th query $\mathbf{q}_{t+1,k}$ in round $t + 1$, the prediction is the following:

$$\hat{y}_{t+1,k} = \begin{cases} f_t^*(\mathbf{q}_{t+1,k}), & \text{if } \frac{\epsilon_t}{\epsilon_{t-1}} > \gamma, \\ f_t(\mathbf{q}_{t+1,k}), & \text{otherwise.} \end{cases} \tag{5.6}$$

The error-ratio rate γ serves a purpose similar to the fundamental learning rate while balancing the exploration of relevant data regions and exploitation of the current models for mining relevant data that improve the prediction performance of the trained models in the process.

Let us elaborate on the decision event $\{\epsilon_t < \gamma\epsilon_{t-1}\}$ used to select the prediction model in round $t + 1$, thus determining how we further exploit the relevant data regions. In the case where $\epsilon_t > \epsilon_{t-1}$ but the ratio is less than γ, that is, $1 < \frac{\epsilon_t}{\epsilon_{t-1}} \leq \gamma$, we still use the f_t model for the predictions in round $t + 1$. Here, we allow the process to develop a more generalizable model over new relevant regions by further mining areas based on the f_t model rather than using the best model so far. This reflects the **exploration** component of the process, where we tolerate a certain level of degradation (associated with parameter γ) in the prediction performance of model f_t to explore more relevant regions, which will be used as training datasets for models in future rounds.

On the other hand, if the ratio exceeds γ, then the relevant areas should be further exploited using the predictions of the best model so far f_t^*. This **exploitation** requires a relatively high prediction accuracy to extract and generate data sets for future model training that accurately reflect the areas of interest defined by the incoming queries.

In the case where f_t at the end of round t is actually the best model so far, that is, $f_t = f_t^* = \arg\min_{\tau=1,\dots,t}\{\epsilon_\tau\}$, we obtain $\epsilon_t < \epsilon_{t-1}$. Therefore, the ratio is less than γ, and the predictions in round $t+1$ are obtained by the best model so far, which is also model f_t. This reflects a full exploitation mode and an improvement in the prediction capacity of model f_t, which replaces the best model.

Furthermore, if for a relatively high number of rounds we obtain ratios less than γ but greater than unity, then we are in a continuous (monotonically) exploration mode, reducing the probability of exploitation and thus the opportunity to further improve the prediction capacity of the model in future rounds. To avoid this situation, we devise a schedule for the γ rate that decreases with round t, that is, $\gamma_t = O(\frac{\gamma_0}{t})$ with an initial value $\gamma_0 > 1$. Specifically, we use:

$$\gamma_t = \min\{1, \frac{\gamma_0}{t+1}\}, \tag{5.7}$$

to achieve less exploration and more exploitation as the number of rounds increases. This approach is conceptually aligned with the SGD learning rate schedule for incremental model training to ensure convergence (Bottou, 2012). The γ_t rate now controls the selection between using the current model f_t or the best model up to round t in round $t + 1$ for exploration or exploitation, respectively.

Lemma 5.1. *Consider the event $\{\epsilon_t < \epsilon_{t-1}\}$, which indicates an improvement in the prediction accuracy of the model f_t. The probability of improvement $\mathcal{P}(\epsilon_t < \epsilon_{t-1})$ increases with round $t > 1$, reflecting that future refined models achieve continuously lower loss compared to past* *models.*

Lemma 5.2. *Consider the event $\{\epsilon_t < \gamma_t \epsilon_{t-1}\}$ with rate γ_t defined in (5.7). Define the variable $z_t = \epsilon_t - \gamma_t \epsilon_{t-1}$ that reflects the difference of two consecutive errors due to exploration in the round t w.r.t. γ_t. Then, the variance $\sigma^2_{z_t}$ of the exploration difference decreases with rate $O\big(\frac{1}{(t+1)^2}\big)$.*

Lemma 5.2 shows that the refining process decreases the possibility of turning to an exploration mode with round t, and therefore, it switches to exploitation mode, thus obtaining a refined model tailored to the query distribution along with identifying the relevant data regions.

When model f_t is replaced with the best model so far f^*_t given the event $\{\epsilon_t > \gamma_t \epsilon_{t-1}\}$ from (5.6), then the relevant regions identified in the round $t+1$ should be incorporated into the process. Specifically, at the end of round $t+1$, a non-replacement random sub-sampling of $\mathcal{D}^*_t$ of size $\eta|\mathcal{D}^*_t|$, with $\eta \in (0,1]$, is concatenated to the relevant data region $\mathcal{D}_{t+1}$. That is:

$$\mathcal{D}_{t+1} = \mathcal{D}_{t+1} \cup \mathcal{D}^*_{t,\eta} \text{ with } \mathcal{D}^*_{t,\eta} = \{(\mathbf{x}, y) \in \mathcal{D}^*_t, |\{(\mathbf{x}, y)\}| = \eta|\mathcal{D}^*_t|\} \tag{5.8}$$

The rationale behind this expansion of the relevant region $\mathcal{D}_{t+1}$ is that, once the best model up to round t is used for predictions at round $t+1$, the relevant data identified by the best model so far are 'carried' to the next round's relevant region for future exploitation. This significantly reduces the fluctuation of model performance in rounds $\tau > t+1$ due to query randomness and helps avoid overfitting in future models trained in subsequent rounds. However, excessively high values of η would increase the chance that the model would get stuck in local optima, which could be detrimental to performance.

Remark 5.1. *The expansion of the relevance data region in (5.8) cannot be applied directly at the start of the process, as we need to allow time for $\mathcal{D}^*_t$ to be updated given that the better models in rounds $t > 1$ will outperform the best model up to $t = 1$. Initially, we have $\mathcal{D}^*_0 = \mathcal{D}$ with $f^*_0 = f_0$.*

The complete node process is provided in Algorithm 2.

5.4.3 Model Selection based on Data Relevance Discriminator

Our goal is to obtain a Query-Driven Learning (QDL) model that identifies relevant data regions and efficiently adapts to the trends of incoming queries. However, in cases where occasional 'outlier' queries follow different distributions and fall into irrelevant data regions, our mechanism should be able to maintain high predictive capacity.

Algorithm 2 Query-Driven Data-Centric Model Learning

```
1: Input: Dataset $\mathcal{D}$, rates $\gamma_0, \eta$, relevance parameter $\lambda$
2: Initialization: initial model $f_0$, error $\epsilon_0$, best model $f_0^* = f_0$, $\mathcal{D}_0^* = \mathcal{D}$
3: for round $t = 1, \ldots, T$ do
4:     $\mathcal{Q} \leftarrow \emptyset$ /*Queries per round*/
5:     for step $k = 1, \ldots, N_t$ do
6:         Receive query $\mathbf{q}_{t,k}$
7:         $\mathcal{Q} = \mathcal{Q} \cup \{\mathbf{q}_{t,k}\}$
8:         Predict $\hat{y}_{t,k}$ using (5.1)
9:         Acquire relevant region $\mathcal{D}_{t,k}$ using (5.2) and incrementally learn $\mathbf{w}_{t,k}$ using (5.4)
10:    end for
11:    Acquire relevant dataset $\mathcal{D}_t$ using (5.3), tailored model $f_t$ and error $\epsilon_t$
12:    Update $\gamma_t$ using (5.7)
13:    if $\epsilon_t > \gamma_t \epsilon_{t-1}$ then
14:        Replace $f_{t+1}$ with $f_t^*$ and obtain data subset $\mathcal{D}_{t,\eta}^*$ in (5.8)
15:    else
16:        Replace $f_{t+1}$ with $f_t$
17:    end if
18:    Update best model $f_t^*$ using (5.5)
19: end for
20: /*End of Round $T$*/
21: Obtain relevant $\mathcal{D}_R$ and irrelevant $\mathcal{D}_I$ regions
22: Train discriminator $f_C$ using $\mathcal{D}_R$ and $\mathcal{D}_I$
23: return tailored final best model $f_T^*$, discriminator $f_C$
```

Therefore, we introduce a *data relevance discriminator* based on a binary classifier f_C. This discriminator determines whether the incoming queries fall into relevant or irrelevant regions, classifying them as relevant or irrelevant queries, respectively. The discriminator f_C is trained based on the relevant data identified $\mathcal{D}_R$ and irrelevant samples $\mathcal{D}_I$ such that $\mathcal{D} = \mathcal{D}_I \cup \mathcal{D}_R$. The sets $\mathcal{D}_R$ and $\mathcal{D}_I = \mathcal{D} \setminus \mathcal{D}_R$ are acquired at the end of the process (round $t = T$) introduced in Section 5.4.2.

We expand each data sample $(\mathbf{x}, y)$ from $\mathcal{D}_R$ and $\mathcal{D}_I$ by adding the label $z = 1$ (relevant) and $z = -1$ (irrelevant), respectively, i.e., $\mathcal{D}'_R = \{\mathbf{z} = (\mathbf{x}, y, z) : (\mathbf{x}, y) \in \mathcal{D}_R \text{ and } z = 1\}$ and $\mathcal{D}'_I = \{\mathbf{z} = (\mathbf{x}, y, z) : (\mathbf{x}, y) \in \mathcal{D}_I \text{ and } z = -1\}$. We adopt a Support Vector Machine (SVM) classifier $f_C(\mathbf{x}) \in \{+1, -1\}$ trained on the dataset $\mathcal{D}'_R \cup \mathcal{D}'_I$, such that:

$$f_C(\mathbf{x}) = \text{sign}\Big(\sum_{\mathbf{v}_i \in \mathcal{D}'_R \cup \mathcal{D}'_I} \alpha_i z_i \mathcal{K}(\mathbf{v}_i, \mathbf{x})\Big). \tag{5.9}$$

We then obtain the support vectors $\mathbf{v}_i \in \mathcal{D}'_R \cup \mathcal{D}'_I$ with non-zero coefficients α_i that define the separating hyperplane between relevant and irrelevant regions. We adopt the radial basis kernel $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \exp(-\frac{1}{2}\|\mathbf{x} - \mathbf{x}'\|_2^2)$; other kernels could also be adopted, such as linear or polynomial kernels.

An incoming (unseen/new) query $\mathbf{q}_t$ after the end of the process, that is, $t > T$, is classified according to the model f_C as relevant if $f_C(\mathbf{q}_t) = 1$; otherwise, it is classified as irrelevant, i.e., $f_C(\mathbf{q}_t) = -1$. Hence, upon receiving the new query $\mathbf{q}_t$, the discriminator selects for future prediction either the model f_0 if $\mathbf{q}_t$ is classified as irrelevant or the best-tailored model f_T^* if $\mathbf{q}_t$ is classified as relevant for $t > T$, that is,

$$\hat{y} = \begin{cases} f_T^*(\mathbf{q}_t), & \text{if } f_C(\mathbf{q}_t) = 1, \\ f_0(\mathbf{q}_t), & \text{if } f_C(\mathbf{q}_t) = -1. \end{cases} \tag{5.10}$$

Given this discriminator-based selection of models, it is guaranteed that the prediction performance of the system upon newly incoming queries is equal to or better than the result acquired by using only model f_0, which does not take into account query access patterns to the identified relevant regions.

5.5 Theoretical Analysis

In this section, we provide a comprehensive theoretical analysis of our approach. We first establish error bounds for the model refinement process (Section 5.5.1), then analyze computational complexity (Section 5.5.2), and derive the optimal learning round horizon based on Optimal Stopping Theory (Section 5.5.3).

5.5.1 Error Bounds

At each step $k \in \{1, \dots, N_t\}$ of learning round t, where a query is received at the node, the model f_t is updated by the input-output query-relevant samples $\{(\mathbf{x}_{t,k}, y_{t,k}) \in \mathcal{D}_{t,k}\}$. The f_t model is parameterized by a vector $\mathbf{w}$, and the loss per input-output sample from $\mathcal{D}_{t,k}$ induced by the model is provided by $\ell_{t,k}(f_t(\mathbf{x}_{t,k}; \mathbf{w}), y_{t,k}) = \ell_{t,k}(\mathbf{w})$. The incremental updating rule is obtained by online SGD on the regularized convex loss:

$$\mathcal{L}_{t,k}(\mathbf{w}) \;=\; \ell_{t,k}(\mathbf{w}) + \frac{\zeta}{2}\|\mathbf{w}\|^2, \tag{5.11}$$

where $\zeta > 0$ controls the amount of regularization. Given a learning rate $\eta_k \in (0,1)$, the update rule for model f_t parameter $\mathbf{w}_{t,k}$ is provided in (5.4). Theorem 5.3 provides the upper bound of the cumulative error of the refining process in Algorithm 2.

Theorem 5.3. *(Total Cumulative Error Bound) Assume that $\|\mathbf{w}\| \leq W$ and the L-Lipschitz convex loss $\ell_{t,k}$. By setting $\zeta = \frac{L}{W}\sqrt{\frac{\log N+1}{N}}$ and $\eta_k = \frac{1}{k\zeta}$, the total cumulative error $\mathcal{E}_T = \sum_{t=1}^{T} \mathcal{E}_t$, with $\mathcal{E}_t = \sum_{k=1}^{N} \ell_{t,k}(f_t)$ at learning round t, is upper bounded by:*

$$\mathcal{E}_T \;=\; \sum_{t=1}^{T}\sum_{k=1}^{N} \ell_{t,k}(\mathbf{w}_t^*) + TWL\sqrt{N(\log N + 1)}, \tag{5.12}$$

where $N_t = N > 0, \forall t$ and $\mathbf{w}_t^* = \arg\min_{\|\mathbf{w}\| \leq W} \sum_{k=1}^{N_t} \ell_{t,k}(\mathbf{w})$ being the best model parameter during round t.

This theorem provides crucial insight into the quality guarantees of our approach. It shows that the total error of our model refinement process is bounded by the sum of the errors of the best possible models in each round, plus a regularization term that grows with T (number of rounds), W (parameter bound), L (Lipschitz constant), and N (queries per round). This bound ensures that our incremental refinement approach remains effective and stable, even when processing many queries across multiple rounds.

Remark 5.2. *In our DCL case, we deal with the cumulative error of the intermediate models during the refining process, which is progressively fed with relevant training data guided by the issued queries, and not of a standalone model during a traditional learning phase.*

5.5.2 Computational Complexity

The computational complexity of our approach can be analyzed in terms of both time and space requirements for each component of our framework.

Time Complexity

For a single learning round t with N_t queries:

- Prediction phase: $O(N_t \cdot P)$, where P is the prediction cost of the model f_t
- Relevance region identification: $O(N_t \cdot |\mathcal{D}|)$ for exhaustively searching through the dataset
- Incremental model training: $O(N_t \cdot |\mathcal{D}_{t,k}| \cdot I)$, where I is the number of iterations per update and $|\mathcal{D}_{t,k}|$ is the cardinality of the relevant region
- Model selection and replacement: $O(1)$ per round

The total time complexity of our framework for T rounds is $O(T \cdot N_t \cdot (P + |\mathcal{D}| + |\mathcal{D}_{t,k}| \cdot I))$. In practice, we maintain efficient data structures to reduce the search complexity for the identification of the relevance region, bringing the average case closer to $O(T \cdot N_t \cdot \log(|\mathcal{D}|))$ for this component.

Space Complexity

The space requirements include:

- Original dataset: $O(|\mathcal{D}|)$
- Model parameters: $O(|\mathbf{w}|)$ for each model, with a total of $O(T \cdot |\mathbf{w}|)$ in the worst case
- Relevant data regions: $O(|\mathcal{D}_R|)$, typically $|\mathcal{D}_R| \ll |\mathcal{D}|$

5.5.3 Optimal Learning Round Horizon

Let us assume T learning rounds for our mechanism, which corresponds to T training processes of a new model in each round. We notate with $c > 0$ the cost for training a model in round $t \in \{1, \ldots, T\}$. The cost for training models (including convex models and non-convex deep learning models) is typically modeled by data size, model size, and training volume (Sharir et al., 2020). In addition, such cost is influenced, for example, by the optimizer and batch size, and by hardware such as the GPU clock speed and the GPU memory bandwidth (Justus et al., 2018). As showcased in (Sharir et al., 2020) and in our experiments, the training effort is directly linked to the number of relevant training instances (percentage of the whole data), which yields stable after convergence. Without spoiling the analysis, we can approximate the training cost per round with a constant value c.

As we proceed with more rounds, the expected loss per round is expected to decrease. This is a result of a more accurate model w.r.t. query patterns, as expected. Moreover, as provided in Lemma 5.1, Lemma 5.2 in Section 5.4.2, we observe an exponentially decaying trend of the average loss per round. This indicates that, at some round T, we should stop our process (i.e., stop feeding the model with relevant data identified by executed queries at that round) since our method has captured most of the users' query access patterns. Hence, any further training and query execution can be considered redundant and inefficient w.r.t. cost. Evidently, the higher the learning horizon round T, the higher our confidence that the latest model f_T^* (or the updated best model at T) has fully captured the query access patterns and the training data are indeed the relevant data that reflect these patterns. However, this comes at the expense of more continuous training rounds. As more rounds are used to mine relevant data, we anticipate the expected loss to be smaller than that achieved by the intermediate models in earlier rounds. We then cope with the challenge of finding an optimal round T such that our confidence in having identified most of the relevant data regions is maximized. This is reflected by progressively small loss values, taking into account the cumulative cost we experience at every extra round.

Based on Lemmas 5.1 and 5.2, where we observe a progressive decay of the average loss per round t, without loss of generality, we assume that the loss values of the trained model at round t, ℓ_t, are drawn from a distribution with decaying mean μq^{t-1} with $\mu = \mathbb{E}[\ell_1]$ at round $t = 1$ and exponent factor $q \in (0, 1)$. As mentioned above, this stochastic decay of the loss ℓ_t per round t reflects our 'cumulative' confidence that we have discovered most of the relevant regions (since the performance of the future trained models increases w.r.t. accuracy).

Let us represent the cumulative confidence at the round horizon T:

$$S_T = \ell_1 + \cdots + \ell_T, \tag{5.13}$$

as the summation of loss values up to T. Evidently, the rate of increase of this summation decreases since, on average, the loss value at round τ is (stochastically/in expectation) smaller than the loss value at round τ' with $\tau > \tau'$. Furthermore, we need to take into consideration the cost per round c, which is accumulated up to T, thus, we obtain a total cost cT at the round horizon T. Our objective is to find a round horizon T that *maximizes the expected rate of confidence per round* taking into account the cost per round, i.e., maximizing the ratio $\frac{\mathbb{E}[S_T - cT]}{\mathbb{E}[T+1]}$. Apparently, due to the stochastic nature of the loss values per round, the confidence sum S_T is also random, thus, we take the expectation of this sum. We then need to devise a policy to find an optimal horizon round T that can maximize the expectation of rate of confidence per round. **Note:** The added unity in the denominator of this ratio represents the (by default) training of the initial model required to initialize our method. We formulate our problem of finding the optimal T w.r.t. cost per round c.

Problem 4. *Fix a learning cost $c > 0$ per round. Seek the optimal horizon round T at which* *we stop further training to maximize our confidence per round, i.e., find an optimal policy over $T \in \mathcal{T} = \{1, 2, \ldots\}$ that maximizes the expected rate of confidence per round:*

$$\max_{T \in \mathcal{T}} \frac{\mathbb{E}[S_T - cT]}{\mathbb{E}[T+1]}, \tag{5.14}$$

where $S_T = \ell_1 + \ldots + \ell_T$.

Theorem 5.4. *Let $\xi \in \mathbb{R}$, such that the $\max_{K \in \mathcal{T}} \mathbb{E}[Y_K - \xi T_K] = 0$ is attained at $T^* \in \mathcal{T} = \{1, 2, \ldots\}$. Then, $T^* \in \mathcal{T}$ is optimal for maximizing $\max_{K \in \mathcal{T}} \frac{\mathbb{E}[Y_K]}{\mathbb{E}[T_K]}$, with $Y_K = S_K - cT_K$. Moreover, if $\max_{K \in \mathcal{T}} \frac{\mathbb{E}[Y_K]}{\mathbb{E}[T_K]} = \xi$, and if the maximum is attained at $T^* \in \mathcal{T}$, then $\max_{K \in \mathcal{T}} \mathbb{E}[Y_K - \xi T_K] = 0$ and the maximum is attained at $T^* \in \mathcal{T}$.*

Theorem 5.5. *The optimal rate of confidence ξ and optimal round horizon T^* of Problem 4 are provided by iteratively solving the expressions starting with any arbitrary value of $\xi_0 < \mu - c$:*

$$T^* = \frac{\log \mu/(\xi + c)}{\log 1/q}, \quad \xi = \frac{\mu \frac{1-q^{T^*}}{1-q} - cT^*}{T^*+1} \tag{5.15}$$

5.6 Relevant Data & Query Updates

In dynamic environments, both data distributions and query patterns may evolve over time, necessitating mechanisms to maintain model accuracy. This section presents two complementary mechanisms: one for detecting and adapting to changes in the underlying data distribution (Section 5.6.1), and another for handling shifts in query access patterns (Section 5.6.2). Both mechanisms employ optimal stopping theory to precisely determine when model retraining should be triggered.

5.6.1 Relevant Data Region Update Mechanism

In dynamic environments, the underlying data distribution is subject to change; therefore, the identified boundaries of relevant data regions may change, producing less accurate models due to changes in the model selection discussed in Section 5.4.3. Assuming an incoming sample $(\mathbf{x}, y)$, the discriminator f_C predicts it as relevant or irrelevant based on the decision $f_C(\mathbf{x}) = +1$ or -1, respectively. In the predicted relevant case, the model f_T^* is expected to provide a more accurate prediction than the model f_0. That is, given the prediction errors $e^*(\mathbf{x}) = y - f_T^*(\mathbf{x})$ and $e_0(\mathbf{x}) = y - f_0(\mathbf{x})$ (yielded on validation data), we obtain:

$$\begin{cases} \text{C1: } e^*(\mathbf{x}) \leq e_0(\mathbf{x}) \text{ and } f_C(\mathbf{x}) = +1 \text{ (agreement)} \\ \text{C2: } e^*(\mathbf{x}) > e_0(\mathbf{x}) \text{ and } f_C(\mathbf{x}) = -1 \text{ (agreement)} \\ \text{C3: } e^*(\mathbf{x}) \leq e_0(\mathbf{x}) \text{ and } f_C(\mathbf{x}) = -1 \text{ (disagreement)} \\ \text{C4: } e^*(\mathbf{x}) > e_0(\mathbf{x}) \text{ and } f_C(\mathbf{x}) = +1 \text{ (disagreement)} \end{cases}$$

The sample $(\mathbf{x}, y)$ is classified as relevant or irrelevant when cases C1 and C2 are in agreement, respectively. Case C1 denotes that the decision of using f_T^* for predictions given that the sample is classified as relevant is correct (with the actual error being less than that of using f_0). Case C2 denotes the opposite, i.e., we correctly decide to use f_0 for predictions given irrelevant classified samples. In these cases, the underlying data distribution does not change and does not spoil the classification of the relevant and irrelevant regions identified. C3 and C4 cases denote a disagreement on using the f_0 and f_T^* models upon relevant and irrelevant samples, respectively. Based on cases C1-C4, at the time instance k, we define the indicator function $\mathcal{I}_k(\mathbf{x}_k)$ for the incoming input $\mathbf{x}_k$:

$$\mathcal{I}_k(\mathbf{x}_k) = \begin{cases} 1 & \text{if C1 or C2 hold true,} \\ 0 & \text{if C3 or C4 hold true,} \end{cases} \tag{5.16}$$

with *probability of agreement* $\mathcal{P}_A(\text{C1} \vee \text{C2})$, where there is an agreement based on cases C1 or C2. Given a series of incoming data $\{(\mathbf{x}, y)_k\}$ to a node, the node classifies them according to f_C in high hopes that the model f_T^* can *still* be performing well for new relevant data. However, if the underlying data distribution changes (concept drift), then new data regions should be investigated whether they form new relevant regions or not. If this change is not significant or does not occur, then model f_T^* and discriminator f_C can insert incoming data into the relevant or irrelevant datasets accordingly. This is reflected by a continuous observation of events $\{\mathcal{I}_k = 1\}$. Hence, in this case, the node does not need to further refine the model or the discriminator. However, when events $\{\mathcal{I}_k = 0\}$ start to appear either sporadically, or gradually, or in some cases abruptly, this reflects a potential change in the data trend w.r.t. their relevance. That is, cases C3 or C4 are triggered indicating either some new regions of relevant data being classified as irrelevant (C3) or the refined model does not outperform the initial model over relevant data (C4). If this disagreement between the relevance classification and the associated performance of the model f_T^* commences then, at some point, the node should decide to refine the model and update the discriminator, to capture these trends. The challenge now becomes the sequential decision-making of the models' updates based on the indicators $\{\mathcal{I}_k\}$. We formulate this problem as follows. Consider the cumulative sum of indicators (random variables) $\mathcal{I}_1, \ldots, \mathcal{I}_k$ up to time k:

$$\mathcal{S}_k = \sum_{\kappa=1}^{k} \mathcal{I}_\kappa. \tag{5.17}$$

Let also a data-change risk factor (represented as probability) $\beta \in (0,1)$ denoting the event where a change in the data relevance trend is not happening, thus, the refined model and/or discriminator are still up-to-date. We then define the random variables $\mathcal{B}_1, \ldots, \mathcal{B}_k$ with $\mathcal{B}_k = 1$ there is no change in the data relevance trend and $\mathcal{B}_k = 0$, there is a data relevance change, i.e., $\mathcal{P}(\mathcal{B}_k = 1) = \beta$ and $\mathcal{P}(\mathcal{B}_k = 0) = 1 - \beta$. The node desires to increase as much the cumulative sum $\mathcal{S}_k$ as possible under a non-change in the data relevance trend, thus, avoiding any model and/or discriminator update. This reflects the 'reward' of the node up to time k. The reward $\mathcal{R}_k$ is bounded below by zero and above by $\max_k\{\mathcal{R}_k\} = \sum_{\kappa=1}^{K} \mathcal{I}_k = \mathcal{S}_K$, where K is the number of observations where no change has been noticed, $K = \max\{k \geq 1 : \prod_{\kappa=1}^{k} \mathcal{B}_\kappa = 1\}$. On the other hand, when a significant change in the data relevance trend happens (with probability $1 - \beta$) then, the node 'loses' the entire reward accumulated so far, thus, enforcing itself to start off a new model refining process. Therefore, we define the reward objective:

$$\mathcal{R}_k = \prod_{\kappa=1}^{k} \mathcal{B}_k \sum_{\kappa=1}^{k} \mathcal{I}_\kappa. \tag{5.18}$$

The objective is to find a stopping policy that maximizes the expected reward in (5.18) by observing the sequence of events $\{\mathcal{I}_k\}$. This time-optimized objective is based on the principles of Optimal Stopping Theory (Albert & Goran, 2006). Once such policy is devised, i.e., an optimal time k^*, which maximizes the expected reward in (5.18), then the node keeps using the refined model and discriminator up to k^*, and then starts off a new process to accommodate the new data relevance trend.

Problem 5. *Given a series of events $\{\mathcal{I}_k\}$, find an optimal policy $k^* > 0$ where the supremum is attained:*

$$\sup_k \mathbb{E}[\mathcal{R}_k]. \tag{5.19}$$

Theorem 5.6. *The optimal stopping policy k^* of the Problem 5 exists and is unique.*

Theorem 5.7. *Given the series of indicators $\mathcal{I}_1, \ldots, \mathcal{I}_k$ and a risk factor β, the node starts off a new process of the refined model and discriminator at the optimal time k^*:*

$$k^* = \min\left\{k > 1 : \sum_{\kappa=1}^{k} \mathcal{I}_\kappa \geq \frac{\beta}{1-\beta} \mathcal{P}_A(C1 \vee C2)\right\}, \tag{5.20}$$

where the probability of agreement $\mathcal{P}_A(C1 \vee C2)$ is a function of the joint distribution of prediction errors $e^*(\mathbf{x})$ and $e_0(\mathbf{x})$.

The data relevance update criterion in Equation (5.20) has an intuitive interpretation: the decision to update the model is triggered when the accumulated evidence of agreement (represented by $\sum_{\kappa=1}^{k} \mathcal{I}_\kappa$) exceeds a threshold that depends on both the risk factor β and the probability of agreement. Higher values of β (indicating higher confidence in the current model) lead to higher thresholds, requiring more evidence before triggering an update. This adaptive mechanism ensures that model updates occur neither too frequently (which would waste computational resources) nor too rarely (which would lead to outdated models).

5.6.2 Query Update Mechanism

On the query patterns side, given a dynamic environment where applications/end-users are interested in numerous and diverse analytics tasks, the query distribution (access patterns) to node's data is subject to change. That is, new regions of interest and, therefore, relevance should be explored and identified by our mechanism. Once these new query patterns start to evolve, the mechanism should start merging the current relevant regions with the newly identified ones.

Inevitably, the process begins by obtaining a further refined model capable of accommodating the new query patterns. Similar to the mechanism in Section 5.6.1, we introduce a query updates mechanism based on incoming queries $\{\mathbf{q}_k\}$. An incoming query is classified as relevant w.r.t. $f_C(\mathbf{q})$. We anticipate this query to fall into a relevant region, signifying our confidence in such classification. We quantify the *degree of belongingness* of a relevant query to a relevant region by the distance of the query $\mathbf{q}$ from a representative point $\mathbf{u}_m \in \mathcal{X}$, where M representatives $\{\mathbf{u}_m\}_{m=1}^{M}$ quantize the relevant input space $\{\mathbf{x} : (\mathbf{x}, y) \in \mathcal{D}_R\}$ into M clusters.

A vector quantization method, e.g., k-means and Gaussian Mixture Models clustering (McLachlan & Peel, 2000) can be adopted to cluster the input space in M clusters, each represented by $\mathbf{u}_m, m = 1, \ldots, M$. However, the number of representatives cannot be known in advance, and the distance of a query to its closest representative is key information for assessing a gradual change in the query distribution. Therefore, we introduce a dynamic and adaptive quantization algorithm based on the principles of adaptive resonance theory (Masuyama et al., 2022), which is leveraged by introducing a variable vigilance to meet the needs of our problem. Vigilance represents the current average distance of an input to its closest (current) representative.

Initially, we designate the first relevant input $\mathbf{u}_1 = \mathbf{x}_1$ as the first representative, with $\mathcal{U} = \{\mathbf{u}_1\}$ (i.e., $M = 1$). As the quantization incrementally evolves, for each relevant input $\mathbf{x}_k : (\mathbf{x}_k, y_k) \in \mathcal{D}_R$, its distance $\|\mathbf{x}_k - \mathbf{u}^*\|_2$ from its closest representative

$$\mathbf{u}^* = \arg \min_{m=1,\ldots,M} \|\mathbf{x}_k - \mathbf{u}_m\|_2 \tag{5.21}$$

is compared against the current vigilance threshold δ_k. The vigilance δ_k is incrementally updated based on the average of all the distances of inputs to their representatives from their clusters $\mathcal{U}_1, \ldots, \mathcal{U}_{M_k}$ so far, with $\mathcal{D}_R = \cup_{m=1}^{M_k} \mathcal{U}_m$, i.e.,

$$\delta_k = \alpha\delta_{k-1} + (1-\alpha)\frac{1}{M_k}\sum_{m=1}^{M_k}\left(\frac{1}{|\mathcal{U}_m|}\sum_{\mathbf{x}\in\mathcal{U}_m}\|\mathbf{x}-\mathbf{u}_m\|_2\right), \tag{5.22}$$

with vigilance updating learning rate $\alpha \in (0, 1)$.

If the distance to its closest representative is less than the vigilance δ_k, the input $\mathbf{x}_k \in \mathcal{U}_*$ is assigned to representative $\mathbf{u}^*$ and updates it accordingly:

$$\mathbf{u}^*_{(k+1)} = \mathbf{u}^*_{(k)} + \rho(\mathbf{x}_k - \mathbf{u}^*_{(k)}), \tag{5.23}$$

with a quantization learning rate $\rho \in (0, 1)$. Otherwise, a new cluster $\mathcal{U}_{M_{k+1}}$, $M_{k+1} = M_k + 1$ is formed with representative $\mathbf{u}_{M_{k+1}} = \mathbf{x}_k$. At the end of the quantization, we obtain M clusters $\{\mathcal{U}_m\}_{m=1}^M$ with their representatives $\{\mathbf{u}_m\}_{m=1}^M$ and final vigilance δ. The quantization process is provided in Algorithm 3.

Algorithm 3 Incremental Data Relevant Space Quantization

```
1: Input: Relevant data $\mathcal{D}_R$
2: Initialize: $\delta_1$, $\mathbf{u}_1 = \mathbf{x}_1$, $\mathcal{U}_1 = \{\mathbf{x}_1\}$, $M_1 = 1$
3: for $k = 2, \ldots, |\mathcal{D}_R|$ do
4:     Find closest $\mathbf{u}^*$ using (5.21)
5:     if $\|\mathbf{x}_k - \mathbf{u}^*\|_2 \leq \delta_k$ then
6:         Update $\mathbf{u}^*$ using (5.23)
7:         Add $\mathbf{x}_k : \mathcal{U}_* = \mathcal{U}_* \cup \{\mathbf{x}_k\}$
8:     else
9:         $M_{k+1} = M_k + 1$
10:        Create new cluster with $\mathbf{u}_{M_{k+1}} = \mathbf{x}_k$
11:    end if
12:    Update vigilance $\delta_k$ using (5.22).
13: end for
14: return vigilance $\delta$, representatives $\{\mathbf{u}_m\}_{m=1}^M$.
```

For all the relevant queries $\mathbf{q} : f_C(\mathbf{q}) = 1$, consider the distances δ^* to closest representatives $\mathbf{u}^*$:

$$\delta^* = \|\mathbf{q} - \mathbf{u}^*\|_2, \text{ with } \mathbf{u}^* = \arg\min_m \|\mathbf{q} - \mathbf{u}_m\|_2, \tag{5.24}$$

and let $\mathcal{P}_{\Delta^*}(\delta^*)$ be the probability distribution of the closest distances δ^*. Given a new incoming query $\mathbf{q}$ to the node, we determine its *potential* relevance by evaluating whether there is **agreement** between the classification outcome $f_C(\mathbf{q})$ and the query's distance to its closest representative being less than the vigilance threshold δ. In the former case, we do not experience a change in the identified query distribution w.r.t. relevance. On the other hand, a disagreement might trigger potential changes in the query distributions, thus obtaining the following cases:

$$\begin{cases} \text{Q1: } \delta^* \leq \delta \text{ and } f_C(\mathbf{q}) = +1 \text{ (agreement)} \\ \text{Q2: } \delta^* > \delta \text{ and } f_C(\mathbf{q}) = -1 \text{ (agreement)} \\ \text{Q3: } \delta^* \leq \delta \text{ and } f_C(\mathbf{q}) = -1 \text{ (disagreement)} \\ \text{Q4: } \delta^* > \delta \text{ and } f_C(\mathbf{q}) = +1 \text{ (disagreement)} \end{cases}$$

Case Q1 indicates that the new query has been correctly considered as relevant since it falls into relevant (quantized) regions. Q2 asserts that a correctly classified irrelevant query falls into irrelevant regions. Q3 and Q4 signify that irrelevant queries appear in relevant regions and vice versa, thus raising a disagreement. Upon a series of incoming queries $\{\mathbf{q}_k\}$, we define:

$$\mathcal{I}'_k(\mathbf{q}_k) = \begin{cases} 1 & \text{if Q1 or Q2 hold true,} \\ 0 & \text{if Q3 or Q4 hold true,} \end{cases} \tag{5.25}$$

with probability of agreement $\mathcal{P}_A(\text{Q1} \vee \text{Q2})$. Using a reasoning similar to that in Section 5.6.1, we introduce the query change factor (represented as probability) $\beta' \in (0, 1)$ that indicates the stage at which the query distribution changes, thus new relevant regions should be identified. By introducing the random variables $\Gamma_1, \ldots, \Gamma_k$, with $\mathcal{P}(\Gamma = 1) = \beta'$ and $\mathcal{P}(\Gamma = 0) = 1 - \beta'$, and the cumulative sum of $\sum_{\kappa=1}^{k} \mathcal{I}'_k$, the node seeks to maximize this sum under the condition that there are no significant changes in the query distribution. However, when a series of events corresponding to cases Q3 and/or Q4 ($\mathcal{I}'_k = 0$) comes into place, then the node should start a new refinement process accommodating the changes of the query distribution along with identifying novel relevant data regions. The objective is to find a stopping policy to maximize the expected reward (similarly as in Section 5.6.1):

$$\mathcal{R}'_k = \prod_{\kappa=1}^{k} \Gamma_k \sum_{\kappa=1}^{k} \mathcal{I}'_\kappa. \tag{5.26}$$

Problem 6. *Given a series of events $\{\mathcal{I}'_k\}$, find an optimal policy $k^* > 0$ where the supremum* *is attained:*

$$\sup_k \mathbb{E}[\mathcal{R}'_k] \tag{5.27}$$

As proved in Section 5.6.1, the optimal stopping policy for problem 6 exists and is unique, as provided in Theorem 5.8

Theorem 5.8. *Given the series of indicators $\mathcal{I}'_1, \ldots, \mathcal{I}'_k$ and a risk factor β', the node starts off a new process for the refined model at optimal time k^*:*

$$k^* = \min \left\{ k > 1 : \sum_{\kappa=1}^{k} \mathcal{I}'_\kappa \geq \frac{\beta'}{1-\beta'} \mathcal{P}_A(Q1 \vee Q2) \right\}, \tag{5.28}$$

where the probability of agreement $\mathcal{P}_A(Q1 \vee Q2)$ depends on the distribution of representative distances and classifier outcomes.

Similar to the data update mechanism, the query update criterion in Equation (5.28) provides an adaptive threshold that balances stability and responsiveness to changing query patterns. The risk factor β' allows system administrators to control how quickly the system adapts to new query trends, with higher values of β' resulting in more conservative updating behavior, while lower values allow for more rapid adaptation to emerging query patterns.

5.6.3 Adaptability Characteristics

Our dual update mechanisms provide a comprehensive framework for maintaining model accuracy in dynamic environments. The two mechanisms offer complementary benefits:

- **Data Update Mechanism:** Responds to concept drift in the underlying data distribution, ensuring that the model remains accurate even when the statistical properties of the data change over time. This is particularly important in environments where data characteristics evolve gradually, such as sensor networks with aging components or customer behavior data showing seasonal patterns.
- **Query Update Mechanism:** Adapts to changing user interests or application requirements reflected in query patterns. This enables the system to maintain high performance even when the types of queries shift, such as when new types of analytics tasks become popular or when different user groups begin accessing the system.

The risk factors β and β' provide configurable parameters that allow system administrators to control the sensitivity of these mechanisms. Higher values result in more conservative behavior (requiring stronger evidence before triggering updates), while lower values make the system more responsive to changes. This flexibility allows the approach to be tailored to different application domains with varying stability requirements.

Both mechanisms guarantee optimal timing for model updates through their theoretical foundations in optimal stopping theory. This ensures that computational resources are used efficiently: updates occur neither too frequently (which would waste resources) nor too rarely (which would lead to outdated models).

The empirical evaluation in Section 5.7 demonstrates the effectiveness of these mechanisms in various change scenarios, showing that our approach maintains high prediction accuracy even in the face of substantial distribution shifts.

Algorithm 4 Data & Query Sequential Updates Mechanisms (in parallel)

```
1:  Input: Refined and initial models $f_T^*$, $f_0$, discriminator $f_C$, factors $\beta, \beta'$.
2:  *Data Updates Mechanism*
3:  Initialize cumulative sum $\mathcal{S}_0 = 0$.
4:  for $k \geq 1$ do
5:      Receive input-output data $(\mathbf{x}_k, y_k)$
6:      Obtain indicator $\mathcal{I}_k$ using (5.16)
7:      Update $\mathcal{S}_k = \mathcal{S}_{k-1} + \mathcal{I}_k$
8:      if criterion in (5.20) holds then
9:          Invoke Algorithm 2
10:     end if
11: end for
12: *Query Updates Mechanism*
13: Initialize cumulative sum $\mathcal{S}'_0 = 0$.
14: for $k \geq 1$ do
15:     Receive query $\mathbf{q}_k$
16:     Obtain indicator $\mathcal{I}'_k$ using (5.25)
17:     Update $\mathcal{S}'_k = \mathcal{S}'_{k-1} + \mathcal{I}'_k$
18:     if criterion in (5.28) holds then
19:         Invoke Algorithm 2
20:     end if
21: end for
```

5.7 Experimental Evaluation

This section presents a comprehensive empirical evaluation of our approach. We first describe our datasets and query workload generation methodology (Section 5.7.1), then outline our performance metrics and comparison approaches (Section 5.7.2). We evaluate the impact of our data-centric learning mechanism (Section 5.7.3), assess the optimality of our learning horizon (Section 5.7.4), and conduct comparative assessments against baseline approaches (Section 5.7.5). Finally, we evaluate our method's performance under data and query distribution drifts (Section 5.7.6).

5.7.1 Datasets & Query Workloads

We used two real datasets to evaluate and compare the performance of our approach with relevant work. These datasets were specifically chosen for their distinct distributional characteristics and their suitability for testing query-driven approaches:

Medical Cost Personal Dataset (MCPD)

The MCPD dataset (Choi, 2018) consists of $d = 7$ attributes (predictors), including age, sex, BMI, children, smoker, region, and charges. We used both 'insurance charges' and 'BMI' as output variables in separate experiments. This dataset was selected because it:

- Contains multi-modal distributions with complex interactions between features
- Exhibits varying data densities across feature space, which helps test our relevant data discovery mechanism
- Includes natural distribution variations that test the adaptive capabilities of our model
- Presents a realistic healthcare analytics scenario where query patterns may focus on specific patient cohorts

Combined Cycle Power Plant (CCPP) Dataset

The CCPP dataset (Tüfekci, 2014) consists of environmental sensor data with $d = 4$ predictors including ambient temperature, pressure, relative humidity, and exhaust vacuum, with the output being the 'electrical energy' of a power plant. We chose this dataset because it:

- Features continuous, real-valued features with temporal dependencies
- Contains periodic patterns that test the adaptive capabilities of our model
- Represents a realistic industrial application with well-documented ground truth
- Allows for controlled simulation of concept drift in sensor data

Query Generation

We generate query patterns based on the access patterns of real users as described in (Anagnostopoulos & Triantafillou, 2017), (Savva et al., 2020a), and (Savva et al., 2020b). We employ two query generation approaches to evaluate our method under different scenarios:

Direct Query Generation For direct generation, we add a selective filter over the output variable y of the dataset $\mathcal{D}$ to separate it into $\mathcal{D}_R$ and $\mathcal{D}_I$ by:

$$\mathcal{D}_R = \{(\mathbf{x}, y)\} : (\mathbf{x}, y) \in \mathcal{D}, y \in [y_L, y_H], \; \mathcal{D}_I = \mathcal{D} \setminus \mathcal{D}_R, \tag{5.29}$$

where $y_{\min} \leq y_L$ and $y_H \leq y_{\max}$ are uniformly selected lower and upper bounds of the output $y \in [y_{\min}, y_{\max}]$. Then, we sample queries $\mathcal{Q}$ directly from $\mathcal{D}_R$, thus generating queries lying on the data space defined by the entire dataset $\mathcal{D}$.

Drift-based Query Generation To evaluate our approach with diverse query patterns, we follow a strategy to generate queries with deviation by inducing controllable concept drifts in the input and output data space. This approach generates (drifted) queries around data regions, where past queries have been issued by users/applications. As in direct query generation, we acquire $\mathcal{D}_R$ and $\mathcal{D}_I$ according to (5.29). Then, using the fitter (Cokelaer et al., 2023), from more than 100 different types of distributions, we search for the best Ω^* that fits the output values $\{y_j\} \in \mathcal{D}_R$. By randomly sampling multiple times from Ω^*, we acquire the outputs $\hat{\mathcal{Y}}$. For each $\hat{y} \in \hat{\mathcal{Y}}$, we select the input-output pair $(\mathbf{x}, y)$ within $\mathcal{D}$ that has the smallest distance between $\hat{y}$ and y. Hence, by combining the input $\mathbf{x}$ from the pair $(\mathbf{x}, y)$ with the acquired output $\hat{y}$, we derive the query set $\tilde{\mathcal{Q}}$.

Then, for each input vector in this set, we add Gaussian noise to each dimension to obtain the final queries $\mathcal{Q}$ with controlled drift. This allows us to systematically evaluate our method's ability to adapt to changing query patterns.

5.7.2 Performance Metrics & Models

Performance Metrics

For predictive analytics tasks, we evaluated several regression models including Gaussian Process Regression (GPR), Support Vector Regression (SVR) and Gradient Boosting Regression (GBR). After assessing their performance with respect to prediction error and stability over optimized tuned parameters, we found that GBR performed best across our experimental scenarios. We therefore adopted GBR ensembles with 100 estimators, squared error loss, and sub-sampling rate 1.0 for our core experiments.

For the discriminator model, after evaluating multiple binary classifiers (including Logistic Regression and Random Forest), we found the most effective classifier in our context to be a Bagging Classifier with an ensemble of 10 SVMs using the non-linear double-layer hierarchical voting scheme described in (Kim et al., 2002).

To assess prediction accuracy, we use the Root Mean Squared Error (RMSE) over testing and validation data, denoted by ϵ_t and ϵ_t^v, respectively. To provide deeper insights into the model refinement process with respect to ground truth relevant data $\mathcal{D}_R$, we introduce three data relevance metrics: R_t, R_t^* and $\bar{R_t}^*$. Due to the inherent randomness of the issued queries, these metrics help us understand the fluctuation of the predictive performance by showing the similarity between the relevant datasets $\mathcal{D}_{t-1}$ and $\mathcal{D}_t$ at consecutive rounds $t-1$ and t:

$$R_t = \frac{|\mathcal{D}_t \cap \mathcal{D}_{t-1}|}{|\mathcal{D}_t|},\ R_t^* = \frac{|\mathcal{D}_t \cap \mathcal{D}_R|}{|\mathcal{D}_R|},\ \bar{R_t}^* = \frac{|\mathcal{D}_t \cap \mathcal{D}_R|}{|\mathcal{D}_t \cap \mathcal{D}_R| + |\mathcal{D}_t \setminus \mathcal{D}_R| + |\mathcal{D}_R \setminus \mathcal{D}_t|}. \quad (5.30)$$

The stability metric R_t quantifies how much of the current relevant dataset is preserved from the previous round. The relevance ratio R_t^* shows the percentage of the ground truth relevant data captured in $\mathcal{D}_t$, helping to explain the performance of model f_t. Ideally, $R_t^* \rightarrow 1$ as $t \rightarrow T$, indicating that by the end of the process, the obtained relevant dataset $\mathcal{D}_T^*$ contains almost all instances from the ground truth relevant data $\mathcal{D}_R$.

While R_t^* measures *relevance recall*, we further introduce $\bar{R}_t^*$ based on Tversky's index (Tversky, 1977), which provides a balanced representation of the quality of $\mathcal{D}_t$ by also considering *irrelevance precision*. The difference $R_t^* - \bar{R}_t^*$ indicates the presence of irrelevant data in $\mathcal{D}_t$ - when $R_t^* - \bar{R}_t^* = 0$, it means that $\mathcal{D}_t \setminus \mathcal{D}_R = \emptyset$, i.e., no irrelevant data are included.

Comparison Approaches

We refer to our proposed approach as **Model A** and compare it against several baselines and state-of-the-art approaches:

Model B As elaborated in Section 5.2.2, the most relevant approaches to ours are (Anagnostopoulos & Triantafillou, 2017) and (Savva et al., 2020b), which rely on the QDL paradigm. We implemented their common mechanism, referred to as **Model B**, for comparative assessment. Unlike our incremental approach, Model B operates at a higher granularity level, training a model f_B in one go using all accumulated queries. It follows three steps:

1. Query quantization: Cluster the accumulated queries Q into K clusters with centroids $\{\mathbf{w}_k\}$.
2. Relevant data selection: For each centroid $\mathbf{w}_k$, gather all data points whose distance from the centroid is less than threshold ϑ: $\mathcal{D}_R = \bigcup_{\mathbf{w}_k} (\mathbf{x}, y) \in \mathcal{D} : \|\mathbf{x} - \mathbf{w}_k\|^2 \leq \vartheta$.
3. Model training: Train model f_B over the relevant data $\mathcal{D}_R$.

For fair comparison, we conducted a grid search over Model B's hyper-parameters (number of query clusters K and threshold ϑ) to find the best-performing values for each dataset and query pattern set. We observed that small values of K and ϑ resulted in $|\mathcal{D}_R| \rightarrow 0$ (insufficient relevant data), while high values led to $\mathcal{D}_R \cap \mathcal{D} \equiv \emptyset$ (treating all data as relevant).

To find appropriate ϑ values for each K, we calculated: $\vartheta_{(K)} = \varsigma \frac{1}{K} \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{\mathbf{x} \in C_k} \left\| \mathbf{x} - \boldsymbol{\mu}_k \right\|^2$, where C_k are data clusters, $\boldsymbol{\mu}_k$ are their centroids, and ς is a dataset-specific coefficient. We denote Model B's RMSE as $\epsilon^{\diamond}$.

To align Model A with Model B's range query approach, we parameterize the maximum expansion of $\mathcal{D}_t$ through the relevance parameter λ using $\psi = \frac{2\lambda}{y_{\max} - y_{\min}}\%$, which represents the percentage of the output range.

Baseline Models We compare our approach with two additional baseline models:

Model E0 (Global model over relevant data): Initial model f_0 trained with the *whole* dataset $\mathcal{D}$ but tested only on relevant data $\mathcal{D}_R$, with the resulting RMSE ϵ_0. This baseline demonstrates how a model trained on all data (including irrelevant data) performs when applied specifically to relevant data regions.

Model E1 (Global model over whole dataset): Initial model f_0 trained and evaluated over the whole dataset $\mathcal{D}$, with the corresponding RMSE $\bar{\epsilon}_0$. This represents the performance of a conventional model that does not distinguish between relevant and irrelevant data.

Ground Truth Model We also built a ground truth model f^* (**Model GT**) trained exclusively on the theoretical (ground truth) relevant data $\mathcal{D}_R$, without any instances from $\mathcal{D}_I$, obtaining RMSE ϵ^*. This represents the theoretical lower bound for predictive error, as it uses perfectly relevant training data.

Table 5.1: Experimental Parameters

Experiment	Parameter	Value(s)
Default	Queries per round N_t	20
	Optimal rounds T^*	30
	Regularization ζ, initial learning rate η_1	0.001, 0.1
	Initial error-ratio γ_0	2.5
	Sampling percentage η	0.15
MCPD (charges)	Coeff. ς, magnitude ψ	1, 2%
MCPD (BMI)	Coeff. ς, magnitude ψ	5, 4%
CCPP	Coeff. ς, magnitude ψ	30, 8%
Drifted Queries	Drifting from s.d. ν	3
	Coeff. ς, magnitude ψ	1, 2%
Update Policies	Data/query-change factor (β, β')	{0.8, 0.95}
	Learning cost/round c	0.1
	Vigilance and quantization rates (α, ρ),	(0.2, 0.01)

Table 5.1 shows the default and optimal per-experiment parameters used in our evaluation. Based on the optimal learning round mechanism introduced in Section 5.5.3, we determined $T^* = 30$ rounds for Model A (see Section 5.7.4 for details).

5.7.3 Impact of Data-Centric Learning

We first investigate the impact of our Data-Centric Learning (DCL) mechanisms introduced in Section 5.4.2 on the performance of the model refinement process. We focus on two key DCL components: (i) model selection and replacement (where the current model f_t can be replaced with the best model so far f_t^* when the consecutive error ratio exceeds the threshold γ), and (ii) sub-sampling from the best relevant data identified so far $\mathcal{D}_t^*$ (controlled by parameter η) and injecting these samples into the currently identified relevant training data $\mathcal{D}_t$.

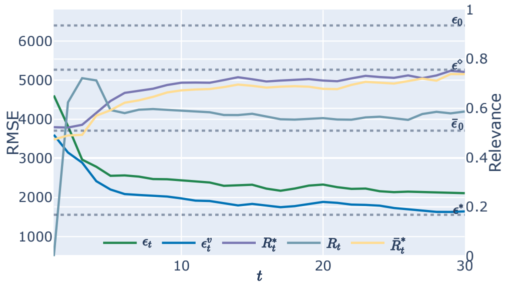


Figure 5.2: Impact of data-centric learning on RMSE and data relevance with DCL mechanisms. MCDP dataset (charges as output).

Figure 5.2 demonstrates the critical importance of these DCL mechanisms for performance stability and progressive refinement quality using the MCDP dataset (with charges as output); similar results were obtained with other datasets.

With the DCL mechanisms enabled, both test error ϵ_t and validation error ϵ_t^v show stable trajectories and reach significantly lower values than without DCL (approximately 50% reduction). The lower R_t values (< 0.6) indicate a more extensive exploration of the data space in each round, while the higher R_t^* and $\bar{R}_t^*$ values (close to 0.8) demonstrate that the approach is more effective in identifying relevant data regions while excluding irrelevant data.

The DCL mechanisms provide crucial stability to the refinement process by reducing the impact of query randomness and enabling a more effective exploration-exploitation balance. The model selection component prevents performance degradation by falling back to the best model when current iterations are not promising, while the sub-sampling component ensures that high-quality relevant data from previous rounds is preserved and utilized in future refinements.

5.7.4 Optimal Performance

Optimal Horizon T^*

We assessed the optimality of the learning round horizon T^* derived from our theoretical analysis in Section 5.5.3. Figure 5.3 (left) shows the expected rate of confidence introduced in Equation (5.14) and the corresponding optimal T^* as a function of the loss exponent factor q.

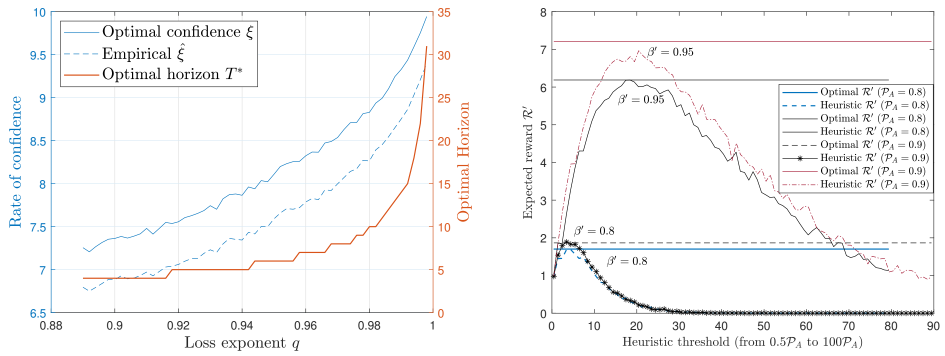


Figure 5.3: (Left) Rate of confidence and optimal horizon T^* against error loss exponent q; learning cost per round $c = 0.01$; (right) Expected reward in query-update policy against optimal and heuristic rules.

For our experimental datasets, we estimated q values and, with a fixed learning cost per round $c = 0.1$, determined that $T^* \approx 30$ represents the optimal horizon. When q approaches 1 (slower error decay), the optimal T^* increases, reflecting the need for additional rounds to explore relevant regions and refine the model. This coincides with an increase in the rate of confidence ξ. Conversely, for smaller q values (faster error decay), fewer rounds are needed as the approach rapidly identifies relevant regions in the early stages.

The figure also shows the empirical estimation of confidence $\hat{\xi}$, which closely tracks the optimal values across all q (average squared estimation error 0.21; $\sigma^2 = 0.001$), validating our theoretical framework.

Expected Reward in Query Updates

We also validated the optimality and uniqueness of our query update policy defined in Equation (5.28). Figure 5.3 (right) compares the expected reward $\mathcal{R}'$ achieved by our optimal policy against various heuristic stopping rules, for different probability of agreement values $\mathcal{P}_A \in \{0.8, 0.9\}$ and query-change factors $\beta' \in \{0.8, 0.95\}$.

The heuristic rules replace our optimal criterion with $\sum_{\kappa=1}^{k} \mathcal{I}'_\kappa \geq \omega \mathcal{P}_A$, where $\omega \in [0.5, 100]$. The results clearly show that maximum reward is achieved when $\omega \approx \frac{\beta'}{1-\beta'}$, which corresponds exactly to our theoretically derived optimal criterion in Equation

(5.28). This confirms that our policy represents a global optimum - stopping too early leads to premature model retraining, while stopping too late causes unnecessary delays in adapting to distribution changes.

Equivalent optimal results were observed with our data update policy, which follows the same underlying principle.

5.7.5 Comparative Performance Evaluation

We conducted a comprehensive comparative evaluation of all models (A, B, E0, E1, and GT) across three dataset configurations: MCPD with charges as output, MCPD with BMI as output, and CCPP. For these experiments, queries were generated using the direct query generation approach described in Section 5.7.1. All results represent averages from 100 independent runs, with shaded areas in Figures 5.4, 5.6, and 5.7 indicating the range (min to max) of test RMSE ϵ_t.

MCPD Dataset with Charges Output

Figure 5.4 (left) shows the performance results for the MCPD dataset with insurance charges as the output variable. The initial model error ϵ_0 (Model E0) significantly exceeds the global model error $\bar{\epsilon}_0$ (Model E1), indicating that the queries target regions that are particularly challenging for the global model. Model B ($\epsilon^{\diamond}$) shows only modest improvement over Model E0, demonstrating the limitations of its one-shot refinement approach.

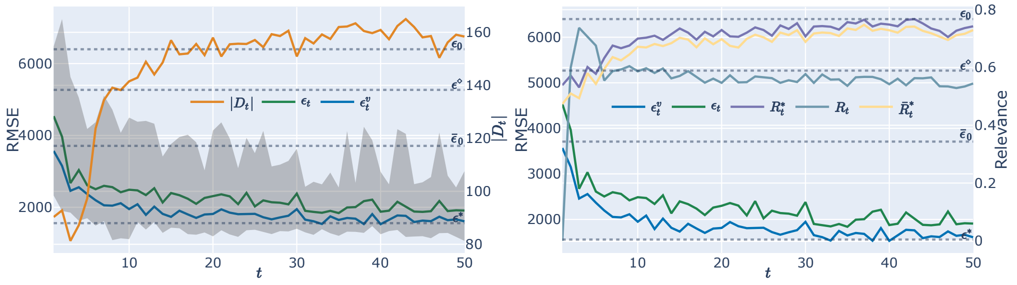


Figure 5.4: Left: RMSE and $|\mathcal{D}_t|$ per round for all approaches; Right: Data relevance R_t, R_t^* and $\bar{R}_t^*$ per round. MCPD dataset (charges as output).

In contrast, our Model A shows steady error reduction as rounds progress, with both the test error ϵ_t and the validation error ϵ_t^v converging to values very close to the performance of the ground truth model ϵ^* by round 30. The size of the relevant dataset $|\mathcal{D}_t|$ also stabilizes, indicating convergence. Notably, the lower bound of the RMSE variance (shaded area) occasionally drops below ϵ^*, suggesting that in some runs, our approach can even outperform the idealized ground truth model.

Figure 5.4 (right) provides insight into the dynamics of relevance of the data. Even after convergence, R_t remains relatively low (around 0.55), indicating substantial data turnover between rounds - approximately 45% of the data differs between consecutive rounds. However, R_t^* reaches around 0.8, showing that 80% of the theoretically optimal relevant data is consistently included in the training set $\mathcal{D}_t$. The small gap between R_t^* and $\bar{R}_t^*$ indicates that very little irrelevant data is erroneously included.

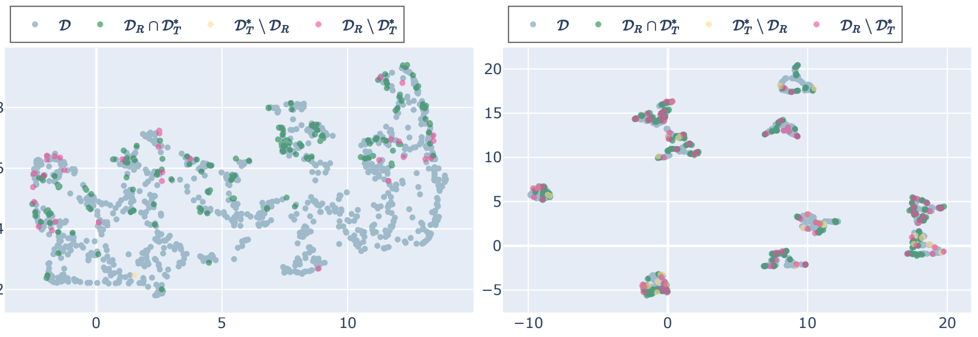


Figure 5.5: Distribution of the whole data $\mathcal{D}$, intersection between optimal and acquired relevant data $\mathcal{D}_R \cap \mathcal{D}_T^*$, and set differences $\mathcal{D}_T^* \setminus \mathcal{D}_R$ and $\mathcal{D}_R \setminus \mathcal{D}_T^*$ of (left) the MCPD dataset (charges as output) and (right) the MCPD dataset (BMI as output). All points are projected onto a 2-dimensional UMAP plane for illustration.

Figure 5.5 (left) visualizes the distribution of data samples projected onto a 2-dimensional UMAP plane, highlighting the intersection between ground truth relevant data and the best relevant data obtained by our approach ($\mathcal{D}_R \cap \mathcal{D}_T^*$), as well as the set differences ($\mathcal{D}_T^* \setminus \mathcal{D}_R$ and $\mathcal{D}_R \setminus \mathcal{D}_T^*$). The distribution shows that 82.72% of the relevant data was successfully identified (green dots), while only 0.63% irrelevant data were incorrectly included (yellow dots), confirming the high precision and recall of our relevance identification mechanism.

MCPD Dataset with BMI Output

Figure 5.6 shows the results for the MCPD dataset with BMI as the output variable. While Model A still outperforms Models B and E0, it achieves less dramatic improvement than with the charges output. After approximately 30 rounds, both ϵ_t and ϵ_t^v stabilize at values substantially below $\epsilon^\diamond$ and ϵ_0, but with a larger gap to ϵ^* than in the charges case.

The relevance metrics in Figure 5.6 (right) reveal why: R_t converges to approximately 0.5, while R_t^* reaches only about 0.6 and $\bar{R}_t^*$ drops to 0.4. This indicates that while 60% relevant data was identified, the training set $\mathcal{D}_T^*$ also contains a significant amount of irrelevant data.

Figure 5.5 (right) visually explains this challenge. The BMI-relevant data are distributed across isolated 'islands' in the feature space, making it difficult to identify coherent relevant regions. This fragmented distribution results from the fact that the BMI has weaker correlations with input attributes compared to charges. Neverthe-

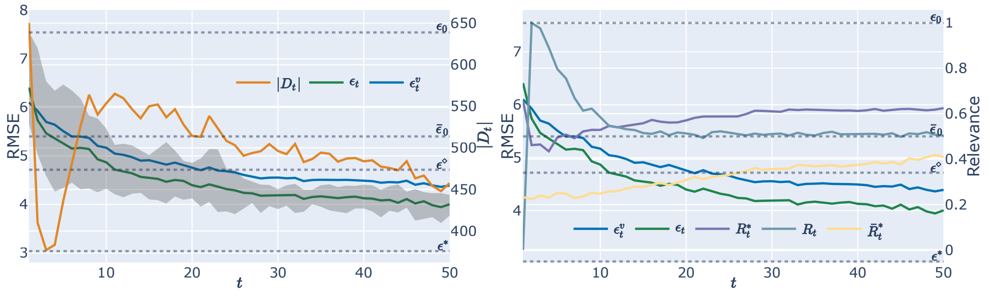


Figure 5.6: Left: RMSE and $|\mathcal{D}_t|$ per round for all approaches; Right: Data relevance R_t, R^*_t and $\bar{R}^*_t$ per round. MCPD dataset (BMI as output).

less, Model A still significantly outperforms baseline approaches, demonstrating its robustness even in challenging scenarios with complex relevance patterns.

CCPP Dataset

Figure 5.7 presents results for the CCPP dataset. Unlike the MCPD results, here we observe $\bar{\epsilon}_0 > \epsilon_0$, indicating that the relevant data regions are actually easier to predict for the global model than the dataset as a whole. Despite this different challenge profile, Model A still significantly outperforms Model B and the baseline models, achieving performance close to the theoretical optimum of Model GT.

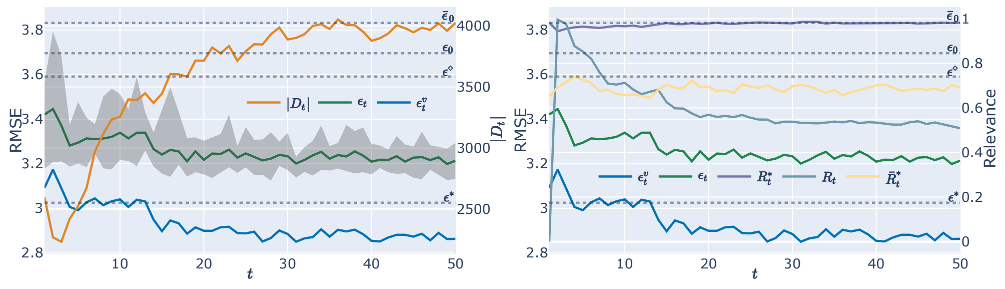


Figure 5.7: Left: RMSE and $|\mathcal{D}_t|$ per round for all approaches; Right: Data relevance R_t, R^*_t and $\bar{R}^*_t$ per round. CCPP dataset.

The relevance metrics in Figure 5.7 (right) and the data distribution in Figure 5.8 explain the strong performance on CCPP. After convergence, R^*_t approaches 1.0, indicating that Model A successfully identifies almost all relevant data. The difference between R^*_t and $\bar{R}^*_t$ (approximately 0.2) reflects the inclusion of some irrelevant data points, which is also visible as yellow dots in Figure 5.8. This occurs because some irrelevant regions overlap with relevant ones in the projected space, making perfect separation challenging.

Despite these challenges, Model A's DCL mechanism effectively manages the impact of irrelevant instances, allowing it to achieve predictive performance significantly better than Models B, E0, and E1, while closely approaching the theoretical optimum of Model GT.

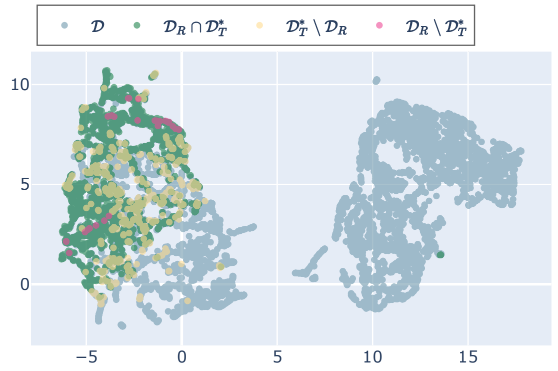


Figure 5.8: Distribution of whole data $\mathcal{D}$, intersection between optimal and acquired relevant data $\mathcal{D}_R \cap \mathcal{D}_T^*$, and set differences $\mathcal{D}_T^* \setminus \mathcal{D}_R$ and $\mathcal{D}_R \setminus \mathcal{D}_T^*$ of CCPP dataset. All points are projected onto a 2-dimensional UMAP plane for illustration.

Table 5.2: Comparative RMSE Results for All Models Across Datasets

Dataset	Model A	Model B	Model E0	Model E1	Model GT
	ϵ_T	$\epsilon^\diamond$	ϵ_0	$\bar{\epsilon}_0$	ϵ^*
MCPD (charges)	2.34 ± 0.21	5.18 ± 0.37	6.39 ± 0.24	4.85 ± 0.19	1.55 ± 0.16
MCPD (BMI)	3.67 ± 0.29	5.93 ± 0.42	7.21 ± 0.36	6.12 ± 0.28	2.85 ± 0.23
CCPP	1.97 ± 0.18	3.72 ± 0.31	4.48 ± 0.29	5.26 ± 0.34	1.64 ± 0.15
MCPD (charges) - drifted	2.51 ± 0.24	5.67 ± 0.41	6.82 ± 0.32	5.13 ± 0.27	1.88 ± 0.19

The exceptionally high R_t^* values (close to 1.0) indicate that Model A's progressive refinement process thoroughly explores and identifies relevant regions, while the DCL mechanisms effectively exclude the impact of irrelevant samples on model performance. This comprehensive coverage of relevant regions explains why Model A achieves such strong predictive performance compared to other approaches.

Table 5.2 summarizes the RMSE results for all models across the different datasets. Model A consistently outperforms Models B, E0, and E1 across all datasets, with performance improvements of up to 63% over Model E0. The closest performance to the theoretical optimum (Model GT) is achieved on the CCPP dataset, where the structured nature of the data facilitates more effective relevant region identification.

5.7.6 Performance Under Distribution Drifts

A critical requirement for real-world analytics systems is the ability to maintain performance when the underlying distributions change. In this section, we assess how all models perform under three types of drift scenarios: query distribution drift, data distribution drift, and combined drift that affects both distributions simultaneously.

Query Distribution Drift

We first evaluated performance under changes in query distribution while keeping the underlying data distribution constant. We used the drift-based query generation approach described in Section 5.7.1, with parameters specified in Table 5.1.

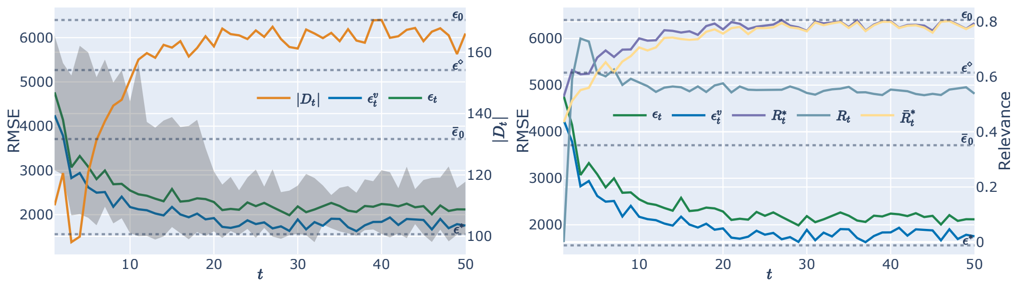


Figure 5.9: Left: RMSE and $|\mathcal{D}_t|$ per round for all approaches; Right: Data relevance R_t, R_t^* and $\bar{R}_t^*$ per round. MCPD dataset (charges as output) with drifted query distribution.

Figure 5.9 shows the results for the MCPD dataset (charges as output); similar patterns were observed with other datasets. The model's behavior follows similar trends to the non-drifted scenario (Figure 5.4), but with notable differences: convergence error is approximately 7% higher, and the proportion of runs where Model A outperforms Model GT (the area of the shaded region below ϵ^*) is smaller.

Interestingly, the relevance metrics in Figure 5.9 (right) show that R_t^* and $\bar{R}_t^*$ values under drifted queries are actually slightly higher than those in the non-drifted scenario, with $R_t^* \approx 0.8$ and a smaller gap between R_t^* and $\bar{R}_t^*$. This suggests that the drifted queries help identify relevant regions more efficiently, possibly because they better explore the boundaries of these regions. Despite the drift, Model A consistently outperforms all comparison approaches.

Data Distribution Drift

Next, we evaluated performance when the underlying data distribution changes while the query distribution remains stable.

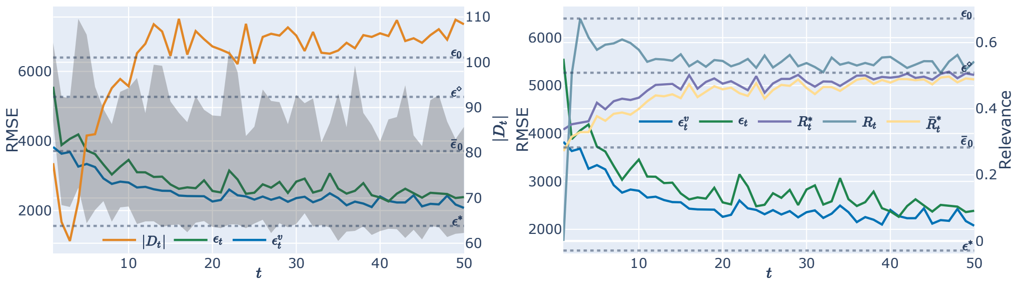


Figure 5.10: Left: RMSE and $|\mathcal{D}_t|$ per round for all approaches; Right: Data relevance R_t, R_t^* and $\bar{R}_t^*$ per round. MCPD dataset (charges as output) under data updates.

The impact of data distribution drift (Figure 5.10) is more substantial than query drift: errors are noticeably higher, the uncertainty range (shaded area) is wider, and both R_t^* and $\bar{R}_t^*$ metrics are significantly lower. This indicates that Model A is more sensitive to changes in the underlying data distribution, which is understandable as such changes directly affect the relationship between inputs and outputs, potentially invalidating previously learned patterns.

Nevertheless, Model A still outperforms Models B, E0, and E1, and the lowest regions of the uncertainty band suggest that, in favorable cases, it can still achieve a performance comparable to or better than Model GT. This resilience stems from the model's ability to progressively adapt to changing data characteristics through its refinement process.

Combined Data and Query Distribution Drifts

Finally, we assessed performance under simultaneous drifts in both query and data distributions. We ran the experiment for 60 rounds, allowing Model A to converge by round 30, then gradually introduced drifts from rounds 31 to 60. The drift proportion increased linearly as $\frac{t-30}{30}$ for $t \in [31, 60]$, with the remainder $1 - \frac{t-30}{30}$ comprising normal samples. By round 60, all queries and training data were drawn from the new distributions.

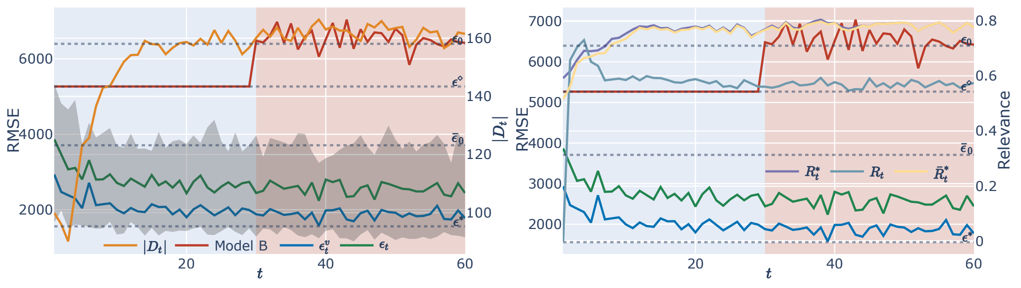


Figure 5.11: Left: RMSE and $|\mathcal{D}_t|$ per round for all approaches; Right: Data relevance R_t, R_t^* and $\bar{R}_t^*$ per round. MCPD dataset (charges as output) under gradual drifted query distribution.

Figure 5.11 shows the results for gradual query distribution drift. Model A demonstrates remarkable resilience—performance metrics remain stable even as the query distribution gradually shifts. This robust behavior stems from our update mechanism (Section 5.6.2), which optimally detects when to trigger model retraining. The average optimal stopping time was $\mathbb{E}[k^*] = 6.2$ rounds after convergence (round 36.2 from the start), precisely when the evidence of the accumulated indicators triggered the criterion in Equation (5.28).

In contrast, Model B showed significant performance deterioration even with a small proportion of drifted queries. This vulnerability arises because even a few drifted queries lead to inaccurate centroid formation, causing the inclusion of many irrelevant samples in the training set.

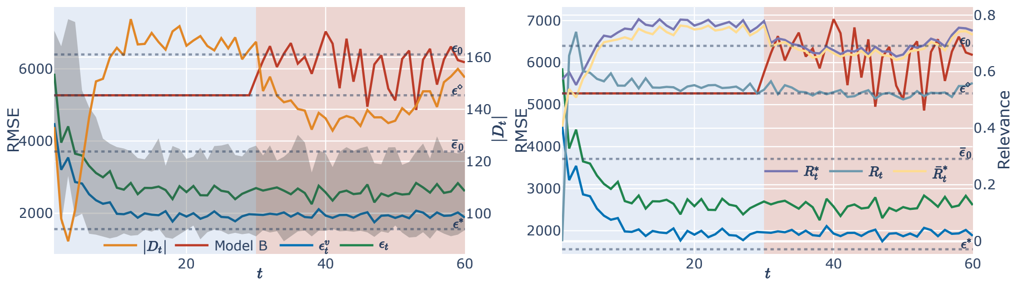


Figure 5.12: Left: RMSE and $|\mathcal{D}_t|$ per round for all approaches; Right: Data relevance R_t, R_t^* and $\bar{R}_t^*$ per round. MCPD dataset (charges as output) under gradual data updates.

Figure 5.12 shows the results for gradual data distribution drift. Here, Model A demonstrates less resilience than with query drift - both R_t^* and $\bar{R}_t^*$ begin dropping immediately as data updates are introduced, and errors show a slight upward trend. However, the model's adaptive capability becomes evident around round 45, when R_t^* and $\bar{R}_t^*$ begin to rise despite increasing data drift. This adaptation occurs when the optimal stopping criterion in Section 5.6.1 triggers model refinement.

Interestingly, Model B handles data updates better than query drifts, showing only moderately higher error under data changes. This suggests that its distance-based approach for identifying relevant data regions has some natural robustness to data distribution shifts.

Across all drift scenarios, Model A consistently outperformed all comparison approaches, demonstrating the effectiveness of our query-driven, data-centric learning approach even in dynamic environments with evolving data and query distributions.

Table 5.3: RMSE Results for All Models Under Different Drift Scenarios

Drift Scenario	Model A	Model B	Model E0	Model E1	Model GT
No Drift	2.34 ± 0.21	5.18 ± 0.37	6.39 ± 0.24	4.85 ± 0.19	1.55 ± 0.16
Query Drift	2.51 ± 0.24	6.72 ± 0.53	6.82 ± 0.32	5.13 ± 0.27	1.88 ± 0.19
Data Drift	2.98 ± 0.31	5.86 ± 0.44	7.15 ± 0.38	5.37 ± 0.29	2.14 ± 0.22
Combined Drift	3.23 ± 0.36	7.24 ± 0.61	7.43 ± 0.42	5.65 ± 0.33	2.27 ± 0.25

Table 5.3 summarizes the performance of all models under different drift scenarios. While performance degrades somewhat under drift conditions, Model A consistently maintains superior performance compared to all baseline and comparison approaches. The smallest degradation occurs under query drift (7.3% increase in RMSE), while combined drift presents the most significant challenge (38.0% increase in RMSE). Even in this most challenging scenario, Model A still outperforms Model B by 55.4% and Model E0 by 56.5%, demonstrating the robustness of our approach.

5.7.7 Discussion of Results

Our comprehensive experimental evaluation demonstrates several key strengths of the proposed approach:

Superior Predictive Performance Model A consistently outperforms both traditional approaches (Models E0 and E1) and the state-of-the-art query-driven approach (Model B) across all datasets and scenarios. In many cases, its performance approaches that of the idealized ground truth model (Model GT), which has perfect knowledge of relevant data.

Efficient Relevant Data Identification The approach successfully identifies relevant data regions with high precision and recall, as evidenced by the R_t^* and $\bar{R}_t^*$ metrics. The dynamic and progressive nature of our refinement process allows it to thoroughly explore the data space and focus computational resources on the most relevant regions.

Adaptability to Distribution Changes Both the data update and the query update mechanisms demonstrate effective adaptation to distribution drifts. The optimal stopping criteria derived from our theoretical analysis provide principled ways to determine when model retraining should be triggered, balancing stability and adaptability.

Robustness to Dataset Characteristics The approach performs well across datasets with different characteristics, from relatively well-structured CCPP data to the more challenging BMI prediction task with its fragmented relevant regions. This robustness stems from the data-centric learning mechanisms that adaptively guide the model refinement process.

Theoretical Guarantees Validated Our experimental results confirm the validity of the theoretical guarantees developed in Sections 5.5.1 and 5.5.3. The optimal round horizon predicted by our theory ($T^* \approx 30$) aligns with empirical observations of convergence, and the update mechanisms trigger at the theoretically optimal points.

These results collectively validate our approach as an effective solution for query-driven predictive analytics in environments with diverse data and query distributions, capable of maintaining high performance even as these distributions evolve over time.

5.8 Discussion

5.8.1 Key Contributions

This chapter introduced a novel approach to query-driven predictive analytics that addresses several fundamental challenges in modern data management systems. Our main contributions can be summarized as follows:

1. **Query-Driven Relevant Data Identification**: We developed a progressive learning mechanism that leverages query patterns to identify and select relevant data for model training, resulting in more accurate and efficient predictive models.
2. **Data-Centric Learning Framework**: Our approach focuses on data quality and relevance rather than model architecture, with mechanisms that balance exploration of new regions with exploitation of known relevant areas.
3. **Optimal Learning Horizon Determination**: We applied Optimal Stopping Theory to determine when to conclude the model refinement process, maximizing performance while minimizing computational cost.
4. **Adaptive Update Mechanisms**: We created theoretically grounded update mechanisms that detect and respond to changes in both data and query distributions, maintaining model accuracy in dynamic environments.
5. **Comprehensive Theoretical Analysis**: We provided error bounds, complexity analysis, and optimality guarantees for our approach, establishing a solid mathematical foundation.

5.8.2 Relationship to Previous Chapters

The query-driven predictive analytics framework presented in this chapter builds upon and complements the topics discussed in previous chapters of this thesis. The key relationships include:

- **Distributed Edge Learning**: While Chapter 3 focused on resilient edge predictive analytics through enhanced local models, this chapter explores how individual edge nodes can optimize their performance by identifying relevant data based on query patterns. Both approaches aim to improve efficiency and effectiveness in edge environments, but from complementary angles.
- **Concept Drift Handling**: Chapter 4 addressed the maintenance of model resilience under concept drift in distributed environments. Our query and data update mechanisms in this chapter provide another approach to dealing with distribution shifts, focusing specifically on the relationship between data relevance and query patterns.

- **Edge Intelligence**: The data-centric learning approach presented here aligns with the edge intelligence concepts introduced in Chapter 2, particularly the notion of 'AI on edge' where edge computing serves as a utility for efficient AI deployment. By focusing only on relevant data, our approach reduces computational and storage requirements, making it particularly suitable for resource-constrained edge environments.

5.8.3 Practical Implications

The approach presented in this chapter has several important practical implications for distributed edge computing systems:

- **Resource Efficiency**: By focusing computational resources on data regions that matter most to analytics queries, our approach can significantly reduce processing, storage, and energy requirements - critical considerations in edge environments.
- **Personalized Analytics**: The framework enables more personalized analytics experiences by tailoring models to specific query patterns, which may vary across users, applications, or edge nodes.
- **Dynamic Adaptation**: The update mechanisms allow systems to adapt to changing conditions without manual intervention, an essential capability for autonomous edge deployments in evolving environments.
- **Improved Responsiveness**: Models tailored to relevant data can provide more accurate responses with lower latency, enhancing the user experience for time-sensitive edge applications.
- **Privacy Enhancement**: By focusing only on relevant data, the approach potentially reduces privacy risks associated with processing and storing unnecessary sensitive information.

5.8.4 Limitations and Future Work

While our approach demonstrates significant advantages over traditional methods, several limitations and opportunities for future work remain:

- **Multi-node Coordination**: Our current framework focuses on the perspective of individual nodes. Future work could explore coordinated relevant data identification across multiple nodes in a distributed system, potentially incorporating federated learning principles.
- **Complex Query Types**: The current framework primarily addresses predictive queries. Extension to more complex query types such as aggregation, ranking, and recommendation queries would broaden the approach's applicability.

- **Online Learning Enhancements**: While our approach operates in an online fashion, incorporating more sophisticated online learning techniques that can handle highly non-stationary environments could further improve adaptability.
- **Privacy-Preserving Techniques**: Integrating differential privacy or other privacy-preserving techniques into the relevance identification process would enhance the framework's suitability for applications involving sensitive data.
- **Theoretical Extensions**: Future work could extend our theoretical analysis to provide tighter bounds under specific conditions or to characterize the trade-offs between exploration and exploitation more precisely.

5.9 Conclusions

In this chapter, we introduced a novel query-driven predictive analytics framework that addresses the challenge of identifying and utilizing relevant data in distributed edge environments. Our approach integrates principles from Data-Centric Learning and Query-Driven Learning to create a system that progressively refines models based on query patterns, focusing computational resources on the data regions that matter most for analytics tasks.

The key innovations of our framework include a progressive relevance identification mechanism, a data-centric learning approach that balances exploration and exploitation, optimal stopping criteria for determining when to conclude refinement, and adaptive update mechanisms for handling changes in both data and query distributions. These components work together to create a system that significantly outperforms both traditional approaches and state-of-the-art query-driven methods across diverse datasets and scenarios.

Through comprehensive theoretical analysis and extensive experimental evaluation, we demonstrated that our approach consistently achieves lower prediction errors, more efficient resource utilization, and greater robustness to distribution shifts compared to existing methods. The framework's ability to maintain high performance even in dynamic environments makes it particularly well-suited for edge computing scenarios where both data characteristics and query patterns may evolve over time.

As distributed edge systems continue to proliferate and generate ever-increasing volumes of data, approaches that selectively focus on relevant data based on evolving analytics needs will become increasingly important. The query-driven predictive analytics framework presented in this chapter provides a theoretically sound, empirically validated foundation for developing such systems, contributing to the broader goal of efficient, adaptive, and resilient edge intelligence.

Chapter 6

Dynamic Aggregation & Decentralized Personalized Federated Learning

Statement of Contribution

This chapter is based on research conducted in collaboration with Dr. Qianyu Long. The work was published as:

Long, Q., **Wang, Q.**, Anagnostopoulos, C., & Bi, D. (2025). "Decentralized Personalized Federated Learning Based on a Conditional "Sparse-to-Sparser" Scheme". *IEEE Transactions on Neural Networks and Learning Systems*, pages 1-15. DOI: 10.1109/TNNLS.2025.3580277

My primary contributions to this work, as the second author, were to implement part of the core algorithms and baselines, design and conduct the cost experiments, draw the system figure, generate the results (figures and tables), and write the manuscript for sections including experiments results and cost analysis.

The project was led by the first author, Dr. Qianyu Long, who conceived the main research idea, led the code development, and wrote the final manuscript. Dr. Christos Anagnostopoulos and Dr. Daning Bi provided supervision and contributed to the theoretical development and review of the manuscript.

6.1 Introduction

Decentralized Federated Learning (DFL) has gained popularity due to its robustness and elimination of centralized coordination requirements. In this paradigm, clients actively participate in training by exchanging models with neighboring nodes in their network. However, DFL introduces significant overhead in both training and communication costs. While existing methods focus primarily on reducing communication costs, they often overlook training efficiency and the challenges of data heterogeneity.

As discussed in Chapter 1 and Chapter 2, federated learning frameworks often struggle to balance communication efficiency, training efficiency, and personalization. These factors are inherently interconnected in Decentralized Personalized FL (DPFL). Reducing communication cost alone without considering data heterogeneity leads to degraded model performance on clients. Conversely, enhancing personalization without efficiency considerations can result in excessive computation overhead, making deployment infeasible for resource-constrained clients. Existing DPFL methods often fail to balance these aspects effectively, leading to either inefficient communication or poor model adaptability.

In this chapter, we introduce DA-DPFL, a novel *sparse-to-sparser* training scheme that initializes with a subset of model parameters which progressively decrease during training through dynamic aggregation. This approach substantially reduces energy consumption while preserving adequate information during critical learning periods. DA-DPFL incorporates two main elements: a fair dynamic scheduling for aggregation of personalized models and a dynamic pruning policy. The innovative scheduling policy allows clients in DA-DPFL to *reuse* trained models *within* the same communication round, which significantly accelerates convergence. Moreover, DA-DPFL involves optimized pruning timing to conduct further pruning, which does not violate clients' computing capacities while achieving communication, training, and inference efficiency.

Our experimental results demonstrate that DA-DPFL significantly outperforms DFL baselines in test accuracy while achieving up to 5× reduction in energy costs. We provide theoretical convergence analysis that validates the applicability of our approach in decentralized and personalized learning contexts.

6.1.1 Motivating Example

In traditional decentralized federated learning (DFL), all clients begin each round simultaneously in a fully synchronized manner. Consider a network of $K = 100$ clients where each client k communicates with $M = 10$ neighbors.

Traditional Parallel DFL (e.g., DisPFL (Dai et al., 2022)): At round t, all clients operate in lockstep:

- $t = 0$s: All 100 clients simultaneously receive neighbors' models from round $t - 1$
- $t = 0$s: All clients aggregate these *stale* models and begin local training in parallel
- $t = 30$s: All clients finish training simultaneously and broadcast their updated models

- Round $t+1$ begins: Models trained in round t are now used for aggregation

This fully parallel approach maximizes computational efficiency (minimal wall-clock time per round) but suffers from knowledge staleness: every client aggregates models that were trained in the *previous* round, not the current one.

The Proposed DA-DPFL Approach: Introduce strategic waiting to enable within-round knowledge reuse. In the same round t:

- $t = 0$s: Clients $\{1, 2, 4\}$ (with no prior neighbors, $\mathcal{N}^t_{(a)} = \emptyset$) immediately aggregate models from round $t-1$ and begin training
- $t = 30$s: Clients $\{1, 2, 4\}$ finish training with fresh models $\boldsymbol{\omega}_1^{t+1}, \boldsymbol{\omega}_2^{t+1}, \boldsymbol{\omega}_4^{t+1}$
- $t = 30$s: Client 5 (which has been waiting) now aggregates these *fresh models from round t* (not stale models from round $t-1$):

$$\tilde{\boldsymbol{\omega}}_5^t = \frac{\boldsymbol{\omega}_1^{t+1} + \boldsymbol{\omega}_2^{t+1} + \boldsymbol{\omega}_4^{t+1} + \text{others}}{M+1} \odot \mathbf{m}_5^t$$

- $t = 30$s–$t = 60$s: Client 5 trains using these fresher aggregated parameters

The Trade-off:

- **Cost:** Increased wall-clock time per round (some clients wait, introducing controlled latency)
- **Benefit:** Within-round knowledge transfer so that clients aggregate models that were trained in the *current* round rather than the previous round, enabling faster knowledge propagation

But waiting alone is insufficient: Even with within-round knowledge reuse, two problems remain:

1. **Dense aggregation:** Aggregating all 11.7M parameters from all neighbors regardless of relevance wastes communication ($117M$ parameters per client) and dilutes personalization
2. **Idle waiting time:** While client 5 waits from $t = 0$ to $t = 30$s for its prior neighbors, this computational resource remains unused

The Complete Solution: DA-DPFL addresses both challenges:

1. **Dynamic scheduling** (controlled by parameter N): Enables within-round model reuse by having clients wait for up to N fastest prior neighbors, trading wall-clock time for fresher knowledge
2. **Sparse-to-sparser training**: During the waiting period, use the client's own gradient information $\nabla F_k(\boldsymbol{\omega}_k^t)$ to compute dynamic pruning masks that determine which parameters to aggregate (5–10% sparsity), achieving:

- 90%+ reduction in communication cost
- Better personalization through selective parameter aggregation
- Productive use of waiting time for mask computation

6.2 Related Work

In distributed machine learning, communication and training costs present significant challenges. Various approaches have been proposed to address these issues, building upon the foundations discussed in Chapter 2.

6.2.1 Compression Techniques in Federated Learning

As outlined in Section 2.4.1, methods like LAQ (Sun et al., 2019) and DGC (Lin et al., 2018) reduce communication costs through gradient quantization and deep gradient compression, respectively. FedSL (Zhang et al., 2024) reduces network traffic by sending only parts of model parameters using layer similarity.

Pruning, as an important model compression technique, helps alleviate device storage constraints and reduce communication costs, as demonstrated by PruneFL (Jiang et al., 2023), FedDST (Bibikar et al., 2022), and FedDIP (Long et al., 2023). pFedGate (Chen et al., 2023) addresses these challenges by learning sparse local models adaptively with a trainable gating layer, enhancing model capacity and efficiency. FedPM (Isik et al., 2023) and FedMask (Li et al., 2021a) focus on efficient model communication using probability masks, with FedMask providing personalized, sparse DNNs for clients.

6.2.2 Personalized Federated Learning

Building upon the PFL approaches introduced in Section 2.2.1, methods such as FedMask (Li et al., 2021a), FedSpa (Huang et al., 2022), and DisPFL (Dai et al., 2022) use personalized masks to address data heterogeneity. Ditto (Li et al., 2021b) offers a fair personalization framework through global-regularized multitask FL, while FOMO (Zhang et al., 2021) focuses on first-order optimization for personalized learning. FedABC (Wang et al., 2023) employs a 'one-vs-all' strategy and binary classification loss for class imbalance, while FedSLR (Huang et al., 2023) integrates low-rank global knowledge for efficient downloading during communication.

6.2.3 Decentralized Federated Learning

Extending the DFL approaches discussed in Section 2.3.3, several methods have emerged that address both communication efficiency and model performance. DFedAvgM (Sun et al., 2022) extends FedAvg to decentralized contexts with momentum SGD. BEER (Zhao et al., 2022) enhances convergence through communication compression and gradient tracking. GossipFL (Tang et al., 2022b) uses bandwidth information to create a gossip matrix allowing communication with one peer using sparsified gradients, reducing communication. DFedSAM (Shi et al., 2023) considers utilizing the Sharpness-Aware-Minimization optimizer, while DisPFL (Dai et al., 2022) utilizes RigL-like pruning in decentralized sparse training to lower generalization error and communication costs.

6.3 System Model and Problem Formulation

6.3.1 Network Topology

We consider a distributed system with K clients indexed by $\mathcal{K} = \{1, 2, \ldots, K\}$. Clients are networked with a topology represented by a graph $\mathcal{G}(\mathcal{K}, \mathbf{V})$, where the adjacency matrix $\mathbf{V} = [v_{i,j}] \in \mathbb{R}^{K \times K}$ defines the neighborhood $\mathcal{G}_k$ of client $k \in \mathcal{K}$, i.e., subset of clients that directly communicate with client k, $\mathcal{G}_k = \{i \in \mathcal{K} : v_{i,k} > 0\}$.

An entry $v_{i,k} = 0$ indicates that there is no communication from client i to client k, i.e., $i \notin \mathcal{N}_k$. Note that $v_{i,k} = v_{k,i}$ may not always be valid for $i \neq k$. The topology can be static or dynamic. In our case, we adopt dynamic communication among clients, i.e., entries in $\mathbf{V}^t$ depend on (discrete) time instance $t \in \mathbb{T} = \{1, 2, \ldots\}$. We define a time-varying and non-symmetric network topology via $\mathbf{V}^t = [v^t_{i,j}] \in \mathbb{R}^{K \times K}$ accommodating temporal neighborhood $\mathcal{G}^t_k$ for k-th client.

6.3.2 Problem Formulation

We consider a scalable DFL setting with K clients with a time-varying topology. Each client $k \in \mathcal{K}$ possesses local data $\mathcal{D}_k = \{(\mathbf{x}, y)\}$ of input-output pairs $\mathbf{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$, and communicates with neighbors $\mathcal{G}^t_k$ exchanging models.

Building on the personalized federated learning formulation from Section 2.2.1, the problem seeks to find the models $\omega_k, \forall k \in \mathcal{K}$, that minimize:

$$\min_{\{\omega_k\}, k \in [K]} f(\{\omega_k\}_{k=1}^K) = \frac{1}{K} \sum_{k=1}^{K} F_k(\omega_k), \tag{6.1}$$

where $F_k(\omega_k) = \mathbb{E}[\mathcal{L}(\omega_k; (\mathbf{x}, y)) | (\mathbf{x}, y) \in \mathcal{D}_k]$ with expected loss function $\mathcal{L}(.;.)$ between the actual and predicted output given local data $\mathcal{D}_k$.

We extend this formulation by adopting model pruning in DFL, aiming to eliminate non-essential model weights through a binary mask $\mathbf{m}$ over a model. The pruning-based (masked) PFL problem is formulated as finding the global model ω and individual masks $\mathbf{m}_k, \forall k \in \mathcal{K}$:

$$\min_{\omega,\{\mathbf{m}_k\},k\in[K]} f(\{\omega_k\}_{k=1}^{K}) = \frac{1}{K}\sum_{k=1}^{K} F_k(\omega \odot \mathbf{m}_k), \tag{6.2}$$

where $F_k(\omega \odot \mathbf{m}_k) = \mathbb{E}[\mathcal{L}(\omega \odot \mathbf{m}_k; (\mathbf{x}, y))|(\mathbf{x}, y) \in \mathcal{D}_k]$; $\odot$ represents the Hadamard product (element-wise product) of two matrices. The individual mask $\mathbf{m}_k$ denotes a pruning operator specific to client k. Given mask $\mathbf{m}_k$, the sparsity $s_k \in [0, 1]$ of $\mathbf{m}_k$ indicates the proportion of zero model weights among all weights.

The goal of Equation 6.2 is to seek a global model $\boldsymbol{\omega}$ and individual masks $\mathbf{m}_k$ such that the optimized personalized model for *each* client $k \in \mathcal{K}$ is given by $\boldsymbol{\omega}_k = \boldsymbol{\omega} \odot \mathbf{m}_k$, while clients communicate at time t only with their neighbors $\mathcal{G}_k^t$ given the time-varying $\mathbf{V}^t$.

6.4 The DA-DPFL Framework

6.4.1 Overview

We introduce the DA-DPFL framework to tackle the problem in Equation 6.2. DA-DPFL not only addresses data heterogeneity efficiently via masked-based PFL but also significantly improves convergence speed by incorporating a fair dynamic communication protocol. Sequential pointing line communication adopts a one-peer-to-neighbors mechanism, striking a balance between computational parallelism and delay.

To overcome limitations such as data heterogeneity and non-scalability, DA-DPFL introduces two key components: **dynamic model aggregation** and **adaptive pruning**. Dynamic model aggregation enables efficient knowledge sharing among clients by leveraging past model updates within the same communication round, reducing the number of communication rounds required for convergence. Adaptive pruning refines model sparsity dynamically, reducing both communication and computational overhead while preserving model accuracy.

Figure 6.1 illustrates the complete DA-DPFL framework, comprising two key components: the dynamic scheduling mechanism (top) and the sparse-to-sparser training process (bottom).

Network Topology and Dynamic Scheduling (Top): The figure depicts a decentralized network with $K = 6$ clients, where each client maintains $M = 2$ neighbors. The dynamic scheduling is controlled by parameter N, which determines the maximum number of prior neighbors a client waits for before beginning its local training.

At the start of each communication round t, clients are assigned random reuse indexes $\mathcal{N}_k^t$, which are then partitioned into two subsets based on the ordering:

- **Prior neighbor set** $\mathcal{N}_{(a)k}^{(*)t}$: Neighbors with reuse indexes less than or equal to k, representing clients that complete training before or simultaneously with client k
- **Posterior neighbor set** $\mathcal{N}_{(b)k}^{(*)t}$: Neighbors with reuse indexes greater than k, representing clients that complete training after client k

The scheduling policy operates as follows:

- When $N = 0$: All clients train in parallel without waiting ($\mathcal{N}_{(a)k}^{(*)t} = \emptyset$ for all k), reducing to traditional parallel DFL
- When $N = 1$: Each client waits for at most 1 fastest prior neighbor. In the example, client 3 waits for client 2, while clients 5 and 6 wait for client 1. Clients 1, 2, and 4 have no prior neighbors and begin training immediately
- When $N = 2$: The waiting capacity increases. Client 6 now waits for both clients 1 and 5 (marked in different colors), enabling deeper hierarchical knowledge transfer within the same round

This hierarchical structure enables within-round model reuse: clients with smaller reuse indexes train first and broadcast their updated models, which are then immediately aggregated by clients with larger indexes *within the same communication round*, rather than waiting until the next round as in traditional parallel DFL.

Training Process Flow (Bottom): For each client k at round t, the training process follows a conditional sparse-to-sparser scheme with five key steps:

Step 1 - Detection Score Calculation: Using the current model $\boldsymbol{\omega}_k^t$, the client computes a detection score based on the relative change in model weights:

$$\text{Detection Score} = \frac{|\Delta_0^t - \Delta_0^{t-1}|}{|\Delta_1^0|}, \quad \text{where } \Delta_0^t = \|\boldsymbol{\omega}^t - \boldsymbol{\omega}^0\|_2$$

When this score falls below threshold δ_{pr}, the client votes that the model has entered a stable learning phase suitable for further pruning. The first pruning time t^* is determined collectively when a majority of clients vote positively (voting threshold δ_v).

Decision Point: At each round, the client checks two conditions:

- Is the current round a pruning round ($t \in \mathcal{T}$)?
- Has the current sparsity not yet reached the target ($s_k < s^*$)?

If **both** conditions are met (“yes” path), the client proceeds to further pruning (Steps 2-5). Otherwise (“no” path), the client performs standard sparse training with the current mask $\mathbf{m}_k^t$ using the RigL algorithm (Evci et al., 2020) for mask updates.

Step 2 - Magnitude-based Pruning: The client computes the PQ Index (PQI) to assess model compressibility, then prunes a fraction c_l^t of parameters with the smallest magnitudes in each layer l:

$$c_l^t = \left\lfloor d_l^t \cdot \min\left(\gamma\left(1 - \frac{r_l^t}{d_l^t}\right), \beta\right)\right\rfloor$$

where d_l^t is the current number of parameters, r_l^t is the minimum required parameters determined by PQI, and γ, β are scaling factors.

Step 3 - Gradient-based Regrowth: To maintain model capacity, the client regrowing parameters are selected based on gradient magnitudes $\nabla F_k(\tilde{\boldsymbol{\omega}}_k^t)$, prioritizing parameters with large gradient flow (Evci et al., 2020). This ensures that pruning does not eliminate parameters critical for learning.

Step 4 - PQI Evaluation: The client evaluates neural network compressibility using the PQ Index (Diao et al., 2023) for each layer:

$$I(\tilde{\boldsymbol{\omega}}_k^{l,t}) = 1 - \left(\frac{1}{d_l^t}\right)^{\frac{1}{q} - \frac{1}{p}} \frac{\|\tilde{\boldsymbol{\omega}}_k^{l,t}\|_p}{\|\tilde{\boldsymbol{\omega}}_k^{l,t}\|_q}$$

where $0 < p \leq 1 < q$ are norm parameters. Higher PQI values indicate greater compressibility, suggesting that further pruning is feasible without significant performance degradation.

Step 5 - Conditional Further Pruning: Based on the PQI evaluation, if the model demonstrates high compressibility, additional pruning is applied to achieve the sparse-to-sparser transition. The pruning frequency is controlled by the schedule $\mathcal{T} = \{t_1, t_2, \ldots\}$ with increasing gaps: $I_\tau = \lceil \frac{t^* + b}{c^{\tau - 1}} \rceil$, where b delays the first pruning and c controls the frequency scaling.

Integration of Components: The key innovation lies in utilizing the waiting period introduced by the dynamic scheduling (top) to perform the computationally intensive tasks in the training process (bottom), particularly the detection score calculation, PQI evaluation, and mask computation. While a client waits for its prior neighbors $\mathcal{N}_{(a)k}^{(*)t}$ to complete training, it productively uses this time to:

1. Analyze its gradient information $\nabla F_k(\boldsymbol{\omega}_k^t)$ from the just-completed training
2. Compute dynamic pruning masks based on parameter importance
3. Prepare for selective aggregation of incoming neighbor models

This transforms the waiting period from idle time into productive computation, enabling high sparsity ($> 90\%$) in both training and communication while improving model accuracy through fresher within-round knowledge transfer.

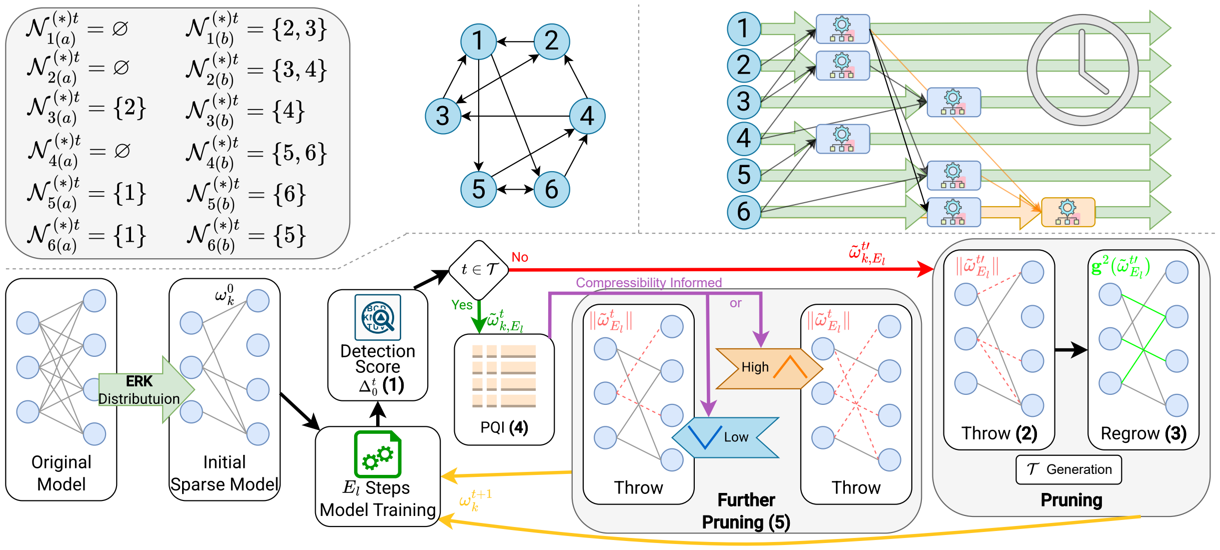

$\mathcal{N}_{1(a)}^{(*)t} = \varnothing$
$\mathcal{N}_{1(b)}^{(*)t} = \{2,3\}$
$\mathcal{N}_{2(a)}^{(*)t} = \varnothing$
$\mathcal{N}_{2(b)}^{(*)t} = \{3,4\}$
$\mathcal{N}_{3(a)}^{(*)t} = \{2\}$
$\mathcal{N}_{3(b)}^{(*)t} = \{4\}$
$\mathcal{N}_{4(a)}^{(*)t} = \varnothing$
$\mathcal{N}_{4(b)}^{(*)t} = \{5,6\}$
$\mathcal{N}_{5(a)}^{(*)t} = \{1\}$
$\mathcal{N}_{5(b)}^{(*)t} = \{6\}$
$\mathcal{N}_{6(a)}^{(*)t} = \{1\}$
$\mathcal{N}_{6(b)}^{(*)t} = \{5\}$
Original Model
ERK Distributuion
Initial Sparse Model
ω_k^0
Detection Score Δ_0^t (1)
E_l Steps Model Training
$t \in \mathcal{T}$
No
Yes
$\tilde{\omega}_{k,E_l}^t$
$\tilde{\omega}_{k,E_l}^{t\prime}$
PQI (4)
Compressibility Informed
or
High
Low
Throw
Throw
Further Pruning (5)
Throw (2)
Regrow (3)
$\mathcal{T}$ Generation
Pruning
ω_k^{t+1}

Figure 6.1: (Top) Client network ($K = 6, M = 2, N = 1$) with *reuse indexes* $\mathcal{N}_{k(a)}^{(*)t}$ and $\mathcal{N}_{k(b)}^{(*)t}$. Learning schedule: while $N = 0$, all nodes train in parallel, i.e., $\mathcal{N}_{k(a)}^{(*)t} = \emptyset$; if $N = 1$, node 3 waits for 2, node 5 and 6 for 1; nodes 1, 2, 4 begin parallel training immediately; $N = 2$ enables node 6 wait for 1 and 5, marked with *different* color. **(Bottom)** Training process at time t for client k. Flow follows $t \in \mathcal{T}$: 'no' leads to normal *sparse* training, 'yes' to proposed *sparser* training. Steps: (1) Detection score calculation using ω_k^t, determining t^*; (2) Magnitude-based weight pruning; (3) Gradient-flow-driven weight recovery; (4) PQI evaluation for NN compressibility; (5) Additional pruning based on compressibility level.

Dynamic Model Aggregation

Instead of waiting for all clients to complete training before aggregation, DA-DPFL allows clients to reuse partially trained models from their neighbors within the same round. This reduces the convergence time and enhances personalization by dynamically adapting to the local data distribution. The approach ensures that clients benefit from previously learned knowledge without excessive communication overhead.

Adaptive Pruning

The decision on when to apply pruning during training is crucial. While *early* pruning reduces computational cost, it can negatively affect performance. Choosing an optimal pruning time can enhance both training and communication efficiency, often with minimal performance degradation or even improvements. As elaborated in previous research (Dai et al., 2022; Hoefler et al., 2021), a sparser model tends to have a reduced generalization bound characterized by a smaller discrepancy between training and test errors.

In order to find sparser models with lower generalization error, DA-DPFL introduces an innovative dynamic pruning strategy that adjusts sparsity levels based on client-specific constraints. This ensures an optimal trade-off between efficiency and model quality, reducing redundant parameters without significantly impacting accuracy.

DA-DPFL, illustrated in Figure 6.1 and Algorithm 5, addresses data heterogeneity while decreasing the number of rounds needed for convergence and achieving high model performance. This comes with a relatively small and controllable delay in learning from trained models.

Algorithm 5 The DA-DPFL Algorithm

```
1: Input: $K$ clients; $T, E_l$ rounds; PQI hyper-param. $\{p, q, \gamma, \eta_c\}$, pruning thr $\delta_{pr}$;
   voting threshold $\delta_v$; factors $b, c$; target sparsity $s^*$
2: Output: Personalized aggregated models $\{\tilde{\boldsymbol{\omega}}_k^T\}_{k=1}^K$
3: Initialization: Initialize $\{\mathbf{m}_k^0\}_{k=1}^K$, $\{\boldsymbol{\omega}_k^0\}_{k=1}^K$, $\mathcal{T} \leftarrow \emptyset$
4: for round $t = 1$ to $T$ do
5:     for each client $k$ do
6:         Generate a random reuse index set $\{\mathcal{N}_k^t\}_{k=1}^K$
7:         Generate a random bijection $\pi_k^t$ between $\mathcal{N}_k^t$ and $\mathcal{G}_k^t$
8:         Form prior and posterior set $\{\mathcal{N}_{k(a)}^{(*)t}, \mathcal{N}_{k(b)}^{(*)t}\}$
9:         Form $\{\mathcal{G}_{k(a)}^{(*)t}, \mathcal{G}_{k(b)}^{(*)t}\}$ by $\{\mathcal{N}_{k(a)}^{(*)t}, \mathcal{N}_{k(b)}^{(*)t}, \pi_k^{-1(t)}\}$
10:        if $\mathcal{G}_{(a)k}^{(*)t} \neq \emptyset$ then
11:            Wait for models from neighbors $\mathcal{G}_{(a)k}^{(*)t}$
12:        end if
13:        Receive neighbor's models $\boldsymbol{\omega}_j^t, j \in \mathcal{G}_k^t$
14:        Obtain mask-based aggregated model $\tilde{\boldsymbol{\omega}}_k^t$
15:        Compute $\tilde{\boldsymbol{\omega}}_{k,\tau}^t$ for $E_l$ local rounds
16:        Calculate $\Delta_0^t(k)$ and $v_t(k)$ based on $\delta_{pr}$
17:        Broadcast $v_t(k)$ to all clients; derive $t^*$
18:        if $t \in \mathcal{T}$ and $s_k < s^*$ then
19:            Call Algorithm 6 to obtain $\tilde{\boldsymbol{\omega}}_{k,E_l}^{t\prime}, \mathbf{m}_k^{t\prime}$
20:            Update sparsity $s_k$
21:        else
22:            Set $(\tilde{\boldsymbol{\omega}}_{k,E_l}^{t\prime}, \mathbf{m}_k^{t\prime}) \leftarrow (\tilde{\boldsymbol{\omega}}_{k,E_l}^{t}, \mathbf{m}_k^{t})$
23:        end if
24:        Call Algorithm 7 to update $\mathbf{m}_k^{t+1}$
25:        Set $\boldsymbol{\omega}_k^{t+1} = \tilde{\boldsymbol{\omega}}_{k,E_l}^{t\prime}$
26:    end for
27:    if $t = t^*$ then
28:        Update $\mathcal{T}$
29:    end if
30: end for
```

6.4.2 Learning Scheduling Policy

In this section, we present the scheduling policy for client participation in DA-DPFL, which applies to various network topologies, including *ring* and *fully connected* structures. The neighborhood sets are defined as $\mathcal{G}_k^t, k \in \{1, 2, \ldots, K\}$. In particular, DA-DPFL is designed for time-varying connected topologies while remaining flexible enough to support static topologies, represented as $\mathcal{G}_k$.

At the start of each communication round, denoted by t, *reuse indexes* for neighborhood sets $\mathcal{N}_k^t$ are randomly assigned to a subset of clients. Let M represent the size of each client's neighborhood, i.e., the number of connected clients that can communicate with client k. We have $|\mathcal{G}_k^t| = |\mathcal{N}_k^t| = M < K$ and $\mathcal{G}_k^t \overset{\pi_k^t}{\leftrightarrow} \mathcal{N}_k^t$, where π_k^t is a random bijection mapping. Given that $\mathcal{G}_k^t$ and $\mathcal{N}_k^t$ are both discrete sets,

$$i = \pi_k^t(j), \tag{6.3}$$

with index $i \in \mathcal{N}_k^t$ and $j \in \mathcal{G}_k^t$. Note that $\mathcal{N}_k^t$ may be equal to $\mathcal{G}_k^t$ if sets are randomly generated. For simplicity, we let $\mathcal{N}_k^t = \mathcal{G}_k^t$ in Figure 6.1, where client k is indexed with reuse index k. The introduction of $\mathcal{N}_k^t$ serves to emphasize the independence in the generation of *reuse* indexes, which are pivotal in guiding the dynamic aggregation process.

Remark 6.1. *The criteria for establishing $\mathcal{G}_k^t$ are influenced by factors such as network bandwidth, geographical location, and link availability. However, $\mathcal{N}_k^t$ is independent of these factors. We reassign client indices for each training round.*

Within DA-DPFL, a client k may defer the reception of models from *some* neighbors, contingent upon $\mathcal{N}_k^t$. We denote $\mathcal{N}_k^t$ for each client k as two subsets based on the reuse indices of neighboring clients: (a) a *prior* client subset $\mathcal{N}_{k(a)}^t = \{n_k^t \leq k : n_k^t \in \mathcal{N}_k^t\}$, and (b) a *posterior* client subset $\mathcal{N}_{k(b)}^t = \{n_k^t > k : n_k^t \in \mathcal{N}_k^t\}$ (refer to Figure 6.1 Top). $\mathcal{N}_{k(b)}^t = \emptyset$ implies $\mathcal{N}_{k(a)}^t = \mathcal{N}_k^t$, which leads to client k waiting for the **slowest** client within $\mathcal{N}_k^t$ before commencing model aggregation and local training on dataset $\mathcal{D}_k$. To enhance scalability, we introduce a threshold to allow waiting for at most N fastest clients in $\mathcal{N}_{k(a)}^t$, where $|\mathcal{N}_{k(a)}^{(*)t}| = N \leq M$. Then, $\mathcal{N}_{k(a)}^{(*)t} \cup \mathcal{N}_{k(b)}^{(*)t} = \mathcal{N}_k^t$. Conversely, the absence of a prior client set ($\mathcal{N}_{k(a)}^{(*)t} = \emptyset$) enables client k to incorporate models from $\mathcal{N}_{k(b)}^{(*)t}$ *without* delay, as illustrated by nodes 1, 2, and 4 in Figure 6.1. Based on the bijection mapping between $\mathcal{N}_k^t$ and $\mathcal{G}_k^t$, $\mathcal{G}_{(a)k}^{(*)t}$ is obtained.

DA-DPFL adopts a hybrid scheme that integrates continual learning with delayed aggregation while also supporting immediate aggregation, enabling dynamic aggregation. Continual learning is achieved by progressively incorporating model updates from clients in the prior set of the client k. By allowing this sequential knowledge transfer, DA-DPFL seeks to maximize knowledge utilization in the training process. It comes at the expense of a potential delay for client k in aggregating the models from $\mathcal{G}_{k(a)}^{(*)t}$. On the other hand, the models of the clients in the posterior set are sent to client k independently without any delay, thereby achieving training parallelism.

DA-DPFL's learning schedule diverges from traditional FL paradigms. In our example, at time t, nodes $\{1, 2, 4\}$ engage in simultaneous (parallel) training, while nodes $\{3, 5, 6\}$ await model reuse from preceding clients. This methodology allows subsequent nodes to train concurrently with preceding ones, as shown by nodes $\{5, 6\}$ training in tandem with node 3. The introduction of a cutoff value N endows our waiting policy with controllability. If $N = 0$, DA-DPFL operates as a parallel FL system with sparse training, whereas $N = M = K$ transforms DA-DPFL into a *sequential* FL.

6.4.3 Time-optimized Dynamic Pruning Policy

Alongside the scheduling of local training and gradual model aggregation achieved by prior and posterior neighbors per client, we introduce a dynamic pruning policy. The initial mask $\mathbf{m}_k^0, \forall k$, is set up in accordance with the Erdos-Renyi Kernel (ERK) distribution (Evci et al., 2020). Subsequently, masks are removed and re-grown based on the importance scores, which are computed from the magnitude of model weights and gradients.

The RigL algorithm (Evci et al., 2020) dynamically updates the sparse connectivity of neural networks by periodically pruning and regrowing connections based on gradient information, rather than relying on fixed sparse structures. This enables training sparse networks from scratch without requiring pre-training a dense model. Our approach extends the centralized RigL (Evci et al., 2020) to DA-DPFL, as detailed in Algorithm 7. Unlike fixed-sparsity training methods such as GraSP (Wang et al., 2020a), SNIP (Lee et al., 2019), and RigL, our method enables *further* pruning by dynamically adjusting sparsity throughout the training process.

The Sparsity-informed Adaptive Pruning (SAP) approach introduced in (Diao et al., 2023) leverages the PQ Index (PQI), which quantitatively evaluates a deep neural network's compressibility by analyzing the distribution of its singular values. The PQI provides insights into the redundancy within model parameters, guiding pruning decisions to maximize sparsity while minimizing accuracy degradation. DA-DPFL leverages PQI by integrating within DFL, which addresses the heterogeneity of various local models by adaptively pruning different models.

We adjust the pruning time detection score as:

$$\frac{|\Delta_0^t - \Delta_0^{t-1}|}{|\Delta_0^1|} < \delta_{pr}; \Delta_0^t := \|\boldsymbol{\omega}^t - \boldsymbol{\omega}^0\|^2, \tag{6.4}$$

where δ_{pr} is a predefined threshold.

In DA-DPFL with K clients, we introduce a *voting majority rule*, where the client k's vote is defined as:

$$v_t(k) = \begin{cases} 1 & \text{if } \frac{|\Delta_0^t(k) - \Delta_0^{t-1}(k)|}{|\Delta_0^1(k)|} < \delta_{pr}, \\ 0 & \text{otherwise.} \end{cases} \tag{6.5}$$

Hence, the first time to prune is determined by K clients as:

$$t^* = \min\{t : \frac{1}{K}\sum_{k=1}^{K} v_t(k) < \delta_v\} \tag{6.6}$$

where δ_v represents the ratio threshold for voting. Given the first pruning time t^*, DA-DPFL determines the pruning frequency for rounds $t > t^*$. Based on the influence of the early training phase, also known as the *critical learning* period (Jastrzebski et al., 2021) on the local curvature of the loss function in DNN, our strategy allows a low pruning frequency during the initial stages, while intensifying the pruning as the model approaches convergence. This balance between communication overhead and model performance yields an optimal pruning frequency that varies across tasks and model architectures. We define the *pruning frequency*, i.e., the gap between consecutive pruning events, and in turn the pruning times by non-evenly dividing the rest of the horizon $T - t^*$:

$$I_\tau := \lceil \frac{t^* + b}{c^{\tau-1}} \rceil, \tau \in \{\mathbb{Z}_{\geq 1}\}. \tag{6.7}$$

The parameter $b > 0$ delays the optimal first pruning time and $c > 0$ is a scaling factor that adjusts the pruning frequency. The p-th pruning time t_p with $t_p > t^*$, is $t_p = \sum_{\tau=1}^{p} I_\tau$ obtaining the pruning times set $\mathcal{T} = \{t_1, \ldots, t_p : t^* < t_p < T\}$.

6.4.4 Masked-based Model Aggregation

For notation compatibility, following the model aggregation operator in DisPFL(Dai et al., 2022) and FedDST (Bibikar et al., 2022), the client k's aggregated model $\tilde{\omega}_k^t$ derived from the models of client k's neighbors in $\mathcal{G}_k^t$ at round t based on masked local model is:

$$\tilde{\omega}_k^t = \left(\frac{\sum_{j \in \mathcal{G}_{k+}^t} \omega_j^t}{\sum_{j \in \mathcal{G}_{k+}^t} \mathbf{m}_j^t} \right) \odot \mathbf{m}_k^t, \tag{6.8}$$

where $\mathcal{G}_{k+}^t = \mathcal{G}_k^t \cup \{k\}$ is client k's neighborhood including client k. The local training rounds $\tau \in E_l$ based on the obtained $\tilde{\omega}_k^t$ are:

$$\tilde{\omega}_{k,\tau+1}^t = \tilde{\omega}_{k,\tau}^t - \eta(\mathbf{g}_{k,\tau}^t \odot \mathbf{m}_{\mathbf{k}}^{\mathbf{t}}), \tag{6.9}$$

where $\mathbf{g}_{k,\tau}^{t}$ is the gradient of the local loss function $F_k(\cdot)$ w.r.t. $\tilde{\omega}_{k,\tau}^{t}$.

6.4.5 Additional Algorithms

Here, we present the core algorithms for the Sparsity-informed Adaptive Pruning (SAP) and RigL mask generation, which are essential components of our DA-DPFL framework.

Algorithm 6 PQI-driven pruning (Layerwise)

```
1: Input: $\tilde{\omega}_{k,E_l}^{t}$, mask $\mathbf{m}_k^t$, norm index $0 < p \leq 1 < q$, compression hyper-parameter
   $\eta_c$, scaling factor $\gamma$, pruning threshold $\beta$, further pruning time $\mathcal{T}$
2: Output: $\tilde{\omega}_{k,E_l}^{t\prime}$, corresponding mask $\mathbf{m}_k^{t\prime}$
3: for $t \in \mathcal{T}$ do
4:     for each layer $l \in |L|$ do
5:         Compute dimensionality of $\tilde{\omega}_{k,E_l}^{l,t}$: $d_t^l = |\mathbf{m}_k^{l,t}|$
6:         Compute PQ Index $I(\tilde{\omega}_{k,E_l}^{l,t}) = 1 - \left(\frac{1}{d_t^l}\right)^{\frac{1}{q}-\frac{1}{p}} \frac{\|\tilde{\omega}_{k,E_l}^{l,t}\|_p}{\|\tilde{\omega}_{k,E_l}^{l,t}\|_q}$
7:         Compute the lower boundary required model parameters to keep $r_t^l = d_t^l(1 +
   \eta_c)^{-\frac{q}{q-p}} \left[1 - I(\tilde{\omega}_{k,E_l}^{l,t})\right]^{\frac{p}{q-p}}$
8:         Compute the number of model parameters to prune
9:         $c_t^l = \left\lfloor d_t^l \cdot \min\left(\gamma\left(1 - \frac{r_t^l}{d_t^l}\right), \beta\right)\right\rfloor$
10:        Prune $c_t^l$ model parameters with the smallest magnitude based on $\tilde{\omega}_{k,E_l}^{l,t}$ and
   $\mathbf{m}_k^{l,t}$
11:        Find new layer mask $\mathbf{m}_k^{l,t\prime}$ and pruned model $\tilde{\omega}_{k,E_l}^{l,t\prime}$ at layer $l$
12:    end for
13:    Obtain $\tilde{\omega}_{k,E_l}^{t\prime}$ and corresponding mask $\mathbf{m}_k^{t\prime}$
14: end for
```

6.5 Experimental Results

6.5.1 Experimental Setup

Datasets & Models

Our experiments were conducted on three widely-used datasets: HAM10000 (Tschandl et al., 2018), CIFAR10, and CIFAR100 (Krizhevsky & Hinton, 2009). We used two distinct partition methods, **Pathological** and **Dirichlet**, to generate non-i.i.d. scenarios. The Dir. partition constructs non-i.i.d. data using a Dir(α) distribution, with $\alpha = 0.3$ for CIFAR10 and CIFAR100, and $\alpha = 0.5$ for HAM10000. For Pat. partitioning, several classes n_{cls} are assigned per client: 2 for CIFAR10 and HAM10000, and 10 for CIFAR100.

Algorithm 7 RigL mask generation

```
1: Input: $\tilde{\omega}^{t\prime}_{k,E_l}$, corresponding mask $\mathbf{m}^{t\prime}_k$, global rounds T, initial annealing ratio $\alpha_0$
2: Output: New mask $\mathbf{m}^{t+1}_k$
3: Compute prune ratio $\alpha_t = \frac{\alpha}{2}\left(1 + \cos\left(\frac{t\pi}{T}\right)\right)$
4: Sample one batch of local training data to calculate dense gradient $\mathbf{g}(\tilde{\omega}^{t\prime}_{k,E_l})$
5: for each layer $l \in |L|$ do
6:     Update mask $\mathbf{m}^{l,t+\frac{1}{2}\prime}_k$ by pruning $\alpha_t$ percentage of weights based on weight magnitude
7:     Update mask $\mathbf{m}^{l,t+1}_k$ via regrowing weights with gradient information $\mathbf{g}(\tilde{\omega}^{t\prime}_{k,E_l})$
8: end for
9: Find new mask $\mathbf{m}^{t+1}_k$
```

To validate the versatility of our pruning methods across various model architectures, we selected AlexNet (Krizhevsky et al., 2012) for HAM10000, ResNet18 (He et al., 2016) for CIFAR10, and VGG11 (Simonyan & Zisserman, 2015) for CIFAR100, ensuring a comprehensive evaluation across diverse model structures.

Baselines

We compare the proposed methods with baselines including CFL: **FedAvg**(McMahan et al., 2017), **Ditto**(Li et al., 2021b) and **FedDST** (Bibikar et al., 2022), and DFL: **GossipFL**(Tang et al., 2022b), **DFedAvgM**(Sun et al., 2022), **DisPFL**(Dai et al., 2022), **BEER**(Zhao et al., 2022), **DFedSAM**(Shi et al., 2023) and **DFedPGP**(Liu et al., 2024).

System Configuration

We consider a network of $K = 100$ clients and select $M = N = 10$ clients (neighbors) per communication round. CFL focuses on communication between the central server and the selected clients. DFL mirrors this communication, allocating identical bandwidth to each of the busiest clients. This ensures that 10 clients are active per round, matching the server's connection load.

All baseline results are average values for three random seeds of the best test model performance. In contrast to CFL, all DFL baselines except DFedSAM are configured with half the communication cost. Our method, FedDST, and DisPFL implement sparse model training for efficiency, which all start with initial sparsity $s^0_k = 0.5$ with $k \in [K]$ for all clients. To ensure a balanced and fair comparison, all DFL benchmarks incorporate personalization by monitoring model performance after local training with a random time-varying connection.

Hyperparameters

Unless otherwise specified, we fix the number of local epochs at 5 for all approaches and employ a Stochastic Gradient Descent (SGD) optimizer with a weight decay set to 5×10^{-4}. The learning rate is initialized at 0.1, undergoing an exponential decay with a factor of 0.998 after each global communication round. The batch size is consistently set to 128 in all experiments. The global communication rounds are conducted 500 times for the CIFAR10 and CIFAR100 datasets, and 300 times for the HAM10000 dataset. We let $\delta_v = 0.5, b = 0, c = 1.3$, $\{p, q, \gamma, \eta_c\} = \{0.5, 1, 0.9, 1\}$ as suggested by (Diao et al., 2023), and $\delta_{pr} \in \{0.01, 0.02, 0.03\}$ for all experiments.

Cost Simulation

For cost analysis, we adopt metrics to calculate the total cost C_{total} defined as:

$$C_{\text{total}} = (1 - \theta)C_{\text{time}} + \theta C_{\text{energy}}, \tag{6.10}$$

where $\theta \in [0, 1]$ is set to 0 for extreme time-sensitive applications and to 1 for extreme energy-sensitive tasks. This metric allows for a unified representation of time and energy costs in monetary units (USD \$).

To provide a realistic insight into how DA-DPFL affects the total cost, we combine the communication and computation cost in the form of energy expenditure:

$$C_{\text{energy}} = C_{\text{comm}} + C_{\text{comp}}, \tag{6.11}$$

where C_{comm} and C_{comp} are the communication and computational cost, respectively.

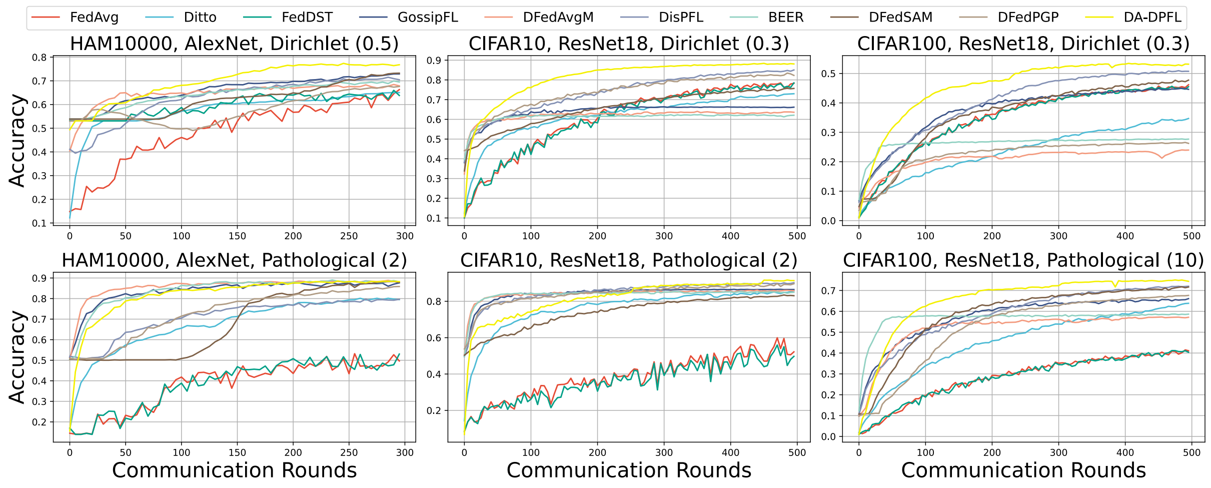


Figure 6.2: Test (*top-1*) accuracy of all baselines, including CFLs and DFLs, across various model architectures and datasets.

6.5.2 Performance Analysis

Test Accuracy Evaluation

DA-DPFL outshines all baselines in *top*-1 accuracy across six out of seven scenarios, maintaining robustness under extreme non-i.i.d. conditions ($n_{cls} = 2$ for CIFAR100) (refer to Figure 6.2 and Table 6.1). It exceeds the next-best DFL baselines (DisPFL and GossipFL) by $2 - 3\%$, with a minor shortfall in HAM10000 ($n_{cls} = 2$) by 0.5% against DFedAvgM. DA-DPFL consistently outperforms DisPFL in sparse model training and generalization while maintaining efficient convergence. In contrast, CFL lags in convergence due to its limited client participation per round. Momentum-based methods like DFedAvgM show accelerated initial learning, while BEER, with gradient tracking, exhibits rapid convergence, but does not necessarily reduce generalization error. DA-DPFL strikes an optimal balance between convergence rate and generalization performance, surpassing other baselines by achieving target accuracy with lower computational and communication costs.

Efficiency

We evaluate the efficiency of our algorithm by analyzing two key metrics: the Floating Point Operations (FLOP) required for inference and the communication overhead incurred during convergence rounds. To ensure a fair comparison, we employ the initialization protocol from DisPFL, thereby standardizing the initial communication costs and FLOP values in the initial pruning phase of the training.

Notably, the pruning stages integral to DA-DPFL lead to a significant reduction in these costs. This is evidenced by the final sparsity levels achieved: $(0.61, 0.56)$ for HAM10000, $(0.65, 0.73)$ for CIFAR10, and $(0.70, 0.73)$ for CIFAR100 under Dir. and Pat. partitioning, respectively. These results are obtained within the constrained communication rounds. A critical observation is that both the busiest communication costs and the training FLOPs for our approach are lower compared to the most efficient DFL baseline, DisPFL. These comparative insights are further elaborated in Table 6.2, with bold values that emphasize the efficiency of DA-DPFL.

Taking CIFAR10 as an example, Figure 6.3 shows the cost-effectiveness of DA-DPFL compared to other DFL baselines. Initially, when $\theta \to 0$ (emphasizing time over energy cost), DA-DPFL incurs a higher time cost. However, as θ increases beyond 0.2, DA-DPFL demonstrates remarkable advantages over the other DFL algorithms (represented by solid lines), with its lead expanding as θ further increases. Due to the system configuration, CFLs have significantly lower communication (1%) and computation (10%) costs compared to DFL, which (indicated by dotted lines) exhibits a higher overall cost efficiency, but a lower convergence speed. Overall, even when considering waiting time,

Table 6.1: Accuracy comparison of federated learning methods across different datasets

	HAM10000		CIFAR10		CIFAR100		Extreme
	Dir. (0.5)	Pat. (2)	Dir. (0.3)	Pat. (2)	Dir. (0.3)	Pat. (10)	**CIFAR100** **Pat.** (2)
FedAvg (McMahan et al., 2017)	65.92 ±0.3	55.68 ±0.4	79.30 ±0.2	60.09 ±0.2	46.21 ±0.4	41.26 ±0.3	*15.53* ±0.3
Ditto (Li et al., 2021b)	65.19 ±0.2	80.17 ±0.1	73.21 ±0.2	85.78 ±0.1	34.83 ±0.2	64.41 ±0.3	*86.39* ±0.2
FedDST (Bibikar et al., 2022)	66.11 ±0.3	55.07 ±0.4	78.47 ±0.2	56.32 ±0.3	46.01 ±0.2	41.42 ±0.2	*15.73* ±0.7
GossipFL (Tang et al., 2022b)	72.92 ±0.1	88.05 ±0.1	66.43 ±0.1	86.60 ±0.1	45.09 ±0.1	66.03 ±0.1	*89.68* ±0.1
DFedAvgM (Sun et al., 2022)	68.30 ±0.1	**88.89** ±0.1	65.05 ±0.1	85.34 ±0.2	24.11 ±0.1	57.41 ±0.1	*89.74* ±0.1
DisPFL (Dai et al., 2022)	71.56 ±0.1	80.09 ±0.1	85.85 ±0.2	90.45 ±0.2	51.05 ±0.3	72.22 ±0.2	*90.19* ±0.1
BEER (Zhao et al., 2022)	69.80 ±0.1	88.75 ±0.2	62.94 ±0.1	85.48 ±0.1	27.79 ±0.1	58.71 ±0.1	*92.07* ±0.1
DFedSAM (Shi et al., 2023)	73.74 ±0.2	88.47 ±0.3	75.74 ±0.2	83.51 ±0.1	47.86 ±0.2	71.76 ±0.1	*91.99* ±0.2
DFedPGP (Liu et al., 2024)	67.62 ±0.2	85.84 ±0.3	83.61 ±0.3	89.68 ±0.1	26.10 ±0.2	67.32 ±0.1	*88.93* ±0.1
DA-DPFL (*Ours*)	**76.32** ±0.3	88.36 ±0.3	**89.08** ±0.3	**91.87** ±0.1	**53.53** ±0.2	**74.91** ±0.1	***92.36*** ±0.1

Table 6.2: Busiest Communication Cost & Final Training FLOPs of all methods.

	HAM10000		CIFAR10		CIFAR100	
	Com. (MB)	FLOP (1e12)	Com. (MB)	FLOP (1e12)	Com. (MB)	FLOP (1e12)
FedAvg	887.8	3.6	426.3	8.3	353.3	2.3
Ditto	887.8	3.6	426.3	8.3	353.3	2.3
FedDST	443.8	2.0	223.1	7.1	176.7	1.6
GossipFL	443.8	3.6	223.1	8.3	176.7	2.3
DFedAvgM	443.8	3.6	223.1	8.3	176.7	2.3
DisPFL	443.8	2.0	223.1	7.1	176.7	1.6
BEER	443.8	3.6	223.1	8.3	176.7	2.3
DFedSAM	887.8	7.2	426.3	17	353.3	4.6
DFedPGP	422.8	3.8	423.9	9.1	351.3	2.5
DA-DPFL_Dir	**346.2**	**1.9**	**149.1**	**4.1**	**107.7**	**1.0**
DA-DPFL_Pat	**394.4**	**2.0**	**115.1**	**3.8**	**94.8**	**0.9**

DA-DPFL successfully achieves both cost and learning (convergence speed) efficiency.

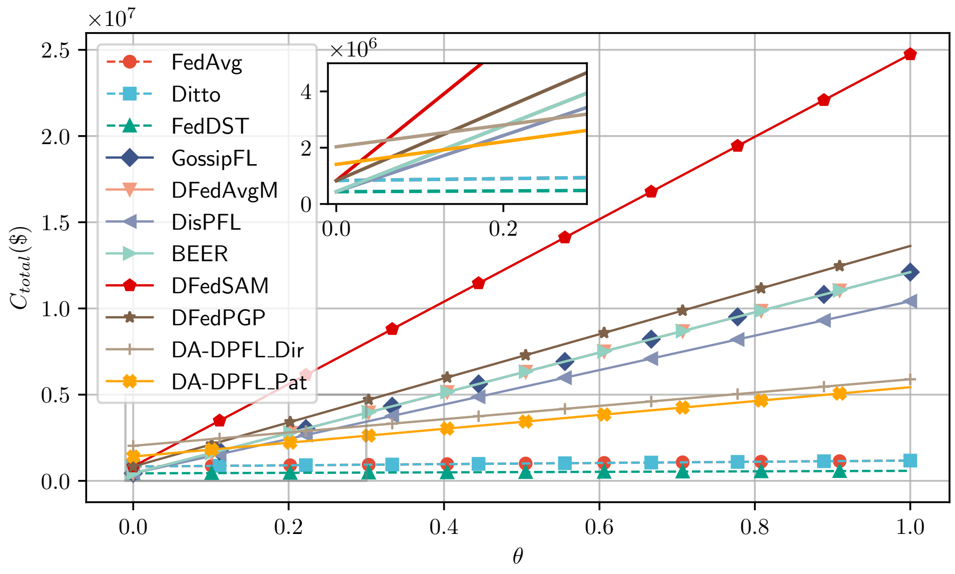


Figure 6.3: Total cost (energy and time cost, in USD) of DA-DPFL and all baselines evaluated on CIFAR10 against θ.

Extended Topology

To demonstrate the adaptability of our DA-DPFL, we conducted further experiments utilizing both *ring* and *fully-connected* (FC) topologies. These experiments were carried out in comparison with the above baselines using a *Dirichlet* partition with $\alpha = 0.3$ on CIFAR10. The results, presented in Table 6.3, show that DA-DPFL consistently surpasses other baselines, achieving higher performance with sparser models, within 500 communication rounds. DA-DPFL maintains a significant lead in performance.

Table 6.3: Performance comparison for ring and fully connected topologies

Method	Ring Topology		Fully Connected (FC)	
	Acc (%)	Sparsity (s)	Acc (%)	Sparsity (s)
GossipFL	66.12 ±0.1	0.00	71.22 ±0.2	0.00
DFedAvgM	65.89 ±0.1	0.00	69.89 ±0.1	0.00
DisPFL	67.65 ±0.2	0.50	86.54 ±0.2	0.50
BEER	62.92 ±0.1	0.00	68.77 ±0.1	0.00
DFedSAM	66.61 ±0.2	0.00	79.63 ±0.3	0.00
DFedPGP	67.88 ±0.3	0.00	84.62 ±0.3	0.00
DA-DPFL	**69.83** ±0.3	0.65	**89.11** ±0.2	0.68

6.5.3 Ablation Study

Analysis on Neighborhood Size M

We train ResNet18 on CIFAR10 to examine the impacts of the hyper-parameter M. The neighborhood size parameter M significantly influences the efficiency of scheduling in DA-DPFL. A higher value of M accelerates the convergence of our approach, primarily by improving the reuse of the trained model throughout the training process, although at the risk of potential delay.

As illustrated in Figure 6.4 (Bottom), while $M = 20$ slightly outperforms $M = 10$, it incurs approximately double the time delays.

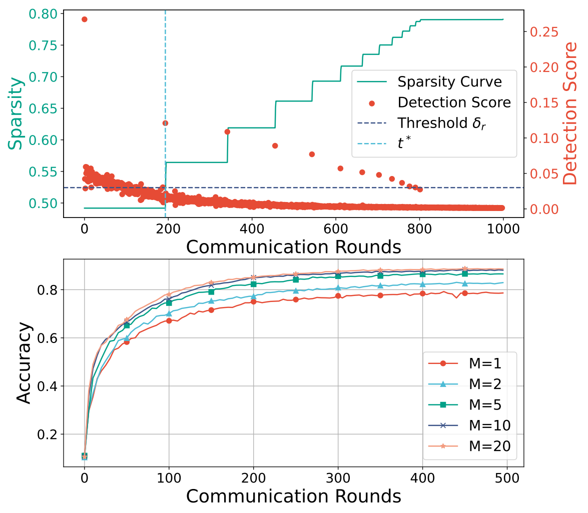


Figure 6.4: **(Top)** Relationship between sparsity and detection score; **(Bottom)** Impact of M involved in each training round on accuracy (CIFAR10, $Dir(0.3)$, $\delta_{pr} = 0.03$).

Threshold δ_{pr}

Extending total communication rounds from 500 to 1000, we ascertain that a target sparsity of $s = 0.8$ is attainable without compromising accuracy (DA-DPFL achieves 89% over DisPFL's 83.27%). This finding challenges the generalization gap assumption in (Dai et al., 2022), reducing the need for precise initial sparsity ratio selection in fixed sparsity pruning as DA-DPFL achieves equivalent or lower generalization error at higher sparsity levels through further pruning.

Figure 6.4 (Top) shows the pruning decisions based on average detection scores across clients and their sparsity trajectories. The initial high detection score validates the substantial disparity between the random mask and the RigL algorithm-derived mask, differing from EarlyCrop's centralized, densely initialized model approach. Post t^*, client models undergo incremental pruning in DA-DPFL, with the pruning scale diminishing due to reduced model compressibility, as evidenced by sparsity increasing at each pruning phase.

To illustrate the effect of the early pruning threshold δ_{pr}, we conducted experiments with CIFAR10 and ResNet18. Figure 6.5 underscores pruning timing significance, indicating varying optimal thresholds for different data partitions and corresponding detection score divergences. Early pruning, though accelerating sparsity achievement, impedes critical learning phases, while excessively delayed pruning equates to post-training pruning, incurring higher costs. Consequently, our results advocate for early-stage further pruning, ideally between 30-40% of total communication rounds, aligning with a threshold range of 0.02-0.03, to balance model performance with energy efficiency. If the model is known to be redundant for the task, a larger threshold can facilitate earlier pruning; otherwise, a smaller threshold should be adopted to postpone pruning. This consideration stems from the constraint on the number of rounds—without such a limitation, PQI would ultimately enable the model to achieve the target sparsity without performance degradation.

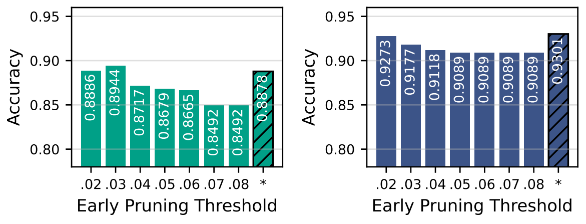


Figure 6.5: Impact of δ_{pr} on final prediction accuracy of achieving sparsity $s = 0.8$ with CIFAR10 ($M = 10$) *Dir* (left) and *Pat* (right) partitions (in which * stands for DA-DPFL without further pruning, i.e., fixed sparsity $s = 0.5$).

Waiting Threshold N

To ensure a fair comparison, we add $N = \{0, 2, 5, 10\}$ with the same experiment setup as in Section 6.5.1 for *Dir* partition. The experimental results in Figure 6.6 demonstrate a clear trend: increasing N consistently improves model accuracy across the CIFAR10 and CIFAR100 datasets, with CIFAR10 seeing up to a 1.87% increase and CIFAR100 seeing a 1.41% increase in accuracy from $N = 0$ to $N = 10$. Interestingly, the HAM10000 dataset shows no gain when further increasing N to 10. With $N = 5$ achieving the best performance, it suggests that there is a limit to the reuse of the model, and task-specific characteristics influence the optimal selection of N. For all the datasets, even the performance of the model for cases $N = 0$ is higher than that of DisPFL, illustrating the effectiveness of our *further pruning* strategy. By selecting N, one can balance the trade-off between waiting and model performance.

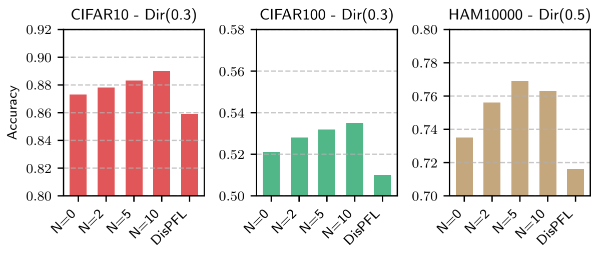


Figure 6.6: Performance w.r.t. number of maximum waiting N.

6.5.4 Scalability

Table 6.4 evaluates DA-DPFL's scalability against DisPFL on CIFAR10 and CIFAR100 with $K = 200$ clients. DA-DPFL consistently outperforms DisPFL, achieving up to 5.27% higher accuracy in CIFAR10 and 1.92% in CIFAR100. These gains highlight its robustness in large-scale FL, effectively mitigating performance degradation from increased heterogeneity.

Table 6.4: Accuracy comparison of DisPFL and DA-DPFL on CIFAR10 and CIFAR100 with $K = 200$ clients

	CIFAR10		CIFAR100	
	Dir. (0.3)	Pat. (2)	Dir. (0.3)	Pat. (10)
DisPFL (Dai et al., 2022)	76.99 ±0.1	85.27 ±0.1	45.49 ±0.1	64.36 ±0.2
DA-DPFL	**82.26** ±0.2	**86.68** ±0.1	**46.75** ±0.1	**66.28** ±0.1

6.5.5 Extreme Data Heterogeneity

As shown in the rightmost column of Table 6.1, DA-DPFL achieves 92.36% accuracy on CIFAR100 under extreme data heterogeneity (Pat., $\alpha = 2$), outperforming all baselines. This setting is particularly challenging, as evidenced by the collapse of centralized methods (FedAvg: 15.53%, FedDST: 15.73%). While traditional decentralized methods show improved resilience compared to centralized approaches, DA-DPFL still achieves a noticeable margin over DisPFL by 2.17%. The performance gains in this extreme scenario demonstrate DA-DPFL's capability to maintain robust performance even when data distributions are highly skewed across clients.

6.6 Conclusions

In this chapter, we presented DA-DPFL, a novel framework that effectively addresses the challenges of communication efficiency, training efficiency, and personalization in decentralized federated learning. DA-DPFL introduces two key innovations: (1) a dynamic aggregation mechanism that allows clients to reuse previously trained models within the same communication round, and (2) a sparse-to-sparser pruning strategy that progressively increases model sparsity based on model compressibility and optimal pruning timing.

Our comprehensive experiments demonstrate that DA-DPFL consistently outperforms both centralized and decentralized FL baselines across various datasets, model architectures, and data partition strategies. DA-DPFL achieves superior test accuracy while significantly reducing communication and computational costs, with up to 5× reduction in energy consumption.

The dynamic aggregation mechanism in DA-DPFL accelerates convergence by enabling knowledge transfer from previously trained models, while the adaptive pruning strategy ensures optimal trade-offs between model sparsity and performance. Our theoretical analysis confirms the convergence properties of DA-DPFL and shows that the framework provides improved generalization bounds compared to fixed-sparsity approaches.

DA-DPFL particularly excels in challenging scenarios with extreme data heterogeneity, where centralized methods often fail completely. The framework's ability to maintain high performance under diverse network topologies and scale to larger client populations further demonstrates its robustness and practical applicability.

The advances presented in this chapter contribute to the broader goal of making federated learning more efficient and effective in real-world deployments, especially in resource-constrained edge computing environments where communication and computational efficiency are paramount concerns.

Chapter 7

MOBILE: Mobility and Outage-Based Intelligent Federated Learning in Mobile Edge Computing

7.1 Introduction

Federated Learning (FL) (McMahan et al., 2017) is a distributed learning paradigm that trains models iteratively while preserving data privacy. Instead of centralizing data in one location, FL allows distributed devices, *clients*, to train models on local data, with a central server coordinating the aggregation process each round. Only model updates, typically as gradients or parameters, are shared with the central server. The final global model is shared among clients after the training. This prevents the transmission of (potentially sensitive) humongous amounts of data among clients, leverages nodes' resources, and promotes improved learning outcomes.

Assuming a distributed computing environment with stationary (non-mobile) clients is often not feasible, particularly in the context of Mobile Edge Computing (MEC). In these settings, clients, e.g., mobile users and vehicles, exhibit significant mobility behavior due to irregular availability of connectivity, fluctuating network conditions, and varying computational resources. Clients may drop out during model training, requiring the system to swiftly adapt to dynamic factors such as communication delays, device failures, and asynchronous updates. In MEC, client mobility and resource heterogeneity introduce additional unpredictability, demanding mechanisms to maintain learning. Intermittent connectivity, where clients only occasionally participate, complicates efforts to synchronize model updates (Lim et al., 2020). The conventional stable client-server relationship in FL is challenged in MEC. The sporadic availability of clients due to mobility undermines the assumptions of consistent participation in training and synchronous model updates, as stated in (Nishio & Yonetani, 2019). Effective resource management and client selection during training are essential due to clients' mobility. (Zhou et al., 2022) selects clients having prominent contributions to distributed training while considering network conditions. (Lai et al., 2021) guides client selection in terms of scalability, staleness, and robustness. OCEAN (Xu & Wang, 2021) attempts client selection under energy budget constraints w.r.t. Channel State Information (CSI). Among these efforts and others, introducing client mobility helps diminish impacts by considering system heterogeneity. However, only a few of them acknowledge the fact that the clients, who have been selected in a model training round, may fail during the learning process due to e.g., unstable connections between them and the server or their mobility patterns, e.g., several clients might become unreachable, thus, they cease to contribute to the training process. Moreover, although these works address the consequences of mobility, none of them directly addresses the current mobility and mobility patterns of clients along with the optimal subset of clients to be engaged *per* round under the influence of mobility, thereby exacerbating the disparity between theoretical frameworks and practical deployments.

This chapter tackles the challenges associated with employing FL in mobile computing due to clients' mobility. We introduce the **M**obility & **O**utage-**B**ased **I**ntelligent Federated **LE**arning (**MOBILE**) framework dealing with regularized Mixed-Integer Quadratic Programming (MIQP), which exploits FL within the intricate landscape of uncertainty due to mobility, stochastic client selection and optimized bandwidth allocation *per* round. MOBILE takes into account the clients mobility patterns to optimally (i) select *the most promising clients per round* that are highly likely to complete the training tasks and, therefore, (ii) allocate the available bandwidth for sharing the model updates during training. By directly incorporating spatiotemporal mobility patterns (current and historical) while maintaining orthogonality to most existing FL algorithms, MOBILE reveals new prospects for FL in MEC.

The remainder of the chapter is organized as follows: Section 7.2 elaborates on related work and contribution. Section 7.3 discusses problem fundamentals, while Section 7.4 introduces MOBILE. Section 7.5 provides experimental evaluation. Section 7.6 concludes the chapter with future agenda.

7.2 Related Work & Contribution

Several studies deal with the efficiency of FL in mobile computing focusing primarily on energy consumption. However, as stated in (El Mokadem et al., 2023), the assumptions used in these approaches result in solutions that are not universally applicable in distributed learning. We provide these key assumptions: **A1:** Energy consumption takes precedence over mobility in client selection and bandwidth allocation. **A2:** The selected clients are assumed to be *always* reliable, reachable, and always expected to return the locally optimized updated models to the FL server. Instead, MOBILE revisits these assumptions by prioritizing optimized client selection and resource allocation based on clients' historical and current mobility rather than focusing exclusively on energy constraints.

MADCA-FL (Zhang et al., 2023) integrates FL into vehicular networks by optimizing model accuracy while meeting stringent latency and energy constraints. Although it leverages Roadside Units (RSUs) for vehicle selections, the energy constraints imposed (ranging from 0.006 to 0.0012 J for communication) are fixed, potentially overlooking the dynamic and unpredictable nature of real-world vehicular environments. OCEAN (Xu & Wang, 2021) seeks to maximize the accuracy of the FL model by optimizing client selection under long-term energy constraints. Similarly, OCEAN performs optimization based on fixed energy budget of 0.15 J, which may not be fully under the typical patterns in real-world scenarios. For instance, modern mmWave devices operate at ~ 5 Watts even at low frequencies (Kanhere et al., 2022). Devices connected in 4G networks consume $1.5 - 2.5$ Watts when transmitting data, while $0.1 - 0.5$ Watts in idle mode (Trust, 2024). This indicates that even the less rigor theoretical energy budget (0.15 J) in OCEAN for the entire FL task for a single client would be depleted in under 0.1 seconds. Moreover, with a typical cell phone battery that holds $66,600$ J of energy, a common phone chip that has a TDP of $5 - 10$ Watts (Bhat et al., 2019), and a Nvidia 4090 GPU having a TDP of 450 Watts, the gap between theoretical and actual energy consumption becomes remarkably clear. This prompts our question: *Do we place objectives only on energy constraints while neglecting the significant factor of client mobility deciding on (i) how many and (ii) which clients to select in a FL process?*

The main focus of those approaches is energy. Nonetheless, such methods fail to adequately account for clients' mobility patterns in client selection and the unreliability of clients, who may not always remain within the coverage area *during* the FL process.

However, introducing client mobility to client selection and resource allocation during learning is not trivial. In studies on human and vehicle trajectory prediction, the interval between samples or prediction horizons typically ranges from a fraction of a second to three minutes (Liang & Zhao, 2022; Messaoud et al., 2021; Shi et al., 2021; Song et al., 2021; Xue et al., 2018). We identified fundamental challenges: **CH1:** These short intervals yield infeasibility to configure FL tasks on a per-round basis, as the time required for model parameters uploads, downloads, and local training extends well beyond such prediction horizons. **CH2:** The length of available trajectories varies significantly between subjects and is often insufficient to cover the full duration required for FL tasks, which generally span 50 to 100 rounds. As a result, directly applying trajectory prediction or generalizing these datasets within the context of mobility-aware

FL is impractical. To address the discrepancies between interval duration, prediction horizon, and misaligned trajectory lengths, research on Point of Interest (POI) prediction (Feng et al., 2018; Liu et al., 2021) aligns with client selection in FL.**CH3**: POI-based methods, however, rely heavily on contextual information, which tends to provide accuracy primarily in a categorical sense (being more accurate in prediction falls in the same category of the ground truth), rather than offering spatial-temporal precision necessary to capture clients' movement and thereby facilitate the FL process. **CH4**: POI prediction methods rely on clients' 'check-ins' introducing inherent inconsistencies in the temporal alignment of mobility data. Although asynchronous FL could theoretically alleviate such alignment issues, it introduces significant challenges in maintaining model consistency and fairness, particularly when combined with the high variability of client mobility patterns. These factors render the integration of POI-based predictions infeasible in mobility-aware FL cases that rely on synchronous communication and round-by-round updates to ensure system-wide consistency.

MOBILE needs to address CH1-CH4 w.r.t. client selection and resource allocation considering clients' *mobility* and *capability* to successfully conclude their assigned training task per round. Our experiments over mobility dataset (Yabe et al., 2024) show the importance of incorporating mobility in the FL process. We showcase that without considering mobility, the probability that a client successfully completes model download, performs local training, and transmits the updated model to the server ranges from 20% to 32%. This emphasizes the need for MOBILE to address mobility to achieve efficiency of FL. MOBILE leverages both current and historical mobility that facilitates optimization, which is largely *orthogonal* to FL approaches thus being applied in series. Our contributions are:

1. We introduce MOBILE for jointly optimizing client selection and resource allocation with minimal overhead in MEC-based FL environments.
2. We formulate MOBILE's objective as a regularized MIQP problem under the principle of means-variance optimization on clients' mobility patterns.
3. We address the cold-start challenge in mobility-aware FL by proposing an adaptive approach.
4. We provide comprehensive experimental evaluation of MOBILE against FL baselines on i.i.d. and non-i.i.d datasets and real-world mobility dataset.

7.3 Problem Fundamentals

7.3.1 Preliminaries in Learning System Model

Consider a FL system operating in a MEC environment, where a central server is deployed on a Base Station (BS) or access point, hereinafter, referred to as BS. All components are part of the Radio Access Network (RAN) while the BS establishes wireless communication with mobile end-user devices (clients) within a limited coverage area, ensuring signal coverage for learning tasks. The BS coordinates the FL process across a set $\mathcal{K}$ of K clients for a total of T training rounds. Each client, indexed by an integer $k \in \{1, 2, \ldots, K\} = \mathcal{K}$ possesses a local dataset $\mathcal{D}_k \in \mathcal{X}_k \times \mathcal{Y}_k$, where $\mathcal{X}_k$ and $\mathcal{Y}_k$ represent the feature and label space of client k. Client k participates in the training a *global* model w by performing local updates. In round $t \in \{1, 2, \ldots, T\}$, BS transmits the current global model w_t to randomly *selected subset* of clients $\mathcal{S}_t \subset \mathcal{K}$. Each selected client minimizes a local objective $\mathcal{L}_k(w)$ over data $\mathcal{D}_k$ obtaining the model $w_k^{(t+1)}$:

$$\mathcal{L}_k(w) = \frac{1}{|\mathcal{D}_k|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_k} \ell(w; \mathbf{x}_i, y_i), \tag{7.1}$$

where $\ell(w; \mathbf{x}_i, y_i)$ is the loss of model w on data $(\mathbf{x}_i, y_i)$. Each client minimizes this objective, thus improving the model locally by using gradient-based optimization. Once local updates are completed, the clients upload their updated models $w_k^{(t+1)}$ back to BS. The BS aggregates the received local models obtaining the updated global model for next round, w_{t+1}. This aggregation is performed by averaging the model parameters, e.g., by adopting FedAvg (McMahan et al., 2017):

$$w_{t+1} = \frac{1}{|\mathcal{S}_t|} \sum_{k \in \mathcal{S}_t} w_k^{(t+1)}, \tag{7.2}$$

across all participating clients $\mathcal{S}_t$ in round t. Alternatively, gradient averaging can be used instead of model parameter averaging, where each client k locally computes the gradient $\nabla \mathcal{L}_k(w)$ w.r.t. its local objective. The global model in BS is then updated by averaging the gradients:

$$w_{t+1} = w_t - \eta \cdot \frac{1}{|\mathcal{S}_t|} \sum_{k \in \mathcal{S}_t} \nabla \mathcal{L}_k(w_t), \tag{7.3}$$

where $\eta \in (0, 1)$ is a learning rate for the global model update.

7.3.2 Communication Model

By operating within RAN, the FL system leverages the proximity of BS and clients to meet the ultra-low latency demands of services. In our FL ecosystem, clients move freely, while the BS remains stationary and positioned at the center of a densely populated client region to maximize coverage. In the model (or gradient) exchange process between BS and clients, existing works, e.g., (Zhang et al., 2023) assume unlimited resources on BS side, thus disregarding the model download process. Similarly, MOBILE focuses on the model upload phase from clients to BS. As highlighted in (Konečný et al., 2017; Lim et al., 2020), the primary communication bottleneck in MEC-based FL environments is model uploading from clients rather than global model downloading from BS. Such bottleneck appears due to higher data rates typically available for downlink than uplink, especially in wireless networks with limited uplink resources. To address this, we employ Orthogonal Frequency Division Multiple Access (OFDMA) for models/gradients uploading (from selected clients to BS), ensuring efficient communication. OFDMA enables parallel data transmission by dividing the available bandwidth B into orthogonal sub-channels. The sub-channels are shared among clients without interference. This allows multiple clients to upload models simultaneously, maximizing bandwidth utilization, thus reducing the latency associated with client uploads. Moreover, to avoid dividing B into relatively small fragments that cannot support a meaningful communication rate, we set a minimum bandwidth fraction, $0 < b_{\min} < 1$, allowed for allocation. Following (Zhang et al., 2023), consider a selected client $k \in \mathcal{S}_t$ in round t communicating with BS with client's transmission power $p_{k,t}$ and distance from BS $d_{k,t}$. Given its characterized CSI h_k^t, the upload communication rate $R_{k,t}$ is:

$$R_{k,t} = b_k^t B \log_2 \left(1 + \frac{p_{k,t} h_{k,t} d_{k,t}^{-\theta}}{\mathbb{P}} \right), \tag{7.4}$$

where $b_k^t \in [b_{\min}, 1)$ is channel bandwidth fraction of each sub-carrier, $h_{k,t}$ is channel power gain, $\mathbb{P}$ is noise power and $\theta > 0$ is the path loss exponent. The BS has a limited coverage area represented as a circular region with BS at the center having maximum radius δ where stable connections are achieved. As the distance $d_{k,t}$ between BS and client k approaches δ, the signal strength deteriorates to the point where a stable connection for models/gradients transmission cannot be maintained reliably. At a distance $d_{k,t} > \delta$, client k is highly unlikely to transmit its locally updated model/gradients to BS on time. This results in (i) absence of client k contribution to FL learning in round t, (ii) wasting client's resources for locally training the model, (iii) degrading the learning process and global model aggregation on BS; although client k was active for training at the start of that round.

7.3.3 Problem Formulation

Rationale of the problem

At the beginning of the round t, all clients report their distances $\{d_{k,t}\}, k \in \mathcal{K}$ from the BS, or the BS estimates the distances based on signal strength. The BS, leveraging clients' distances, mobility patterns, and correlations among their movements (as elaborated later), determines optimally (i) *which* clients $k \in \mathcal{K}'_t \subset \mathcal{K}$ will participate in the current round t while (ii) optimally allocating bandwidth fraction b^t_k to each selected client to support model/gradients transmission such that $\sum_{k \in \mathcal{K}'_t} b^t_k = 1$. After client selection, each selected client $k \in \mathcal{K}'_t$ downloads the global model w_t from BS, performs local training of w_t over local $\mathcal{D}_k$, and *attempts* to upload the updated model/gradients $w^{(t+1)}_k$ back to BS using the allocated bandwidth b^t_k. If a selected client $k \in \mathcal{K}'_t$ moves out of BS's coverage i.e., $d_{k,t+1} > \delta$, or is unable to complete the updated model upload within a predefined duration $\mathcal{T} > 0$ (as a fraction of time duration between consecutive rounds), it is referred to as an *unsuccessful selected* client k. Unsuccessful selected clients contribute minimally (or not at all) to global model aggregation for round t. On the other hand, successful clients $\mathcal{S}_t \subseteq \mathcal{K}'_t \subset \mathcal{K}$ are those within the coverage having completed models/gradients uploads within $\mathcal{T}$. Only the uploaded models of successful clients $k \in \mathcal{S}_t$ are aggregated by BS to update the global model w_{t+1} for the next round $t+1$ in (7.2) or (7.3). The local models of unsuccessful selected clients $k \in \mathcal{K}'_t \setminus \mathcal{S}_t$, evidently, cannot be used for global model updating at round t. $\mathcal{T}$ is application specific and refers to the duration between successive global model aggregations w_t and w_{t+1} at round t and $t+1$, respectively, minus the time for global model download along with other tasks like model aggregation.

Fundamentals of client selection problem

MOBILE diverges from sequential client upload and download process employed in (Nishio & Yonetani, 2019), where client communications proceed in sequence under OFDM. Instead, at each t, MOBILE optimizes client selection and resource allocation focusing on clients $\mathcal{S}_t$ with the highest likelihood of completing local training and successfully uploading model updates at any duration $0 < \tau_{k,t} \leq \mathcal{T}$ between rounds t and $t+1$, $\forall k \in \mathcal{S}_t$. By strategically allocating channel resources, we maximize the number of successful interactions. i.e., selected clients succeed in uploading their locally updated model/gradients to BS on time, per round, ultimately supporting effective and resilient FL. Consider $a^t_k \in \{0, 1\}$ as an indicator that represents whether client k is selected to participate in training of round t. The K-dimensional vector $\mathbf{a}_t = [a^t_1, \ldots, a^t_K]^\top$ captures the selection of all K clients at the beginning of round t, i.e., $\mathcal{K}'_t = \{k \in \mathcal{K} : a^t_k = 1\}$. Hence, our aim is selected clients in $\mathcal{K}'_t$ be able to successfully upload their models within $\mathcal{T}$, i.e., we target obtaining $\mathcal{S}_t \equiv \mathcal{K}'_t$ before or at the end of $\mathcal{T}$. Similar to the rationale of (Nishio & Yonetani, 2019), we aim to engage as many clients as possible in distributed learning at round t under the risk that the selected clients will finish on time. For each selected client, it should hold in an Independent and Identically Distributed (i.i.d.) setup that $P(d_{k,\tau} \leq \delta) \rightarrow 1$ at the time of model uploading with $\tau_{k,t} \in (0, \mathcal{T}]$. The more clients' models aggregated in the BS (see (7.2) or (7.3)), the higher the quality of the global model at round $t+1$, which will be shared with the newly selected clients at round $t+1$. Such a setup has proven effective in non-i.i.d data as well; see Section 7.5.6. MOBILE attempts to select the most promising clients at t, that will show as successful clients at the end of round t. We seek those clients such that the ratio $\frac{|\mathcal{S}_t|}{|\mathcal{K}'_t|} \rightarrow 1$. A selected client $k \in \mathcal{K}'_t$ is classified as a successful selected client $k \in \mathcal{S}_t$ right *after* the deadline $\mathcal{T}$.

Fundamentals of selected client bandwidth allocation

The BS's bandwidth B is assigned as fractions $b_k^t \in [b_{\min}, 1)$ to selected clients $k \in \mathcal{K}'_t$ at the beginning of round t. Consider the vector $\mathbf{b}_t = [b_1^t, \ldots, b_K^t]^\top$, where $b_k^t \geq b_{\min}$ if client $k \in \mathcal{K}'_t$, otherwise, $b_k^t = 0$ for non-selected clients; there is no allocated bandwidth to clients that do not participate in training at t. Since we target to obtain as many successful selected clients $\mathcal{S}_t$ out of all selected clients $\mathcal{K}'_t$ as possible, we define as *resource waste* the amount of bandwidth allocated to selected client, who are not successful after deadline $\mathcal{T}$, i.e., $B^- = B(1 - \sum_{k \in \mathcal{S}_t} b_k^t)$. We not only attempt to increase the successful clients participation ratio for increasing the predictive quality of the global model, but also to engage the most promising clients having the resource waste minimized.

Problem 7. *Given deadline duration $\mathcal{T}$, seek a subset of selected clients $\mathcal{K}'_t \subset \mathcal{K}$ such that ratio of successful selected clients $\mathcal{S}_t \subseteq \mathcal{K}'_t$ out of selected ones is maximized, i.e.,*

$$\max \frac{|\mathcal{S}_t|}{|\mathcal{K}'_t|}, \textit{s.t. } \mathcal{K}'_t \in \mathcal{P}^+(\mathcal{K}), \tag{7.5}$$

where $\mathcal{P}^+(\mathcal{K})$ is the power-set of $\mathcal{K}$ excluding the empty set.

Problem 7 maximizes the number of selected clients participating successfully in aggregation at each round t while minimizing resource waste. The $|\mathcal{S}_t|$ clients successfully contribute to model updates in round t w.r.t. $\mathcal{T}$. However, due to inherent uncertainty introduced by client mobility, (7.5) cannot be optimized directly.

Transforming Problem 7

To approximate (7.5), one pursues an indirect approach by integrating probabilistic estimations of client mobility. This approach enables selection of clients and resource allocation, enhancing the likelihood of successful interactions despite fluctuating mobility patterns and varying connectivity conditions.

In FL environments, BS decides on clients' selection. Unlike existing methods like (Xu & Wang, 2021), one could account for client mobility in client selection; a factor typically ignored. One could begin by minimizing the total communication time across T rounds while meeting specific system constraints. Communication time is highly sensitive to spatial distribution of client locations. Clients located farther from BS introduce delays acting as *stragglers* in the learning process. The communication cost at round t, denoted as C_t, is the cumulative uploading times, $\forall k \in \mathcal{K}$:

$$C_t = \sum_{k \in \mathcal{K}} a_k^t \frac{G_k}{R_k^t}, \tag{7.6}$$

where G_k represents the size (amount) of gradients/model parameters that each client k needs to upload to BS. Therefore, one could transform Problem 7 into:

Problem 8.

$$\min \sum_{t=0}^{T-1} C_t - \rho \sum_{t=0}^{T-1} \|\mathbf{a}_t\|_1 \tag{7.7}$$

$$s.t. \quad b_{\min} \leq b_k^t \leq 1, \quad \forall k, \forall t, \tag{7.8}$$

$$\sum_{k \in \mathcal{K}} b_k^t = 1 \; and \; a_k^t \in \{0,1\}, \quad \forall k, \forall t, \tag{7.9}$$

$$\tag{7.10}$$

Problem 8 minimizes C_t, where one can effectively reduce $\tau_t^\star = \min\{\tau_{k,t}\}, k \in \mathcal{K}_t'$, which represents the completion time of the latest-finishing client within $\mathcal{T}$. Hence, this minimizes the probability of $\tau_t^\star > \mathcal{T}$. Moreover, this approach increases the success probability of selected clients and subsequently reduces the bandwidth waste $B(1 - \sum_{k=0}^{|\mathcal{S}_t|} b_k^t)$. In addition, (7.7) encourages higher client participation. Note, ρ is a tunable parameter that balances the dual objectives of minimizing the total communication time and maximizing client selection by seeking a high value of L_1 norm (Manhattan) of $\mathbf{a}_t$, $\|\mathbf{a}_t\|_1 = |a_1^t| + \cdots + |a_K^t|$.

However, two significant challenges arise: **(i)** While the current client distances d_k^t are known at the time of selecting the clients and thus allocating bandwidth, the end-of-round distances (d_k^{t+1}) and channel conditions during uploading (i.e., close to the deadline $\mathcal{T}$) are *unknown*. **(ii)** (7.7) is a mixed-integer nonlinear programming objective due to both integer decision variables (a_k^t) and nonlinearities in the objective such as $\frac{G_k}{R_k^t}$ with R_k^t defined in (7.4). Since our target incorporates client mobility for selection and resource allocation, we face the inherent challenge of the uncertain future distances of the selected clients from BS during and more importantly at the end of round t.

7.4 The MOBILE Framework

MOBILE addresses the challenges in FL with mobile clients described in Problem 8. MOBILE performs client selection based on current distances and recent correlated clients' mobility patterns along with optimized bandwidth allocation accounting for outages during each training round.

7.4.1 Mobility Patterns Representation for FL

At the beginning of round t, we define the *return* vector $\mathbf{r}_t = [r_1^t, \ldots, r_K^t]^\top$, which prioritizes clients based on current proximity to BS. Return r_k^t for client $k \in \mathcal{K}$ is defined as:

$$r_k^t = 1 - \frac{d_{k,t}}{d_{\max,t}}, k \in \mathcal{K}, \tag{7.11}$$

where $d_{k,t}$ is the distance of client k from BS at the start of round t and $d_{\max,t}$ is the max distance to the BS of all the clients in round t. The return r_k^t ensures that a client k closer to BS is given a higher priority for bandwidth allocation at round t, as this client is more likely to be successful at the end of round t w.r.t. local model training and uploading to BS. Based on clients returns, central to our approach is a $K \times K$ matrix Q that captures the spatio-temporal correlations of clients' mobility w.r.t. their distances from BS in the recent past. These correlations reflect how clients' movements historically align, providing insight into their future behaviors relative to BS. If several clients have shown consistent mobility patterns that bring them 'closer' to BS, this would be reflected in higher correlations within Q; therefore, these clients are more likely to be successful at the end of a round t. To construct the elements of $Q_t = [\xi_{(k,\kappa),t}]$, $k, \kappa \in \mathcal{K}$ at the beginning of round t, we adopt a 'sliding window' $\mathcal{W}_t = \{\mathbf{r}_{t-W+1}, \ldots, \mathbf{r}_t\}$ of size W representing the W most recent historical returns (reflecting distances from BS) of the clients at $t - W + 1 \leq \tau \leq t$.

For each client k and round $\tau \in [t - W + 1, t]$, we sample its accumulated distances from BS and calculate returns $\{r_k^{t-W+1}, \ldots, r_k^{\tau}, \ldots r_k^t\}$. The covariance matrix Q_t is defined as:

$$Q_t = \frac{1}{W} \sum_{\tau=t-W+1}^{t} (\mathbf{r}_\tau - \bar{\mathbf{r}}_t)(\mathbf{r}_\tau - \bar{\mathbf{r}}_t)^\top, \tag{7.12}$$

where $\bar{\mathbf{r}}$ represents the average return over window $\mathcal{W}_t$:

$$\bar{\mathbf{r}}_t = \frac{1}{W} \sum_{\tau=t-W+1}^{t} \mathbf{r}_\tau. \tag{7.13}$$

Q influences client selection and bandwidth allocation at the beginning of round t by capturing historical returns reflecting mobility patterns and correlations among clients in the last W rounds. Clients with correlated movement patterns (and returns) increase the likelihood of stable BS connections, thus, they are prioritized in participating in FL process due to potentially enhancing the overall stability of training.

7.4.2 MOBILE Optimization Principles

Given current $\mathbf{r}_t$ and mobility patterns reflected via the correlations of clients' returns Q_t up to t, we construct the optimization objective for MOBILE. For convenience, we repeat the participation variables $\mathbf{a}_t = [a_1^t, \dots, a_K^t]^\top$ with $a_k^t \in \{0, 1\}$ indicating whether client k is selected or not, and bandwidth fraction allocation $\mathbf{b}_t = [b_1^t, \dots, b_K^t]^\top$ with $b_k^t \in [b_{\min}, 1]$ at the beginning of round t. Once client k is selected to participate in round t ($a_k^t = 1$), we seek the best bandwidth fraction b_k^t. Otherwise, we obtain $a_k^t = 0$, thus, no bandwidth allocation ($b_k^t = 0$). To represent this conditional dependency between a_k^t and b_k^t, we introduce the constraint for each client k: $b_{\min} a_k^t \leq b_k^t \leq a_k^t$, with $\sum_{k \in \mathcal{K}} b_k^t = 1$.

Each element $\xi_{(k,\kappa),t}$ in Q_t shows how clients k and κ returns tend to 'move' together. A high covariance indicates a strong relationship between these clients, while from the mobility perspective, it indicates how they move in relation to each other. Positive covariance means two clients tend to move together, while negative one means they move in opposite directions (from/to BS). If the covariance among clients is minimized, the overall variance of the returns is also minimized. A lower covariance implies that selected clients in the FL process have less correlated returns, which reduce the overall variability. Lowering covariance helps create a diversified client participation. By holding clients with low or negative correlations, we avoid putting all 'risky' clients (having a high probability of being unsuccessful) in round t. Minimizing covariance among clients' returns leads to amplifying the diversification effect as, gains in one (successful) client can offset losses in another (unsuccessful). MOBILE aims to achieve the maximum return for a given level of risk; the latter is quantified by the number of selected and unsuccessful clients and bandwidth waste. By maximizing the expected returns of all selected clients while minimizing Q_t, MOBILE achieves the same return but with a reduced risk, leading to better overall performance on a risk-adjusted basis. Minimizing Q_t allows MOBILE to structure a list of participants that is less sensitive to mobility fluctuations. This stability makes client selection less vulnerable to short-term shifts. MOBILE considers having no more than M selected clients with $M \leq K$, thus, yielding $|\mathcal{K}_t^t|$ selected clients with $0 < |\mathcal{K}_t^t| \leq M$. We then enforce a linear constraint on the number of selected participants, i.e., $0 < \sum_{k\in\mathcal{K}_t'} a_k^t \leq M$. MOBILE's objective is formulated as a regularized version of MIQP under means-variance optimization of clients' returns:

Problem 9 (MOBILE Objective)**.**

$$\max_{\mathbf{a},\mathbf{b}} \quad \sum_{t=0}^{T-1} \mathbf{r}_t^\top \mathbf{b}_t - \lambda \mathbf{b}_t^\top Q_t \mathbf{b}_t + \gamma \|\mathbf{a}_t\|_1 \tag{7.14}$$

$$\textit{s.t. } C1{:}a_k^t \in \{0,1\}, \quad k \in \mathcal{K}, \forall t \in \{0,\dots,T-1\} \tag{7.15}$$

$$C2{:}0 < \sum_{k\in\mathcal{K}_t'} a_k^t \leq M \leq K, \forall t \in \{0,\dots,T-1\} \tag{7.16}$$

$$C3{:} \sum_{k\in\mathcal{K}} b_k^t = 1, \quad \forall t \in \{0,\dots,T-1\} \tag{7.17}$$

$$C4{:}b_{\min} a_k^t \leq b_k^t \leq a_k^t, k \in \mathcal{K}, \forall t \in \{0,\dots,T-1\} \tag{7.18}$$

Problem 9 integrates both historical mobility and current clients' proximity, aiming to optimize client selection and bandwidth allocation across rounds.

Objective Explanation: The term $\mathbf{r}_t^\top \mathbf{b}_t$ expresses the expected return of selected clients based on their proximity to BS, thus, maximizing the probability of being successful at the end of round t. Maximizing this term reflects maximizing the number of the most appropriate clients to be selected and successful as discussed in Problem 7. The term $\lambda \mathbf{b}_t^\top Q_t \mathbf{b}_t$ leverages the correlations in Q_t to favor clients whose historical returns indicate a higher likelihood of stable connectivity. The regularization term $\gamma\|\mathbf{a}_t\|_1$ controls the client selection to approximate the optimal number of participants reducing resource waste, i.e., increasing the number of selected clients at the expense of selecting clients that cannot be successful. Given BS coverage δ, bandwidth B, and duration limit $\mathcal{T}$, there exists an upper bound $|\mathcal{S}_t^\star|$ on number of clients that can successfully complete training and model uploading. If selected clients $|\mathcal{K}_t'|$ significantly exceeds $|\mathcal{S}_t^\star|$, some clients will inevitably fail due to bandwidth constraints. Conversely, if $|\mathcal{K}_t'|$ is significantly less than $|\mathcal{K}|$, the system under-utilizes bandwidth missing the chance for optimal performance. To address the mismatch between number of selected clients and (ground truth) optimal number of participants per round, the regularization $\gamma\|\mathbf{a}_t\|_1$ helps in adjusting the number of selected clients by tuning γ encouraging $|\mathcal{K}_t'|$ to approximate $|\mathcal{S}_t^\star|$.

Constraints Explanation: C1 and C2 ensure either a client is selected or not, eliminating any status in-between. C3 ensures that $b_k^t \in [b_{\min}, 1)$ for each selected client $k \in \mathcal{K}_t'$ while $b_k^t = 0$ for unselected client $k \in \mathcal{K} \setminus \mathcal{K}_t'$. This is crucial for maintaining system stability. C4 ensures that all available bandwidth B is allocated to only selected clients $k \in \mathcal{K}_t'$.

Bandwidth Allocation Adjustment for Selected Clients: MOBILE takes into account the practical aspects of bandwidth allocation. The objective (7.14) naturally allocates more resources to more favorable clients—those with higher returns and reliable history. For clients with better coverage (closer to BS), less bandwidth is needed to transfer the same amount of model parameters/gradients within limit $\mathcal{T}$. Hence, MOBILE introduces a return weight-based adjustment on optimized allocations for the selected clients only. Given the optimized allocation $\{b_k^t : k \in \mathcal{K}_t'\}$, we adjust b_k^t for a selected client k:

$$b_k^{\prime,t} = \frac{\frac{1}{r_k^t} b_k^t}{\sum_{j \in \mathcal{K}_t'} \frac{1}{r_j^t}}, \tag{7.19}$$

where clients with lower returns (i.e., farther from BS and with poorer signal coverage) receive a relatively larger share of bandwidth. The normalization term $\sum_{j \in \mathcal{K}_t'} \frac{1}{r_j^t}$ maintains the total bandwidth allocation consistent redistributing in a way that compensates clients with weaker signals. This adjustment strategy mitigates the issue of inefficient bandwidth w.r.t clients' actual data transmission needs.

7.4.3 Distance-based Baseline & MOBILE Algorithm

The distance-based baseline: Before elaborating on the MOBILE algorithm in MEC-based FL, we first describe an intuitive mobility-based baseline in client selection. Departing from the mobility-agnostic random client selection in e.g., traditional FL (McMahan et al., 2017), i.e., selecting $\lceil \max(1, xK) \rceil$ clients with $x \sim U(0, 1)$, a baseline for client selection takes into account the ranking of current client distances from the BS. That is, at the start of each round t, one can assume stationary clients for the *duration* of this round. Under this assumption, the top $n < K$ closest clients to BS are evenly allocated $\frac{B}{n}$ bandwidth. The distance-based baseline is provided Algorithm 8.

MOBILE Cold-Start Phase: If the historical distances of clients are available, MOBILE is applied directly. Otherwise, in case of a *cold start*, MOBILE requires data to build the Q matrix in initial rounds. We devise a strategy that initially defaults to a baseline approach and then gradually integrates MOBILE as more client distance data from BS become available. By setting up a threshold $0 < W^\star \leq W$ to determine this transition point, the cold-start strategy is as follows: (i) For rounds t with $t < W^\star$, client selection and bandwidth allocation are performed using the distance-based baseline in

Algorithm 8 Distance-based Baseline

```
1: Input: $K$ clients; $T$ rounds; bandwidth $B$
2: Output: Global model $w^T$
3: Initialize $\{w_k^0\}_{k=1}^K$
4: for round $t = 1$ to $T$ do
5:     Calculate distances $\{d_k^t\}$ from BS
6:     Select the top-$n$ closest clients to BS forming $\mathcal{K}_t'$
7:     Allocate bandwidth $b_k^t = \frac{1}{n}$ to each client $k \in \mathcal{K}_t'$
8:     BS distributes $w^t$ to clients in $\mathcal{K}_t'$
9:     for each client $k \in \mathcal{K}_t'$ in parallel do
10:        Optimize $w_k^t$ and upload $w_k^t$ to BS with bandwidth $b_k^t B$
11:    end for
12:    BS receives models $\{w_k^t\}$ from successful clients in $\mathcal{S}_t \subseteq \mathcal{K}_t'$ within limit $\mathcal{T}$ and
       aggregates models w.r.t. (7.2) or (7.3)
13: end for
14: return $w^T$
```

Algorithm 8. (ii) For rounds t, $W^\star \le t < W$, we optimize (7.14) using $|\mathcal{W}_t| = t$, reflecting an increasing amount of historical data. (iii) Finally, when $t \ge W$, MOBILE operates fully with the optimization process. This staged strategy ensures a smooth transition from the distance-based baseline to full optimization as mobility data accumulate over time. We provide the MOBILE process in Algorithm 9.

7.5 Experimental Evaluation

7.5.1 Mobility Dataset

We use the YJMob100K-Dataset1 (Yabe et al., 2024) dataset, which contains human mobility records collected over 75 days in a mid-sized and highly populated metropolitan area in Japan. The dataset spans a 200 by 200 grid, where each cell represents a 0.25km^2 area. The records are discretized into 30-minute intervals, resulting in 48 intervals per day. Each record consists of: *uid* (user identifier), d (day index; ranging from 0 to 74), t (time interval; ranging from 0 to 47), x (x-coordinate of the grid cell), and y (y-coordinate of the grid cell).

Data Preprocessing

Handling Missing Data: To address intermittent records, we identified users and periods in mostly dense populated areas. Continuous periods of 5 days were examined to find those with the highest record density. **User Selection**: Users with at least 75% of their records within the identified periods were selected, resulting in 856 users. **Consecutive Missing Value Filtering**: To ensure reliable interpolation, users with long

Algorithm 9 MOBILE framework for MEC-based FL

```
1: Input: $K$ clients, $T$ rounds, bandwidth $B$, window size $W$, cold-start threshold $W^\star$
2: Output: Global model $w^T$
3: Initialize: $\{w_k^0\}_{k=1}^K$, historical returns $\mathcal{W}_0 = \emptyset$
4: for round $t = 1$ to $T$ do
5:     if Cold-start and $t < W^\star$ then
6:         Invoke Baseline (Algorithm 8)
7:         Jump to Step 3
8:     end if
9:     Step 1: Mobility Representation
10:    Calculate return $\mathbf{r}_t$ and append to $\mathcal{W}_t = \{\mathbf{r}_{t-W+1}, \ldots, \mathbf{r}_t\}$, where $|\mathcal{W}_t| = \min(W, t)$
11:    Construct $Q_t$ w.r.t. (7.12)
12:    Step 2: Optimization
13:    Solve regularized MIQP in Problem 9
14:    Get optimal selection $\mathbf{a}_t$ and bandwidth allocation $\mathbf{b}_t$
15:    Adjust return-weight bandwidth $b_k'^{,t}$, $k \in \mathcal{K}_t'$ w.r.t. (7.19)
16:    Step 3: Federated Learning Process
17:    Distribute the global model $w_t$ to selected clients $\mathcal{K}_t'$
18:    for each client $k \in \mathcal{K}_t'$ in parallel do
19:        Optimize and upload $w_k^t$ to BS with bandwidth $b_k'^{,t} B$
20:    end for
21:    BS receives models $\{w_k^t\}$ from successful clients in $\mathcal{S}_t \subseteq \mathcal{K}_t'$ within limit $\mathcal{T}$ and
       aggregates models w.r.t. (7.2) or (7.3)
22: end for
23: return $w^T$
```

consecutive missing values were excluded while only ones with consecutive missing values below a threshold were retained. **Interpolation**: Missing values for the remaining users were imputed with linear interpolation.

Top-100 Users Selection

To identify the optimal location for the BS (server) for FL and ensure maximal coverage, we evaluated various potential BS locations within the grid. The location $(x, y) = (137, 78)$ with a coverage radius $\delta = 20$ grid units (equivalent to 10 km) were found to provide the best coverage, encompassing 12,393 data points (see Fig. 7.1). The proportion of each user's trajectory covered by this BS was calculated. Users were ranked based on the percentage of their trajectory within the BS's coverage radius. The top 100 users with the highest coverage proportion were selected for further analysis. This selection process ensured that these users had the most reliable and consistent mobility patterns within the BS's coverage area, making them ideal candidates for investigating mobility, client selection, and bandwidth allocation in the FL process. Figure 7.1 demonstrates the locations during the period of interest of all clients remained after the consecutive missing value filtering detailed in Section 7.5.1, with the locations belonging to the top-100 clients marked with 'red' and the rest with 'blue' dots.

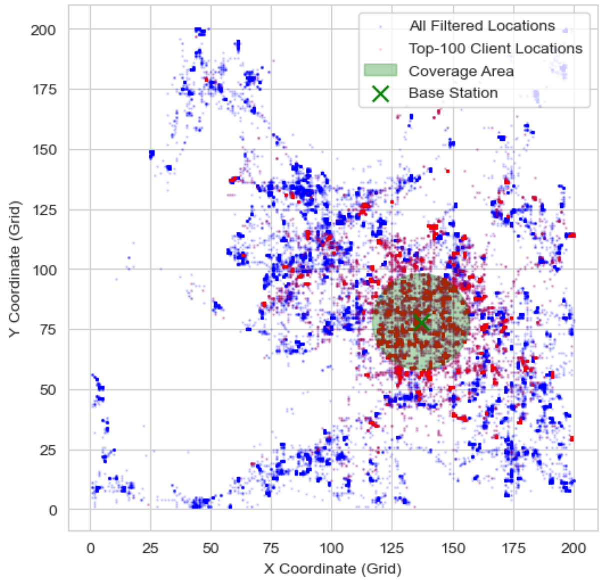


Figure 7.1: Locations of trajectories of filtered clients; red dots refer to top-100 clients, blue dots refer to rest of clients; BS's coverage area is shown in green.

7.5.2 Dataset for FL Models Training

Our experiments were conducted on three benchmark datasets: HAM10000 (Tschandl et al., 2018), CIFAR10, and CIFAR100 (Krizhevsky & Hinton, 2009), which offer a diverse set of image classification tasks. HAM10000 contains medical images for skin lesion classification with $|\mathcal{Y}| = 7$ classes. CIFAR10 and CIFAR100 contain natural images with $|\mathcal{Y}| = 10$ and $|\mathcal{Y}| = 100$ classes, respectively. To validate the robustness and versatility of our proposed methods across different DL model architectures and problem domains, we employed the following model configurations: AlexNet (Krizhevsky et al., 2012) for HAM10000, due to its effectiveness in handling medical images; ResNet18 (He et al., 2016) for CIFAR10, known for its deep residual connections that perform well on standard image datasets; and VGG11 (Simonyan & Zisserman, 2015) for CIFAR100, which provides a deeper network to tackle the increased complexity of the dataset. Such DL model choices ensure a comprehensive evaluation across various model depths, architectures, and dataset complexities, thereby demonstrating the general applicability of MOBILE.

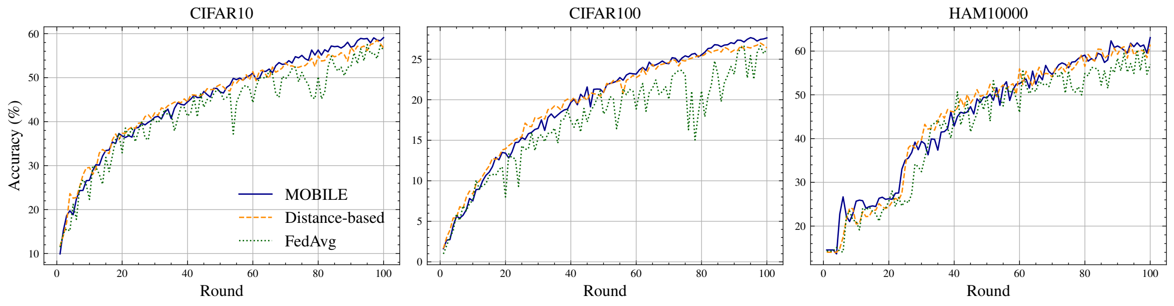


Figure 7.2: Comparison of test accuracy across 100 communication rounds on three datasets: CIFAR10, CIFAR100, and HAM10000. The performance of our proposed MOBILE algorithm (solid line) is compared against two baselines: the Distance-based method (dashed line) and Naive FedAvg (dotted line).

7.5.3 Performance Metrics

We perform 10-class, 100-class, and 7-class image classification tasks (CIFAR10, CIFAR100 and HAM10000) with the final global FL model w^T. We report the test accuracy of w^T on the test data of each client (the test data were not involved in any training process). We introduce the following metrics:

Number of Successful Clients: This is the actual number of clients that are successful in finishing both local optimization and model uploading corresponding to $|\mathcal{S}_t|$. We report this as the average *number of successful clients* and average *successful rate*, respectively:

$$\frac{1}{T}\sum_{t=1}^{T}|\mathcal{S}_t| \text{ and } \frac{1}{T}\sum_{t=1}^{T}\frac{|\mathcal{S}_t|}{|\mathcal{K}'_t|}. \tag{7.20}$$

End-of-Round Distance of Selected Clients: At every round clients move freely. For a client k, the *end-of-round* t distance from BS is used as the *start-of-round* $(t+1)$ distance from BS, $d_{k,t+1}$. The $d_{k,t+1}$ is used to approximate client k's distance from BS when uploading the updated model/gradients. Note, $d_{k,t+1}$ is purely related to client's mobility and apparently is unknown to MOBILE for client selection and resource allocation at the beginning of round t. MOBILE requires to determine whether clients are able to send model updates back to BS or not. A smaller end-of-round t distance $d_{k,t+1}$ for a selected client k reflects higher likelihood for this client to be re-selected at $t+1$. We report the average *end-of-round distance of selected clients* at t:

$$\frac{1}{|\mathcal{K}'_t|}\sum_{k\in\mathcal{K}'_t} d_{k,t+1} \tag{7.21}$$

Proportion of Wasted Throughput: In an ideal case, we allocate all available bandwidth B across all selected clients $\mathcal{K}'_t$ to be able to finish their models uploading up to/before limit $\mathcal{T}$, thereby maximizing the utilization of bandwidth. However, since end-of-round distance $d_{k,t+1}$ of selected clients is unknown at the time of bandwidth allocation, inevitably a fraction of allocated bandwidth will be wasted, i.e., $B^- > 0$ where (1) clients are out of BS coverage thus cannot start uploading; (2) clients that begin uploading, fail to complete within $\mathcal{T}$). We define the *throughput usage* as:

$$\psi = \sum_{t=1}^{T} \sum_{k \in \mathcal{K}'_t} b_k^t \tau_k^t, \tag{7.22}$$

where τ_k^t is duration within $\mathcal{T}$ where the allocated bandwidth is *not* utilized by client k. τ_k^t is defined into two cases: (i) **Selected & Finished Client**, i.e., client $k \in \mathcal{S}_t$. Client k successfully completes uploading before $\mathcal{T}$, while unused time τ_k^t is calculated as remaining time *after* the upload finishes:

$$\tau_k^t = \max(0, \mathcal{T} - \frac{|w_k^t|}{r_k^t}). \tag{7.23}$$

$|w_k^t|$ is the data size of the model update for client k, and r_k^t is the uplink rate of client k at round t.

(ii) **Selected but Failed Client**, i.e., client $k \in \mathcal{K}'_t \setminus \mathcal{S}_t$. Client k either starts but fails to finish uploading within $\mathcal{T}$ or cannot start due to being out of coverage. For this client, we consider τ_k^t as the entire time $\mathcal{T}$ since none of the allocated bandwidth contributes to successful transmission: $\tau_k^t = \mathcal{T}$.

We measure the *overall proportion of wasted throughput* across all rounds ψ' and the *proportion of throughput wasted by failed clients* ψ'_F, respectively, as:

$$\psi' = \frac{\sum_{t=1}^{T} \sum_{k \in \mathcal{K}'_t} b_k^t \tau_k^t}{\sum_{t=1}^{T} B\mathcal{T}} \text{ and } \psi'_F = \frac{\sum_{t=1}^{T} \sum_{k \in \mathcal{K}'_t \setminus \mathcal{S}_t} b_k^t \mathcal{T}}{\sum_{t=1}^{T} B\mathcal{T}}. \tag{7.24}$$

ψ'_F captures the fraction of the total allocated bandwidth that was wasted due to clients failing to complete their uploads. The difference $\psi' - \psi'_F$ indicates the throughput wasted by allocating superfluous bandwidth to successful clients in $\mathcal{S}_t$.

Table 7.1: Final test accuracy (%) over all datasets and methods.

Method	**CIFAR10**	**CIFAR100**	**HAM10000**
FedAvg	56.65±0.69	26.35±0.31	59.13±1.22
Distance-based	57.76±0.30	26.75±0.13	61.08±0.40
MOBILE	**58.97**±0.11	**27.57**±0.08	**62.11**±0.57
MOBILE-CS	58.77±0.15	27.54±0.10	62.11±0.58

7.5.4 System Configuration

We use FedAvg as a mobility-agnostic FL baseline to contrast the impact of mobility. FedAvg randomly selects $n = \lceil \max(1, xK) \rceil$ clients with $x \sim U(0, 1)$ at every round t irrespective of locations and mobility patterns. Subsequently, bandwidth B is evenly allocated $b_k^t = \frac{1}{n}, k \in \mathcal{K}_t'$ over the selected clients. The maximum expected number of successful clients $\max_t \mathbb{E}[|\mathcal{S}_t|]$ in (7.20) was achieved with $n = 10$. This value was used in all experiments using FedAvg. While for the distance-based baseline, $n = 12$. MOBILE's regularization parameter tuning was systematically conducted under equivalent conditions to those for FedAvg and distance-based baselines leading $\lambda = 100$ and $\gamma = 10^{-4}$. An optimal Q was found with window size $W = 10$ over $T = 100$ rounds. A small W compromises predictability, whereas a large W infuses the model with 'stale' mobility data. $\mathcal{T}$ is set to 40% of the duration between rounds maximizing the probability a client completes its task. Equation (7.14) is solved using SCIP and ECOS in CVXPY (Diamond & Boyd, 2016). For neural network training and hyperparameter tuning, we followed (Long et al., 2025).

7.5.5 Performance Analysis

Overall, our experiments across three datasets showcase that MOBILE consistently demonstrated superior performance compared to other methods. As summarized in Table 7.1, MOBILE achieved the highest final test accuracy across all datasets. For CIFAR10, MOBILE outperformed FedAvg by 2.3% reaching final accuracy of 58.97%. On CIFAR100, which poses challenging classification tasks due to large number of classes, MOBILE maintained a significant lead over the other methods. It achieved an accuracy of 27.57%, surpassing FedAvg by 1.22%. This highlights the robustness of MOBILE in scenarios involving higher data complexity. For HAM10000, MOBILE's performance gain was even more pronounced, with an improvement of about 3% over FedAvg. The distance-based approach showed moderate improvement over FedAvg, particularly on HAM10000 (achieving 61.08%). However, it fell short compared to MOBILE, highlighting the importance of optimized client selection that accounts for mobility dynamics.

In Fig. 7.2, the fluctuations observed in accuracy curve of FedAvg are attributed to biased updates caused by the limited number of successful clients at each round. Unlike MOBILE, which ensures a more consistent and diverse set of client contributions, FedAvg's reliance on randomly selected and often unreliable clients results in instability in global model's updates. Such inconsistency is exacerbated by a small pool of clients whose models are successfully uploaded, leading to significant variability in the aggregated updates.

Cold-Start Phase

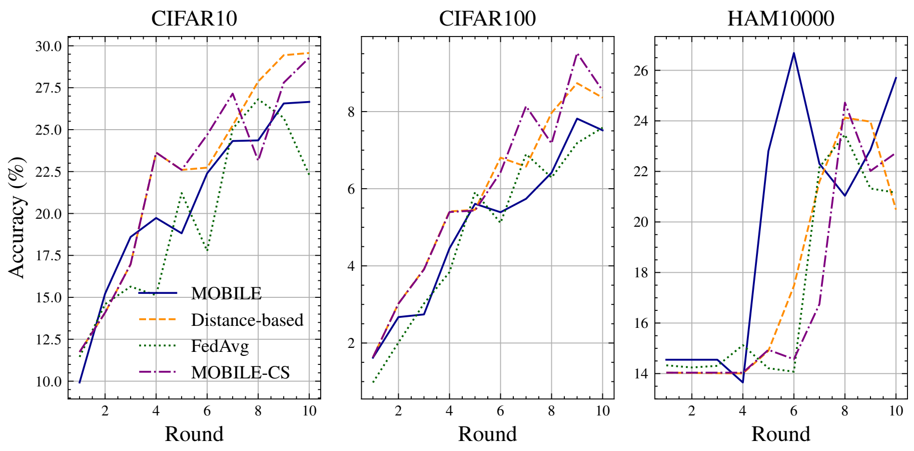


Figure 7.3: Test accuracy comparison over the first 10 rounds of training where potential cold-start of MOBILE takes place on CIFAR10, CIFAR100, and HAM10000 datasets using different methods.

A cold-start phase is necessary in cases of lack of historical mobility data required to construct Q; this MOBILE variant is referred to as MOBILE-CS. Since the optimal window size W is relatively small (as detailed in Section 7.5.4), the cold-start phase is expected to be brief and have minimal impact on the final global model accuracy. We set the transition threshold $W^\star = 5$ such that during the first 5 rounds, MOBILE-CS operates identically to the distance-based baseline. From rounds $W^\star$ to W, MOBILE-CS behaves like MOBILE with an increasing history length, $|\mathcal{W}_t| = t$. Figure 7.3 illustrates the test accuracy curve during the first 10 rounds of cold-start phase. The performance of MOBILE-CS overlaps with the distance-based baseline during the initial 5 rounds and diverges between rounds 6 and 10. Despite the divergence, MOBILE-CS remains closely aligned with MOBILE throughout the cold-start phase, reflecting its ability to quickly adapt as mobility data becomes available.

As presented in Table 7.1, MOBILE-CS achieved final test accuracy 58.77% and 27.54% for CIFAR10 and CIFAR100, respectively, which are slightly lower than those of MOBILE without the cold-start phase. Nonetheless, the impact of the cold-start phase is minimal, which is further supported by the results on HAM10000, where MOBILE-CS achieved the same final accuracy of 62.11% as MOBILE. This showcases MOBILE robustness even in initial lack of mobility history.

Quality of Selected Clients

To assess the client selection efficacy, we report on the average number of successful clients per round, average successful rate, and average end-of-round distance per method linked to the original Problem 7. The results showcase that MOBILE and

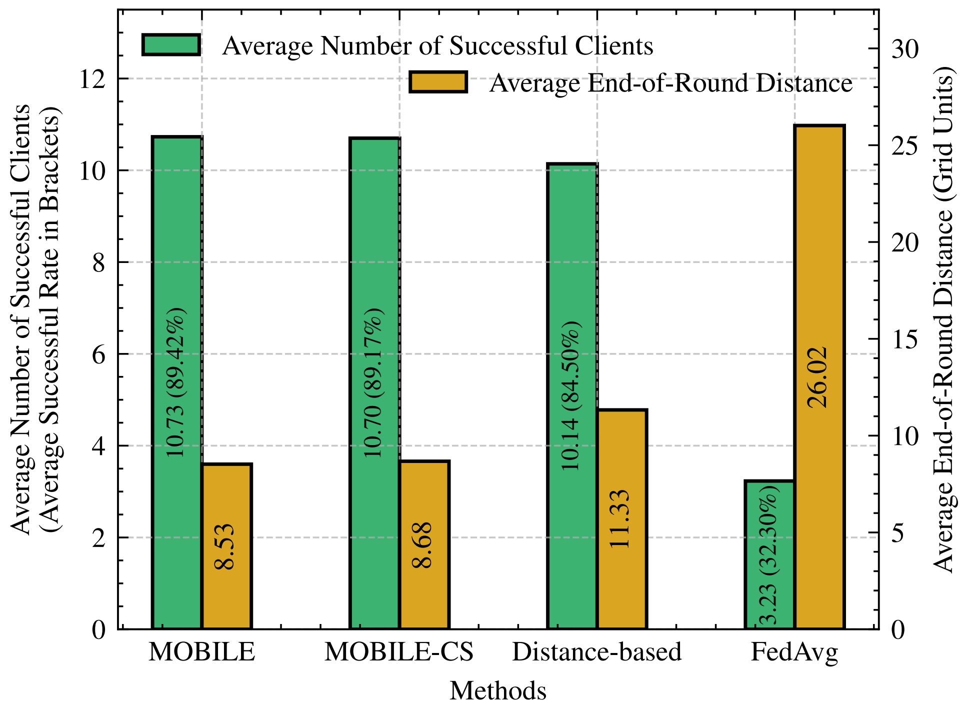


Figure 7.4: Average number of successful clients with average successful rates (% in brackets), and average end-of-round distances from BS for all methods.

MOBILE-CS outperform other methods achieving approximately 10.73 and 10.7 successful clients on average, with corresponding successful rates 89.42% and 89.17% and end-of-round distances of 8.53 and 8.68 grid units from the BS, respectively. The distance-based method follows with 10.14 successful clients (84.50% successful rate) and an average distance of 11.33 grid units. FedAvg significantly underperforms with only 3.23 successful clients on average (32.30% successful rate) and average distance of 26.02 grid units.

These findings highlight the clients selectivity capacity of mobility-aware MOBILE & MOBILE-CS compared to FedAvg client selection strategy. MOBILE not only finds the most appropriate clients to engage in FL process per round, but also obtains better accuracy than FedAvg and distance-based methods by selecting the best *number* of clients along with the *most promising ones* among them that finish their learning task per round. Therefore, we need to assess the *quality* of the most promising selected clients based on minimizing the wasted bandwidth.

Bandwidth Utilization & Throughput

Analysis on wasted throughput highlights the resource utilization efficiency of each method. MOBILE and MOBILE-CS exhibit the lowest overall wasted throughput with 30.65% and 30.93% of the total throughput, respectively, of which 10.37% and 10.62% are attributed to failed clients. The distance-based method incurs higher wastage at 41.47%

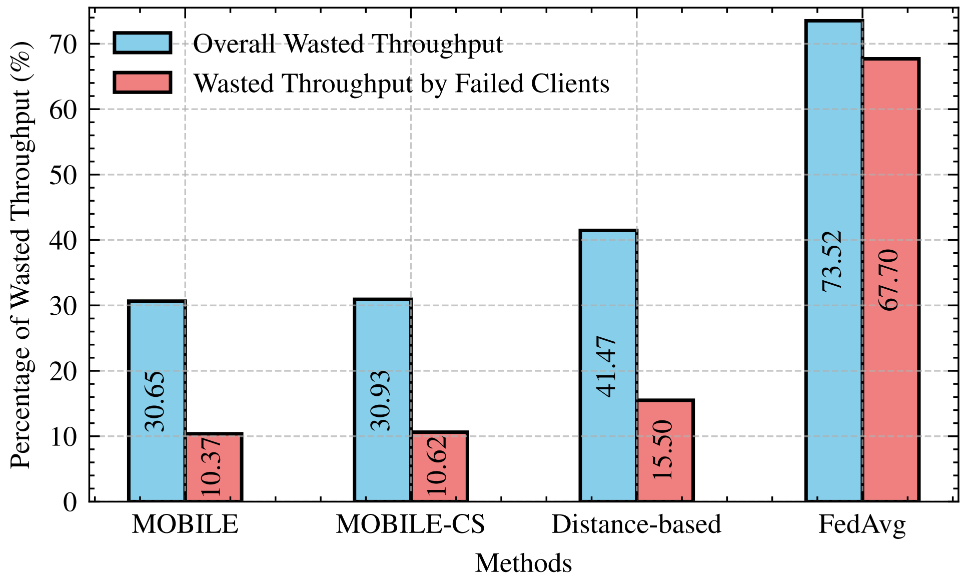


Figure 7.5: % of wasted throughput across methods; blue bars represent wasted throughput while coral bars denote the wasted throughput caused by failed clients. The values within the bars indicate the exact percentages, normalized against the total throughput to ensure comparability between methods.

overall, i.e., 35.30% more wasted bandwidth compared to MOBILE, while including 15.50% due to failed clients, i.e., 49.46% more wasted bandwidth from the failed clients compared to MOBILE. Moreover, FedAvg demonstrates the poorest performance with 73.52% of throughput wasted overall, and a substantial 67.70% arising from failed clients; corresponding to 139.87% more bandwidth wasted compared to MOBILE.

From clients' perspective, the average *failure rate*, defined as $1-$average *successful rate* in (7.20) represents the proportion of selected clients which despite committing their resources to locally train the model in light of contributing to the FL process, fail to upload their models to BS. Such failures not only lead to wasted resources on the clients but also discourage their future participation in FL process; e.g., their incentives to contribute are undermined. Fig. 7.6 shows the superiority of MOBILE & MOBILE-CS in achieving higher accuracy while keeping the failure rate at a much lower level. This highlights the robustness of MOBILE in maintaining model performance while respecting clients' contributions, thus fostering a reliable and client-centric FL process.

Overall, our results underscore the effectiveness of our mobility-aware FL methods in (i) minimizing throughput wastage compared to FL and distance-based approaches, while achieving optimized clients selection ensuring (ii) higher accuracy and (iii) identifying the most promising clients that successfully deliver their learning tasks.

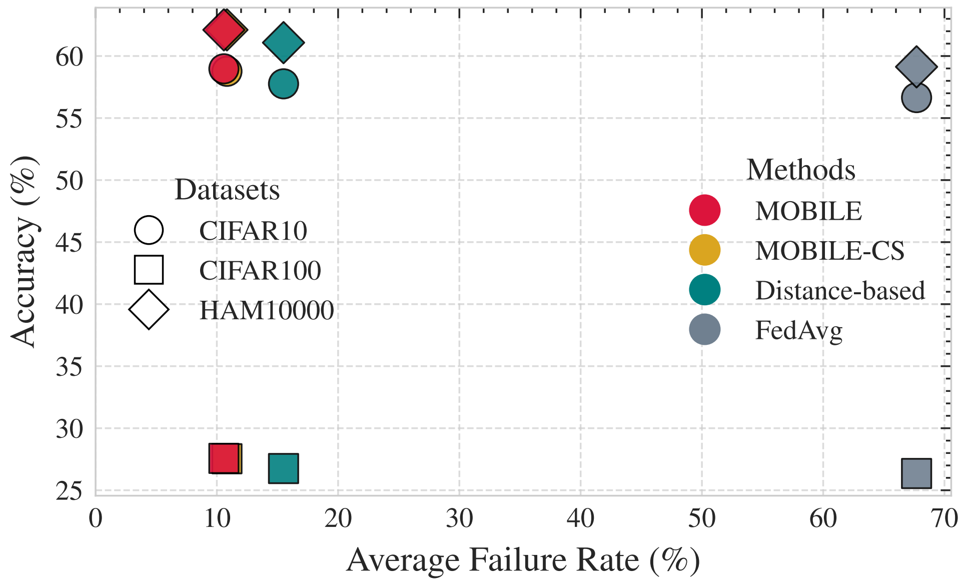


Figure 7.6: Scatter plot showing the relationship between the average failure rate (%) and the accuracy (%) achieved across datasets for each method.

7.5.6 Heterogeneous & non-i.i.d. Data

In cases with non-i.i.d client data distributions, a high number of successful clients $|\mathcal{S}_t|$ does not necessarily imply improved learning. The inclusion of additional clients in the aggregation can exacerbate weight divergence, causing the global model not to generalize well (Li et al., 2020; Zhao et al., 2018). MOBILE adopts a modification inspired by (Tang et al., 2022a) to address this challenge, which not only improves its capacity to leverage a larger $|\mathcal{S}_t|$ but also ensures consistently superior results w.r.t. accuracy. After each client $k \in \mathcal{S}_t$ receives the global model $\mathbf{w}_t$ from BS, it performs local evaluation on its test dataset $\mathcal{D}_k^{\text{test}}$ before proceeding with local optimization. Client k computes the loss $\ell_k = \mathcal{L}_{loss}(\mathbf{w}_t; \mathcal{D}_k^{\text{test}})$ on $\mathcal{D}_k^{\text{test}}$ and transmits the scalar ℓ_k back to BS with negligible communication overhead.

The BS, upon receiving $\{\ell_k : k \in \mathcal{S}_t\}$, ranks the clients in $\mathcal{S}_t$ w.r.t. losses obtaining an ordered set of clients $\mathcal{S}_t^{\text{ranked}} = \{k_1, k_2, \ldots, k_{|\mathcal{S}_t|}\}$, where $\ell_{k_1} \leq \ell_{k_2} \leq \cdots \leq \ell_{k_{|\mathcal{S}_t|}}$. The BS selects the top-l clients $\mathcal{S}_t^{\text{top}} = \{k_1, k_2, \ldots, k_l\}$ for updating the global model using (7.2) or (7.3). We conducted experiments across all datasets for MOBILE, distance-based, and FedAvg under non-i.i.d data. Each client's data were partitioned using Dirichlet distribution ($\alpha = 0.5$) for creating non-i.i.d. distributions in FL.

Table 7.2 shows the test accuracy while Fig. 7.7 shows the methods performance on CIFAR10 under non-i.i.d data (similar results are obtained with the rest of datasets; not shown due to space limitations). MOBILE achieves the highest accuracy in $T = 100$ rounds with 4.04% improvement over FedAvg, demonstrating effectiveness in leveraging client mobility under non-i.i.d data. The distance-based method also incorporates

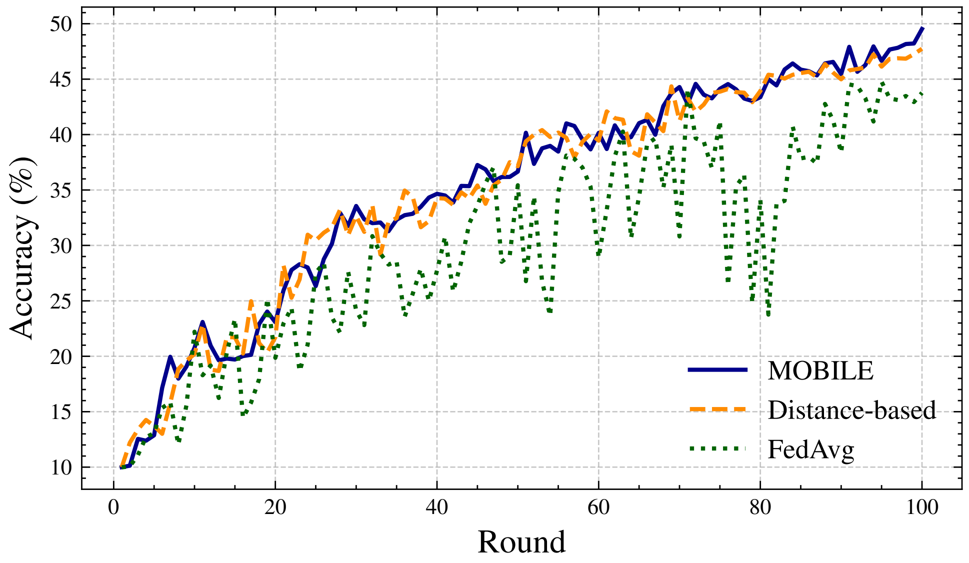


Figure 7.7: Comparison of test accuracy across $T = 100$ rounds on CIFAR10 under non-i.i.d data (Dirichlet $\alpha = 0.5$).

the top-l adjustment (not applicable to FedAvg due to small $|\mathcal{S}_t|$), reducing the disproportionate influence of almost 'stationary' clients near BS. These clients, if repeatedly included, bias the global model under non-i.i.d data.

Aggregating updates from top-l clients, MOBILE and distance-based methods ensure more generalized global model. In Table 7.2, MOBILE consistently achieves superior accuracy: 23.18% on CIFAR100 (0.93% over FedAvg) and 60.69% on HAM10000 (2.09% over FedAvg). This indicates MOBILE's robustness in weight divergence and ability to leverage $|\mathcal{S}_t|$ over non-i.i.d data.

Table 7.2: Final test accuracy (%) on non-i.i.d data.

Method	**CIFAR10**	**CIFAR100**	**HAM10000**
FedAvg	44.31±0.34	22.25±0.24	58.60±0.58
Distance-based	47.19±0.32	22.92±0.16	60.54±0.37
MOBILE	**48.35**±0.20	**23.18**±0.17	**60.69**±0.32
#top-l clients	8	9	8

7.6 Conclusions

We proposed MOBILE, a novel framework that leverages MIQP for client selection and bandwidth allocation in FL environments with mobile clients. Comprehensive evaluations demonstrate the superiority of MOBILE over both the distance-based baseline and traditional FedAvg approaches. MOBILE consistently achieves the highest percentage of successful clients while maintaining the lowest end-of-round distances from BS, indicating better client retention and efficient resource utilization. Notably, MOBILE significantly reduced wasted throughput to just 30.65%, compared to the concerning 73.52% observed in FedAvg, emphasizing its efficiency in MEC-based FL environments.

The results across multiple datasets with varying complexity (CIFAR10, CIFAR100, and HAM10000) confirm that our approach's effectiveness is not limited to specific data types or model architectures. Furthermore, our cold-start variant, MOBILE-CS, demonstrates that even with limited historical mobility data, the framework can quickly adapt and achieve performance comparable to the full implementation.

For non-i.i.d. data scenarios, which are common in real-world federated learning deployments, MOBILE's modified approach of selecting top-performing clients based on loss metrics proves highly effective. This modification ensures that MOBILE maintains its advantage even when faced with the additional challenge of heterogeneous data distributions across clients.

In future work, we plan to extend MOBILE to study client selection based on clients' heterogeneity in terms of computational capabilities and data characteristics, potentially leading to even more resilient FL systems. Additionally, exploring the integration of more sophisticated mobility prediction models could further enhance the effectiveness of client selection strategies in dynamic mobile environments.

Chapter 8

Conclusion and Future Work

8.1 Summary of Contributions

This thesis has advanced the field of distributed edge learning by addressing key challenges in resilience, adaptability, data efficiency, communication optimization, and mobility awareness. The frameworks and methodologies presented provide a comprehensive approach to making distributed learning more robust, efficient, and effective in dynamic environments.

The main contributions of this thesis can be summarized as follows:

8.1.1 Resilient Edge Predictive Analytics

In Chapter 3, we presented a comprehensive framework for enhancing the resilience of predictive analytics in edge computing environments. By leveraging statistical signatures of nodes' data, we developed strategies to identify surrogate nodes capable of providing uninterrupted services during node failures. This approach eliminates the need for extensive data replication or transfer to the cloud, making it particularly suitable for resource-constrained edge environments. Our experimental results demonstrated that the proposed framework maintains and, in some cases, improves predictive performance during node failures, significantly outperforming traditional centralized approaches.

The key innovation lies in the dynamic identification of optimal surrogate nodes and the enhancement of local models through selective information exchange. This not only ensures continuity of service but also provides an efficient mechanism for load balancing during failures.

8.1.2 Model Maintenance under Concept Drift

Chapter 4 extended our resilience framework to address the challenge of concept drift in distributed edge learning environments. We investigated how concept drifts of different magnitudes affect the performance of enhanced models and developed strategies for maintaining model accuracy as data distributions evolve.

Our maintenance strategies demonstrated different trade-offs between the amount of data needed to be transmitted and the performance of the maintained models. Through comprehensive experimentation on both synthetic and real datasets, we showed that our approach can effectively maintain the performance of enhanced models even as the underlying data distributions change over time.

This contribution is particularly significant in dynamic environments where data characteristics evolve continuously, ensuring that distributed edge learning systems remain effective over extended periods without requiring complete retraining or massive data transfers.

8.1.3 Query-Driven Predictive Analytics

In Chapter 5, we introduced a novel query-driven data-relevance approach for predictive analytics. Rather than building models on entire datasets, our framework progressively discovers relevant data regions based on query patterns and uses these regions to refine predictive models. This leads to more accurate and resource-efficient models tailored to the actual patterns of data access.

The proposed approach includes mechanisms for determining the optimal learning horizon based on Optimal Stopping Theory, as well as techniques for adapting to changing data and query distributions. Our experimental evaluation demonstrated significant improvements in prediction accuracy compared to both traditional approaches and state-of-the-art query-driven methods.

By focusing computational resources on the most relevant data, this framework enables more efficient use of limited resources in edge environments while maintaining high predictive performance.

8.1.4 Dynamic Aggregation and Decentralized Personalized Federated Learning

Chapter 6 presented DA-DPFL, a novel framework for decentralized personalized federated learning that addresses the challenges of communication cost, training efficiency, and data heterogeneity. By introducing dynamic aggregation mechanisms that allow clients to reuse trained models within the same communication round, and a sparse-to-sparser pruning strategy that progressively increases model sparsity, our approach achieves significant reductions in both communication and computational costs.

Our extensive experiments demonstrated that DA-DPFL consistently outperforms both centralized and decentralized federated learning baselines across various datasets and model architectures, while reducing energy consumption by up to 5×. The framework's ability to maintain high performance under diverse network topologies and extreme data heterogeneity further highlights its robustness and practical applicability.

8.1.5 Mobility-Aware Federated Learning

In Chapter 7, we introduced MOBILE, a framework for optimizing federated learning in mobile computing environments. By considering both current and historical mobility patterns of clients, MOBILE optimizes client selection and bandwidth allocation to maximize the number of successful client interactions in each training round. This approach is formulated as a regularized Mixed-Integer Quadratic Programming problem that balances the expected returns based on client proximity with the correlations in historical mobility patterns.

Our experimental evaluation using real-world mobility data demonstrated that MOBILE consistently achieves higher accuracy and lower resource wastage compared to mobility-agnostic approaches. The framework's ability to maintain performance even under cold-start conditions and with non-i.i.d. data distributions further underscores its practical utility in mobile edge computing scenarios.

8.2 Integration of Contributions

While each chapter of this thesis addresses specific challenges in distributed edge learning, there are important synergies between the contributions that create a cohesive framework for resilient and adaptive learning in dynamic environments.

The resilience mechanisms introduced in Chapter 3 and extended in Chapter 4 provide a foundation for ensuring service continuity and model accuracy in the face of node failures and concept drift. These mechanisms can be combined with the query-driven approach from Chapter 5 to create systems that are not only resilient to failures but also efficiently focus computational resources on the most relevant data regions.

Similarly, the dynamic aggregation and pruning techniques from Chapter 6 can be integrated with the mobility-aware client selection from Chapter 7 to create federated learning systems that are both communication-efficient and robust to client mobility. By combining these approaches, edge computing systems can achieve higher levels of resilience, efficiency, and adaptability than would be possible with any single technique.

Together, these contributions advance the state-of-the-art in distributed edge learning, enabling more robust, efficient, and adaptive systems that can operate effectively in dynamic and uncertain environments.

8.3 Limitations and Future Work

While this thesis presents significant advances in distributed edge learning, several limitations and opportunities for future work remain:

8.3.1 Enhanced Model Generalization

The enhanced models proposed in Chapters 3 and 4 demonstrate good performance across different datasets and scenarios, but their generalization capabilities could be further improved. Future work could explore more sophisticated techniques for information extraction and model enhancement, potentially leveraging advances in transfer learning and domain adaptation to create models that can better generalize across different data distributions.

8.3.2 Scalability to Larger Networks

The evaluation of our frameworks was conducted on networks with relatively modest numbers of nodes or clients. Further investigation is needed to understand how these approaches scale to much larger networks with thousands or millions of nodes. This includes developing more efficient algorithms for surrogate node selection, relevant data discovery, and client selection that can operate at scale.

8.3.3 Integration with Deep Learning

While our frameworks are largely model-agnostic, some of the specific implementations and evaluations focused primarily on traditional machine learning models. Extending these approaches to deep learning models presents additional challenges related to model size, computational requirements, and optimization strategies. Future work could explore adaptations of our frameworks specifically tailored to deep neural networks.

8.3.4 Privacy and Security Considerations

The sharing of model information and statistical signatures between nodes, while more privacy-preserving than raw data sharing, still raises privacy and security concerns. Future research could incorporate differential privacy, secure multi-party computation, or other privacy-enhancing technologies to further protect sensitive information while maintaining the benefits of our distributed learning approaches.

8.3.5 Adaptive Parameter Selection

Many of the frameworks presented in this thesis include parameters that control various aspects of their behavior, such as the relevance parameter in query-driven learning or the pruning threshold in DA-DPFL. Developing methods for automatically adapting these parameters based on system conditions and performance metrics would enhance the practical applicability of these frameworks.

8.3.6 Theoretical Analysis

While theoretical foundations were provided for several aspects of our work, further theoretical analysis could strengthen the understanding of the proposed frameworks. This includes developing tighter bounds on performance guarantees, analyzing convergence properties under various conditions, and establishing theoretical connections between different components of our frameworks.

8.4 Closing Remarks

The rapid proliferation of edge computing and distributed machine learning has created new opportunities for real-time, context-aware intelligent services. However, realizing these opportunities requires addressing significant challenges related to resilience, adaptability, resource efficiency, and mobility.

This thesis has presented a comprehensive set of frameworks and methodologies that tackle these challenges, enabling more robust, efficient, and effective distributed learning in dynamic edge environments. By enhancing resilience to node failures, adapting to concept drift, focusing on relevant data, optimizing communication and computation, and accounting for client mobility, these contributions provide a foundation for next-generation edge intelligence systems.

As edge computing continues to evolve and expand into new domains, the approaches developed in this thesis can help ensure that distributed learning systems remain reliable, efficient, and effective, even in the face of the inherent volatility and resource constraints of edge environments. This will enable a new generation of intelligent applications that can bring the benefits of machine learning closer to where data is generated, supporting real-time decision-making and context-aware services across a wide range of domains.

Publications

Journal Articles

[J3] Qianyu Long, **Qiyuan Wang**, Christos Anagnostopoulos, Daning Bi: *Decentralized Personalized Federated Learning based on a conditional 'sparse-to-sparser' scheme*. IEEE Transactions on Neural Networks and Learning Systems (2025). Early Online Publication. DOI: 10.1109/TNNLS.2024.3513297.
Available at: https://eprints.gla.ac.uk/358095/
IEEE Xplore: https://ieeexplore.ieee.org/document/11060892

[J2] Saleh ALFahad, **Qiyuan Wang**, Christos Anagnostopoulos, Kostas Kolomvatsos: *Task offloading in mobile edge computing using cost-based discounted optimal stopping*. Open Computer Science, 14(1):20230115 (2024). DOI: 10.1515/comp-2023-0115.
Available at: https://eprints.gla.ac.uk/323227/

[J1] **Qiyuan Wang**, Saleh ALFahad, Jordi Mateo-Fornés, Christos Anagnostopoulos, Kostas Kolomvatsos: *Resilient edge predictive analytics by enhancing local models*. Open Computer Science, 14(1):20230116 (2024). DOI: 10.1515/comp-2023-0116.

Conference Papers

[C4] **Qiyuan Wang**: *Bridging Mobility and Federated Learning: Toward Dynamic Client Orchestration in Distributed Edge Systems*. In: 45th IEEE International Conference on Distributed Computing Systems (ICDCS), Glasgow, UK, 20–23 July 2025.
Available at: https://eprints.gla.ac.uk/352341/

[C3] **Qiyuan Wang**, Qianyu Long, Christos Anagnostopoulos: *MOBILE: Mobility and Outage-Based Intelligent Federated Learning in Mobile Computing*. In: 28th European Conference on Artificial Intelligence (ECAI 2025), Bologna, Italy, 25–30 October 2025.
Available at: https://eprints.gla.ac.uk/359977/

[C2] **Qiyuan Wang**, Christos Anagnostopoulos, Jordi Mateo-Fornés, Kostas Kolomvatsos, Andreas Vrachimis: *Maintenance of Model Resilience in Distributed Edge Learning Environments*. In: 19th IEEE International Conference on Intelligent Environments

(IE'23), Mauritius, 27–30 June 2023, pp. 1–8. DOI: 10.1109/IE57519.2023.10179109.
Available at: https://eprints.gla.ac.uk/292826/

[C1] **Qiyuan Wang**, Jordi Mateo-Fornés, Christos Anagnostopoulos, Kostas Kolomvatsos: *Predictive Model Resilience in Edge Computing*. In: IEEE 8th World Forum on Internet of Things (WF-IoT2022), Yokohama, Japan, 26 October – 11 November 2022. DOI: 10.1109/WF-IoT54382.2022.10152282.
Available at: https://eprints.gla.ac.uk/278311/

[C6] Ke Xiao, **Qiyuan Wang**, Christos Anagnostopoulos, Kevin Bryson: *Resilient Inference for Personalized Federated Learning in Edge Computing Environments*. In: 45th IEEE International Conference on Distributed Computing Systems (ICDCS), Glasgow, UK, 20–23 July 2025.
Available at: https://eprints.gla.ac.uk/352270/

[C5] Zhuoran Tan, **Qiyuan Wang**, Christos Anagnostopoulos, Shameem P. Parambath, Jeremy Singer, Sam Temple: *Distributed Log-driven Anomaly Detection System Based on Evolving Decision Making*. In: 45th IEEE International Conference on Distributed Computing Systems (ICDCS), Glasgow, UK, 20–23 July 2025.
Available at: https://eprints.gla.ac.uk/352009/

Papers Under Review

[S3] **Qiyuan Wang**, Christos Anagnostopoulos: *Query-driven Predictive Analytics via Discovered Relevant Data: A Data-centric Learning Approach*. Submitted to Knowledge and Information Systems (KAIS).

[S2] Qianyu Long, **Qiyuan Wang**, Christos Anagnostopoulos: *FedPhD: Federated Pruning with Hierarchical Learning of Diffusion Models*. Submitted to IEEE Transactions on Cybernetics (TCYB).
Preprint available at: https://arxiv.org/abs/2507.06449

[S1] Ke Xiao, **Qiyuan Wang**, Christos Anagnostopoulos: *FLgym: Towards Robust and Byzantine-Resilient Federated Learning*. Submitted to IEEE Transactions on Information Forensics and Security (TIFS).

Bibliography

Abad, M. S. H., Ozfatura, E., Gunduz, D., & Ercetin, O. (2020). Hierarchical federated learning across heterogeneous cellular networks. *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8866–8870.

Agrawal, S., Chaudhuri, S., & Narasayya, V. R. (2000). Automated selection of materialized views and indexes in sql databases. *VLDB*, *2000*, 496–505.

Albert, S., & Goran, P. (2006). Optimal stopping and free-boundary problems. In *Optimal stopping and free-boundary problems* (pp. 123–142). Birkhäuser Basel. https://doi.org/10.1007/978-3-7643-7390-0_3

Alistarh, D., Grubic, D., Li, J., Tomioka, R., & Vojnovic, M. (2017). Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Advances in neural information processing systems*, *30*.

Anagnostopoulos, C. (2020). Edge-centric inferential modeling & analytics. *Journal of Network and Computer Applications*, *164*, 102696. https://doi.org/https://doi.org/10.1016/j.jnca.2020.102696

Anagnostopoulos, C., Savva, F., & Triantafillou, P. (2018). Scalable aggregation predictive analytics: A query-driven machine learning approach. *Applied Intelligence*, *48*, 2546–2567.

Anagnostopoulos, C., & Triantafillou, P. (2015). Learning to accurately count with query-driven predictive analytics. *2015 IEEE international conference on big data (big data)*, 14–23.

Anagnostopoulos, C., & Triantafillou, P. (2017). Query-driven learning for predictive analytics of data subspace cardinality. *ACM Trans. Knowl. Discov. Data*, *11*(4). https://doi.org/10.1145/3059177

Anik, A. I., & Bunt, A. (2021). Data-centric explanations: Explaining training data of machine learning systems to promote transparency. *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/3411764.3445736

Audu, A.-R. A., Cuzzocrea, A., Leung, C. K., MacLeod, K. A., Ohin, N. I., & Pulgar-Vidal, N. C. (2019). An intelligent predictive analytics system for transportation analytics on open data towards the development of a smart city. *Conference on Complex, Intelligent, and Software Intensive Systems*, 224–236.

Bach, S. H., & Maloof, M. A. (2008). Paired learners for concept drift. *2008 Eighth IEEE International Conference on Data Mining*, 23–32. https://doi.org/10.1109/ICDM.2008.119

Bai, T., Luo, J., Zhao, J., Wen, B., & Wang, Q. (2021). Recent advances in adversarial training for adversarial robustness. *CoRR*, *abs/2102.01356*. https://arxiv.org/abs/2102.01356

Bethel, E. W., Campbell, S., Dart, E., Stockinger, K., & Wu, K. (2006). Accelerating network traffic analytics using query-driven visualization. *2006 IEEE Symposium On Visual Analytics Science And Technology*, 115–122. https://doi.org/10.1109/VAST.2006.261437

Beutel, J., Römer, K., Ringwald, M., & Woehrle, M. (2009, October). Deployment techniques for sensor networks. https://doi.org/10.1007/978-3-642-01341-6_9

Bhat, G., Gumussoy, S., & Ogras, U. Y. (2019). Power and thermal analysis of commercial mobile platforms: Experiments and case studies. *DATE*, 144–149.

Bibikar, S., Vikalo, H., Wang, Z., & Chen, X. (2022). Federated dynamic sparse training: Computing less, communicating less, yet learning better. *Proceedings of the AAAI Conference on Artificial Intelligence*, *36*(6), 6080–6088.

Bifet, A., & Gavalda, R. (2007). Learning from time-changing data with adaptive windowing. *Proceedings of the 2007 SIAM international conference on data mining*, 443–448.

Blalock, D., Gonzalez Ortiz, J. J., Frankle, J., & Guttag, J. (2020). What is the state of neural network pruning? *Proceedings of machine learning and systems*, *2*, 129–146.

Blanchard, P., El Mhamdi, E. M., Guerraoui, R., & Stainer, J. (2017). Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in neural information processing systems*, *30*.

Bonomi, F., Milito, R., Zhu, J., & Addepalli, S. (2012). Fog computing and its role in the Internet of Things. *Proceedings of the 1st Edition of the MCC Workshop on Mobile Cloud Computing*, 13–16. https://doi.org/10.1145/2342509.2342513

Borg, A., Baumbach, J., & Glazer, S. (1983). A message system supporting fault tolerance. *ACM SIGOPS Operating Systems Review*, *17*(5), 90–99.

Bornstein, M., Rabbani, T., Wang, E., Bedi, A. S., & Huang, F. (2022). SWIFT: Rapid decentralized federated learning via wait-free model communication. *Advances in Neural Information Processing Systems*, *35*.

Bottou, L. (2012). Stochastic gradient descent tricks. In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade: Second edition* (pp. 421–436). Springer Berlin Heidelberg.

Bram, J., & McKay, A. (2005). The evolution of commuting patterns in the new york city metro area. *Current Issues in Economics and Finance*, *11*(10).

Brauneis, R., & Goodman, E. P. (2018). Algorithmic transparency for the smart city. *Yale JL & Tech.*, *20*, 103.

Brown, L., Marx, E., Bali, D., Amaro, E., Sur, D., Kissel, E., Monga, I., Katz-Bassett, E., Krishnamurthy, A., McCauley, J., Narechania, T. N., Panda, A., & Shenker, S. (2024). An architecture for edge networking services. *Proceedings of the ACM SIGCOMM 2024 Conference*.

Cao, P., Li, D., & Ma, K. (2022). Image quality assessment: Integrating model-centric and data-centric approaches. *arXiv preprint arXiv:2207.14769*.

Caruana, R. (1997). Multitask learning. *Machine learning*, *28*(1), 41–75.

Chai, C., Wang, J., Luo, Y., Niu, Z., & Li, G. (2022). Data management for machine learning: A survey. *IEEE Transactions on Knowledge and Data Engineering*, *35*(5), 4646–4667.

Chatfield, C. (1995). Model uncertainty, data mining and statistical inference. *Journal of the Royal Statistical Society. Series A (Statistics in Society)*, *158*(3), 419–466. http://www.jstor.org/stable/2983440

Chen, A. N. K., Goes, P. B., & Marsden, J. R. (2002). A query-driven approach to the design and management of flexible database systems. *Journal of Management Information Systems*, *19*(3), 121–154. Retrieved June 27, 2023, from http://www.jstor.org/stable/40398596

Chen, D., Yao, L., Gao, D., Ding, B., & Li, Y. (2023, 23–29 Jul). Efficient personalized federated learning via sparse model-adaptation. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, & J. Scarlett (Eds.), *Proceedings of the 40th international conference on machine learning* (pp. 5234–5256, Vol. 202). PMLR. https://proceedings.mlr.press/v202/chen23aj.html

Chen, J., Xue, J., Wang, Y., Liu, Z., & Huang, L. (2024). Classifier clustering and feature alignment for federated learning under distributed concept drift. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, & C. Zhang (Eds.), *Advances in neural information processing systems* (pp. 81360–81388, Vol. 37). Curran Associates, Inc. https://doi.org/10.52202/079017-2586

Chen, M., Yang, Z., Saad, W., Yin, C., Poor, H. V., & Cui, S. (2020). A joint learning and communications framework for federated learning over wireless networks. *IEEE transactions on wireless communications*, *20*(1), 269–283.

Choi, M. (2018). *Insurance*. Kaggle. https://www.kaggle.com/mirichoi0218/insurance

Cocarascu, O., Stylianou, A., Čyras, K., & Toni, F. (2020). Data-empowered argumentation for dialectically explainable predictions. In *Ecai 2020* (pp. 2449–2456). IOS Press.

Cokelaer, T., Kravchenko, A., lahdjirayhan, msat59, Varma, A., L, B., Stringari, C. E., Brueffer, C., Broda, E., Pruesse, E., Singaravelan, K., Li, Z., mark padgham, & negodfre. (2023, January). *Cokelaer/fitter: V1.5.2* (Version v1.5.2). Zenodo. https://doi.org/10.5281/zenodo.7497983

Collins, L., Hassani, H., Mokhtari, A., & Shakkottai, S. (2021). Exploiting shared representations for personalized federated learning. *International conference on machine learning*, 2089–2099.

Cully, B., Lefebvre, G., Meyer, D., Feeley, M., Hutchinson, N., & Warfield, A. (2008). Remus: High availability via asynchronous virtual machine replication. *Proceedings of the 5th USENIX symposium on networked systems design and implementation*, 161–174.

Dai, R., Shen, L., He, F., Tian, X., & Tao, D. (2022). Dispfl: Towards communication-efficient personalized federated learning via decentralized sparse training. *International Conference on Machine Learning*, 4587–4604.

Daumé III, H. (2007, June). Frustratingly easy domain adaptation. In A. Zaenen & A. van den Bosch (Eds.), *Proceedings of the 45th annual meeting of the association of computational linguistics* (pp. 256–263). Association for Computational Linguistics. https://aclanthology.org/P07-1033/

Delic, K. A. (2016). On resilience of iot systems: The internet of things (ubiquity symposium). *Ubiquity*, *2016*(February), 1–7.

Deng, S., Zhao, H., Fang, W., Yin, J., Dustdar, S., & Zomaya, A. Y. (2020a). Edge intelligence: The confluence of edge computing and artificial intelligence. *IEEE Internet of Things Journal*, *7*(8), 7457–7469. https://doi.org/10.1109/JIOT.2020.2984887

Deng, S., Zhao, H., Fang, W., Yin, J., Dustdar, S., & Zomaya, A. Y. (2020b). Edge intelligence: The confluence of edge computing and artificial intelligence. *IEEE Internet of Things Journal*, *7*(8), 7457–7469. https://doi.org/10.1109/JIOT.2020.2984887

Deng, Y., Kamani, M. M., & Mahdavi, M. (2020c). Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*.

Diamond, S., & Boyd, S. (2016). CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, *17*(83), 1–5.

Diao, E., Wang, G., Zhang, J., Yang, Y., Ding, J., & Tarokh, V. (2023). Pruning deep neural networks from a sparsity perspective. *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=i-DleYh34BM

Domingos, P., & Hulten, G. (2000). Mining high-speed data streams. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 71–80.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., & Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. *International conference on machine learning*, 647–655.

El Mokadem, R., Ben Maissa, Y., & El Akkaoui, Z. (2023). Federated learning for energy constrained devices: A systematic mapping study. *Cluster Computing*, *26*(2), 1685–1708.

Evci, U., Gale, T., Menick, J., Castro, P. S., & Elsen, E. (2020). Rigging the lottery: Making all tickets winners. *International Conference on Machine Learning*, 2943–2952.

Fails, J. A., & Olsen, D. R. (2003). Interactive machine learning. *Proceedings of the 8th International Conference on Intelligent User Interfaces*, 39–45. https://doi.org/10.1145/604045.604056

Fallah, A., Mokhtari, A., & Ozdaglar, A. (2020a). Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in neural information processing systems*, *33*, 3557–3568.

Fallah, A., Mokhtari, A., & Ozdaglar, A. (2020b). Personalized federated learning: A meta-learning approach. https://arxiv.org/abs/2002.07948

Fang, Z., He, Y., & Procter, R. (2021). A query-driven topic model. *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. https://doi.org/10.18653/v1/2021.findings-acl.154

Farahani, A., Voghoei, S., Rasheed, K., & Arabnia, H. R. (2021). A brief review of domain adaptation. *Advances in Data Science and Information Engineering*, 877–894.

Feng, J., Li, Y., Zhang, C., Sun, F., Meng, F., Guo, A., & Jin, D. (2018). Deepmove: Predicting human mobility with attentional recurrent networks. *Proceedings of the 2018 World Wide Web Conference*, 1459–1468.

Frankle, J., & Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *International Conference on Learning Representations*. https://openreview.net/forum?id=rJl-b3RcF7

Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection. *Brazilian symposium on artificial intelligence*, 286–295.

Gama, J., Rocha, R., & Medas, P. (2003). Accurate decision trees for mining high-speed data streams. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 523–528.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014a). A survey on concept drift adaptation. *ACM Computing Surveys*, *46*(4), 44:1–44:37. https://doi.org/10.1145/2523813

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014b). A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, *46*(4), 1–37.

Ge, X., Chen, F., Jose, J. M., Ji, Z., Wu, Z., & Liu, X. (2021). Structured multi-modal feature embedding and alignment for image-sentence retrieval. *Proceedings of the 29th ACM international conference on multimedia*, 5185–5193.

Ghorbani, A., & Zou, J. Y. (2019). Data shapley: Equitable valuation of data for machine learning. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning, ICML 2019, 9-15 june 2019, long beach, california, USA* (pp. 2242–2251, Vol. 97). PMLR.

Ghosh, A., Chung, J., Yin, D., & Ramchandran, K. (2022). An efficient framework for clustered federated learning. *IEEE Transactions on Information Theory*, *68*(12), 8076–8091. https://doi.org/10.1109/TIT.2022.3192506

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 580–587.

Gomes, H. M., Barddal, J. P., Enembreck, F., & Bifet, A. (2017). A survey on ensemble learning for data stream classification. *ACM Computing Surveys (CSUR)*, *50*(2), 1–36.

Gu, X., Wu, Q., Fan, Q., et al. (2024). Mobility-aware federated self-supervised learning in vehicular network. *Urban Lifeline*, *2*, 10. https://doi.org/10.1007/s44285-024-00020-5

Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, *28*.

Harchol, Y., Mushtaq, A., Fang, V., McCauley, J., Panda, A., & Shenker, S. (2020). Making edge-computing resilient. *Proceedings of the 11th ACM Symposium on Cloud Computing*, 253–266. https://doi.org/10.1145/3419111.3421278

Harth, N., & Anagnostopoulos, C. (2018a). Edge-centric efficient regression analytics. *2018 IEEE International Conference on Edge Computing (EDGE)*, 93–100. https://doi.org/10.1109/EDGE.2018.00020

Harth, N., & Anagnostopoulos, C. (2018b). Edge-centric efficient regression analytics. *2018 IEEE EDGE*, 93–100.

He, C., Annavaram, M., & Avestimehr, S. (2020). Group knowledge transfer: Federated learning of large CNNs at the edge. *Advances in Neural Information Processing Systems*, *33*, 14068–14080.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hegedűs, I., Danner, G., & Jelasity, M. (2019). Gossip learning as a decentralized alternative to federated learning. *IFIP International Conference on Distributed Applications and Interoperable Systems*, 74–90. https://doi.org/10.1007/978-3-030-22496-7_5

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., & Peste, A. (2021). Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, *22*(241), 1–124.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Huang, T., Liu, S., Shen, L., He, F., Lin, W., & Tao, D. (2022). Achieving personalized federated learning with sparse local models. *arXiv preprint arXiv:2201.11380*.

Huang, T., Shen, L., Sun, Y., Lin, W., & Tao, D. (2023). Fusion of global and local knowledge for personalized federated learning. *Transactions on Machine Learning Research*. https://openreview.net/forum?id=QtrjqVIZna

Huang, Z., Cautis, B., Cheng, R., Zheng, Y., Mamoulis, N., & Yan, J. (2018). Entity-based query recommendation for long-tail queries. *ACM Trans. Knowl. Discov. Data*, *12*(6). https://doi.org/10.1145/3233186

Hulten, G., Spencer, L., & Domingos, P. (2001). Mining time-changing data streams. *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 97–106.

Isik, B., Pase, F., Gunduz, D., Weissman, T., & Michele, Z. (2023). Sparse random networks for communication-efficient federated learning. *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=k1FHgri5y3-

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2704–2713.

Jan, M. A., Zakarya, M., Khan, M., Mastorakis, S., Menon, V. G., Balasubramanian, V., & Rehman, A. U. (2021). An ai-enabled lightweight data fusion and load optimization approach for internet of things. *Future Generation Computer Systems*, *122*, 40–51.

Jarrahi, M. H., Memariani, A., & Guha, S. (2022). The principles of data-centric ai (dcai). *arXiv preprint arXiv:2211.14611*.

Jastrzebski, S., Arpit, D., Astrand, O., Kerg, G. B., Wang, H., Xiong, C., Socher, R., Cho, K., & Geras, K. J. (2021). Catastrophic fisher explosion: Early phase fisher matrix impacts generalization. *International Conference on Machine Learning*, 4772–4784.

Jiang, Y., Zhu, Y., Lan, C., Yi, B., Cui, Y., & Guo, C. (2020). A unified architecture for accelerating distributed DNN training in heterogeneous GPU/CPU clusters. *14th*

USENIX Symposium on Operating Systems Design and Implementation (OSDI 20), 463–479.

Jiang, Y., Wang, S., Valls, V., Ko, B. J., Lee, W.-H., Leung, K. K., & Tassiulas, L. (2023). Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems*, *34*(12), 10374–10386. https://doi.org/10.1109/TNNLS.2022.3166101

Jothimurugesan, E., Hsieh, K., Wang, J., Joshi, G., & Gibbons, P. B. (2023). Federated learning under distributed concept drift. *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, *206*, 5834–5853.

Justus, D., Brennan, J., Bonner, S., & McGough, A. S. (2018). Predicting the computational cost of deep learning models. *2018 IEEE International Conference on Big Data (Big Data)*, 3873–3882. https://doi.org/10.1109/BigData.2018.8622396

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. (2021). Advances and open problems in federated learning. *Foundations and trends® in machine learning*, *14*(1–2), 1–210.

Kanhere, O., Poddar, H., Xing, Y., Shakya, D., Ju, S., & Rappaport, T. S. (2022). A power efficiency metric for comparing energy consumption in future wireless networks in the millimeter-wave and terahertz bands. *IEEE Wireless Communications*, *29*(6), 56–63.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S. J., Stich, S. U., & Suresh, A. T. (2021). Scaffold: Stochastic controlled averaging for federated learning. https://arxiv.org/abs/1910.06378

Khan, M. M. H., Le, H. K., LeMay, M., Moinzadeh, P., Wang, L., Yang, Y., Noh, D. K., Abdelzaher, T., Gunter, C. A., Han, J., et al. (2010). Diagnostic powertracing for sensor node failure analysis. *Proceedings of the 9th ACM/IEEE international conference on information processing in sensor networks*, 117–128.

Kim, H.-C., Pang, S., Je, H.-M., Kim, D., & Bang, S.-Y. (2002). Support vector machine ensemble with bagging. In S.-W. Lee & A. Verri (Eds.), *Pattern recognition with support vector machines* (pp. 397–408). Springer Berlin Heidelberg.

Konečný, J., McMahan, B., & Ramage, D. (2015). Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*.

Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., & Bacon, D. (2017). Federated learning: Strategies for improving communication efficiency. https://arxiv.org/abs/1610.05492

Kraska, T., Talwalkar, A., Duchi, J. C., Griffith, R., Franklin, M. J., & Jordan, M. I. (2013). Mlbase: A distributed machine-learning system. *CIDR*, *1*, 1–2.

Krishnamoorthi, R. (2018). Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*.

Krizhevsky, A., & Hinton, G. (2009). *Learning multiple layers of features from tiny images* (Technical Report). University of Toronto. Toronto, Ontario. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*.

Kurmanji, M., & Triantafillou, P. (2023). Detect, distill and update: Learned db systems facing out of distribution data. *Proc. ACM Manag. Data*, *1*(1). https://doi.org/10.1145/3588713

Lai, F., Zhu, X., Madhyastha, H. V., & Chowdhury, M. (2021). Oort: Efficient federated learning via guided participant selection. *15th OSDI*, 19–35.

Lee, N., Ajanthan, T., & Torr, P. (2019). SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY. *International Conference on Learning Representations*. https://openreview.net/forum?id=B1VZqjAcYX

Li, A., Sun, J., Zeng, X., Zhang, M., Li, H., & Chen, Y. (2021a). Fedmask: Joint computation and communication-efficient personalized federated learning via heterogeneous masking. *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, 42–55.

Li, C., Zeng, X., Zhang, M., & Cao, Z. (2022). PyramidFL: A fine-grained client selection framework for efficient federated learning. *Proceedings of the 28th Annual International Conference on Mobile Computing and Networking*, 158–171. https://doi.org/10.1145/3495243.3517017

Li, L., & Liu, J. (2012). An efficient and flexible web services-based multidisciplinary design optimisation framework for complex engineering systems. *Enterprise Information Systems*, *6*(3), 345–371.

Li, M., Andersen, D. G., Park, J. W., Smola, A. J., Ahmed, A., Josifovski, V., Long, J., Shekita, E. J., & Su, B.-Y. (2014). Scaling distributed machine learning with the parameter server. *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, 583–598.

Li, T., Hu, S., Beirami, A., & Smith, V. (2021b). Ditto: Fair and robust federated learning through personalization. *International Conference on Machine Learning*, 6357–6368.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. (2020). Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, *2*, 429–450.

Li, T., Sanjabi, M., Beirami, A., & Smith, V. (2019). Fair resource allocation in federated learning. *arXiv preprint arXiv:1905.10497*.

Li, X., Jiang, M., Zhang, X., Kamp, M., & Dou, Q. (2021c). Fedbn: Federated learning on non-iid features via local batch normalization. https://arxiv.org/abs/2102.07623

Liang, Y., & Zhao, Z. (2022). Nettraj: A network-based vehicle trajectory prediction model with directional representation and spatiotemporal attention mechanisms. *IEEE TITS*, *23*(9), 14470–14481. https://doi.org/10.1109/TITS.2021.3129588

Lim, W. Y. B., Luong, N. C., Hoang, D. T., Jiao, Y., Liang, Y.-C., Yang, Q., Niyato, D., & Miao, C. (2020). Federated learning in mobile edge networks: A comprehensive survey. *IEEE Comm. Srv & Tutor*, *22*(3), 2031–2063.

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.-M., Wang, W.-C., Xiao, G., Dang, X., Gan, C., & Han, S. (2024). Awq: Activation-aware weight quantization for on-device llm compression and acceleration. In P. Gibbons, G. Pekhimenko, & C. D. Sa (Eds.), *Proceedings of machine learning and systems* (pp. 87–100, Vol. 6).

Lin, Y., Han, S., Mao, H., Wang, Y., & Dally, B. (2018). Deep gradient compression: Reducing the communication bandwidth for distributed training. *International Conference on Learning Representations*. https://openreview.net/forum?id=SkhQHMW0W

Liu, Y., Shi, Y., Li, Q., Wu, B., Wang, X., & Shen, L. (2024). Decentralized directed collaboration for personalized federated learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 23168–23178.

Liu, Y., Pei, A., Wang, F., Yang, Y., Zhang, X., Wang, H., Dai, H., Qi, L., & Ma, R. (2021). An attention-based category-aware gru model for the next poi recommendation. *J. Intelligent Systems*, *36*(7), 3174–3189.

Liu, Z., Sun, M., Zhou, T., Huang, G., & Darrell, T. (2019). Rethinking the value of network pruning. https://arxiv.org/abs/1810.05270

Long, Q., Anagnostopoulos, C., Parambath, S. P., & Bi, D. (2023). Feddip: Federated learning with extreme dynamic pruning and incremental regularization. *2023 IEEE International Conference on Data Mining (ICDM)*, 1187–1192.

Long, Q., Wang, Q., Anagnostopoulos, C., & Bi, D. (2025). Decentralized personalized federated learning based on a conditional "sparse-to-sparser" scheme. *IEEE Transactions on Neural Networks and Learning Systems*, 1–15. https://doi.org/10.1109/TNNLS.2025.3580277

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2018). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, *31*(12), 2346–2363.

Ma, M., Li, T., & Peng, X. (2024). Beyond the federation: Topology-aware federated learning for generalization to unseen clients. *Proceedings of the 41st International Conference on Machine Learning*, *235*, 33794–33810.

Ma, Q., & Triantafillou, P. (2022). Query-centric regression. *Information Systems*, *104*, 101736. https://doi.org/https://doi.org/10.1016/j.is.2021.101736

Macedo, D., Santos, D., Perkusich, A., & Valadares, D. (2023). A mobility-aware federated learning coordination algorithm. *Journal of Supercomputing*, *79*, 19049–19063. https://doi.org/10.1007/s11227-023-05372-3

Masuyama, N., Amako, N., Yamada, Y., Nojima, Y., & Ishibuchi, H. (2022). Adaptive resonance theory-based topological clustering with a divisive hierarchical structure capable of continual learning. *IEEE Access*, *10*, 68042–68056. https://doi.org/10.1109/ACCESS.2022.3186479

McLachlan, G. J., & Peel, D. (2000). *Finite mixture models*. John Wiley & Sons.

McMahan, B., Moore, E., Ramage, D., Hampson, S., & Arcas, B. A. y. (2017, April). Communication-Efficient Learning of Deep Networks from Decentralized Data. In A. Singh & J. Zhu (Eds.), *Proceedings of the 20th international conference on artificial intelligence and statistics* (pp. 1273–1282, Vol. 54). PMLR. https://proceedings.mlr.press/v54/mcmahan17a.html

Messaoud, K., Yahiaoui, I., Verroust-Blondet, A., & Nashashibi, F. (2021). Attention based vehicle trajectory prediction. *IEEE Transactions on Intelligent Vehicles*, *6*(1), 175–185. https://doi.org/10.1109/TIV.2020.2991952

Meyer, M., Gosink, L. J., Anderson, J. C., Bethel, E. W., & Joy, K. I. (2008). Query-driven visualization of time-varying adaptive mesh refinement data. *IEEE Transactions on Visualization and Computer Graphics*, *14*(6), 1715–1722. https://doi.org/10.1109/TVCG.2008.157

Minku, L. L., White, A. P., & Yao, X. (2009). The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on knowledge and Data Engineering*, *22*(5), 730–742.

Miranda, L. J. (2021). Towards data-centric machine learning: A short review. *ljvmiranda921.github.io*. https://ljvmiranda921.github.io/notebook/2021/07/30/data-centric-ml/

Mudassar, M., Zhai, Y., & Lejian, L. (2022). Adaptive fault-tolerant strategy for latency-aware iot application executing in edge computing environment. *IEEE Internet of Things Journal*, *9*(15), 13250–13262. https://doi.org/10.1109/JIOT.2022.3144026

Nagalapatti, L., Mittal, R. S., & Narayanam, R. (2022). Is your data relevant?: Dynamic selection of relevant data for federated learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *36*(7), 7859–7867.

Ning, Z., Tian, C., Xiao, M., Fan, W., Wang, P., Li, L., Wang, P., & Zhou, Y. (2024). Fedgcs: A generative framework for efficient client selection in federated learning via gradient-based optimization. In *Ijcai* (pp. 4760–4768). https://www.ijcai.org/proceedings/2024/526

Nishio, T., & Yonetani, R. (2019). Client selection for federated learning with heterogeneous resources in mobile edge. *IEEE ICC*, 1–7.

Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, *22*(10), 1345–1359.

Panchal, K., Choudhary, S., Mitra, S., Mukherjee, K., Sarkhel, S., Mitra, S., & Guan, H. (2023). Flash: Concept drift adaptation in federated learning. *Proceedings of the 40th International Conference on Machine Learning*, *202*, 26931–26962.

Qi, S., Ramakrishnan, K., & Lee, M. (2024). Lifl: A lightweight, event-driven serverless platform for federated learning. *Proceedings of Machine Learning and Systems*, *6*, 408–425.

Reddi, S., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., & McMahan, H. B. (2021). Adaptive federated optimization. https://arxiv.org/abs/2003.00295

Ren, J., Pan, Y., Goscinski, A., & Beyah, R. A. (2018). Edge computing for the internet of things. *IEEE Network*, *32*(1), 6–7. https://doi.org/10.1109/MNET.2018.8270624

Rothchild, D., Panda, A., Ullah, E., Ivkin, N., Stoica, I., Braverman, V., Gonzalez, J., & Arora, R. (2020). FetchSGD: Communication-efficient federated learning with sketching. *Proceedings of the 37th International Conference on Machine Learning*, *119*, 8253–8265.

Rübel, O., Bethel, E., Prabhat, M., & Wu, K. (2012). *Query-driven visualization and analysis* (tech. rep.). Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States).

Samanta, A., Esposito, F., & Nguyen, T. G. (2021). Fault-tolerant mechanism for edge-based iot networks with demand uncertainty. *IEEE Internet of Things Journal*, *8*(23), 16963–16971. https://doi.org/10.1109/JIOT.2021.3075681

Sambasivan, N., Kapania, S., Highfill, H., Akrong, D., Paritosh, P., & Aroyo, L. M. (2021). Everyone wants to do the model work, not the data work: Data cascades in high-stakes ai. *proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 1–15.

Sampaio, G. S., de Aguiar Vallim Filho, A. R., da Silva, L. S., & da Silva, L. A. (2019). Prediction of motor failure time using an artificial neural network. *Sensors*, *19*(19), 4342. https://doi.org/10.3390/s19194342

Satyanarayanan, M. (2017). The emergence of edge computing. *Computer*, *50*(1), 30–39. https://doi.org/10.1109/MC.2017.9

Satyanarayanan, M., Bahl, P., Cáceres, R., & Davies, N. (2009). The case for VM-based cloudlets in mobile computing. *IEEE Pervasive Computing*, *8*(4), 14–23. https://doi.org/10.1109/MPRV.2009.82

Savva, F., Anagnostopoulos, C., & Triantafillou, P. (2020a). Surf: Identification of interesting data regions with surrogate models. *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, 1321–1332. https://doi.org/10.1109/ICDE48307.2020.00118

Savva, F., Anagnostopoulos, C., Triantafillou, P., & Kolomvatsos, K. (2020b). Large-scale data exploration using explanatory regression functions. *ACM Trans. Knowl. Discov. Data*, *14*(6). https://doi.org/10.1145/3410448

Seide, F., Fu, H., Droppo, J., Li, G., & Yu, D. (2014). 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. *Interspeech, 2014*, 1058–1062.

Settles, B. (2009). *Active learning literature survey* (Computer Sciences Technical Report No. 1648). University of Wisconsin-Madison.

Shafahi, A., Najibi, M., Xu, Z., Dickerson, J., Davis, L. S., & Goldstein, T. (2020). Universal adversarial training. *AAAI Conference on Artificial Intelligence*, *34*(04), 5636–5643.

Shao, S., Huang, X., Stanley, H. E., & Havlin, S. (2015). Percolation of localized attack on complex networks. *New Journal of Physics*, *17*(2), 023049. https://doi.org/10.1088/1367-2630/17/2/023049

Sharir, O., Peleg, B., & Shoham, Y. (2020). The cost of training NLP models: A concise overview. *arXiv preprint arXiv:2004.08900*.

Shen, T., Qi, J., Jiang, J., Wang, X., Wen, S., Chen, X., Zhao, S., Wang, S., Chen, L., Luo, X., Zhang, F., & Cui, H. (2022). SOTER: Guarding black-box inference for general neural networks at the edge. *2022 USENIX Annual Technical Conference (USENIX ATC 22)*, 723–738.

Sherry, J., Gao, P. X., Basu, S., Panda, A., Krishnamurthy, A., Maciocco, C., Manesh, M., Martins, J., Ratnasamy, S., Rizzo, L., et al. (2015). Rollback-recovery for middleboxes. *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 227–240.

Shi, L., Wang, L., Long, C., Zhou, S., Zhou, M., Niu, Z., & Hua, G. (2021). Sgcn: Sparse graph convolution network for pedestrian trajectory prediction. *IEEE/CVF CVPR*, 8994–9003.

Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, *3*(5), 637–646. https://doi.org/10.1109/JIOT.2016.2579198

Shi, Y., Shen, L., Wei, K., Sun, Y., Yuan, B., Wang, X., & Tao, D. (2023, July). Improving the model consistency of decentralized federated learning. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, & J. Scarlett (Eds.), *Proceedings of the 40th international conference on machine learning* (pp. 31269–31291, Vol. 202). PMLR. https://proceedings.mlr.press/v202/shi23d.html

Siavvas, M., & Gelenbe, E. (2019). Optimum interval for application-level checkpoints. *2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/ 2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, 145–150. https://doi.org/10.1109/CSCloud/EdgeCom.2019.000-4

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations (ICLR 2015)*.

Smith, V., Chiang, C.-K., Sanjabi, M., & Talwalkar, A. S. (2017). Federated multi-task learning. *Advances in neural information processing systems*, *30*.

Song, C., Qu, Z., Blumm, N., & Barabási, A.-L. (2010). Limits of predictability in human mobility. *Science*, *327*(5968), 1018–1021.

Song, X., Chen, K., Li, X., Sun, J., Hou, B., Cui, Y., Zhang, B., Xiong, G., & Wang, Z. (2021). Pedestrian trajectory prediction based on deep convolutional lstm network. *IEEE TITS*, *22*(6), 3285–3302.

Sterbenz, J. P., Hutchison, D., Çetinkaya, E. K., Jabbar, A., Rohrer, J. P., Schöller, M., & Smith, P. (2010). Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines [Resilient and Survivable networks]. *Computer Networks*, *54*(8), 1245–1265. https://doi.org/https://doi.org/10.1016/j.comnet.2010.03.005

Subramonyam, H., Seifert, C., & Adar, M. E. (2021). How can human-centered design shape data-centric ai. *Proceedings of NeurIPS Data-Centric AI Workshop*, 3.

Sun, J., Chen, T., Giannakis, G., & Yang, Z. (2019). Communication-efficient distributed learning via lazily aggregated quantized gradients. *Advances in Neural Information Processing Systems*, *32*.

Sun, T., Li, D., & Wang, B. (2022). Decentralized federated averaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(4), 4289–4301.

Symeonides, M., Trihinas, D., Georgiou, Z., Pallis, G., & Dikaiakos, M. (2019). Query-driven descriptive analytics for iot and edge computing. *2019 IEEE International Conference on Cloud Engineering (IC2E)*, 1–11. https://doi.org/10.1109/IC2E.2019.00-12

Takao, D., Sugiura, K., & Ishikawa, Y. (2021). Approximate fault-tolerant data stream aggregation for edge computing. *Big-Data-Analytics in Astronomy, Science, and Engineering: 9th International Conference on Big Data Analytics, BDA 2021, Virtual Event, December 7–9, 2021, Proceedings*, 233–244. https://doi.org/10.1007/978-3-030-96600-3_17

Tang, M., Ning, X., Wang, Y., Sun, J., Wang, Y., Li, H., & Chen, Y. (2022a). Fedcor: Correlation-based active client selection strategy for heterogeneous federated learning. https://arxiv.org/abs/2103.13822

Tang, Z., Shi, S., Li, B., & Chu, X. (2022b). Gossipfl: A decentralized federated learning framework with sparsified and adaptive communication. *IEEE Transactions on Parallel and Distributed Systems*, *34*(3), 909–922.

Teso, S., & Kersting, K. (2019). Explanatory interactive machine learning. *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, 239–245.

Toneva, M., Sordoni, A., des Combes, R. T., Trischler, A., Bengio, Y., & Gordon, G. J. (2019). An empirical study of example forgetting during deep neural network learning. *International Conference on Learning Representations*. https://openreview.net/forum?id=BJlxm30cKm

Trust, E. H. (2024). Reports on the increasing energy consumption of wireless systems and digital ecosystem [Accessed: 2024-05-18]. https://ehtrust.org/science/reports-on-power-consumption-and-increasing-energy-use-of-wireless-systems-and-digital-ecosystem/

Tschandl, P., Rosendahl, C., & Kittler, H. (2018). The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific data*, *5*(1), 1–9.

Tüfekci, P. (2014). Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. *International Journal of Electrical Power & Energy Systems*, *60*, 126–140. https://doi.org/https://doi.org/10.1016/j.ijepes.2014.02.027

Tversky, A. (1977). Features of similarity. *Psychological review*, *84*(4), 327.

Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., & Rellermeyer, J. S. (2020a). A survey on distributed machine learning. *ACM Comput. Surv.*, *53*(2). https://doi.org/10.1145/3377454

Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., & Rellermeyer, J. S. (2020b). A survey on distributed machine learning. *ACM Comput. Surv.*, *53*(2). https://doi.org/10.1145/3377454

Vogels, T., Karimireddy, S. P., & Jaggi, M. (2019). PowerSGD: Practical low-rank gradient compression for distributed optimization. *Advances in Neural Information Processing Systems*, *32*.

Wang, C., Gill, C., & Lu, C. (2019a). Frame: Fault tolerant and real-time messaging for edge computing. *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 976–985.

Wang, C., Zhang, G., & Grosse, R. (2020a). Picking winning tickets before training by preserving gradient flow. *International Conference on Learning Representations*. https://openreview.net/forum?id=SkgsACVKPH

Wang, D., Shen, L., Luo, Y., Hu, H., Su, K., Wen, Y., & Tao, D. (2023). Fedabc: Targeting fair competition in personalized federated learning. *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*. https://doi.org/10.1609/aaai.v37i8.26203

Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., & Khazaeni, Y. (2020b). Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*.

Wang, J., Liu, Q., Liang, H., Joshi, G., & Poor, H. V. (2020c). Tackling the objective inconsistency problem in heterogeneous federated optimization. https://arxiv.org/abs/2007.07481

Wang, J., Pambudi, S., Wang, W., & Song, M. (2019b). Resilience of iot systems against edge-induced cascade-of-failures: A networking perspective. *IEEE Internet of Things Journal*, *6*(4), 6952–6963. https://doi.org/10.1109/JIOT.2019.2913140

Wang, K., Lu, J., Liu, A., Song, Y., Xiong, L., & Zhang, G. (2022a). Elastic gradient boosting decision tree with adaptive iterations for concept drift adaptation. *Neurocomputing*, *491*, 288–304. https://doi.org/https://doi.org/10.1016/j.neucom.2022.03.038

Wang, Q., Fornes, J. M., Anagnostopoulos, C., & Kolomvatsos, K. (2022b). Predictive model resilience in edge computing. *2022 IEEE 8th World Forum on Internet of Things (WF-IoT)*, 1–6. https://doi.org/10.1109/WF-IoT54382.2022.10152282

Wang, S., Tuor, T., Salonidis, T., Leung, K. K., Makaya, C., He, T., & Chan, K. (2019c). Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications*, *37*(6), 1205–1221. https://doi.org/10.1109/JSAC.2019.2904348

Webb, G. I., Hyde, R., Cao, H., Nguyen, H. L., & Petitjean, F. (2016). Characterizing concept drift. *Data Mining and Knowledge Discovery*, *30*(4), 964–994.

Welbourne, E., Battle, L., Cole, G., Gould, K., Rector, K., Raymer, S., Balazinska, M., & Borriello, G. (2009). Building the internet of things using rfid: The rfid ecosystem experience. *IEEE Internet computing*, *13*(3), 48–55.

Wong, E., Rice, L., & Kolter, J. Z. (2020). Fast is better than free: Revisiting adversarial training. *CoRR*, *abs/2001.03994*. https://arxiv.org/abs/2001.03994

Wu, C., Wu, F., Lyu, L., Huang, Y., & Xie, X. (2022). Communication-efficient federated learning via knowledge distillation. *Nature Communications*, *13*, 2032. https://doi.org/10.1038/s41467-022-29763-x

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., & Han, S. (2023). SmoothQuant: Accurate and efficient post-training quantization for large language models. *Proceedings of the 40th International Conference on Machine Learning*, *202*, 38087–38099.

Xie, C., Koyejo, S., & Gupta, I. (2019). Asynchronous federated optimization. *arXiv:1903.03934*.

Xu, J., & Wang, H. (2021). Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective. *IEEE TWC*, *20*(2), 1188–1200.

Xue, H., Huynh, D. Q., & Reynolds, M. (2018). Ss-lstm: A hierarchical lstm model for pedestrian trajectory prediction. *IEEE WACV*, 1186–1194. https://doi.org/10.1109/WACV.2018.00135

Yabe, T., Tsubouchi, K., Shimizu, T., Sekimoto, Y., Sezaki, K., Moro, E., & Pentland, A. (2024). Yjmob100k: City-scale and longitudinal dataset of anonymized human mobility trajectories. *Scientific Data*, *11*(1), 397.

Yang, H., & Fong, S. (2012). Incrementally optimized decision tree for noisy big data. *Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, 36–44. https://doi.org/10.1145/2351316.2351322

Yang, H., & Fong, S. (2015). Countering the concept-drift problems in big data by an incrementally optimized stream mining model. *Journal of Systems and Software*, *102*, 158–166. https://doi.org/https://doi.org/10.1016/j.jss.2014.07.010

Yang, L., Huang, J., Lin, W., & Cao, J. (2023). Personalized federated learning on non-iid data via group-based meta-learning. *ACM Trans. Knowl. Discov. Data*, *17*(4). https://doi.org/10.1145/3558005

Yang, Z., Chen, M., Saad, W., Hong, C. S., & Shikh-Bahaei, M. (2020). Energy efficient federated learning over wireless communication networks. *IEEE Transactions on Wireless Communications*, *20*(3), 1935–1949.

Yau, C.-Y., & Wai, H. T. (2023). Docom: Compressed decentralized optimization with near-optimal sample complexity. *Transactions on Machine Learning Research*. https://openreview.net/forum?id=W0ehjkl9x7

Yin, C., Xiong, Z., Chen, H., Wang, J., Cooper, D., & David, B. (2015). A literature survey on smart cities. *Science China Information Sciences*, *58*(10), 1–18.

Yuan, L., Sun, L., Yu, P. S., & Wang, Z. (2023). Decentralized federated learning: A survey and perspective. *arXiv preprint arXiv:2306.01603*.

Zha, D., Bhat, Z. P., Lai, K.-H., Yang, F., & Hu, X. (2023). Data-centric ai: Perspectives and challenges. *arXiv preprint arXiv:2301.04819*.

Zhang, M., Sapra, K., Fidler, S., Yeung, S., & Alvarez, J. M. (2021). Personalized federated learning with first order model optimization. *International Conference on Learning Representations*. https://openreview.net/forum?id=ehJqJQk9cw

Zhang, W., Zhou, T., Lu, Q., Yuan, Y., Tolba, A., & Said, W. (2024). Fedsl: A communication-efficient federated learning with split layer aggregation. *IEEE Internet of Things Journal*, *11*(9), 15587–15601. https://doi.org/10.1109/JIOT.2024.3350241

Zhang, X., Chang, Z., Hu, T., Chen, W., Zhang, X., & Min, G. (2023). Vehicle selection and resource allocation for federated learning-assisted vehicular network. *IEEE TMC*, 1–12. https://doi.org/10.1109/TMC.2023.3283295

Zhao, H., Li, B., Li, Z., Richtárik, P., & Chi, Y. (2022). Beer: Fast $O(1/T)$ rate for decentralized nonconvex optimization with communication compression. *Advances in Neural Information Processing Systems*, *35*, 31653–31667.

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., & Chandra, V. (2018). Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*.

Zhou, P., Xu, H., Lee, L. H., Fang, P., & Hui, P. (2022). Are you left out? an efficient and fair federated learning for personalized profiles on wearable devices of inferior networking conditions. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, *6*(2). https://doi.org/10.1145/3534585

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, *109*(1), 43–76.